Spatial Intensity Estimation of Fish Distribution in Colombian River Networks Using Hydrological Distance and Point Pattern Analysis on Linear Networks

Nubia Edith Cespedes Prieto[1*], Martha Patricia Bohorquez Castañeda[2], Nicolas Felipe Romero Solano[3]

[1]Escuela de Ingenieros Militares, Ejército Nacional de Colombia, Bogotá 111611, Colombia, Email: necespedesp@unal.edu.co,
[2]Facultad de Ciencias, Departamento de Estadística, Universidad Nacional de Colombia, Bogotá, Colombia
[3]Facultad de Ciencias, Departamento de Estadística, Universidad Nacional de Colombia, Bogotá, Colombia
*Corresponding Author: Nubia Edith Cespedes Prieto, Escuela de Ingenieros Militares, Ejército Nacional de Colombia, Bogotá 111611, Colombia, Email: necespedesp@unal.edu.co

Abstract
Colombia possesses one of the most extensive and biodiverse hydrographic networks in the world, creating the need for spatial methodologies capable of accurately representing the structure and connectivity of river systems. This study presents a methodological framework for analyzing the spatial distribution of fish populations in river networks using hydrological distance rather than traditional Euclidean distance. The proposed approach incorporates flow direction, network connectivity, and watershed topology, which are essential features for properly characterizing ecological processes in freshwater environments.
The methodology was implemented in Python through the construction of a directed graph based on HydroRIVERS data for the department of Cundinamarca, Colombia. A total of 500 simulated points representing fish observations were generated and assigned to their nearest river segments through a snapping procedure. Subsequently, a hydrological distance matrix was computed among all points, and an intensity function was estimated using a Gaussian kernel adapted to linear networks. The bandwidth parameter was objectively selected through leave-one-out cross-validation to optimize the intensity estimation process.
The results enabled the identification of areas with higher concentrations and intensities of fish occurrences across the river network, demonstrating that hydrological distance provides a more realistic representation of spatial relationships in aquatic ecosystems than conventional Euclidean metrics. Furthermore, the proposed implementation offers a reproducible and scalable framework for ecological, hydrological, and conservation studies. The findings highlight that incorporating network connectivity and flow direction into spatial modeling significantly improves the characterization of fish distribution patterns and other phenomena associated with hydrographic networks.

Keywords: Hydrological distance; Point pattern analysis; River networks; Fish distribution; Intensity estimation; Gaussian kernel; Colombia.

1. Introduction

Colombia is distinguished by its abundant water resources, supported by strategic ecosystems such as moorlands, snow-capped mountains, wetlands, rivers, lakes, and estuaries. These ecosystems support high levels of biodiversity. These ecosystems fulfill essential ecological, economic, and cultural functions. They provide water resources, food, transportation, and livelihoods for many communities, as well as contributing significantly to ecosystem balance and climate regulation. In this context, it is important to recognize that Colombia has a broad and complex hydrographic network made up of large basins and bodies of water that are important for conserving biodiversity and developing the territory [6]. This network requires systematic monitoring of water quality, aquaculture behavior, and related measurements to ensure the efficient and sustainable management of hydrographic networks.
Specific patterns are especially relevant in Colombia, as numerous epidemiological, ecological, hydrological, and socioenvironmental phenomena are distributed throughout the country's extensive river network. Aquatic species, discharge points, water quality monitoring stations, pollution events, and hydraulic infrastructures are not arranged on a continuous plane. Rather, they are organized according to the geometry and connectivity of drainage networks. This demonstrates the need to incorporate these characteristics into point pattern structures.
In this framework, selecting the appropriate distance metric is fundamental to analyzing point patterns because it allows one to characterize the spatial structure of the data more precisely. Traditionally, spatial relationships have been quantified by

Euclidean distance. However, these metrics are insufficient to adequately describe the existing spatial relationships when the phenomenon of interest requires other distances, as is the case with linear networks. In these cases, the shortest path distance is one of the most commonly used distances because it explicitly incorporates the topology of the network and the length of its segments [2].

In linear networks, such as river systems, however, spatial dynamics depend not only on connectivity but also on directionality and flow structure. In this context, spatial autocorrelation is particularly relevant in freshwater river ecosystems, where nested watersheds and flow-imposed connectivity generate spatial patterns not adequately captured by either Euclidean or shortest path distances [7]. Thus, understanding spatial processes in these systems requires incorporating metrics that respect the system's physical behavior, particularly the direction of flow and the hierarchy of channels. Hydrological distance (stream distance) is a specialized measure that quantifies the separation between points along the river network by following the functional trajectory of river flow. Therefore, implementing this metric is especially relevant in applications such as hydrological modeling, water quality prediction, and analyzing spatial patterns in watersheds.

Covariance matrices constructed from Euclidean distance fail to adequately represent the spatial configuration, longitudinal connectivity, flow, and flow directionality of a river network. Additionally, many conventional autocovariance functions lack general validity when the Euclidean distance is replaced by a measure of hydrological distance. These functions also have limitations in ecological interpretation in certain contexts [12]. A generally valid autocovariance function must generate a positive, symmetric, and defined covariance matrix with nonnegative diagonal elements regardless of the configuration of
river segments or the location of the sampling points. When these conditions are not met, positive prediction variances are not guaranteed, which invalidates the results [7].

According to Peterson (2010), there are currently two main types of distance measurements that can be used to construct valid covariance matrices in geostatistical modeling of river networks when used with the appropriate autocovariance functions:

- Euclidean distance: corresponds to the distance in a straight line between two locations. All locations within the study area have the potential to be spatially correlated.
- Hydrological distance: corresponds to the distance between two locations measured exclusively along the river network. Unlike Euclidean distance, not all locations necessarily have spatial autocorrelation potential, as rules based on network connectivity and flow direction can be imposed to represent different hydrological relationships.

In practice, hydrological distance has rarely been used in empirical applications or simulations. Its computational implementation is challenging because it requires the explicit representation of the river network as a directed graph and the definition of connectivity rules and weighting schemes for the calculation of covariance structures.

For non-homogeneous point processes, using hydrological distance allows for a more accurate estimation of the intensity function. This makes it possible to identify spatial patterns and areas with specific characteristics, such as an increased recurrence of certain types of fish in the study area.

Although there are proposals for calculating this metric using the Riverdist library in the R programming language, limited empirical evidence and methodological documentation leaves open the discussion on the validity and reproducibility of such implementations. Consequently, this work opts for developing the methodology in Python (version 3.12.3) due to its advantages in computational efficiency, handling large volumes of vector data, and flexibility in representing linear network structures.

2. Methodology

2.1. Linear networks

A linear lattice consists of the joining of a finite number of line segments on the plane. The linear network is then

[1, 9

$$L = \bigcup_{i=1}^{n} l_i$$

$\mathrm{conl}_i = [u_i, v_i] = \{tu_i + (1-t)v_i : 0 \le t \le 1\}$, where are the endpoints of the segment. The total length of the network is denoted by $.u_i, v_i l_i L|L|$

It is also possible to represent as a graph embedded in the plane, composed of a finite set of vertices $Lv \in V$ and edges. Each edge corresponds to a segment of the form with $e \in Ee = \overline{vv'}v, v' \in V$, ensuring that the intersection between two different edges contains at most a single point, which, if it exists, will be a vertex. The degree of a vertex v, denoted as , represents the number of segments that have a $\deg(v)e = \overline{vv'}v$ as one of its ends. A vertex is said to be terminal if its degree is equal to 1 [5].

A path between two points and along xyL is a succession[5]

$$\pi = (x, v_1, v_2, \dots, v_p, y),$$

where $v_1, \ldots, v_P$ are vertices and each edge of is given by $Le_j = [v_j, v_{j+1}]$ for each $j = 1, \ldots, P-1$. In addition, the points x y must belong to the same edge of $v_1 e_0 L$, and in a similar way, y and must be on the same edge of $.v_P e_P L$

The length of the path is defined as:

$$\ell(\gamma) = \|v_1 - x\| + \sum_{j=1}^{P-1} \|v_{j+1} - v_j\| + \|y - v_P\|,$$

where it denotes Euclidean distance.$\|\cdot\|$

2.2 Hydrological distance

Let's suppose that a stream segment can be represented as a line and that the lines of a stream form a network. Consider that a downstream point has a lower actual number than an upstream point and that point 0 is the lowest point in the entire network. Any point in the network can be connected to a lower point via a line. The upstream distance will be the length of that line.

Let now be I the index set of the line segments that make up a network. Denote by x_i the distance upstream in the i-th segment, by the lowest point of the -th segment, and by the highest point of the -th segment (which can be $l_i i u_i i \infty$).

Thus, the index sets are defined as follows:

- U_{x_i} : Index set of all upstream segments of x_i, excluding the segment .i
- $U_{[i]}$: Index set of all upstream segments of the segment i.
- D_{x_i} : Index set of all downstream stream segments of x_i.
- $D_{[i]}$: Index set of all downstream segments of the segment i, including the i-th segment.

Therefore, two points (s_i, t_j) are flow-connected if $D_{s_i} \cap D_{t_j} = D_{s_i} o D_{t_j}$ in the same way that the concept of flow connection between a point and a segment and between segments is defined.

Now, the formula for calculating distance per flow is:

$$d(s_i, t_j) = \begin{cases} |s_i - t_j|, & \text{si } s_i \text{ y } t_j \text{ son flow-connected} \\ (s_i - u) + (t_j - u), & \text{en otro caso.} \end{cases}$$

where

$$u = \text{máx}\{u_k : k \in D_{s_i} \cap D_{t_j}\}.$$

See the following graph, in which, in addition to highlighting the difference between points connected and not connected by flow, the distinctions between Euclidean distance and hydrological distance are illustrated.

2.2.1. Calculation of hydrological distances

When two locations are connected by flow, the downstream hydrological distance from the upstream location to the downstream location is strictly greater than zero. In the opposite direction, however, the distance is zero. In contrast, when two locations are not connected by flow, the downstream hydrological distance is greater than zero in both directions.

Note that the hydrological distance from a site to itself is always zero. Finally, if two sites do not belong to the same stream network — meaning they do not share a common outlet point downstream — there is no hydrological path between them. Therefore, it is impossible to define a hydrological distance between them. To address this, an extremely large hydrological distance is assigned in both directions, equivalent to infinity in practice. This ensures that the distance exceeds the model's range parameter and prevents spatial correlation between the sites.

2.3. Intensity function

Intensity is defined such that the number of points that fall within a subset of the linear lattice $B \subset L$,, denoted as $N(X \cap B)$, has an expectation given by

$$E[N(X \cap B)] = \int_B \lambda(u) d_1 u.$$

Intuitively, $\lambda(u)$ represents the expected number of points per unit length of the network in the vicinity of the point u.

Since n is a fixed value, the estimate of density f and intensity λ are equivalent [5], so that

$$\lambda(x) = nf(x), \forall x \in L,$$

where f(x) is the probability density function from which the locations of the points are assumed to come.

The kernel estimate of the density function, in its general form, is given by

$$\hat{f}(u) = \frac{1}{n} \sum_{i=1}^{n} K(u \mid x_i).$$

Equivalently, the kernel intensity estimator is defined as

$$\hat{\lambda}(u) = \sum_{i=1}^{n} K(u \mid x_i),$$

Where

$K(\cdot \mid x_i)$ is a kernel defined on top of the linear network originating from the location $x_i, yx_1, \ldots, x_n$ represent the observed data points.

A fundamental requirement is that $K(\cdot \mid x)$ constitutes a density of probability above all L for all $x \in L$. This implies that $K(\cdot \mid x) \geq 0$ and that the kernel has a total unit mass.

2.4. Gaussian kernel

In the context of kernel intensity estimation over linear networks, the Euclidean distance $\|s - s_i\|$ is replaced by the hydrological distance $d_H(s, s_i)$, so that the intensity estimator respects the structure and connectivity of the river system. The resulting estimator is expressed as

$$\hat{\lambda}(s) = \frac{1}{h}\sum_{i=1}^{n} K\left(\frac{d_H(s, s_i)}{h}\right),$$

where h is the smoothing parameter (bandwidth) and $K(\cdot)$ corresponds to the Gaussian kernel.

Where the Gaussian kernel is given by

$$K_h(d) = \frac{1}{\sqrt{2\pi}h} \exp\left(-\frac{d^2}{2h^2}\right).$$

2.5. Cross-validation meditating bandwidth estimation

The leave-one-out scheme consists of excluding the contribution of the point x_i in the estimation of its intensity, so that

$$\hat{\lambda}_{-i}(x_i) = \sum_{\substack{j=1 \\ j \neq i}}^{n} K_h\left(d(x_i, x_j)\right).$$

Equivalently, this procedure can be expressed by imposing

$$K_h\big(d(x_i, x_i)\big) = 0,$$

which eliminates the term diagonal in the kernel array. This correction avoids the positive bias induced by the self-contribution of the point, ensuring that the estimated intensity at each location depends solely on the interaction with the other points in the process [11].

3. Python implementation

3.1. Data from hydrological networks

The hydrographic network data used in this study were downloaded from the HydroRIVERS dataset, available through the HydroSHEDS platform [3]. This product provides a vector representation of river networks on a global scale and is especially useful for including relevant hydrological attributes, such as the NEXT_DOWN variable, which describes the connectivity and direction of flow between river segments, and the ORD_CLAS variable, which indicates the order of each segment. These characteristics have made it a widely used source in hydrological, ecological and spatial analysis studies.

In addition, the data were cut to the geographical area corresponding to the department of Cundinamarca, Colombia. On this river network, 500 random points were generated in a simulated way (see figure [1], with the purpose of building a synthetic dataset that would allow evaluating the behavior of hydrological distance and its application in the analysis of point patterns in linear networks.

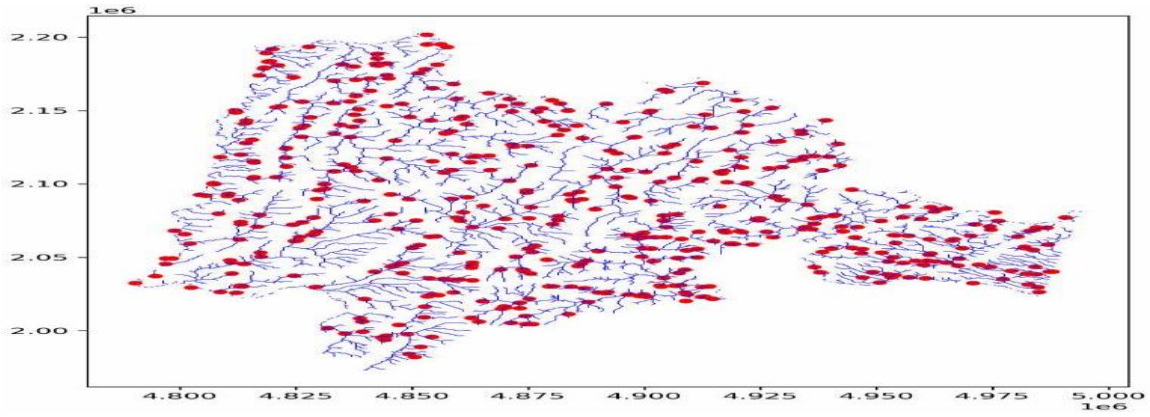

Figure 1. 500 simulated points in the department of Cundinamarca, Colombia

3.2. Creating the Directed Graph

Below is the Python code snippet that implements the graph construction:

```
G = nx. DiGraph()for _, row in rios_Cundinamarca.iterrows(): geom = row.geometry hyd_id = row["HYRIV_ID"] next_down = row["NEXT_DOWN"] G.add_node(hyd_id, geometry=geom, ord_clas=row[" ORD_CLAS"]) if next_down != 0 and not pd.isna(next_down): G.add_edge(hyd_id, next_down, weight=float( geom.length))
```

1. Create the empty graph:

```
G = nx. DiGraph()
```

A directed graph is initialized. Directionality is essential because it is modeled downstream flow.

2. Iterate over each segment of the GeoDataFrame:

```
for _, row in rios_Cundinamarca.iterrows():
```

Each row represents a river segment with geometry and attributes.

3. Extract relevant attributes:

- geom = row.geometry - the geometry (LineString or MultiLineString).
- hyd_id = row["HYRIV_ID"] - unique identifier of the segment [4].
- next_down = row["NEXT_DOWN"] - identifier of the downstream segment (0 indicates an unconnected line downstream, i.e. the last stretch of the river that flows into the ocean or an inland sinkhole)[4].

4. Add the node to the graph with attributes:

```
G.add_node(hyd_id, geometry=geom,ord_clas=row["ORD_CLAS")
```

Here each node is tagged by its hyd_id and attributes are associated with it:

- geometry.
- ord_clas: Indicator of the fluvial order according to the classic management system: order 1 represents the main channel from its source to its drain; order 2 represents all the tributaries that flow into a river of the first order; order 3 represents all the tributaries that flow into a river of second order; etc [4].

5. Add downstream directed edges:

```
if next_down != 0 and not pd.isna(next_down): G.add_edge(hyd_id, next_down,weight =float(geom.length))
```

- The condition avoids creating edges when there is no downstream segment (value 0 or NaN).
- weight=float(geom.length) assigns the geometric length of the segment as the weight. This weight will be the metric used for shorter paths.

The points simulated, observed or collected must be associated with the nearest river segment, in the same coordinates of the graph. This procedure, known as snapping, ensures that each point is topologically linked to the grid, which is essential for subsequent downstream distance calculations and intensity estimation.

```
def snap_point_to_segment(point, rios): distances = rios.geometry.distance(point) nearest_idx = distances.idxmin() return rios.loc[nearest_idx, "HYRIV_ID"] puntos_ids = [snap_point_to_segment(pt, rios_Cundinamarca) for pt in puntos.geometry]
```

1. snap_point_to_segment function: receives a point point and the GeoDataFrame of rivers.
2. Distance calculation:

```
distances = rios.geometry.distance(point)
```

Calculates the Euclidean distance between the point and each river segment geometry.

3. Identification of the nearest segment:

```
nearest_idx = distances.idxmin()
```

Gets the index of the segment whose distance to the point is minimum. This ensures that each point is associated with the closest segment on the network.

4. Return of the segment identifier:

```
return rios.loc[nearest_idx,"HYRIV_ID"]
```

It returns the HYRIV_ID of the nearest segment, which will serve as a node in subsequent analysis.

5. Application to all points:

```
puntos_ids = [snap_point_to_segment(pt,rios_Cundinamarca) for pt in puntos.geometry]
```

Each point in the GeoDataFrame points is "snapped" to the nearest segment, generating a list of segment identifiers corresponding to each point.

3.3. Creating the Distance Matrix

Once the points have been associated with the corresponding river segments and the directed graph has been constructed G, the next step is to calculate the matrix of downstream distances between all the points.

```
INF = 1e12
```

```
n = len(puntos_ids)
D = np.zeros((n, n), dtype=float)
for i, src in enumerate(puntos_ids):
    for j, tgt in enumerate(puntos_ids):
        if i == j:
            D[i, j] = 0.0
            continue
        if nx.has_path(G, src, tgt):
            D[i, j] = nx.shortest_path_length(
                G, source=src, target=tgt, weight="
                    weight"
            )
        else:
            D[i, j] = INF
D_clean = D.copy()
D_clean[D_clean >= INF] = np.nan
np.fill_diagonal(D_clean, 0.0)
```

1. Matrix initialization:

```
D = np.zeros((n, n), dtype=float)
```

A square matrix of size n by n, where n is the number of points. Initially all entries are zero.

2. Route of pairs of points:

```
for i, src in enumerate(puntos_ids):
for j, tgt in enumerate(puntos_ids):
```

Each pair of points is evaluated to calculate the distance downstream.

3. Diagonal:

```
if i == j: D[i,j] = 0.0
```

The distance of a point to itself is zero.

4. Downstream connectivity check:

```
if nx.has_path(G, src, tgt)
```

The distance is only calculated if there is a directed path from src to tgt. This respects the direction of flow in the river network.

5. Distance calculation:

```
nx.shortest_path_length(G,
source=src, target=tgt,
weight="weight")
```

The downstream distance is calculated as the minimum cumulative length of the intermediate segments.

6. Handling of unconnected pairs:

$$D[i, j] = \text{INF}$$

If no downstream path exists, a very large value (INF) is assigned to indicate that the pair is not reachable.

7. Cleaning the Matrix:

```
D_clean = D.copy()
D_clean[D_clean >= INF] = np.nan
np.fill_diagonal(D_clean, 0.0)
```

This stage converts the infinite values to NaN, preventing them from influencing the kernel's estimation. In addition, it ensures that the diagonal remains zero.

3.4. Intensity estimation using a Gaussian kernel

Now, the intensity of the point process along the river network is estimated using a Gaussian kernel.

```
def compute_lambda_hat(Dmat, h):
    """Returns K(n,n) and lambda_hat(n,)"""
    K = np.exp(-(Dmat**2) / (2*h**2))
    K = K / (np.sqrt(2*np.pi) * h)
    np.fill_diagonal(K, 0.0) #leave-one-out
    lambda_hat = np.nansum(K, axis=1)
```

```
    return K, lambda_hat
```

1. Function Inputs:

- Dmat: matrix of distance (n × n) between points.
- h: Gaussian kernel bandwidth, which controls the degree of smoothing.

2. Construction of the Gaussian kernel:

```
K = np.exp(-(Dmat**2) / (2*h**2))
K = K / (np.sqrt(2*np.pi) * h)
```

Each element K_{ij} measures the influence of the point j on the point i according to the distance downstream:

$$K_{ij} = \frac{1}{\sqrt{2\pi}h} \exp\left(-\frac{Dmat_{ij}^2}{2h^2}\right)$$

3. Leave-one-out scheme:

```
np.fill_diagonal(K, 0.0)
```

The diagonal of K is set to zero so that the point does not count with itself in the intensity estimate.

4. Calculation of the estimated intensity:

```
lambda_hat = np.nansum(K, axis=1)
```

For each point i, the influence of all other upstream and downstream points is added together. np.nansum is used to ignore NaN values, corresponding to pairs of unconnected points.

5. Output:

- K: kernel array (n × n), which contains the individual influences
- lambda_hat: estimated intensity vector (n × 1) for each point on the river network.

3.5. Optimal selection of h (bandwidth)

The bandwidth parameter controls the degree of smoothing of the Gaussian kernel. Their selection is critical: a small value produces highly variable estimates, while a large value over-smooths the intensity. To choose objectively, a leave-one-out (LOO) validation criterion is used.hh

```
def loo_score(Dmat, h, eps=1e-12):
    K, lambda_hat = compute_lambda_hat(Dmat, h)
    lambda_hat = np.maximum(lambda_hat, eps)
    return -np.sum(np.log(lambda_hat))
valid_dist = D_clean[np.isfinite(D_clean)]
median_dist = np.nanmedian(valid_dist)
h_min = max(10.0, median_dist * 0.05)
h_max = median_dist * 2.0
hs = np.linspace(h_min, h_max, 30)
scores = []
for h in hs:
    scores.append(loo_score(D_clean, h))
best_h = hs[np.argmin(scores)]
```

1. Function loo_score: Calculates the negative leave-one-out likelihood score:

$$\text{score}(h) = -\sum_{i=1}^{n} \log\left(\hat{\lambda}_{-i}(x_i)\right)$$

where

$\hat{\lambda}_{-i}$ is the estimated intensity at the point isin to count its own contribution.

2. Prevention of very small values:

```
lambda_hat = np.maximum(lambda_hat,eps)
```

Prevents the logarithm of near-zero values from producing .$-\infty$

3. Search Range h **:**

```
h_min = max(10.0, median_dist*0.05)
h_max = median_dist*2.0
```

A range is defined based on the median of valid distances, ensuring that it is not too small or too large.

4. Grid of :h

```
hs = np.linspace(h_min, h_max, 30)
```

30 values equispaced within the range are generated to evaluate the LOO score.
5. Evaluation of each h : For each bandwidth, the LOO score is calculated by:

```
scores.append(loo_score(D_clean, h))
```

6. Selection of the optimal h:

```
best_h = hs[np.argmin(scores)]
```

Bandwidth that minimizes the negative logplausibility score is considered optimal.

3.6. Intensity calculation

With the D_clean downstream distance matrix and the optimal best_h bandwidth, the final intensity of each point is calculated using the Gaussian kernel. The results are organized in a DataFrame for later analysis.

```
_, lambda_hat = compute_lambda_hat(D_clean, best_h)
resultados = pd.DataFrame({
   "segmento": puntos_ids,
   "lambda_hat": lambda_hat
})
```

1. Intensity calculation:

```
_, lambda_hat =
compute_lambda_hat(D_clean, best_h)
```

- The compute_lambda_hat function is called using the clean distance matrix and the optimal bandwidth besth.
- The variable lambda_hat contains the vector of estimated intensities $\hat{\lambda}_i$ for each point on the river network.
- The underscore '_' indicates that kernel array K is discarded, as only intensity is of interest.

2. Organization in DataFrame:

```
results = pd. DataFrame(
"segment": puntos_ids, "lambda_hat":
lambda_hat )
```

- A Panda DataFrame is created that associates each point with its estimated intensity.
- The "segment" column contains the river segment identifier for each point.
- The column "lambda_hat" contains the intensity $\hat{\lambda}_i$ estimated by downstream Gaussian kernel.

4. Results

Once the intensity $\hat{\lambda}$ at the observation points has been estimated, this information can be propagated along the segments of the river network by means of spatial smoothing based on a Gaussian kernel.

```
# Copia de rios
rios_suave = rios_Cundinamarca.copy()
rios_suave["int_suavizada"] = np.nan
# Bandwidth para suavizado
h = 5000 # metros aproximados
for i, row in rios_suave.iterrows():
   # Calcular distancias desde este segmento a todos
        los puntos
   dists = puntos.geometry.distance(row.geometry).
      values
   # Pesos tipo kernel Gaussiano
   weights = np.exp(-(dists **2) / (2*h**2))
   # Intensidades de puntos
   intensidades = puntos_plot["intensidad"].values
   # Promedio ponderado
   weighted_mean = np.sum(weights * intensidades) /
      np.sum(weights)
   rios_suave.loc[i, "int_suavizada"] =
      weighted_mean
# Visualizacion
fig, ax = plt.subplots(figsize=(10, 10))
rios_suave.plot(
   ax=ax,
   column="int_suavizada",
   cmap="viridis",
```

```
    linewidth=2,
    legend=True
)
ax.set_title("Intensidad suavizada sobre la red
    fluvial")
ax.set_axis_off()
plt.show()
```

1. Copying the River DataFrame: Creating rios_suave to keep the original data intact, and adding the int_suavizada column to store the smoothed intensity.
2. Bandwidth h Definition : The parameter h controls the Gaussian kernel extent, i.e., the range of influence of each point on the segments. In this example, $h = 5000$ meters.
3. Iteration over segments: For each river segment:

- The distance from the segment to all observation points is calculated.
- Gaussian kernel weights apply:

$$w_i = \exp\left(-\frac{d_i^2}{2h^2}\right)$$

where is the distance from the segment to the point .d_ii

- The average weighted intensity is obtained:

$$\text{int\_suavizada} = \frac{\sum_i w_i \lambda_i}{\sum_i w_i}$$

where d_i is the estimated intensity at the point i

4. Smoothing Intensity Mapping: The calculated value is stored in column int_suavizada for each segment.

5. Visualization: The river network is plotted by coloring each segment according to the softened intensity using a continuous color map (viridis). This allows us to visually observe the distribution of intensity throughout the network (Figure 2).

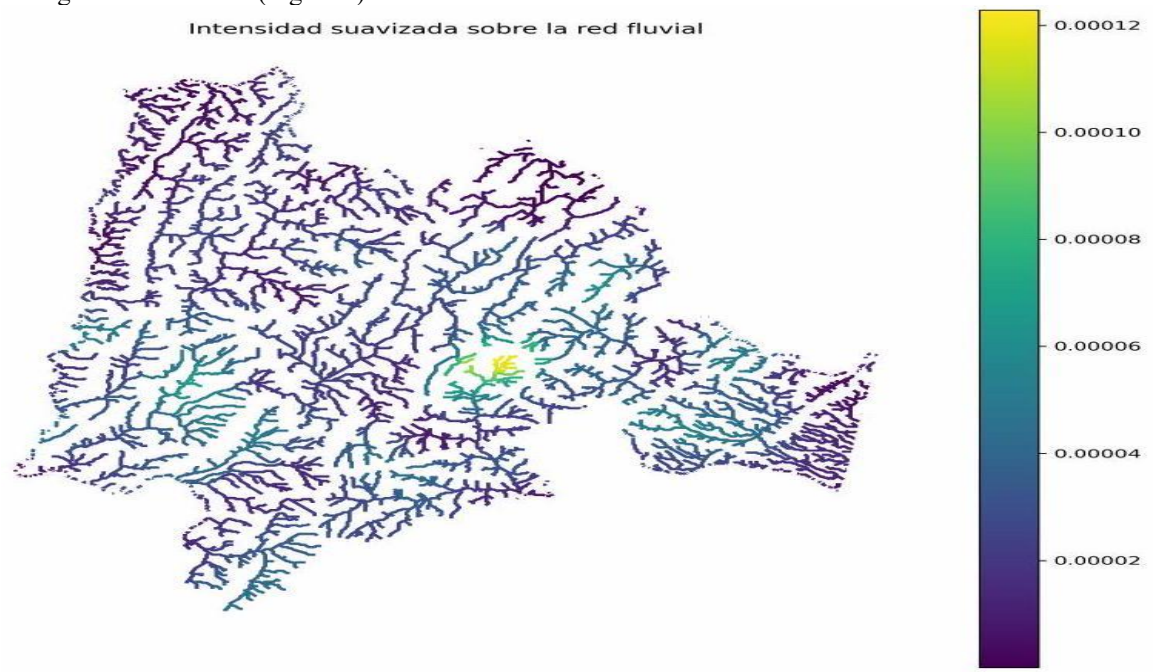

Figure 2. Estimated intensity in Cundinamarca, Colombia, for 500 simulated points

5. Conclusions

- In the case of snapping, if a point is equidistant from two segments, it will be assigned to the first one detected in processing. Modifying this assignment must be done manually if you want to prioritize another segment.
- Although, by definition, a hydrographic network cannot contain cycles, moving this distance to another field can generate ambiguity, since the distance matrix could not unequivocally determine which path to select.

- A more precise approximation would be to consider the exact position of the point on the segment and split the segment at that point to calculate the distance from this new start. However, this strategy would significantly increase the computational load.
- The choice of bandwidth h has a direct impact on the smoothness of the result: small values produce maps with greater local detail, while large values generate a more general smoothing of the intensity over the network.

References

[1] Ang, Q. W., Baddeley, A., and Nair, G. (2012). Geometrically corrected second order analysis of events on a linear network, with applications to ecology and criminology. Scandinavian Journal of Statistics, 39(4):591-617.

[2] Baddeley, A., Nair, G., Rakshit, S., and McSwiggan, G. (2017). "Stationary" point processes are uncommon on linear networks. Stat, 6:135-146.

[3] HydroSheds (2023). Hydrorivers - global river network dataset. https://www.hydrosheds.org/products/ hydrorivers. Accessed: 2025-06-20.

[4] Lehner, B. (2019). Hydrorivers technical documentation v1.0. https://data.hydrosheds.org/file/ technical-documentation/HydroRIVERS_TechDoc_ v10.pdf Accedido: 2025-06-20.

[5] McSwiggan, G., Baddeley, A., and Nair, G. (2016). Kernel density estimation on a linear network. Scandinavian Journal of Statistics.

[6] Montero, Sofia (2025). ¿cómo está formada la hidrografía colombiana? https://colombiaverde.com.co/geografia/hidrografia/como-esta-formada-la-hidrografia-colombiana/.

[7] Peterson, E. E. and Ver Hoef, J. M. (2010). A mixedmodel moving-average approach to geostatistical modeling in stream networks. Ecology, 91(3):644-651.

[8] Pérez, M. (2024). Ríos en colombia: Importancia para las comunidades. https://www.radionacional.co/actualidad/medio-ambiente/rios-en-colombia-importancia-para-lascomunidades. Publicado el 14 de marzo de 2024.

[9] Rakshit, S., Nair, G., and Baddeley, A. (2017). Secondorder analysis of point patterns on a network using any distance metric. Spatial Statistics, 22:129-154.

[10] Señal Memoria (2021). La travesía del agua: ríos de colombia. https://www.senalmemoria.co/agua-travesia-rios-de-colombia. Publicado el 26 de agosto de 2021.

[11] Silverman, B. W. (1986). Density Estimation for Statistics and Data Analysis. Chapman and Hall.

[12] Ver Hoef, J. M., Peterson, E. E., and Theobald, D. M. (2006). Some new spatial statistical models for stream networks. Environmental and Ecological Statistics, 13:449464.