Clase 03: Fundamentos de datos geoespaciales

Tipos y estructuras de datos espaciales

Profesor: Francisco Parra O.

26 de agosto de 2025

USACH - Ingeniería Civil en Informática

Agenda

Datos vectoriales: puntos, líneas, polígonos

Datos raster: grillas y resolución

Formatos de archivos comunes

Atributos y geometrías

Datos vectoriales: puntos, líneas, polígonos

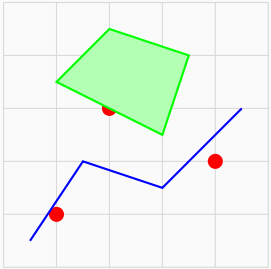
Modelo Vectorial: Fundamentos

Características:

- Representación discreta
- Geometrías precisas
- Topología explícita
- Atributos asociados

Ventajas:

- Alta precisión
- Eficiente en almacenamiento
- Análisis topológico



Modelo vectorial

Puntos: Localizaciones Discretas

Características:

- Coordenadas X, Y (Z opcional)
- Sin dimensión espacial
- Múltiples atributos

Aplicaciones:

- Ubicación de sensores
- Puntos de interés (POI)
- Eventos geográficos
- Muestras de campo

```python
from shapely.geometry import Point
import geopandas as gpd

# Crear punto
estacion = Point(-70.651, -33.438)
print(f"X: {estacion.x}")
print(f"Y: {estacion.y}")

# GeoDataFrame con puntos
puntos = gpd.GeoDataFrame({
    'id': [1, 2, 3],
    'tipo': ['sensor', 'muestra', 'POI'],
    'geometry': [
        Point(-70.651, -33.438),
        Point(-70.649, -33.437),
        Point(-70.652, -33.439)
    ]
})

```

Líneas: Conexiones y Redes

Características:

- Secuencia de vértices
- Longitud medible
- Dirección opcional
- Conectividad de red

Aplicaciones:

- Redes viales
- Ríos y cursos de agua
- Líneas de transmisión
- Rutas de transporte

```python
from shapely.geometry import LineString

# Crear linea
ruta = LineString([
    (-70.651, -33.438),
    (-70.649, -33.437),
    (-70.648, -33.439),
    (-70.650, -33.441)
])

# Propiedades
print(f"Longitud: {ruta.length}")
print(f"Vertices: {len(ruta.coords)}")

# Operaciones
punto_medio = ruta.interpolate(0.5,
                               normalized=True)
buffer_ruta = ruta.buffer(0.001)

```

Polígonos: Áreas y Regiones

Características:

- Anillo exterior cerrado
- Posibles huecos internos
- Área y perímetro
- Relaciones topológicas

Aplicaciones:

- Límites administrativos
- Parcelas/predios
- Zonas de cobertura
- Áreas de influencia

```python
from shapely.geometry import Polygon

# Crear poligono
manzana = Polygon([
    (-70.650, -33.440),
    (-70.648, -33.440),
    (-70.648, -33.438),
    (-70.650, -33.438),
    (-70.650, -33.440)
])

# Propiedades
print(f"Area: {manzana.area}")
print(f"Perimetro: {manzana.length}")

# Operaciones espaciales
edificio = Point(-70.649, -33.439)
print(manzana.contains(edificio))

```

Datos raster: grillas y resolución

Modelo Raster: Fundamentos

Características:

- Matriz regular de celdas
- Resolución espacial fija
- Valores continuos o discretos
- Múltiples bandas

Ventajas:

- Datos continuos
- Análisis de superficie
- Teledetección

Matriz de píxeles

Resolución Espacial

Concepto clave:

- Tamaño del píxel en terreno
- Trade-off: detalle vs. tamaño
- Escala de análisis

Resoluciones típicas

- Landsat: 30m
- Sentinel-2: 10m
- PlanetScope: 3m
- WorldView: 0.3m

```python
import rasterio
import numpy as np

# Abrir raster
with rasterio.open('imagen.tif') as src:
    # Metadatos
    print(f"CRS: {src.crs}")
    print(f"Res: {src.res}")
    print(f"Bounds: {src.bounds}")

    # Leer banda
    banda1 = src.read(1)

    # Estadisticas
    print(f"Min: {banda1.min()}")
    print(f"Max: {banda1.max()}")
    print(f"Mean: {banda1.mean()}")

```

Bandas Espectrales

Imágenes multiespectrales:

- RGB (visible)
- Infrarrojo cercano (NIR)
- Infrarrojo medio (SWIR)
- Térmico (TIR)

Índices espectrales:

- NDVI (vegetación)
- NDWI (agua)
- NDBI (construcciones)

```python
import rasterio
import numpy as np

# Abrir bandas
with rasterio.open('B4_red.tif') as red:
    b4 = red.read(1).astype(float)

with rasterio.open('B5_nir.tif') as nir:
    b5 = nir.read(1).astype(float)

# Calcular NDVI
ndvi = (b5 - b4) / (b5 + b4 + 1e-10)

# Clasificar vegetacion
vegetacion = ndvi > 0.3
agua = ndvi < 0
suelo = (ndvi >= 0) & (ndvi <= 0.3)

```

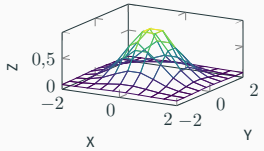
Modelos de Elevación Digital (DEM)

Tipos de DEM:

- DSM: Superficie con objetos
- DTM: Terreno sin objetos
- Derivados: pendiente, aspecto

Aplicaciones:

- Análisis hidrológico
- Visibilidad
- Modelado 3D
- Riesgo de inundación



Fuentes de DEM:

- SRTM (30m global)
- ASTER GDEM (30m)
- LiDAR (alta precisión)

Formatos de archivos comunes

Formatos Vectoriales

Formato	Características	Uso
Shapefile	Estándar ESRI, múltiples archivos	Universal
GeoJSON	Texto JSON, web-friendly	Web mapping
GeoPackage	SQLite, todo-en-uno	Moderno, móvil
KML/KMZ	XML, Google Earth	Visualización
PostGIS	Base de datos espacial	Producción

Recomendación

- Intercambio: GeoPackage ¿ Shapefile
- Web: GeoJSON para datos pequeños
- Producción: PostGIS para grandes volúmenes

Shapefile: El Estándar Legacy

Componentes obligatorios:

- .shp - geometrías
- .shx - índice espacial
- .dbf - atributos

Limitaciones:

- Nombres: 10 chars
- Tamaño máx: 2GB
- Sin topología

```python
import geopandas as gpd

# Leer shapefile
gdf = gpd.read_file('comunas.shp')

# Explorar estructura
print(gdf.head())
print(gdf.crs)
print(gdf.geometry.type.unique())

# Filtrar y guardar
santiago = gdf[gdf['COMUNA'] == 'Santiago']
santiago.to_file('santiago.shp')

# Convertir a otros formatos
gdf.to_file('comunas.geojson', driver='GeoJSON')
gdf.to_file('comunas.gpkg', driver='GPKG')

```

GeoJSON: Intercambio Web

Ventajas:

- Legible por humanos
- Soporte nativo web
- Un solo archivo
- Estándar RFC 7946

Estructura:

- FeatureCollection
- Features
- Geometry + Properties
- CRS:84 (WGS84)

```geojson
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [-70.651, -33.438]
      },
      "properties": {
        "nombre": "USACH",
        "tipo": "Universidad",
        "estudiantes": 22000
      }
    }
  ]
}
```

Formatos Raster

Formato	Características	Uso
GeoTIFF	TIFF + georeferencia	Universal
COG	Cloud Optimized GeoTIFF	Web/cloud
NetCDF	Multidimensional	Clima/tiempo
HDF5	Jerárquico, comprimido	Satélites
JP2000	Compresión wavelet	Ortofotos

Consideraciones:

- Compresión: sin pérdida vs con pérdida
- Pirámides: visualización multiescala
- Tiles: acceso eficiente a porciones

Cloud Optimized GeoTIFF (COG)

Optimizaciones:

- Tiles internos
- Overviews (pirámides)
- HTTP range requests
- Compresión eficiente

Beneficios:

- Streaming parcial
- Sin descarga completa
- Visualización rápida
- Cloud-native

```
# Convertir a COG
gdal_translate input.tif output_cog.tif \
  -of COG \
  -co COMPRESS=LZW \
  -co BLOCKSIZE=512

```

```
# Leer COG desde URL
import rasterio
from rasterio.windows import Window

url = 'https://example.com/cog.tif'
with rasterio.open(url) as src:
    # Leer solo una ventana
    window = Window(0, 0, 512, 512)
    data = src.read(1, window=window)

```

Atributos y geometrías

Integración Atributos-Geometría

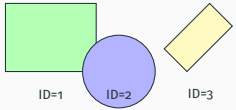
Modelo relacional espacial:

- Tabla = Feature Class
- Fila = Feature
- Columna geométrica especial
- Índice espacial

Tipos de atributos:

- Identificadores
- Descriptivos
- Numéricos
- Temporales
- Relacionales

ID	Nombre	Geometry
1	Parque	POLYGON(...)
2	Plaza	POLYGON(...)
3	Lago	POLYGON(...)



GeoPandas: DataFrames Espaciales

Extensión de Pandas:

- GeoDataFrame
- GeoSeries
- Operaciones vectorizadas
- Integración ecosistema

Funcionalidades:

- Joins espaciales
- Agregaciones
- Visualización
- I/O múltiples formatos

```python
import geopandas as gpd
import matplotlib.pyplot as plt

# Cargar datos
comunas = gpd.read_file('comunas.shp')
puntos = gpd.read_file('colegios.geojson')

# Join espacial
colegios = gpd.sjoin(puntos, comunas,
                     predicate='within')

# Agregacion
resumen = colegios.groupby('COMUNA').size()

# Visualizacion
ax = comunas.plot(column='POBLACION',
                  legend=True)
puntos.plot(ax=ax, color='red', markersize=2)

```

Operaciones Espaciales Fundamentales

Operaciones geométricas:

- Buffer
- Intersección
- Unión
- Diferencia
- Simplificación

Relaciones espaciales:

- Contains/Within
- Intersects
- Touches
- Overlaps
- Crosses

```python
from shapely.ops import unary_union
import geopandas as gpd

# Buffer
zonas_influencia = gdf.buffer(100)

# Interseccion
intersec = gdf1.overlay(gdf2,
                        how='intersection')

# Union de geometrias
union_total = unary_union(gdf.geometry)

# Consultas espaciales
cerca_metro = puntos[
    puntos.distance(estacion) < 500
]

# Dissolve por atributo
regiones = comunas.dissolve(by='REGION')

```

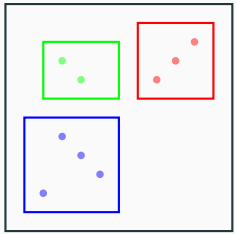
Indexación Espacial

R-tree (Rectangle tree):

- Estructura jerárquica
- Bounding boxes
- Búsqueda eficiente
- O(log n) promedio

Aplicaciones:

- Consultas de proximidad
- Joins espaciales
- Intersecciones
- KNN espacial



R-tree structure

Reduce búsquedas de O(n) a O(log n)

Validación y Limpieza Geométrica

Problemas comunes:

- Self-intersections
- Duplicados
- Gaps/Overlaps
- Slivers
- Topología inválida

Herramientas:

- Shapely: is_valid
- Buffer(0) trick
- Simplificación
- Snap to grid

```python
import geopandas as gpd
from shapely.validation import explain_validity

# Verificar validez
gdf['valido'] = gdf.geometry.is_valid

# Explicar problemas
invalidos = gdf[~gdf['valido']]
for idx, row in invalidos.iterrows():
    print(explain_validity(row.geometry))

# Corregir con buffer(0)
gdf['geometry'] = gdf.geometry.buffer(0)

# Eliminar slivers
gdf = gdf[gdf.geometry.area > 0.001]

# Simplificar
gdf['geometry'] = gdf.geometry.simplify(
    tolerance=1.0, preserve_topology=True)

```

Sistemas de Referencia de Coordenadas (CRS)

Componentes:

- Datum (modelo de la Tierra)
- Proyección cartográfica
- Unidades de medida
- Códigos EPSG

CRS comunes:

- WGS84 (EPSG:4326)
- Web Mercator (EPSG:3857)
- UTM zonas
- Lambert Conformal

Chile continental:

- EPSG:32718 (UTM 18S)
- EPSG:32719 (UTM 19S)
- EPSG:5361 (SIRGAS-Chile)

Transformaciones:

- Reproyección
- Cambio de datum
- Distorsiones inevitables

Machine Learning Espacial

Particularidades:

- Autocorrelación espacial
- Primera ley de Tobler
- Cross-validation espacial
- Feature engineering

Aplicaciones:

- Predicción de precios
- Clasificación cobertura
- Interpolación espacial
- Detección de patrones

```python
from sklearn.ensemble import RandomForestRegressor
import geopandas as gpd

# Features espaciales
gdf['dist_centro'] = gdf.distance(centro)
gdf['dist_metro'] = gdf.distance(metro)
gdf['n_vecinos'] = gdf.buffer(500).apply(
    lambda x: puntos.within(x).sum()
)

# Lag espacial
from libpysal.weights import KNN
w = KNN.from_dataframe(gdf, k=5)
gdf['precio_lag'] = w.lag(gdf['precio'])

# Modelo
X = gdf[['area', 'dist_centro',
         'dist_metro', 'precio_lag']]
y = gdf['precio']
model = RandomForestRegressor()
model.fit(X, y)
```

Resumen: Vector vs Raster

Aspecto	Vector	Raster
Estructura	Objetos discretos	Matriz continua
Precisión	Alta	Depende de resolución
Almacenamiento	Eficiente para objetos	Grande para áreas
Topología	Explícita	Implícita
Análisis	Redes, buffers	Álgebra de mapas
Visualización	Escalable	Pixelada al zoom
Casos de uso	Catastro, redes	Teledetección, DEM

Integración Vector-Raster:

- Rasterización: vector → raster
- Vectorización: raster → vector
- Zonal statistics: resumen raster por polígono
- Point sampling: extracción de valores raster

Ejercicio Práctico Integrador

Tarea: Analizar ubicación óptima para nuevo colegio

Datos disponibles:

- Manzanas censales (vector)
- Colegios existentes (puntos)
- Red vial (líneas)
- Población por edad (raster)
- Elevación (DEM)

Criterios:

1. Max población 5-18 años
2. Min 500m de otro colegio
3. Max 100m de vía principal
4. Pendiente ¡ 5 grados

```python
# 1. Preparar datos
manzanas = gpd.read_file('manzanas.shp')
colegios = gpd.read_file('colegios.geojson')
vias = gpd.read_file('vias.shp')

# 2. Crear buffers
buffer_colegios = colegios.buffer(500)
buffer_vias = vias.buffer(100)

# 3. Areas candidatas
candidatas = manzanas.copy()
candidatas = candidatas[
    ~candidatas.intersects(
        buffer_colegios.unary_union)]
candidatas = candidatas[
    candidatas.intersects(
        buffer_vias.unary_union)]

# 4. Score por poblacion
# (continua...)

```

Actividades Prácticas

Para implementar en el laboratorio:

1. **Exploración de datos vectoriales**
   - Cargar shapefile de comunas
   - Calcular área y perímetro
   - Identificar comuna más grande
2. **Análisis raster básico**
   - Cargar imagen satelital
   - Calcular NDVI
   - Clasificar cobertura vegetal
3. **Operaciones espaciales**
   - Buffer de 500m en estaciones
   - Contar POIs en buffers
   - Overlay de capas vectoriales

Próxima clase: CRS + Lab 1

Cierre

¿Preguntas?

francisco.parra.o@usach.cl

Material disponible en:
Plataforma del curso

Próxima sesión:
Jueves - CRS y Laboratorio 1