

# Geospatial Artificial Intelligence

Julien Ah-Pine ([julien.ah-pine@sigma-clermont.fr](mailto:julien.ah-pine@sigma-clermont.fr))



Clermont-Ferrand, 9-12 Décembre 2024

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

1 / 534

Introduction et motivations générales

## Rappel du Sommaire

### 1 Introduction et motivations générales

### 2 Concepts de base et outils Python associés

### 3 Analyse de la dépendance spatiale

### 4 Méthodes de régression spatiale

### 5 Méthodes d'ensemble en ML et extensions spatiales

### 6 Eléments de géostatistique

### 7 Méthodes de DL et extensions spatiales

## Sommaire

### 1 Introduction et motivations générales

### 2 Concepts de base et outils Python associés

### 3 Analyse de la dépendance spatiale

### 4 Méthodes de régression spatiale

### 5 Méthodes d'ensemble en ML et extensions spatiales

### 6 Eléments de géostatistique

### 7 Méthodes de DL et extensions spatiales

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

2 / 534

Introduction et motivations générales Motivations et définitions de base

## Rappel du Sommaire

### 1 Introduction et motivations générales

- Motivations et définitions de base
- Objectifs et contenu du cours

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

3 / 534

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

4 / 534

## Données spatiales et raisonnement géographique

- Les data scientists travaillent depuis de longue date avec des **données géographiques/spatiales**. Les cartes, en particulier, sont un type très courants d'“infographique” à l’ère d’internet.
- Les **données spatiales sont omniprésentes** ! Tout a un emplacement dans l'espace-temps, et cet emplacement peut être utilisé directement pour faire de meilleures prédictions ou déductions. La “géographie” est une position sur la Terre et la localisation permet de mieux comprendre les relations entre les observations.
- Ce sont souvent des **relations** qui sont utiles en science des données, car elles nous permettent de contextualiser nos données, en établissant des liens au sein des données existantes et au-delà (extrapolation).
- D’après le géographe Waldo Tobler, les **choses proches sont susceptibles d’être plus liées que les choses éloignées**, tant dans l’espace que dans le temps. Ainsi, si nous tirons profit de ces info. contextuelles, cela peut nous aider à construire de meilleurs modèles.

## Raisonnement géo. : importance des relations spatiales

- Comme un modèle statistique, une carte est une représentation approximative du processus spatial sous-jacent, mais n'est donc pas le processus lui-même ! Ces représentations ne sont pas exactement correctes dans un certain sens, mais elles sont malgré tout utiles pour comprendre ce qui est important dans un proc. spatial.
- Nous utiliserons le terme “**modèle de données**” pour désigner la manière dont nous représentons conceptuellement un proc. spatial.
- Nous utiliserons le terme “**structure de données**” pour désigner la manière dont les données spatiales sont représentées en machine.
- Nous abordons dans ce cours plusieurs **modèles** de données géographiques classiques, puis nous présentons leurs liens avec des **structures** de données géographiques typiques.

## Importance des données spatiales

- L’analyse de données spatiales est ainsi bien plus que la cartographie d’entités ou de mesures dans l’espace. Elle est fondamentale dans de nbs disciplines et applications comme par ex. :
  - Gestion de traffic en zone urbaine.
  - Progression géographique d’une maladie.
  - Gestion de risque naturel (coulées de lave, zones inondables, …)
  - Objectifs de Développement Durable, …
- En Sciences Humaines et Sociales, l’analyse de données spatiales trouve aussi bcp d’applications :
  - Sociologie : l’analyse des inégalités salariales régionales à l’échelle d’un pays montre clairement des dépendances spatiales.
  - Science politique : l’analyse des intentions de votes d’individus dépend souvent des opinions des individus à proximité.
  - Relations internationales : les pays subissant ou exposés à des conflits armés sont souvent voisins de pays avec la même expérience.
  - D’autres exemples dans [Goodchild and Janelle, 2004], …

## Modèle de données (représentation conceptuelle)

- Les **modèles** classiques de données utilisés dans les **Système d’Information Géographique (SIG)**, emploient les définition suivantes pour représenter un processus géographique :
  - Objects* : entités qui occupent une position spécifique dans le temps et l’espace. Par exemple : des bâtiments (obj. concret), des villes ou départements (obj. abstrait), …
  - Fields* : sont des surfaces continues qui peuvent, en théorie, être mesurées à n’importe quel endroit de l’espace et du temps. Par exemple : la densité de population, la température, …
  - Networks* : reflètent un ensemble de connexions entre des objets ou entre des positions dans un *field*. Par exemple : routes entre deux adresses, …
- La différence entre ces trois types d’éléments est importante dans la mesure où elle permet de considérer différents types de relations.

## Structure de données (représentation computationnelle)

- Les **modèles** de données sont des abstractions qui nous permettent de clarifier ce qui est important de représenter dans le processus spatial à l'étude de ce qui ne l'est pas.
- Les **structures de données** nous permettent d'instancier en **machine** les données relatives aux objets, champs et réseaux discutés précédemment en vue de leurs analyses statistiques.
- Les **structures de données** que nous étudions par la suite sont :
  - ▶ *Geographic Tables*,
  - ▶ *Surfaces (et cubes)*,
  - ▶ *Spatial Graphs*.
- Chacune de ces structures permet de stocker et manipuler des données relatives aux trois entités *objects*, *fields* et *networks*.

## Structure de données : Surfaces

- Les **Surfaces (SF)** permettent de stocker les mesures d'une var. observées sur un *field*. En théorie, il y a une infinité de localisations (surface) qu'il faudrait couvrir. En pratique, il y a un ens. fini d'obs.
- En **théorie**, il est important de considérer la **continuité** dans l'espace (et dans le temps) car cela permet de tenir compte des relations spatiales des données. En **pratique**, les positions sont représentées par une **grille** de points répartis de façon uniforme en 2D. Les dimensions de la grille sont relatives au domaine qu'elle représente.
- Contrairement aux *geographic tables*, les lignes et les colonnes d'une *surface 2D* indiquent des positions et les valeurs associées sont les observations de la variable considérée.
- Si l'on souhaite représenter plusieurs variables, par ex. la pollution de l'air et l'altitude d'un territoire, ou les mesures d'une variable spatiale en plusieurs dates, on a recourt à des *surface 3D* que l'on appelle également *cubes* ou *volumes*.

## Structure de données : *Geographic Tables*

- Les **Geographic Tables (GT)** permettent de stocker des données sur les *objects*. C'est une table à 2 entrées, les lignes représentant les *objects* distincts et les colonnes, les attributs de ces *objects*.
- Illustration :

	pop_est	continent		name	iso_a3	gdp_md_est	geometry
0	920938	Oceania		Fiji	FJI	8374.0	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...
1	53950935	Africa		Tanzania	TZA	150600.0	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...
2	603253	Africa		W. Sahara	ESH	906.5	POLYGON ((-8.66559 27.65643, -8.66512 27.58948...
3	35623680	North America		Canada	CAN	1674000.0	MULTIPOLYGON (((-122.84000 49.00000, -122.9742...
4	326625791	North America	United States of America	USA	USA	18560000.0	MULTIPOLYGON (((-122.84000 49.00000, -120.0000...
5	18556698	Asia		Kazakhstan	KAZ	460700.0	POLYGON ((87.35997 49.21498, 86.59878 48.54918...
6	29748859	Asia		Uzbekistan	UZB	202300.0	POLYGON ((55.96819 41.30864, 55.92892 44.9586...
7	6909701	Oceania		Papua New Guinea	PNG	28020.0	MULTIPOLYGON (((141.00021 -2.60015, 142.73525 ...
8	260580739	Asia		Indonesia	IDN	3028000.0	MULTIPOLYGON (((141.00021 -2.60015, 141.01706 ...
9	44293293	South America		Argentina	ARG	879400.0	MULTIPOLYGON (((-68.63401 -52.63637, -68.25000...

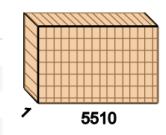
Extrait de [Rey et al., 2020]

## Structure de données : Surfaces

Extrait de [Rey et al., 2020]

xarray.DataArray (band: 1, y: 3515, x: 5510)

	Array	Chunk
Bytes	38.74 MB	386.80 kB
Shape	(1, 3515, 5510)	(1, 551, 351)
Count	113 Tasks	112 Chunks
Type	int16	numpy.ndarray



## ▼ Coordinates:

band	(band)	int64	1
y	(y)	float64	33.51 33.5 33.5 ... 32.53 32.53
x	(x)	float64	-117.6 -117.6 ... -116.1 -116.1

## ▼ Attributes:

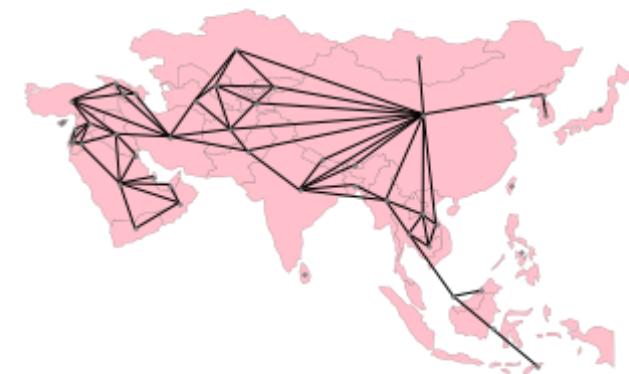
transform :	(0.0002777777777777778, 0.0, -117.61125, 0.0, -0.0002777777777777778, 8, 33.50513888888894)
crs :	+init=epsg:4326
res :	(0.0002777777777777778, 0.0002777777777777778)
is_tiled :	0
nodatavals :	(-32768.0,)
scales :	(1.0,)
offsets :	(0.0,)
AREA_OR_POINT :	Area

## Structure de données : Spatial Graphs

- Les **Spatial Graphs (SG)** permettent de stocker des relations de nature spatiale ou topologique entre *objects*. Contrairement aux *geographic tables*, qui traitent les *objects* de façon indépendante, les *spatial graphs* indiquent comment chq. *object* est spatialement connecté aux autres *objects* du jeu de données et permet donc de les mettre en relation. Il existe plusieurs façons de définir des relations spatiales (cf plus loin).
- Les *spatial graphs* encodent dans des structures de données des **relations spatiales entre objects** ce qui est un prérequis pour toute analyse spatiale des données géographiques.
- Ces analyses sont riches et proviennent de **différentes communautés** : analyse exploratoire des autocorrélations spatiales, économétrie régionale, géostatistique, ... . De ce fait, sous le vocable *spatial graphs*, on regroupe les structures suivantes : *spatial weights matrices*, *adjacency matrices*, *spatial networks*, ...

## Structure de données : Spatial Graphs

Extrait de [Rey et al., 2020]



## Data is data ! (pour un data scientist)

- Traditionnellement, les entités principales dans les SIG sont les *objects* et les *fields* avec le couplage suivant :
  - ▶ Les *objects* sont des **vecteurs** dont les coordonnées sont stockées dans la GT.
  - ▶ Les *fields* sont des **rasters (images matricielles)** dont chaque pixel représente une localisation du domaine.
- Les SG n'étaient pas des entités principales au sein des SIG. Mais cela a évolué dernièrement. Dans la perspective du **spatial data science**, cette modélisation en deux catégories, *objects* et *fields*, est moins primordiale. En effet, les méthodes de **Machine Learning (ML)** utilisent l'une ou l'autre des représentations et peuvent également en créer de nouvelles. Si la finalité est la prédiction en des localisations nouvelles (*Out Of Sample -OOS-*), les méthodes de ML utiliseront la représentation donnant les meilleures performances. Dans ce contexte, l'importance de la représentation des données spatiales, telle qu'elle est considérée au sein de la communauté des SIG, est plus faible.

## Data is data ! (pour un data scientist) (suite)

- A l'ère du **Big Data** et de l'**IoT**, nous avons accès à de nombreuses sources d'informations ouvertes :
  - ▶ **Données ouvertes géoréférencées :**
    - Ex. : données de recensement, infrastructures (routes, bâtiments), données climatiques, ...
    - Ces données sont disponibles gratuitement sur des plateformes comme OpenStreetMap, GeoNames ou Data.gov.
    - Elles peuvent servir pour la planification urbaine, la gestion des transports, l'analyse socio-économique, la veille sanitaire, ...
  - ▶ **Images Satellites :**
    - Ex. : Sentinel-2 (ESA), Landsat (NASA), MODIS (NASA).
    - Ces données permettent une couverture globale avec l'accès à des images récentes et historiques couvrant toute la planète.
    - Les technologies de l'imagerie multispectrale permettent de surveiller la végétation, l'eau, les sols, ...
    - Elles permettent la surveillance environnementale, la gestion des ressources naturelles, la gestion des catastrophes naturelles, ...

## Rappel du Sommaire

- 1** Introduction et motivations générales
  - Motivations et définitions de base
  - Objectifs et contenu du cours

## Contenu du cours (suite)

- Le programme du cours sur **4 jours** est le suivant (suite) :
  - ▶ **Jour 4** (avec Khaled Al-Saih, Phd)
    - Introduction aux images satellites.
    - Introduction aux réseaux de neurones à convolution.
    - Modèles de Deep Learning pour l'analyse d'images satellites.
    - Méthodes de fusion d'informations avec les images satellites.
- La partie pratique se fait avec le **langage Python**.

## Contenu du cours

- **Introduction à la science des données spatiales et au GeoAI.**
- Le programme du cours sur **4 jours** est le suivant :
  - ▶ **Jour 1**
    - Définitions de base en analyse de données spatiales.
    - Concepts et outils pour l'analyse des dépendances spatiales.
    - Les modèles de régression linéaire spatiale.
  - ▶ **Jour 2**
    - Introduction aux méthodes d'ensemble en Machine Learning.
    - Représentation de données spatiales par Moran Eigenvector Maps.
    - Modèle spatialement explicite Geographical Random Forest.
  - ▶ **Jour 3**
    - Introduction aux champs aléatoires et à la géostatistique.
    - Introduction aux réseaux de neurones et à l'apprentissage profond.
    - Modèle DeepKriging pour les champs spatiaux non stationnaires.

## Références du cours

- Les sections 1 à 4 du sommaire sont reprises de l'excellent cours en ligne suivant [Rey et al., 2020] :
  - ▶ **Geographic Data Science with Python**.
  - ▶ Réalisé par : Sergio J. Rey, Dani Arribas-Bel, Levi J. Wolf.
  - ▶ En ligne : <https://geographicdata.science/book/intro.html>.
  - ▶ Sous la Licence Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
- Les modifications ponctuelles apportées concernent :
  - ▶ la traduction partielle des notes de cours,
  - ▶ la mise en forme sous forme de présentation avec Beamer,
  - ▶ l'utilisation dérivée des notebooks pour le suivi des cours.
- Le support de cours ne doit pas être diffusé.
- Les sections 5 à 7 reposent sur plusieurs références et articles scientifiques cités au fil de l'eau.

## Supports de cours

- Pour récupérer les supports de cours, scripts et données :
   
<https://drive.uca.fr/d/80e900ceb53a48c78560/>



- Pour installer les librairies et activer l'environnement :
  - ▶ Installer conda si cela n'est pas encore fait :
   
<https://www.anaconda.com/download>.
  - ▶ Ouvrez un terminal conda et exécutez :
   
`conda env create -f geoai.yml`
  - ▶ Activez l'environnement :
   
`conda activate geoai.yml`

## Rappel du Sommaire

- ### 2 Concepts de base et outils Python associés
- Manipuler des données spatiales avec Python
    - Spatial Weights (SW)
      - Relations binaires de contiguïté
      - Relations spatiales basées sur les distances

## Rappel du Sommaire

- 1 Introduction et motivations générales
- 2 Concepts de base et outils Python associés
- 3 Analyse de la dépendance spatiale
- 4 Méthodes de régression spatiale
- 5 Méthodes d'ensemble en ML et extensions spatiales
- 6 Eléments de géostatistique
- 7 Méthodes de DL et extensions spatiales

## Introduction

- Nous voyons comment s'instancient les modèles et structures de données précédentes dans le langage Python et de ses modules adéquats. Dans la suite, nous utiliserons les abréviations suivantes :
  - ▶ GT pour *Geographic Tables*,
  - ▶ SF pour *Surfaces*,
  - ▶ SG pour *Spatial Graphs*.
- Dans cette perspective, nous présenterons le code et ses résultats. Vous avez à votre disposition des scripts vous permettant également d'exécuter ces codes à partir de votre machine.
- Les modules, classes et commandes qui suivent ne sont pas exhaustives : il existe d'autres façons d'obtenir ces mêmes résultats.
- Nous utiliserons respectivement les modules Python suivants :
  - ▶ geopandas,
  - ▶ xarray,
  - ▶ NetworkX.

## Geographic Tables (GT)

- Les **GT** servent à **décrire des objets géographiques** et peuvent être vues comme une feuille de calculs.
- Chq. ligne de la table correspond à un seul objet géographique et les différentes colonnes enregistrent les mesures de différents attributs ou variables de ces objets.
- Il existe spécifiquement **au moins une colonne encodant une information géométrique** sur les objets géographiques.
- Ce type de structure de données permet de tenir compte des éléments géographiques dans des bases de données relationnelles.
- Dans notre cas, en science des données, nous utilisons Python pour manipuler et analyser ce type d'objets et le module geopandas est l'outil de référence pour les GT.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

25 / 534

## \*\*Code\*\* : GT et geopandas (suite)

- Chq. ligne de la table est un pays. Chq. pays a deux attributs :
  - le nom du pays (**ADMIN**) qui est de type **str**.
  - la géométrie du pays représentée par un polygone (**geometry**) qui est un type d'objet en Python encodant des objets géométriques :

```

1 # print the object type of gt_polygons
2 print(type(gt_polygons))
3 # print the object type of gt_polygons.geometry
4 print(type(gt_polygons.geometry[0]))

```

```

1 <class 'geopandas.geodataframe.GeoDataFrame'>
2 <class 'shapely.geometry.multipolygon.MultiPolygon'>

```

- Chq. ligne et colonne a un indice spécifique permettant de l'identifier de façon unique dans la table.
- Ceci est un ex. de GT et nous allons donc utiliser la classe `geopandas.GeoDataFrame` pour ce type de structure de données.
- Remarque : l'extension `.gpkg` est un format de GT classique dans les SIG.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

27 / 534

## \*\*Code\*\* : GT et geopandas

```

1 # install geopandas if necessary
2 # import sys
3 # !{sys.executable} -m conda install geopandas
4 #import geopandas module without alias for clarity reasons
5 import geopandas
6 #load the data in countries_clean.gpkg into gt_polygons
7 gt_polygons = geopandas.read_file("countries_clean.gpkg")
8 #look at an excerpt of gt_polygons to get a glance of a geographical table
9 gt_polygons.head()

```

	ADMIN	geometry
0	Indonesia	MULTIPOINT ((13102705.696 463877.598, 13102...
1	Malaysia	MULTIPOINT ((13102705.696 463877.598, 13101...
2	Chile	MULTIPOLYGON (((-7737827.685 -1979875.500, -77...
3	Bolivia	POLYGON ((-7737827.685 -1979875.500, -7737828...
4	Peru	MULTIPOINT ((-7737827.685 -1979875.500, -77...

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

26 / 534

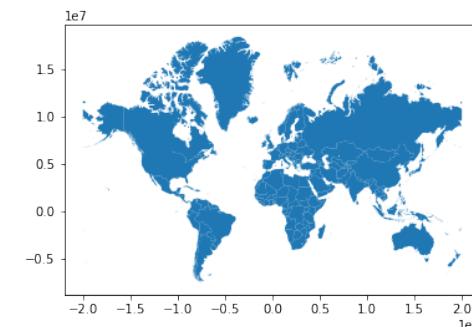
## \*\*Code\*\* : GT et geopandas (suite)

- La colonne de type **geometry** est particulière et possède des propriétés distinctes des colonnes "normales" comme **ADMIN** de type **str**.
- La méthode `plot` d'un `GeoDataFrame` utilise la colonne `geometry` afin de tracer les formes des objets.

```

1 # plot the geometry objects based on the default geometry feature
2 gt_polygons.plot()

```



J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

28 / 534

## \*\*Code\*\* : GT et geopandas (suite)

- La colonne `geometry` possède des propriétés particulières, on peut par ex., déterminer un centroïde pour chq. polygone (pays) ce qui correspond au point qui minimise en moyenne la distance avec ts les points sur la frontière du polygone.

```

1 # compute the centroid of each country and store in a new column
2 gt_polygons["centroid"] = gt_polygons.geometry.centroid
3 print(gt_polygons.head())
4 # set_geometry allows one to specify the geometry to used to represent objects
5 ax = gt_polygons.set_geometry("centroid").plot("ADMIN", markersize=5)
6 # plot polygons without color filling and add the centroid
7 gt_polygons.plot("ADMIN", ax=ax, facecolor="none", edgecolor="k", linewidth=0.2)

```

```

1      ADMIN           geometry \
2 0  Indonesia  MULTIPOLYGON (((13102705.696 463877.598, 13102...
3 1  Malaysia   MULTIPOLYGON (((13102705.696 463877.598, 13101...
4
5             centroid
6 0  POINT (13055431.810 -248921.141)
7 1  POINT (12211696.493 422897.505)

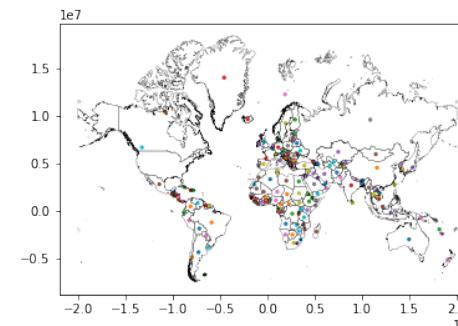
```

## Remarque sur les geographic objects

- Le *Open Geospatial Consortium* (OGC) a établi un ens. de types "abstrait" pour définir toutes sortes de géométries ou formes correspondant à des objets géographiques. Ceux-ci sont codifiés par la norme ISO 19125-1 (*simple features specification*) :
  - Point : position avec deux coordonnées *x* et *y*.
  - LineString : segment composé d'un ens. de plus de un Point.
  - Polygon : surface qui a au moins une LineString qui débute et se termine au même Point.
  - ...
- Tous ces types ont des variantes Multi qui indiquent une collection d'éléments de même type.

## \*\*Code\*\* : GT et geopandas (suite)

Extrait de [Rey et al., 2020]



## \*\*Code\*\* : GT et geopandas (suite)

- Exemple d'un objet de type Polygon :

```

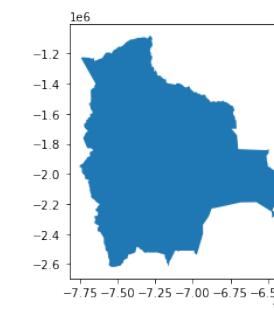
1 # print the features of the line whose ADMIN value is 'Bolivia'
2 print(gt_polygons.query('ADMIN == "Bolivia")')
3 # plot the polygon whose ADMIN value is 'Bolivia',
4 gt_polygons.query('ADMIN == "Bolivia").plot()

```

```

1      ADMIN           geometry \
2 3  Bolivia    POLYGON ((-7737827.685 -1979875.500, -7737828....

```

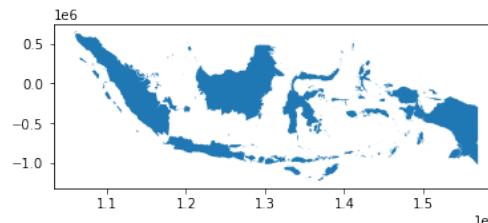


## \*\*Code\*\* : GT et geopandas (suite)

- Exemple d'un objet de type MultiPolygon :

```
1 # print the features of the line whose ADMIN value is "Indonesia"
2 print(gt_polygons.query('ADMIN == "Indonesia"'))
3 # plot the multipolygon whose ADMIN value is "Indonesia",
4 gt_polygons.query('ADMIN == "Indonesia").plot()
```

```
1      ADMIN           geometry \
2 0  Indonesia  MULTIPOLYGON (((13102705.696 463877.598, 13102...
```



## \*\*Code\*\* : SF et xarray

```
1 # install rasterio and xarray if necessary
2 # !{sys.executable} -m conda install rasterio
3 # !{sys.executable} -m conda install xarray
4 # load the data in ghs1_sao_paulo.tif into pop
5 pop = xarray.open_rasterio("ghs1_sao_paulo.tif")
6 # print the object type of pop : xarray allows multi-dimensional labeled arrays
7 print(type(pop))
8 # print the coordinates
9 print(pop.coords)
10 # print information required to convert pixels in the array to locations on the Earth
    # surface (eg. transform and crs)
11 print(pop.attrs)
12 # print the dimension of pop as a xarray.DataArray
13 print(pop.shape)
```

## Surfaces (SF)

- Les **SF** sont utilisées pour stocker un **champ de données** représentant des observations d'un phénomène spatial.
- En théorie, ce champ est une surface qui en toute position d'un espace continu associe une valeur.
- En pratique, les champs sont mesurés en des ensembles discrets et finis de points.
- Afin de se rapprocher d'un espace continu, les points de mesure sont répartis de façon structurée et uniforme au travers du domaine. Il s'agit en fait d'une grille 2D régulière de points et chacun d'entre eux est une localisation en laquelle une valeur est observée.
- Une grille 2D peut être vue comme une table à deux entrées mais contrairement aux GT, **les lignes et colonnes correspondent à une position spatiale**. A l'inverse, dans une GT, l'information spatiale est encodée dans une colonne.
- Nous utiliserons la lib. `xarray` pour représenter et manipuler les SF.

## \*\*Code\*\* : SF et xarray (suite)

```
1 <class 'xarray.core.dataarray.DataArray'>
2 Coordinates:
3   * band      (band) int64 1
4   * y         (y) float64 -2.822e+06 -2.822e+06 ... -2.926e+06 -2.926e+06
5   * x         (x) float64 -4.482e+06 -4.482e+06 ... -4.365e+06 -4.365e+06
6   {'transform': (250.0, 0.0, -4482000.0, 0.0, -250.0, -2822000.0), 'crs': '+proj=moll +
    lon_0=0 +x_0=0 +y_0=0 +datum=WGS84 +units=m +no_defs=True', 'res': (250.0, 250.0),
    'is_tiled': 0, 'nodatavals': (-200.0), 'scales': (1.0,), 'offsets': (0.0,), 'AREA_OR_POINT': 'Area', 'grid_mapping': 'spatial_ref'}
7 (1, 416, 468)
```

## \*\*Code\*\* : SF et xarray (suite)

```

1 # reduce the 3rd dimension and keep only the two geographic
2 pop.sel(band=1).data
3 # examine the same attributes then before
4 print(type(pop.sel(band=1)))
5 print(pop.sel(band=1).coords)
6 print(pop.sel(band=1).attrs)
7 print(pop.sel(band=1).shape)
8 # in each position a value is stored. The attribute data is an numpy array
9 print(pop.sel(band=1).data[0:5,:,:])
10 # plot the data
11 pop.sel(band=1).plot();

```

```

1 <class 'xarray.core.dataarray.DataArray'>
2 Coordinates:
3   band    int64 1
4   * y      float64 -2.822e+06 -2.822e+06 ... -2.926e+06 -2.926e+06
5   * x      float64 -4.482e+06 -4.482e+06 ... -4.365e+06 -4.365e+06
6   'transform': (250.0, 0.0, -4482000.0, 0.0, -250.0, -2822000.0), 'crs': '+proj=moll +
    lon_0=0 +x_0=0 +y_0=0 +datum=WGS84 +units=m +no_defs=True', 'res': (250.0, 250.0),
    'is_tiled': 0, 'nodatavals': (-200.0,), 'scales': (1.0,), 'offsets': (0.0,), 'AREA_OR_POINT': 'Area', 'grid_mapping': 'spatial_ref'}
7 (416, 468)
8 [[-200. -200. -200. ... -200. -200.]
9 [-200. -200. -200. -200. -200.]
10 [-200. -200. -200. ... -200. -200.]
11 [-200. -200. -200. ... -200. -200.]
12 [-200. -200. -200. ... -200. -200.]]

```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

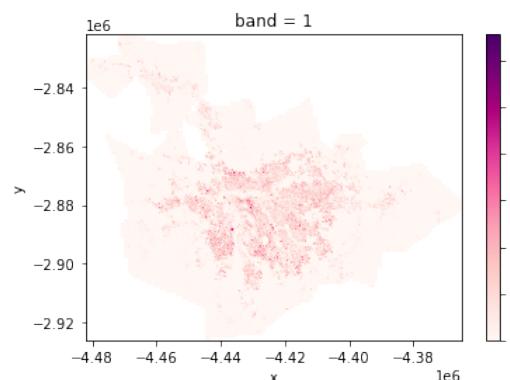
37 / 534

## \*\*Code\*\* : SF et xarray (suite)

- On peut sélectionner uniquement les valeurs différentes de -200 avec la méthode `where`. Les valeurs retenues seront donc non négatives.

```
1 pop.where(pop != -200).sel(band=1).plot(cmap="RdPu");
```

Extrait de [Rey et al., 2020]



J. Ah-Pine

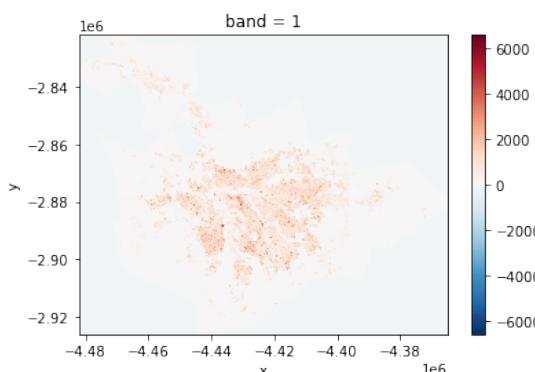
Masterclass IHEDD-CERDI-GDN GeoAI 24

39 / 534

## \*\*Code\*\* : SF et xarray (suite)

- Densité de population dans la région de São Paulo au Brésil.
- La valeur négative -200 encode en fait une donnée manquante (cf nodatavals précédemment) et correspond à une couleur bleu pastel.

Extrait de [Rey et al., 2020]



J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

38 / 534

## Spatial Graphs (SG)

- Les SG enregistrent des **connections spatiales entre objets géographiques**.
- Ces connections peuvent être de différentes natures :
  - Topologie géographique (par ex., la contiguïté).
  - Distances.
  - Flux d'interactions (par ex., échanges commerciaux, réseaux de communications, ...)
- Les SG sont distincts des GT et des SF :
  - Ils ne stockent pas les mesures d'un phénomène donné mais se focalisent avant tout sur les **connections entre objets**.
  - Les données sont de nature relationnelle et donc **non structurées**. Par ex., un objet peut être connecté à un unique autre objet alors qu'un autre objet peut être connecté à plusieurs autres objets.
- Nous illustrons ici les SG à l'aide de la lib. osmnx permettant d'interroger des données d'OpenStreetMap. Cependant networkx est la lib. sous-jacente permettant de manipuler des graphes en Python.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

40 / 534

**\*\*Code\*\*** : SG et osmnx et networkx

- Nous allons interroger OpenStreetMap et représenter les chemins de la place Bellecour.
  - Chaque noeud est une position et une arête rejoint deux noeuds s'il définit dans l'espace un chemin piéton.

```
1 # install osmnx if necessary
2 # !{sys.executable} -m conda install osmnx
3 # import osmnx
4 import osmnx
5 # store a SG of place Bellecour in graph
6 graph = osmnx.graph_from_place(query="Place Bellecour, Lyon"
7 # print the object type of graph
8 print(type(graph))
9 # print the nb of nodes in the graph
10 print(len(graph.nodes))
11 # print the nb of edges in the graph
12 print(len(graph.edges))
13 # plot the graph
14 osmnx.plot_graph(graph)
```

**\*\*Code\*\*** : SG et osmnx et networkx (suite)

```
1 <class 'networkx.classes.multidigraph.MultiDiGraph'>
2 26
3 58
```

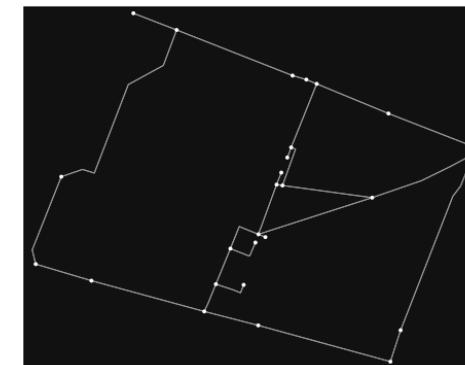
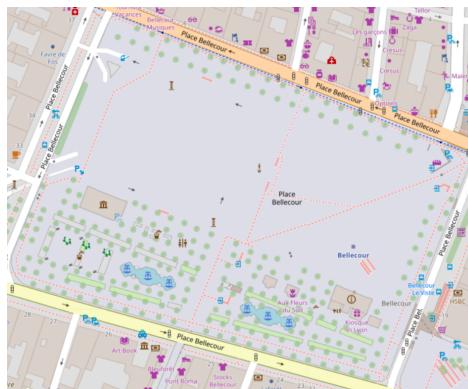


Image carte OSM de la Place Bellecour



## Rappel du Sommaire

### 2 Concepts de base et outils Python associés

- Manipuler des données spatiales avec Python
- Spatial Weights (SW)
  - Relations binaires de contiguïté
  - Relations spatiales bases sur les distances

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

45 / 534

## SW, SG, topologie, matrice d'adjacence

- En pratique, il s'agit de déterminer **pour chq. paire de geographic objects**, une **quantification de la relation spatiale** considérée. Pour chq. objet, nous définissons ainsi une **notion de voisinage** et cela renvoie au concept mathématique de **topologie**.
- Les **SW sont en fait des SG** qui encodent une topologie exprimant les relations de connectivité entre objets et qui nous permettent d'analyser spatialement ces derniers.
- En pratique, **les SW sont des matrices carrés de comparaison par paires** identiques au concept de **matrices d'adjacences** binaires ou pondérées représentant un graphe.
- Dans le contexte de données géographiques, nous distinguons deux types de relations spatiales :
  - ▶ Poids associés à des **rel. de contiguïté/adjacence** entre objets.
  - ▶ Poids associés à des **mesures de distances** entre objets.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

47 / 534

## Introduction

- Le vocable **Spatial Weights (SW)** renvoie au concept de *Spatial Graph* vu précédemment. C'est une dénomination très utilisée en stat. spatiales que nous reprenons en science de données spatiales.
- Implicitement, il s'agit de connecter des *geographic objects* représentés dans des GT en utilisant des **relations spatiales**. Ces dernières peuvent être de différentes natures et correspondent, par ex., à des réponses à des questions aussi diverses que les suivantes :
  - ▶ Quels sont les voisins autour d'un objet donné ?
  - ▶ Combien de stations essences sont à moins de 5 km de ma position ?
  - ▶ ...
- Les notions de proximité ou de connection géographique entre objets sont centrales et représentent en fait le **mécanisme principal au travers duquel les relations spatiales sont intégrées dans les méthodes d'analyses** que nous explorerons par la suite.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

46 / 534

## Poids associés à des relations de contiguïté

- Deux *geographic objects* sont **contigus ou adjacents** s'ils ont une **frontière en commun**.
- La relation de contiguïté n'est pas aussi simple qu'elle le paraît. Il existe plus. façons de définir la notion de "partager une frontière".
- Dans ce qui suit, nous allons illustrer ces dif. possibilités en utilisant du code permettant de générer des exemples illustrant le propos.
- Ce code sert également comme moyen de pratiquer la programmation Python pour la science des données spatiales.
- Nous utiliserons notamment la lib. `pysal` qui est centrale en science des données spatiales en Python.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

48 / 534

## \*\*Code\*\* : Contiguïté

- Nous considérons une grille  $3 \times 3$  tel un échiquier et le **déplacement possible d'une tour aux échecs**.

```

1 # install pysal if necessary
2 # !{sys.executable} -m conda install pysal
3 # import the following modules or submodules
4 import numpy
5 from shapely.geometry import Polygon
6 # Create a sequence 0, 1, 2
7 l = numpy.arange(3)
8 # Create a grid with the cartesian product of the previous sequence
9 xs, ys = numpy.meshgrid(l, l)
10 # Initialize a list
11 polys = []
12 # Generate polygons that will pave a 3 times 3 grid with squares
13 for x, y in zip(xs.flatten(), ys.flatten()):
14     poly = Polygon([(x, y), (x + 1, y), (x + 1, y + 1), (x, y + 1)])
15     polys.append(poly)
16 # Convert to GeoSeries
17 polys = geopandas.GeoSeries(polys)
18 gdf = geopandas.GeoDataFrame(
19     {
20         "geometry": polys,
21         "id": ["P-%s" % str(i).zfill(2) for i in range(len(polys))],
22     }
23 )

```

## \*\*Code\*\* : Contiguïté (suite)

Extrait de [Rey et al., 2020]

P-06	P-07	P-08
P-03	P-04	P-05
P-00	P-01	P-02

## \*\*Code\*\* : Contiguïté (suite)

```

1 # import matplotlib.pyplot
2 import matplotlib.pyplot as plt
3 # Plot grid geotable
4 ax = gdf.plot(facecolor="w", edgecolor="k")
5 # Loop over each cell and add the text
6 for x, y, t in zip(
7     [p.centroid.x - 0.25 for p in polys],
8     [p.centroid.y - 0.25 for p in polys],
9     [i for i in gdf["id"]]),
10 ):
11     plt.text(
12         x,
13         y,
14         t,
15         verticalalignment="center",
16         horizontalalignment="center",
17     )
18 # Remove axes
19 ax.set_axis_off()
20 plt.show()

```

## \*\*Code\*\* : Contiguïté de type tour et pysal

- Une tour se déplace uniquement verticalement ou horizontalement.  
Nous définissons la **contiguïté Rook** en conséquence.

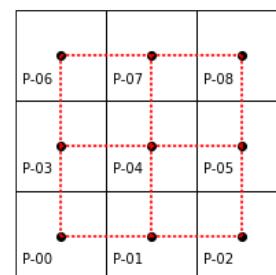
```

1 # import tools from pysal
2 from pysal.lib import weights
3 # Build a rook contiguity matrix from a regular 3x3
4 wr = weights.contiguity.Rook.from_dataframe(gdf)
5 # Set up figure
6 f, ax = plt.subplots(1, 1, subplot_kw=dict(aspect="equal"))
7 # Plot grid
8 gdf.plot(facecolor="w", edgecolor="k", ax=ax)
9 # Loop over each cell and add the text
10 for x, y, t in zip(
11     [p.centroid.x - 0.25 for p in polys],
12     [p.centroid.y - 0.25 for p in polys],
13     [i for i in gdf["id"]]),
14 ):
15     plt.text(
16         x,
17         y,
18         t,
19         verticalalignment="center",
20         horizontalalignment="center",
21     )
22 # Plot weights connectivity
23 wr.plot(gdf, edge_kws=dict(color="r", linestyle=":"), ax=ax)
24 # Remove axes
25 ax.set_axis_off()

```

## \*\*Code\*\* : Contiguïté de type tour et pysal (suite)

Extrait de [Rey et al., 2020]



- Déplacement de la tour d'une position à une position contigüe :
  - De P-00, la tour peut aller en P-01 ou P-03 (mais pas en P-004).
  - De P-04, la tour peut aller en P-01 ou P-03 ou P-05 ou P-07.
  - ...

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

53 / 534

## \*\*Code\*\* : Contiguïté de type tour et pysal (suite)

- On peut également représenter le graphe par la matrice d'adjacence.

```
1 # import pandas
2 import pandas
3 # convert an unpacked list into a dataframe
4 pandas.DataFrame(*wr.full()).astype(int)
```

```
1 Out [44]:
2   0 1 2 3 4 5 6 7 8
3 0 0 1 0 1 0 0 0 0 0
4 1 1 0 1 0 1 0 0 0 0
5 2 0 1 0 0 0 1 0 0 0
6 3 1 0 0 0 1 0 1 0 0
7 4 0 1 0 1 0 1 0 1 0
8 5 0 0 1 0 1 0 0 0 1
9 6 0 0 0 1 0 0 0 1 0
10 7 0 0 0 0 1 0 1 0 1
11 8 0 0 0 0 0 1 0 1 0
```

- Rq. : la matrice d'adjacence est remplie de bcp de 0. Si le sommet P-00 est adjacent à P-01 et P-03 cela veut aussi dire qu'il ne l'est pas des autres sommets. On pourrait ne retenir que l'information des voisins de chq. sommet comme précédemment et libérer l'espace des cellules prises par les valeurs nulles (format sparse).

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

55 / 534

## \*\*Code\*\* : Contiguïté de type tour et pysal (suite)

- Le graphe en rouge précédent peut être encodé par une matrice d'adjacence où les sommets sont les 9 positions possibles et les arêtes indiquent les positions sur lesquelles peuvent aller une tour en quittant une position donnée.
- Un graphe non orienté comportant au plus une arête entre sommets peut être représenté de façon équivalente par la liste des sommets adjacents/voisins de chaque sommet.

```
1 wr.neighbors
```

```
1 Out [9]:
2 {0: [1, 3],
3  1: [0, 2, 4],
4  2: [1, 5],
5  3: [0, 4, 6],
6  4: [1, 3, 5, 7],
7  5: [8, 2, 4],
8  6: [3, 7],
9  7: [8, 4, 6],
10 8: [5, 7]}
```

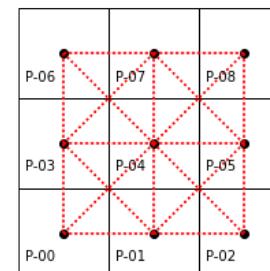
J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

54 / 534

## \*\*Code\*\* : Contiguïté de type reine et pysal

- Poursuivant l'alégorie des échecs, nous pouvons également définir une **contiguïté de type reine** et obtenir le graphe de contiguïté et la matrice d'adjacence suivants.



Extrait de [Rey et al., 2020]

```
1 Out [52]:
2   0 1 2 3 4 5 6 7 8
3 0 0 1 0 1 1 0 0 0
4 1 1 0 1 1 1 1 0 0
5 2 0 1 0 0 1 1 0 0
6 3 1 1 0 0 1 0 1 1
7 4 1 1 1 1 0 1 1 1
8 5 0 1 1 0 1 0 0 1
9 6 0 0 0 1 1 0 0 1
10 7 0 0 0 1 1 1 1 0
11 8 0 0 0 0 1 1 0 1
```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

56 / 534

## \*\*Code\*\* : Contiguïté de type reine et pysal (suite)

- Un objet de type weights de pysal possède plusieurs attributs relatifs à des concepts de graphe comme le degré, ...

```

1 # Akin to how the 'neighbors' dictionary encodes the contiguity relations, the 'weights' dictionary encodes the strength of the link connecting the focal to each neighbor. For contiguity weights, observations are usually either considered "linked" or "not linked," so the resulting weights matrix is binary.
2 print(wq.weights)
3 # The 'cardinalities' attribute reports the number of neighbors for each observation
4 print(wq.cardinalities)
5 # The related 'histogram' attribute provides an overview of the distribution of these cardinalities
6 print(wq.histogram)
7 # We can obtain a quick visual representation by converting the cardinalities into a 'pandas.Series' and creating a histogram
8 pandas.Series(wq.cardinalities).plot.hist(color="k");

```

```

1 {0: [1.0, 1.0, 1.0], 1: [1.0, 1.0, 1.0, 1.0], 2: [1.0, 1.0, 1.0], 3: [1.0, 1.0, 1.0, 1.0], 4: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0], 5: [1.0, 1.0, 1.0, 1.0, 1.0], 6: [1.0, 1.0, 1.0], 7: [1.0, 1.0, 1.0, 1.0, 1.0], 8: [1.0, 1.0, 1.0]}
2 {0: 3, 1: 5, 2: 3, 3: 5, 4: 8, 5: 5, 6: 3, 7: 5, 8: 3}
3 [(3, 4), (4, 0), (5, 4), (6, 0), (7, 0), (8, 1)]

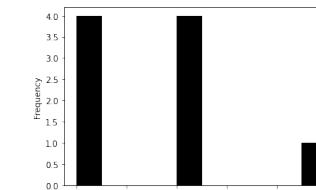
```

## Application à un cas d'étude réel : données issues d'une GT

- Précédemment, nous avons utilisé un ex. artificiel pour introduire la logique et certaines propriétés de la classe weights de la lib. pysal.
- En pratique, ces relations de contiguïté ne sont pas explicites et nous devons les déterminer à partir d'informations issues de GT ou de SF.
- Le premier exemple que vous voyons consiste à déterminer la topologie d'un ensemble d'objets représentés par les lignes d'une GT. L'étude de cas concerne des données de recensement de la ville de San Diego :
  - Les objets sont les districts (tracts).
  - La relation de contiguïté est basée sur les frontières entre districts.
- La lib. pysal permet d'extraire facilement des rel. topo. d'un GT.

## \*\*Code\*\* : Contiguïté de type reine et pysal (suite)

Extrait de [Rey et al., 2020]



- Les attr. cardinalities et histogram permettent de constater des asymétries dans la distribution du nb. de voisins. Ceci peut avoir un certain intérêt dans le cadre de méthodes d'analyse.
- Les attr. s0 et pct\_nonzero donnent le nb. et le pourc. de cellules de valeurs non nulles et donc une mesure de la densité du graphe.

```

1 print(wq.s0)
2 print(wq.pct_nonzero)

```

```

1 40.0
2 49.382716049382715

```

## \*\*Code\*\* : GT, contiguïté reine, pysal, San Diego

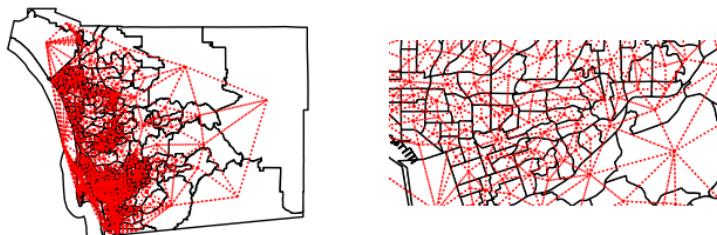
```

1 # read the data from gpkg file
2 san_diego_tracts = geopandas.read_file("sandiego_tracts.gpkg")
3 # build a queen contiguity relations between tracts
4 w_queen = weights.contiguity.Queen.from_dataframe(san_diego_tracts)
5 # Plot tract geography
6 f, axs = plt.subplots(1, 2, figsize=(8, 4))
7 for i in range(2):
8     ax = san_diego_tracts.plot(
9         edgecolor="k", facecolor="w", ax=axs[i]
10    )
11    # Plot graph connections
12    w_queen.plot(
13        san_diego_tracts,
14        ax=axs[i],
15        edge_kws=dict(color="r", linestyle=":", linewidth=1),
16        node_kws=dict(marker=""),
17    )
18    # Remove the axis
19    axs[i].set_axis_off()
20 # Make a zoom in for the 2nd subplot
21 axs[1].axis([-13040000, -13020000, 3850000, 3860000]);

```

## \*\*Code\*\* : GT, contiguïté reine, pysal, San Diego (suite)

Extrait de [Rey et al., 2020]



- Ce graphe est bcp. moins dense que le précédent.

```
1 print(w_queen.n)
2 print(w_queen.s0)
3 print(w_queen.pct_nonzero)
4 print(w_queen.s0/w_queen.n**2)
```

```
1 628
2 4016.0
3 1.018296888311899
4 0.01018296888311899
```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

61 / 534

## Illustration avec des données issues d'une SF

- La majorité des cas concernent les GT comme précédemment. Nous voyons néanmoins un cas où les données en input proviennent d'une SF. Il s'agit de données sur la densité de population de São Paulo.

```
1 # read the geotiff raster file format
2 sao_paulo = xarray.open_rasterio("ghsl_sao_paulo.tif")
3 # build the weight matrix based on the Queen contiguity
4 w_sao_paulo = weights.contiguity.Queen.from_xarray(sao_paulo)
5 # print an excerpt of the adjacency matrix in the sparse format
6 print(w_sao_paulo.sparse[:3])
```

```
1 (0, 1)      1
2 (0, 20)     1
3 (0, 21)     1
4 (1, 0)      1
5 (1, 2)      1
6 (1, 20)     1
7 (1, 21)     1
8 (1, 22)     1
9 (2, 1)      1
10 (2, 3)     1
11 (2, 21)    1
12 (2, 22)    1
13 (2, 23)    1
```

J. Ah-Pine

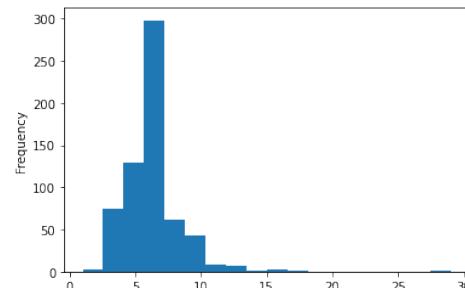
Masterclass IHEDD-CERDI-GDN GeoAI 24

63 / 534

## \*\*Code\*\* : GT, contiguïté reine, pysal, San Diego (suite)

- La dist. des degrés est aussi très différente du *toy example* précédent.

```
1 # convert the degrees into a pandas.Series...
2 s = pandas.Series(w_queen.cardinalities)
3 # ... in order to generate a histogram easily
4 s.plot.hist(bins=s.unique().shape[0]);
```



Extrait de [Rey et al., 2020]

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

62 / 534

## Poids binaires issus de mesures de distances et des k plus proches voisins

- Ici nous définissons le voisinage comme une fonction de la distance séparant deux objets.
- En supposons que les objets géographiques  $X_i$  soient représentés par des *polygon*, on se ramène comme précédemment à leur centroïde  $c_i$ .
- Ensuite, une **mesure de distance** entre centroïdes est appliquée pour toute paire de *geographic objects* et on détermine la matrice carrée des distances.
- Puis étant donné un paramètre  $k$ , l'ensemble des  $k$  plus proches voisins de chq. objet basé sur cette matrice de distances est retenu comme voisinage de chq. objet.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

64 / 534

## \*\*Code\*\* : GT, k-nn, pysal, San Diego

- On illustre le propos par les données de San Diego où :

- Le nb. de plus proches voisins  $k$  est fixé à 4.
- Chq. district (polygone) est représenté par son centroïde.
- La distance Euclidienne entre centroïde est utilisée.

```

1 # determine the knn graph from the GT with k=4
2 wk4 = weights.distance.KNN.from_dataframe(san_diego_tracts, k=4)
3 # islands are objects with no neighbor but knn does not have this issue
4 print(wk4.islands)
5 # print an excerpt of the neighbors
6 [print(wk4.neighbors[i]) for i in range(4)]
7 # every object has 4 neighbors
8 print(wk4.histogram)

```

```

1 []
2 [548, 383, 384, 386]
3 [386, 83, 384, 462]
4 [464, 554, 228, 89]
5 [312, 150, 237, 232]
6 [(4, 628)]

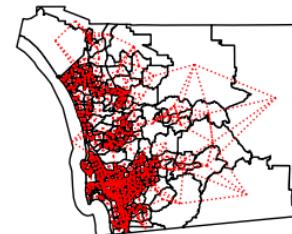
```

## Poids non négatifs issus de mesures de distances et des k plus proches voisins

- L'approche précédente encode de façon binaire le voisinage. Nous voyons maintenant comment appliquer des **fcts noyaux** utilisées en stat. spat. afin de donner des **poids compris entre 0 et 1**.
- Le centroïde  $c_i$  de l'objet  $X_i$  est à nouveau utilisé comme représentant et une mesure de distance  $D$  entre centroïde est calculée pour toutes les paires de centroïdes.
- La matrice carrée de distance est ensuite transformée par une **fonction noyau** qui généralise l'information géographique au travers d'une **mesure de proximité** dépendant de la distance initiale.
- La lib. `pysal` permet d'utiliser plus. types de fcts noyaux et également plusieurs types de sélection de voisinage parmi les suivants (cf plus loin) :
  - Le graphe des  $k$  plus proches voisins.
  - Le graphe des voisins dans un rayon donné.
  - Le graphe des voisins issus d'une variable qualitative.

## \*\*Code\*\* : GT, k-nn, pysal, San Diego (suite)

- Nous pouvons vérifier visuellement le graphe obtenu.



- Certains objets sont connectés à plus de 4 voisins. Pourquoi ?

## Poids non négatifs issus de mesures de distances et des k plus proches voisins (suite)

- La fct. noyau pondère la proximité/similarité spatiale entre centroïdes à partir de la mesure de distance. Chq. type de fct. noyau fait décroître différemment la similarité vis à vis de la distance.
- Les fcts noyaux emploient le concept de **bande** qui peut être interprétée comme étant une valeur de distance à partir de laquelle la fct. noyau est appliquée.
- Ci-dessous, nous faisons une digression sur les fcts noyaux utilisées dans le cadre de l'**estimation non paramétrique de densité** et qui sont donc utilisées en stat. spatiales.

## Backgrd : estimation non paramétrique de densité

- On commence par le cas 1D. Supposons que nous ayons n observations  $x_1, \dots, x_n$  qui soient les réalisations de n va. réelles  $\mathcal{X}_1, \dots, \mathcal{X}_n$  i.i.d. selon la va. réelle mère  $\mathcal{X}$  de densité notée  $f_{\mathcal{X}}(x)$ .
- $f_{\mathcal{X}}$  est supposée inconnue et le pb. consiste à déterminer une estimation  $\hat{f}_{\mathcal{X}}$  de  $f_{\mathcal{X}}$  à partir de  $x_1, \dots, x_n$ .
- Nous nous plaçons dans un cadre non paramétrique, c'est à dire que nous ne faisons pas l'hypothèse que  $f$  est un élément d'une famille paramétrique de fonctions. Autrement dit, nous ne supposons pas que les  $x_1, \dots, x_n$  sont échantillonés selon une loi particulière (loi normale par ex.) et qu'il suffirait de déterminer les paramètres de cette loi (espérance et écart-type par ex.).

## Backgrd : estimation non paramétrique de densité (suite)

- Si  $f_{\mathcal{X}}$  est la densité, par définition, nous avons :

$$f_{\mathcal{X}}(x) = \lim_{h \rightarrow 0} \frac{1}{2h} P(x - h < \mathcal{X} < x + h)$$

- Par conséquent, on peut définir l'**estimateur naïf** comme suit :

$$\hat{f}_{\mathcal{X}}(x) = \frac{1}{2h} \frac{\text{Nb. de } x_i \text{ dans } [x - h, x + h]}{n}$$

- Celui-ci peut être exprimé par une **fct. de poids uniforme** :

$$w(x) = \begin{cases} \frac{1}{2} & \text{si } -1 < x < 1 \\ 0 & \text{sinon} \end{cases}$$

- L'**estimateur naïf** peut alors être exprimé de façon équivalente par :

$$\hat{f}_{\mathcal{X}}(x) = \frac{1}{nh} \sum_{i=1}^n w\left(\frac{x - x_i}{h}\right)$$

## Backgrd : estimation non paramétrique de densité (suite)

- L'**histogramme** est l'estimateur de densité le plus simple [Silverman, 2018].
- Les *bins* sont des **intervalles** que l'on formalise comme suit :

$$[x_0 + mh, x_0 + (m + 1)h]$$

où  $x_0$  est une origine,  $m$  un entier relatif,  $h$  la largeur du *bin* (bande).

- On a alors :

$$\hat{f}_{\mathcal{X}}(x) = \frac{\text{Nb. de } x_i \text{ dans le même bin que } x}{nh}$$

- Limites de l'histogramme :
  - Pas différentiable.
  - Comment choisir  $x_0$  ?
  - Extension au cas multidimensionnel ?

## Backgrd : estimation non paramétrique de densité (suite)

- L'estimateur peut être construit à l'aide de rectangles de largeur  $2h$  et de hauteur  $\frac{1}{2nh}$  centrés en chq.  $x_i$ . Puis, étant donné un  $x$  on somme les rectangles recouvrant celui-ci.
- Cet estimateur naïf a aussi des limites : il est constant par morceaux donc non différentiable en tous les  $x$  présentant un saut et partout ailleurs, la dérivée est nulle.
- Ces limites proviennent de la fct. de poids  $w$  qui est constante sur un intervalle de largeur  $2h$  et donc non différentiable en ces bornes.
- L'idée est alors de remplacer  $w$  par une **fct. de poids plus générale** que l'on dénotera par  $K$  pour **kernel (noyau)**. Cette fct. continue définit un système de pondération et telle une fct. de densité, ses valeurs doivent être non négatives et son intégrale doit sommer à 1 :

$$\int_{-\infty}^{+\infty} K(x) dx = 1$$

## Backgrd : estimation non paramétrique de densité (suite)

- L'utilisation de K conduit à la déf. de l'**estimateur à noyau** :

$$\hat{f}_X(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

- Dans ce contexte, h est appelé paramètre de **lissage**. Cet (hyper)paramètre a un impact sur l'estimateur obtenu. S'il est trop petit, l'estimateur est très (trop) variable, s'il est trop grand, il gomme bcp (trop) les disparités existantes entre différents sous-domaines.

- Noyau triangulaire** :

$$K(t) = \begin{cases} 1 - |t| & \text{si } -1 < t < 1 \text{ (c.à.d. } |t| < 1) \\ 0 & \text{sinon} \end{cases}$$

- Noyau Gaussien** :

$$K(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}t^2\right)$$

## Poids non négatifs issus de distances et de fcts noyaux

- Les fcts noyaux introduites dans le cadre de l'estim. de densité sont les mêmes que celles employées en stats. spat. et permettent ainsi de transformer une mesure de distance en une mesure de proximité spat., cette dernière étant inversement proportionnelle à la première.
- La lib. pysal implémente plus. fcts noyaux. Il existe 2 types d'hyperparamètres :
  - La **fct. noyau** qui donne le **type de dépendance fonctionnelle** entre la dist. et la prox. spat. (qui sera encodée dans le graphe de voisinage).
  - La **bande** indique la **val. de la dist. à partir de laquelle la fct. noyau agit**. Pour des dist. supérieurs à la bande, la prox. spat. est nulle.  
Rq : la bande en stat. spat. correspond au paramètre de lissage en estim. de densité.
- La fct. noyau utilisée par défaut est le noyau triangulaire avec une bande fixée à la valeur maximale parmi les valeurs de distance de chaque objet à son 2ème plus proche voisin.

## Backgrd : estimation non paramétrique de densité (suite)

Extrait de [Silverman, 2018]

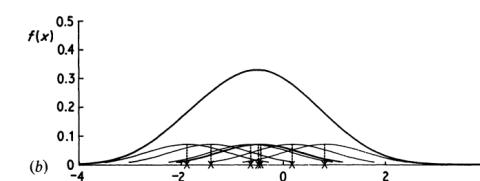
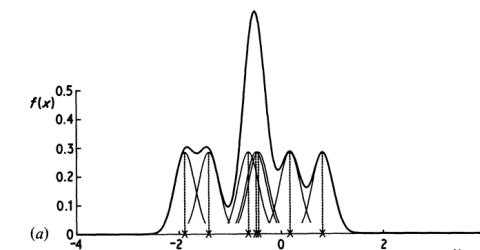


Fig. 2.5 Kernel estimates showing individual kernels. Window widths: (a) 0.2; (b) 0.8.

## \*\*Code\*\* : Fcts noyaux et pysal

```

1 # compute the kernel weights of centroids of the 3 x 3 grid
2 w_kernel = weights.distance.Kernel.from_dataframe(gdf)
3 # centroids of each cell
4 print(w_kernel.data[0:3,:])
5 # default kernel
6 print(w_kernel.function)
7 # Show the first three values of bandwidths
8 print(w_kernel.bandwidth[0:3])
9 # neighbors of each node
10 print(w_kernel.neighbors)
11 # kernel weights between each node (key of the dict) and its neighbors
12 print(w_kernel.weights)

```

```

1 [[0.5 0.5]
2 [1.5 0.5]
3 [2.5 0.5]]
4 triangular
5 [[1.0000001]
6 [1.0000001]
7 [1.0000001]]
8 {0: [0, 1, 3], 1: [0, 1, 2, 4], 2: [1, 2, 5], 3: [0, 3, 4, 6], 4: [1, 3, 4, 5, 7], 5:
9 [2, 4, 5, 8], 6: [3, 6, 7], 7: [4, 6, 7, 8], 8: [5, 7, 8]}
9 {0: [1.0, 9.9999900663795e-08, 9.9999900663795e-08], 1: [9.9999900663795e-08, 1.0,
9.9999900663795e-08, 9.9999900663795e-08], 2: [9.9999900663795e-08, 1.0,
9.9999900663795e-08], 3: [9.9999900663795e-08, 1.0, 9.9999900663795e-08,
9.9999900663795e-08], 4: [9.9999900663795e-08, 9.9999900663795e-08, 1.0,
9.9999900663795e-08, 9.9999900663795e-08], 5: [9.9999900663795e-08, 1.0,
9.9999900663795e-08, 9.9999900663795e-08], 6: [9.9999900663795e-08, 1.0,
9.9999900663795e-08, 9.9999900663795e-08], 7: [9.9999900663795e-08, 9.9999900663795e-08, 1.0,
9.9999900663795e-08], 8: [9.9999900663795e-08, 9.9999900663795e-08, 1.0]}

```

## Bande adaptative basée sur les k plus proches voisins

- La bande est également par défaut la même pour tous les objets. Cette stratégie est simple mais pas toujours adéquate. En reprenant l'exemple de l'estim. de densité, si une région du domaine est moins représentée au sein de l'échantillon alors il serait propice d'élargir la bande afin de lisser davantage.
- Une stratégie de détermination adaptative de la bande pour chaque objet est donc intéressante. L'utilisation des k plus proches voisins est à nouveau exploitée dans ce cas : la bande d'un objet est égale à la distance le séparant de son k plus proche voisin.

## \*\*Code\*\* : Bande adaptative et pysal (suite)

- Pour avoir une bande adaptative, il faut utiliser le paramétrage `fixed=False`. Puis, le paramètre `k` permet de fixer la bande pour chq. sommet en fonction de sa distance avec son k-ème plus proche voisin.

```

1 # Build weights with adaptive bandwidth
2 w_adaptive = weights.distance.Kernel.from_dataframe(
3     sub_30, fixed=False, k=15
4 )
5 # Print first five bandwidth values
6 w_adaptive.bandwidth[:5]

```

```

1 array([[7065.74020822],
2        [3577.22591841],
3        [2989.74807871],
4        [2891.46196945],
5        [3965.08354232]])

```

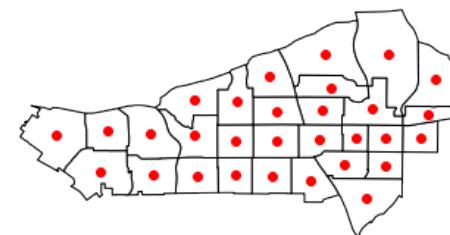
## \*\*Code\*\* : Bande adaptative et pysal

```

1 # Create subset of tracts
2 sub_30 = san_diego_tracts.query("sub_30 == True")
3 # Plot polygons
4 ax = sub_30.plot(facecolor="w", edgecolor="k")
5 # Create and plot centroids
6 sub_30.head(30).centroid.plot(color="r", ax=ax)
7 # Remove axis
8 ax.set_axis_off();

```

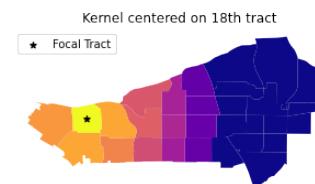
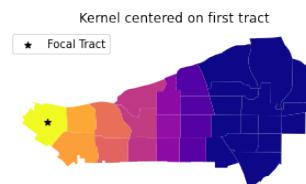
Extrait de [Rey et al., 2020]



- Ici, les centroïdes ne sont pas espacés de façon totalement régulière.

## \*\*Code\*\* : Bande adaptative et pysal (suite)

Extrait de [Rey et al., 2020]



- Le code précédent permet de générer les graphiques ci-dessus. Ceux-ci permettent de visualiser sur deux objets/noeuds distincts (1 et 18) leurs distributions des proximités spatiales/spatial weights et ainsi la sortie de la fonction noyau.
- La bande étant distincte entre ces deux noeuds, ils n'ont pas la même couleur respective (dc la même valeur) entre 1 et 18 et entre 18 et 1.

## Remarque sur la taille du voisinage

- Les SW basés sur la contiguïté donne des matrices d'adjacence binaires sparses en comparaison des SW basés sur les distances et les fcts noyau, lorsque les bandes utilisées dans ces dernières sont basés sur un voisinage relativement grand.
- Dans ce cas, les SW utilisant les fcts noyaux requièrent davantage de ressources machine à la fois en mémoire (matrice d'adjacence totalement dense au pire des cas) et en temps de traitement (calcul de la proximité spatiale pour toutes les paires possibles).
- Toutefois pour satisfaire le principe du géographe Tobler, les **chooses proches sont susceptibles d'être plus liées que les choses éloignées**, le voisinage de tout objet sera limité.

## \*\*Code\*\* : Distance based SW et pysal

```

1 # binary distance based spatial weight
2 w_bdb = weights.distance.DistanceBand.from_dataframe(gdf, 1.5, binary=True)
3 print(w_bdb.neighbors[0])
4 print(w_bdb.weights[0])
5 # non negative distance based spatial weight
6 w_hy = weights.distance.DistanceBand.from_dataframe(gdf, 1.5, binary=False)
7 print(w_hy.neighbors[0])
8 print(w_hy.weights[0])

```

```

1 [1, 3, 4]
2 [1.0, 1.0, 1.0]
3 [1, 3, 4]
4 [1.0, 1.0, 0.7071067811865475]

```

## Poids issus de dist. limitées à un seuil (rayon d'un cercle)

- On peut également considérer un cercle centré en chaque objet et de rayon donné et garder dans le voisinage de cet objet, ceux qui se situent dans le cercle.
- Deux types de pondération peuvent alors être prises en compte :
  - ▶ Poids binaires : l'objet est dans le cercle (1) ou pas (0).
  - ▶ Poids non négatifs : pour les objets dans le cercle, la proximité spatiale est donnée par une mesure dépendant de l'inverse de la distance ; pour les objets hors du cercle cette proximité spatiale est fixée à 0.
- Rq : dans le cas d'objets géographiques comme les pays, il est plus pertinent de mesurer des géodésiques que des distances Euclidiennes sur un planisphère. Autrement dit, il faut tenir compte de la courbure de la surface de la terre afin de déterminer de façon plus juste les distances entre objets géoréférencés. Il est bien sûr possible de faire ce type de traitement avec pysal mais ne n'aborderons pas ce point.

## Poids binaires issus d'une variable qualitative nominale

- Dans ce cas on considère que l'**appartenance** d'un sous-ensemble d'objets géographiques à **un même groupe** définit une relation de proximité "spatiale" binaire. Les objets n'appartenant pas au même groupe sont donc non connectés.
- Si chq. obj. appartient à un seul groupe, la relation d'appartenance d'un obj. à un groupe induit une partition des objets et la matrice de SW peut être réorganisée de sorte avoir des **blocs le long de la diagonale**. Chq. bloc représente un groupe et les membres de ce groupe.
- Dans les GT, un **attribut de type qualitatif nominal** est alors interprété comme une liste de groupes. On peut donc inférer de var. catégorielles ce type de SW.
- Dans un cadre purement géographique, nous illustrons ce type de SW par l'appartenance d'un *district* à un *county* dans le jeu de données de San Diego. Dans ce cas, les *district* appartenant au même *county* sont aussi proche spatialement. Autre exemple : l'appartenance des

## \*\*Code\*\* : Block weight, pysal, San Diego

```

1 san_diego_tracts[["GEOID", "state", "county", "tract"]].head()
2 # NOTE: since this is a large dataset, it might take a while to process
3 w_b1=weights.util.block_weights(san_diego_tracts["county"].values,ids=san_diego_tracts
4 ["GEOID"].values,)
"06073018601" in w_b1["06073018300"]

```

1 True

## Rappel du Sommaire

1 Introduction et motivations générales

2 Concepts de base et outils Python associés

3 Analyse de la dépendance spatiale

4 Méthodes de régression spatiale

5 Méthodes d'ensemble en ML et extensions spatiales

6 Eléments de géostatistique

7 Méthodes de DL et extensions spatiales

## SW hybrides

- On peut combiner plusieurs SW ! Dans ce cas la proximité spatiale dépend donc de plusieurs critères.
- Dans le cas des SW binaires, les 1 et 0 peuvent être interprétés comme indicateur de l'appartenance d'une paire à un ensemble et dans ce cas des opérations classiques de la théorie des ensembles permettent de combiner les SW et SG.

## Introduction

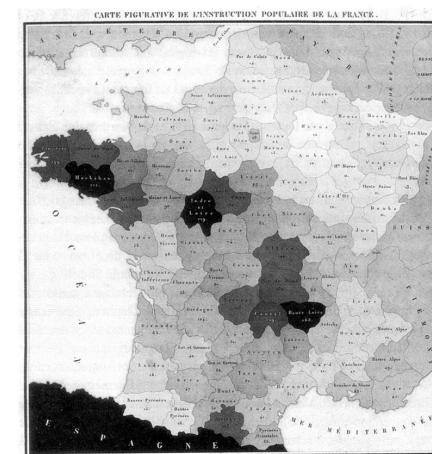
- Nous commençons par représenter visuellement la dépendance spatiale au travers des **cartes choroplèthes**. Il s'agit de cartes thématiques où les régions sont colorées ou remplies d'un motif qui montrent une mesure statistique, tels la densité de population ou le revenu par habitant, ....
  - Ce type de carte facilite la comparaison d'une mesure statistique d'une région à l'autre ou montre la variabilité de celle-ci pour une région donnée.
  - Dans le cas de variables quantitatives continues, pour des raisons techniques initialement et cognitives par la suite, on ne représente pas chaque valeur observée de la variable par un motif ou une couleur spécifique. En pratique, on regroupe les valeurs dans des intervalles (pas plus de 12) et chaque intervalle est alors associé à un motif ou une couleur.

## Principe

- Pour qu'une carte choroplète soit efficace, il est important que la visualisation rende compte du "message" principal que l'on souhaite véhiculer.
  - Dans le cas d'une variable quantitative, pour que la charge cognitive ne soit pas lourde, il faut donc discréteriser la variable et dans ce cas considérer les étapes suivantes :
    - ▶ Choisir un nb. d'intervalles/groupes/classes plus petit que n le nb. d'observations.
    - ▶ Choisir un algorithme de discréétisation qui permet de transformer la variable quantitative en variable qualitative.
    - ▶ Choisir le motif ou la couleur de chaque intervalle afin de refléter le "message" à transmettre.
  - Dans ce qui suit, nous voyons en particulier des méthodes de discréétisation.

## Introduction (suite)

- Voici l'une des 1ères cartes choroplèthes de l'histoire par Charles Dupin en 1826. Elle représente le taux de scolarisation des garçons dans chaque département de France [Wikipedia, 2022] :



# Discretisation d'une variable quantitative

- Cette tâche fait partie des prétraitements que l'on rencontre en *data science*. Elle peut être vue comme un problème de catégorisation en k classes d'un ens. d'individus décrits dans un espace 1D.
  - Dans un espace 1D, il existe plusieurs façons d'aborder le pb. ce qui le met malgré tout dans une catégorie à part. Ces diverses techniques vont de méthodes très simples à plus sophistiquées.
  - Soit une variable aléatoire dénotée  $\mathcal{X}$  et soient  $x_1, \dots, x_n$  un échantillon d'observations de celle-ci. Formellement, le pb. consiste à déterminer les bornes des k intervalles  $C_j = ]c_j, c_{j+1}]$ ,  $\forall j = 1, \dots, k$  :
$$c_j < x_i \leq c_{j+1}, \forall x_i \in C_j$$
  - Rq. :  $C_k = ]c_k, c_{k+1}]$  et dans ce cas  $c_{k+1}$  est uniquement la borne supérieure du dernier intervalle.

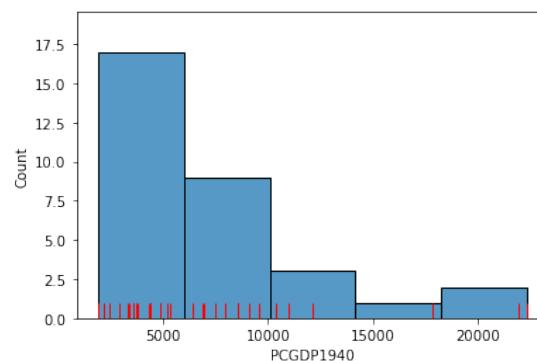
$$c_j < x_i \leq c_{j+1}, \forall x_i \in C_j$$

## Discréétisation d'une variable quantitative (suite)

- On appellera également  $C_j$ , la classe  $C_j$ . Ainsi,  $\cup_{j=1}^k C_j$  est un partitionnement du segment réel dans lequel  $\mathcal{X}$  prend les valeurs. Chaque observation  $x_i, \forall i = 1, \dots, n$  appartient alors à une et une seule classe.
- Avant de choisir une méthode de discréétisation, il est recommandé de procéder aux prétraitements classiques tels que la détection et suppression de données aberrantes. Il est également préférable de tenir compte de la distribution des données  $x_1, \dots, x_n$  que l'on pourra apprécier à l'aide d'un histogramme.

## \*\*Code\*\* : histogramme, seaborn, Mexique (suite)

Extrait de [Rey et al., 2020]



- La distribution est positivement asymétrique ce qui est généralement le cas dans les études régionales des revenus. Autrement dit, la queue de distribution à droite est plus longue. Une caractéristique de ce type d'asymétrie est que la médiane est en dessous de la moyenne.

## \*\*Code\*\* : histogramme, seaborn, Mexique

- Nous illustrons ceci à l'aide de l'exemple du PIB par habitant (*Per Capita Gross Domestic Product*) des dif. régions du Mexique en 1940.

```

1 import geopandas
2 mx = geopandas.read_file("mexicojoin.shp")#comes with other mexicojoin files
3 mx[["NAME", "PCGDP1940"]].head()
4 # Plot histogram
5 import seaborn
6 import matplotlib.pyplot as plt
7 ax = seaborn.histplot(mx["PCGDP1940"], bins=5)
8 # Add rug on horizontal axis
9 seaborn.rugplot(mx["PCGDP1940"], height=0.05, color="red", ax=ax);
10 plt.show()

```

	NAME	PCGDP1940
0	Baja California Norte	22361.0
1	Baja California Sur	9573.0
2	Nayarit	4836.0
3	Jalisco	5309.0
4	Aguascalientes	10384.0

## \*\*Code\*\* : histogramme, seaborn, Mexique (suite)

- La création d'un histogramme correspond en soi à une discréétisation d'une var. quanti. Etant donné le nb. de classes/*bins*  $k = 5$ , celles-ci sont définies de sorte à ce que leur longueur soit égales.
- En pratique, la procédure met en oeuvre les étapes suivantes :
  - les données sont triées dans l'ordre croissant :

$$x_1, x_2, \dots, x_n \rightarrow x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n)}$$

- où  $x_{(1)} = \min(x_1, \dots, x_n)$  et  $x_{(n)} = \max(x_1, \dots, x_n)$ .
- la discréétisation revient alors à sélectionner les  $k$  valeurs  $c_1, \dots, c_k$  du segment  $[x_{(1)}, x_{(n)}]$  telles que  $c_{j+1} - c_j = \frac{x_{(n)} - x_{(1)}}{k}$ ,  $\forall j = 1, \dots, k$  avec  $c_1 = x_{(1)}$  et  $c_{k+1} = x_{(n)}$ .

## \*\*Code\*\* : histogramme, seaborn, Mexique (suite)

```

1 counts, bins, patches = ax.hist(mx["PCGDP1940"], bins=5)
2 print(counts)
3 print(bins)
4 print("---")
5 print(bins[1]-bins[0])
6 print(bins[2]-bins[1])
7 print(bins[3]-bins[2])
8 print(bins[4]-bins[3])
9 print(bins[5]-bins[4])
10 print("---")
11 print((bins[5]-bins[0])/5)

```

```

1 [17.  9.  3.  1.  2.]
2 [ 1892.    5985.8  10079.6  14173.4  18267.2  22361. ]
3 ---
4 4093.8
5 4093.8
6 4093.800000000001
7 4093.799999999993
8 4093.799999999993
9 ---
10 4093.8

```

## \*\*Code\*\* : histogramme, seaborn, Mexique (suite)

```

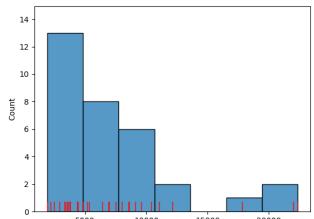
1 # Plot histogram using seaborn with Freedman-Diaconis rule
2 ax = seaborn.histplot(mx["PCGDP1940"], bins='auto')
3 # Add rug on horizontal axis
4 seaborn.rugplot(mx["PCGDP1940"], height=0.05, color="red", ax=ax);
5 plt.show()
6 counts, bins, patches = ax.hist(mx["PCGDP1940"], bins='auto')
7 print(counts)
8 print(bins)
9 print("---")
10 k_fd=len(bins)-1
11 print(k_fd)
12 print("---")
13 h_fd=bins[1]-bins[0]
14 print(h_fd)
15 print("---")
16 print((mx["PCGDP1940"].max()-mx["PCGDP1940"].min())/h_fd)

```

```

1 [13.  8.  6.  2.  0.  1.  2.]
2 [ 1892.    4816.14285714
   7740.28571429  10664.42857143
   13588.57142857 16512.71428571
   19436.85714286 22361.      ]
3 ---
4 7
5 ---
6 2924.142857142857
7 ---
8 7.000000000000001

```



## Discr. d'une var. quanti. : Freedman-Diaconis rule

- Le nb. d'intervalles était fixé à  $k = 5$  précédemment. Il existe une méthode robuste aux données aberrantes afin de déterminer automatiquement l'amplitude des intervalles, il s'agit de la règle de Freedman-Diaconis [Freedman and Diaconis, 1981].
- Cette règle utilise l'intervalle InterQuartile (IQR). Rappelons qu'un percentile en stat. desc. est une des  $p = 1, \dots, 99$  valeurs qui divisent une distribution de données en 100 parts égales de sorte que le  $p$ -ième percentile soit la valeur qui sépare d'un côté  $p\%$  des valeurs et de l'autre  $100 - p\%$  des autres valeurs.
- Le 1er et le 3ème quartile sont resp. les percentiles pour  $p = 25$  et  $p = 75$  dénotés  $p_{25}$  et  $p_{75}$ . Nous avons  $IQR = p_{75} - p_{25}$ . L'amplitude de Freedman-Diaconis des intervalles est donnée par :

$$h_{fd} = 2IQRn^{-1/3}$$

## Disct. d'une var. quant. avec mapclassify

- Nous allons approfondir la question de la discréttisation d'une variable quantitative continue dans le contexte des cartes choroplèthes à l'aide de la librairie `mapclassify`.
- Il existe en fait bcp. de méthodes de discréttisation et cet outil nous permet de les mettre en oeuvre efficacement. Dans ce qui suit :
  - ▶ Nous verrons (ou reverrons) les principes des méthodes suivantes :
    - Intervalles uniformes
    - Quantiles
    - Mean-standard deviation
    - Maximum breaks
    - Head-Tail breaks
    - Jenks-Caspall
    - Fisher-Jenks
    - Maxp
  - ▶ Nous verrons des graphiques comparant les résultats de ces modèles.
  - ▶ Nous verrons une méthode de sélection de modèles.

## Discr. d'une var. quanti. : Intervalles uniformes et définis par percentiles

- Intervalles uniformes (vus précédemment) :
  - ▶ Chq. intervalle défini à la même amplitude.
  - ▶ Il faut spécifier le nb. de classes k (ou utiliser la règle de Freedman-Diaconis).
- Intervalles définis par percentiles (ou quantiles) :
  - ▶ Les intervalles sont d'amplitudes différentes.
  - ▶ Les intervalles sont définis de sorte à ce que le nb. d'éléments dans chacun d'eux est approximativement égale (d'où l'utilisation de percentiles).
  - ▶ Il faut spécifier le nb. de classes k.

## \*\*Code\*\* : discréétisation, `mapclassify`, Mexique

```

1 import mapclassify
2 ei5 = mapclassify.EqualInterval(mx["PCGDP1940"], k=5)
3 print(ei5)
4 q5 = mapclassify.Quantiles(mx.PCGDP1940, k=5)
5 print(q5)
6 print(q5.bins[1:] - q5.bins[:-1])

```

```

1 EqualInterval
2
3   Interval          Count
4 -----
5 [ 1892.00,  5985.80] |    17
6 ( 5985.80, 10079.60] |     9
7 (10079.60, 14173.40] |     3
8 (14173.40, 18267.20] |     1
9 (18267.20, 22361.00] |     2
10 Quantiles
11
12   Interval          Count
13 -----
14 [ 1892.00,   3576.20] |     7
15 ( 3576.20,   4582.80] |     6
16 ( 4582.80,   6925.20] |     6
17 ( 6925.20,   9473.00] |     6
18 ( 9473.00, 22361.00] |     7
19 [ 1006.6  2342.4  2547.8 12888. ]

```

## Discr. d'une var. quanti. : Mean-standard deviation

- *Mean-standard deviation* (adaptée aux variables normales)
  - ▶ Utilise les moments empiriques d'ordre 1 et 2 :
$$\hat{\mu} = \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad ; \quad \hat{\sigma}_x = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$
- ▶ Les bornes des intervalles sont définies par le biais d'une distance vis-à-vis de  $\hat{\mu}$ . Cette distance est un multiple de  $\hat{\sigma}_x$ .
- ▶ Pour k = 5 par ex. (valeur par défaut), les bornes sont définies par :
  - $c_1 = -\infty$
  - $c_2 = \hat{\mu} - 2\hat{\sigma}_x$
  - $c_3 = \hat{\mu} - \hat{\sigma}_x$
  - $c_4 = \hat{\mu} + \hat{\sigma}_x$
  - $c_5 = \hat{\mu} + 2\hat{\sigma}_x$
  - $c_6 = +\infty$
- ▶ Mis à part  $C_1 = ]-\infty, c_2]$  et  $C_5 = ]c_5, +\infty[$ , les autres intervalles sont d'amplitudes toutes égales à  $\hat{\sigma}_x$ .
- ▶ Commande : `mapclassify.StdMean`

## Discr. d'une var. quanti. : Maximum breaks

- *Maximum breaks* :
  - ▶ Trie les données dans l'ordre croissant puis utilise les écarts entre données consécutives pour définir des valeurs de séparations (*break points*) qui engendrent des intervalles.
  - ▶ Soit les  $n - 1$  écarts entre deux données triées consécutives :  $(x_{(i+1)} - x_{(i)})_{i=1,\dots,n-1}$ . Soit k le nb. d'intervalles souhaité.
    - La méthode sélectionne les  $k - 1$  plus grands écarts.
    - Si  $x_{(i+1)} - x_{(i)}$  est un écart sélectionné, la méthode définit une borne d'un intervalle de la discréet. (c.à.d. un *break point*),  $c_j$ , en prenant le milieu de l'intervalle défini par les deux valeurs successives ayant produit l'écart :
$$c_j = x_{(i)} + \frac{x_{(i+1)} - x_{(i)}}{2}$$
  - ▶ On sépare les obs. lorsqu'il y a un écart très grand entre deux valeurs ordonées successives. Cela permet de s'assurer que les obs. dans deux classes successives sont très éloignées entre elles.
  - ▶ Il faut spécifier k. Commande : `mapclassify.MaximumBreaks`

## Discr. d'une var. quanti. : Head-tail breaks

- **Head-tail breaks** (pour distrib. à queues épaisses) [Jiang, 2013] :
  - ▶ Algo. de partitionnement binaire (*head versus tail*) récursif :
    - Tri des données dans l'ordre décroissant. La classe initiale comprend toutes les observations.
    - A chq. itér., la méthode *split* la classe en deux, sa *head* et sa *tail*, en prenant comme *break point* la moyenne des obs. dans la classe. Les valeurs au dessus de la moy. vont dans la nouvelle *head*.
    - Les *split* se poursuivent récursivement sur chq. nouvelle *head* jusqu'à ce qu'il y ait un équilibre entre la nouvelle *tail* ("petites" valeurs) et la nouvelle *head* ("grandes" valeurs), ou alors lorsque le cardinal de la nouvelle *head* est inférieur à  $0.4n$  (condition d'arrêt par défaut).
  - ▶ L'approche donne une classif. hiérarchique qui respecte la distrib. initiale des données lorsque celles-ci suivent une loi *heavy-tailed*.
  - ▶ Commande : `mapclassify.HeadTailBreaks`.
  - ▶ Rq. : dans la dénomination classique en probabilité, la queue de distribution correspond aux grandes valeurs (ici c'est la *head*).

## Discr. d'une var. quanti. : Fisher Jenks

- **Fisher Jenks** :
  - ▶ Résoud optimalement le pb. de la min. de la somme des écarts à la moyenne (locale) en valeur absolue en utilisant la programmation dynamique.
  - ▶ Il faut spécifier k. Commande : `mapclassify.FisherJenks`.

```
1 numpy.random.seed(12345)
2 fjs = mapclassify.FisherJenks(mx["PCGDP1940"], k=5)
3 print(fjs)
```

FisherJenks	
Interval	Count
[ 1892.00, 5309.00]	17
( 5309.00, 9073.00]	8
( 9073.00, 12132.00]	4
(12132.00, 17816.00]	1
(17816.00, 22361.00]	2

## Discr. d'une var. quanti. : Jenks Caspall

- **Jenks Caspall** [Jenks and Caspall, 1971] :
  - ▶ Algo. de partitionnement itératif visant à minimiser la somme des écarts à la moyenne (locale) en valeur absolue :
    - Initialisation de k classes basées par exemple sur les quantiles.
    - Dans chq. classe, on identifie les obs. les plus proches des bornes (celle vers la borne inf. et celle vers la borne sup.) et on calcule le changement de la valeur du critère d'optim. si on transférait ces obs. dans les classes adjacentes (resp. à gauche et à droite).
    - L'obs. qui permet de réduire au maximum le critère est sélectionné, on opère le transfert et on met à jour les moyennes.
    - On répète ces mouvements de transfert jusqu'à ce qu'une condition d'arrêt d'arrêt soit vérifiée comme la convergence du critère par ex.
  - ▶ Cet algo. est une version de l'algo. de *clustering* k-means mais sur des obs. dans un espace 1D.
- Il faut spécifier k. Commande : `mapclassify.JenksCaspall`.

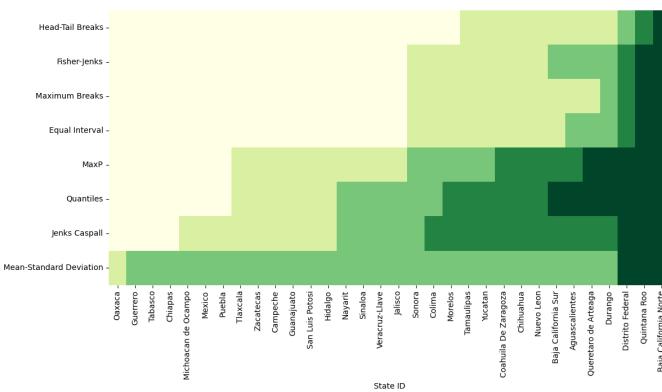
## Discr. d'une var. quanti. : Maxp

- **Maxp** :
  - ▶ Méthode adaptée d'un algo. de *clustering* introduit dans [Duque et al., 2012], qui modélise le pb. de regroupement de données surfaciques (*areal data*). Dans ce pb., des unités spatiales doivent être regroupées en satisfaisant des contraintes de contiguïté et en cherchant à minimiser un critère d'hétérogénéité à partir de variables.
  - ▶ Le pb. introduit peut être modélisé par programmation linéaire en nb. mixtes mais qui est complexe en temps de résolution. Une heuristique est également proposée pour résoudre approximativement le problème dans des temps plus raisonnables.
  - ▶ Dans notre contexte, nous sommes en 1D et la méthode de discrétt. Maxp est ainsi une adaptation de cette heuristique en cas 1D.
  - ▶ L'heuristique est similaire à l'approche de Jenks et Caspall qui considère un transfert d'une obs. à une classe adjacente si ce mouvement permet de diminuer le critère d'hétérogénéité.
- Il faut spécifier k. Commande : `mapclassify.MaxP`

## Comparaison et sélection des méthodes de discréétisation

- Plusieurs méthodes sont possibles et même pour une méthode, plusieurs valeurs de k pourraient être proposées. Comment alors comparer les modèles entre eux et choisir ?
- Ce problème est général en *data science* et concerne la sélection de modèles. Toutefois, étant donnés la nature spatiale des données, et le contexte de visualisation par carte choroplète, d'autres éléments peuvent entrer en jeu. *In fine*, il faut que la discréét. choisie procure une carte qui apporte une information pertinente, c'est à dire du sens à la représentation spatiale des couleurs qui en découle. Néanmoins, sur cet aspect, c'est souvent des connaissances expertes sur le phénomène associé aux données à l'étude qui sont nécessaires.
- Nous traiterons donc uniquement l'aspect stat. de la sél. de modèle. Dans ce cas, pour comparer les modèles, il faut (au moins) un critère. Pour le pb. de discréét., un critère courant est la moyenne des écarts en valeurs absolus aux médianes dans chq. classe/*absolute deviation around class medians* (ADCM).

## \*\*Code\*\* : Compar. des rés. des méth. avec k = 5 (suite)



- Les régions (axe horiz.) ont été ordonnées selon l'ordre croissant de la var. PCGDP1940.
- Pour les méth. (axe vert.), les intervalles sont de couleurs de + en + sombres pour les intervalles/classes de valeurs de + en + grandes.

## \*\*Code\*\* : Comp. des résultats des méthodes avec k = 5

```

1 # Append class values as a separate column
2 mx["Quantiles"] = q5.yb
3 mx["Equal Interval"] = ei5.yb
4 mx["Head-Tail Breaks"] = ht.yb
5 mx["Maximum Breaks"] = mb5.yb
6 mx["Mean-Standard Deviation"] = msd.yb
7 mx["Fisher-Jenks"] = fj5.yb
8 mx["Jenks Caspall"] = jc5.yb
9 mx["MaxP"] = mp5.yb
10 f, ax = plt.subplots(1, figsize=(9, 3))
11 seaborn.heatmap(
12     mx.set_index("NAME")
13     .sort_values("PCGDP1940")[
14         [
15             "Head-Tail Breaks",
16             "Fisher-Jenks",
17             "Maximum Breaks",
18             "Equal Interval",
19             "MaxP",
20             "Quantiles",
21             "Jenks Caspall",
22             "Mean-Standard Deviation",
23         ]
24     ],
25     .T,
26     cmap="YlGn",
27     cbar=False,
28     ax=ax,
29 )
30 ax.set_xlabel("State ID");
31 plt.show()

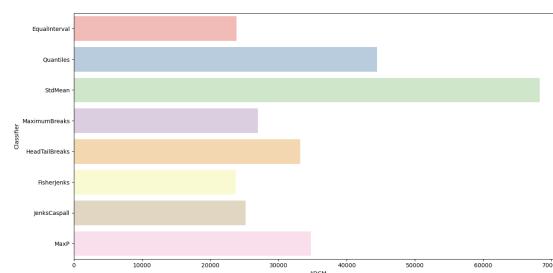
```

## \*\*Code\*\* : Sél. de mod. de discréét.

```

1 # Bunch classifier objects
2 class5 = ei5, q5, msd, mb5, ht, fj5, jc5, mp5
3 # Collect ADCM for each classifier
4 fits = numpy.array([c.adcm for c in class5])
5 import pandas
6 # Convert ADCM scores to a DataFrame
7 adcms = pandas.DataFrame(fits)
8 # Add classifier names
9 adcms["classifier"] = [c.name for c in class5]
10 # Add column names to the ADCM
11 adcms.columns = ["ADCM", "Classifier"]
12 ax = seaborn.barplot(y="Classifier", x="ADCM", data=adcms, palette="Pastel1")
13 plt.show()

```



## \*\*Code\*\* : carte choroplète

- La méthode *Fisher-Jenks* donne le meilleur critère.

```

1 ax = mx.plot(
2     column="PCGDP1940", # Data to plot
3     scheme="fisherjenks", # Classification scheme
4     cmap="YlGn", # Color palette
5     legend=True, # Add legend
6     legend_kwds={"fmt": "{:.0f}"}, # Remove decimals in legend
7 )
8 ax.set_axis_off();
9 plt.show()

```



## \*\*Code\*\* : carte choroplète (suite)

- Comme abordé plus haut, le choix des couleurs a son importance.

```

1 ax = mx.plot(
2     column="PCGDP1940", # Data to plot
3     scheme="fisherjenks", # Classification scheme
4     cmap="coolwarm", # Color palette
5     legend=True, # Add legend
6     legend_kwds={"fmt": "{:.0f}"}, # Remove decimals in legend
7 )
8 ax.set_axis_off();
9 plt.show()

```



## \*\*Code\*\* : carte choroplète (suite)

- Comme abordé plus haut, le choix des couleurs a son importance.

```

1 ax = mx.plot(
2     column="PCGDP1940", # Data to plot
3     scheme="fisherjenks", # Classification scheme
4     cmap="Set1", # Color palette
5     legend=True, # Add legend
6     legend_kwds={"fmt": "{:.0f}"}, # Remove decimals in legend
7 )
8 ax.set_axis_off();
9 plt.show()

```



## Rappel du Sommaire

### 3 Analyse de la dépendance spatiale

- Carte choroplète et discréétisation de variables quantitatives
- Autocorrélation spatiale globale
- Autocorrélation spatiale locale

## Introduction

- Selon Luc Anselin<sup>1</sup> la notion d'**autocorrélation spatiale (AS)** repose sur l'existence d'une “*functional relationship between what happens at one point in space and what happens elsewhere*” [Anselin, 1988].
- L'AS vise à mesurer comment et de combien la similarité entre observations d'une variable est en relation avec la similarité entre les localisations spatiales de ces observations.
- La notion d'AS est proche (mais non identique) de celle de **corrélation entre deux var.** qui vise à mesurer comment et de combien les valeurs observées d'une 1ère var. sont en relation avec les valeurs observées d'une 2ème var.
- La notion d'AS est aussi proche (mais également non identique) de celle d' **autocorrélation temporelle** qui vise à mesurer comment et de combien les valeurs observées d'une var. en des instants  $t$  sont en relation avec les valeurs observées de cette même var. mais en des instants passés  $t' < t$ .

1. Professeur ayant bcp contribué à l'économétrie spatiale.

## ASG par l'exemple : les données de vote du Brexit

- A l'aide d'un jeu de données réel nous allons davantage expliciter ce concept en questionnant la présence, la nature, et l'intensité de l'ASG.
- Nous allons utiliser pour cela des outils provenant de l'Analyse Exploratoire de Données Spatiales/*Exploratory Spatial Data Analysis* (ESDA) et la librairie Python associée splot, qui permet de connecter les analyses possibles avec pysal et les outils de visualisation de matplotlib.
- Nous étudions dans ce qui suit, les données officielles sur le pourcentage de votes au niveau local des “Remain” et “Leave” opinions dans le cas du referendum vis-à-vis du Brexit au Royaume-Uni en 2016.
- Deux fichiers seront utilisés dans cette perspective :
  - ▶ Un fichier .csv contenant dans une table les données électorales de 382 localités UK que l'on va lire avec pandas.
  - ▶ Un fichier .geojson contenant les données num. vectorielles donnant les frontières des localités UK que l'on va lire avec geopandas.

## Introduction (suite)

- Une autre façon de comprendre la notion d'AS est la suivante : elle représente une mesure de l'information qu'apporte les valeurs observées d'une var. en une position spatiale sur les valeurs de cette même var. mais en des localisations spatiales distinctes.
- Dans la même veine, on peut se référer à la notion de **structure spatiale totalement aléatoire (complete spatial randomness)** qui indique que la localisation d'une observation d'une var. ne donne aucune information sur les valeurs que peut prendre cette var. Ainsi, il y a AS s'il y a absence de structure spatiale totalement aléatoire.
- Lorsqu'il y a de l'AS, il est possible d'extraire des **motifs spatiaux (spatial patterns)** c.à.d. que la répartition des valeurs dans l'espace laisse apparaître des motifs indiquant que les localisations en lesquelles les évènements apparaissent sont interdépendantes.
- Nous nous intéressons d'abord à l'**autocorrélation spatiale globale (ASG)** qui considère une mesure globale de la relation entre les valeurs observées d'une var. et leurs positions spatiales.

## \*\*Code\*\* : ASG, Brexit

```

1 import pandas
2 brexit_data_path = "brexit_vote.csv"
3 ref = pandas.read_csv(brexit_data_path, index_col="Area_Code")
4 ref.info()
5
6 import geopandas
7 lads = geopandas.read_file("local_authority_districts.geojson").set_index("lad16cd")
8 lads.info()

1 <class 'pandas.core.frame.DataFrame'>
2 Index: 382 entries, E06000031 to E08000036
3 Data columns (total 20 columns):
4 #   Column           Non-Null Count Dtype
5 ---  -----
6 0   id              382 non-null    int64
7 1   Region_Code     382 non-null    object
8 2   Region          382 non-null    object
9 3   Area             382 non-null    object
10 ...
11 9   Valid_Votes    382 non-null    int64
12 10  Remain          382 non-null    int64
13 11  Leave            382 non-null    int64
14 ...
15 17  Pct_Remain     382 non-null    float64
16 18  Pct_Leave       382 non-null    float64
17 19  Pct_Rejected   382 non-null    float64
18 dtypes: float64(4), int64(13), object(3)
19 memory usage: 62.7+ KB

```

## \*\*Code\*\* : ASG, Brexit (suite)

```

1 <class 'geopandas.geodataframe.GeoDataFrame'>
2 Index: 391 entries, E06000001 to W06000023
3 Data columns (total 10 columns):
4 #   Column      Non-Null Count  Dtype  
5 ---  --          --          --    
6 0   objectid    391 non-null    int64  
7 1   lad16nm     391 non-null    object  
8 2   lad16nmw    22 non-null    object  
9 3   bng_e       391 non-null    int64  
10 4  bng_n       391 non-null    int64  
11 5  long        391 non-null    float64
12 6  lat         391 non-null    float64
13 7  st_areasha  391 non-null    float64
14 8  st_lengths  391 non-null    float64
15 9  geometry    391 non-null    geometry
16 dtypes: float64(4), geometry(1), int64(3), object(2)
17 memory usage: 41.7+ KB

```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

121 / 534

## \*\*Code\*\* : ASG, Brexit (suite)

```

1 import contextily
2 import matplotlib.pyplot as plt
3 f, ax = plt.subplots(1, figsize=(9, 9))
4 db.plot(
5     column="Pct_Leave",
6     cmap="viridis",
7     scheme="quantiles",
8     k=5,
9     edgecolor="white",
10    linewidth=0.0,
11    alpha=0.75,
12    legend=True,
13    legend_kwds={"loc": 2},
14    ax=ax,
15 )
16 contextily.add_basemap(
17     ax,
18     crs=db.crs,
19     source=contextily.providers.Stamen.TerrainBackground,
20 )
21 ax.set_axis_off()
22 plt.show()

```

- `contextily` est une lib. Python permettant de retrouver des cartes de tuiles (*tile maps*) d'internet qui peuvent ensuite être ajoutées comme fond de graphiques générés par `matplotlib`.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

123 / 534

## \*\*Code\*\* : ASG, Brexit (suite)

```

1 db = (
2     geopandas.GeoDataFrame(
3         lads.join(ref[["Pct_Leave"]]), crs=lads.crs
4     )
5     .to_crs(epsg=3857)[
6         ["objectid", "lad16nm", "Pct_Leave", "geometry"]
7     ]
8     .dropna()
9 )
10 db.info()

```

```

1 <class 'geopandas.geodataframe.GeoDataFrame'>
2 Index: 380 entries, E06000001 to W06000023
3 Data columns (total 4 columns):
4 #   Column      Non-Null Count  Dtype  
5 ---  --          --          --    
6 0   objectid    380 non-null    int64  
7 1   lad16nm     380 non-null    object  
8 2   Pct_Leave   380 non-null    float64
9 3   geometry    380 non-null    geometry
10 dtypes: float64(1), geometry(1), int64(1), object(1)
11 memory usage: 22.9+ KB

```

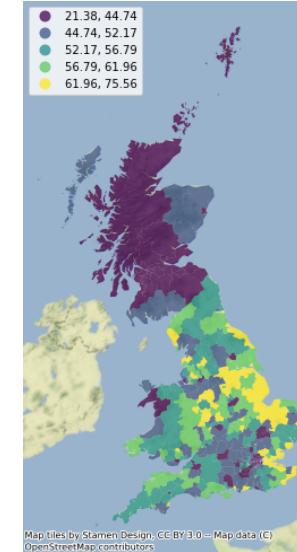
J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

122 / 534

## \*\*Code\*\* : ASG, Brexit (suite)

Extrait de [Rey et al., 2020]



J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

124 / 534

## \*\*Code\*\* : ASG, k-nn, pysal, Brexit

- Il faut à présent définir une matrice de SW représentant les relations de voisinage que nous allons tester au sens de l'ASG par la suite.
- Nous allons utiliser l'approche basée sur les k-nn avec  $k = 8$ .
- Nous allons également normer les poids de chaque ligne de sorte à ce que leur somme fasse 1.

```

1 # Generate W from the GeoDataFrame
2 from pysal.lib import weights
3 w = weights.KNN.from_dataframe(db, k=8)
4 # Get the dense adjacency matrix wmat
5 wmat, ids = w.full()
6 # print the row sum (degree) of 5 first nodes
7 print(wmat.sum(axis=1)[0:5])
8 # Row-standardization
9 w.transform = "R"
10 # Get the dense adjacency matrix wmat
11 wmat, ids = w.full()
12 # print the row sum (degree) of 5 first nodes
13 print(wmat.sum(axis=1)[0:5])

```

```

1 [8. 8. 8. 8. 8.]
2 [1. 1. 1. 1. 1.]

```

## L'opérateur de décalage spatial

- Le *spatial lag operator* est l'une des applications les plus directes et classiques des *spatial weights* (SW).
- Soit  $\mathbf{y}$  une variable mesurée sur  $n$  objets géographiques  $\{X_i\}_{i=1,\dots,n}$  et supposons que  $\mathbf{W}$  est une matrice carrée d'ordre  $n$  encodant une SW.
- Dans le cas discret, l'action de l'opérateur de  $\mathbf{W} = (w_{ii'})_{i,i'=1,\dots,n}$  sur  $\mathbf{x} = (x_i)_{i=1,\dots,n}$  est donné par le produit matriciel  $\mathbf{Wx}$ . Le résultat de l'opération est une vecteur de taille  $n$  que l'on dénotera par  $\mathbf{x}^s = (x_i^s)_{i=1,\dots,n}$  :

$$\mathbf{Wx} = \mathbf{x}^s \text{ avec } x_i^s = \sum_{i'=1}^n w_{ii'} x_{i'}$$

- Pour tout  $X_i$ ,  $x_i^s$  est une combinaison linéaire des valeurs  $x_{i'}$  de ses voisins  $X_{i'}$  dont les coefficients sont donnés par la matrice de SW  $\mathbf{W}$ . En effet, puisque  $w_{ii'} = 0$  si  $X_i$  et  $X_{i'}$  ne sont pas adjacents, seul le voisinage direct de  $X_i$  est pris en compte.

## Analyse de l'ASG des données du Brexit

- En visualisant la carte, il est assez probant de supposer une ASG positive : les localités avec un haut pourcentage de votes de "Leave" l'UE ont tendance à être adjacents (régions à l'est par ex.), et il en est de même pour les localités ayant un pourcentage de votes faible (Ecosse par ex.).
- Il faut toutefois faire attention à cette première impression. Malgré nos capacités humaines de reconnaître des motifs, nous pouvons en fait nous tromper. Il est nécessaire de tester au sens statistique du terme la présence ou l'absence d'ASG.
- Il existe des statistiques permettant de caractériser le degré de *clustering* spatial associé à une carte. Avant de les introduire, nous présentons un concept central, le décalage spatial/*spatial lag*.

## L'opérateur de décalage spatial (suite)

- L'opérateur de décalage spatial  $\mathbf{W}$  appliqué à  $\mathbf{x}$  produit un lissage local des valeurs selon les poids des relations spatiales qu'entretient chaque objet avec son voisinage. Nous avons les cas particuliers notables suivants :
  - Si  $\mathbf{W}$  est binaire alors ce lissage est en fait la somme des valeurs des voisins.
  - Si les lignes de  $\mathbf{W}$  sont normées<sup>2</sup> à 1, c.à.d. si  $\mathbf{W}\mathbf{1}_n = \mathbf{1}_n$  où  $\mathbf{1}_n$  est le vect. de taille  $n$  rempli de 1, alors ce lissage est une moyenne des valeurs des voisins.
- La lib. *pysal* permet d'appliquer facilement les opérateurs de décalage spatial.

2. On parle également de matrice stochastique.

## \*\*Code\*\* : ASG, k-nn, pysal, Brexit (suite)

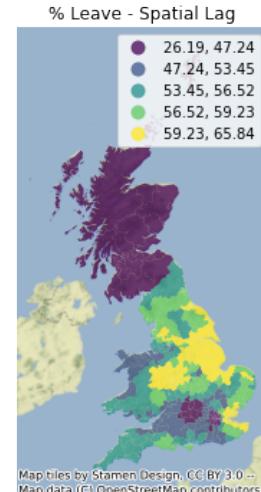
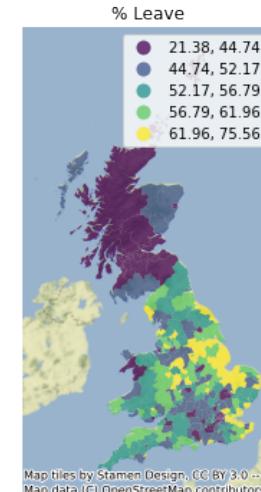
```

1 db["Pct_Leave_lag"] = weights.spatial_lag.lag_spatial(w, db["Pct_Leave"])
2 db.loc[[ "E08000012", "S12000019"], ["Pct_Leave", "Pct_Leave_lag"]]
3 f, axs = plt.subplots(1, 2, figsize=(12, 6))
4 ax1, ax2 = axs
5 db.plot(
6     column="Pct_Leave",
7     cmap="viridis",
8     scheme="quantiles",
9     k=5,
10    edgecolor="white",
11    linewidth=0.0,
12    alpha=0.75,
13    legend=True,
14    ax=ax1,
15 )
16 ax1.set_axis_off()
17 ax1.set_title("% Leave")
18 contextily.add_basemap(
19     ax1,
20     crs=db.crs,
21     source=contextily.providers.Stamen.TerrainBackground,
22 )

```

## \*\*Code\*\* : ASG, k-nn, pysal, Brexit (suite)

Extrait de [Rey et al., 2020]



## ASG, Cas binaire : fréquence jointe

- Il s'agit du cas où les observations des variables sont binaires :  $x \in \{0, 1\}^n$ . Dans ce cas, ce qui nous intéresse du point de vue de l'ASG, c'est de voir si la valeur observée en une localisation est entourée de valeurs identiques.
- Dans le cas des données du Brexit, nous allons transformer la var. quanti. du pourcentage de vote en faveur du "Leave" (Pct\_Leave) en une variable binaire renvoyant 1 ("Leave") si ce pourcentage est strictement supérieur à 50% (majorité) et 0 ("Remain") sinon.

```

1 db["Leave"] = (db["Pct_Leave"] > 50).astype(int)
2 db[["Pct_Leave", "Leave"]].tail()

```

	Pct_Leave	Leave
lad16cd		
W06000018	57.63	1
W06000019	62.03	1
W06000021	49.56	0
W06000022	55.99	1
W06000023	53.74	1

## \*\*Code\*\* : cas binaire, fréquence jointe, Brexit

- Nous éditons une carte choroplèthe de cette nouvelle var. binaire.
- Rq. : ci-dessous, nous n'avons pas ajouté de carte de tuiles *via* `contextily`.

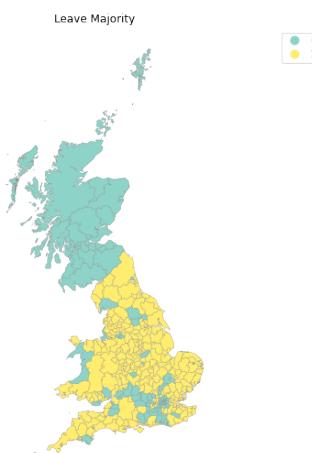
```

1 f, ax = plt.subplots(1, figsize=(9, 9))
2 db.plot(
3     ax=ax,
4     column="Leave",
5     categorical=True,
6     legend=True,
7     edgecolor="0.5",
8     linewidth=0.25,
9     cmap="Set3",
10    figsize=(9, 9),
11 )
12 ax.set_axis_off()
13 ax.set_title("Leave Majority")
14 plt.axis("equal")
15 plt.show()

```

## \*\*Code\*\* : cas binaire, fréquence jointe, Brexit (suite)

Extrait de [Rey et al., 2020]



- Il semble exister une ASG positive.

## \*\*Code\*\* : cas binaire, fréquence jointe, Brexit (suite)

- La lib. esda de pysal permet de calculer les fréquences jointes.

```

1 print(w.transform)
2 # transform w back to a non-standardized state:
3 w.transform = "0"
4 print(w.transform)
5
6 from pysal.explore import esda
7 from numpy.random import seed
8 seed(1234)
9 jc = esda.join_counts.Join_Counts(db["Leave"], w)
10 print(jc.bb)
11 print(jc.ww)
12 print(jc.bw)

```

```

1 871.0
2 302.0
3 347.0

```

- Par défaut, les codes sont b et w pour *black* (0) et *white* (1).

## ASG, Cas binaire : fréquence jointe, (suite)

- Pour quantifier cet *a priori* sur l'ASG positive de la var. binaire Leave, nous pouvons utiliser la statistique de fréquence jointe. Il s'agit de compter le nb. d'occurrences de paires de valeurs possibles parmi les paires d'objets en relation selon la matrice de SW (binaire) définie.
- Les paires étant non ordonnées, nous avons 3 cas possibles :
  - ▶ 11
  - ▶ 00
  - ▶ 01 (ou 10)
- Dans ce cas, l'ASG positive est d'autant plus avérée que la fréquence de 11 et de 00 est grande en comparaison de 01.
- En statistique inférentielle, on compare les observations empiriques à des observations théoriques dans le cas d'un modèle de structure spatiale totalement aléatoire. Plus on s'éloigne de cette situation, plus l'ASG positive est avérée.
- Rq. : *a contratio*, si le nb. d'occ. de 01 est plus important que ceux de 00 et 11, alors on parle d'ASG négative.

## ASG, Cas binaire : fréquence jointe, (suite)

- La statistique de test est basée sur la comparaison des nb. de 00, 11 et 01 empiriques (c.à.d. dans l'échantillon) versus les nb. de 00, 11 et 01, théoriques (c.à.d. que l'on aurait observé si une structure spatiale totalement aléatoire générât les données).
- En stat. inf. cela revient à poser l'hypothèse nulle suivante : "les valeurs sont générées par une structure spatiale totalement aléatoire".
- Une façon d'approximer ces nbs théoriques sous l'hypothèse nulle est de permutez les valeurs de la var. parmi les objets et de calculer pour cette permutation le nb. de 00, 11 et 01. On répète cette opération plusieurs fois et on peut alors calculer la moyenne des nb. de 00, 11 et 01 sur l'ens. des permutations réalisées.
- Intuitivement, si la moy. des nbs de 00 et 11 sur l'ens. des permutations est en dessous des valeurs empiriques alors on aura tendance à rejeter l'hypothèse nulle.

## ASG, Cas binaire : fréquence jointe, (suite)

- On peut également formaliser et quantifier cette démarche au travers du concept de *p-value* qui donne ici la probabilité d'observer une statistique de test de valeur au moins aussi extrême que celle que l'on observerait si l'hypothèse nulle était vérifiée.
- Si la *p-value* est très petite (en-dessous de 5%), cela veut dire que la valeur de la statistique de test sur l'échantillon est très extrême et on aura donc tendance à rejeter l'hypothèse nulle. Le risque (de 1ère espèce) que l'on prend en décidant de rejeter l'hypothèse nulle (si elle était en fait vraie) est alors égale à la *p-value*.
- Dans le contexte de l'ASG, du cas binaire et de la fréquence jointe, la *p-value* pour la configurations 00/11 (ASG positive) peut être estimée par la proportion de nb. de 00 et 11 dans l'ens. des résultats obtenus avec les permutations qui sont au-dessus du nb. de 00 et 11 dans l'échantillon.

## ASG, Cas continu : graphe et indice de Moran

- La démarche développée avec le cas binaire se poursuit dans le cas continu. On suppose donc que  $\mathbf{x} = (x_i)_{i=1,\dots,n}$  est un vecteur de  $\mathbb{R}^n$ . La matrice de SW est tjs dénotée  $\mathbf{W} = (w_{ii'})$ .
- Nous commençons par donner la formule de l'indice de Moran dénoté I appliqué à  $\mathbf{x}$  et dépendant de  $\mathbf{W}$  :

$$I(\mathbf{x}, \mathbf{W}) = \frac{n}{\sum_{i,i'=1}^n w_{ii'}} \frac{\sum_{i,i'=1}^n w_{ii'}(x_i - \bar{x})(x_{i'} - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

où  $\bar{x} = \frac{1}{n} \sum_i x_i$

- Pour appréhender cette formule, nous introduisons le graphe de Moran qui est un nuage de points en 2D. En abscisse on représente les valeurs centrées de  $\mathbf{x}$  et en ordonnée les valeurs centrées de  $\mathbf{W}\mathbf{x}$ .

## \*\*Code\*\* : cas binaire, fréquence jointe, Brexit (suite)

```

1 # observed nb of occ of 00 and 11 cases
2 print(jc.bb+jc.ww)
3 # observed nb of occ of 01 or 10 cases
4 print(jc.bw)
5 print("---")
6 # expectation under null hyp of 00/11 cases
7 print(jc.mean_bb)
8 # expectation under null hyp of 01/10 cases
9 print(jc.mean_bw)
10 print("****")
11 # pseudo \tit{p-value} of the 00/11 case
12 print(jc.p_sim_bb)

```

```

1 1173.0
2 347.0
3 ---
4 727.4124124124124
5 649.3233233233233
6 ***
7 0.001

```

## \*\*Code\*\* : cas continu, graphe de Moran, Brexit

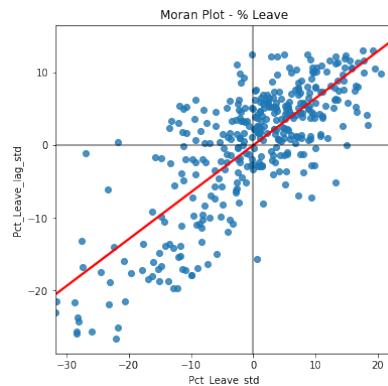
```

1 db["Pct_Leave_std"] = db["Pct_Leave"] - db["Pct_Leave"].mean()
2 db["Pct_Leave_lag_std"] = (
3     db["Pct_Leave_lag"] - db["Pct_Leave_lag"].mean()
4 )
5
6 f, ax = plt.subplots(1, figsize=(6, 6))
7 seaborn.regplot(
8     x="Pct_Leave_std",
9     y="Pct_Leave_lag_std",
10    ci=None,
11    data=db,
12    line_kws={"color": "r"},
13 )
14 ax.axvline(0, c="k", alpha=0.5)
15 ax.axhline(0, c="k", alpha=0.5)
16 ax.set_title("Moran Plot - % Leave")
17 plt.show()

```

## \*\*Code\*\* : cas continu, graphe de Moran, Brexit (suite)

Extrait de [Rey et al., 2020]



- La ligne rouge est la droite des MCO. La pente est positive. Autrement dit,  $x$  et sa transformée par un lissage local,  $\mathbf{W}x$ , sont deux vecteurs positivement corrélés :
  - Lisser une valeur forte (resp. faible)  $x_i$ , par une moyenne arith. de ses plus 8 proches voisins a tendance à donner une valeur forte (resp. faible).
  - Le voisinage d'une observation a tendance à être similaire à cette observation (ASG positive).

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

141 / 534

## ASG, Cas continu : graphe et indice de Moran (suite)

- L'indice de Moran, vise à résumer, en une statistique, la tendance de la relation entre  $x$  et  $\mathbf{W}x$  que l'on peut visualiser à l'aide du graphe de Moran comme ci-dessus.
- Il existe en fait une connection directe entre le graphe et l'indice de Moran :  $I$  est en fait la pente de la droite des MCO du nuage de points donné par  $(x, \mathbf{W}x)$ .
- L'indice de Moran s'obtient facilement via le package esda.

```
1 w.transform = "R"
2 moran = esda.moran.Moran(db["Pct_Leave"], w)
3 print(moran.I)
```

```
1 0.6454521298096587
```

- Rq. : il existe d'autres mesures d'ASG (Geary, Getis et Ord) mais nous ne les aborderons pas.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

142 / 534

## ASG, Cas continu : graphe et indice de Moran (suite)

- esda permet également de déterminer une  $p$ -value associée à la stat. de test  $I$ . Dans ce cas, ce sont les localisations spat. des valeurs observées qui sont mélangées aléatoirement. On calcule l'ind. de Moran sur les données permutées. On peut en fait transformer  $I$  de sorte à ce qu'elle suive une loi normale centrée réduite.
- En faisant plusieurs simulations (999 par défaut), esda permet de calculer une mesure de confiance sur la vraisemblance d'obtenir un motif de carte similaire à celui observé dans l'échantillon sous l'hypothèse d'un processus de génération totalement aléatoire. Cette mesure de probabilité peut être interprétée comme une  $p$ -value.
- Si cette  $p$ -value est très petite, on rejette l'hypothèse nulle de structure spatiale totalement aléatoire.

```
1 moran.p_sim
```

```
1 0.001
```

J. Ah-Pine

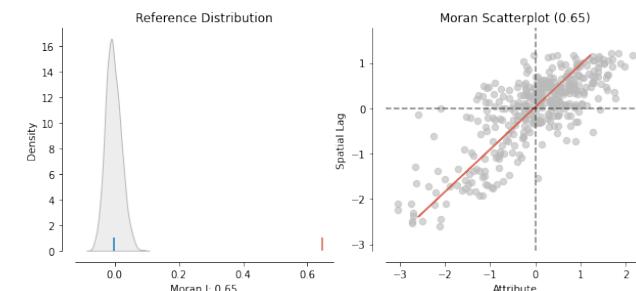
Masterclass IHEDD-CERDI-GDN GeoAI 24

143 / 534

## \*\*Code\*\* : cas continu, I de Moran, Brexit (suite)

- La sous lib. splot de pysal permet d'enrichir ces données numériques par des graphiques.

```
1 from pysal.viz import splot
2 from splot.esda import plot_moran
3 plot_moran(moran)
```



Extrait de [Rey et al., 2020]

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

144 / 534

## Rappel du Sommaire

### 3 Analyse de la dépendance spatiale

- Carte choroplète et discréétisation de variables quantitatives
- Autocorrélation spatiale globale
- Autocorrélation spatiale locale

## Introduction (suite)

- L'indice de Moran, s'il est positif (négatif resp.), est une mesure de l'AS Globale qui résume le degré de *clustering* spatial (dispersion spatiale resp.) des données. Cependant, il ne nous indique pas dans quelles régions de l'espace les associations de valeurs sont particulièrement similaires (opposées resp.).
- Dans ce qui suit, nous introduisons des **mesures Locales d'Autocorrélation Spatiale (ASL)**. Dans ce contexte, nous nous intéressons à comment la valeur observée de **chaque objet** est en relation avec les valeurs observées de ses objets voisins.
- Il s'agit donc d'une distribution de scores sur l'ens. des objets. La démarche reste cependant proche des mesures globales précédentes. Certaines mesures locales peuvent notamment être agrégées et donner lieu à une mesure globale.
- Nous verrons en particulier **LISA** (*Local Indicators of Spatial Association*) introduit dans [Anselin, 1995] qui est une mesure locale dont l'agrégation permet de retrouver l'indice de Moran.

## Introduction

- La présence d'une ASG est intéressante pour l'analyse car elle nous indique que la distribution des valeurs d'une var. au sein des objets à l'étude peut être associée à un processus stochastique spatial (ou champ aléatoire) non totalement aléatoire qui permettrait de modéliser comment les associations de valeurs sont générées dans les dif. régions de l'espace.
- En pratique, une ASG positive indique qu'il est opportun de chercher à modéliser des phénomènes tels que la contagion en épidémie ou les externalités environnementales négatives (*spillover effects*) des activités humaines en économie de l'environnement, ... Dans toutes ces applications, la valeur observée en un point de l'espace influence de façon forte la valeur observée en des points voisins.
- Toutefois, il peut s'avérer que l'ASG constatée *a priori* provienne en fait d'un bruit et non de la var. Par ex., un appareil de mesure peut être biaisé selon la position géographique, ou être bruité comme avec la présence d'un nuage pour des instruments de télédétection, ...

## ASL par l'exemple : les données de vote du Brexit

- Nous continuons l'illustration des concepts clefs à l'aide de l'exemple du jeu de données Brexit.
- Les mêmes fichiers .csv et .geojson sont utilisés et nous repartons des mêmes prétraitements que précédemment.
- En pratique, il s'agit d'exécuter à nouveau les codes donnés en slides 120, 122, 125.
- En particulier, nous utilisons la SW basée sur les 8 plus proches voisins. Par ailleurs, la matrice **W** obtenue est transformée de sorte à ce que la somme de chaque ligne soit 1 (matrice stochastique).

```

1 # Generate W from the GeoDataFrame
2 from pysal.lib import weights
3 w = weights.KNN.from_dataframe(db, k=8)
4 # Row-standardization
5 w.transform = "R"

```

- Afin de motiver la mesure LISA, nous revenons sur le graphe de Moran de la variable Pct\_Leave (cf également slide 141).

## \*\*Code\*\* : ASL, graphe de Moran, Brexit

- Nous appliquons à nouveau l'opérateur de décalage spatial  $\mathbf{W}$  à la var. Pct\_Leave que nous notons cette fois-ci  $w\_Pct\_Leave$ .

```

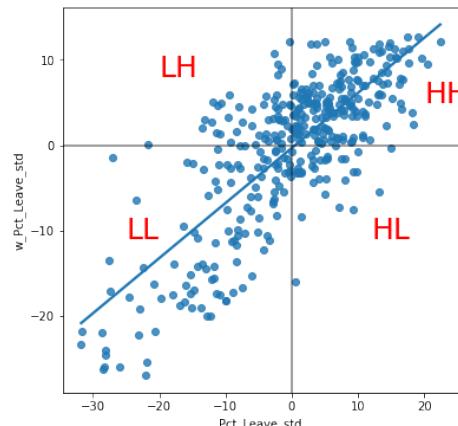
1 db["w_Pct_Leave"] = weights.spatial_lag.lag_spatial(
2   w, db["Pct_Leave"])
3 )
4 db["Pct_Leave_std"] = db["Pct_Leave"] - db["Pct_Leave"].mean()
5 db["w_Pct_Leave_std"] = db["w_Pct_Leave"] - db["Pct_Leave"].mean()
6
7 # Setup the figure and axis
8 f, ax = plt.subplots(1, figsize=(6, 6))
9 # Plot values
10 seaborn.regplot(
11   x="Pct_Leave_std", y="w_Pct_Leave_std", data=db, ci=None
12 );

```

- Comme les var. Pct\_Leave\_std et w\_Pct\_Leave\_std sont des pourcentages qui ont été centrés, nous pouvons découper le nuage de points en 4 parties correspondants aux **4 quadrants du plan (espace 2D)** qui croisent les valeurs positives qui indiquent des valeurs initiales "très" grandes (au-dessus de la moyenne) (High) et les valeurs négatives indiquant des valeurs "très" petites (Low).

## \*\*Code\*\* : ASL, graphe de Moran, Brexit (suite)

Extrait de [Rey et al., 2020]



## \*\*Code\*\* : ASL, graphe de Moran, Brexit (suite)

```

1 # Setup the figure and axis
2 f, ax = plt.subplots(1, figsize=(6, 6))
3 # Plot values
4 seaborn.regplot(
5   x="Pct_Leave_std", y="w_Pct_Leave_std", data=db, ci=None
6 )
7 # Add vertical and horizontal lines
8 plt.axvline(0, c="k", alpha=0.5)
9 plt.axhline(0, c="k", alpha=0.5)
10 # Add text labels for each quadrant
11 plt.text(20, 5, "HH", fontsize=25, c="r")
12 plt.text(12, -11, "HL", fontsize=25, c="r")
13 plt.text(-20, 8.0, "LH", fontsize=25, c="r")
14 plt.text(-25, -11.0, "LL", fontsize=25, c="r")
15 # Display
16 plt.show()

```

## Indices locaux de Moran I<sub>i</sub>

- Une façon d'interpréter le graphique précédent est qu'il représente une sorte de **classification des observations** (380 points/circonscriptions représentés) qui dépend des valeurs prises par celles-ci et ainsi de celles prises par leurs voisins respectifs.
- Rq. : cette classification fait d'une certaine façon écho aux quantités 11, 00, 10 et 01 utilisées précédemment dans le cas binaire.
- Afin de savoir si en une localisation spatiale donnée, les valeurs observées dans son voisinage présentent une concentration de valeurs particulièrement similaires (c.à.d. statistiquement significative), il nous faut comparer cette situation à celle que l'on aurait en cas de structure spatiale totalement aléatoire.
- Comme précédemment, notre hypothèse nulle est de type : "les valeurs dans le voisinage de cette position sont générées par une structure spatiale totalement aléatoire".

## Indices locaux de Moran $I_i$ (suite)

- La mesure LISA permet de quantifier de combien la concentration de valeurs similaires autour d'une position géographique est particulièrement grande.
- En fait, la statistique LISA est une version locale de l'indice de Moran.
- L'idée centrale consiste à **identifier les cas** pour lesquels, la valeur observée en une position et la valeur moyenne observée dans son voisinage est plus similaire (HH ou LL dans le graphique précédent) que dissimilaire (HL et LH) par rapport à ce que l'on aurait observé si les valeurs étaient distribuées de façon totalement aléatoire.
- La même démarche que pour  $I$  est appliquée mais à chq. objet. Ainsi pour  $X_i$ , sa mesure LISA, dénotée  $I_i$ , est définie par :

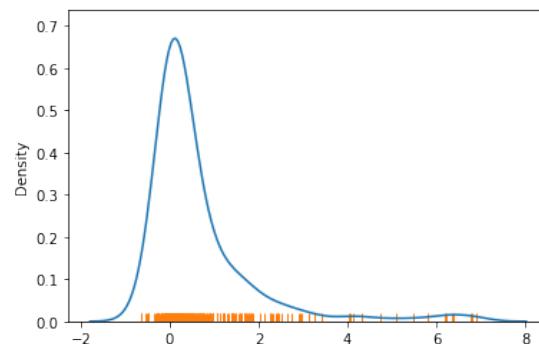
$$I_i(\mathbf{x}, \mathbf{W}) = \frac{x_i - \bar{x}}{\hat{\sigma}_x^2} \sum_{i'=1}^n w_{ii'}(x_{i'} - \bar{x}) \text{ où } \hat{\sigma}_x^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}$$

## \*\*Code\*\* : cas continu, LISA, Brexit

```

1 from pysal.explore import esda
2 lisa = esda.moran.Moran_Local(db["Pct_Leave"], w)
3 # Draw KDE line
4 ax = seaborn.kdeplot(lisa.Is,
5 # Add one small bar (rug) for each observation along horizontal axis
6 seaborn.rugplot(lisa.Is, ax=ax);
7 plt.show()

```



## Indices locaux de Moran $I_i$ (suite)

- La mesure LISA est utilisée dans de nbx domaines afin d'identifier des clusters géographiques de valeurs ou pour détecter des outliers géographiques.
- Elle est un outil pratique pour déterminer rapidement les régions de l'espace présentant une concentration forte de valeurs similaires et argumenter sur la présence d'un processus stochastique spatial (non totalement aléatoire).
- LISA a été utilisée par exemple pour identifier des clusters géographiques de pauvreté [Dawson et al., 2018], ou pour identifier dans l'espace des clusters d'individus atteints de la COVID [Zhang et al., 2020], ...
- En Python, la librairie esda nous permet facilement de calculer les mesures LISA pour chaque objet.

## Indices locaux de Moran $I_i$ , Brexit

- L'histogramme des valeurs de  $\{I_i\}_{i=1,\dots,n}$  présente une asymétrie qui est due à la forte proportion des ASL positives : 296/380 cas.
- Une AS positive en une localisation spatiale correspond au fait que si la valeur qui y est observée est forte, alors elle a tendance à être entourée par de fortes valeurs également (HH). Mais une AS locale positive peut aussi être affirmée si étant donnée la faible valeur observée en un point, alors il a tendance à être entourée par de faibles valeurs également (LL).
- Dans les deux cas, la mesure LISA est positive et celle-ci ne permet donc pas de faire le *distinguo* entre les situations HH et LL.
- Une AS négative en une position spatiale correspond au cas où la valeur observée est opposée aux valeurs dans le voisinage. Il existe également deux situations : valeur forte entourée par des valeurs faibles (HL) ou valeur faible entourée par des valeurs fortes (LH). Ici aussi, la mesure LISA ne fait pas le *distinguo*.

## Indices locaux de Moran $I_i$ , Brexit (suite)

- Visualiser la liste des valeurs numériques des mesures LISA  $\{I_i\}$ , n'est pas la façon la plus utile d'exploiter cet outil. Il est en revanche intéressant de visualiser ces valeurs par l'intermédiaire d'une carte choroplèthe (et donc à l'issue d'une discréétisation) afin d'apprecier ce que peut révéler cette mesure du point de vue spatial.
- En raison des arguments avancés précédemment, c.à.d. le fait que la mesure LISA ne permet pas de faire la distinction entre les cas HH, LL, HL et LH, une carte choroplèthe permettant de visualiser ces 4 quadrants est également un bon outil d'analyse.
- Les graphiques qui suivent permettent d'avoir des éléments de réponses à des questions de type :
  - ▶ Deux clusters géographiques présentant des ASL fortement positives sont-ils de même nature ? (Ecosse *versus* Est de l'Angleterre par ex.)
  - ▶ Les valeurs de LISA proches de 0 ont tendance à ne pas être statistiquement significatives d'une ASL. Quelles valeurs de LISA sont-elles stat. signif., quelles sont celles qui ne le sont pas ?

## \*\*Code\*\* : cas continu, LISA, carte choro., Brexit

```

1 from splot import esda as esdaplot
2
3 # Set up figure and axes
4 f, axs = plt.subplots(nrows=1, ncols=2, figsize=(12, 12))
5 # Make the axes accessible with single indexing
6 axs = axs.flatten()
7
8 # Subplot 1 #
9 # Choropleth of local statistics
10 # Grab first axis in the figure
11 ax = axs[0]
12 # Assign new column with local statistics on-the-fly
13 db.assign(
14     Islisa.Islisa
15     # Plot choropleth of local statistics
16 ).plot(
17     columns="Is",
18     cmap="plasma",
19     scheme="quantiles",
20     k=5,
21     edgecolor="white",
22     linewidth=0.1,
23     alpha=0.75,
24     legend=True,
25     ax=ax,
26 )

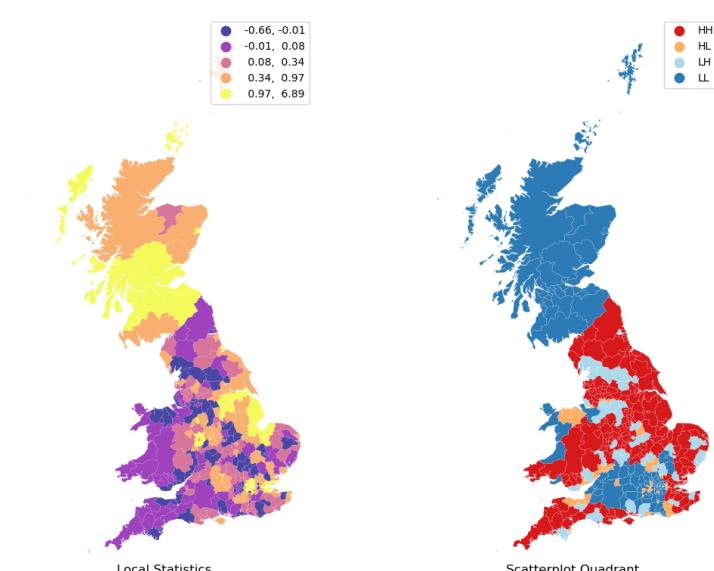
```

## \*\*Code\*\* : cas continu, LISA, carte choro., Brexit (suite)

```

1 # Subplot 2 #
2 # Quadrant categories
3 # Grab second axis of local statistics
4 ax = axs[1]
5 # Plot Quadrant colors (note to ensure all polygons are assigned a
6 # quadrant, we "trick" the function by setting significance level to
7 # 1 so all observations are treated as "significant" and thus assigned
8 # a quadrant color
9 esdaplot.lisa_cluster(lisa, db, p=1, ax=ax)
10 for i, ax in enumerate(axs.flatten()):
11     ax.set_axis_off()
12     ax.set_title(
13         [
14             "Local Statistics",
15             "Scatterplot Quadrant"
16         ][i],
17         y=0,
18     )
19
20 plt.show()

```



## Indices locaux de Moran $I_i$ , Brexit (suite)

- Le graphique de gauche est une carte choroplèthe de la var. LISA,  $\{I_i\}_{i=1,\dots,n}$ , obtenue avec une discréttisation quantiles.
- Le graph. de droite est aussi une carte choro. de la var. LISA mais la discrét. est donnée par l'appartenance à l'un des 4 quadrants HH, LL, HL, LH (cf slide 150).
- Sur le graphe de gauche, nous voyons que l'Ecosse et l'Est de l'Angleterre sont deux régions avec des valeurs de LISA fortement positives mais nous ne connaissons pas la nature de cette ASL.
- Le graphique de droite permet alors de faire cette distinction : l'Ecosse présente ainsi une forte ASL de type LL ("Remain") tandis que l'Est de l'Angleterre montre une forte ASL de type HH ("Leave").
- Ces deux graphiques sont donc complémentaires.

## Indices locaux de Moran $I_i$ , Brexit (suite)

- La librairie esda produit également des pseudo *p-value* pour tester la significativité des ASL par le biais de la stat. LISA.
- Il est important en effet de comparer les valeurs de LISA empiriques vis-à-vis de valeurs simulées sous l'hypothèse d'une structure spatiale totalement aléatoire.
- Pour chq. objet  $X_i$ , une *p-value* associée à la valeur  $I_i$  est générée à l'aide de simulations. Si cette *p-value* est plus grande que 0.05 (5%) alors l'hypothèse nulle n'est pas rejetée et l'ASL (pos. ou nég.) est ainsi considérée comme non significative. Si la *p-value* est plus petite que 0.05, on considère qu'avec un risque de 1ère espèce faible, l'ASL en  $X_i$  est acceptée.
- La *cluster map* une une carte choro. qui permet de visualiser les pos. spat. dont les mesures de LISA sont non signifi. versus celles qui sont stat. signif. De plus, parmi les obs. dont la stat.  $I_i$  est signif., elle permet de distinguer le type d'ASL associé entre HH, LL, HL et LH.

## \*\*Code\*\* : cas continu, LISA, carte choro., Brexit (suite)

- L'appartenance de chq. obs. à l'un des 4 quadrants est donné dans lisa.q avec l'encodage : 1 (HH), 2 (LH), 3 (LL) et 4 (HL)

```
1 print(lisa.q[:10])
```

```
1 [1 1 1 1 1 1 4 1 4 1]
```

- On peut alors compter le nb. d'occurrences de chq. cas.

```
1 counts = pandas.value_counts(lisa.q)
2 print(counts)
```

```
1      183
2      113
3       50
4       34
5  dtype: int64
```

- Le nb. d'ASL pos. ( $183+113=296$ ) est prédominant (cf slide 156).

## \*\*Code\*\* : cas continu, LISA, carte choro., Brexit

```
1 f, axs = plt.subplots(nrows=1, ncols=2, figsize=(12, 12))
2 # Make the axes accessible with single indexing
3 axs = axs.flatten()
4 # Subplot 3 #
5 # Significance map
6 # Grab third axis of local statistics
7 ax = axs[0]
8 #
9 # Find out significant observations
10 labels = pandas.Series(
11     1 * (lisa.p_sim < 0.05), # Assign 1 if significant, 0 otherwise
12     index=db.index # Use the index in the original data
13     # Recode 1 to "Significant" and 0 to "Non-significant"
14 ).map({1: "Significant", 0: "Non-Significant"})
15 # Assign labels to 'db' on the fly
16 db.assign(
17     cl=labels
18     # Plot choropleth of (non-)significant areas
19 ).plot(
20     column="cl",
21     categorical=True,
22     k=2,
23     cmap="Paired",
24     linewidth=0.1,
25     edgecolor="white",
26     legend=True,
27     ax=ax,
28 )
```

## \*\*Code\*\* : cas continu, LISA, carte choro., Brexit (suite)

```

1 # Subplot 4 #
2 # Cluster map
3 # Grab second axis of local statistics
4 ax = axs[1]
5 # Plot Quadrant colors In this case, we use a 5% significance
6 # level to select polygons as part of statistically significant
7 # clusters
8 esdaplot.lisa_cluster(lisa, db, p=0.05, ax=ax)
9
10 for i, ax in enumerate(axes.flatten()):
11     ax.set_axis_off()
12     ax.set_title(
13         [
14             "Statistical Significance",
15             "Moran Cluster Map"
16         ][i],
17         y=0,
18     )
19
20 plt.show()

```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

165 / 534

## \*\*Code\*\* : cas continu, LISA, carte choro., Brexit (suite)

- Sur la carte, moins de la moitié des polygones sont colorés avec une couleur impliquant la significativité statistique.

```
1 print((lisa.p_sim < 0.05).sum() * 100 / len(lisa.p_sim))
```

```
1 41.05263157894737
```

- On identifie alors 3 régions contre le Brexit : l'Ecosse, Londres et Oxford (nord-ouest de Londres).
- Par ailleurs, dans la carte choro. en slide 160, de nbs régions de l'Angleterre indiquent, en rouge, être en faveur du Brexit mais la carte précédente montre que seules le nord-est et l'ouest de l'Angleterre sont les régions où l'ASL positive HH est stat. significative (donc non due à "la chance").

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

167 / 534

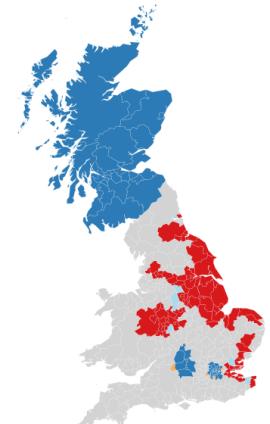
## \*\*Code\*\* : cas continu, LISA, carte choro., Brexit (suite)

Non-Significant  
Significant



Statistical Significance

HH  
HL  
LH  
LL  
ns



Moran Cluster Map

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

166 / 534

## Rappel du Sommaire

- 1 Introduction et motivations générales
- 2 Concepts de base et outils Python associés
- 3 Analyse de la dépendance spatiale
- 4 Méthodes de régression spatiale
- 5 Méthodes d'ensemble en ML et extensions spatiales
- 6 Eléments de géostatistique
- 7 Méthodes de DL et extensions spatiales

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

168 / 534

## Introduction

- La régression est un modèle mathématique qui exprime une variable dite dépendante (ou endogène ou à expliquer) notée  $Y$  comme fct. de plus. autres variables dites indépendantes (ou exogènes ou explicatives) notées  $X^1, \dots, X^p$ .
- On note une instance quelconque de  $(X^1, \dots, X^p, Y)$  par le couple  $(\mathbf{x}, y)$  où  $\mathbf{x} = (x_1, \dots, x_p) \in \mathbb{R}^p$  et  $y \in \mathbb{R}$ .
- Dans le cadre paramétrique, une classe de fonction  $\mathbb{H}$  dépendante d'un ensemble de paramètres représente l'ensemble des dépendances que l'on suppose entre la var. indépendante et les var. dépendantes.
- Dans le cadre non paramétrique, la régression utilise des méthodes proches des techniques d'estimation non paramétrique de densité utilisant des fcts noyaux.
- Dans la suite nous nous focalisons sur les méthodes paramétriques qui permettent de mieux comprendre l'influence des var. explicatives sur la var. à expliquer.

## Régression spatiale

- Dans le cas de données géoréférencées, les structures spatiales permettent d'augmenter la pertinence de modèles de régression :
  - ▶ Lorsque le **phénomène à l'étude est lui-même de nature géographique**. Par ex. : pour les prix de l'immobilier, un bien identique n'aura pas le même prix selon son emplacement. Un même appartement coûtera plus cher dans un quartier calme, verdoyant et avec des commerces, et moins cher dans un quartier loin de tout centre d'intérêt, pollué et bruyant. Dans ce cas, la "géographie" peut être une var. explicite qui aide à prédire le prix des biens.
  - ▶ Lorsque l'analyse des **erreurs du modèle de prédiction montre une structure de clustering qui proviendrait d'une structure spatiale** (des observations proches dans l'espace donnent des erreurs proches). Cependant, cette structure spatiale d'intérêt peut être difficile à identifier car elle pourrait être causée par l'absence d'une var. ou par l'interaction complexe entre plus. variables du modèle... Dans ce contexte, c'est davantage les informations sur le voisinage d'une observation qui permettrait d'améliorer la modélisation.

## Introduction (suite)

- Dans le cas particulier de la régression **linéaire multiple**, on suppose que la var. endogène  $Y$  est obtenue par une combinaison linéaire des var. exogènes  $\{X^j\}_{j=1, \dots, p}$ .
- Nous noterons les coefficients de cette combinaison linéaire par  $a_1, \dots, a_p$ . On ajoute dans le modèle de façon facultative une constante  $a_0$  appelée biais ou *offset*. Formellement, cela revient à prendre comme classe de fonctions :

$$\mathbb{H} = \{f : \mathbb{R}^p \rightarrow \mathbb{R}, f(\mathbf{x}) = a_0 + a_1 x_1 + \dots + a_p x_p; a_0, a_1, \dots, a_p \in \mathbb{R}^p\}$$

- La régression linéaire multiple est de façon générale une approche très classique en modélisation statistique. Elle permet de tester des hypothèses sur des dépendances/associations entre var. et de faire des prédictions. Il s'agit d'un modèle interprétable : on peut aisément appréhender la dépendance de la var. à expliquer  $Y$  vis-à-vis d'une var. explicative  $X^j$  en considérant  $\partial f(X^1, \dots, X^p) / \partial X^j = a_j$ .

## De la régression linéaire à la régression spatiale

- Nous commencerons par rappeler des concepts de base en régression linéaire multiple sans information spatiale.
- Nous verrons par la suite trois approches distinctes permettant d'intégrer des structures spatiales :
  - ▶ Intégration par une ou plus. var. exogènes de nature géographique.
  - ▶ Prise en compte de l'hétérogénéité spatiale dans la modélisation.
  - ▶ Intégration de l'influence des voisins par différentes structures ou modèles.
- Notre objectif sera avant tout d'aborder les principes importants des différentes méthodes sans rentrer dans les détails techniques.
- Nous continuons l'approche par l'exemple afin d'illustrer aisément l'intérêt, et les limites des méthodes. Les données que nous exploitons sont les prix de locations sur le site AirBnB dans la région de San Diego, Californie, USA.

## De la régression linéaire à la régression spatiale (suite)

- Les lib. que nous utiliserons dans ce qui suit sont les suivantes.

```

1 from pysal.lib import weights
2 from pysal.explore import esda
3 import numpy
4 import pandas
5 import geopandas
6 import matplotlib.pyplot as plt
7 import seaborn
8 import contextily

```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

173 / 534

## Rég. spatiale par l'exemple : San Diego AirBnB (suite)

```

1 <class 'geopandas.geodataframe.GeoDataFrame'>
2 RangeIndex: 6110 entries, 0 to 6109
3 Data columns (total 20 columns):
4 #   Column      Non-Null Count  Dtype  
5 --- 
6 0   accommodates    6110 non-null   int64  
7 1   bathrooms       6110 non-null   float64 
8 2   bedrooms        6110 non-null   float64 
9 3   beds            6110 non-null   float64 
10 4   neighborhood    6110 non-null   object  
11 5   pool            6110 non-null   int64  
12 6   d2balboa        6110 non-null   float64 
13 7   coastal          6110 non-null   int64  
14 8   price           6110 non-null   float64 
15 9   log_price        6110 non-null   float64 
16 10  id              6110 non-null   int64  
17 11  pg_Apartment    6110 non-null   int64  
18 12  pg_Condominium  6110 non-null   int64  
19 13  pg_House         6110 non-null   int64  
20 14  pg_Other         6110 non-null   int64  
21 15  pg_Townhouse    6110 non-null   int64  
22 16  rt_Entire_home/apt 6110 non-null   int64  
23 17  rt_Private_room  6110 non-null   int64  
24 18  rt_Shared_room   6110 non-null   int64  
25 19  geometry         6110 non-null   geometry 
26 dtypes: float64(6), geometry(1), int64(12), object(1)
27 memory usage: 954.8+ KB

```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

175 / 534

## Rég. spatiale par l'exemple : San Diego AirBnB

- Ce jeu de données comportent 6110 observations correspondant à des biens immobiliers mis à la location sur la plateforme AirBnB dans l'agglomération de San Diego en Californie.
- Les variables numériques sont de différentes natures et peuvent être quantitatives (ex. : nb. de lits par ex.), qualitative (ex. : type de bien -appartement, maison, ...-), binaire (ex. : piscine -oui ou non-).
- Le jeu de données comportent également des variables qui encodent explicitement une information géographique comme la distance du *Balboa Park* (var. quantitative), l'appartenance à un quartier comme par ex. *North Hills, Mission Bay, ...* (var. qualitative)

```

1 db = geopandas.read_file("regression_db.geojson")
2 db.info()

```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

174 / 534

## Rappel du Sommaire

### 4 Méthodes de régression spatiale

- Rappels en régression linéaire multiple
- Application de la rég. lin. multiple à des données spatiales
- Ajout de variables explicatives de nature géographique
- Modèles d'hétérogénéité spatiale
- Modèles de dépendance spatiale
- Geographically Weighted Regression

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

176 / 534

## Backgrd : Régression linéaire multiple et MCO

- Rappelons que nous souhaitons déterminer une fonction  $f$  modélisant la relation entre la variable à expliquer  $Y$  et les variables explicatives  $\{X^1, X^2, \dots, X^p\}$ .
- Le modèle de régression linéaire multiple est le suivant :

$$Y = f(X^1, \dots, X^p) + \epsilon = a_0 + \sum_{j=1}^p a_j X^j + \epsilon$$

où  $\epsilon$  indique l'erreur ou le résidu entre la réalité  $Y$  et la prédiction du modèle  $f(X)$ .

- Les variables explicatives peuvent être :
  - Les variables initiales du problème.
  - Des transformations des variables initiales.
  - Des expansions de bases des variables initiales [Hastie et al., 2011].
- Le modèle reste une fonction linéaire des paramètres  $\mathbf{p} = (a_j)_{j=0, \dots, p}$ .

## Backgrd : Régression linéaire multiple et MCO (suite)

- Nous avons :

$$\mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_p \end{pmatrix} ; \quad \mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \dots & \dots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} ; \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

- Notons par ailleurs  $\mathbf{X}^\top$  la matrice transposée de  $\mathbf{X}$ .
- Nous avons alors l'écriture matricielle suivante :

$$\text{Scr}(f) = (\mathbf{y} - \mathbf{X}\mathbf{a})^\top (\mathbf{y} - \mathbf{X}\mathbf{a})$$

- On cherche à déterminer les paramètres  $\mathbf{p} = (a_j)_{j=0, \dots, p}$  représentés par le vecteur  $\mathbf{a}$  qui minimise  $\text{Scr}(f)$  : c'est un problème d'optimisation quadratique non contraint :

$$\hat{\mathbf{a}}_{mco} = \arg \min_{\mathbf{a} \in \mathbb{R}^{p+1}} (\mathbf{y} - \mathbf{X}\mathbf{a})^\top (\mathbf{y} - \mathbf{X}\mathbf{a})$$

- La solution s'obtient en recherchant les points  $\mathbf{a}$  tel que  $\nabla \text{Scr}(\mathbf{a}) = \mathbf{0}$ .

## Backgrd : Régression linéaire multiple et MCO (suite)

- L'étape d'induction consiste à estimer les paramètres  $\mathbf{p}$  étant données l'échantillon  $\mathbb{E} = \{(\mathbf{x}_i), y_i\}_{i=1, \dots, n}$ .
- La méthode classique est les **Moindres Carrés Ordinaires** (MCO) qui vise la minimisation de la Somme des Carrés des Résidus (Scr) :

$$\begin{aligned} \text{Scr}(f) &= \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 \\ &= \sum_{i=1}^n (y_i - (a_0 + \sum_{j=1}^p a_j x_{ij}))^2 \end{aligned}$$

- Du point de vue statistique, l'utilisation de ce modèle suppose que les obs.  $y_i$  sont des réalisations de va.  $Y_i$  i.i.d.
- Introduisons les notations suivantes :
  - $\mathbf{a}$ , le vecteur colonne de taille  $p + 1$  contenant les paramètres.
  - $\mathbf{X}$ , la matrice des données de taille  $(n \times (p + 1))$  à laquelle on a ajouté une 1ère colonne remplie de 1.

## Backgrd : Rappels en calcul différentiel

- Si  $f : \mathbb{R}^{p+1} \rightarrow \mathbb{R}$  est différentiable, alors la fonction  $\nabla f$  défini par :

$$\nabla f(\mathbf{a}) = \begin{pmatrix} \frac{\partial f}{\partial a_0}(\mathbf{a}) \\ \frac{\partial f}{\partial a_1}(\mathbf{a}) \\ \vdots \\ \frac{\partial f}{\partial a_p}(\mathbf{a}) \end{pmatrix}$$

est appelé **gradient** de  $f$ .

- $\nabla f$  est une fonction de  $\mathbb{R}^{p+1}$  dans  $\mathbb{R}^{p+1}$  et peut être vue comme un **champ de vecteurs** (fonction qui associe à tout point un vecteur).
- Voici qques formules de dérivation dans le cas multivarié. La dérivée est calculée par rapport à la variable  $\mathbf{x}$ .  $\mathbf{A}$  est une matrice de réels de taille  $(m \times n)$  et  $\mathbf{y}$  un vecteur de réels de taille  $(m \times 1)$  :
  - Si  $f(\mathbf{x}) = \mathbf{y}^\top \mathbf{A} \mathbf{x}$  ou si  $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}^\top \mathbf{y}$  alors  $\nabla f(\mathbf{x}) = \mathbf{A}^\top \mathbf{y}$ .
  - Si  $\mathbf{A}$  est carrée et  $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$  alors  $\nabla f(\mathbf{x}) = (\mathbf{A} + \mathbf{A}^\top) \mathbf{x}$ .
  - Si  $\mathbf{A}$  est carrée symétrique et  $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$  alors  $\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x}$ .

## Backgrd : Régression linéaire multiple et MCO (suite)

- On développe  $\text{Scr}(f)$  de la manière suivante :

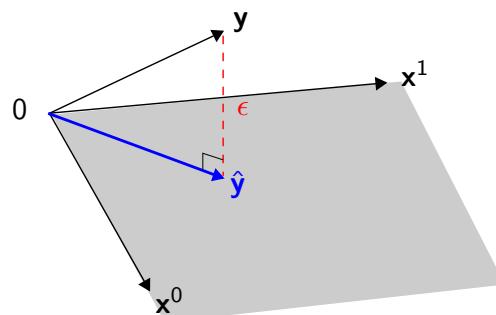
$$\begin{aligned}\text{Scr}(f) &= (\mathbf{y} - \mathbf{X}\mathbf{a})^\top (\mathbf{y} - \mathbf{X}\mathbf{a}) \\ &= (\mathbf{y}^\top - (\mathbf{X}\mathbf{a})^\top)(\mathbf{y} - \mathbf{X}\mathbf{a}) \\ &= \mathbf{y}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X}\mathbf{a} - (\mathbf{X}\mathbf{a})^\top \mathbf{y} + (\mathbf{X}\mathbf{a})^\top \mathbf{X}\mathbf{a} \\ &= \mathbf{y}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X}\mathbf{a} - \mathbf{a}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{a}^\top \mathbf{X}^\top \mathbf{X}\mathbf{a}\end{aligned}$$

- On a donc la solution suivante :

$$\begin{aligned}\nabla \text{Scr}(f) = \mathbf{0} &\Leftrightarrow 2\mathbf{X}^\top \mathbf{X}\mathbf{a} - 2\mathbf{X}^\top \mathbf{y} = \mathbf{0} \\ &\Leftrightarrow \hat{\mathbf{a}}_{mco} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}\end{aligned}$$

## Backgrd : Régression linéaire multiple et MCO (suite)

- Interprétation géométrique :



$$\hat{\mathbf{y}} = \underbrace{\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top}_{\text{Opérateur de projection}} \mathbf{y}$$

- Les MCO consistent à projeter orthogonalement  $\mathbf{y}$  sur le sous-espace de  $\mathbb{R}^n$  engendré par  $\{\mathbf{x}^0, \dots, \mathbf{x}^p\}$  (les  $p+1$  colonnes de  $\mathbf{X}$ ).
- Rq : comme on cherche à minimiser  $\text{Scr}(f)$ , on voit que la plus courte distance entre  $\mathbf{y}$  et le sous-espace est donnée par la projection orthogonale.

## Backgrd : Régression linéaire multiple et MCO (suite)

- Une fois estimé  $\hat{\mathbf{a}}_{mco}$  on peut calculer les prédictions du modèle pour un  $\mathbf{x}$  quelconque :

$$\hat{f}(\mathbf{x}) = \mathbf{x}^\top \hat{\mathbf{a}}_{mco} = \mathbf{x}^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

- Pour calculer l'erreur de prédiction on regarde ce que prédit le modèle estimé pour les données  $\mathbb{E}$  données par les lignes de  $\mathbf{X}$  :

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{a}}_{mco} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

- L'erreur de prédiction est donc donnée par :

$$\begin{aligned}\text{Scr}(\hat{f}) &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\ &= \|\mathbf{y} - \hat{\mathbf{y}}\|^2\end{aligned}$$

où  $\|\cdot\|$  est la norme euclidienne.

## Backgrd : Le modèle Gaussien

- Nous réinterprétons la régression linéaire multiple dans un cadre probabiliste. Nous avons le modèle suivant pour tout  $i = 1, \dots, n$  :

$$y_i = \mathbf{x}_i^\top \mathbf{a} + \epsilon_i$$

- Nous faisons de plus l'hypothèse que le vecteur  $\epsilon = (\epsilon_1, \dots, \epsilon_n) \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 \mathbf{I}_n)$  où  $\mathbf{I}_n$  est la matrice identité d'ordre  $n$ .
- Autrement dit les  $\epsilon_i$  sont iid. selon  $\mathcal{N}(0, \sigma^2)$ .
- On en déduit la relation suivante :

$$P(Y = y | X = \mathbf{x}, \mathbf{a}, \sigma^2) \sim \mathcal{N}(\mathbf{x}^\top \mathbf{a}, \sigma^2)$$

- L'étude de la régression linéaire multiple dans un cadre probabiliste permet d'introduire le principe d'inférence de **maximum de vraisemblance (MV)** ainsi que des propriétés statistiques des estimateurs associés.

## Backgrd : Le modèle Gaussien (suite)

- La **vraisemblance (likelihood)** est la probabilité d'observer l'échantillon :

$$\text{Vr}(\mathbf{a}, \sigma^2) = P(\mathcal{Y}_1 = y_1, \dots, \mathcal{Y}_n = y_n | X_1 = \mathbf{x}_1, \dots, X_n = \mathbf{x}_n; \mathbf{a}, \sigma^2)$$

- Les  $\mathcal{Y}_i$  sont supposés iid. nous avons alors :

$$\begin{aligned}\text{Vr}(\mathbf{a}, \sigma^2) &= \prod_{i=1}^n P(\mathcal{Y}_i = y_i | X_i = \mathbf{x}_i; \mathbf{a}, \sigma^2) \\ &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \left(\frac{y_i - \mathbf{x}_i^\top \mathbf{a}}{\sigma}\right)^2\right)\end{aligned}$$

- L'estimateur du MV est la valeur des paramètres qui maximise la probabilité d'observer l'échantillon. On résoud donc le problème :

$$\max_{(\mathbf{a}, \sigma^2) \in \mathbb{R}^{p+1} \times \mathbb{R}} \prod_{i=1}^n P(\mathcal{Y}_i = y_i | X_i = \mathbf{x}_i; \mathbf{a}, \sigma^2)$$

## Backgrd : Rappels de probabilités

- Nous rappelons des résultats qui nous seront utiles par la suite.
- Nous supposons ci-dessous que :

- **X est un vecteur aléatoire,**
- **b et A sont un vecteur et une matrice carrée donnés.**

- Nous avons alors les propriétés suivantes :

- Propriétés de linéarité de l'opérateur espérance :

$$\mathbb{E}_X \{AX + b\} = A\mathbb{E}_X \{X\} + b$$

- Propriété de l'opérateur variance :

$$\text{V}_X \{AX + b\} = A\text{V}_X \{X\} A^\top$$

## Backgrd : Le modèle Gaussien (suite)

- On maximise, de manière équivalente, le **log. de la vraisemblance** :

$$\begin{aligned}\text{Lvr}(\mathbf{a}, \sigma^2) &= \ln\left(\prod_i P(\mathcal{Y}_i = y_i | X_i = \mathbf{x}_i; \mathbf{a}, \sigma^2)\right) \\ &= \sum_i \ln(P(\mathcal{Y}_i = y_i | X_i = \mathbf{x}_i; \mathbf{a}, \sigma^2))\end{aligned}$$

- Dans le modèle Gaussien cela se réduit à :

$$\begin{aligned}\text{Lvr}(\mathbf{a}, \sigma^2) &= \sum_{i=1}^n \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \left(\frac{y_i - \mathbf{x}_i^\top \mathbf{a}}{\sigma}\right)^2\right)\right) \\ &= \sum_{i=1}^n \left(-\ln(\sqrt{2\pi\sigma^2}) - \frac{1}{2} \left(\frac{y_i - \mathbf{x}_i^\top \mathbf{a}}{\sigma}\right)^2\right) \\ &= -\frac{n}{2} \ln(2\pi) - \frac{n}{2} \ln(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \mathbf{a})^2\end{aligned}$$

- Nous avons la propriété suivante :  $\max \text{Vr}(\mathbf{a}, \sigma^2) \Leftrightarrow \min \text{Scr}(f)$

## Backgrd : Le modèle Gaussien (suite)

- Estimateur du MV :

$$\mathbf{a}_{mv} = \arg \min_{\mathbf{a} \in \mathbb{R}^{p+1}} \sum_{i=1}^n (\mathcal{Y}_i - \mathbf{x}_i^\top \mathbf{a})^2$$

- On a la solution analytique :

$$\mathbf{a}_{mv} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}, \quad \text{avec } \mathbf{Y} = (\mathcal{Y}_1, \dots, \mathcal{Y}_n)$$

- Espérance de  $\mathbf{a}_{mv}$  :

$$\begin{aligned}\mathbb{E}_{\mathbf{Y}|\mathbf{X}} \{\mathbf{a}_{mv} | \mathbf{X}\} &= \mathbb{E}_{\mathbf{Y}|\mathbf{X}} \left\{ (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} | \mathbf{X} \right\} \\ &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbb{E}_{\mathbf{Y}|\mathbf{X}} \{\mathbf{Y} | \mathbf{X}\} \\ &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{X} \mathbf{a} = \mathbf{a}\end{aligned}$$

- L'estimateur du MV est donc **sans biais**.

## Backgrd : Le Théorème de Gauss-Markov

- Variance de  $\mathbf{a}_{mv}$  :

$$\begin{aligned} V_{Y|X}\{\mathbf{a}_{mv}|X\} &= V_{Y|X}\left\{\left(X^T X\right)^{-1} X^T Y|X\right\} \\ &= \left(X^T X\right)^{-1} X^T V_{Y|X}\{Y|X\} X \left(X^T X\right)^{-1} \\ &= \sigma^2 \left(X^T X\right)^{-1} \end{aligned}$$

- Efficacité de l'estimateur du MV :

### Théorème. (Théorème de Gauss-Markov)

Sous l'hypothèse que le vecteur des résidus  $\epsilon = (\epsilon_1, \dots, \epsilon_n)$  vérifie  $E_\epsilon\{\epsilon\} = \mathbf{0}_n$  (espérance nulle) et  $V_\epsilon\{\epsilon\} = \sigma^2 I_n$  (variance constante et non corrélation), l'estimateur du MV (ou MCO)  $\mathbf{a}_{mv} = (X^T X)^{-1} X^T Y$  est, parmi les estimateurs linéaires (c.à.d. des fonctions linéaires des  $\{Y_i\}$ ) qui soient sans biais, celui de variance minimale.

## Régression linéaire multiple : AirBnB

- Dans l'ex. traité, nous voulons prédire le (logarithme) du prix de la nuitée d'une location à partir des caractéristiques de celle-ci.
- Nous n'intégrons pas dans un premier temps des var. associées à une info. géo. quelconque. La liste de var. sélectionnée est alors la suivante et donne simplement les caractéristiques intrinsèques des biens.

```

1 variable_names = [
2     "accommodates", # Number of people it accommodates
3     "bathrooms", # Number of bathrooms
4     "bedrooms", # Number of bedrooms
5     "beds", # Number of beds
6     # Below are binary variables, 1 True, 0 False
7     "rt_Private_room", # Room type: private room
8     "rt_Shared_room", # Room type: shared room
9     "pg_Condominium", # Property group: condo
10    "pg_House", # Property group: house
11    "pg_Other", # Property group: other
12    "pg_Townhouse", # Property group: townhouse
13 ]

```

## Rappel du Sommaire

### 4 Méthodes de régression spatiale

- Rappels en régression linéaire multiple
- Application de la rég. lin. multiple à des données spatiales
- Ajout de variables explicatives de nature géographique
- Modèles d'hétérogénéité spatiale
- Modèles de dépendance spatiale
- Geographically Weighted Regression

## \*\*Code\*\* : Rég. lin. mult., AirBnB

- Les lib. de référence en Python pour la modélisation stat. est statsmodels et pour le machine learning (incluant la rég. lin. mult.), sklearn. Mais, comme nous allons intégrer par la suite des info. spatiales, nous utiliserons plutôt la classe spreg de pysal.

```

1 from pysal.model import spreg
2
3 # Fit OLS model
4 m1 = spreg.OLS(
5     # Dependent variable
6     db[["log_price"]].values,
7     # Independent variables
8     db[variable_names].values,
9     # Dependent variable name
10    name_y="log_price",
11    # Independent variable name
12    name_x=variable_names
13 )
14
15 print(m1.summary)

```

- Cette fct. donne en sortie plusieurs stat./tests sur l'estimation du modèle à partir de l'attribut summary que nous utiliserons souvent.

## \*\*Code\*\* : Rég. lin. mult., AirBnB (suite)

- Statistiques sur la pertinence globale du modèle.

```

1 REGRESSION
2 -----
3 SUMMARY OF OUTPUT: ORDINARY LEAST SQUARES
4 -----
5 Data set : unknown
6 Weights matrix : None
7 Dependent Variable : log_price Number of Observations: 6110
8 Mean dependent var : 4.9958 Number of Variables : 11
9 S.D. dependent var : 0.8072 Degrees of Freedom : 6099
10 R-squared : 0.6683
11 Adjusted R-squared : 0.6678
12 Sum squared residual: 1320.148 F-statistic : 1229.0564
13 Sigma-square : 0.216 Prob(F-statistic) : 0
14 S.E. of regression : 0.465 Log likelihood : -3988.895
15 Sigma-square ML : 0.216 Akaike info criterion : 7999.790
16 S.E. of regression ML: 0.4648 Schwarz criterion : 8073.685

```

## \*\*Code\*\* : Rég. lin. mult., AirBnB (suite)

- Statistiques sur les hypothèses du modèle de régression linéaire Gaussien qui est en correspondance avec les MCO.

```

1 REGRESSION DIAGNOSTICS
2 MULTICOLLINEARITY CONDITION NUMBER 11.964
3
4 TEST ON NORMALITY OF ERRORS
5 TEST DF VALUE PROB
6 Jarque-Bera 2 2671.611 0.0000
7
8 DIAGNOSTICS FOR HETROSKEDEASTICITY
9 RANDOM COEFFICIENTS
10 TEST DF VALUE PROB
11 Breusch-Pagan test 10 322.532 0.0000
12 Koenker-Bassett test 10 135.581 0.0000
13 ===== END OF REPORT =====

```

## \*\*Code\*\* : Rég. lin. mult., AirBnB (suite)

- Statistiques sur la pertinence de chq. variable exogène.

	Variable	Coefficient	Std. Error	t-Statistic	Probability
4	CONSTANT	4.3883830	0.0161147	272.3217773	0.0000000
5	accommodates	0.0834523	0.0050781	16.4336318	0.0000000
6	bathrooms	0.1923790	0.0109668	17.5419773	0.0000000
7	bedrooms	0.1525221	0.0111323	13.7009195	0.0000000
8	beds	-0.0417231	0.0069383	-6.0134430	0.0000000
9	rt_Private_room	-0.5506868	0.0159046	-34.6244758	0.0000000
10	rt_Shared_room	-1.2383055	0.0384329	-32.2198992	0.0000000
11	pg_Condominium	0.1436347	0.0221499	6.4846529	0.0000000
12	pg_House	-0.0104894	0.0145315	-0.7218393	0.4704209
13	pg_Other	0.1411546	0.0228016	6.1905633	0.0000000
14	pg_Townhouse	-0.0416702	0.0342758	-1.2157316	0.2241342
15					

## Analyse des résultats de m1, AirBnB

- Globalement, l'ajustement des données est assez bon lorsque le critère du  $R^2$  est au moins égal à 2/3 ce qui est le cas ici avec  $R\text{-squared}=0.6683$ . En effet, cela veut dire que le modèle permet de restituer 67% de la variance de la var. à expliquer.
- Toutefois, lorsque l'on applique cette méthode, il est nécessaire d'analyser les résidus afin de vérifier si les hypothèses du modèle sont bien satisfaites : les erreurs doivent être i.i.d. selon une loi normale centrée de variance constante.
- Les sorties de la classe spreg du slide précédent sont très utiles à cet égard mais des analyses graphiques supplémentaires peuvent nous donner plus d'informations sur une éventuelle structure cachée.
- Dans le cas de nos données, on peut se demander si le prix de la nuitée d'une location de San Diego qui se trouve proche de la mer est sous-estimée par notre modèle ou pas. En effet, cette information géographique n'est pas intégrée dans le modèle m1.

## \*\*Code\*\* : Rég. lin. mult., analyse des rés., AirBnB

- On crée alors une var. permettant de distinguer les biens proches de la mer de ceux qui ne le sont pas.

```

1 # Create a Boolean (True/False) with whether a
2 # property is coastal or not
3 is_coastal = db.coastal.astype(bool)
4 # Split residuals (m1.u) between coastal and not
5 coastal = m1.u[is_coastal]
6 not_coastal = m1.u[~is_coastal]
7 # Create histogram of the distribution of coastal residuals
8 plt.hist(coastal, density=True, label="Coastal")
9 # Create histogram of the distribution of non-coastal residuals
10 plt.hist(
11     not_coastal,
12     histtype="step",
13     density=True,
14     linewidth=4,
15     label="Not Coastal",
16 )
17 # Add Line on 0
18 plt.vlines(0, 0, 1, linestyle=":", color="k", linewidth=4)
19 # Add legend
20 plt.legend()
21 # Display
22 plt.show()

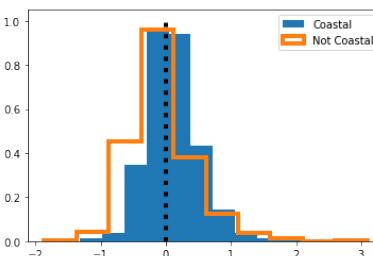
```

## Analyse des résultats de m1, AirBnB (suite)

- Le cas *coastal versus not coastal* est insuffisant pour inférer plus en détail le rôle d'une éventuelle structure spatiale. En effet, il se pourrait que certaines quartiers soient plus attractifs que d'autres en raison de facteurs latents ou non pris en compte dans le modèle m1 comme par ex. un effet d'une campagne de publicité ...
- Par ex. : malgré le fait que le quartier Camp Pendleton soit proche de la mer, il s'agit d'une base maritime dans le nord de la ville ce qui le pénalise en raison de la pollution et du bruit. Ce type d'information n'est pas dans les données. Une source externe d'information provenant d'un expert du domaine par exemple est alors fort utile pour intégrer ce type de connaissance.
- Cela suggère qu'il est intéressant d'analyser les résidus au sein de chaque quartier de San Diego afin d'approfondir notre étude sur l'impact de l'information géographique dans la modélisation du prix.
- Nous traçons les **boîtes à moustache** des prix dans chq. quartier.

## \*\*Code\*\* : Rég. lin. mult., analyse des rés., AirBnB (suite)

Extrait de [Rey et al., 2020]



- On peut compléter par un test de Student comparant les moyennes de ces deux distributions empiriques.

```

1 from scipy.stats import ttest_ind
2 ttest_ind(coastal, not_coastal)

```

```

1 Ttest_indResult(statistic=array
([13.98193858]), pvalue=array
([9.442438e-44]))

```

- On constate une structure de *clustering* spatial des erreurs : les distributions des locations proche de la côte *versus* non proche de la côte ont des allures différentes.

## \*\*Code\*\* : Rég. lin. mult., ana. rés. par quartier, AirBnB

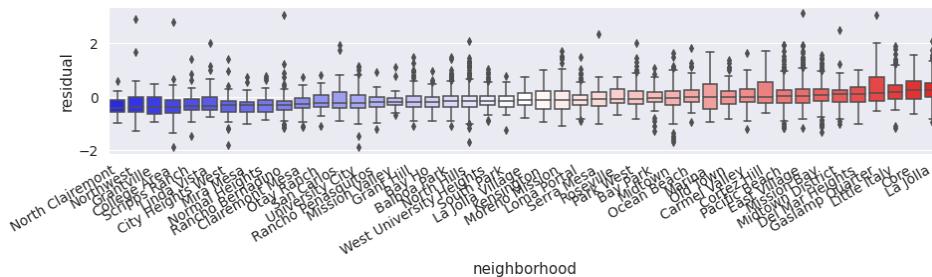
```

1 # Create column with residual values from m1
2 db["residual"] = m1.u
3 # Obtain the median value of residuals in each neighbourhood
4 medians = (
5     db.groupby("neighborhood")
6     .residual.median()
7     .to_frame("hood_residual")
8 )
9 # Increase fontsize
10 seaborn.set(font_scale=1.25)
11 # Set up figure
12 f = plt.figure(figsize=(15, 3))
13 # Grab figure's axis
14 ax = plt.gca()
15 # Generate boxplot of values by neighbourhood
16 # Note the data includes the median values merged on-the-fly
17 seaborn.boxplot(
18     x="neighborhood",
19     y="residual",
20     ax=ax,
21     data=db.merge(
22         medians, how="left", left_on="neighborhood", right_index=True
23     ).sort_values("hood_residual"),
24     palette="bwr",
25 )
26 # Auto-format of the X labels
27 f.autofmt_xdate()
28 # Display
29 plt.show()

```

## \*\*Code\*\* : Rég. lin. mult., ana. rés. par quartier, AirBnB (suite)

Extrait de [Rey et al., 2020]



- Les quartiers sont ordonnés selon la médiane de leurs résidus. *North Clairemont* est le quartier où le modèle fait, en médiane, le moins d'erreurs contrairement à *La Jolla*. Les quartiers à droite du graphe comme *Gaslamp Quarter*, *Little Italy* et *The Core* sont touristiques et cette information de popularité n'est pas intégrée dans le modèle.

## \*\*Code\*\* : ASG des erreurs de m1, AirBnB

- Nous allons utiliser une matrice de SW (*spatial weight*) binaire qui encode le 1 plus proche voisin (cf slides 64 et 125) et comparer l'erreur de chq. observation avec celle de son plus proche voisin.

```

1 knn = weights.KNN.from_dataframe(db, k=1)
2
3 lag_residual = weights.spatial_lag.lag_spatial(knn, m1.u)
4 ax = seaborn.regplot(
5     m1.u.flatten(),
6     lag_residual.flatten(),
7     line_kws=dict(color="orangered"),
8     ci=None,
9 )
10 ax.set_xlabel("Model Residuals - $u$")
11 ax.set_ylabel("Spatial Lag of Model Residuals - $W u$");
12 plt.show()

```

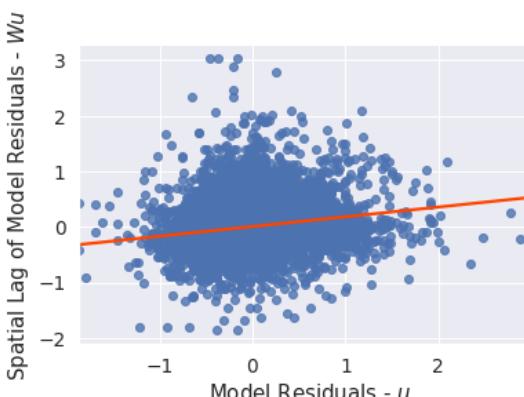
## Analyse des résultats de m1, AirBnB (suite)

- On s'aperçoit que ces 3 quartiers sont proches.
- La dépendance pourrait provenir d'un effet de *spillovers spatial* : ces quartiers étant adjacents il pourrait avoir un effet de "mimétisme" entre eux ce qui pourrait expliquer que leurs erreurs soient similaires.
- Pour tester cette hypothèse nous allons analyser s'il y a ou pas une forte ASG des erreurs à l'aide du graphe de Moran (cf slide 139).



## \*\*Code\*\* : ASG des erreurs de m1, AirBnB (suite)

Extrait de [Rey et al., 2020]



- La pente étant positive, nous pouvons donc dire que les erreurs de prédiction ont tendance à avoir une structure spatiale.

## \*\*Code\*\* : ASL des erreurs de m1, AirBnB

- Etant donné ce premier résultat, nous allons étudier maintenant un voisinage de cardinal 20 et tenter d'identifier cette fois-ci une ASL.

```

1 # Re-weight W to 20 nearest neighbors
2 knn.reweight(k=20, inplace=True)
3 # Row standardise weights
4 knn.transform = "R"
5 # Run LISA on residuals
6 outliers = esda.moran.Moran(m1.u, knn, permutations=9999)
7 # Select only LISA cluster cores
8 error_clusters = outliers.q % 2 == 1
9 # Filter out non-significant clusters
10 error_clusters &= outliers.p_sim <= 0.001
11 # Add 'error_clusters' and 'local_I' columns
12 ax = (
13     db.assign(error_clusters=error_clusters, local_I=outliers.I # Retain error
14         .clusters only
15     ).query("error_clusters" # Sort by I value to largest plot on top
16     ).sort_values("local_I" # Plot I values
17     ).plot("local_I", cmap="bwr", marker="."))
18 # Add basemap
19 contextily.add_basemap(ax, crs=db.crs)
20 # Remove axes
21 ax.set_axis_off()
22 plt.show()

```

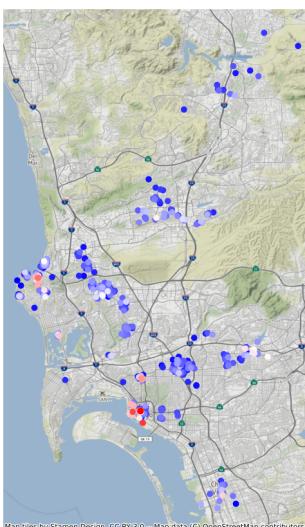
## Rappel du Sommaire

### 4 Méthodes de régression spatiale

- Rappels en régression linéaire multiple
- Application de la rég. lin. multiple à des données spatiales
- Ajout de variables explicatives de nature géographique
- Modèles d'hétérogénéité spatiale
- Modèles de dépendance spatiale
- Geographically Weighted Regression

## \*\*Code\*\* : ASL des erreurs de m1, AirBnB (suite)

Extrait de [Rey et al., 2020]



## Intégrer l'espace dans le modèle de régression

- Il existe plus. types de structures spatiales qu'il serait intéressant d'intégrer dans les modèles de régression.
- De nombreuses méthodes ont été proposées pour étendre des modèles de régression afin de tenir compte de ces structures spatiales. Le sous-domaine de la régression spatiale est ainsi riche de nombreuses approches.
- Nous explorons brièvement 4 types de modèle qui tiennent compte d'informations géographiques ou de voisinage :
  - L'ajout simple de variables spatiales.
  - Les modèles d'hétérogénéité spatiale (SFE, *Spatial Regimes*).
  - Les modèles de dépendance spatiale (SLX, SE, SLY).
  - Le modèle *Geographically Weighted Regression* (GWR).

## Ajout de variables spatiales ou de proximité

- L'idée centrale est d'exploiter des renseignements de nature géographique afin de construire de nouvelles variables encodant des informations spatiales qu'on ajoute simplement ("plugged in") comme facteur explicatif au modèle de régression linéaire multiple.
- Dans l'ex. traité, un site touristique central à San Diego est le Balboa park qui contient des musées, théâtres, cafés, restaurants, magasins et un zoo.
- Notre hypothèse est que les personnes souhaitant louer un bien sur AirBnB à San Diego sont prêtes à payer plus chers si celui-ci est proche du Balboa park.
- Si cette hypothèse est vraie alors ajouter une var. indiquant la distance de la location au Balboa park pourra améliorer les prédictions et on observera un coef. négatif associée à cette var. exogène.

## \*\*Code\*\* : m2, ajout de d2balboa, AirBnB

```

1 balboa_names = variable_names + ["d2balboa"]
2
3 m2 = spreg.OLS(
4     db[["log_price"]].values,
5     db[balboa_names].values,
6     name_y="log_price",
7     name_x=balboa_names,
8 )
9
10 pandas.DataFrame(
11     [[m1.r2, m1.ar2], [m2.r2, m2.ar2]],
12     index=["M1", "M2"],
13     columns=["R2", "Adj. R2"],
14 )

```

	R2	Adj. R2
M1	0.668345	0.667801
M2	0.668502	0.667904

- Le modèle m2 qui intègre d2balboa ne donne pas globalement des résultats meilleurs que m1.

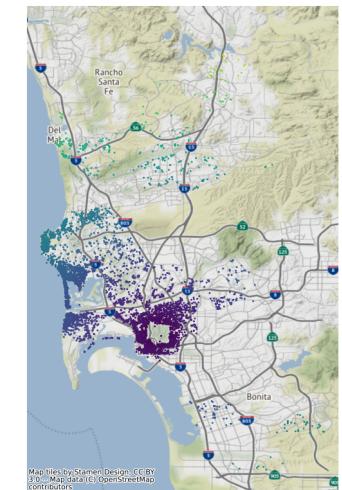
## \*\*Code\*\* : Ajout de d2balboa, AirBnB

- A cet effet, nous disposons dans les données de la var. d2balboa que nous allons donc intégrer dans le modèle de régression. Mais avant cela, nous visualisation cette information géographique.

```

1 ax = db.plot("d2balboa", marker=".", s=5)
2 contextily.add_basemap(ax, crs=db.crs)
3 ax.set_axis_off()
4
5 plt.show()

```



## \*\*Code\*\* : m2, ajout de d2balboa, AirBnB (suite)

```

1 # Set up table of regression coefficients
2 pandas.DataFrame(
3     {
4         # Pull out regression coefficients and
5         # flatten as they are returned as Nx1 array
6         "Coeff.": m2.betas.flatten(),
7         # Pull out P-values from t-stat object
8         "P-Value": [i[1] for i in m2.t_stat],
9     },
10    index=m2.name_x,
11 )

```

	Coeff.	P-Value
CONSTANT	4.379624	0.000000e+00
accommodates	0.083644	1.156896e-59
bathrooms	0.190791	9.120139e-66
bedrooms	0.150746	7.418035e-41
beds	-0.041476	2.394322e-09
rt_Private_room	-0.552996	2.680270e-240
rt_Shared_room	-1.235521	2.586867e-209
pg_Condominium	0.140459	2.803765e-10
pg_House	-0.013302	3.630396e-01
pg_Other	0.141176	6.309880e-10
pg_Townhouse	-0.045784	1.826992e-01
d2balboa	0.001645	8.902052e-02

- Le coef. de d2balboa obtenu infirme aussi notre hypothèse.

## Spatial heterogeneity

- Au lieu d'intégrer une var. commune à toutes les observations comme la distance du *Balboa park*, nous allons plutôt considérer le fait que les différents quartiers d'une ville n'ont pas les mêmes prix moyens et ce indépendamment qu'ils soient proches ou pas d'un site particulier comme le *Balboa park*.
- L'approche basée sur l'**hétérogénéité spatiale** suppose que le modèle de régression doit être flexible et en partie différent selon les régions géographiques afin de tenir compte des spécificités locales.
- Par ex., en supposant que le biais  $a_0$  peut être différent selon les quartiers de San Diego, cela nous permettrait de modéliser le fait que le "prix de base" de la nuité d'un quartier à un autre est distinct.
- Par ailleurs, supposer que les coef. des var. indépendantes peuvent changer selon la région permettrait d'indiquer que l'importance d'un facteur varie selon la position dans l'espace.
- Nous voyons ces deux approches respectives dans ce qui suit : *spatial fixed effects* et *spatial regimes*.

## Effets spatiaux fixes (SFE)

- Dans le cas des modèles *Spatial Fixed Effects* (SFE), on suppose que la position géographique joue un rôle mais qu'on n'observe pas certains facteurs permettant de caractériser cette variabilité spatiale.
- Dans ce contexte, on "synthétise" alors ces **facteurs latents** par une variable binaire représentant l'appartenance d'un individu à une région spécifique. L'ensemble de ces variables binaires permettent ainsi d'encoder la variabilité géographique au sein de la population.
- Formellement cela revient à considérer comme modèle :

$$f_{sfe}(\mathbf{x}) = \sum_{r=1}^s a_r^r \text{Ind}_{\{\mathbf{x} \in r\}} + \sum_{j=1}^p a_j x_j$$

où  $\text{Ind}$  est la fct. indicatrice et  $\mathbf{x} \in r$  indique que  $X$  est dans la région  $r$ . On suppose qu'il y a  $s$  régions distinctes.

## Rappel du Sommaire

### 4 Méthodes de régression spatiale

- Rappels en régression linéaire multiple
- Application de la rég. lin. multiple à des données spatiales
- Ajout de variables explicatives de nature géographique
- Modèles d'hétérogénéité spatiale
- Modèles de dépendance spatiale
- Geographically Weighted Regression

## Effets spatiaux fixes (SFE) (suite)

- Du point de vue spatial, on peut interpréter la fct.  $f$  comme la combinaison de plus. fcts  $f^r(\mathbf{x}) = a_r^r + \sum_{j=1}^p a_j x_j$ ; qui ont le même vecteur  $\mathbf{a} = (a_1, \dots, a_p)$  mais des biais  $a_0^r$  différents :

$$f_{sfe}(\mathbf{x}) = \sum_{r=1}^s f^r(\mathbf{x}) \text{Ind}_{\{\mathbf{x} \in r\}}$$

- Pour mettre en pratique ce modèle sur notre jeu de données, nous utiliserons deux librairies :
  - ▶ Dans le 1er cas, `statsmodels`, nous permet pédagogiquement de comprendre en pratique le lien entre les effets spatiaux fixes et l'intégration de variables binaires encodant l'appartenance d'une observation à un quartier.
  - ▶ Dans le 2ème cas, nous allons utiliser la classe `spreg` de `pysal`. Nous verrons comment mettre en place les effets spatiaux fixes à l'aide de cet outil. Les résultats obtenus seront similaires à ceux de `statsmodels` mais l'utilisation de `spreg` ici nous permettra ensuite de bien comprendre la généralisation aux cas des modèles de régimes spatiaux.

## \*\*Code\*\* : SFE, stasmodels, AirBnB

```

1 import statsmodels.formula.api as sm
2
3 f = (
4     "log_price ~ "
5     + " + ".join(variable_names)
6     + " + neighborhood - 1"
7 )
8 print(f)
9
10 m3 = sm.ols(f, data=db).fit()
11
12 # Store variable names for all the spatial fixed effects
13 sfe_names = [i for i in m3.params.index if "neighborhood[" in i]
14 # Create table
15 pandas.DataFrame(
16     {
17         "Coef.": m3.params[sfe_names],
18         "Std. Error": m3.bse[sfe_names],
19         "P-Value": m3.pvalues[sfe_names],
20     }
21 )
22
23 log_price ~ accommodates + bathrooms + bedrooms + beds + rt_Private_room + rt_Shared_
24 room + pg_Condominium + pg_House + pg_Other + pg_Townhouse + neighborhood - 1

```

## \*\*Code\*\* : SFE, spreg, AirBnB (suite)

- Le code précédent montre le lien entre les SFE et la régression linéaire multiple intégrant une variable qualitative nomimale encodant le quartier, représentée par une table disjonctive complète (*dummy variables*). Notons que la lib. statsmodels est généraliste et n'est pas spécifique aux données géoréférencées.
- Nous montrons comment obtenir ces mêmes résultats mais à l'aide d'outils développés dans le contexte de la régression spatiale en utilisant la classe spreg. Cette permet de mettre en oeuvre le cas général des modèles à régimes spatiaux et dont les SFE sont un cas particulier. En effet, les SFE représentent des régimes spatiaux où seuls les biais peuvent varier. De façon plus générale, les modèles de régimes spatiaux permettent d'avoir des coef. des var. explicatives distincts pour chaque région.
- Dans ce qui suit nous commençons donc par utiliser spreg dans le cas simple des SFE et nous verrons par la suite comment ce même outil permet de traiter le cas plus général des régimes spatiaux.

## \*\*Code\*\* : SFE, stasmodels, AirBnB (suite)

	Coeff.	Std. Error	P-Value
1 neighborhood[Balboa Park]	4.280766	0.033292	0.0
2 neighborhood[Bay Hol]	4.198251	0.076878	0.0
3 neighborhood[Bay Park]	4.329223	0.050987	0.0
4 neighborhood[Carmel Valley]	4.389261	0.056553	0.0
5 neighborhood[City Heights West]	4.053518	0.058378	0.0
6 neighborhood[Clairemont Mesa]	4.095259	0.047699	0.0
7 neighborhood[College Area]	4.033697	0.058258	0.0
8 neighborhood[Core]	4.726186	0.052643	0.0
9 neighborhood[Cortex Hill]	4.608090	0.051526	0.0
10 neighborhood[Del Mar Heights]	4.496910	0.054337	0.0
11 neighborhood[East Village]	4.545469	0.029373	0.0
12 neighborhood[Gaslamp Quarter]	4.775799	0.047304	0.0
13 neighborhood[Grant Hill]	4.306742	0.052365	0.0
14 neighborhood[Grantville]	4.053298	0.071396	0.0
15 neighborhood[Kensington]	4.302671	0.077176	0.0
16 neighborhood[La Jolla]	4.682084	0.025809	0.0
17 neighborhood[La Jolla Village]	4.330311	0.077237	0.0
18 neighborhood[Linda Vista]	4.191149	0.056916	0.0
19 neighborhood[Little Italy]	4.666742	0.046838	0.0
20 neighborhood[Loma Portal]	4.301909	0.033236	0.0
21 neighborhood[Marina]	4.558298	0.047994	0.0
22 neighborhood[Midtown]	4.366661	0.028394	0.0
23 neighborhood[Midtown District]	4.584938	0.065087	0.0
24 neighborhood[Mira Mesa]	3.989562	0.056101	0.0
25 neighborhood[Mission Bay]	4.515479	0.022422	0.0
26 ...			

- Les quartiers touristiques précédents ont des biais relativement élevés.

## \*\*Code\*\* : SFE, spreg, AirBnB (suite)

```

1 m4 = spreg.OLS_Regimes(
2     # Dependent variable
3     db[["log_price"]].values,
4     # Independent variables
5     db[variable_names].values,
6     # Variable specifying neighborhood membership
7     db["neighborhood"].tolist(),
8     # Allow the constant term to vary by group/regime
9     constant_regi="many",
10    # Variables to be allowed to vary (True) or kept
11    # constant (False). Here we set all to False
12    cols2regi=[False] * len(variable_names),
13    # Allow separate sigma coefficients to be estimated
14    # by regime (False so a single sigma)
15    regime_err_sep=False,
16    # Dependent variable name
17    name_y="log_price",
18    # Independent variables names
19    name_x=variable_names,
20 )
21 import numpy
22 numpy.round(m4.betas.flatten() - m3.params.values, decimals=12)

```

```

1 array([ 0.e+00, -0.e+00,  0.e+00,  0.e+00,  0.e+00,  0.e+00,
2        ...
3        -0.e+00, -0.e+00,  0.e+00,  0.e+00,  0.e+00])

```

- Les coefficients de m3 et m4 sont bien identiques.

## Analyse des résultats de m3 (ou m4), AirBnB

- Les SFE que l'on a intégré dans m3 implique qu'au lieu de considérer que toutes les locations sont dans l'absolu comparables étant données les var. explicatives  $X^1, \dots, X^p$ ; on considère plutôt le fait qu'elles sont comparables sur la base des var. explicatives  $X^1, \dots, X^p$  uniquement si elles sont du même quartier.

- Nous pouvons expliciter ce point de 2 façons distinctes :

- En rég. lin. mult., le coef.  $a_j$  indique l'effet de  $X^j$  sur  $Y$  "**toute chose étant égale par ailleurs**". Supposons que deux locations  $X_i$  et  $X_{i'}$  de vecteurs  $\mathbf{x}_i$  et  $\mathbf{x}_{i'}$  resp., soient identiques sauf pour  $X^0$  la région avec  $x_{i0} = r$  et  $x_{i'0} = r'$  et  $r \neq r'$ . On a alors :

$$f_{\text{sfe}}(\mathbf{x}_i) - f_{\text{sfe}}(\mathbf{x}_{i'}) = a_0^r + \sum_j a_j x_{ij} - (a_0^{r'} + \sum_j a_j x_{i'j}) = a_0^r - a_0^{r'}$$

On voit que le SFE permet d'isoler l'impact de la var. géographique indiquant le quartier dans lequel se situent les biens.

## \*\*Code\*\* : SFE, spreg, AirBnB (suite)

```

1 neighborhood_effects = m3.params.filter(like="neighborhood")
2
3 # Create a sequence with the variable names without
4 # 'neighborhood[' and ']'
5 stripped = neighborhood_effects.index.str.strip(
6     "neighborhood["
7 ).str.strip("]")
8 # Reindex the neighborhood_effects Series on clean names
9 neighborhood_effects.index = stripped
10 # Convert Series to DataFrame
11 neighborhood_effects = neighborhood_effects.to_frame("fixed_effect")
12 # Print top of table
13 print(neighborhood_effects.head())

```

	fixed_effect
Balboa Park	4.280766
Bay Ho	4.198251
Bay Park	4.329223
Carmel Valley	4.389261
City Heights West	4.053518

## Analyse des résultats de m3 (ou m4), AirBnB (suite)

- Nous pouvons expliciter ce point de 2 façons distinctes (suite) :

- Dans le même esprit, supposons deux locations  $X_i$  et  $X_{i'}$  de vecteurs d'attributs  $\mathbf{x}_i$  et  $\mathbf{x}_{i'}$  resp., qui soient dans le même quartier et qui soient aussi identiques pour toutes les var. exogènes sauf pour une seule,  $X^j$ , avec  $x_{ij} \neq x_{i'j}$ . On a dans ce cas :

$$f_{\text{sfe}}(\mathbf{x}_i) - f_{\text{sfe}}(\mathbf{x}_{i'}) = a_j(x_{ij} - x_{i'j})$$

Le SFE permet aussi d'isoler l'impact d'une var. explicative  $X^j$  lorsque l'on compare des locations d'un même quartier. Si nous n'avions pas intégré la var. qualitative nominale du quartier,  $X^0$  (comme dans m2) alors le coef.  $a_j$  aurait été biaisé par cet effet spatial (cf slide 215).

- Nous cherchons maintenant à visualiser sur une carte cette dépendance spatiale que nous supposons avoir pu capturer grâce à la variable qualitative nominale du quartier. Pour cela le code qui suit procède à quelques prétraitements permettant d'aligner les variables de notre modèle avec des informations géographiques.

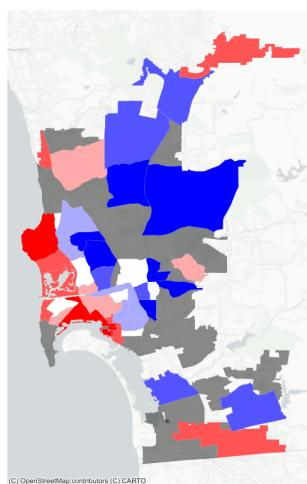
## \*\*Code\*\* : SFE, spreg, AirBnB (suite)

```

1 neighborhoods = geopandas.read_file("neighbourhoods.geojson")
2 # Plot base layer with all neighbourhoods in grey
3 ax = neighborhoods.plot(color="k", linewidth=0, alpha=0.5, figsize=(12, 6))
4 # Merge SFE estimates (note not every polygon receives an estimate since not every
5 # polygon contains AirBnb properties)
6 neighborhoods.merge(
7     neighborhood_effects,
8     how="left",
9     left_on="neighbourhood",
10    right_index=True
11 ).dropna(
12     subset=["fixed_effect"]
13     # Plot quantile choropleth
14 ).plot(
15     "fixed_effect", # Variable to display
16     scheme="quantiles", # Choropleth scheme
17     k=7, # No. of classes in the choropleth
18     linewidth=0.1, # Polygon border width
19     cmap="bwr", # Color scheme
20     ax=ax, # Axis to draw on
21 )
22 # Add basemap
23 contextily.add_basemap(
24     ax,
25     crs=neighborhoods.crs,
26     source=contextily.providers.CartoDB.PositronNoLabels)
27 # Remove axis
28 ax.set_axis_off()
29 plt.show()

```

## \*\*Code\*\* : SFE, spreg, AirBnB (suite)



- Les biais ( $a_0^r$ ) (c.à.d. les effets spatiaux fixes) les plus hauts sont en rouge et les plus bas sont en bleu (avec une discrémination de type quantile avec 7 intervalles).
- Les biais les plus forts ont tendance à correspondre à des quartiers de la côte tandis que les bas à ceux dans les terres.
- Le modèle avec effets spatiaux fixes permet donc de mettre en lumière une hétérogénéité spatiale des prix pratiqués selon les quartiers et selon si ils se trouvent près de la côté ou non.
- Rq. : ce résultat est à mettre en relation avec ceux du slide 198 mais précédemment c'étaient les erreurs du modèle (m1) que nous étudions.

## \*\*Code\*\* : Régimes spatiaux, AirBnB

- C'est la classe `OLS_Regimes` de `spreg` qui permet de mettre en oeuvre l'approche. Nous ne rentrerons pas dans les détails techniques d'inférence.

```

1 # PySal spatial regimes implementation
2 m5 = spreg.OLS_Regimes(
3     # Dependent variable
4     db[["log_price"]].values,
5     # Independent variables
6     db[variable_names].values,
7     # Variable specifying neighborhood membership
8     db[["coastal"]].tolist(),
9     # Allow the constant term to vary by group/regime
10    constant_regi="many",
11    # Allow separate sigma coefficients to be estimated
12    # by regime (False so a single sigma)
13    regime_err_sep=False,
14    # Dependent variable name
15    name_y="log_price",
16    # Independent variables names
17    name_x=variable_names,
18 )

```

## Régimes spatiaux

- L'hétérogénéité spatiale abordée précédemment était limitée aux biais. Nous voyons le cas général connu sous le vocable de **régimes spatiaux** dans le cadre duquel, les **coefficients des var. indépendantes peuvent variées selon une dimension spatiale**.
- Formellement on s'intéresse au modèle suivant :

$$f_{rs}(\mathbf{x}) = \sum_{r=1}^s (a_0^r + \sum_{j=1}^p a_j^r x_j) \text{Ind}_{\{\mathbf{x} \in r\}}$$

où  $\text{Ind}$  est la fct. indicatrice et  $\mathbf{x} \in r$  indique que  $X$  est dans la rég.  $r$ .

- Afin d'illustrer cette approche sur notre jeu de données, nous utiliserons l'opérateur de "différentiation spatiale" indiquant si un bien est dans un quartier de la côte ou pas (`coastal_neig`) afin de définir des régimes.
- Ce choix est motivé par les observations empiriques précédentes où on suppose qu'un individu est prêt à payer plus cher chaque caractéristique d'un bien si celui-ci est proche de la mer.

oeuvre l'approche.

## \*\*Code\*\* : Régimes spatiaux, AirBnB

- L'attribut `summary` de `m5` permet de visualiser les différentes estimations et tests permettant de valider ou infirmer certaines hypothèses relatives à l'approche (notamment la significativité des régimes). Nous affichons partiellement la sortie.

```

1 print(m5.summary)

REGRÉSSION
-----
SUMMARY OF OUTPUT: ORDINARY LEAST SQUARES - REGIMES
-----
5 Data set           : unknown
6 Weights matrix     : None
7 Dependent Variable : log_price
8 Mean dependent var : 4.9958
9 S.D. dependent var : 0.8072
10 R-squared          : 0.6853
11 Adjusted R-squared : 0.6843
12 Sum squared residual: 1252.489
13 Sigma-square        : 0.206
14 S.E. of regression   : 0.484
15 Sigma-square ML      : 0.205
16 S.E. of regression ML: 0.4528

```

## \*\*Code\*\* : Régimes spatiaux, AirBnB

Variable	Coefficient	Std.Error	t-Statistic	Probability
0_CONSTANT	4.4072424	0.0215156	204.8392695	0.0000000
0_accommodates	0.0901860	0.0064737	13.9311338	0.0000000
0_bathrooms	0.1433760	0.0142680	10.0487871	0.0000000
0_bedrooms	0.1129626	0.0138273	8.1695568	0.0000000
0_beds	-0.0262719	0.0088380	-2.9726102	0.0029644
0_rt_Private_room	-0.5293343	0.0189179	-27.9805699	0.0000000
0_rt_Shared_room	-1.2244586	0.0425969	-28.7452834	0.0000000
0_pg_Condominium	0.1053065	0.0281309	3.7434523	0.0001832
0_pg_House	-0.0454471	0.0179571	-2.5308637	0.0114032
0_pg_Other	0.0607526	0.0276365	2.1982715	0.0279673
0_pg_Townhouse	-0.0103973	0.0456730	-0.2276456	0.8199294
1_CONSTANT	4.4799043	0.0250938	178.5260014	0.0000000
1_accommodates	0.0484639	0.0078806	6.1497397	0.0000000
1_bathrooms	0.2474779	0.0165661	14.9388057	0.0000000
1_bedrooms	0.1897404	0.0179229	10.5864676	0.0000000
1_beds	-0.0506077	0.0107429	-4.7107925	0.0000025
1_rt_Private_room	-0.5586281	0.0283122	-19.7309699	0.0000000
1_rt_Shared_room	-1.0528541	0.0841745	-12.5079997	0.0000000
1_pg_Condominium	0.2044470	0.0339434	6.0231780	0.0000000
1_pg_House	0.0753534	0.0233783	3.2232188	0.0012743
1_pg_Other	0.2954848	0.0386455	7.6460385	0.0000000
1_pg_Townhouse	-0.0735077	0.0493672	-1.4889984	0.1365396

## \*\*Code\*\* : Régimes spatiaux, AirBnB (suite)

- La table précédente n'est pas très pratique afin de comparer les estimations selon les deux régimes.
- On extrait et stocke les estimations dans la variable `res` afin de les retravailler pour les visualiser différemment.

```

1 # Results table
2 res = pandas.DataFrame(
3     {
4         # Pull out regression coefficients and
5         # flatten as they are returned as Nx1 array
6         "Coeff.": m5.betas.flatten(),
7         # Pull out and flatten standard errors
8         "Std. Error": m5.std_err.flatten(),
9         # Pull out P-values from t-stat object
10        "P-Value": [i[1] for i in m5.t_stat],
11    },
12    index=m5.name_x,
13 )

```

## \*\*Code\*\* : Régimes spatiaux, AirBnB (suite)

- Le code qui suit permet d'extraire les résultats en fonction de l'opérateur de différentiation spatiale considéré et de les comparer en les visualisant côté à côté.

```

1 # Coastal regime
2 ## Extract variables for the coastal regime
3 coastal = [i for i in res.index if "1_" in i]
4 ## Subset results to coastal and remove the 1_ underscore
5 coastal = res.loc[coastal, :].rename(lambda i: i.replace("1_", ""))
6 ## Build multi-index column names
7 coastal.columns = pandas.MultiIndex.from_product(
8     [["Coastal"], coastal.columns]
9 )
10 # Non-coastal model
11 ## Extract variables for the non-coastal regime
12 ncoastal = [i for i in res.index if "0_" in i]
13 ## Subset results to non-coastal and remove the 0_ underscore
14 ncoastal = res.loc[ncoastal, :].rename(lambda i: i.replace("0_", ""))
15 ## Build multi-index column names
16 ncoastal.columns = pandas.MultiIndex.from_product(
17     [["Non-coastal"], ncoastal.columns]
18 )
19 # Concat both models
20 pandas.concat([coastal, ncoastal], axis=1)

```

## \*\*Code\*\* : Régimes spatiaux, AirBnB (suite)

	Coastal			Non-coastal		
	Coeff.	Std. Error	p-value	Coeff.	Std. Error	p-value
CONSTANT	4.479904	0.025094	0.0000000e+00	4.407242	0.021516	0.0000000e+00
accommodates	0.048464	0.007881	8.253761e-10	0.090186	0.006474	1.893020e-01
bathrooms	0.247478	0.016566	1.381278e-49	0.143376	0.014268	1.418804e-01
bedrooms	0.189740	0.017923	5.783965e-26	0.112963	0.013827	3.731742e-01
beds	-0.050608	0.010743	2.522348e-06	-0.026272	0.008838	2.964354e-01
rt_Private_room	-0.558628	0.028312	4.723759e-84	-0.529334	0.018918	3.546091e-01
rt_Shared_room	-1.052854	0.084174	1.836512e-35	-1.224459	0.042597	1.657163e-01
pg_Condominium	0.204447	0.033943	1.810152e-09	0.105307	0.028131	1.831822e-01
pg_House	0.075353	0.023378	1.274269e-03	-0.045447	0.017957	1.140318e-01
pg_Other	0.295485	0.038645	2.394157e-14	0.060753	0.027637	2.796727e-01
pg_Townhouse	-0.073508	0.049367	1.365396e-01	-0.010397	0.045673	8.199294e-01

## \*\*Code\*\* : Régimes spatiaux, test de Chow, AirBnB

- Le **test de Chow** permet de tester l'hypothèse nulle que les estimations provenant des différents régimes sont identiques. Si cette hypothèse est rejetée alors cela veut dire que la différenciation des régimes est pertinente pour la modélisation et la prise en compte de l'information spatiale. Les résultats des tests de Chow global et unitaires (pour chq. var. explicative) sont disponibles dans m5.summary.

REGIMES DIAGNOSTICS - CHOW TEST				
	VARIABLE	DF	VALUE	PROB
1	CONSTANT	1	4.832	0.0279
2	accommodates	1	16.736	0.0000
3	bathrooms	1	22.671	0.0000
4	bedrooms	1	11.504	0.0007
5	beds	1	3.060	0.0802
6	pg_Condominium	1	5.057	0.0245
7	pg_House	1	16.793	0.0000
8	pg_Other	1	24.410	0.0000
9	pg_Townhouse	1	0.881	0.3480
10	rt_Private_room	1	0.740	0.3896
11	rt_Shared_room	1	3.309	0.0689
12	Global test	11	328.869	0.0000
13				
14				

## Introduction

- Dans les modèles d'hétérogénéité spatiale précédents, l'information géographique était directement liée au phénomène à l'étude et était responsable de la variation de la variable endogène et des structures de *clustering*.
- Dans ce qui suit, c'est davantage la configuration spatiale des observations et l'impact des voisins sur la valeur de la variable à expliquer en un point qui nous intéresse.
- Par ex. : on peut supposer que le prix d'une location dépend de la nature du bien s'il s'agit d'un appartement ou d'une maison mais aussi du fait qu'une maison est entourée ou pas d'autres maisons ou si un appartement est entouré ou pas de bcp d'immeubles composés d'appartements. (quartiers historiques et pittoresques *versus* quartiers modernes de grands immeubles et "anonyme").

## Rappel du Sommaire

### 4 Méthodes de régression spatiale

- Rappels en régression linéaire multiple
- Application de la rég. lin. multiple à des données spatiales
- Ajout de variables explicatives de nature géographique
- Modèles d'hétérogénéité spatiale
- Modèles de dépendance spatiale
- Geographically Weighted Regression

## Introduction (suite)

- Vis-à-vis de notre étude de cas, nous allons donc nous intéresser non seulement à prédir le prix de la nuitée en fonction des caractéristiques du bien mais aussi en fonction des caractéristiques des biens dans son voisinage. Ceci est différent des modèles précédents dans la mesure où les caractéristiques des locations dans un voisinage ne sont pas liées à une quelconque information externe de nature géographique.
- Ce type de relation où les obs. en une position dépendent des obs. dans un certain voisinage est appelé **dépendance spatiale**.
- Il existe plusieurs modèles permettant de tenir compte de la dépendance spatiale entre individus (géoréférencés ou pas) avec des degrés de sophistication plus ou moins grands. Mais un instrument en commun à ces approches est la **matrice de SW spatial weight W**, que nous avons déjà introduite (cf slide 46).
- Nous verrons, sans entrer dans les détails, les modèles suivants : *Spatial Lag of X (SLX), Spatial error (SE), Spatial Lag (of Y) (SLY)*.

## Modèle Spatial Lag of X (SLX)

- Dans le **modèle SLX**, on ajoute comme variables explicatives des transformations de ces dernières données par un **opérateur de décalage spatial** associé à une matrice de SW (cf slide 127).
- Il est important de retenir que dans SLX ce sont des **lissages de variables exogènes**  $\{X^j\}_{j=1,\dots,p}$  qui peuvent être ajoutés dans l'ensemble des var. explicatives.
- Formellement, étant donnée une matrice de SW  $\mathbf{W} = (w_{ii'})$ , nous avons le modèle suivant :

$$f_{SLX}(\mathbf{x}_i) = a_0 + \sum_{j=1}^p a_j x_{ij} + \sum_{j=1}^p b_j \left( \sum_{i'=1}^n w_{ii'} x_{i'j} \right)$$

- En posant  $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})$ ,  $\mathbf{a} = (a_1, \dots, a_p)$ ,  $\mathbf{b} = (b_1, \dots, b_p)$ ,  $\mathbf{X} = (x_{ij})$  et  $[\mathbf{WX}]_i = (\sum_{i'} w_{ii'} x_{i'1}, \dots, \sum_{i'} w_{ii'} x_{i'p})$  (le vec. colonne de  $\mathbb{R}^p$  issu de la  $i$ ème ligne de  $\mathbf{WX} \in \mathbb{R}^{n \times p}$ ), on a aussi :

$$f_{SLX}(\mathbf{x}_i) = a_0 + \mathbf{a}^\top \mathbf{x}_i + \mathbf{b}^\top [\mathbf{WX}]_i$$

## \*\*Code\*\* : SLX, AirBnB

- Dans notre étude de cas, nous allons lisser uniquement les variables dont l'intitulé commence par "pg\_" qui sont des variables binaires indiquant le type de location (appartement, maison, ...). Les var. lissées donneront pour chq. location le nb. moyen de chq. type de location dans son voisinage.

```

1 # Select only columns in 'db' containing the keyword 'pg_'
2 wx = (
3     db.filter(
4         like="pg_"
5         # Compute the spatial lag of each of those variables
6     )
7     .apply(
8         lambda y: weights.spatial.lag.lag_spatial(knn, y)
9         # Rename the spatial lag, adding w_ to the original name
10    )
11    .rename(
12        columns=lambda c: "w_"
13        + c
14        # Remove the lag of the binary variable for apartments
15    )
16    .drop("w_pg_Apartment", axis=1)
17 )

```

## Modèle Spatial Lag of X (SLX) (suite)

- Interprétations des variables lissées et de leur coefficient :
  - Prenons le coefficient  $b_j$  de la var. lissée  $[\mathbf{WX}]^j = (\sum_{i'}, w_{1i'} x_{i'j}, \dots, \sum_{i'}, w_{ni'} x_{i'j})$  (le vec. colonne de  $\mathbb{R}^n$  issu de la  $j$ ème colonne de  $\mathbf{WX} \in \mathbb{R}^{n \times p}$ ).
  - Supposons que pour une location  $X_i$  donnée, la valeur  $[\mathbf{WX}]_{ij}$  augmente d'une unité alors cela veut dire que dans le voisinage de  $X_i$  la valeur lissée de  $X^j$  a augmenté d'une unité. Dans ce cas, selon le modèle, le prix de la location de  $X_i$  varira de  $b_j$ .
- Impact de la variation d'une valeur de  $X^j$  :
  - Prenons une var. explicative (non lissée)  $X^j$  et un individu de l'échantillon  $X_i$  et supposons que la valeur  $x_{ij}$  augmente d'une unité. L'impact de ce changement est indirect et pluriel. Il concerne l'ensemble des individus dont  $X_i$  est un voisin, c'est à dire l'ens. des  $X_i'$  pour lesquels  $w_{ii'} \neq 0$ . L'effet est indirect et intervient par le biais de la var. lissée  $[\mathbf{WX}]^j$ . On dit que  $X_i$  a un **spillover effect (sur son voisinage)**.

## \*\*Code\*\* : SLX, AirBnB (suite)

- En pratique, une fois déterminée les **variables lissées**, il suffit de les **ajouter dans le modèle sous forme de var. explicatives** et d'appliquer classiquement la régression linéaire multiple par MCO.

```

1 # Merge original variables with the spatial lags in 'wx'
2 slx_exog = db[variable_names].join(wx)
3 # Fit linear model with 'spreg'
4 m6 = spreg.OLS(
5     # Dependent variable
6     db[["log_price"]].values,
7     # Independent variables
8     slx_exog.values,
9     # Dependent variable name
10    name_y="log_price",
11    # Independent variables names
12    name_x=slx_exog.columns.tolist(),
13 )

```

- Comme précédemment, l'attribut **summary** permet d'obtenir sous forme d'un rapport les résultats de l'estimation du modèle.
- Cependant pour mettre davantage en avant les différences de **m6** avec les modèles précédents, nous retravaillons les sorties pour les afficher différemment.

## \*\*Code\*\* : SLX, AirBnB (suite)

```

1 # Collect names of variables of interest
2 vars_of_interest = (
3     db[variable_names].filter(like="pg_").join(wx).columns
4 )
5 # Build full table of regression coefficients
6 pandas.DataFrame(
7     {
8         # Pull out regression coefficients and
9         # flatten as they are returned as Nx1 array
10        "Coeff.": m6.betas.flatten(),
11        # Pull out and flatten standard errors
12        "Std. Error": m6.std_err.flatten(),
13        # Pull out P-values from t-stat object
14        "P-Value": [i[1] for i in m6.t_stat],
15    },
16    index=m6.name_x
17    # Subset for variables of interest only and round to
18    # four decimals
19 ).reindex(vars_of_interest).round(4)

```

## \*\*Code\*\* : SLX, AirBnB (suite)

	Coeff.	Std. Error	P-Value
2 pg_Condominium	0.1063	0.0222	0.0000
3 pg_House	0.0328	0.0157	0.0368
4 pg_Other	0.0862	0.0240	0.0003
5 pg_Townhouse	-0.0277	0.0338	0.4130
6 w_pg_Condominium	0.5928	0.0690	0.0000
7 w_pg_House	-0.0774	0.0319	0.0152
8 w_pg_Other	0.4851	0.0551	0.0000
9 w_pg_Townhouse	-0.2724	0.1223	0.0260

- Les var. lissées sont celle commençant par `w_pg_`. On voit qu'à l'exception de `w_pg_Townhouse` (maisons de ville), toutes les var. ajoutées sont unitairement significatives indiquant que notre hypothèse que le type de bien dans le voisinage a un impact sur le prix de la nuitée se tient.
- Etant donnée une var.  $X^j$ , utiliser dans le modèle à la fois les valeurs initiales et les valeurs lissées permet de différencier les **effets directs** et **indirects** de celle-ci sur la var. endogène.

## Modèle Spatial Error (SE)

- Le **modèle SE** intègre une transformation par l'**opérateur de décalage spatial du terme d'erreur** (et non pas des var. explicatives).
- Rappelons le modèle de rég. lin. mult. (cf slide 177) :

$$Y = f(X^1, \dots, X^p) + \epsilon = a_0 + \sum_{j=1}^p a_j X^j + \epsilon$$

- Etant donné une matrice de SW  $\mathbf{W} = (w_{ii'})$ , le modèle SE se formalise alors comme suit :

$$f_{se}(\mathbf{x}_i) = a_0 + \sum_{j=1}^p a_j x_{ij} + u_i$$

$$u_i = \lambda \sum_{i'} w_{ii'} u_{i'} + \epsilon_i$$

## \*\*Code\*\* : SE, GMM, AirBnB

- Contrairement aux modèles précédents utilisant les MCO (ou le MV), le modèle SE ne peut pas être estimé par cette méthode car il **ne respecte pas les hypothèses i.i.d.** de celle-ci.
- Des techniques d'inférence spécifiques sont alors utilisées comme la **méthode des moments généralisée** (*Generalized Methods of Moments* -GMM-). Nous n'aborderons pas celles-ci.
- Nous voyons en pratique comment obtenir ces estimations en Python en utilisant la classe `spreg` mais avec des méthodes spécifiques.

```

1 # Fit spatial error model with 'spreg'
2 # (GMM estimation allowing for heteroskedasticity)
3 m7 = spreg.GM_Error_Het(
4     # Dependent variable
5     db[["log_price"]].values,
6     # Independent variables
7     db[variable_names].values,
8     # Spatial weights matrix
9     w=knn,
10    # Dependent variable name
11    name_y="log_price",
12    # Independent variables names
13    name_x=variable_names,
14 )

```

## \*\*Code\*\* : SE, GMM, AirBnB (suite)

- Pour aller à l'essentiel et valider le bien fondé du modèle, nous extrayons l'estimation et le résultat du test de significativité du paramètre  $\lambda$  (coefficient du lissage des erreurs).

```

1 # Build full table of regression coefficients
2 pandas.DataFrame(
3     {
4         # Pull out regression coefficients and
5         # flatten as they are returned as Nx1 array
6         "Coeff.": m7.betas.flatten(),
7         # Pull out and flatten standard errors
8         "Std. Error": m7.std_err.flatten(),
9         # Pull out P-values from t-stat object
10        "P-Value": [i[1] for i in m7.z_stat],
11    },
12    index=m7.name_x
13    # Subset for lambda parameter and round to
14    # four decimals
15 ).reindex(["lambda"]).round(4)

```

	Coeff.	Std. Error	P-Value
lambda	0.6449	0.0187	0.0

- $\lambda$  étant stat. signif. dif. de 0 cela confirme que  $m7$  a de l'intérêt.

## \*\*Code\*\* : SLY, TS-LS, AirBnB

```

1 # Fit spatial lag model with 'spreg'
2 # (GMM estimation)
3 m8 = spreg.GM_Lag(
4     # Dependent variable
5     db[["log_price"]].values,
6     # Independent variables
7     db[variable_names].values,
8     # Spatial weights matrix
9     w=knn,
10    # Dependent variable name
11    name_y="log_price",
12    # Independent variables names
13    name_x=variable_names,
14 )

```

- Les résultats sont à nouveau accessibles par l'attribut `summary`.

## Modèle *Spatial Lag* (of $Y$ ) (SLY)

- Le modèle **SLY** intègre une transformation par l'**opérateur de décalage spatial de la var. dépendante  $Y$**  (et non pas des var. indépendantes ou des résidus).
- Etant donné une matrice de SW  $\mathbf{W} = (w_{ii'})$ , le modèle SLY se formalise alors comme suit :

$$f_{sly}(\mathbf{x}_i) = a_0 + \sum_{j=1}^p a_j x_{ij} + \rho \sum_{i'} w_{ii'} y_{i'} + \epsilon_i$$

- Ce modèle **ne respecte pas l'hypothèse d'exogénéité** puisque la var. endogène  $Y$  est utilisé comme var. explicative. Comme précédemment, les MCO ne peuvent donc pas être employées.
- Ici aussi des techniques d'inférence spécifiques ont été proposées comme la **méthode des doubles moindres carrés** (*two-stage least square*) utilisée en économétrie dans le cas de l'usage de var. instrumentales. Nous ne développerons pas cette approche mais montrons néanmoins comment la mettre en pratique avec Python.

## \*\*Code\*\* : SLY, TS-LS, AirBnB (suite)

- Le code suivant permet à nouveau d'extraire les résultats les éléments qui nous intéressent en particulier et de les présenter de façon simple.

```

1 # Build full table of regression coefficients
2 pandas.DataFrame(
3     {
4         # Pull out regression coefficients and
5         # flatten as they are returned as Nx1 array
6         "Coeff.": m8.betas.flatten(),
7         # Pull out and flatten standard errors
8         "Std. Error": m8.std_err.flatten(),
9         # Pull out P-values from t-stat object
10        "P-Value": [i[1] for i in m8.z_stat],
11    },
12    index=m8.name_z
13    # Round to four decimals
14 ).round(4)

```

## \*\*Code\*\* : SLY, TS-LS, AirBnB (suite)

	Coeff.	Std. Error	P-Value
CONSTANT	2.7440	0.0727	0.0000
accommodates	0.0698	0.0048	0.0000
bathrooms	0.1627	0.0104	0.0000
bedrooms	0.1604	0.0105	0.0000
beds	-0.0365	0.0065	0.0000
rt_Private_room	-0.4981	0.0151	0.0000
rt_Shared_room	-1.1157	0.0366	0.0000
pg_Condominium	0.1073	0.0209	0.0000
pg_House	-0.0004	0.0137	0.9766
pg_Other	0.1208	0.0215	0.0000
pg_Townhouse	-0.0186	0.0323	0.5653
W_log_price	0.3416	0.0148	0.0000

- Pour aller à l'essentiel : la var. lissée `w_log_price` a un coef. qui est stat. signif. différent de 0 ce qui valide l'intérêt du modèle SLY m8.
- Il faut toutefois faire attention à l'interprétation quant à l'impact indirect d'une variation de  $Y$ .
- Dans ce type d'approche, il est important de compléter l'analyse par des simulations de prédiction étant donnés plusieurs scénarii. Ceci est également valable pour le modèle SLX.

## Modèle *Geographically Weighted Regression* (GWR)

- Dans les modèles de régression SLX, SE et SLY précédents :
  - Les coefficients sont fixes sur l'ensemble du domaine considéré : le vecteur  $\mathbf{a}$  est le même en chaque localisation. Ces modèles font l'**hypothèse de stationnarité des relations spatiales entre var.**
  - L'ajout de valeurs moyennées sur le voisinage (des var. explicatives et/ou des erreurs et/ou à expliquer) permet d'intégrer des informations spatiales mais avec l'**hypothèse que les coefficients sont constants**.
- Dans le modèle **GWR Geographically Weighted Regression** [Brunsdon et al., 1996] :
  - Les relations entre les variables peuvent variées d'une localisation à une autre.** Le modèle permet une analyse locale détaillée des dépendances spatiales. L'hypothèse de stationnarité est relâchée.
  - Il y a autant de vecteurs de coefficients  $\mathbf{a}^i$  qu'il y a de localisation  $i = 1, \dots, n$  dans l'échantillon.
  - Le principe clé est celui de régression locale pondérée selon la distance des voisins.

## Rappel du Sommaire

### 4 Méthodes de régression spatiale

- Rappels en régression linéaire multiple
- Application de la rég. lin. multiple à des données spatiales
- Ajout de variables explicatives de nature géographique
- Modèles d'hétérogénéité spatiale
- Modèles de dépendance spatiale
- Geographically Weighted Regression

## Modèle *Geographically Weighted Regression* (GWR) (suite)

- Formellement le **modèle GWR** est le suivant :

$$f(\mathbf{x}_i) = a_0^i + \sum_{j=1}^p a_j^i x_{ij} + \epsilon_i, \quad \forall i = 1, \dots, n$$

où  $\mathbf{a}^i = (a_0^i, \dots, a_p^i)$  est le vect. de coef. pour l'objet géo.  $X_i$ .

- Notons que le modèle de régression linéaire usuel suppose  $\mathbf{a}^i = \mathbf{a}, \forall i$ .
- La méthode d'inférence en chaque  $X_i$  est de type **moindres carrés pondérés** où on veut minimiser :

$$\text{Scp}(\mathbf{a}^i) = \sum_{i'=1}^n (y_{i'} - \sum_{j=0}^p a_j^i x_{i'j})^2 w_{ii'}$$

où  $\{w_{ii'}\}_{i'=1, \dots, n}$  est l'ens. de poids de la ligne  $i$  de la mat. de SW  $\mathbf{W}$ .

- Comme chaque ligne de  $\mathbf{W}$  est distincte on a donc des  $\mathbf{a}^i$  différents.

## Modèle Geographically Weighted Regression (GWR) (suite)

- La solution analytique des moindres carrés pondérés est donnée par :

$$\hat{\mathbf{a}}_{gwr}^i = (\mathbf{X}^\top \mathbf{W}^i \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W}^i \mathbf{y}, \quad \forall i = 1, \dots, n$$

où  $\mathbf{W}^i$  est diagonale avec  $w_{ii'}^i = w_{ii'}, \forall i' = 1, \dots, n$  avec  $\mathbf{W} = (w_{ii'})_{i,i'=1,\dots,n}$  la matrice de SW.

- La matrice de SW peut être :
  - binaire basée sur la contiguïté ou les k plus proches voisins (cf slides 48 et 64),
  - non négative basée sur une fonction noyau (cf slide 67),
  - hybride (cf slide 86).
- GWR est très proche du modèle de régression locale (LOWESS *Locally WEighted Scatterplot Smoothing*). Dans LOWESS la matrice de SW est définie dans l'espace des variables alors que dans GWR,  $\mathbf{W}$  est définie du point de vue géographique.

## \*\*Code\*\* : GWR, AirBnB

- Le package `mgwr` permet de mettre en oeuvre la méthode GWR.
- Il fournit également des fonctions pour estimer la matrice de SW. Le noyau `bisquare` est utilisé par défaut :

$$w_{ii'} = \begin{cases} 1 - \frac{D^2(X_i, X_{i'})}{h} & \text{si } D^2(X_i, X_{i'}) \leq h \\ 0 & \text{sinon} \end{cases}$$

- La fonction `Sel_Bw` implémente l'approche décrite dans l'ouvrage de référence pour estimer la bande h [Fotheringham et al., 2009].

```

1 from mgwr.gwr import GWR
2 from mgwr.sel_bw import Sel_BW
3 # Load your data
4 coords = numpy.array(list(zip(db.geometry.x, db.geometry.y)))
5 # Extract dependent and independent variables
6 y = db[["log_price"]].values
7 X = db[variable_names].values
8 # Select optimal bandwidth
9 sel_bw = Sel_BW(coords, y, X)
10 bw = sel_bw.search(criterion='BIC')
11 # Fit GWR model
12 gwr_model = GWR(coords, y, X, bw)
13 gwr_results = gwr_model.fit()

```

## Modèle Geographically Weighted Regression (GWR) (suite)

- L'efficacité de GWR dépend de la définition de la matrice de SW.
  - Supposons une fonction noyau Gaussien,  $\mathbf{W} = (w_{ii'})$  est donnée par :

$$w_{ii'} = \frac{1}{\sqrt{2\pi}} \exp(-\beta D^2(X_i, X_{i'})), \quad \forall i, i' = 1, \dots, n$$

où  $D(X_i, X_{i'})$  est une distance entre les objets géographiques  $X_i$  et  $X_{i'}$ .

- L'hyperparamètre  $\beta > 0$  influe sur la notion de voisinage : plus  $\beta$  est petit plus le poids est influencé par les points les plus proches.
- On peut déterminer la valeur de  $\beta$  par validation croisée (LOOCV).
- GWR peut faire des prédictions en toute position géographique du domaine étudié. Il faut toutefois faire attention aux points suivants :
  - La localisation doit rester dans la zone d'influence des données d'entraînement.
  - Les relations spatiales et les variables explicatives doivent rester similaires à celles des données d'entraînement.
- Dans [Brunsdon et al., 1996], les auteurs développent des procédures stat. testant l'hypothèse nulle qu'un modèle globale stationnaire est plus pertinent que le modèle GWR supposant la non stationnarité.

## \*\*Code\*\* : GWR, AirBnB (suite)

- Voici les résultats obtenus pour le code précédent mettant en oeuvre la méthode GWR sur les données AirBnB.

```

1 # Extract and print interesting statistics
2 print("== Global Statistics ==")
3 print(f"R-squared: {gwr_results.R2:.4f}")
4 print(f"Adjusted R-squared: {gwr_results.adj_R2:.4f}")
5 print(f"AICc (Corrected Akaike Information Criterion): {gwr_results.aicc:.4f}")

```

```

1 === Global Statistics ===
2 R-squared: 0.7106
3 Adjusted R-squared: 0.7072
4 AICc (Corrected Akaike Information Criterion): 7289.8616

```

## Rappel du Sommaire

- 1 Introduction et motivations générales
- 2 Concepts de base et outils Python associés
- 3 Analyse de la dépendance spatiale
- 4 Méthodes de régression spatiale
- 5 Méthodes d'ensemble en ML et extensions spatiales
- 6 Eléments de géostatistique
- 7 Méthodes de DL et extensions spatiales

## Introduction : Econométrie versus Machine Learning

- Dans ce contexte, l'analyse des résidus sur les données IS permet de déterminer la pertinence du modèle de régression spatiale : si les résidus sont spatialement corrélées (Indice de Moran grand par ex.) alors le modèle ne capte pas correctement les dépendances spatiales.
- Dans la suite, nous nous plaçons dans le cadre du **Machine Learning** où le but premier est la **généralisation** c.à.d. la capacité d'un modèle appris sur des données IS à **prédirer avec grande précision la var. dépendante de données OOS**.
- Nous cherchons à **faire le moins d'hypothèses possibles** par soucis de généralité et de flexibilité dans la modélisation du phénomène aléatoire spatial. On suppose que :
  - ▶ les **dépendances** entre les variables peuvent être **non linéaires**,
  - ▶ les **dépendances spatiales** peuvent varier selon les positions géographiques et sont donc **non stationnaires**,
  - ▶ l'interprétabilité du modèle et le test d'hypothèses ne sont pas des objectifs premiers.

## Introduction : Récap. sur les modèles économétriques

- Précédemment nous avons parcouru des méthodes de régression spatiale dont le but premier est de modéliser la dépendance spatiale en vue de tester des hypothèses. La finalité n'est pas d'utiliser les modèles estimés sur des données *In Sample* (IS) pour prédire avec précision sur des données *Out Of Sample* (OOS).
- Les hypothèses spatiales sous-jacentes aux différents modèles de régression sont les suivantes :
  - ▶ SFE : intègre des biais différents pour chaque unité spatiale.
  - ▶ SR : intègre des modèles linéaires différents pour chaque unité spatiale.
  - ▶ SLX : intègre les *spillover effects* des covariables, provenant des voisins.
  - ▶ SE : intègre l'autocorrélation spatiale des résidus,
  - ▶ SLY : intègre l'influence des valeurs de la var. dépendante des voisins.
- Les tests statistiques sur les coefficients des différentes variables du modèle ajusté permettent (par ex.) de valider ou pas une hypothèse.

## Introduction : Méthodes d'évaluation des modèles

- Evaluation des modèles de régression spatiale en économétrie :
  - ▶ Mesure de l'ajustement du modèle aux données IS :  $R^2$ ,  $R^2$  ajusté, ...
  - ▶ Analyse des résidus : espérance, variance, ASG, ASL.
  - ▶ Tests statistiques : significativité des covariables, du modèle, ...
- Evaluation des **modèles de régression en ML** :
  - ▶ Mesure de l'**erreur en généralisation** : validation croisée, *bootstrap*.
  - ▶ Critères d'évaluation : MSE, RMSE, MAE, ...
- Protocole expérimental utilisant la **validation croisée** en  $k$  folds :
  - ▶ Découpage aléatoire des données annotées  $\mathbb{E}$  en  $k$  sous-ens.  $\mathbb{E}^1, \dots, \mathbb{E}^k$ .
  - ▶ Pour chaque fold  $l = 1, \dots, k$  :
    - Apprentissage du modèle sur  $\mathbb{E} \setminus \mathbb{E}^l$  (données IS).
    - Prédictions du modèle estimé sur  $\mathbb{E}^l$  (données OOS).
    - Mesure du critère d'évaluation sur  $\mathbb{E}^l$ .
  - ▶ Moyenne des  $k$  mesures d'évaluation (**erreur en généralisation**).
  - ▶ Si le modèle possède des hyperparamètres ou si on compare plusieurs modèles alors un ens. de validation pourra être utilisé par ailleurs.

## Introduction : Apprentissage supervisé et notations

- La tâche en ML correspondant à notre problématique est celle de l'**apprentissage supervisée** : on veut prédire une var. cible  $Y$  à partir de plusieurs var. explicatives (ou covariables)  $X^1, \dots, X^p$ .
- On supposera que les  $X^1, \dots, X^p$  sont des var. réelles.
- Selon le domaine  $\mathbb{Y}$  de  $Y$  on a deux types de sous-problèmes :
  - Si  $Y$  est continu ( $\mathbb{Y} \subset \mathbb{R}$ ) : **pb. de régression**.
  - Si  $Y$  est discret ( $\mathbb{Y} = \{-1, 1\}$  par ex.) : **pb. de catégorisation (ou classification)**.
- Nous avons à **disposition des exemples de réalisations** de  $(X^1, \dots, X^p, Y)$  c.à.d. un ens. de  $n$  observations  $\{(x_i, y_i)\}_{i=1, \dots, n}$ , que l'on appelle **ens. de données annotées**, dénoté  $\mathbb{E}$ .
  - Le vect. des covariables observées pour un individu  $X_i$  est noté :
  - $x_i = (x_{i1}, \dots, x_{ip}) \in \mathbb{X} \subset \mathbb{R}^p$ .
  - La variable cible observée pour un individu  $X_i$  est notée :  $y_i \in \mathbb{Y}$ .
- A partir de  $\mathbb{E}$ , on cherche à **inférer une fonction**  $f : \mathbb{X} \rightarrow \mathbb{Y}$ .
- Un objet quelconque  $X$  est représenté par un vecteur  $x \in \mathbb{X}$  et sa variable cible par  $y$ . On cherche donc à avoir :  $\hat{f}(x) \approx y, \forall x \in \mathbb{X}$ .

## Le théorème du jury de Condorcet

- Un jury doit décider collectivement sur une question dont les réponses possibles sont 0 ou 1.
- Supposons que la bonne réponse soit 1, qu'un votant quelconque a une probabilité  $p$  de donner la bonne réponse et que chaque votant est indépendant des autres.
- Le mode de scrutin est le **vote majoritaire**.
- Quel serait le nb. de votants  $t$  qu'il faudrait dans le jury pour avoir une grande probabilité  $P$  que la majorité donne la bonne réponse 1 ?
- Tout dépend de  $p$  :
  - Si  $p > 1/2$  alors ajouter des votants dans le jury permet d'augmenter la probabilité  $P$  que la décision majoritaire soit 1. De plus, si  $p > 1/2$  alors  $P \rightarrow 1$  lorsque  $t \rightarrow \infty$ .
  - Si  $p \leq 1/2$  alors ajouter des votants dans le jury fait décroître  $P$  et dans ce cas le jury optimal est composé d'un seul individu ( $t = 1$ ).

## Introduction : Arbres de décision et méthodes d'ensemble

- Nous nous intéresserons qu'à quelques méthodes de ML.
- En premier lieu, nous présentons les **arbres de décision** comme approche *standalone* mais également comme prédicteur de base pour des méthodes d'ensemble.
- Les **méthodes d'ensemble** sont des techniques reposant sur le principe du "décider en comité" où il est question de **combiner plusieurs prédicteurs**. On montre en effet, que si la variance d'un prédicteur de base est forte alors la moyenne de plusieurs prédicteurs donne des résultats meilleurs qu'un seul prédicteur (cf plus loin). De plus, on évoque ci-après le théorème du jury de Condorcet qui est un résultat qui motive également le principe du "décider en comité".
- Parmi les méthodes d'ensemble nous aborderons :
  - le **bagging** à base d'arbres,
  - les **forêts aléatoires (random forest)**,
  - le **gradient boosting** à base d'arbres.

## Jury de Condorcet et méthodes d'ensemble

- Ce résultat peut s'appliquer aux pbs d'apprentissage automatique et donne l'intuition de base de nombreuses méthodes d'ensemble :
  - Supposons que pour un pb. de catégorisation binaire, nous disposons de plusieurs classifieurs indépendants les uns des autres et que chacun d'entre eux possède un taux d'erreur inférieur à 50% (*weak classifier*).
  - Alors, un vote majoritaire de ces classifieurs donnerait (selon le théorème du jury de Condorcet) un taux d'erreur collectif plus petit que les taux d'erreurs individuels.
- Dans le cas d'un problème de régression, nous pouvons intuitivement transposer ce raisonnement en prenant une **moyenne** (ou une tendance centrale) de plusieurs régresseurs.
- Remarquons cependant que ce résultat fait l'hypothèse que les votants sont mutuellement indépendants.

## Rappel du Sommaire

### 5 Méthodes d'ensemble en ML et extensions spatiales

- Arbres de décision
  - Cas du problème de catégorisation
  - Cas du problème de régression
  - Compléments sur les arbres de décision
- Bagging
- Forêts aléatoires (RF)
- Gradient boosting (GB)
- Moran Eigenvector Maps (MEM)
- Geographical Random Forest (GRF)

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

265 / 534

## Principe des arbres décisionnels (suite)

- $f^m$ , la fonction de discrimination du nœud  $m$  est une **fonction simple**. Mais, l'ensemble des fonctions  $f^m$  de chaque nœud de l'arbre tout entier aboutit à une **fonction de décision complexe**.
- Les méthodes de cette famille se distinguent selon :
  - ▶ Le type de fonction  $f^m$  choisi pour discriminer un ensemble de points.
  - ▶ Le type de critère évaluant la qualité d'une fonction de discrimination.
- **A chaque feuille de l'arbre est associé un élément de  $\mathbb{Y}$**  :
  - ▶ Pour un problème de catégorisation il s'agit donc d'une classe.
  - ▶ Pour un problème de régression il s'agit donc d'un réel.
- Chaque feuille correspond à une région de  $\mathbb{X}$  et tout  $x$  appartenant à une même feuille a le même élément de  $\mathbb{Y}$  associé à la feuille.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

267 / 534

## Principe des arbres décisionnels

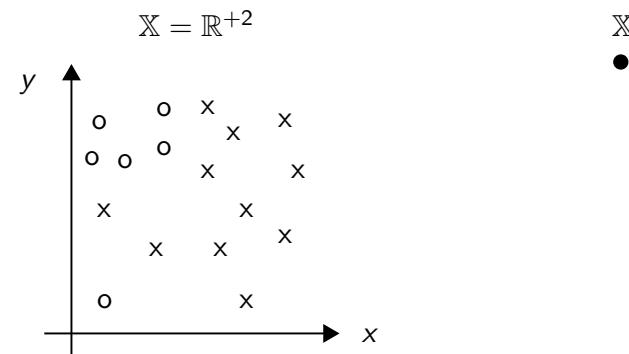
- Un arbre décisionnel est une **structure hiérarchique** qui peut être représentée par un graphe dont les nœuds représentent des sous-espaces de  $\mathbb{X}$ .
- La racine contient tout  $\mathbb{X}$  tandis que les feuilles des régions unitaires.
- Entre la racine et les feuilles, les nœuds intermédiaires représentent des **régions emboîtées** :  $\mathbb{X} = \mathbb{X}^1 \oplus \dots \oplus \mathbb{X}^m \oplus \dots \oplus \mathbb{X}^{p'}$  avec  $p' \leq p$  et chaque  $\mathbb{X}^m$  peut être à nouveau décomposé en sous-régions...
- **A chaque nœud  $m$**  est associé une région  $\mathbb{X}^m \subset \mathbb{X}$  et **une fonction de décision** dénotée  $f^m$  qui prend en entrée un élément  $x \in \mathbb{X}^m$  et qui donne en sortie un sous-espace  $\mathbb{X}^{m'} \subset \mathbb{X}^m$ .
- Les arbres décisionnels sont considérés comme des **méthodes non paramétriques** dans la mesure où :
  - ▶ Aucune hypothèse sur la distribution des données n'est faite.
  - ▶ La structure de l'arbre n'est pas donnée à l'avance : on ajoute nœuds, arcs et feuilles (les paramètres du modèle), selon les données.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

266 / 534

## Prenons un exemple !



J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

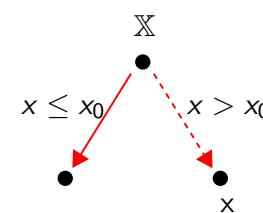
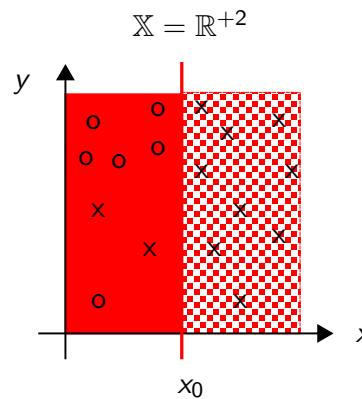
268 / 534

J. Ah-Pine

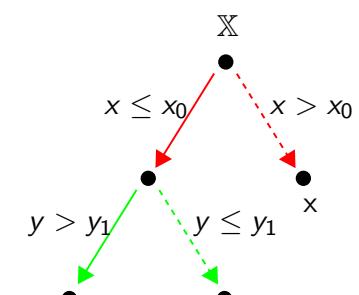
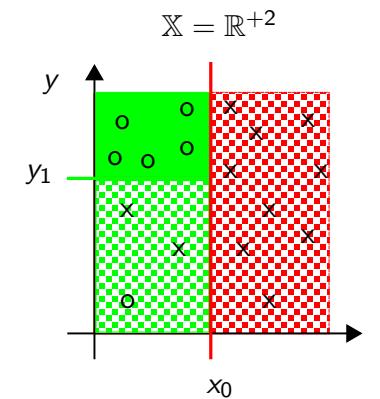
Masterclass IHEDD-CERDI-GDN GeoAI 24

268 / 534

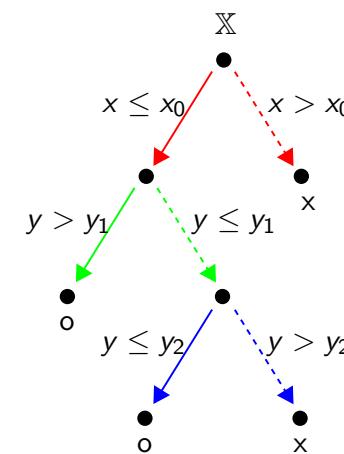
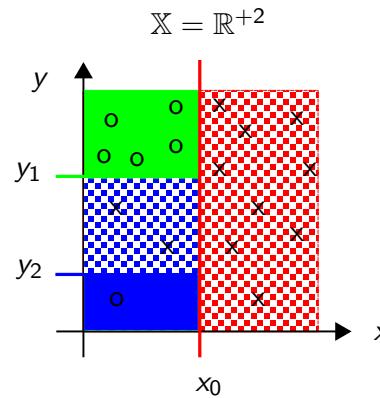
## Prenons un exemple ! (suite)



## Prenons un exemple ! (suite)



## Prenons un exemple ! (suite)



## Arbre de décision pour la catégorisation

- On considère  $\mathbb{Y} = \{C_1, \dots, C_q\}$  comme un ensemble discret. On parle alors d'**arbre de classification**.
- $\mathbb{X}$  peut être mixte c.à.d. un mélange de var. continues et discrètes.
- Nous traiterons essentiellement des méthodes **univariées** c.à.d. à chaque nœud  $m$  on utilise une seule variable  $X^j$  pour définir  $f^m$ .
- Si  $X^j$  est **discrète** avec  $q_j$  **catégories**  $\{X^{j,1}, \dots, X^{j,q_j}\}$  alors :

$$\forall \mathbf{x} \in \mathbb{X}, f^m(\mathbf{x}) \in \{X^{j,1}, \dots, X^{j,q_j}\}$$

Il s'agit dans ce cas d'une séparation ou division en  $q_j$  régions.

- Si  $X^j$  est **continue** alors :

$$\forall \mathbf{x} \in \mathbb{X}, f^m(\mathbf{x}) \in \{X^{j,l}, X^{j,r}\}$$

où  $X^{j,l} = \{\mathbf{x} \in \mathbb{X} : x_j \leq \delta_j\}$  et  $X^{j,r} = \{\mathbf{x} \in \mathbb{X} : x_j > \delta_j\}$  et  $\delta_j \in \mathbb{R}$  est une valeur permettant de faire une séparation adéquate. Il s'agit dans ce cas d'une **division en 2 régions** (séparation binaire de l'espace).

## Arbre de décision pour la catégorisation (suite)

- L'apprentissage (*tree induction*) consiste à construire un AD (Arbre de Décision) étant donné  $\mathbb{X}$ .
- Il existe de très nombreux arbres permettant de découper l'espace  $\mathbb{X}$  de sorte à n'avoir aucune erreur.
- Si on applique le principe du rasoir d'Occam<sup>3</sup>, on cherche l'**AD d'erreur nulle qui est le plus petit** en terme de nombre de noeuds.
- Ce pb est **NP-complet**, on utilise donc des **heuristiques** :
  - On part de  $\mathbb{X}$  tout entier : ce noeud représente la racine.
  - Pour chaque noeud  $m$  on détermine la fonction  $f^m$  permettant d'optimiser localement un critère.
  - La fonction  $f^m$  permet alors de séparer l'espace en plusieurs régions (arcs) et chacune d'entre elles représente un nouveau noeud.
  - On ajoute noeuds et arcs jusqu'à satisfaire un critère d'arrêt.

3. Si deux modèles sont de performances comparables alors on choisira celui qui est le moins complexe.

## Arbre de décision pour la catégorisation (suite)

- Pour mesurer la pureté d'une séparation, nous utiliserons la méthode classique proposée par [Quinlan, 1986] qui est basée sur l'**entropie** :

$$\begin{aligned} \text{Ent}(p^m) &= - \sum_{l=1}^q P(C_l|\mathbf{x}, m) \log_2(P(C_l|\mathbf{x}, m)) \\ &= - \sum_{l=1}^q p_l^m \log_2(p_l^m) \end{aligned}$$

où par convention  $0 \log_2(0) = 0$ .

- L'entropie correspond intuitivement à la quantité d'information contenue ou délivrée par une source d'information.
- Dans le cas binaire, si  $p_1^m = 1$  et  $p_2^m = 0$  : il faut 0 bit pour transmettre l'information.
- Si  $p_1^m = p_2^m = 1/2$  alors la quantité d'information est maximale : il faut 1 bit (1 pour la classe  $C_1$  et 0 pour la classe  $C_2$ ) pour transmettre l'information.

## Arbre de décision pour la catégorisation (suite)

- La mesure de qualité d'une division en plusieurs branches est relative au concept d'**impureté**.
- Une séparation  $f^m$  est pure si chacune des branches conduit à des noeuds dont les éléments sont tous de la même catégorie.
- Dénotons par  $\mathbf{N}^m$  le nombre d'éléments du noeud  $m$ . Pour la racine on a donc  $\mathbf{N}^m = n$ .
- Parmi les éléments du noeud  $m$ , dénotons par  $\mathbf{N}_l^m$  le nombre de ceux appartenant à la classe  $C_l$ . On a alors :

$$P(C_l|\mathbf{x}, m) = \frac{\mathbf{N}_l^m}{\mathbf{N}^m}$$

- Le noeud  $m$  est pur** si  $\exists C_l \in \mathbb{Y} : P(C_l|\mathbf{x}, m) = 1$ . Ainsi,  $f^m$  est pure si pour tous les noeuds qu'elle engendre, ceux-ci sont purs.
- Pour alléger les formules nous utiliserons la notation suivante :

$$P(C_l|\mathbf{x}, m) = p_l^m$$

## Arbre de décision pour la catégorisation (suite)

- Si un noeud  $m$  n'est pas pur alors l'objectif est de séparer le sous-ensemble de ce noeud de sorte à réduire les impuretés.
- Comme on cherche l'AD le plus petit, intuitivement, **on choisit localement la séparation qui réduit le plus l'impureté**.
- Supposons qu'au noeud  $m$ , il y a  $\mathbf{N}_k^m$  objets qui suivent la branche  $k$ . Il s'agit des objets  $\mathbf{x}$  tel que  $f^m(\mathbf{x}) = X^{j,k}$  où  $X^j$  est la variable ayant servi à faire la séparation.
- Supposons que le nb. de branches est  $q_j$  ( $q_j = 2$  si  $X^j$  est quantitative) alors nous avons  $\sum_{k=1}^{q_j} \mathbf{N}_k^m = \mathbf{N}^m$ .
- Considérons la variable cible  $Y$  et soit  $\mathbf{N}_{kl}^m$  le nombre d'objets de  $C_l$  ayant suivis la branche donnée par  $X^{j,k}$ . La probabilité d'observer la classe  $C_l$  dans le noeud issu de la branche  $X^{j,k}$  vaut :

$$P(C_l|\mathbf{x}, m, X^{j,k}) = \frac{\mathbf{N}_{kl}^m}{\mathbf{N}_k^m}$$

## Arbre de décision pour la catégorisation (suite)

- Pour alléger les formules notons :

$$P(C_l | \mathbf{x}, m, X^{j,k}) = p_{kl}^m$$

- L'**impureté totale** issue de la division engendrée par  $X^j$  est :

$$\text{Ent}(p^m, X^j) = - \sum_{k=1}^{q_j} \frac{\mathbf{N}_k^m}{\mathbf{N}^m} \sum_{l=1}^q p_{kl}^m \log_2(p_{kl}^m)$$

- A chaque nœud on détermine  $X^j$  qui minimise  $\text{Ent}(p^m, X^j)$ .**
- Si  $X^j$  est **qualitative**, les séparations sont données par les différentes catégories  $\{X^{j,1}, \dots, X^{j,q_j}\}$ .
- Si  $X^j$  est **quantitative**, il faut déterminer  $\delta_j$  donnant la meilleure division  $\{X^{j,l}, X^{j,r}\}$ . Il y a  $\mathbf{N}^m - 1$  possibilités mais le meilleur point de division est tjs entre deux objets adjacents de classes distinctes.

## Arbre de décision pour la catégorisation (suite)

- Un autre problème survient si le critère d'arrêt est l'obtention de **feuilles toutes pures** (c.à.d. on s'arrête une fois que tous les nœuds terminaux obtenus n'ont qu'une seule classe représentée). Dans ce cas, on risque (i) d'avoir un **AD trop grand** et (ii) de faire du **sur-apprentissage**<sup>4</sup>.
- Pour remédier à ce problème, on se donne un **seuil**  $\theta \in [0, 1]$  en dessous duquel on estime que la pureté obtenue est suffisante.
- Ainsi la condition d'arrêt de l'apprentissage est que pour tout nœud final  $m$  :  $\text{Ent}(p^m) \leq \theta$ .
- Chaque feuille  $m$  est alors associée à la classe la plus représentative c.à.d. la classe  $C_l$  tel que  $\forall l' \neq l : p_{l'}^m \geq p_l^m$ .
- Dans certaines applications, on représente chaque feuille  $m$  par la distribution de probabilités  $(p_1^m, \dots, p_q^m)$ . Par ex. si on souhaite calculer un **risque associé aux catégorisations** données par l'AD.

4. Le modèle est trop spécifique aux données d'entraînement et généralise mal sur des données non encore observées.

## Arbre de décision pour la catégorisation (suite)

- L'AD se construit de façon récursive** : à chaque nœud on cherche localement la variable  $X^j$  minimisant l'impureté d'une nouvelle division et ce jusqu'à ce que l'on obtienne une séparation pure.
- Il existe un **biais** à cette approche : les variables qualitatives ayant beaucoup de catégories donnent une plus faible entropie.
  - Nous pouvons alors décider de nous restreindre à des **AD binaires** c.à.d. chaque division est composée de deux branches. Mais dans le cas d'une variable qualitative à  $q_j$  catégories, il existe  $2^{q_j-1} - 1$  possibilités et dans le cas général, si  $q_j$  est grand le problème devient exponentiel.
  - En revanche, pour un problème de catégorisation binaire ( $\{C_1, C_2\}$ ), on peut ordonner les catégories de  $X^j$  dans l'ordre décroissant de  $p_{k1}^m$  et traiter ensuite cet ordre telle une variable ordonnée avec cette fois-ci uniquement  $q_j - 1$  possibilités de séparation.
  - Dans ce cas on préférera un AD binaire puisque celui-ci peut retrouver l'AD avec plusieurs branches si ce cas était le meilleur.

## Arbre de décision pour la catégorisation (suite)

- $\theta$  peut être vu comme un **param. de la complexité de l'AD** similaire au  $k$  du k-ppv dans le contexte des méthodes non paramétriques.
- Si  $\theta$  est petit, la variance est large** alors que l'AD est grand de sorte à reproduire les données d'entraînement de façon précise.
- Si  $\theta$  est grand, la variance est plus faible** et l'arbitrage biais-variance nous indique que le biais risque d'être plus grand.
- Dans la suite nous utiliserons les notations suivantes :
  - $\mathbf{X}$  est la matrice initiale des données avec  $n$  objets  $\{X_1, \dots, X_n\}$  et  $p$  attributs  $\{X^1, \dots, X^p\}$ .
  - $\mathbf{X}^m$  est la matrice des données relatives au nœud  $m$  qui comporte  $\mathbf{N}^m$  objets et les  $p$  attributs. Il s'agit d'une sous-matrice de  $\mathbf{X}$ .
  - On remarquera qu'un nœud fils comporte un sous-ensemble des objets de son nœud parent.
  - L'algorithme qui suit synthétise différentes évolutions des AD (CART [Breiman et al., 1984], ID3 [Quinlan, 1986], C4.5 [Quinlan, 1993]).
  - Rq : d'autres façons existent afin de contrôler la profondeur de l'arbre en ajoutant des contraintes et hyperparamètres sur les divisions.

## Pseudo-code du tree induction (catégorisation)

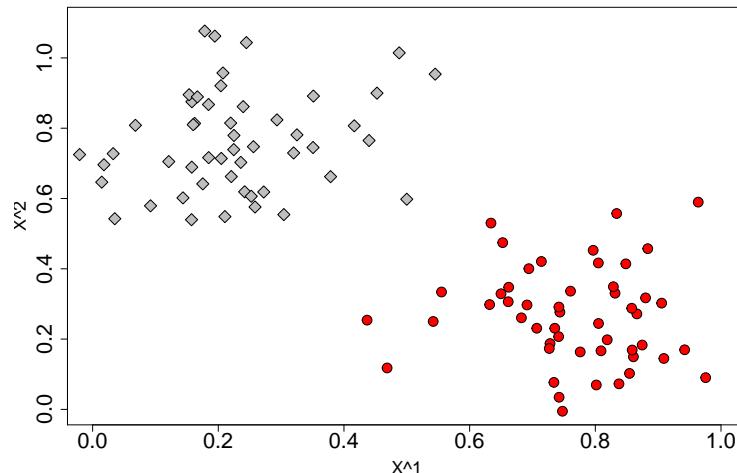
```

Fonction : ArbreGeneration
Input :  $\mathbf{X}^m, \mathbf{y}^m, \theta$ 
1   Si  $\text{Ent}(p^m) \leq \theta$  faire
2       Créer une feuille et l'étiqueter avec la classe majoritaire
3       Retour
4   Fin Si
5    $j^* \leftarrow \text{DivisionAttribut}(\mathbf{X}^m)$ 
6   Initialiser un sous-arbre  $S$ 
7   Pour toute Branche  $m'$  dans  $\{X^{j^*,1}, \dots, X^{j^*,q_j}\}$  faire
8       Déterminer  $\mathbf{X}^{m'}, \mathbf{y}^{m'}$ 
9        $S' \leftarrow \text{ArbreGeneration}(\mathbf{X}^{m'}, \mathbf{y}^{m'}, \theta)$ 
10      Ajouter  $S'$  à une branche de  $S$ 
11  Fin Pour

```

## Prenons un exemple !

- Reprenons l'exemple précédent :



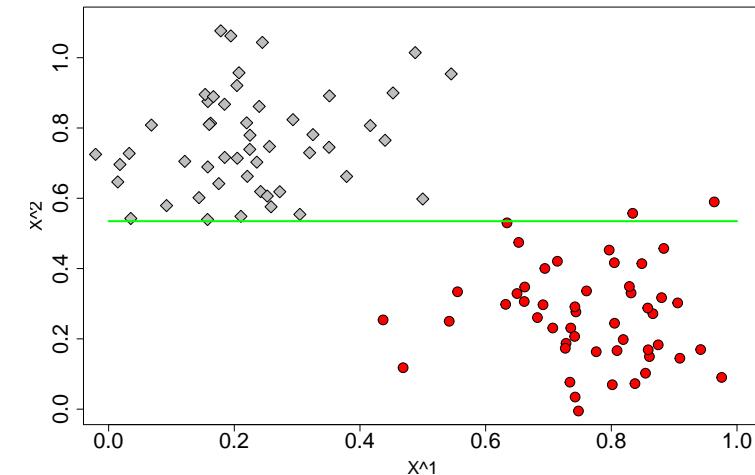
## Pseudo-code du tree induction (catégorisation) (suite)

```

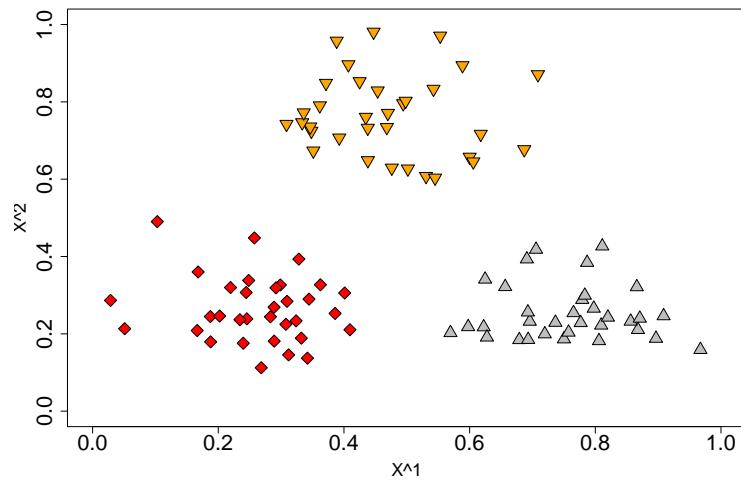
Fonction : DivisionAttribut
Input :  $\mathbf{X}^m, \mathbf{y}^m$ 
1    $MinE \leftarrow +\infty$ 
2   Pour tout Attribut  $X^j$  de  $\{X^1, \dots, X^P\}$  faire
3       Si  $X^j$  est qualitative avec  $q_j$  catégories faire
4            $E \leftarrow \text{Ent}(p^m, X^j)$ 
5           Si  $E < MinE$  faire  $MinE \leftarrow E, j^* \leftarrow j$  Fin Si
6       Sinon ( $X^j$  est quantitative)
7           Pour toute Séparation en  $\{X^{j,l}, X^{j,r}\}$  possibles faire
8                $E \leftarrow \text{Ent}(p^m, \{X^{j,l}, X^{j,r}\})$ 
9               Si  $E < MinE$  faire  $MinE \leftarrow E, j^* \leftarrow j$  Fin Si
10          Fin Pour
11      Fin Si
12  Fin Pour
13  Output :  $j^*$ 

```

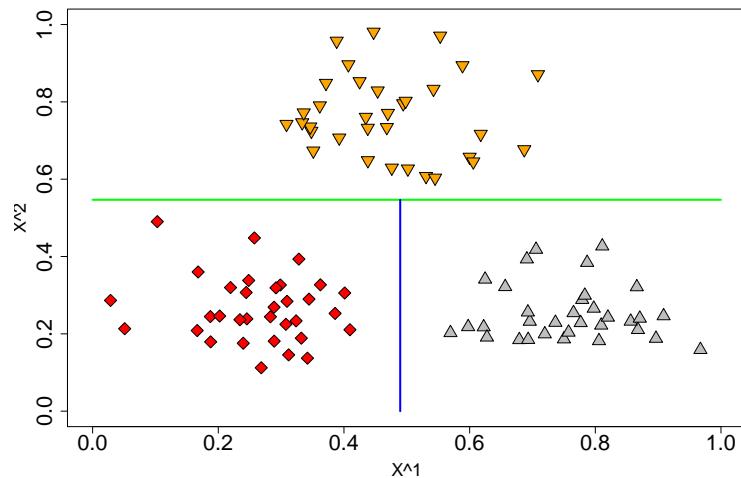
## Prenons un exemple ! (suite)



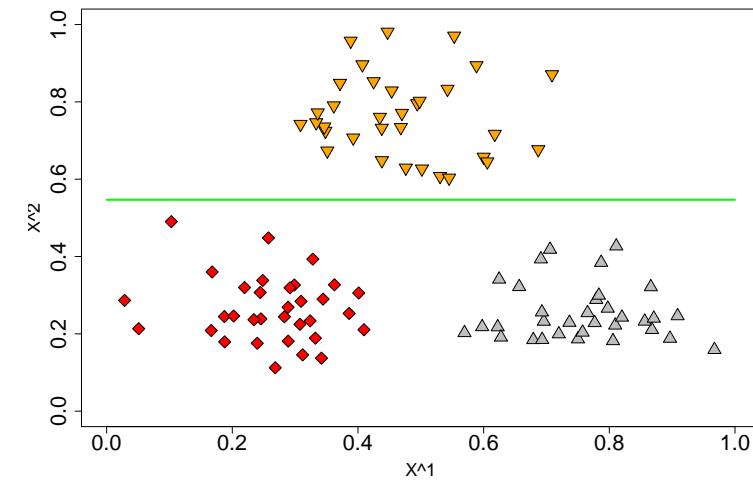
## Prenons un exemple ! (suite)



## Prenons un autre exemple ! (suite)



## Prenons un autre exemple !



## \*\*Code\*\* Catégorisation, *Two moons*

- Nous allons illustrer le code Python sur une tâche de catégorisation binaire à partir de données simulées en 2D : les deux lunes (données non linéairement séparables).

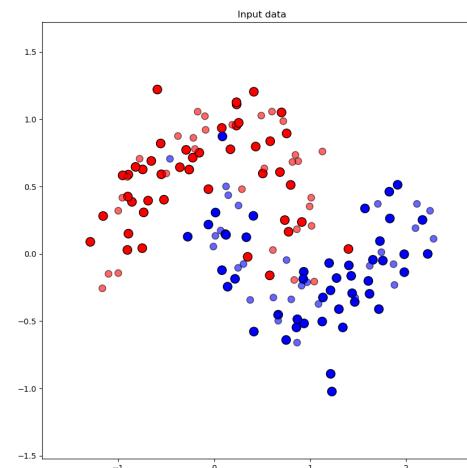
```

1  from sklearn.datasets import make_moons
2  from matplotlib.colors import ListedColormap
3  # Generate two moons data
4  X, y = make_moons(n_samples=150, noise=0.2, random_state=0)
5  # Instanciate an empty figure
6  figure = plt.figure(figsize=(10, 10))
7  # Split the data into train and test
8  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state
9  =42)
10 x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
11 y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
12 # Set the colors for two classes and for train and test
13 cm_bright = ListedColormap([ "#FF0000", "#0000FF"] )# for test data
14 # Plot the dataset
15 ax = plt.subplot(1, 1, 1)
16 ax.set_title("Input data")
17 # Plot the training points
18 ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright, edgecolors="k", s
19 =125)
20 # Plot the testing points
21 ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, alpha=0.6, edgecolors
22 = "k", s=75)
23 ax.set_xlim(x_min, x_max)
24 ax.set_ylim(y_min, y_max)

```

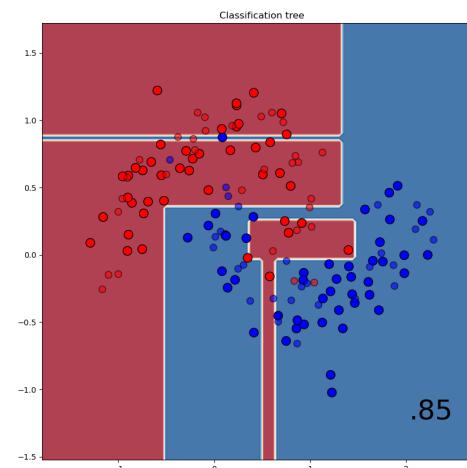
## \*\*Code\*\* Catégorisation, *Two moons* (suite)

- NUAGE DE POINTS représentant les deux classes en rouge et bleu. Les données d'entraînement sont les points plus foncés et plus grands.



## \*\*Code\*\* Arbre de classif., *Two moons* (suite)

- LIGNES DE NIVEAU DES SCORES de l'arbre de décision appris.



## \*\*Code\*\* Arbre de classification, *Two moons*

- Ex. de code Python pour l'arbre de décision pour la tâche de catégorisation avec tuning de plus. hyperparamètres par *grid search*.

```

1  from sklearn.tree import DecisionTreeClassifier # Decision Tree
2  # Define the model and fit on training data
3  dt = DecisionTreeClassifier()
4  dt_params = {'max_depth': [None, 1, 2, 3, 4, 5], 'min_samples_split': [2, 5, 10]} # Parameters to test
5  # Use GridSearchCV to find the best parameters using 5-fold cross-validation
6  dt_cv = GridSearchCV(dt, dt_params, cv=5, scoring='accuracy')
7  dt_cv.fit(X_train, y_train)
8  score = dt_cv.score(X_test,y_test)
9  # Instantiate an empty figure and plot the decision contours
10 figure = plt.figure(figsize=(10, 10))
11 ax = plt.subplot(1, 1, 1)
12 DecisionBoundaryDisplay.from_estimator(dt_cv, X, cmap=cm, alpha=0.8, ax=ax, eps=0.5)
13 # Plot the training points then the test points
14 ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright, edgecolors="k", s=125)
15 ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, edgecolors="k", alpha=0.6, s=75)
16 ax.set_xlim(x_min, x_max)
17 ax.set_ylim(y_min, y_max)
18 ax.set_title("Classification tree")
19 ax.text(x_max - 0.3, y_min + 0.3, ("%.2f" % score).lstrip("0"), size=35, horizontalalignment="right")

```

## Arbre de décision pour la régression

- On considère maintenant  $\mathbb{Y} = \mathbb{R}$ . On parle alors d'**arbre de régression**.
- $\mathbb{X}$  peut toujours être mixe (mélange de var. continues et discrètes).
- Ce qui change par rapport aux arbres de classification c'est la fonction d'"impureté" (c.à.d. la fonction objectif).
- Soit  $\mathbf{X}^m$  la sous-matrice de données de taille  $(\mathbf{N}^m \times p)$  relative au nœud  $m$ . Par abus de notation on dira qu'il s'agit de l'ensemble des objets qui ont suivi le chemin pour arriver jusqu'à  $m$ . Notons alors la fonction indicatrice suivante :

$$\text{Ind}(\mathbf{x} \in \mathbf{X}^m) = \begin{cases} 1 & \text{si } \mathbf{x} \in \mathbf{X}^m \\ 0 & \text{sinon} \end{cases}$$

## Arbre de décision pour la régression (suite)

- Pour mesurer la pureté d'un nœud  $m$  on utilise le critère suivant :

$$\text{Err}(m) = \frac{1}{\mathbf{N}^m} \sum_{i=1}^n (y_i - \mu^m)^2 \text{Ind}(\mathbf{x}_i \in X^m)$$

où  $\mathbf{N}^m = \sum_{i=1}^n \text{Ind}(\mathbf{x}_i \in X^m)$  est le nombre d'objets de  $X^m$ .

- $\mu^m$  est une tendance centrale relative au nœud  $m$  : c'est une mesure qui résume les valeurs des objets appartenants à  $m$ . On utilise la moyenne (la médiane si les données sont très bruitées) :

$$\mu^m = \frac{\sum_{i=1}^n y_i \text{Ind}(\mathbf{x}_i \in X^m)}{\sum_{i=1}^n \text{Ind}(\mathbf{x}_i \in X^m)} = \frac{1}{\mathbf{N}^m} \sum_{i=1}^n y_i \text{Ind}(\mathbf{x}_i \in X^m)$$

- Dans ce cas  $\text{Err}(m)$  est une variance locale au nœud  $m$ .
- Le critère qui arrête la progression de l'Arbre de décision est  $\text{Err}(m) \leq \theta$ .  $\theta$  est donc un seuil en-dessous duquel on estime que la variance de la région relative au nœud  $m$  est suffisamment basse.

## Arbre de décision pour la régression (suite)

- Pour choisir la variable séparatrice  $X^j$ , on prend celle qui minimise :

$$\text{Err}(m, X^j) = \frac{1}{\mathbf{N}^m} \sum_{k=1}^{q_j} \sum_{i=1}^n (y_i - \mu_k^m)^2 \text{Ind}(\mathbf{x}_i \in X^m \cap X^{j,k})$$

- Le **pseudo-code** donné précédemment dans le cas de la catégorisation s'adapte entièrement au problème de régression en remplaçant :
  - ▶  $\text{Ent}(p^m)$  par  $\text{Err}(m)$  dans le pseudo-code ArbreGeneration.
  - ▶  $\text{Ent}(p^m, X^j)$  par  $\text{Err}(m, X^j)$  dans le pseudo-code DivisionAttribut.
- Le paramètre  $\theta$  est comme précédemment un param. de la complexité de l'AD. Il existe par ailleurs, d'autres types de contraintes simples permettant de contrôler la profondeur de l'arbre. Par ex., nous pouvons inscrire un hyperparam. de la taille minimale d'un nœud pour décider s'il est suffisamment fourni pour être divisé. Ceci est motivé par des considérations stat. relatives à la taille d'un échantillon.

## Arbre de décision pour la régression (suite)

- Il nous faut également spécifier un **critère pour décider de  $f^m$**  c.à.d. la division à utiliser au nœud  $m$  si celui-ci est de variance (ou "impureté") encore trop forte.
- Prenons une variable  $X^j$  qui induit une séparation en  $q_j$  branches  $\{X^{j,1}, \dots, X^{j,q_j}\}$  et introduisons  $\forall k = 1, \dots, q_j$  :

$$\text{Ind}(\mathbf{x} \in \mathbf{X}^m \cap X^{j,k}) = \begin{cases} 1 & \text{si } \mathbf{x} \in \mathbf{X}^m \wedge \mathbf{x} \in X^{j,k} \\ 0 & \text{sinon} \end{cases}$$

- Soit  $\mu_k^m$  la tendance centrale des objets de la branche  $X^{j,k}$  de  $m$  :

$$\mu_k^m = \frac{\sum_{i=1}^n y_i \text{Ind}(\mathbf{x}_i \in \mathbf{X}^m \cap X^{j,k})}{\sum_{i=1}^n \text{Ind}(\mathbf{x}_i \in \mathbf{X}^m \cap X^{j,k})} = \frac{1}{\mathbf{N}_k^m} \sum_{i=1}^n y_i \text{Ind}(\mathbf{x}_i \in \mathbf{X}^m \cap X^{j,k})$$

## Elagage de l'arbre de décision

- D'autres critères permettent d'améliorer les performances de l'AD : on parle d'**élagage** de l'AD. Ce process peut s'effectuer au cours de la construction de l'AD (**pré-élagage**) ou après la construction de l'AD (**post-élagage**).
- Exemple de **pré-élagage** :
  - ▶ Si le pourcentage de données au nœud  $m$  est en-dessous d'un seuil  $\alpha$  on ne sépare pas le nœud : prendre une décision de division sur trop peu d'éléments augmente la variance du modèle et donc potentiellement des erreurs en généralisation.
- Principe du **post-élagage** :
  - ▶ On construit l'AD jusqu'à avoir des feuilles pures ( $\theta = 0$ ) sur  $\mathbb{E}$ .
  - ▶ On tente de détecter des sous-arbres qui causent du sur-apprentissage et on les enlève de l'AD.
  - ▶ Pour chq. sous-arbre enlevé, on le remplace par une feuille étiquetée avec la classe majoritaire (catégo.) ou la tendance centrale (rég.).

## Extraction de règles à partir d'un arbre de décision

- Un AD fait de la **sélection de variable** : dans un AD il se peut que certaines variables  $X^j$  ne soient pas du tout utilisées.
- Les variables de division proches du nœud racine sont globalement plus importantes.
- Contrairement aux SVM et les réseaux de neurones par exemple, les AD sont **facilement interprétables**.
- Tout **chemin** du nœud racine à une feuille est une **conjonction de plusieurs tests**.
- Pour l'ex. précédent : **Si** ( $x_{i2} < 0.54$ ) **et** ( $x_{i1} < 0.48$ ) **alors** ( $X_i \in C_1$ ).
- On a donc pour un AD un **ensemble de "IF-THEN" règles** qui permet de faire de l'**extraction de connaissances**.
- Les AD sont en revanche des modèles à **forte variance** : de petits changements dans  $\mathbb{E}$  peuvent causer de nombreux changements dans l'AD.

## \*\*Code\*\* Arbre de régression, AirBnB

- Ex. de code Python pour l'arbre de décision pour la tâche de régression avec tuning de plus. hyperparamètres par *grid search*.

```

1 from sklearn.tree import DecisionTreeRegressor # Decision Tree
2
3 # Decision Tree Regressor with hyperparameter tuning (max_depth and min_samples_split)
4 dt = DecisionTreeRegressor()
5 dt_params = {'min_samples_split': [2, 5, 10], 'max_depth': [None, 3, 6, 10]} # Parameters to test
6
7 # Use GridSearchCV to find the best combination of max_depth and min_samples_split
8 # using 5-fold cross-validation
9 dt_cv = GridSearchCV(dt, dt_params, cv=5, scoring='neg_mean_squared_error', n_jobs=4)
10 dt_cv.fit(X_train, y_train)
11
12 # Print the best parameters and the RMSE on the test set
13 print('Best Decision Tree params:', dt_cv.best_params_)
14 y_pred_dt = dt_cv.predict(X_test)
15 print('Decision Tree RMSE:', np.sqrt(mean_squared_error(y_test, y_pred_dt)))

```

```

1 Best Decision Tree params: {'max_depth': 6, 'min_samples_split': 10}
2 Decision Tree RMSE: 0.4489844947078088

```

## \*\*Code\*\* Var. spatiales, ens. *training* et *test*, AirBnB

- Ex. de code Python pour ajouter la longitude et la latitude parmi les variables explicatives dans AirBnB et pour séparer les données annotées en ens. d'entraînement et ens. de test.

```

1 db = gpd.read_file("regression_db.geojson")
2 variable_names = [
3     "accommodates", # Number of people it accommodates
4     "bathrooms", # Number of bathrooms
5     "bedrooms", # Number of bedrooms
6     "beds", # Number of beds
7     "# Below are binary variables, 1 True, 0 False
8     "rt_Private_room", # Room type: private room
9     "rt_Shared_room", # Room type: shared room
10    "pg_Condominium", # Property group: condo
11    "pg_House", # Property group: house
12    "pg_Other", # Property group: other
13    "pg_Townhouse" # Property group: townhouse
14 ]
15 X = db[variable_names].values # Select exogeneous var in an np.array
16 y = db["log_price"].values # Put target var in a np.array
17 lon = np.array(db["geometry"].x) # Extract longitude from the geopandas df
18 lat = np.array(db["geometry"].y) # Extract latitude from the geopandas df
19 X = np.column_stack((X, lon, lat)) # Add the two spatial features to data matrix
20 variable_names.append("longitude")
21 variable_names.append("latitude")
22 # Split the data into training and test sets
23 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

## Rappel du Sommaire

### 5 Méthodes d'ensemble en ML et extensions spatiales

- Arbres de décision
  - Cas du problème de catégorisation
  - Cas du problème de régression
  - Compléments sur les arbres de décision
- Bagging
  - Forêts aléatoires (RF)
  - Gradient boosting (GB)
  - Moran Eigenvector Maps (MEM)
  - Geographical Random Forest (GRF)

## Introduction

- Méthode proposée par Breiman [Breiman, 1996].
- Le *bagging* consiste à :
  - 1 créer plusieurs échantillons bootstrap à partir de  $\mathbb{E}$ ,
  - 2 inférer une fonction de prédiction d'un même modèle d'apprentissage de base sur chaque échantillon bootstrap,
  - 3 agréger les prédictions données par chaque fonction :
    - par un **vote majoritaire** si c'est un problème de catégorisation,
    - par une **moyenne** si c'est un problème de régression.
- La méthode d'apprentissage de base utilisée en 2 doit être de **forte variance** : un "petit" changement de l'ensemble d'apprentissage doit générer un "grand" changement dans la fonction de prédiction estimée (sinon manque de **diversité** et le *bagging* n'apporte pas grand chose).
- Le modèle utilisé est ainsi en général les arbres décisionnels. Mais d'autres techniques reposant sur des familles hypothèses complexes tels que les réseaux de neurones, peuvent être utilisées.

## Erreur Out Of Bag

- Etant donné un échantillon en cours de constitution, si les tirages sont mutuellement indépendants, la probabilité pour qu'un objet de  $\mathbb{E}$  ne soit pas tiré après  $n$  tirages est de  $\sim 37\%$ .
- Un échantillon bootstrap  $\mathbb{E}^b$  contient uniquement  $\sim 63\%$  des objets de  $\mathbb{E}$ . Le complémentaire  $\mathbb{T}^b = \mathbb{E} \setminus \mathbb{E}^b$  contient  $\sim 37\%$  des objets de  $\mathbb{E}$ . Ils sont dits **Out Of Bag** (OOB) et n'interviennent pas dans l'estimation de la fonction de prédiction à partir de  $\mathbb{E}^b$ .
- Les objets OOB de  $\mathbb{T}^b$  sont exploités afin d'estimer l'erreur en généralisation (sans passer par une validation croisée !) et également pour estimer les hyperparamètres du modèle.
- L'estimation de l'erreur OOB consiste à :
  - ▶ Pour chaque échantillon bootstrap  $(\mathbb{E}^b, \mathbb{T}^b)$  :
    - Apprendre une fonction de prédiction à partir de  $\mathbb{E}^b$ .
    - Déterminer les prédictions des objets OOB dans  $\mathbb{T}^b$ .
  - ▶ Moyenner les erreurs évaluées sur les prédictions des objets OOB de chaque échantillon bootstrap. On parle alors d'**erreur OOB**.

## Bootstrap

- Un bref rappel sur le bootstrap : la méthode consiste à générer de nouveaux échantillons à partir de l'échantillon initial :
  - ▶ On tire aléatoirement avec remise  $n$  objets de  $\mathbb{E}$  et on obtient ainsi  $\mathbb{E}^b$ .
  - ▶ On infère de  $\mathbb{E}^b$  une fonction de prédiction.
- On répète le processus et on crée ainsi  $t$  échantillons bootstrap permettant d'inférer  $t$  fonctions de prédiction.
- Le bootstrap est une méthode d'échantillonage utilisé en statistique inférentielle pour étudier les propriétés des estimateurs définis sur une population. Dans notre cas, l'idée est de moyenner l'erreur de prédiction donnée par les  $t$  fonctions de prédiction ce qui permet d'avoir une estimation plus robuste. Mais, il faut faire attention à bien définir pour chaque fonction estimée un ensemble de test adéquat.

## Analyse théorique du bagging en régression

- Notons  $f_{\mathbb{E}}$  une fonction de prédiction inférée d'un ensemble d'entraînement  $\mathbb{E} = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$  dont les éléments sont des réalisations iid. d'une loi jointe inconnue  $p_{\mathbf{X},y}(\mathbf{x}, y)$  où  $y \in \mathbb{R}$ .
- Dans ce cas, la fonction de prédiction du *bagging* est :

$$f_{\text{bag}}(\mathbf{x}) = E_{\mathbb{E}}(f_{\mathbb{E}}(\mathbf{x}))$$

- Soit  $(\mathbf{x}, y)$  un couple quelconque donné, l'espérance de l'erreur quadratique sous  $\mathbb{E}$  de  $f_{\mathbb{E}}$  pour ce couple vaut :

$$E_{\mathbb{E}}([y - f_{\mathbb{E}}(\mathbf{x})]^2) = y^2 - 2yE_{\mathbb{E}}(f_{\mathbb{E}}(\mathbf{x})) + E_{\mathbb{E}}([f_{\mathbb{E}}(\mathbf{x})]^2)$$

- Comme pour toute va.  $\mathcal{Z}$ ,  $E(\mathcal{Z}^2) \geq (E(\mathcal{Z}))^2$ , en prenant  $\mathcal{Z} = f_{\mathbb{E}}(\mathbf{x})$ , on voit alors que  $E_{\mathbb{E}}([f_{\mathbb{E}}(\mathbf{x})]^2) \geq (E_{\mathbb{E}}(f_{\mathbb{E}}(\mathbf{x})))^2$  et donc :

$$E_{\mathbb{E}}([y - f_{\mathbb{E}}(\mathbf{x})]^2) \geq (y - E_{\mathbb{E}}(f_{\mathbb{E}}(\mathbf{x})))^2$$

## Analyse théorique du *bagging* en régression (suite)

- Supposons désormais que nous raisonnons avec un couple aléatoire  $(X, Y)$  de loi  $p_{X,Y}(x, y)$ . L'inégalité précédente conduit à la relation :

$$\text{E}_{X,Y} \left( \text{E}_{\mathbb{E}} \left( [Y - f_{\mathbb{E}}(X)]^2 \right) \right) \geq \text{E}_{X,Y} ((Y - \text{E}_{\mathbb{E}}(f_{\mathbb{E}}(X)))^2)$$

- Ceci est équivalent à :

$$\text{E}_{\mathbb{E}} \left( \text{E}_{X,Y} \left( [Y - f_{\mathbb{E}}(X)]^2 \right) \right) \geq \text{E}_{X,Y} \left( (Y - \underbrace{\text{E}_{\mathbb{E}}(f_{\mathbb{E}}(X))}_{f_{\text{bag}}(X)})^2 \right)$$

- L'espérance de la perte quadratique de  $f_{\text{bag}}$  est plus petite que l'espérance sous  $\mathbb{E}$  de l'espérance de la perte quadratique de  $f_{\mathbb{E}}$ .
- Ce résultat montre que la **moyenne de plusieurs fonctions de prédiction apprises sur différents échantillons** fait en moyenne moins d'erreur qu'une seule fonction de prédiction apprise sur un échantillon.

## Bagging et arbres de décision

- Nous venons de voir le *bagging* et ses différentes propriétés.
- En particulier, il est recommandé d'utiliser un modèle d'apprentissage de base qui soit de forte variance et de faible biais.
- C'est le cas des AD qui sont typiquement appliqués avec le bagging :
  - Si l'arbre est profond, celui-ci peut capturer les structures complexes des données et avoir un faible biais.
  - Les AD sont fortement variables (changer les données d'entraînement peut changer drastiquement un AD) et donc le principe de "moyenner" plusieurs arbres sous-jacent au *bagging* permet de réduire la variance (et donc améliorer en principe l'erreur en généralisation) tout en maintenant un faible biais.
- Les forêts aléatoires<sup>5</sup> sont une extension substantielle du bagging+AD qui a été également proposé par Breiman [Breiman, 2001].
- L'idée principale est de modifier le *bagging* de sorte à avoir des AD décorrélés. Cette approche fait écho au principe d'**indépendance exposé précédemment** mais non encore traité jusqu'à présent.

<sup>5</sup> [https://www.usu.edu/math/adele/forests/cc\\_home.htm](https://www.usu.edu/math/adele/forests/cc_home.htm)

## Analyse théorique du *bagging* en régression (suite)

- Le gain potentiel que l'on peut obtenir avec le *bagging* dépend de l'écart entre ces deux éléments :

$$\text{E}_{\mathbb{E}} \left( [f_{\mathbb{E}}(X)]^2 \right) \geq (\text{E}_{\mathbb{E}}(f_{\mathbb{E}}(X)))^2$$

- Si la variance de  $f_{\mathbb{E}}(X)$  est très forte** alors  $\text{E}_{\mathbb{E}} \left( [f_{\mathbb{E}}(X)]^2 \right) - (\text{E}_{\mathbb{E}}(f_{\mathbb{E}}(X)))^2$  est grand et **le gain est important**.
- Il est donc préférable d'utiliser une méthode de base de forte variance afin d'espérer observer une amélioration forte due au bagging.
- On remarquera que si la variance est nulle alors  $\text{E}_{\mathbb{E}} \left( [f_{\mathbb{E}}(X)]^2 \right) = (\text{E}_{\mathbb{E}}(f_{\mathbb{E}}(X)))^2$  et l'inégalité du slide précédent devient également une égalité (pas de gain).

## \*\*Code\*\* Bagged trees, Two moons

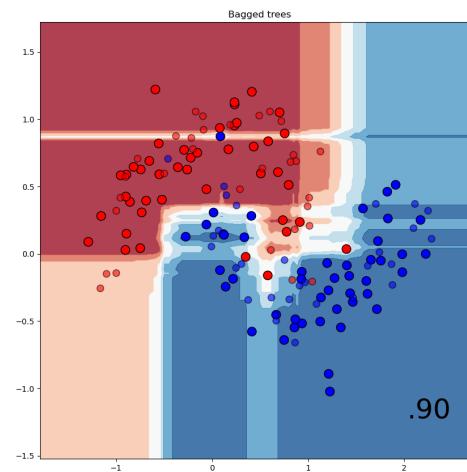
- Ex. de code Python pour les *bagged trees* pour la tâche de catégorisation avec tuning de plus. hyperparamètres par *grid search*.

```

1 from sklearn.ensemble import BaggingClassifier
2 # Define the model and fit on training data
3 bagging = BaggingClassifier(base_estimator=DecisionTreeClassifier())
4 bagging_params = {'n_estimators': [10, 50, 100], 'max_samples': [0.5, 1.0]} # Parameters to test
5 # Use GridSearchCV to find the best parameters using 5-fold cross-validation
6 bagging_cv = GridSearchCV(bagging, bagging_params, cv=5, scoring='accuracy', n_jobs=4)
7 bagging_cv.fit(X_train, y_train)
8 score = bagging_cv.score(X_test, y_test)
9 # Instantiate an empty figure and plot the decision contours
10 figure = plt.figure(figsize=(10, 10))
11 ax = plt.subplot(1, 1, 1)
12 DecisionBoundaryDisplay.from_estimator(bagging_cv, X, cmap=cm, alpha=0.8, ax=ax, eps=0.5)
13 # Plot the training points then the test points
14 ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright, edgecolors="k", s=125)
15 ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, edgecolors="k", alpha=0.6, s=75)
16 ax.set_xlim(x_min, x_max)
17 ax.set_ylim(y_min, y_max)
18 ax.set_title("Bagged trees")
19 ax.text(x_max - 0.3, y_min + 0.3, ("%.2f" % score).lstrip("0"), size=35, horizontalalignment="right")
```

## \*\*Code\*\* Bagged trees, Two moons (suite)

- Lignes de niveau des scores de l'arbre de décision appris.



## Réduire la variance ... encore et toujours !

- Pour mieux comprendre les fondements des forêts aléatoires, il est utile de rappeler au préalable quelques résultats en probabilité.
- Soit  $\mathcal{Z}_1, \dots, \mathcal{Z}_t$ , des v.a. identiquement distribuées de variance  $\sigma^2$ .
- Soit  $\bar{\mathcal{Z}} = \frac{1}{t} \sum_{b=1}^t \mathcal{Z}_b$ .
- La variance de  $\bar{\mathcal{Z}}$  vaut :

$$\rho\sigma^2 + \frac{1-\rho}{t}\sigma^2$$

où  $\rho$  est la corrélation supposée positive entre deux va.

- Si les  $\mathcal{Z}^b$  sont de plus indépendantes ( $\rho = 0$ ), alors la variance de  $\bar{\mathcal{Z}}$  est plus petite et se réduit à :

$$\frac{\sigma^2}{t}$$

## \*\*Code\*\* Bagged trees, AirBnB

- Ex. de code Python pour les *bagged trees* pour la tâche de régression avec tuning de plus. hyperparamètres par *grid search*.

```

1 from sklearn.ensemble import BaggingRegressor
2
3 # Bagging Regressor (ensemble of decision trees) with hyperparameter tuning (n_
4 # estimators, max_samples, max_features)
5 bagging = BaggingRegressor()
6 bagging_params = {'n_estimators': [50, 100], 'max_samples': [0.5, 1.0], 'max_features':
7 : [0.5, 1.0]} # Parameters to test
8
9 # Use GridSearchCV to find the best combination of parameters using 5-fold cross-
10 # validation
11 bagging_cv = GridSearchCV(bagging, bagging_params, cv=5, scoring='neg_mean_squared_
12 error', n_jobs=4)
13 bagging_cv.fit(X_train, y_train)
14
15 # Print the best parameters and the RMSE on the test set
16 print('Best Bagging params:', bagging_cv.best_params_)
17 y_pred_bagging = bagging_cv.predict(X_test)
18 print('Bagging RMSE:', np.sqrt(mean_squared_error(y_test, y_pred_bagging)))

```

```

1 Best Bagging params: {'max_features': 1.0, 'max_samples': 0.5, 'n_estimators': 50}
2 Bagging RMSE: 0.42821018686145423

```

## Rappel du Sommaire

- 5 Méthodes d'ensemble en ML et extensions spatiales
  - Arbres de décision
    - Cas du problème de catégorisation
    - Cas du problème de régression
    - Compléments sur les arbres de décision
  - Bagging
  - Forêts aléatoires (RF)
    - Gradient boosting (GB)
    - Moran Eigenvector Maps (MEM)
    - Geographical Random Forest (GRF)

## Les forêts aléatoires

- Dans le bagging, du fait du tirage aléatoire avec remplacement du bootstrap, les échantillons ne sont pas indépendants. Donc, les fonctions de prédiction apprises sur ces échantillons ne le sont pas non plus ! Nous sommes ainsi uniquement dans le contexte "identiquement distribué" et nous voulons aller vers la situation "**indépendant et id.**".
- L'objectif des forêts aléatoires est donc de réduire la variance du *bagging* en réduisant la corrélation entre les AD
- Pour ce faire, l'idée est d'ajouter de l'aléatoire dans l'induction d'un arbre en tirant au hasard un sous-ensemble de variables pour être candidat à la division.

## Forêt aléatoires : pseudo-code

**Input :**  $\mathbb{E}, \theta$  (seuil de pureté),  $t$  (nb. d'échantillons bootstrap)

1   **Pour tout**  $b = 1, \dots, t$  **faire**

2     Déterminer un échantillon bootstrap  $\mathbb{E}^b$

3     Induire un AD  $\hat{g}^b$  à partir de  $\mathbb{E}^b$  en appliquant la procédure :

4       **Tant que** l'arbre n'est pas globalement pur

5         Sélectionner aléatoirement  $r$  attributs

6         Déterminer la meilleure division parmi ces  $r$  variables

7         Séparer le nœud selon la division précédente

8     **Fin Tant que**

9   **Fin Pour**

10   **Output :**  $\{\hat{g}^b\}_{b=1, \dots, t}$

## Les forêts aléatoires (suite)

- Plus spécifiquement : avant chaque division d'un nœud  $m$ , on choisit aléatoirement  $r$  ( $r \leq p$ ) attributs comme candidats à la division.
- Intuitivement, si  $r$  est petit alors les arbres appris sur différents échantillons bootstrap sont de moins en moins corrélés et leur agrégation sera de variance plus petite.
- Les forêts aléatoires utilisent à la fois plusieurs ensembles d'entraînement et plusieurs espaces de description (principe du sous-espace aléatoire ou **random subspace**).

## Prédiction des forêts aléatoires

- A partir des  $t$  AD estimés  $\{\hat{f}^b\}_{b=1, \dots, t}$  on calcule les prédictions de la façon suivante.
- Soit  $\mathbf{x}$  un objet quelconque représenté dans  $\mathbb{X}$ .
- S'il s'agit d'un problème de régression :

$$\hat{f}_{rf}(\mathbf{x}) = \frac{1}{t} \sum_{b=1}^t \hat{g}^b(\mathbf{x})$$

- S'il s'agit d'un problème de catégorisation :

$$\hat{f}_{rf}(\mathbf{x}) = \arg \max_{C_{l'} \in \mathbb{Y}} \sum_{b=1}^t \text{Ind}(\hat{g}^b(\mathbf{x}) = C_{l'})$$

## Tuning des hyperparamètres

- Il est connu que les RF sont des méthodes performantes mais il est toutefois nécessaire d'estimer les 3 hyperparamètres suivants de façon appropriée [Cutler et al., 2012] :
  - r le nb. de variables choisis aléatoirement à chaque noeud,
  - t le nb. d'échantillons bootstrap (égal au nb. d'arbres dans la forêt),
  - un paramètre influant sur la taille de l'arbre :
    - $\theta$  le seuil d'impureté,
    - nb. minimal d'éléments dans un noeud pour pouvoir être séparé,
    - nb. maximal de feuilles dans un arbre,
    - ...
- L'hyperparamètre le plus sensible est r.
  - Par défaut, il est fixé à :
    - $r = \sqrt{p}$  pour la tâche de catégorisation,
    - $r = n/3$  pour la tâche de régression.
  - Sinon, on utilise l'erreur OOB pour choisir r parmi un ens. de valeurs.

## \*\*Code\*\* Random forest, Two moons

- Ex. de code Python pour les *random forest* pour la tâche de catégorisation avec tuning de plus. hyperparamètres par *grid search*.

```

1 from sklearn.ensemble import RandomForestClassifier
2 # Define the model and fit on training data
3 rf = RandomForestClassifier()
4 rf_params = {'n_estimators': [50, 100, 200], 'max_depth': [None, 5, 10], 'min_samples_
5 split': [2, 5]} # Parameters to test
6 rf_cv = GridSearchCV(rf, rf_params, cv=5, scoring='accuracy', n_jobs=4)
7 rf_cv.fit(X_train, y_train)
8 score = rf_cv.score(X_test, y_test)
9 # Instanciate an empty figure and plot the decision contours
10 figure = plt.figure(figsize=(10, 10))
11 ax = plt.subplot(1, 1, 1)
12 DecisionBoundaryDisplay.from_estimator(rf_cv, X, cmap=cm, alpha=0.8, ax=ax, eps=0.5)
13 # Plot the training points then the test points
14 ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright, edgecolors="k", s_
15 =125)
16 ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, edgecolors="k", alpha_
17 =0.6, s=75)
18 ax.set_xlim(x_min, x_max)
19 ax.set_ylim(y_min, y_max)
20 ax.set_title("Random forest")
21 ax.text(x_max - 0.3, y_min + 0.3, ("%.2f" % score).lstrip("0"), size=35,
22 horizontalalignment="right")

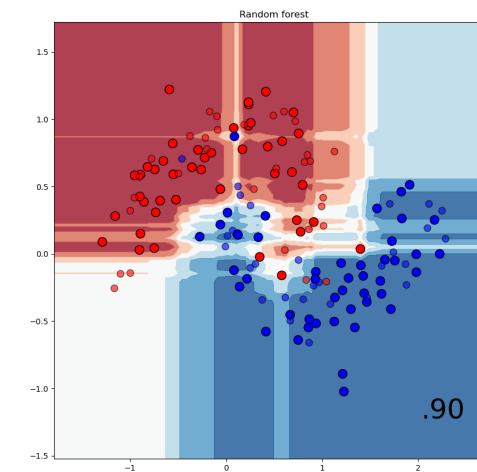
```

## Tuning des hyperparamètres (suite)

- Concernant t, plus il est grand plus l'erreur en généralisation converge vers une valeur fixe :
  - Pas de pb. de sur-apprentissage associé à un t trop grand,
  - t ne doit pas être petit en revanche.
- Concernant la taille des arbres :
  - Breiman recommandait dans son travail original [Breiman, 2001] de prendre des arbres profonds ( $\theta$  très petit par exemple).
  - Des travaux postérieurs [Segal and Xiao, 2011], indique qu'il peut avoir un intérêt à estimer la taille de l'arbre (en raisonnant avec l'un des trois paramètres sus-mentionnés) à l'aide de l'erreur OOB.

## \*\*Code\*\* Random forest, Two moons (suite)

- Lignes de niveau des scores du *random forest* appris.



## Importance des variables

- Pouvoir quantifier l'importance d'une variable  $X^j$  au sein d'un modèle d'apprentissage est une propriété attrayante pour différentes raisons :
  - Cela favorise la compréhension et l'interprétation du phénomène qui a généré les données d'observations. En particulier, on peut classer les attributs afin de connaître ceux qui ont permis de mieux prédire la variable cible  $Y$ .
  - Cela permet d'avoir du recul sur l'espace de description utilisé dans le modèle et faire, par exemple, de la sélection de variables ce qui pourrait potentiellement améliorer les performances.
- Dans le contexte des méthodes basées sur les AD, il existe plusieurs approches pour quantifier l'importance des variables. Nous verrons les deux suivantes :
  - Méthode basée sur la réduction de l'impureté des noeuds des arbres.
  - Méthode (plus robuste) basée sur la permutation des valeurs d'un attribut.

## \*\*Code\*\* Random forest, AirBnB

- Ex. de code Python pour le *random forest* pour la tâche de régression avec tuning de plus. hyperparamètres par *grid search*.

```

1 from sklearn.ensemble import RandomForestRegressor
2
3 # Random Forest Regressor with hyperparameter tuning (n_estimators, max_depth, min_
4 # samples_split)
5 rf = RandomForestRegressor()
6 rf_params = {'n_estimators': [50, 100], 'min_samples_split': [2, 5, 10], 'max_depth':
7 [3, 6, 10]} # Parameters to test
8
9 # Use GridSearchCV to find the best combination of parameters using 5-fold cross-
10 # validation
11 rf_cv = GridSearchCV(rf, rf_params, cv=5, scoring='neg_mean_squared_error', n_jobs=4)
12 rf_cv.fit(X_train, y_train)
13
14 # Print the best parameters and the RMSE on the test set
15 print('Best Random Forest params:', rf_cv.best_params_)
16 y_pred_rf = rf_cv.predict(X_test)
17 print('Random Forest RMSE:', np.sqrt(mean_squared_error(y_test, y_pred_rf)))

```

```

1 Best Random Forest params: {'max_depth': 10, 'min_samples_split': 5, 'n_estimators':
2 100}
3 Random Forest RMSE: 0.42681453756242743

```

## Importance des variables basée sur l'impureté

- Dans la construction d'un AD, pour chaque noeud  $m$ , on choisit la variable  $X^{j*}$  qui permet de réduire le plus l'entropie cf slide 277. Cela est équivalent à maximiser la réduction moyenne de l'entropie (*Mean Decrease Impurity -MDI-*). Cette mesure notée  $\Delta_j^m$  est donnée par :

$$\Delta_j^m = \text{Ent}(p^m) - \text{Ent}(p^m, X^j)$$

- Notons par  $\text{Imp}^j$  la mesure de l'importance de  $X^j$  dans les prédictions du modèle ensembliste estimé. Dans le contexte d'un ensemble d'arbres comme le RF, Breiman (cf [Louppe et al., 2013]) propose de mesurer  $\text{Imp}^j$  en moyennant sur l'ensemble des arbres les  $\Delta_j^m$  pour tout noeud  $m$  où  $X^j$  est utilisé pour la séparation :

$$\text{Imp}^j = \frac{1}{t} \sum_{b=1}^t \sum_{m=1}^{\hat{s}^b} \frac{\mathbf{N}^m}{n} \Delta_j^m \text{Ind}(\text{"}X^j \text{ sépare } m\text{"})$$

où  $\hat{s}^b$  est le nb. de noeuds de l'arbre  $\hat{g}^b$  (cf notations slide 277).

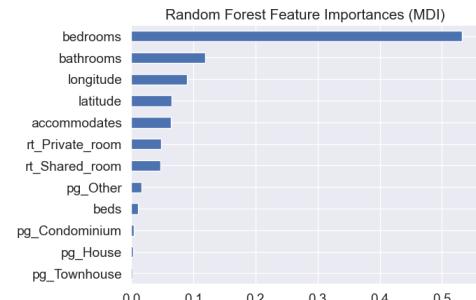
## \*\*Code\*\* Random forest, Var. Imp. MDI, AirBnB

- Ex. de code Python permettant de visualiser l'importance des variables mesurée par le MDI issue d'un *random forest*.

```

1 rf = RandomForestRegressor(**rf_cv.best_params_)
2 rf.fit(X_train, y_train)
3 feature_names = variable_names
4 mdi_importances = pd.Series(rf.feature_importances_, index=feature_names).sort_values(ascending=True)
5 ax = mdi_importances.plot.barh()
6 ax.set_title("Random Forest Feature Importances (MDI)")

```



## Importance des variables basée sur les permutations

- Les RF permettent également d'obtenir une mesure de l'importance d'une var. en exploitant les prédictions du modèle inféré et en permutant les valeurs de la var. choisie.
- L'approche qui est propre aux RF procède comme suit. Supposons que nous nous intéressions à la variable  $X^j$  :
  - Pour chaque arbre estimé, calculer les prédictions des objets OOB.
  - Les valeurs de la variable  $X^j$  sont permutees parmi les objets OOB (les valeurs des autres variables restent inchangées).
  - Calcul des prédictions des objets OOB modifiés pour  $X^j$ . Ainsi pour chaque objet OOB, on a deux prédictions l'une pour l'objet authentique et l'autre pour l'objet modifié pour  $X^j$ .
  - Pour chq. objet OOB, calcul de la différence entre les erreurs commises entre les préd. des objets modifiés et celles des objets authentiques.
  - Moyenner les différences des erreurs individuelles précédentes pour avoir une estimation globale de l'importance de  $X^j$ .

## Importance des variables : remarques complémentaires

- Plus  $\text{Imp}^j$  est grand plus les changements des valeurs de la variable  $X^j$  impacte la qualité des prédictions et plus  $X^j$  est importante.
- L'approche peut quantifier une **importance locale** à un individu ou groupe d'individus en utilisant qu'une ou une partie des valeurs  $\text{Imp}_i$ .
- En catégorisation, l'approche peut quantifier l'importance d'une variable de façon différenciée **selon les classes** de  $\mathbb{Y}$ . Comme précédemment, on utilise uniquement les  $\text{Imp}_i$  relatifs à la (ou aux) classe(s) choisie(s).
- Cette approche qui est basée sur les **permutations** est plus **robuste** que celle où on compare les perf. en considérant  $X^j$  dans l'ens. des var. *versus* celles que l'on obtient quand  $X^j$  est retiré de l'ens. des var. En effet, si  $X^j$  est corrélée à d'autres var., sa suppression pourrait être compensée par ces dernières et dans ce cas, la variation des perf. obtenue n'est pas un bon indicateur de l'importance de  $X^j$ . La permutation des valeurs permet de contourner ce défaut.

## Imp. des variables basée sur les permutations : pseudo-code

```

Input :  $\mathbb{E}$ ,  $\{\hat{g}^b, \mathbb{E}^b\}_{b=1,\dots,t}$ ,  $X^j$  (variable d'intérêt)
Pour tout  $x_i \in \mathbb{E}$  faire
  Déterminer  $\mathbb{J}_i = \{b : (x_i, y_i) \notin \mathbb{E}^b\}$ 
  Déterminer  $\hat{y}_{i,b} = \hat{g}^b(x_i)$  pour tout  $b \in \mathbb{J}_i$ 
Fin Pour
Pour tout  $b = 1, \dots, t$  faire
  Déterminer  $\mathbb{T}^b = \{(x_i, y_i) : (x_i, y_i) \notin \mathbb{E}^b\}$ 
  Permuter aléatoirement les valeurs  $x_{ij}$  parmi les objets de  $\mathbb{T}^b$ 
  Soit  $x'_i$  la version modifiée de  $x_i$ . Soit  $\mathbb{T}'^b = \{(x'_i, y_i) : (x_i, y_i) \in \mathbb{T}^b\}$ 
  Déterminer  $\hat{y}'_{i,b} = \hat{g}^b(x'_i)$  pour tout  $x'_i \in \mathbb{T}'^b$ 
Fin Pour
Pour tout  $x_i \in \mathbb{E}$  faire
   $\text{Imp}_i = \frac{1}{|\mathbb{J}_i|} \sum_{b \in \mathbb{J}_i} (\text{Ind}(y_i \neq \hat{y}'_{i,b}) - \text{Ind}(y_i \neq \hat{y}_{i,b}))$  si catégorisation
   $\text{Imp}_i = \frac{1}{|\mathbb{J}_i|} \sum_{b \in \mathbb{J}_i} ((y_i - \hat{y}'_{i,b})^2 - (y_i - \hat{y}_{i,b})^2)$  si régression
Fin Pour
Output :  $\text{Imp}^j = \frac{1}{n} \sum_{i=1}^n \text{Imp}_i$ 

```

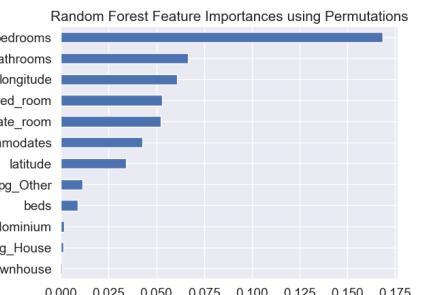
## \*\*Code\*\* Random forest, Var. Imp. Permutations, AirBnB

- Ex. de code Python permettant de calculer et de visualiser l'importance des variables selon la méthode basée sur les permutations.
- Illustration avec la méthode *random forest* mais la fonction et l'approche **peut être appliquée à toute autre technique de ML**.

```

1 from sklearn.inspection import permutation_importance
2 # Assuming rf_cv is a fitted
# RandomForestRegressor and X_test, y_test are test data
3 feature_perm_importances = permutation_importance_(rf_cv, X_test, y_test,
4 scoring='neg_mean_squared_error', n_repeats=30, random_state=42)
5 feature_names = variable_names
6 perm_importance = pd.Series(feature_perm_importances.importances.mean(axis=1),
7 , index=feature_names).sort_values(ascending=True)
8 ax = perm_importance.plot.barh()
9 ax.set_title("Random Forest Feature Importances using Permutations")

```



## Rappel du Sommaire

### 5 Méthodes d'ensemble en ML et extensions spatiales

- Arbres de décision
  - Cas du problème de catégorisation
  - Cas du problème de régression
  - Compléments sur les arbres de décision
- Bagging
- Forêts aléatoires (RF)
- Gradient boosting (GB)
- Moran Eigenvector Maps (MEM)
- Geographical Random Forest (GRF)

## Gradient Boosting (suite)

- En effet, la méthode s'apparente à la procédure de **descente de gradient avec un pas optimal** en optimisation numérique.
- Toutefois, dans le cas du Gradient Boosting, la descente de gradient se fait dans un **espace de fonctions**.
- Notre problème général est tjs le suivant :

$$\min_f \text{E}_{X,Y}(\ell(Y, f(X))) = \min_f \text{E}_X(\text{E}_{Y|X}(\ell(Y, f(X))|X))$$

où  $\ell$  est une fct. de perte que l'on va supposer différentiable.

- Par exemple :

- ▶  $\ell_2(Y, f(X)) = (Y - f(X))^2$ ,
- ▶  $\ell_{\exp}(Y, f(X)) = \exp(-Yf(X)), \dots$

## Gradient Boosting

- Le Gradient Boosting (GB) est une famille de techniques introduite par Friedman en 1999 [Friedman, 2001].
- C'est une méthode de Boosting donc :
  - ▶ il s'agit d'une approche itérative,
  - ▶ à chq. itération une pondération différente des objets est utilisée,
  - ▶ à chq. itération un *weak learner* est estimé puis ajouté séquentiellement au strong(er) learner.
- Quelle est la différence avec AdaBoost ?
- Il s'agit d'un cadre méthodologique plus général et plus élégant : **toute fonction de perte différentiable**  $\ell$  peut être utilisée.
- Les travaux précédents de Friedman et al. avaient déjà permis de rapprocher le Boosting du FSMA comme vu précédemment. Avec le Gradient Boosting, c'est un autre point de vue éclairant sur le bien fondé du Boosting qui est apporté.

## Gradient Boosting (suite)

- Nous avons traité jusqu'à présent des **modèles paramétriques** qui supposent que  $f$  est un élément de  $\mathbb{H}$  (espace d'hypothèses) qui dépend d'un ensemble fini de paramètres.
- Nous supposons en particulier que  $f$  est un **modèle additif** :

$$f(\mathbf{x}; \{\beta^{(b)}, \mathbf{c}^{(b)}\}_{b=1,\dots,t}) = \sum_{b=1}^t \beta^{(b)} g^{(b)}(\mathbf{x}; \mathbf{c}^{(b)})$$

- La fct. de base  $g(\mathbf{x}; \mathbf{c})$  est un *weak learner* (c.à.d. une fct. paramétrique simple) caractérisée par le vecteur de paramètres  $\mathbf{c}$ .
- Dans le contexte du Boosting, on s'intéresse particulièrement aux AD et dans ce cas,  $\mathbf{c}$  représente :
  - ▶ les variables de séparation,
  - ▶ les branches et paramètres de séparation associés
  - ▶ les valeurs estimées dans chaque feuille de l'arbre  $g^{(b)}$ .

## GB : optimisation numérique cas paramétrique

- Dans le cadre d'un **modèle paramétrique**, résoudre  $\arg \min_{f \in \mathbb{H}} \text{Ex}(\mathbb{E}_{\mathcal{Y}|X}(\ell(\mathcal{Y}, f(X))|X))$  revient à résoudre :

$$\arg \min_{\mathbf{a}} \text{Ex}(\mathbb{E}_{\mathcal{Y}|X}(\ell(\mathcal{Y}, f(\mathbf{X}; \mathbf{a}))|X))$$

- Pour la plupart des fct.  $\ell$ , on a recours à des algo. d'optim. num., ce qui implique souvent que la solution se construit itérativement et s'exprime comme suit ( $f$  est une fct. dont les param. sont mis à jour) :

$$\hat{\mathbf{a}}^* = \sum_{b=0}^t \hat{\mathbf{c}}^{(b)} \text{ et } \hat{f}^*(\mathbf{x}) = f(\mathbf{x}; \hat{\mathbf{a}}^*)$$

où  $\hat{\mathbf{c}}_0$  est une solution initiale (souvent choisie aléatoirement) et les  $\hat{\mathbf{c}}^{(b)}$  sont des incrément successifs dont la forme particulière dépend de l'algo. d'optim. utilisé.

## GB : optimisation numérique ds un espace de fonctions

- Nous nous plaçons **maintenant dans un cadre non paramétrique** et appliquons la descente de gradient dans un **espace de fonctions**.
- Nous définissons une fct. objectif  $\phi$  qui correspond à l'espérance de la **fct. de perte**  $\ell$  mais qui **dépend de  $f$  directement** :

$$\phi(f) = \text{Ex}(\mathbb{E}_{\mathcal{Y}|X}(\ell(\mathcal{Y}, f)|X))$$

- $f : \mathbb{X} \rightarrow \mathbb{Y}$  est donc la variable de  $\phi$ . L'**espace auquel appartient  $f$**  n'est plus de dimension fini comme auparavant.  $f$  étant une fonction, elle appartient à un **ensemble de dimension infinie**.
- Pour fixer les idées vous pouvez voir chaque évaluation de  $f$  en un  $\mathbf{x} \in \mathbb{X}$  comme une "dimension" de  $f$  et comme il y a une infinité de points dans  $\mathbb{X}$ ,  $f$  est donc dans un espace de dimension infinie.
- Nous allons donc chercher à **min.  $\phi$  en fct. de  $f$  directement**.

## GB : optimisation numérique cas paramétrique (suite)

- Nous nous intéressons en particulier à la méthode de **descente de gradient à pas optimal** (*steepest (gradient) descent*) qui définit l'incrément  $\mathbf{c}^{(b)}$  à partir du gradient  $\mathbf{h}^{(b)}$  de  $\ell$  défini comme suit :

$$h_i^{(b)} = \frac{\partial \ell(\mathbf{a})}{\partial a_i} \Big|_{\mathbf{a}=\mathbf{a}^{(b-1)}}, \quad i = 1, \dots, p$$

où  $\mathbf{a}^{(b-1)} = \sum_{b'=0}^{(b-1)} \mathbf{c}^{(b')}$ .

- L'incrément suit la direction opposée du gradient qui correspond à la pente la plus "raide" pour s'approcher d'un minimiseur :

$$\mathbf{c}^{(b)} = -\alpha^{(b)} \mathbf{h}^{(b)}$$

- Dans cette direction, l'incrément s'arrête au pas  $\alpha^{(b)}$  qui minimise optimalement la fct. de perte :

$$\alpha^{(b)} = \arg \min_{\alpha \in \mathbb{R}} \text{Ex}(\mathbb{E}_{\mathcal{Y}|X}(\ell(\mathcal{Y}, f(\mathbf{x}; \mathbf{a}^{(b-1)} - \alpha \mathbf{h}^{(b)}))|X))$$

## GB : optimisation num. ds un espace de fonctions (suite)

- On utilise la **descente de gradient vue comme un modèle additif** :

$$f^* = \sum_{b=0}^t g^{(b)}$$

où  $g^0$  est une fct. initiale et les fct.  $g^{(b)}$  sont des incrément successifs.

- On peut raisonner pour un  $\mathbf{x}$  donné et se focaliser comme on l'a vu à plusieurs reprises sur l'espérance conditionnelle  $\mathcal{Y}|\mathbf{x}$  de la fct. de perte et minimiser de façon équivalente :

$$\phi(f(\mathbf{x})) = \mathbb{E}_{\mathcal{Y}|\mathbf{x}}(\ell(\mathcal{Y}, f(\mathbf{x}))|\mathbf{x})$$

- Pour la descente de gradient on a l'expression suivante où  $h^{(b)}(\mathbf{x})$  dénote la fct. **gradient de  $\phi(f)$**  et  $\alpha^{(b)} > 0$  le pas :

$$g^{(b)}(\mathbf{x}) = -\alpha^{(b)} h^{(b)}(\mathbf{x})$$

## GB : optimisation num. ds un espace de fonctions (suite)

- La fonction gradient  $h^{(b)} = \nabla_f \phi(f)$  évaluée en  $f^{(b-1)}$  est donnée par :

$$h^{(b)}(\mathbf{x}) = \frac{\partial \phi(f(\mathbf{x}))}{\partial f(\mathbf{x})} \Big|_{f(\mathbf{x})=f^{(b-1)}(\mathbf{x})} = \frac{\partial E_{\mathcal{Y}|\mathbf{X}}(\ell(\mathcal{Y}, f(\mathbf{x}))|\mathbf{x})}{\partial f(\mathbf{x})} \Big|_{f(\mathbf{x})=f^{(b-1)}(\mathbf{x})}$$

où  $f^{(b-1)}(\mathbf{x}) = \sum_{b=0}^{(b-1)} g^{(b)}(\mathbf{x})$ .

- En supposant que les fonctions en jeux sont suffisamment régulières (ou lisses) de sorte à pouvoir interchanger l'opérateur de différentiation et celui d'intégration relatif à l'espérance, on a :

$$h^{(b)}(\mathbf{x}) = E_{\mathcal{Y}|\mathbf{X}} \left( \frac{\partial \ell(\mathcal{Y}, f(\mathbf{x}))}{\partial f(\mathbf{x})} \Big|_{f(\mathbf{x})=f^{(b-1)}(\mathbf{x})} \mid \mathbf{x} \right)$$

- Le pas est obtenu comme précédemment en résolvant :

$$\alpha^{(b)} = \arg \min_{\alpha \in \mathbb{R}} E_{\mathbf{X}}[E_{\mathcal{Y}|\mathbf{X}}[\ell(\mathcal{Y}, f^{(b-1)}(\mathbf{x}) - \alpha h^{(b)}(\mathbf{x}))|\mathbf{X} = \mathbf{x}]]$$

## GB : ens. fini d'obs. (suite)

- Le pb. précédent est complexe car il cherche à estimer d'un seul coup tous les paramètres des fct. paramétrées participant au modèle additif.
- En pratique, on adopte l'approche *Forward Stagewise* et on considère itérativement :

$$f^b(\mathbf{x}) = f^{(b-1)}(\mathbf{x}) + \beta^{(b)} g^{(b)}(\mathbf{x}; \mathbf{c}^{(b)})$$

- On fait clairement le lien avec le point de vue FSAM du Boosting.
- On a ici un autre point de vue qui est celui de la descente de gradient dans un espace de fcts que l'on représente sous forme paramétrique en raison de la taille limitée des données observées dans l'échantillon.
- Qu'apporte alors ce point de vue au Boosting ?
- Le fait d'avoir supposé initialement des fcts lisses notamment en ce qui concerne la fct. de perte, cela permet d'appliquer ce cadre générale à tout type de fct. de perte différentiable !

## GB : ens. fini d'obs.

- Pour estimer les fct.  $g^{(b)}$ , nous n'avons en revanche que les observations de  $\mathbb{E} = \{(y_i, \mathbf{x}_i)\}_{i=1,\dots,n}$ . Dans ce cas, l'estimation de  $\phi(f)$  ne peut pas être très précise en tout  $\mathbf{x} \in \mathbb{X}$ .
- La solution consiste alors à prendre pour  $f$  **une forme paramétrée et de type additive model** ( $f$  est une somme de fcts) :

$$f(\mathbf{x}; \{\beta^{(b)}, \mathbf{c}^{(b)}\}_{b=1,\dots,t}) = \sum_{b=1}^t \beta^{(b)} g^{(b)}(\mathbf{x}; \mathbf{c}^{(b)})$$

- L'estim. de  $f$  passe ainsi par l'estim. des param.  $\{\beta^{(b)}, \mathbf{c}^{(b)}\}_{b=1,\dots,t}$  en visant la min. de l'estim. empirique de l'espérance de la fct. de perte :

$$\{\hat{\beta}^{(b)}, \hat{\mathbf{c}}^{(b)}\}_{b=1,\dots,t} = \arg \min_{\{\beta^{(b)}, \mathbf{c}^{(b)}\}_{b=1,\dots,t}} \sum_{i=1}^n \ell(y_i, \sum_{b=1}^t \beta^{(b)} g^{(b)}(\mathbf{x}_i; \mathbf{c}^{(b)}))$$

## GB : gradient et pseudo-erreurs

- Nous traitons le pb. de façon itérative et additive. Etant donné  $\hat{f}^{(b-1)}(\mathbf{x})$  estimé à partir des données, on cherche à déterminer au coup d'après, le meilleur *boost*,  $\beta^{(b)} g^{(b)}(\mathbf{x}; \mathbf{c}^{(b)})$ , permettant de se rapprocher au mieux de la fct. optimale.

$$(\hat{\beta}^{(b)}, \hat{\mathbf{c}}^{(b)}) = \arg \min_{\beta, \mathbf{c}} \sum_{i=1}^n \ell(y_i, \hat{f}^{(b-1)}(\mathbf{x}_i) + \beta^{(b)} g^{(b)}(\mathbf{x}_i; \mathbf{c}^{(b)}))$$

- Notre stratégie d'optimisation étant la descente de gradient, nous pouvons évaluer à l'itération  $b$  les valeurs du gradient  $\partial \phi(f(\mathbf{x}))/\partial f(\mathbf{x})$  en les  $\mathbf{x}_i, \forall i = 1, \dots, n$  :

$$\hat{h}^{(b)}(\mathbf{x}_i) = \frac{\partial \ell(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)} \Big|_{f(\mathbf{x})=\hat{f}^{(b-1)}(\mathbf{x})}$$

- Les estimations obtenues à partir de l'échantillon  $\mathbb{E}$ , nous permettent de définir un vecteur gradient de ces  $n$  données :

$$\hat{\mathbf{h}}^{(b)} = (\hat{h}^{(b)}(\mathbf{x}_i))_{i=1,\dots,n}$$

## GB : gradient et pseudo-erreurs (suite)

- Le vecteur  $\hat{\mathbf{h}}^{(b)}$  nous donne la descente la plus "raide" pour s'approcher d'un minimiseur mais dans l'espace de l'échantillon.
- En effet, ce vecteur est défini uniquement pour les objets de l'échantillon et ne peut pas être généralisé aux autres données  $\mathbf{x} \in \mathbb{X}$ .
- C'est à ce niveau que nous estimons notre fonction gradient à partir d'un modèle paramétrique  $g^{(b)}(\mathbf{x}; \mathbf{c}^{(b)})$  où  $\mathbf{c}^{(b)}$  est le vecteur de paramètres que l'on doit estimer.
- On doit faire en sorte à ce que  $g^{(b)}(\cdot; \mathbf{c}^{(b)})$  approxime  $-\hat{\mathbf{h}}^{(b)}(\cdot)$  sur la base des observations fournies par les objets de  $\mathbb{E}$  et on résoud :

$$\hat{\mathbf{c}}^{(b)} = \arg \min_{\mathbf{c}, \gamma} \sum_{i=1}^n (-\hat{\mathbf{h}}^{(b)}(\mathbf{x}_i) - \gamma g^{(b)}(\mathbf{x}_i; \mathbf{c}))^2$$

C'est un problème d'estimation par MCO, la variable  $\gamma$  peut être vue comme un facteur de mise à l'échelle (scaling).

- $\hat{g}^{(b)}(\cdot; \hat{\mathbf{c}}^{(b)})$  permet alors de prédire l'opposée de la fct. gradient  $\forall \mathbf{x} \in \mathbb{X}$  !

## GB : pseudo-code

- Ainsi, pour tout modèle paramétrique  $g(\mathbf{x}; \mathbf{c})$  pour lequel il est possible d'appliquer les MCO pour approximer les pseudo-erreurs, il est possible d'utiliser l'approche décrite pour minimiser une fct. de perte différentiable  $\ell$ , en conjonction avec FSAM.

**Input :**  $\mathbb{E}, t$  (nb. d'itérations),  $g$  (un weak learner)

$$1 \quad \hat{f}^{(0)}(\mathbf{x}) = \arg \min_{\gamma} \sum_{i=1}^n \ell(y_i, \gamma) \quad (\text{fct. constante})$$

2 **Pour tout**  $b = 1, \dots, t$  **faire**

3      Calculer  $\tilde{y}_i = -\hat{\mathbf{h}}^{(b)}(\mathbf{x}_i)$ ,  $\forall i = 1, \dots, n$

$$5 \quad \text{Résoudre } \hat{\mathbf{c}}^{(b)} = \arg \min_{\mathbf{c}, \gamma} \sum_{i=1}^n (\tilde{y}_i - \gamma g^{(b)}(\mathbf{x}_i; \mathbf{c}))^2 \quad (\text{MCO})$$

$$6 \quad \text{Résoudre } \hat{\beta}^{(b)} = \arg \min_{\beta} \sum_{i=1}^n \ell(y_i, \hat{f}^{(b-1)}(\mathbf{x}_i) + \beta \hat{g}^{(b)}(\mathbf{x}_i; \hat{\mathbf{c}}^{(b)}))$$

$$7 \quad \text{Mettre à jour } \hat{f}^{(b)}(\mathbf{x}) = \hat{f}^{(b-1)}(\mathbf{x}) + \hat{\beta}^{(b)} \hat{g}^{(b)}(\mathbf{x}; \hat{\mathbf{c}}^{(b)})$$

8 **Fin Pour**

9 **Output :**  $\{\hat{g}^{(b)}, \hat{\mathbf{c}}^{(b)}, \hat{\beta}^{(b)}\}_{b=1, \dots, t}$

- La prédiction est donnée par :  $\hat{f}_{gb}(\mathbf{x}) = \hat{f}^0(\mathbf{x}) + \sum_{b=1}^t \hat{\beta}^{(b)} \hat{g}^{(b)}(\mathbf{x}; \mathbf{c}^b)$

## GB : gradient et pseudo-erreurs (suite)

- $-\hat{\mathbf{h}}^{(b)}(\cdot)$  est remplacée par son approximation paramétrique  $\hat{g}^{(b)}(\cdot; \hat{\mathbf{c}}^{(b)})$ . Le pas de la descente de gradient est donné par :

$$\hat{\beta}^{(b)} = \arg \min_{\beta \in \mathbb{R}} \sum_{i=1}^n \ell(y_i, \hat{f}^{(b-1)}(\mathbf{x}_i) + \beta \hat{g}^{(b)}(\mathbf{x}_i; \hat{\mathbf{c}}^{(b)}))$$

- La mise à jour de la fct. visée est alors obtenue comme suit :

$$\hat{f}^{(b)}(\mathbf{x}) = \hat{f}^{(b-1)}(\mathbf{x}) + \hat{\beta}^{(b)} \hat{g}^{(b)}(\mathbf{x}; \hat{\mathbf{c}}^{(b)})$$

comme annoncé en slide 339 et slide 332.

- Le point saillant ici est le fait que la fct.  $\hat{g}^{(b)}(\cdot; \hat{\mathbf{c}}^{(b)})$ , qui participe au modèle additif, est une approx. paramétrique de l'opposée de la fct. gradient  $-\hat{\mathbf{h}}^{(b)}(\cdot)$ , dont l'estimation est opérée sur la base des données  $\{\tilde{y}_i = -\hat{\mathbf{h}}^{(b)}(\mathbf{x}_i)\}$  que l'on appelle communément "pseudo-erreurs".

## GB : AD comme weak learner

- Nous développons en particulier le cas où  $g(\mathbf{x}; \mathbf{c})$ , le weak learner, est un AD. Dans ce cas, le vecteur de paramètre  $\mathbf{c}$  est composé de 2 types d'éléments  $\mathbf{c} = \{\mathbf{d}_m, \mathbb{S}_m\}_{m=1, \dots, s}$  où :

►  $\mathbb{S}_m \subset \mathbb{X}$  est une région de  $\mathbb{X}$  définie par une feuille de l'AD. Notons que ces régions sont mutuellement disjointes et qu'elles forment une partition de  $\mathbb{X}$ .

►  $\mathbf{d}_m$  sont les paramètres permettant de définir  $\mathbb{S}_m$  (chemin de l'arbre vers la feuille) et  $d_m \in \mathbf{d}_m$  la valeur associée à cette région. Par abus de notation nous supposerons la sémantique suivante : si  $\mathbf{x} \in \mathbb{S}_m$  alors  $g(\mathbf{x}, \mathbf{c}) = d_m$ .

- L'AD est lui-même un modèle additif que l'on peut écrire :

$$g(\mathbf{x}; \{\mathbf{d}_m, \mathbb{S}_m\}_m) = \sum_{m=1}^s d_m \text{Ind}(\mathbf{x} \in \mathbb{S}_m)$$

- Il est important de noter que l'AD est un arbre de régression appris par MCO (line 5) et qui vise à prédire les pseudo-erreurs qui sont des réels qu'il s'agisse d'une tâche de catégorisation ou de régression.

## GB : AD comme weak learner (suite)

- Dans le cas des AD comme méthode d'apprentissage de base, l'étape 7 du pseudo-code précédent du GB donné en slide 343 devient :

$$\hat{f}^{(b)}(\mathbf{x}) = \hat{f}^{(b-1)}(\mathbf{x}) + \hat{\beta}^{(b)} \underbrace{\sum_{m=1}^{\hat{s}^{(b)}} \hat{d}_m^{(b)} \text{Ind}(\mathbf{x} \in \hat{\mathbb{S}}_m^{(b)})}_{\hat{g}^{(b)}(\mathbf{x}; \hat{\mathbf{c}}^{(b)})}$$

où  $\hat{g}^{(b)}(\mathbf{x}; \hat{\mathbf{c}}^{(b)})$  avec  $\hat{\mathbf{c}}^{(b)} = \{\mathbf{d}_m^{(b)}, \hat{\mathbb{S}}_m^{(b)}\}_{m=1}^{\hat{s}^{(b)}}$  est l'AD appris pour prédire les pseudo-erreurs  $(\tilde{y}_i)_{i=1}^n$  de l'itération  $b$ .

- Dans l'expression précédente, nous pouvons distribuer le facteur  $\hat{\beta}^{(b)}$  et, en définissant  $\hat{\gamma}_m^{(b)} = \hat{\beta}^{(b)} \hat{d}_m^{(b)}$ , nous avons :

$$\hat{f}^{(b)}(\mathbf{x}) = \hat{f}^{(b-1)}(\mathbf{x}) + \sum_{m=1}^{\hat{s}^{(b)}} \hat{\gamma}_m^{(b)} \text{Ind}(\mathbf{x} \in \hat{\mathbb{S}}_m^{(b)})$$

## GB : AD comme weak learner, cas de la régression

- Dans le cas de la rég., l'utilisation des AD comme *weak learner* donne en sortie des régions qui sont indépendantes les unes des autres<sup>6</sup>.
- Reprendons l'expression vue en slide 346, permettant de formaliser, à l'itération  $b$ , la détermination du coefficient  $\gamma_m^{(b)}$  associé à une feuille  $\hat{\mathbb{S}}_m^{(b)}$  obtenue par l'AD appris pour prédire les pseudo-erreurs :

$$(\hat{\gamma}_m^{(b)})_{m=1, \dots, \hat{s}^{(b)}} = \arg \min_{\gamma \in \mathbb{R}^{\hat{s}^{(b)}}} \sum_{i=1}^n \ell(y_i, \hat{f}^{(b-1)}(\mathbf{x}_i) + \sum_{m=1}^{\hat{s}^{(b)}} \gamma_m \text{Ind}(\mathbf{x}_i \in \hat{\mathbb{S}}_m^{(b)}))$$

- Dans le cas de la régression, les régions étant disjointes, le pb. peut se résoudre localement de façon équivalente,  $\forall m = 1, \dots, \hat{s}^{(b)}$  :

$$\hat{\gamma}_m^{(b)} = \arg \min_{\gamma \in \mathbb{R}} \sum_{i=1}^n \ell(y_i, \hat{f}^{(b-1)}(\mathbf{x}_i) + \gamma \text{Ind}(\mathbf{x}_i \in \hat{\mathbb{S}}_m^{(b)}))$$

6. Ceci n'était pas le cas pour la tâche de catégorisation où les feuilles des différents arbres étaient associées à plusieurs classes.

## GB : AD comme weak learner (suite)

- Avec l'expression précédente, nous pouvons interpréter ce modèle comme un FSAM qui ajoute à chaque itération  $b$ , les éléments de la base de fonctions  $\{\text{Ind}(\mathbf{x} \in \hat{\mathbb{S}}_m^{(b)})\}_{m=1}^{\hat{s}^{(b)}}$ . Les éléments de la base sont des fcts inférées par l'AD afin de mieux prédire les pseudo-erreurs et changent donc à chaque itération  $b$ .
- D'ailleurs au lieu d'attribuer le coefficient  $\hat{\beta}^{(b)} \hat{d}_m^{(b)}$  pour la fct.  $\text{Ind}(\mathbf{x} \in \hat{\mathbb{S}}_m^{(b)})$ , on peut trouver le coefficient optimal en résolvant :

$$(\hat{\gamma}_m^{(b)})_{m=1, \dots, \hat{s}^{(b)}} = \arg \min_{\gamma \in \mathbb{R}^{\hat{s}^{(b)}}} \sum_{i=1}^n \ell(y_i, \hat{f}^{(b-1)}(\mathbf{x}_i) + \underbrace{\sum_{m=1}^{\hat{s}^{(b)}} \gamma_m \text{Ind}(\mathbf{x}_i \in \hat{\mathbb{S}}_m^{(b)})}_{\hat{g}^{(b)}(\mathbf{x}; \hat{\mathbf{c}}^{(b)})})$$

## GB : régression avec perte $\ell_1$

- Nous appliquons le modèle de GB utilisant les AD avec la perte  $\ell_1$ . Etant donné un couple  $(\mathbf{x}, y)$ , nous avons :

$$\ell_1(f, \mathbf{x}, y) = |y - f(\mathbf{x})|$$

- Dans ce cas, les pseudo-erreurs valent,  $\forall i = 1, \dots, n$  :

$$\begin{aligned} \tilde{y}_i &= -\frac{\partial \ell_1(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)} \Big|_{f(\mathbf{x})=f^{(b-1)}(\mathbf{x})} \\ &= \text{Sign}(y_i - f^{(b-1)}(\mathbf{x}_i)) \end{aligned}$$

- Ceci implique que  $g(\mathbf{x}; \mathbf{c}^{(b)})$  ligne 5 dans l'Algo. général de GB donné en slide 343, est ajusté par MCO aux signes des résidus courants  $\{\tilde{y}_i\}$ .

## GB : régression avec perte $\ell_1$ (suite)

- Si nous utilisons des AD, cela revient à estimer les coefficients dans une base de fct. comme vu en slide 346. Nous avons argumenté slide 347 que ce pb. pouvait se traiter au niveau local pour la tâche de régression. Nous avons dans le cas de  $\ell_1$ ,  $\forall m = 1, \dots, \hat{s}^{(b)}$  :

$$\hat{\gamma}_m^{(b)} = \text{Median} \left( \left\{ y_i - \hat{f}^{(b-1)}(\mathbf{x}_i) \right\}_{\mathbf{x}_i \in \hat{\mathbb{S}}_m^{(b)}} \right)$$

- Ainsi, à chaque itération, un arbre de rég. est inféré par MCO pour prédire le signe des pseudo-erreurs courants  $\{\text{Sign}(y_i - \hat{f}^{(b-1)}(\mathbf{x}_i))\}$ .
- Ensuite, l'approximation est mise à jour en ajoutant la médiane des pseudo-erreurs dans chacun des noeuds de l'AD.
- Ceci est résumé dans le pseudo-code suivant.

## GB : Régularisation

- Les méthodes de GB sont sujettes au pb. de sur-apprentissage et il est nécessaire en pratique de régulariser afin d'obtenir de bonnes performances en généralisation.
- Le GB étant fondé sur le principe *additive model*, une méthode simple pour régulariser consiste à contrôler l'hyperparamètre  $t$ , c.à.d. le nb. de *weak learner* dans l'ensemble :
  - $t$  trop petit conduit à du sous-apprentissage,
  - $t$  trop grand conduit à du sur-apprentissage,
  - La valeur appropriée pour  $t$  peut être inférée par validation croisée à l'aide d'un ensemble de validation.
- Le principe fondamental ici repose sur l'hypothèse de "sparsité" vis à vis du nb. d'éléments dans l'ensemble : peu de "votants" donnerait de meilleures prédictions.

## GB : pseudo-code régression avec perte $\ell_1$ et prédition

**Input** :  $\mathbb{E}, t$  (nb. d'itérations),  $g$  (un AD)

```

1    $\hat{f}^0(\mathbf{x}) = \text{Median}(\{y_i\})$ 
2   Pour tout  $b = 1, \dots, t$  faire
3       Calculer  $\tilde{y}_i = \text{Sign}(y_i - f^{(b-1)}(\mathbf{x}_i))$ ,  $\forall i = 1, \dots, n$ 
4       Résoudre  $\{\hat{\mathbb{S}}_m^{(b)}\}_{m=1, \dots, \hat{s}^{(b)}} = \arg \min_{\mathbf{c}, \gamma} \sum_{i=1}^n (\tilde{y}_i - \gamma g^{(b)}(\mathbf{x}_i; \mathbf{c}))^2$ 
5       Calculer  $\hat{\gamma}_m^{(b)} = \text{Median} \left( \left\{ y_i - \hat{f}^{(b-1)}(\mathbf{x}_i) \right\}_{\mathbf{x}_i \in \hat{\mathbb{S}}_m^{(b)}} \right)$ 
6       Mettre à jour  $\hat{f}^{(b)}(\mathbf{x}) = \hat{f}^{(b-1)}(\mathbf{x}) + \sum_{m=1}^{\hat{s}^{(b)}} \hat{\gamma}_m^{(b)} \text{Ind}(\mathbf{x} \in \hat{\mathbb{S}}_m^{(b)})$ 
7   Fin Pour
8   Output :  $\{\hat{\mathbb{S}}_m^{(b)}, \hat{\gamma}_m^{(b)}\}_{b=1, \dots, t; m=1, \dots, \hat{s}^{(b)}}$ 
```

- La prédition du modèle *Least Absolute Deviation* est donnée par :

$$\hat{f}_{gb-lad}(\mathbf{x}) = \sum_{b=1}^t \sum_{m=1}^{\hat{s}^{(b)}} \hat{\gamma}_m^{(b)} \text{Ind}(\mathbf{x} \in \hat{\mathbb{S}}_m^{(b)})$$

## GB : Régularisation (suite)

- Une autre approche de régularisation consiste en des stratégies de rétrécissement (*shrinkage*) similaire, dans l'esprit, aux méthodes ridge et lasso vues précédemment.
- Toutefois dans le contexte du GB, une stratégie simple de shrinkage vise à intégrer un hyperparamètre  $\nu \in ]0, 1]$  appelé taux d'apprentissage (*learning rate*) dans les formules de mise à jour du prédicteur ensembliste à chaque itération  $b$ . En reprenant le pseudo-code général du GB donné en slide 343, la ligne 7 devient :

$$\hat{f}^{(b)}(\mathbf{x}) = \hat{f}^{(b-1)}(\mathbf{x}) + \nu \hat{\beta}^{(b)} \hat{g}^{(b)}(\mathbf{x}; \hat{c}^{(b)})$$

- Les deux stratégies de régularisation peuvent être combinées mais les deux hyperparamètres ne sont pas indépendants [Friedman, 2001] :
  - Des petites valeurs de  $\nu (< 0.125)$  donnent de meilleurs résultats que  $\nu = 1$ .
  - Des petites valeurs de  $\nu$  donnent des plus grandes valeurs de  $t$ .

## \*\*Code\*\* Gradient Boosting, AirBnB

- Ex. de code Python pour le *Gradient Boosting* pour la tâche de régression avec tuning de plus. hyperparamètres par *grid search*.

```

1 from sklearn.ensemble import GradientBoostingRegressor
2
3 # Gradient Boosting Regressor with L1 loss function
4 gbr = GradientBoostingRegressor(loss='absolute_error') # Set the loss to 'absolute_
5 # error' for L1 loss
6 gbr_params = {'n_estimators': [50, 100], 'learning_rate': [0.01, 0.1, 0.2], 'max_depth':
7 ': [3, 6, 10]} # gbr_params = {'n_estimators': [50, 100, 200], 'learning_rate':
8 '[0.01, 0.1, 0.2], 'max_depth': [3, 6, 10]} # Parameters to test
9
10 # Use GridSearchCV to find the best combination of parameters using 5-fold cross-
11 # validation
12 gbr_cv = GridSearchCV(gbr, gbr_params, cv=5, scoring='neg_mean_absolute_error', n_jobs
13 =4) # Use MAE as scoring metric
14 gbr_cv.fit(X_train, y_train)
15
16 # Print the best parameters and the MAE on the test set
17 print('Best Gradient Boosting params:', gbr_cv.best_params_)
18 y_pred_gbr = gbr_cv.predict(X_test)
19 print('Gradient Boosting RMSE:', np.sqrt(mean_squared_error(y_test, y_pred_gbr)))

```

1 Best Gradient Boosting params: {'learning\_rate': 0.1, 'max\_depth': 6, 'n\_estimators': 100}  
2 Gradient Boosting RMSE: 0.421889873469355

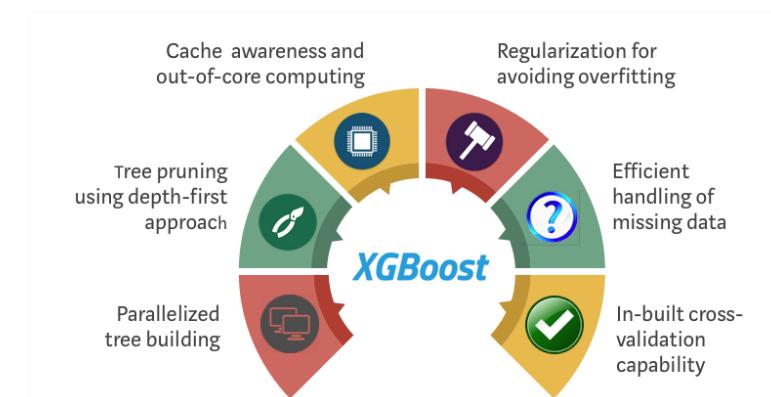
## XGBoost (suite)

- La scalabilité de XGBoost est également améliorée grâce à :
  - L'utilisation d'une heuristique pour la recherche de la variable de *split* qui représente une étape complexe en temps de traitement de l'approche (cf section 3.2 de [Chen and Guestrin, 2016]).
  - L'utilisation d'une heuristique pour générer rapidement des points de *split* candidats. Cette heuristique repose sur des résultats théoriques garantissant son efficacité (cf section 3.3 de [Chen and Guestrin, 2016]).
  - La gestion spécifique des variables contenant bcp de valeurs nulles (données manquantes, variables avec bcp de valeurs nulles, variables discrètes encodées par des variables binaires -dummy ou one-hot encoding-) (cf section 3.4 de [Chen and Guestrin, 2016]).
- XGBoost est réputée car ses performances sont souvent parmi les meilleures sur la plateforme kaggle.

## XGBoost

- XGBoost est une librairie qui implémente le GB avec des arbres de décision de façon distribuée et optimisée. Il s'agit d'un module qui “*scales beyond billions of examples using far fewer resources than existing systems*” et dont le but est d'être “*highly efficient, flexible and portable*” [Chen and Guestrin, 2016].
- Du point de vue méthodologique, XGBoost minimise un critère avec un terme de pénalité qui “*will tend to select a model employing simple and predictive functions*” [Chen and Guestrin, 2016]. Le terme de régularisation pénalise le nb. de feuilles de chaque arbre et ajoute une norme  $\ell_2$  sur les valeurs cibles associées à chaque feuille.
- Au-delà du critère d'optimisation, XGBoost intègre d'autres techniques permettant de limiter le sur-apprentissage comme le random subspace pratiqué dans le RF.

## XGBoost (suite)



## \*\*Code\*\* XGBoost, Two moons

- Ex. de code Python pour XGBoost pour la tâche de catégorisation avec tuning de plus. hyperparamètres par *grid search*.

```

1 from xgboost import XGBClassifier # XGBoost classifier
2 # Define the model and fit on training data
3 xgb = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss') # Disable label
4     encoder warning
5 xgb_params = {'n_estimators': [50, 100, 200], 'learning_rate': [0.01, 0.1, 0.2], 'max_
6     depth': [3, 6, 10]} # Parameters to test
7 # Use GridSearchCV to find the best parameters using 5-fold cross-validation
8 xgb_cv = GridSearchCV(xgb, xgb_params, cv=5, scoring='accuracy', n_jobs=4)
9 xgb_cv.fit(X_train, y_train)
10 score = xgb_cv.score(X_test, y_test)
11 # Instanciate an empty figure and plot the decision contours
12 figure = plt.figure(figsize=(10, 10))
13 ax = plt.subplot(1, 1, 1)
14 DecisionBoundaryDisplay.from_estimator(xgb_cv, X, cmap=cm, alpha=0.8, ax=ax, eps=0.5)
15 # Plot the training points then the test points
16 ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright, edgecolors="k", s
17 =125)
18 ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, edgecolors="k", alpha
19 =0.6, s=75)
20 ax.set_xlim(x_min, x_max)
21 ax.set_ylim(y_min, y_max)
22 ax.set_title("XGBoost classification")
23 ax.text(x_max - 0.3, y_min + 0.3, ("%.2f" % score).lstrip("0"), size=35,
24 horizontalalignment="right")

```

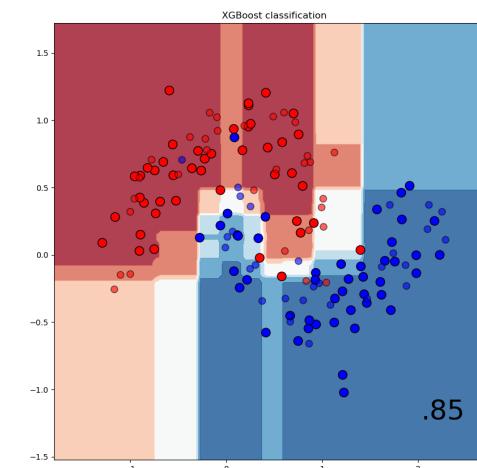
J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

357 / 534

## \*\*Code\*\* XGBoost, Two moons (suite)

- Lignes de niveau des scores du XGBoost appris.



J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

358 / 534

## \*\*Code\*\* XGBoost, AirBnB

- Ex. de code Python pour XGBoost pour la tâche de régression avec tuning de plus. hyperparamètres par *grid search*.

```

1 from xgboost import XGBRegressor # XGBoost model
2 # XGBoost Regressor with hyperparameter tuning (n_estimators, learning_rate, max_depth
3 )
4 xgb = XGBRegressor()
5 xgb_params = {'n_estimators': [50, 100], 'learning_rate': [0.01, 0.1, 0.2], 'max_depth
6     ': [3, 6, 10]} # Parameters to test
7 # Use GridSearchCV to find the best combination of parameters using 5-fold cross-
8 validation
9 xgb_cv = GridSearchCV(xgb, xgb_params, cv=5, scoring='neg_mean_squared_error', n_jobs
10 =4)
11 xgb_cv.fit(X_train, y_train)
12 # Print the best parameters and the RMSE on the test set
13 print('Best XGBoost params:', xgb_cv.best_params_)
14 y_pred_xgb = xgb_cv.predict(X_test)
15 print('XGBoost RMSE:', np.sqrt(mean_squared_error(y_test, y_pred_xgb)))

```

```

1 Best XGBoost params: {'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 50}
2 XGBoost RMSE: 0.4195629472606508

```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

359 / 534

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

360 / 534

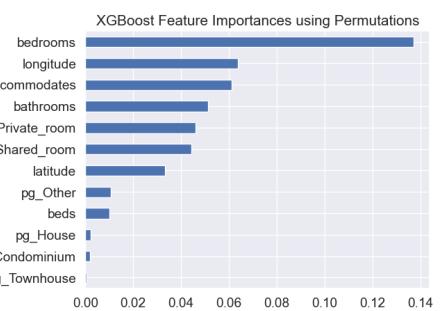
## \*\*Code\*\* XGBoost, Var. Imp. Permutations, AirBnB

- Ex. de code Python permettant de calculer et de visualiser l'importance des variables selon la méthode basée sur les permutations.

```

1 from sklearn.inspection import permutation_
2     importance
3 # Assuming xgb_cv is a fitted
4     RandomForestRegressor and X_test, y_
5     test are test data
6 feature_perm_importances = permutation_
7     importance(xgb_cv, X_test, y_test,
8     scoring='neg_mean_squared_error', n_
9     repeats=30, random_state=42)
10 feature_names = variable_names
11 perm_importance = pd.Series(feature_perm_
12     importances.importances.mean(axis=1),
13     index=feature_names).sort_values(
14     ascending=True)
15 ax = perm_importance.plot.bart()
16 ax.set_title("XGBoost Feature Importances
17     using Permutations")

```



## Rappel du Sommaire

### 5 Méthodes d'ensemble en ML et extensions spatiales

- Arbres de décision
  - Cas du problème de catégorisation
  - Cas du problème de régression
  - Compléments sur les arbres de décision
- Bagging
- Forêts aléatoires (RF)
- Gradient boosting (GB)
- Moran Eigenvector Maps (MEM)
- Geographical Random Forest (GRF)

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

361 / 534

## Backgrd : Décomposition spectrale d'une matrice

- Soit  $\mathbf{K} = (k_{ii'}) \in \mathbb{R}^{n \times n}$  une matrice vérifiant les propriétés suivantes :
  - ▶  $\mathbf{K}$  est symétrique.
  - ▶  $\mathbf{K}$  est semi-définie positive (s.d.p.) :

$$\mathbf{x}^\top \mathbf{K} \mathbf{x} \geq 0, \forall \mathbf{x} \in \mathbb{R}^n$$

- Une telle matrice  $\mathbf{K}$  est dite matrice de Gram. Elle encode implicitement des produits scalaires entre  $n$  vecteurs.
- La décomposition spectrale de  $\mathbf{K}$  est t.q. :
  - ▶ ses valeurs propres (supposées discrètes) sont positives ou nulles :

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$$

- ▶ ses vecteurs propres resp. (supposés normés) sont mutuellement orthogonaux :

$$\langle \mathbf{u}_m, \mathbf{u}_{m'} \rangle = 0, \forall m, m' = 1, \dots, n; m \neq m'$$

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

363 / 534

## Motivations

- L'approche que nous allons voir a été introduite dans [Dray et al., 2006] en écologie où la dépendance spatiale peut être causée par des facteurs environnementaux (agissant comme variables confondantes, ...) ou des processus locaux aux entités étudiées (par ex : dynamique des populations, ...). Ces sources de variation rendent l'**analyse complexe**.
- Il est nécessaire de modéliser explicitement les relations spatiales pour éviter des biais dus à une mauvaise prise en compte des corrélations spatiales ce qui peut fausser l'estimation de paramètres de modèles et conduire à de mauvaises conclusions.
- Une approche possible est l'**analyse de surface** en utilisant par exemple, des **fonctions de base polynomiales des variables géographiques comme variables descriptives** mais celle-ci est performante lorsque les localisations sont réparties de façon homogène ce qui n'est pas toujours le cas en pratique.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

362 / 534

## Backgrd : Décomposition spectrale d'une matrice (suite)

- La matrice  $\mathbf{U} = (\mathbf{u}_1 \ \dots \ \mathbf{u}_n) \in \mathbb{R}^{n \times n}$  formée des vecteurs propres (en colonne) est orthogonale :

$$\mathbf{U}^\top \mathbf{U} = \mathbf{I}_n$$

- Notons par  $\Lambda$  la matrice diagonale des valeurs propres :

$$\Lambda = \text{Diag}(\lambda_1, \dots, \lambda_n)$$

- Nous avons alors la propriété suivante (factorisation) :

$$\mathbf{K} = \mathbf{U} \Lambda \mathbf{U}^\top$$

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

364 / 534

## Backgrd : Décomposition spectrale d'une matrice (suite)

- Considérons les matrices partielles suivantes :

- ▶  $\mathbf{U}_k = (\mathbf{u}_1 \ \dots \ \mathbf{u}_k) \in \mathbb{R}^{n \times k}$ .
- ▶  $\mathbf{\Lambda}_k = \text{Diag}(\lambda_1, \dots, \lambda_k, \underbrace{0, \dots, 0}_{(n-k) \text{ fois}})$ .

- Posons alors :

$$\mathbf{K}_k = \mathbf{U}_k \mathbf{\Lambda}_k \mathbf{U}_k^\top = \sum_{m=1}^k \lambda_m \mathbf{u}_m \mathbf{u}_m^\top$$

- Nous avons la propriété suivante (**théorème d'Eckart-Young**) :

$$\mathbf{K}_k = \arg \min_{\mathbf{X} \in \mathbb{R}^{n \times n}} \|\mathbf{K} - \mathbf{X}\|_F^2, \text{ s.l.c. } \text{Rg}(\mathbf{X}) \leq k$$

où  $\|\mathbf{K} - \mathbf{X}\|_F^2 = \sum_{i,i'=1}^n (k_{ii'} - x_{ii'})^2$  est la distance de Frobenius.

- $\mathbf{K}_k$  est la meilleure approximation de rang  $k$  de la matrice  $\mathbf{K}$ .

## Backgrd : Décomposition spectrale d'une matrice (suite)

- Le théorème d'Eckart-Young (et la décomposition en valeurs singulières) est central en analyse de données. Il fonde les techniques telle que l'Analyse en Composantes Principales.
- Si on raisonne avec une matrice de distances Euclidiennes le résultat précédent montre que l'utilisation de la matrice de centrage permet de se ramener à des produits scalaires. Cette démarche est connue sous le nom de *Multidimensional Scaling* (MDS) de Torgerson [Torgerson, 1958] ou de *Principal Coordinates Analysis* (PCoA)
- Plus généralement les techniques de **Multidimensional Scaling** ont pour objectif de représenter dans un espace Euclidien (*embedding*) de faible dimension (objectif de visualisation), des objets pour lesquels nous disposons uniquement que d'une matrice de comparaisons par paires de dissimilarités (pas forcément des distances Euclidiennes).

## Backgrd : Décomposition spectrale d'une matrice (suite)

- Soit  $\mathbf{D} = (d_{ii'}) \in \mathbb{R}^{n \times n}$  une mat. de distances Euclidiennes au carré :

$$d_{ii'} = \|\mathbf{x}_i - \mathbf{x}_{i'}\|^2$$

- Soit la **matrice dite de centrage** d'ordre  $n$  définie par :

$$\mathbf{I}_n - \frac{\mathbf{1}_n \mathbf{1}_n^\top}{n}$$

- On a alors :

$$\left( \mathbf{I}_n - \frac{\mathbf{1}_n \mathbf{1}_n^\top}{n} \right) \mathbf{D} \left( \mathbf{I}_n - \frac{\mathbf{1}_n \mathbf{1}_n^\top}{n} \right) = \mathbf{K}$$

où le terme général  $k_{ii'}$  est t.q. :

$$\begin{aligned} k_{ii'} &= d_{ii'} - \frac{1}{n} \sum_{i'} d_{ii'} - \frac{1}{n} \sum_i d_{ii'} + \frac{1}{n^2} \sum_{i,i'} d_{ii'} \\ &= \langle \mathbf{x}_i - \bar{\mathbf{x}}, \mathbf{x}_{i'} - \bar{\mathbf{x}} \rangle \end{aligned}$$

- $\mathbf{K}$  est la mat. des produits scalaires canoniques entre vecteurs centrés.

## Le modèle PCNM

- **PCNM** *Principal Coordinates of Neighbour Matrices* est une approche introduite dans [Borcard and Legendre, 2002].
- PCNM est *data-driven* et **emploie les vect. prop. d'une mat. de dist. tronquées** comme variables explicatives supplémentaires.
- Le pseudocode de PCNM est le suivant :

1. Calcul d'une mat. de dist. Euclidiennes (géographiques)  $\mathbf{D} = (d_{ii'})_{i,i'=1,\dots,n}$ .
2. Choix d'un seuil  $h > 0$ , calcul de la mat. des dist. au carré tronquée (*squared and sparsified*)  $\mathbf{D}^{ss} = (d_{ii'}^{ss})$  avec :

$$d_{ii'}^{ss} = \begin{cases} d_{ii'}^2 & \text{si } d_{ii'} \leq h \\ (4\theta)^2 & \text{si } d_{ii'} > h \end{cases}$$

3. Calcul de la mat. des dist. tronquée au carré centrée  $\Delta_{\mathbf{D}^{ss}}$  de terme général :

$$[\Delta_{\mathbf{D}^{ss}}]_{ii'} = -\frac{1}{2} \left( d_{ii'}^{ss} - \frac{1}{n} \sum_{i'} d_{ii'}^{ss} - \frac{1}{n} \sum_i d_{ii'}^{ss} + \frac{1}{n^2} \sum_{i,i'} d_{ii'}^{ss} \right)$$

## Le modèle PCNM (suite)

- Le pseudocode de PCNM est le suivant (suite) :

4. Décomposition spectrale de  $\Delta_{Dss} \in \mathbb{R}^{n \times n}$  :

$$\text{Sp}(\Delta_{Dss}) = \{\lambda_1, \dots, \lambda_n\}$$

avec  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$  les valeurs propres ordonnées de  $\Delta_{Dss}$ .

5. Sélection des vecteurs propres de valeurs propres positives et normalisation. Soit  $u_m \in \mathbb{R}^n$  le vecteur propre associé à  $\lambda_m$ . On prend comme var. descriptives supplémentaires, les vecteurs suivants :

$$\frac{1}{\sqrt{\lambda_m}} u_m, \forall m : \lambda_m > 0$$

### Remarques :

- $\Delta_{Dss}$  étant issue de la mat. des dist. Euclidiennes au carré tronquée, elle n'est pas semi-définie positive et possède des valeurs propres négatives.
- Garder les vecteurs propres de valeurs propres strictement positives permet de plonger (*embedding*) les obs. dans un espace Euclidien.

## Distances, similarités et matrices SW (suite)

- On voit ensuite que  $\Delta_{D^s}$  a les mêmes vecteurs propres que  $S$  centré :

$$\left( I_n - \frac{1_n 1_n^\top}{n} \right) S \left( I_n - \frac{1_n 1_n^\top}{n} \right)$$

Les val. propres sont identiques au fact. multiplicatif  $\frac{\max((d_{ii'}^2)_{i,i'})}{2}$  près.

- Ce point de vue nous permet d'introduire naturellement les matrices SW  $W$  qui sont basées sur des mesures de proximité et non pas de distance. Elles sont donc de même nature que  $S$ .
- PCNM peut donc être interprété telle la décomposition spectrale d'une matrice de SW  $W = (w_{ii'})_{i,i'}$  avec :

$$w_{ii'} = s_{ii'} = 1 - \frac{d_{ii'}^2}{\max((d_{ii'}^2)_{i,i'})} \in [0, 1]$$

- On peut généraliser PCNM avec des mesures d'association spatiale.

## Distances, similarités et matrices SW

- L'opération de centrage utilisée peut être formalisée matriciellement (cf slide 366). Appliquons la à  $D^s = (d_{ii'}^2)_{i,i'=1,\dots,n}$  :

$$\Delta_{D^s} = -\frac{1}{2} \left( I_n - \frac{1_n 1_n^\top}{n} \right) D^s \left( I_n - \frac{1_n 1_n^\top}{n} \right).$$

- En PCNM, on peut raisonner de façon équivalente avec des similarités. Soit la matrice  $S = (s_{ii'})_{i,i'} \in \mathbb{R}^{n \times n}$  de terme général :

$$s_{ii'} = 1 - \frac{d_{ii'}^2}{\max((d_{ii'}^2)_{i,i'})}$$

- On montre que  $\Delta_{D^s}$  défini ci-dessus peut s'obtenir à partir de  $S$  :

$$\Delta_{D^s} = \frac{\max((d_{ii'}^2)_{i,i'})}{2} \left( I_n - \frac{1_n 1_n^\top}{n} \right) S \left( I_n - \frac{1_n 1_n^\top}{n} \right)$$

## Moran's Eigenvector Maps (MEM)

- On suppose une var. spatiale  $X$  et un échantillon de  $n$  localisations.  $x \in \mathbb{R}^n$  est le vect. des réalisations de  $X$  en les  $n$  positions géo.
- On suppose que les relations spatiales entre ces  $n$  localisations sont données par une matrice de SW  $W = (w_{ii'})_{i,i'=1,\dots,n}$  :
- Rappelons que l'indice de Moran est donné par (cf slide 139) :

$$I(x, W) = \frac{n}{\sum_{i,i'=1}^n w_{ii'}} \frac{\sum_{i,i'=1}^n w_{ii'}(x_i - \bar{x})(x_{i'} - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

- On peut exprimer l'**indice de Moran par calculs matriciels** :

$$I(x, W) = \frac{n}{1_n^\top W 1_n} \frac{x^\top \left( I_n - \frac{1_n 1_n^\top}{n} \right) W \left( I_n - \frac{1_n 1_n^\top}{n} \right) x}{x^\top \left( I_n - \frac{1_n 1_n^\top}{n} \right) x}$$

## Moran's Eigenvector Maps (MEM)

- Etant donné une mat. SW  $\mathbf{W}$ , les val. prop. extrêmes de l'indice ce Moran valent  $(n/\mathbf{1}_n^\top \mathbf{W} \mathbf{1}_n)\lambda_{max}$  et  $(n/\mathbf{1}_n^\top \mathbf{W} \mathbf{1}_n)\lambda_{min}$  où  $\lambda_{max}$  et  $\lambda_{min}$  sont resp. la plus grande et la plus petite val. prop. de la matrice suivante [de Jong et al., 1984] :

$$\Omega = \left( \mathbf{I}_n - \frac{\mathbf{1}_n \mathbf{1}_n^\top}{n} \right) \mathbf{W} \left( \mathbf{I}_n - \frac{\mathbf{1}_n \mathbf{1}_n^\top}{n} \right)$$

- On infère donc les propriétés suivantes :

- ▶ Les vecteurs propres de  $\Omega$  sont des vecteurs de norme unitaire, mutuellement orthogonaux maximisant l'indice de Moran.
- ▶ Les valeurs propres peuvent être soit positives, soit négatives.
- ▶ Les vecteurs propres associés aux plus grandes valeurs propres ont une ASG très forte et décrivent des structures spatiales globales.
- ▶ Les vect. prop. associées à des val. prop. dont les valeurs absolues sont proches de 0 correspondent à des autocorrelations spatiales de faible intensité et ne sont pas propices pour décrire des structures spatiales.

## MEM et choix de la matrice de SW $\mathbf{W}$ (suite)

- Pour un modèle estimé  $\hat{f}$ , le critère AIC est défini comme suit :

$$AIC = -2 \log \left( \frac{Lvr(\hat{f})}{n} \right) + 2q$$

où  $Lvr(\hat{f})$  est la valeur (maximale) de la log-vraisemblance évaluée en son maximiseur  $\hat{f}$  et  $q$  est le nb. de paramètres du modèle estimé  $\hat{f}$ .

- Si  $n$  est petit rapport à  $q$  ( $n/q \leq 40$ ) on a une version corrigée :

$$AIC_c = AIC + \frac{2q(q+1)}{n-q-1}$$

- Plus la valeur AIC ou  $AIC_c$  est petite, meilleur est le modèle  $\hat{f}$ .

## MEM et choix de la matrice de SW $\mathbf{W}$

- MEM suggère que l'on peut employer plusieurs types de relations spatiales et il existe une multitude d'approches pour le faire (contiguïté, noyaux, hybrides, ...). Le choix de la matrice de SW  $\mathbf{W}$  est crucial car il impacte les résultats de l'analyse spatial.
- Dans le cas d'un échantillon uniforme des localisations (par ex. grille régulière), les structures définies par les vecteurs propres sont assez similaires pour différentes définitions de  $\mathbf{W}$ .
- Au contraire, pour un échantillonnage irrégulier, les valeurs et vecteurs propres varient bcp selon la définition de  $\mathbf{W}$ .
- Un expert des données pourrait recommander certaines approches plutôt que d'autres.
- Pour un pb. de régression où on prédit  $Y$  continue en fonction de  $X^1, \dots, X^p$  continues, on peut avoir une approche de sélection de modèles à l'aide du critère AIC (*Akaike Information Criterion*).

## MEM et choix de la matrice de SW $\mathbf{W}$ (suite)

- La procédure suivante pour sélectionner  $\mathbf{W}$  est proposée dans [Dray et al., 2006] :
  1. Définir un ensemble de matrices SW candidates.
  2. Pour chaque matrice de SW  $\mathbf{W}$  candidate :
    - a. Décomposition spectrale de  $\Omega$ . On rassemble les  $q$  MEM (vecteurs propres) dans une matrice  $\mathbf{U} \in \mathbb{R}^{n \times q}$ .
    - b. Sélection du meilleur modèle :
      - Tri des vect. prop. par ordre décroissant de leur val. prop.
      - Ajout des vect. prop. un après l'autre dans un nouveau modèle.
      - Calcul du critère AIC pour chacun des  $q$  modèles.
      - Sélection du modèle avec la plus petite valeur de AIC.
    3. Sélection de la matrice de SW donnant le modèle avec la plus petite valeur de AIC.

## \*\*Code\*\* : MEM, AirBnB

- Ex. de code Python pour enlever la longitude et la latitude des covariables. Puis, utilisation de la fonction `weights` de la lib. `pysal` pour déterminer une matrice de SW.
- On a pris une matrice  $\mathbf{W}$  pondérée par un noyau Gaussien dont l'hyperparamètre est adaptatif, et le nombre de voisins est 20.

```

1 # Remove Lat and Lon from X
2 Xmem = X.copy()
3 variable_names_mem = variable_names
4 Xmem = np.delete(Xmem, [10, 11], axis=1)
5 variable_names_mem.remove('latitude')
6 variable_names_mem.remove('longitude')
7
8 from pysal.lib import weights
9 # Extract the longitude and latitude from geopandas df db
10 coords = np.array(list(zip(db.geometry.x, db.geometry.y)))
11 # Construct one SW matrix W using Gaussian kernel, adaptative bandwidth with 20
# nearest neighbors
12 knn = 20
13 gaussian_weights = weights.Kernel(coords, k=knn, fixed=False, function='gaussian')
14 # Make the SW matrix W symmetric
15 W = gaussian_weights.full()[0]
16 W = (W + W.T)/2

```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

377 / 534

## \*\*Code\*\* : MEM, Implémentation

- Ex. de code Python pour la décomposition spectrale de  $\Omega$  et l'extraction des vecteurs propres formant les MEM.

```

1 # Extract eigenvectors from a double centered SW matrix
2 def select_mem(omega, k):
3     # Compute spectral decomposition
4     eigen_val, eigen_vec = np.linalg.eigh(omega)
5     # Sort indices wrt decreasing order of absolute value of eigenvalues
6     sorted_indices = np.argsort(-np.abs(eigen_val))
7     # Select non null eigenvalues
8     sorted_indices = sorted_indices[eigen_val[sorted_indices] > 1e-7]
9     if k > len(sorted_indices):
10         return eigen_vec[:, sorted_indices]
11     # Sort eigenvectors according to the sorted indices
12     return eigen_vec[:, sorted_indices[:k]]

```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

379 / 534

## \*\*Code\*\* : MEM, Implémentation

- Ex. de code Python pour le double centrage de la matrice de SW  $\mathbf{W}$ .

```

1 # Double centering operator
2 def double_centering(W):
3     # Find the matrix order
4     n = W.shape[0]
5     # Initialize a null matrix of order n
6     omega = np.zeros(shape=(n, n))
7     # Compute sums for all rows
8     row_sums = np.array(W.sum(axis=1)).reshape(-1)
9     # Compute sums for all cols (same as row_sums if W is symmetric)
10    col_sums = np.array(W.sum(axis=0)).reshape(-1)
11    # Compute sum over all cells of W
12    sums = W.sum()
13    # Do double centering without n**2 complexity: use vectors operations
14    for i in range(n):
15        omega[i, :] += - row_sums[i]/n
16    for j in range(n):
17        omega[:, j] += - col_sums[j]/n
18    omega = omega + W + sums/n**2
19    # W[i, j] - RowSums[i]/n - ColSums[j]/n + Sums/n**2
20    return omega

```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

378 / 534

## Méthodes d'ensemble en ML et extensions spatiales Moran Eigenvector Maps (MEM)

## \*\*Code\*\* : MEM, AirBnB

- Ex. de code Python pour l'enrichissement des données initiales par des variables spatiales données par les 10 premiers MEM. Puis, préparation des données pour l'évaluation des méthodes.

```

1 # Set the number of MEM features
2 k = 10
3 omega = double_centering(W)
4 mem = select_mem(omega, k)
5 mem = np.array(mem)
6
7 # Add the MEM features to data matrix
8 Xmem = np.column_stack((Xmem, mem))
9 variable_names_mem.extend([f'mem_{i}' for i in range(1, mem.shape[1]+1)])
10
11 # Split the data into training and test sets
12 X_train, X_test, y_train, y_test = train_test_split(Xmem, y, test_size=0.2, random_state=42)

```

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

380 / 534

## \*\*Code\*\* : MEM et toutes les méthodes, AirBnB

- Nous avons testé l'ens. des méthodes de ML précédentes avec la représentation étendue par les 10 premiers MEM. Voici les résultats :

```

1 Linear Regression RMSE: 0.4561093609635568
2 Best Decision Tree params: {'max_depth': 6, 'min_samples_split': 10}
3 Decision Tree RMSE: 0.46848490288239025
4 Best Bagging params: {'max_features': 1.0, 'max_samples': 0.5, 'n_estimators': 100}
5 Bagging RMSE: 0.4444913068182282
6 Best Random Forest params: {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 50}
7 Random Forest RMSE: 0.44862448827184453
8 Best Gradient Boosting params: {'learning_rate': 0.2, 'max_depth': 6, 'n_estimators': 50}
9 Gradient Boosting RMSE: 0.4451483432222317
10 Best XGBoost params: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100}
11 XGBoost RMSE: 0.43626673203395505

```

- Les performances sont moins bonnes qu'avec les variables spatiales longitude et latitude. Ces dernières semblent mieux adaptées au problème AirBnB que les MEM.

## Rappel du Sommaire

### 5 Méthodes d'ensemble en ML et extensions spatiales

- Arbres de décision
  - Cas du problème de catégorisation
  - Cas du problème de régression
  - Compléments sur les arbres de décision
- Bagging
- Forêts aléatoires (RF)
- Gradient boosting (GB)
- Moran Eigenvector Maps (MEM)
- Geographical Random Forest (GRF)

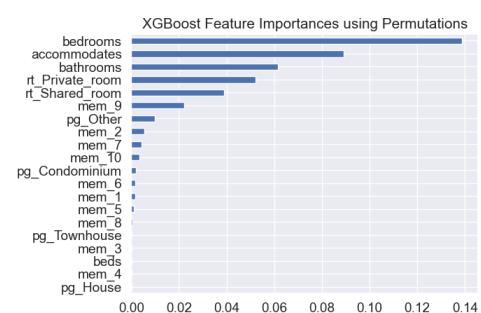
## \*\*Code\*\* : MEM, XGBoost + MEM, Var. Imp. Permutations, AirBnB

- Nous avons malgré tout évalué l'importance des variables selon la méthode basée sur les permutations pour le meilleur modèle XGBoost.

```

1 from sklearn.inspection import permutation_
2   _importance
3 # Assuming xgb_cv is a fitted
4 # RandomForestRegressor and X_test, y_
5 # test are test data
6 feature_perm_importances = permutation_
7   _importance(xgb_cv, X_test, y_test,
8   scoring='neg_mean_squared_error', n_
9   repeats=30, random_state=42)
10 feature_names = variable_names_mem
11 perm_importance = pd.Series(feature_perm_
12   _importances.importances.mean(axis=1),
13   index=feature_names).sort_values(
14   ascending=True)
15 ax = perm_importance.plot.barh()
16 ax.set_title("XGBoost Feature Importances
17   using Permutations")

```



- Quelques MEM (mem\_9, mem\_2) apportent des informations spatiales pertinentes pour la résolution de la tâche.

## Motivations

- L'approche que nous présentons ci-dessous est introduite dans [Georganos et al., 2021]. Les auteurs mettent en lumière l'utilisation de plus en plus forte des méthodes de ML en géographie. Ils exposent plus en particulier des forêts aléatoires (RF) et la tâche d'estimation de densité de population à partir d'images satellite.
- Toutefois, l'utilisation classique des RF sur des données géoréférencées ne tient pas compte de l'hétérogénéité spatiale des phénomènes à l'étude. Les auteurs proposent alors une extension substantielle des RF pour pallier ce manque et introduisent le modèle **Geographical Random Forest (GRF)**.
- L'idée centrale de l'approche reprend celle mise en avant par le modèle GWR vu précédemment : **GRF propose d'estimer un RF en chaque localisation du domaine** en exploitant prioritairement les données d'entraînement des sites géographiques voisins.
- Le modèle GRF propose toutefois de **mélanger un modèle local et un modèle global pour la prédiction**.

## Modèle Geographical Random Forest (GRF)

- Soit un ens. d'objets géo.  $\{X_i\}_{i=1,\dots,n}$ . Soit  $K$  un noyau qui sélectionne pour chq.  $X_i$  un sous-ens. de voisins selon une dist. géo.  $D$ . Cela infère une matrice de SW binaire,  $\mathbf{W} = (w_{ii'})$  avec  $w_{ii'} = K(D(X_i, X_{i'}))$ .
- On peut distinguer les noyaux  $K$  dits adaptatif ou fixe :
  - Le noyau adaptatif : sélectionne les  $k$  plus proches voisins de l'objet géographique  $X_i$ .
  - Le noyau fixe : sélectionne les sites dont la distance géo. avec  $X_i$  est inférieures à un seuil  $h$  fixé.
- Dans [Georganos et al., 2021], un noyau adaptatif est employé : cette approche est meilleure lorsque l'échantillonage est irrégulier.
- Soit  $\mathbb{V}_k(X_i)$  l'ens. des  $k$  objets géographiques  $X_{i'}$  les plus proches de  $X_i$  au sein l'échantillon d'entraînement et selon une dist. géographique.
- Le modèle local de GRF pour  $X_i$ , noté  $f_{grf}^i$ , est construit uniquement à partir des données d'entraînement dont les objets sont dans  $\mathbb{V}_k(X_i)$ .
- Le modèle gloable de GRF, noté  $f_{grf}^g$ , est construit à partir de toutes les données d'entraînement  $\mathbb{E}$ .

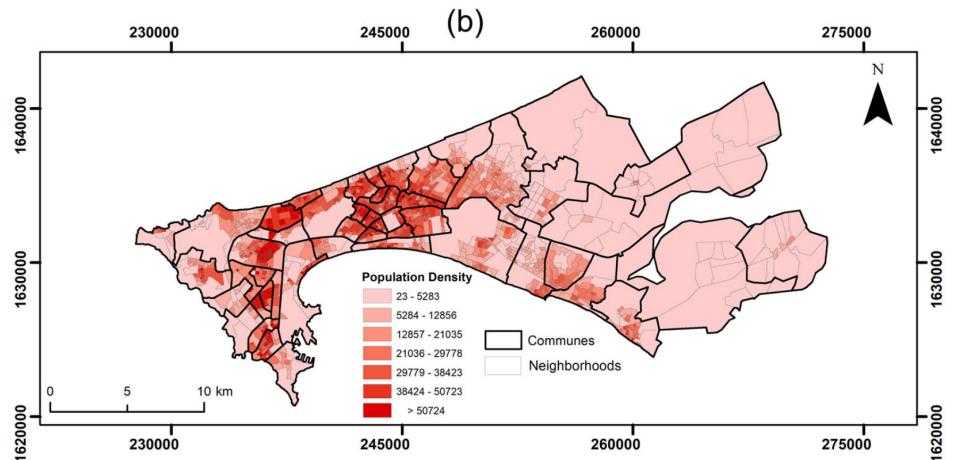
J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

385 / 534

## Illustration, densité de population, Sénégal

- Variable cible : densité de population.



Extrait de [Georganos et al., 2021]

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

387 / 534

## Modèle Geographical Random Forest (GRF)

- La prédiction de GRF pour un élément  $X_i$  dont le vecteur  $\mathbf{x}_i \in \mathbb{R}^p$  est dans  $\mathbb{E}$  (*In Sample*), est donnée par :

$$f_{grf}(\mathbf{x}_i) = \alpha f_{grf}^i(\mathbf{x}_i) + (1 - \alpha) f_{grf}^g(\mathbf{x}_i)$$

- La prédiction pour un objet  $X$  quelconque du domaine (*Out Of Sample*) représenté par  $\mathbf{x} \in \mathbb{R}^p$  est donnée par :

$$f_{grf}(\mathbf{x}) = \alpha f_{grf}^{i*}(\mathbf{x}) + (1 - \alpha) f_{grf}^g(\mathbf{x})$$

où  $X_{i*}$  est l'objet le plus proche de  $X$  géographiquement selon  $D$ .

- La fusion d'un modèle local avec le modèle global permet en théorie un meilleur contrôle de l'arbitrage biais-variance : le modèle local est à faible biais tandis que le modèle globale est à faible variance.

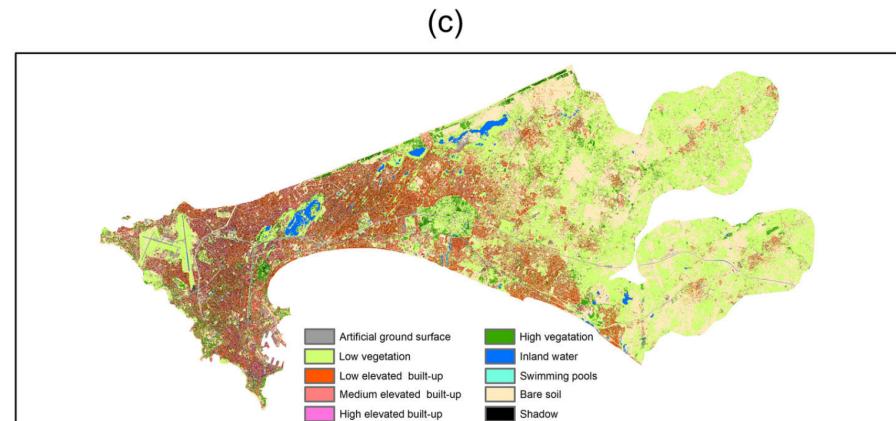
J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

386 / 534

## Illustration, densité de population, Sénégal (suite)

- Variables explicatives : indicateurs de *Land Cover and Land Use* extraits d'images satellites.



Extrait de [Georganos et al., 2021]

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

388 / 534

## Illustration, densité de population, Sénégal (suite)

- 4 modèles testés utilisant différents sous-ens. de var. explicatives.

**Table 1.** Independent variables and models used in the analysis.

Variable	LC_XY	LC	3BU_XY	3BU
Low elevated built-up (<5 m)	✓	✓	✓	✓
Medium elevated built-up (5–10 m)	✓	✓	✓	✓
High elevated built-up (>10 m)	✓	✓	✓	✓
Bare ground	✓	✓		
Low vegetation	✓	✓		
High vegetation	✓	✓		
Shadows	✓	✓		
Inland water	✓	✓		
XY coordinates	✓		✓	

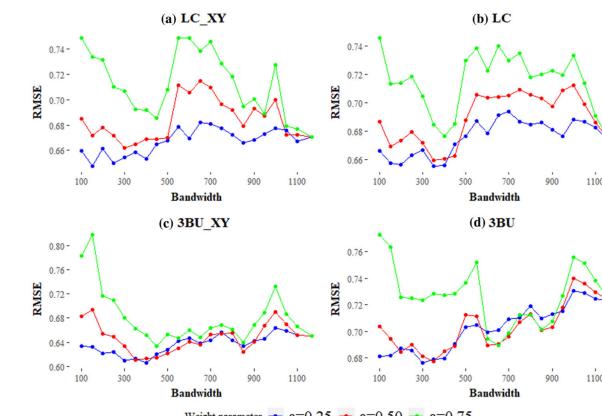
Extrait de [Georganos et al., 2021]

## Modèles *spatially explicit*

- Les auteurs de [Janelle and Goodchild, 2011, Janowicz et al., 2020] ont proposé des tests afin de vérifier si un modèle pouvait être qualifié de *spatially explicit*.
- Les 4 tests sont les suivants :
  - ▶ *Variance test* : est-ce que les prédictions du modèle varient selon la position géographique ?
  - ▶ *Representation test* : est-ce que le modèle intègre une représentation de l'information spatiale (coordonnées, localisation, ...)?
  - ▶ *Formulation test* : est-ce que le modèle utilise des concepts de l'analyse spatiale (voisinage, autocorrélation, ...)?
  - ▶ *Outcome test* : est-ce que les sorties du modèle ont une structure spatiale qui est différente de celles des données en entrée ?
- Si un modèle vérifie au moins un test alors il est dit *spatially explicit*.
- Le modèle GRF vérifie les 4 tests.
- Les modèles de régression linéaire spatiale vus précédemment sont tous *spatially explicit*. La régression linéaire multiple sans var. spatiales n'est pas *spatially explicit*.

## Illustration, densité de population, Sénégal (suite)

- RMSE pour  $\alpha \in \{0.25, 0.5, 0.75\}$  et plus. valeurs de k (en abscisse).



**Figure 4.** RMSE of GRF with increasing bandwidth (number of nearest neighbours) using proportions of (a) all LC classes and geographical coordinates as explanatory factors, (b) all LC classes, (c) 3 types of built-up and geographical coordinates and (d) 3 types of built-up as input. The last point in each graph represents the global RF model.

Extrait de [Georganos et al., 2021]

## Extensions de GRF

- Malgré les propriétés intéressantes du modèle GRF, nous avons vu sur les exemples que les performances sont très variables. Il existe en effet plusieurs limites à l'approche GRF identifiées dans [Sun et al., 2024] :
  - ▶ Le modèle est très sensible aux valeurs des hyperparamètres k (nombre de voisins) ou h (rayon du voisinage) ainsi que  $\alpha$  (poids du modèle local versus modèle global) et la détermination de ces valeurs sont critiques.
  - ▶ Les sous-modèles n'utilisent qu'une partie des données d'entraînement et ce sous-ensemble d'éléments peut, dans certains cas, être de taille réduite ce qui conduit à des modèles de prédiction peu performants.
  - ▶ Pour une donnée quelconque  $X$ , la prédiction du modèle GRF est basée sur le modèle général ainsi que sur le sous-modèle du plus proche voisin  $X_i$ . Or ce dernier peut être sous-performant conduisant alors à de mauvaises prédictions pour  $X$ .
- Les auteurs [Sun et al., 2024] proposent 3 extensions de GRF visant à prendre en considération ces défauts.

## Extension 1 de GRF : calibration des hyperparamètres

- Dans [Georganos et al., 2021], les auteurs de GRF proposent un réglage par essai-erreur pour les hyperparam. de la bande ( $k$  ou  $h$ ) et du mélange ( $\alpha$ ) mais ceci est coûteux en termes computationnels.
- Dans [Sun et al., 2024], les auteurs proposent d'**utiliser les mesures d'autocorrélation spatiale de la var.  $Y$**  afin de calibrer ces hyperparam. :
  - La largeur de la bande est déterminée en analysant l'autocorrélation spatiale à dif. échelles afin d'identifier la valeur la plus significative.
  - Le poids local est calculé en fonction de la force et de la signification statistique de l'indice de Moran I (cf slide 143) :

$$\alpha = \begin{cases} I(\mathbf{y}, \mathbf{W}) & \text{si } I(\mathbf{y}, \mathbf{W}) > 0 \text{ et } p\text{-value du test } I(\mathbf{y}, \mathbf{W}) < 0.05 \\ 0 & \text{sinon} \end{cases}$$

- L'analyse spatiale de  $Y$  est menée avant l'inférence du modèle. Elle permet de définir des valeurs adéquates de la bande et du paramètre de mélange. Par la suite, **un seul modèle GRF est estimé**.

## Extension 3 de GRF : lissage spatial pour la prédiction

- La prédiction du modèle GRF initial pour un  $X$  quelconque,  $f_{grf}(\mathbf{x}) = \alpha f_{grf}^{i*}(\mathbf{x}) + (1 - \alpha) f_{grf}^g(\mathbf{x})$ , peut être mauvaise si  $f_{grf}^{i*}(\mathbf{x})$  est sous-performant (si  $X_{i*}$  est un *outlier* par ex.).
- Dans [Sun et al., 2024], les auteurs proposent de lisser la prédiction pour  $X$  en tenant compte non pas d'un seul mais de plus sous-modèles se situant dans le voisinage de  $X$ . Cela est d'ailleurs en adéquation avec le 1er principe de Tobler.
- On suppose le même noyau  $K$  et même dist. géo.  $D$  que celui utilisé pour la sélection des données pour les sous-modèles de GRF. La nouvelle prédiction locale utilise une approche dite *spatially weighted* est notée par  $f_{grf}^{sw}$  et définie par :

$$f_{grf}^{sw}(\mathbf{x}) = \frac{\sum_{i=1}^n K(D(X, X_i)) f_{grf}^i(\mathbf{x})}{\sum_{i=1}^n K(D(X, X_i))}$$

- La prédiction devient alors  $f_{grf}(\mathbf{x}) = \alpha f_{grf}^{sw}(\mathbf{x}) + (1 - \alpha) f_{grf}^g(\mathbf{x})$ .

## Extension 2 de GRF : augmentation des données

- Le sous-modèle  $f_{grf}^i$  de  $X_i$  apprend à partir des données  $\mathbb{V}_h(X_i)$ . Si  $h$  est petit et/ou si la répartition des données est hétérogène en termes d'échelles alors  $|\mathbb{V}_h(X_i)|$  peut être trop restreint.
- Dans [Sun et al., 2024], les auteurs proposent d'**augmenter la taille de l'échantillon des sous-modèles par bootstrap** : on fait des tirages avec remise jusqu'à atteindre une taille minimale. Par ex., on peut doubler le nb. de données selon la taille initiale du sous-ens.  $|\mathbb{V}_h(X_i)|$  ou selon le nb. d'arbres du RF t.
- Si  $|\mathbb{V}_h(X_i)|$  est grand alors on n'ajoute pas de données.
- La procédure d'augmentation de données est formalisée comme suit :

$$\mathbb{V}_h^{aug}(X_i) \leftarrow \begin{cases} \text{Bootstrap}(\mathbb{V}_h(X_i), \underbrace{\min(2|\mathbb{V}_h(X_i)|, 2t)}_{\text{taille finale}}) & \text{si } |\mathbb{V}_h(X_i)| < 2t \\ \mathbb{V}_h(X_i) & \text{sinon} \end{cases}$$

## \*\*Code\*\* : GRF, AirBnB

- Pour tester GRF sur AirBnB, nous préparons les données comme suit.

```

1 # We recover the data matrix
2 db = gpd.read_file("regression_db.geojson")
3 variable_names = [
4     "accommodates", # Number of people it accommodates
5     "bathrooms", # Number of bathrooms
6     "bedrooms", # Number of bedrooms
7     "beds", # Number of beds
8     # Below are binary variables, 1 True, 0 False
9     "rt_Private_room", # Room type: private room
10    "rt_Shared_room", # Room type: shared room
11    "pg_Condominium", # Property group: condo
12    "pg_House", # Property group: house
13    "pg_Other", # Property group: other
14    "pg_Townhouse" # Property group: townhouse
15 ]
16 X = db[variable_names] # Select exogeneous var in an np.array
17 y = db["log_price"] # Put target var in a np.array
18 X[‘longitude’] = db[“geometry”].x
19 X[‘latitude’] = db[“geometry”].y
20 # Split the data into training and test sets
21 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
22 =42)
23 coords_train = X_train[['longitude', 'latitude']]
24 coords_test = X_test[['longitude', 'latitude']]
25 X_train.drop(columns=['longitude', 'latitude'])
26 X_test.drop(columns=['longitude', 'latitude'])

```

## \*\*Code\*\* : GRF, AirBnB (suite)

- Le code suivant est adapté de l'exemple fourni par les auteurs de [Sun et al., 2024] sur leur github : ici.

```

1 from PyGRF import PyGRF
2 #Create a PyGRF model by specifying hyperparameters
3 pygrf = PyGRF.PyGRFBuilder(n_estimators=100, max_features=1, band_width=20, train_
4     weighted=True, predict_weighted=True, bootstrap=False, resampled=True, random_
5     state=42)
6
7 #Fit the created PyGRF model based on training data and their spatial coordinates
8 # xy_cord is the two-dimensional coordinates of training samples
9
10 pygrf.fit(X_train, y_train, coords_train)
11
12 #Make predictions for testing data using the fitted PyGRF model and you specified
13 # local model weight
14 predict_combined, predict_global, predict_local = pygrf.predict(X_test, coords_test,
15     local_weight=0.5)
16
17 # Print RMSE on the test set
18 print('PyGRF combined RMSE:', np.sqrt(mean_squared_error(y_test, predict_combined)))
19 print('PyGRF global RMSE:', np.sqrt(mean_squared_error(y_test, predict_global)))
20 print('PyGRF local RMSE:', np.sqrt(mean_squared_error(y_test, predict_local)))

```

```

1 PyGRF combined RMSE: 0.4419222759704579
2 PyGRF global RMSE: 0.44499696404812755
3 PyGRF local RMSE: 0.4890657487532134

```

## Introduction

- Le problème de régression abordé précédemment, consiste à prédire une variable cible (ou endogène)  $Y$  à partir de plusieurs variables explicatives (ou exogènes).
- Nous abordons maintenant le problème d'**interpolation** où on s'intéresse à une variable cible que l'on dénotera par  $Z$ . Cette variable est supposée continue sur un espace 2D. En géostatistique, on emploie une modélisation probabiliste de la variable  $Z$ . La notation  $\mathcal{Z}$  sera utilisée et renvoie à la notion de fonction aléatoire.
- On dispose de  $n$  observations  $z(\mathbf{p}_i)$  en les points du plan  $\mathbf{p}_i \subset \mathbb{R}^2$  avec  $i = 1, \dots, n$ . L'objectif est alors de prédire la valeur de  $\mathcal{Z}(\mathbf{p})$  en tout point  $\mathbf{p} \subset \mathbb{R}^2$  en exploitant les données  $\{z(\mathbf{p}_i)\}_{i=1,\dots,n}$ .
- Dans ce contexte, une technique classique est celle du Krigeage (*Kriging*) du nom de Daniel Krige qui a développé dans les années 1940-50 des outils statistiques pour estimer de façon plus fiable la distribution spatiale des ressources en or dans les gisements souterrains en Afrique du Sud.

## Rappel du Sommaire

- 1 Introduction et motivations générales
- 2 Concepts de base et outils Python associés
- 3 Analyse de la dépendance spatiale
- 4 Méthodes de régression spatiale
- 5 Méthodes d'ensemble en ML et extensions spatiales
- 6 Eléments de géostatistique
- 7 Méthodes de DL et extensions spatiales

## Introduction (suite)

- Les techniques de D. Krige ont ensuite été formalisées par Georges Matheron dans les années 1960. Ces travaux ont donné naissance à la discipline **géostatistique** dont voici les principales caractéristiques :
  - ▶ **Analyse de la variabilité spatiale** : la géostat. étudie les variations spatiales de phénomènes continus en quantifiant les corrélations entre les données échantillonnées.
  - ▶ **Estimation et interpolation** : elle utilise des méthodes comme le Krigeage pour fournir des valeurs estimées en des points non échantillonnés, en cherchant à minimiser les erreurs.
  - ▶ **Optimisation et prise de décision** : elle soutient la gestion des ressources et la planification en réduisant les coûts et les risques grâce à des estimations précises.
- Ex. d'applications : industrie minière, agriculture de précision, environnement, épidémiologie, santé publique, géographie urbaine, analyses spatiales de la pauvreté et des inégalités sociales, ...
- Nous faisons au préalable des **rappels de probabilités** qui sont nécessaires avant l'introduction des éléments de géostatistiques.

## Backgrd : Variable aléatoire réelle et fct. de répartition

- Soit  $(\Omega, \Sigma, P)$  un espace probabilisé et  $\mathcal{Z}$  une application  $\Omega \rightarrow \mathbb{R}$ .
- $\mathcal{Z}$  est une **variable aléatoire réelle (v.a.r.) ou continue** si elle est une **fonction mesurable** de  $(\Omega, \Sigma, P)$ , c.à.d. si pour tout intervalle  $\mathbb{I}$  de  $\mathbb{R}$ , l'image réciproque  $\mathcal{Z}^{-1}(\mathbb{I}) = \{\omega \in \Omega : \mathcal{Z}(\omega) \in \mathbb{I}\}$  existe et correspond à un évènement de  $\Sigma$ . Dans le cas d'une v.a.r., cela implique que l'intervalle  $\mathbb{I}$  appartient à la tribu borélienne  $\mathcal{B}(\mathbb{R})$ .
- Plus formellement, on associe à  $\mathcal{Z}$  une **mesure de probabilité**  $P_{\mathcal{Z}}$  sur l'espace mesurable  $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$  définie par,  $\forall \mathbb{I} \in \mathcal{B}(\mathbb{R})$  :

$$P_{\mathcal{Z}}(\mathbb{I}) = P(\mathcal{Z}^{-1}(\mathbb{I}))$$

où  $P$  est la mesure de probabilité sur l'espace mesurable  $(\Omega, \Sigma)$ .

- Soit  $F_{\mathcal{Z}}$  la **fonction de répartition** de  $\mathcal{Z}$  qui est définie par,  $\forall z \in \mathbb{R}$  :

$$F_{\mathcal{Z}}(z) = P_{\mathcal{Z}}([-\infty, z]) = P(\mathcal{Z} \leq z) = P(\mathcal{Z}^{-1}([-\infty, z]))$$

- $F_{\mathcal{Z}}$  (donc  $P_{\mathcal{Z}}$ ) spécifie la **loi de probabilité** de la v.a.r.  $\mathcal{Z}$ .

## Backgrd : Fonction de densité de probabilité

- Si  $\mathcal{Z}$  est une v.a.r. :
  - ▶ Un point  $z \in \mathbb{R}$  n'a pas en soi de mesure de probabilité car il est vu tel un intervalle de longueur nulle. Ainsi, dans le cas continu  $P_{\mathcal{Z}}([z, z]) = 0$  pour tout  $z$  de  $\mathbb{R}$ .
  - ▶ En revanche, on définit une **fonction de densité de probabilité** notée  $f_{\mathcal{Z}}$  définie sur  $\mathbb{R}$  et à valeur dans  $\mathbb{R}^+$ .  $f_{\mathcal{Z}}(z)$  peut-être interprétée comme étant la mesure de probabilité d'appartenir à l'intervalle de longueur infinitésimale  $[z, z + dz]$  :  $f_{\mathcal{Z}}(z) = P_{\mathcal{Z}}([z, z + dz])$ .
  - ▶ La fonction de répartition est alors définie par :

$$F_{\mathcal{Z}}(z) = \int_{-\infty}^z f_{\mathcal{Z}}(x) dx$$

- ▶ On a également les propriétés suivantes :

$$\begin{aligned} P_{\mathcal{Z}}([a, b]) &= \int_a^b f_{\mathcal{Z}}(z) dz, \text{ pour tout } a < b \\ P_{\mathcal{Z}}(\mathbb{I}) &= \int_{z \in \mathbb{I}} f_{\mathcal{Z}}(z) dz, \text{ pour tout } \mathbb{I} \subseteq \mathbb{R} \end{aligned}$$

## Backgrd : Propriétés de la fonction de répartition

- Propriétés de la **fonction de répartition** :
  - ▶  $F_{\mathcal{Z}}$  est **non décroissante** sur  $\mathbb{R}$ .
  - ▶  $F_{\mathcal{Z}}$  varie entre 0 et 1 quand  $z$  varie entre  $-\infty$  et  $+\infty$ . En particulier :
    - $F_{\mathcal{Z}}(-\infty) = 0$
    - $F_{\mathcal{Z}}(+\infty) = 1$
  - ▶  $F_{\mathcal{Z}}$  est **continue à droite** et a une **limite à gauche** en tout  $z \in \mathbb{R}$ .
  - ▶ On a les probabilités d'intervalles suivantes :
    - Pour tout  $]a, b] \subset \mathbb{R}$  on a :  $P_{\mathcal{Z}}(]a, b]) = P(a < \mathcal{Z} \leq b) = F_{\mathcal{Z}}(b) - F_{\mathcal{Z}}(a)$ .
    - Pour tout  $[a, b] \subset \mathbb{R}$  on a :  $P_{\mathcal{Z}}([a, b]) = P(a \leq \mathcal{Z} \leq b) = F_{\mathcal{Z}}(b) - F_{\mathcal{Z}}(a) + P(\mathcal{Z} = a)$ .
    - Pour tout  $[a, b] \subset \mathbb{R}$  on a :  $P_{\mathcal{Z}}([a, b]) = P(a \leq \mathcal{Z} < b) = F_{\mathcal{Z}}(b) - F_{\mathcal{Z}}(a) - P(\mathcal{Z} = b) + P(\mathcal{Z} = a)$ .
  - ▶ On a la probabilité complémentaire suivante :
    - Pour tout  $a \in \mathbb{R}$  on a :  $P_{\mathcal{Z}}(]a, +\infty[) = 1 - P_{\mathcal{Z}}(]-\infty, a]) = 1 - F_{\mathcal{Z}}(a)$ .

## Backgrd : Quantiles, médiane et intervalle interquartile

- La notion de **quantile d'une v.a.r.**  $\mathcal{Z}$  est associée à la fct. de répartition  $F_{\mathcal{Z}}$ .
  - ▶ Le **quantile de niveau**  $\alpha \in [0, 1]$ , noté  $q(\alpha)$ , est le réel tel que :
 
$$P(\mathcal{Z} \leq q(\alpha)) = P_{\mathcal{Z}}(]-\infty, q(\alpha)]) = \alpha,$$
 ce qui est équivalent à,
 
$$F_{\mathcal{Z}}(q(\alpha)) = \alpha$$
  - ▶ C'est donc le réel tel que la probabilité que  $\mathcal{Z}$  lui soit inférieur vaut  $\alpha$ .
  - ▶ La **médiane** correspond par exemple au quantile de niveau  $\alpha = 0.5$  :
 
$$\text{Med}(\mathcal{Z}) = q(0.5) = F^{-1}(0.5)$$
- ▶ L'**intervalle interquartile** est défini par :  $\text{IR}(\mathcal{Z}) = [q(0.25), q(0.75)]$ .

## Backgrd : Espérance mathématique

- L'**espérance mathématique** de la v.a.r.  $\mathcal{Z}$  est, si elle existe, la valeur réelle dénotée  $E\{\mathcal{Z}\}$  et définie par :

$$E\{\mathcal{Z}\} = \int_{-\infty}^{+\infty} zdF_{\mathcal{Z}}(z) = \int_{-\infty}^{+\infty} zf_{\mathcal{Z}}(z)dz$$

- L'**espérance mathématique de la transformation**  $g(\mathcal{Z})$  de  $\mathcal{Z}$  peut aussi être définie sous condition d'existence de l'intégrale :

$$E\{g(\mathcal{Z})\} = \int_{-\infty}^{+\infty} g(z)dF_{\mathcal{Z}}(z) = \int_{-\infty}^{+\infty} g(z)f_{\mathcal{Z}}(z)dz$$

- **Propriété de linéarité de l'opérateur espérance** : soient  $\{\mathcal{Z}_k\}_k$  une suite de v.a.r. et  $\{a_k\}_k$  une suite de réels, on a :

$$E\left\{\sum_k a_k \mathcal{Z}_k\right\} = \sum_k a_k E\{\mathcal{Z}_k\}$$

## Backgrd : Couples de v.a.r.

- On suppose maintenant **deux v.a.r.**  $\mathcal{X}$  et  $\mathcal{Y}$  définis sur le même espace probabilisé  $(\Omega, \Sigma, P)$  et nous présentons des outils permettant de caractériser les relations entre ces deux variables. Ces deux v.a.r. forment un **couple**  $(\mathcal{X}, \mathcal{Y})$  et leurs réalisations prennent valeurs dans  $\mathbb{R} \times \mathbb{R} = \mathbb{R}^2$ , l'ens. des couples de réels.
- Ainsi, la **mesure de probabilité** dite **jointe**  $P_{\mathcal{X}, \mathcal{Y}}$  est une fonction portant sur les parties de  $\mathbb{R}^2$ .
- On appelle **fonction de répartition jointe** du couple  $(\mathcal{X}, \mathcal{Y})$  que l'on note par  $F_{\mathcal{X}, \mathcal{Y}}$ , la fonction définie sur  $\mathbb{R}^2$  par :

$$\begin{aligned} F_{\mathcal{X}, \mathcal{Y}}(x, y) &= P((\mathcal{X} \leq x) \cap (\mathcal{Y} \leq y)) = P(\mathcal{X} \leq x, \mathcal{Y} \leq y) \\ &= P_{\mathcal{X}, \mathcal{Y}}([-\infty, x] \times [-\infty, y]) \end{aligned}$$

- On retrouve la **fonction de répartition (marginale)** de chaque v.a.r. de la façon suivante :

$$F_{\mathcal{X}}(x) = F_{\mathcal{X}, \mathcal{Y}}(x, +\infty) \text{ et } F_{\mathcal{Y}}(y) = F_{\mathcal{X}, \mathcal{Y}}(+\infty, y)$$

## Backgrd : Variance et écart-type

- Dénotons par  $\mu_{\mathcal{Z}} = E\{\mathcal{Z}\}$ , l'espérance (ou moyenne) de  $\mathcal{Z}$ .
- La **variance mathématique** de la v.a.r.  $\mathcal{Z}$  est, si elle existe, la valeur réelle non négative dénotée  $V\{\mathcal{Z}\}$  et définie par :

$$\begin{aligned} V\{\mathcal{Z}\} &= E\{(Z - E\{\mathcal{Z}\})^2\} = E\{(Z - \mu_{\mathcal{Z}})^2\} \\ &= \int_{-\infty}^{+\infty} (z - \mu_{\mathcal{Z}})^2 dF_{\mathcal{Z}}(z) = \int_{-\infty}^{+\infty} (z - \mu_{\mathcal{Z}})^2 f_{\mathcal{Z}}(z) dz \end{aligned}$$

- On a également l'expression :

$$V\{\mathcal{Z}\} = E\{\mathcal{Z}^2\} - (E\{\mathcal{Z}\})^2 = E\{\mathcal{Z}^2\} - \mu_{\mathcal{Z}}^2$$

- Remarque :  $E\{(\mathcal{Z} - \mu_{\mathcal{Z}})^r\}$  s'appelle le **moment centré d'ordre r**.
- L'**écart-type** de  $\mathcal{Z}$  dénotée  $\sigma_{\mathcal{Z}}$  est défini par :  $\sigma_{\mathcal{Z}} = \sqrt{V\{\mathcal{Z}\}}$ .
- $V\{\mathcal{Z}\}$  et  $\sigma_{\mathcal{Z}}$  permettent de mesurer la **dispersion autour de la moyenne** de  $\mathcal{Z}$ . De plus,  $\sigma_{\mathcal{Z}}$  s'exprime dans la même unité que  $\mathcal{Z}$ .

## Backgrd : Couples de v.a.r. (suite)

- La **fonction de densité de probabilités jointes** notée  $f_{\mathcal{X}, \mathcal{Y}}$  est définie par tout couple  $(x, y) \in \mathbb{R}^2$  à partir de la relation suivante :

$$F_{\mathcal{X}, \mathcal{Y}}(x, y) = \int_{-\infty}^x \int_{-\infty}^y f_{\mathcal{X}, \mathcal{Y}}(s, t) ds dt$$

- Clairement, nous avons la définition suivante :

$$f_{\mathcal{X}, \mathcal{Y}}(x, y) = \frac{\partial^2 F_{\mathcal{X}, \mathcal{Y}}(x, y)}{\partial x \partial y}$$

- La **fonction de densité de probabilité marginale** de chaque v.a.r. est déterminée par :

$$\begin{aligned} \forall x \in \mathbb{R} : f_{\mathcal{X}}(x) &= \int_{-\infty}^{+\infty} f_{\mathcal{X}, \mathcal{Y}}(x, y) dy \\ \forall y \in \mathbb{R} : f_{\mathcal{Y}}(y) &= \int_{-\infty}^{+\infty} f_{\mathcal{X}, \mathcal{Y}}(x, y) dx \end{aligned}$$

## Backgrd : Indépendance entre deux v.a.r.

- La **fonction de densité conditionnelle de  $\mathcal{X}$  sachant  $\mathcal{Y}$** ,  $\forall y \in \mathbb{R}$  :

$$\forall x \in \mathbb{R} : f_{\mathcal{X}|\mathcal{Y}}(x|y) = \frac{f_{\mathcal{X},\mathcal{Y}}(x,y)}{f_{\mathcal{Y}}(y)}$$

- On peut de la même façon déterminer  $f_{\mathcal{Y}|\mathcal{X}}$ .
- $\mathcal{X}$  et  $\mathcal{Y}$  sont **indépendantes** ce que l'on note par  $\mathcal{X} \perp \mathcal{Y}$ , ssi :

$$\forall x, y \in \mathbb{R}^2 : F_{\mathcal{X},\mathcal{Y}}(x,y) = F_{\mathcal{X}}(x)F_{\mathcal{Y}}(y)$$

- Dans le cas continu,  $\mathcal{X} \perp \mathcal{Y}$  ssi :

$$\forall (x,y) \in \mathbb{R}^2 : f_{\mathcal{X},\mathcal{Y}}(x,y) = f_{\mathcal{X}}(x)f_{\mathcal{Y}}(y)$$

- Si  $\mathcal{X}$  et  $\mathcal{Y}$  sont indépendantes, alors  $g(\mathcal{X})$  et  $h(\mathcal{Y})$  sont également indépendantes pour toutes fonctions (mesurables)  $g$  et  $h$ .

## Backgrd : Covariance et indépendance

- On voit également facilement la **propriété de symétrie** de  $C$  :

$$C\{\mathcal{X}, \mathcal{Y}\} = C\{\mathcal{Y}, \mathcal{X}\}$$

- Nous avons ensuite les **propriétés de bilinéarité** suivantes :

$$C\{a\mathcal{X} + b\mathcal{Y}, \mathcal{Z}\} = aC\{\mathcal{X}, \mathcal{Z}\} + bC\{\mathcal{Y}, \mathcal{Z}\}$$

$$C\{a\mathcal{X} + b, c\mathcal{Y} + d\} = acC\{\mathcal{X}, \mathcal{Y}\}$$

- La covariance permet d'appréhender la relation entre  $\mathcal{X}$  et  $\mathcal{Y}$ . Notamment, **la covariance est nulle en cas d'indépendance** :

$$\mathcal{X} \perp \mathcal{Y} \Rightarrow C\{\mathcal{X}, \mathcal{Y}\} = 0$$

- Attention !** la réciproque est fausse. En cas de **covariance nulle** on parle de **non corrélation** :

$$C\{\mathcal{X}, \mathcal{Y}\} = 0 \Rightarrow \mathcal{X} \text{ et } \mathcal{Y} \text{ sont non corrélées}$$

## Backgrd : Espérance et covariance d'un couple de v.a.r.

- On peut généraliser l'**espérance** de  $g(\mathcal{X})$  au cas **d'une fonction d'un couple de v.a.r.** Soit  $g(\mathcal{X}, \mathcal{Y})$  une v.a. à valeur réelle :

$$E_{\mathcal{X},\mathcal{Y} \sim f_{\mathcal{X},\mathcal{Y}}} \{g(\mathcal{X}, \mathcal{Y})\} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(x, y) f_{\mathcal{X},\mathcal{Y}}(x, y) dx dy$$

- Pour alléger les notations, s'il n'y a pas d'ambiguïté, on notera  $E_{\mathcal{X},\mathcal{Y} \sim f_{\mathcal{X},\mathcal{Y}}} \{g(\mathcal{X}, \mathcal{Y})\}$  par  $E\{g(\mathcal{X}, \mathcal{Y})\}$ .
- La **propriété de linéarité de l'opérateur E** se généralise également :

$$E\{ag(\mathcal{X}) + bh(\mathcal{Y})\} = aE\{g(\mathcal{X})\} + bE\{h(\mathcal{Y})\}$$

- On introduit un nouvel opérateur, la **covariance**, défini comme suit :

$$\begin{aligned} C\{\mathcal{X}, \mathcal{Y}\} &= E_{\mathcal{X},\mathcal{Y} \sim f_{\mathcal{X},\mathcal{Y}}} \{(\mathcal{X} - E_{\mathcal{X} \sim f_{\mathcal{X}}} \{\mathcal{X}\})(\mathcal{Y} - E_{\mathcal{Y} \sim f_{\mathcal{Y}}} \{\mathcal{Y}\})\} \\ &= E\{(\mathcal{X} - E\{\mathcal{X}\})(\mathcal{Y} - E\{\mathcal{Y}\})\} \\ &= E\{(\mathcal{X} - \mu_{\mathcal{X}})(\mathcal{Y} - \mu_{\mathcal{Y}})\} \end{aligned}$$

- On peut facilement montrer que :  $C\{\mathcal{X}, \mathcal{Y}\} = E\{\mathcal{X}\mathcal{Y}\} - \mu_{\mathcal{X}}\mu_{\mathcal{Y}}$ .

## Backgrd : Variance et indépendance

- On remarquera le lien suivant entre la covariance et la variance :

$$C\{\mathcal{X}, \mathcal{X}\} = V\{\mathcal{X}\}$$

- Ensuite, nous avons la propriété :

$$\begin{aligned} V\{\mathcal{X} + \mathcal{Y}\} &= C\{\mathcal{X} + \mathcal{Y}, \mathcal{X} + \mathcal{Y}\} \\ &= C\{\mathcal{X}, \mathcal{X}\} + C\{\mathcal{X}, \mathcal{Y}\} + C\{\mathcal{Y}, \mathcal{X}\} + C\{\mathcal{Y}, \mathcal{Y}\} \\ &= V\{\mathcal{X}\} + V\{\mathcal{Y}\} + 2C\{\mathcal{X}, \mathcal{Y}\} \end{aligned}$$

- En cas d'indépendance, nous avons donc :

$$\mathcal{X} \perp \mathcal{Y} \Rightarrow V\{\mathcal{X} + \mathcal{Y}\} = V\{\mathcal{X}\} + V\{\mathcal{Y}\}$$

## Backgrd : Coefficient de corrélation linéaire

- Le **coefficient de corrélation linéaire** est une mesure statistique classique permettant d'apprécier certains types de dépendance entre deux v.a.r.  $\mathcal{X}$  et  $\mathcal{Y}$ . Il est noté  $\rho_{\mathcal{X},\mathcal{Y}}$  et est défini par :

$$\rho_{\mathcal{X},\mathcal{Y}} = \frac{C\{\mathcal{X}, \mathcal{Y}\}}{\sigma_{\mathcal{X}}\sigma_{\mathcal{Y}}}$$

- Nous avons les propriétés suivantes pour  $\rho_{\mathcal{X},\mathcal{Y}}$  :

- ▶ Symétrie :  $\rho_{\mathcal{X},\mathcal{Y}} = \rho_{\mathcal{Y},\mathcal{X}}$ .
- ▶ Bornes :  $-1 \leq \rho_{\mathcal{X},\mathcal{Y}} \leq 1$  ( $\rho_{\mathcal{X},\mathcal{Y}}$  plus facile à interpréter que  $C\{\mathcal{X}, \mathcal{Y}\}$ ).
- ▶ Bornes atteintes en cas de dépendance linéaire : si  $\mathcal{Y} = a\mathcal{X} + b$  avec  $a \neq 0$ , alors  $|\rho_{\mathcal{X},\mathcal{Y}}| = 1$ .
- ▶ Invariante par transformation affine de  $\mathcal{X}$  et/ou  $\mathcal{Y}$  : si  $g(\mathcal{X}) = a\mathcal{X} + b$  et  $h(\mathcal{Y}) = c\mathcal{Y} + d$ , avec  $a, c \neq 0$ , alors  $\rho_{g(\mathcal{X}), h(\mathcal{Y})} = \rho_{\mathcal{X},\mathcal{Y}}$ .
- ▶ Nulle en cas d'indépendance :  $\mathcal{X} \perp \mathcal{Y} \Rightarrow \rho_{\mathcal{X},\mathcal{Y}} = 0$ .
- ▶ **Attention !** ici aussi,  $\rho_{\mathcal{X},\mathcal{Y}} = 0 \not\Rightarrow \mathcal{X} \perp \mathcal{Y}$ .
- ▶ **Attention !** corrélation  $\not\Rightarrow$  causalité.

## Backgrd : Variogramme, cas d'un couple de v.a.r. ( $\mathcal{X}, \mathcal{Y}$ )

- Le **variogramme** (ou **semivariogramme**) est une autre mesure permettant d'apprécier certains types de dépendance entre deux v.a.r.  $\mathcal{X}$  et  $\mathcal{Y}$ . Il est noté  $\gamma_{\mathcal{X},\mathcal{Y}}$ , il est non négatif et est défini par :

$$\begin{aligned}\gamma_{\mathcal{X},\mathcal{Y}} &= \frac{1}{2}V\{\mathcal{X} - \mathcal{Y}\} \\ &= \frac{1}{2}C\{\mathcal{X} - \mathcal{Y}, \mathcal{X} - \mathcal{Y}\} \\ &= \frac{1}{2}(C\{\mathcal{X}, \mathcal{X}\} - C\{\mathcal{X}, \mathcal{Y}\} - C\{\mathcal{Y}, \mathcal{X}\} + C\{\mathcal{Y}, \mathcal{Y}\}) \\ &= \frac{1}{2}(V\{\mathcal{X}\} + V\{\mathcal{Y}\} - 2C\{\mathcal{X}, \mathcal{Y}\})\end{aligned}$$

- Si on standardise les v.a.r.,  $\mathcal{X}' = \frac{\mathcal{X} - \mu_{\mathcal{X}}}{\sigma_{\mathcal{X}}}$  et  $\mathcal{Y}' = \frac{\mathcal{Y} - \mu_{\mathcal{Y}}}{\sigma_{\mathcal{Y}}}$ , on a :

$$\gamma_{\mathcal{X}',\mathcal{Y}'} = 1 - \rho_{\mathcal{X},\mathcal{Y}}, \text{ et on voit que } \gamma_{\mathcal{X}',\mathcal{Y}'} \in [0, 2]$$

- $\gamma_{\mathcal{X},\mathcal{Y}} \leftrightarrow$  dissimilarité tandis que  $\rho_{\mathcal{X},\mathcal{Y}} \leftrightarrow$  similarité.

## Backgrd : Coefficient de corrélation linéaire (suite)

- La **standardisation** d'une v.a.r.  $\mathcal{X}$  consiste à retrancher la moyenne et à diviser par l'écart-type. Nous définissons alors une nouvelle v.a.r. que nous noterons ici  $\mathcal{X}'$  donnée par :

$$\mathcal{X}' = \frac{\mathcal{X} - \mu_{\mathcal{X}}}{\sigma_{\mathcal{X}}}$$

- Une variable standardisée  $\mathcal{X}'$  vérifie les propriétés suivantes :

$$\mu_{\mathcal{X}'} = 0 \text{ et } \sigma_{\mathcal{X}'} = 1, \text{ autrement dit } \mathcal{X}' \text{ est centrée et réduite.}$$

- Soient  $\mathcal{X}'$  et  $\mathcal{Y}'$  deux v.a.r. standardisées, on a la propriété suivante :

$$\begin{aligned}C\{\mathcal{X}', \mathcal{Y}'\} &= E\{\mathcal{X}'\mathcal{Y}'\} - \mu_{\mathcal{X}'}\mu_{\mathcal{Y}'} \\ &= E\left\{\left(\frac{\mathcal{X} - \mu_{\mathcal{X}}}{\sigma_{\mathcal{X}}}\right)\left(\frac{\mathcal{Y} - \mu_{\mathcal{Y}}}{\sigma_{\mathcal{Y}}}\right)\right\} \\ &= \rho_{\mathcal{X},\mathcal{Y}} \\ &\in [-1, 1]\end{aligned}$$

## Fct. variogramme et variogramme pour un couple de v.a.r.

- Remarque : **en géostatistique**, l'utilisation de la mesure de variogramme  $\gamma_{\mathcal{X},\mathcal{Y}}$  pour le couple de v.a.r.  $(\mathcal{X}, \mathcal{Y})$  peut se faire de différentes façons selon les cas où  $\mathcal{X}$  et  $\mathcal{Y}$  peuvent correspondre soit :
  - 1 à deux variables différentes mesurées en la même position comme par ex.  $\mathcal{X} = Z^1(\mathbf{p})$  et  $\mathcal{Y} = Z^2(\mathbf{p})$ ,
  - 2 à la même variable mais mesurés à deux positions différentes comme par ex.  $\mathcal{X} = Z(\mathbf{p} + \mathbf{h})$  et  $\mathcal{Y} = Z(\mathbf{p})$ ,
  - 3 à deux variables différentes et mesurées en deux positions distinctes comme par ex.  $\mathcal{X} = Z^1(\mathbf{p} + \mathbf{h})$  et  $\mathcal{Y} = Z^2(\mathbf{p})$ .
- Dans tous ces cas,  $\gamma_{\mathcal{X},\mathcal{Y}}$  indique une mesure de dissimilarité/variabilité entre les v.a.r. correspondantes  $\mathcal{X}$  et  $\mathcal{Y}$ .
- Le **cas 2** en géostat. correspond à la **tâche d'interpolation spatiale** : il s'agit d'être en mesure de prédire  $Z(\mathbf{p})$  en tout  $\mathbf{p} \in \mathbb{D}$  en exploitant les  $n$  observations  $\{z(\mathbf{p}_i)\}_{i=1,\dots,n}$ .
- Dans ce qui suit, nous nous intéressons spécifiquement au cas 2 et ainsi au problème d'interpolation d'une fonction  $Z$  pouvant prendre des valeurs aléatoires en tout point  $\mathbf{p}$  d'un domaine  $\mathbb{D}$ .

## Backgrd : Moments empiriques

- Mettons nous dans un **cadre statistique** où on étudie les v.a.r.  $\mathcal{X}$  et  $\mathcal{Y}$  à partir d'un **ensemble d'observations** que nous supposons de taille n :  $\{x_i\}_{i=1,\dots,n}$  et  $\{y_i\}_{i=1,\dots,n}$ .
- Les statistiques inférentielles consistent notamment à inférer des données observées des **estimations des paramètres** des v.a.r.
- Nous avons les **estimateurs classiques** pour les moments de  $\mathcal{X}$  :

$$\widehat{E\{\mathcal{X}\}} = \widehat{\mu_{\mathcal{X}}} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\widehat{E\{\mathcal{X}^2\}} = \widehat{\mu_{\mathcal{X}^2}} = \frac{1}{n} \sum_{i=1}^n x_i^2$$

$$\widehat{V\{\mathcal{X}\}} = \widehat{\mu_{\mathcal{X}^2}} - (\widehat{\mu_{\mathcal{X}}})^2$$

$$\widehat{\sigma_{\mathcal{X}}} = \sqrt{\widehat{\mu_{\mathcal{X}^2}} - (\widehat{\mu_{\mathcal{X}}})^2}$$

## Backgrd : Moments empiriques (suite)

- Dans le cas du couple de v.a.r.  $(\mathcal{X}, \mathcal{Y})$ , nous introduisons :

$$\widehat{E\{\mathcal{X}\mathcal{Y}\}} = \widehat{\mu_{\mathcal{X}\mathcal{Y}}} = \frac{1}{n} \sum_{i=1}^n x_i y_i$$

- Nous avons alors les **mesures bivariées empiriques** suivantes :

$$\begin{aligned} \widehat{\rho_{\mathcal{X},\mathcal{Y}}} &= \frac{\widehat{C\{\mathcal{X},\mathcal{Y}\}}}{\widehat{\sigma_{\mathcal{X}}}\widehat{\sigma_{\mathcal{Y}}}} \\ \widehat{\gamma_{\mathcal{X},\mathcal{Y}}} &= \frac{1}{2} \left( \underbrace{(\widehat{\sigma_{\mathcal{X}}})^2}_{\widehat{V\{\mathcal{X}\}}} + \underbrace{(\widehat{\sigma_{\mathcal{Y}}})^2}_{\widehat{V\{\mathcal{Y}\}}} - 2\widehat{C\{\mathcal{X},\mathcal{Y}\}} \right) \end{aligned}$$

## Champ aléatoire, variogramme et Krigeage

- Nous présentons ci-après des éléments centraux en **géostatistique** :
  - ▶ Nous introduisons d'abord des concepts mathématiques importants en géostatistique : le phénomène à l'étude est vu tel un **processus aléatoire spatial continu**  $\mathcal{Z}$  défini sur un domaine  $\mathbb{D} \subset \mathbb{R}^p$  ( $p = 2$  en général en stat. spatiale, on parle alors de champ aléatoire). Il s'agit d'un cas spécifique de **fonction aléatoire**.
  - ▶ Puis, nous détaillons l'utilisation spécifique du **variogramme** en stat. spatiale qui permet de décrire la structure de dépendance spatiale des données. Il s'agit d'une fonction qui mesure la variance des différences entre les valeurs d'échantillons en fonction de la distance qui les sépare.
  - ▶ Enfin, nous présenterons la **méthode d'interpolation probabiliste du Krigeage** pour estimer des valeurs non échantillonées. Le Krigeage repose sur la théorie des **fonctions aléatoires possédant des propriétés de stationnarité** et utilise la moyenne et la covariance spatiales pour produire une estimation de  $\mathcal{Z}$  en tout  $\mathbf{p} \in \mathbb{D}$ .
  - ▶ En analyse spatiale,  $\text{Dim}(\mathbb{D}) = 2$  en général.

## Rappel du Sommaire

### 6 Eléments de géostatistique

- Les champs aléatoires
- Variogramme
- Les méthodes de Krigeage

## Fonction aléatoire

- Soit  $\mathbb{D}$  un sous-ensemble de  $\mathbb{R}^p$  et soit  $(\Omega, \Sigma, P)$  un espace probabilisé.
- Une **fonction aléatoire**  $\mathcal{Z}$  définie sur  $\mathbb{D} \times \Omega$  est une fonction de deux arguments  $\mathcal{Z}(\mathbf{p}, \omega)$  tel que pour tout  $\mathbf{p} \in \mathbb{D}$ , la fonction partielle  $\mathcal{Z}(\mathbf{p}, .)$  en  $\mathbf{p}$  est une **variable aléatoire sur**  $(\Omega, \Sigma, P)$ .
- La fonction partielle  $\mathcal{Z}(., \omega)$  est une **fonction définie sur**  $\mathbb{D}$  qui représente la **réalisation de la fct. aléatoire**  $\mathcal{Z}$  pour  $\omega \in \Omega$  donné.
- Si on raisonne avec un quelconque  $\omega \in \Omega$ , on dénote alors plus brièvement  $\mathcal{Z}(\mathbf{p})$  la fonction aléatoire et par  $z(\mathbf{p})$  la réalisation de  $\mathcal{Z}$  en  $\mathbf{p} \in \mathbb{D}$  (pour un quelconque  $\omega \in \Omega$ ).
- En théorie des probabilités, lorsque  $\text{Dim}(\mathbb{D}) = 1$ , la fonction aléatoire est plus communément appelée **processus stochastique**, et lorsque  $\text{Dim}(\mathbb{D}) > 1$ , on parle de **champ aléatoire** (ou stochastique).

## Distribution spatiale (suite)

- Cependant, des développements techniques en mathématiques reposant sur l'**hypothèse de séparabilité** de la fonction aléatoire permettent de surmonter cette difficulté : une fonction aléatoire est **séparable** si toute probabilité impliquant un ens. non dénombrable de points peut être déterminée de façon unique à partir de probabilités d'ens. dénombrable de points (ex. points de coordonnées rationnelles dans  $\mathbb{R}^p$  -ens. dense dans  $\mathbb{R}^p$ -).
- Un résultat fondamental de [Doob, 1990] établit le fait que pour toute fonction aléatoire il existe toujours une fonction aléatoire séparable ayant la même distribution spatiale. On peut donc toujours se ramener à une fonction aléatoire séparable.
- Désormais, on supposera que la fonction aléatoire  $\mathcal{Z}$  est à valeur réelle càd que  $\mathcal{Z}(\mathbf{p}, \omega) \in \mathbb{R}, \forall \mathbf{p} \in \mathbb{D}, \forall \omega \in \Omega$ . De plus, on supposera que  $\mathcal{Z}(\mathbf{p})$  possède une fonction de densité de probabilité  $f_{\mathcal{Z}(\mathbf{p})}, \forall \mathbf{p} \in \mathbb{D}$ .

## Distribution spatiale

- Une fonction aléatoire est décrite par ses **fonctions de distribution de dimensions finies** (ou **distributions fini-dimensionnelles**) : c'est l'ensemble pour tout  $k \in \mathbb{N}, k < \infty$ , de toutes les fonctions de distribution de probabilité de tous les  $k$ -uplets  $(\mathcal{Z}(\mathbf{p}_1), \mathcal{Z}(\mathbf{p}_2), \dots, \mathcal{Z}(\mathbf{p}_k))$  pour toutes les configurations spatiales possibles des points  $\mathbf{p}_1, \dots, \mathbf{p}_k$  :

$$\begin{aligned} F_{\mathcal{Z}(\mathbf{p}_1), \dots, \mathcal{Z}(\mathbf{p}_k)}(z_1, \dots, z_k) &= P(\mathcal{Z}(\mathbf{p}_1) \leq z_1, \dots, \mathcal{Z}(\mathbf{p}_k) \leq z_k) \\ &= P_{\mathcal{Z}(\mathbf{p}_1), \dots, \mathcal{Z}(\mathbf{p}_k)}([-\infty, z_1] \times \dots \times [-\infty, z_k]) \end{aligned}$$

- L'ens. de ces fonctions est appelée **distribution spatiale** de  $\mathcal{Z}$ .
- En théorie, la distribution spatiale de  $\mathcal{Z}$  n'est pas suffisante afin de calculer la prob. d'évènements impliquant des ens. de points non dénombrables (donc infini) comme par ex.  $P(\text{Sup}(\mathcal{Z}(\mathbf{p}) : \mathbf{p} \in \mathbb{V}) < z_0)$  (prob. que la valeur max. dans le domaine  $\mathbb{V}$  soit plus petite que  $z_0$ ).

## Moments d'une fonction aléatoire à valeur réelle

- La **moyenne d'une fct. aléatoire**  $\mathcal{Z}$  est une fct. notée  $\mu$ , qui est définie sur  $\mathbb{D} \subset \mathbb{R}^p$  et donnée par l'opérateur espérance,  $\forall \mathbf{p} \in \mathbb{D}$  :

$$\mu(\mathbf{p}) = E\{\mathcal{Z}(\mathbf{p}, \omega)\} = \int_{\Omega} \mathcal{Z}(\mathbf{p}, \omega) dP(\omega) = \int_{\mathbb{R}} z f_{\mathcal{Z}(\mathbf{p})}(z) dz$$

- La **fonction de covariance (centrée)** d'une fct. aléatoire  $\mathcal{Z}$  est une fct.  $C$  définie sur  $\mathbb{D} \times \mathbb{D}$  qui pour tout couple défini  $(\mathbf{p}, \mathbf{p}')$  est égale à l'opérateur covariance appliqué au couple de v.a.r.  $(\mathcal{Z}(\mathbf{p}), \mathcal{Z}(\mathbf{p}'))$  :

$$\begin{aligned} C(\mathbf{p}, \mathbf{p}') &= E\{(\mathcal{Z}(\mathbf{p}, \omega) - \mu(\mathbf{p}))(\mathcal{Z}(\mathbf{p}', \omega) - \mu(\mathbf{p}'))\} \\ &= C\{\mathcal{Z}(\mathbf{p}), \mathcal{Z}(\mathbf{p}')\} \end{aligned}$$

- Si  $\mathbf{p} = \mathbf{p}'$  alors  $C(\mathbf{p}, \mathbf{p}) = C\{\mathcal{Z}(\mathbf{p}), \mathcal{Z}(\mathbf{p})\} = V\{\mathcal{Z}(\mathbf{p})\}$ , l'opérateur variance de la fct. aléatoire  $\mathcal{Z}$  évaluée en la v.a.r.  $\mathcal{Z}(\mathbf{p})$ .
- On supposera désormais que  $\mathcal{Z}$  est une **fct. aléatoire à valeur réelle de 2nd ordre**. Autrement dit, tous les moments d'ordre 2 sont finis :

$$E\{\mathcal{Z}(\mathbf{p}, \omega)^2\} < \infty, \quad \forall \mathbf{p} \in \mathbb{D}$$

## Stationnarité(s) d'une fonction aléatoire

- $\mathcal{Z}$  est dite **stationnaire au sens fort** si sa distribution spatiale ne change pas sous l'action d'une translation  $\mathbf{h} \in \mathbb{R}^p$  c.à.d. si :

$$P(\mathcal{Z}(\mathbf{p}_1) < z_1, \dots, \mathcal{Z}(\mathbf{p}_k) < z_k) = P(\mathcal{Z}(\mathbf{p}_1 + \mathbf{h}) < z_1, \dots, \mathcal{Z}(\mathbf{p}_k + \mathbf{h}) < z_k)$$

- $\mathcal{Z}$  est dite **stationnaire au sens faible** (ou de 2nd ordre, ou **SRF** -Second order Random Function-) si elle vérifie (notations allégées, les espérances sont par rapport à  $f_{\mathcal{Z}(\mathbf{p})}$  sans perte de généralité) :

$$\begin{cases} E\{\mathcal{Z}(\mathbf{p})\} = \mu & \forall \mathbf{p} \in \mathbb{D} \\ C\{\mathcal{Z}(\mathbf{p} + \mathbf{h}), \mathcal{Z}(\mathbf{p})\} = C(\mathbf{h}) & \forall \mathbf{p} \in \mathbb{D}, \forall \mathbf{p}' = \mathbf{p} + \mathbf{h} \in \mathbb{D} \end{cases}$$

Autrement dit :

- ▶ **La moyenne est constante.**
- ▶ **La fct. de covariance  $C$  ne dépend que de  $\mathbf{h} = \mathbf{p}' - \mathbf{p}$ .**
- Si de plus la covariance dépend uniquement de la norme  $\|\mathbf{h}\|$  et non pas de la direction de  $\mathbf{h}$ , on dit dans ce cas que  $\mathcal{Z}$  est une fonction aléatoire SRF qui est **isotropique**.

## Propriétés de la fct. de variogramme d'une SRF

- Si  $\mathcal{Z}$  est une SRF alors :

- ▶ **La fonction de variogramme aussi dénotée  $\gamma$  ne dépend que de  $\mathbf{h}$ :**

$$\begin{aligned} \gamma(\mathbf{p} + \mathbf{h}, \mathbf{p}) &\equiv \gamma_{\mathcal{Z}(\mathbf{p} + \mathbf{h}), \mathcal{Z}(\mathbf{p})} \equiv \frac{1}{2} V\{\mathcal{Z}(\mathbf{p} + \mathbf{h}) - \mathcal{Z}(\mathbf{p})\} \\ &= \frac{1}{2} (C(\mathbf{0}) + C(\mathbf{0}) - 2C(\mathbf{h})) \\ &= C(\mathbf{0}) - C(\mathbf{h}) \\ &= \gamma(\mathbf{h}) \end{aligned}$$

- Si  $\mathcal{Z}$  est une SRF alors sa fct. de variogramme  $\gamma$  vérifie :

- ▶  $\gamma(\mathbf{0}) = 0$ .
- ▶  $\gamma(\mathbf{h}) \geq 0$ .
- ▶  $\gamma(-\mathbf{h}) = \gamma(\mathbf{h})$ .
- ▶  $\gamma(\mathbf{h}) = C(\mathbf{0}) - C(\mathbf{h})$  (vue précédemment).
- ▶  $\lim_{\|\mathbf{h}\| \rightarrow \infty} \gamma(\mathbf{h}) = C(\mathbf{0}) - \lim_{\|\mathbf{h}\| \rightarrow \infty} C(\mathbf{h}) = C(\mathbf{0})$ . La fonction variogramme atteint une valeur limite qui est égale à la variance.

## Propriétés d'une SRF

- Si  $\mathcal{Z}$  est une SRF alors nous pouvons la représenter comme suit :

$$\mathcal{Z}(\mathbf{p}) = \mu + \epsilon(\mathbf{p})$$

où  $\epsilon$  est une SRF avec  $E\{\epsilon(\mathbf{p})\} = 0, \forall \mathbf{p} \in \mathbb{D}$  et de covariance t.q. :

$$\begin{aligned} C\{\epsilon(\mathbf{p} + \mathbf{h}), \epsilon(\mathbf{p})\} &= E\{\epsilon(\mathbf{p} + \mathbf{h})\epsilon(\mathbf{p})\} \\ &= C\{\mathcal{Z}(\mathbf{p} + \mathbf{h}), \mathcal{Z}(\mathbf{p})\} \\ &= C(\mathbf{h}) \end{aligned}$$

- Si  $\mathcal{Z}$  est une SRF alors nous avons les propriétés suivantes :

- ▶ **La variance est constante :**

$$V\{\mathcal{Z}(\mathbf{p})\} = C(\mathbf{0}) = \sigma^2, \quad \forall \mathbf{p} \in \mathbb{D}$$

- ▶ **La fonction de corrélation aussi dénotée  $\rho$  ne dépend que de  $\mathbf{h}$  :**

$$\begin{aligned} \rho(\mathbf{p} + \mathbf{h}, \mathbf{p}) &\equiv \rho_{\mathcal{Z}(\mathbf{p} + \mathbf{h}), \mathcal{Z}(\mathbf{p})} = \frac{C(\mathbf{h})}{C(\mathbf{0})} \\ &= \rho(\mathbf{h}) \end{aligned}$$

## Fonction aléatoire intrinsèquement stationnaire

- La stationnarité au sens faible (et au sens fort) représente des hypothèses restrictives. En géostatistique, on utilise plus couramment le concept plus large de **stationnarité intrinsèque**.

- $\mathcal{Z}$  est dite **intrinsèquement stationnaire** (ou **IRF** -Intrinsic Random Function-) si les fonctions aléatoires associées aux **accroissements**  $\mathcal{Y}_{\mathbf{h}}(\mathbf{p}) = \mathcal{Z}(\mathbf{p} + \mathbf{h}) - \mathcal{Z}(\mathbf{p})$  sont faiblement stationnaires pour tout vecteur  $\mathbf{h}$ . Cela revient à satisfaire aux relations suivantes :

$$\begin{cases} E\{\mathcal{Z}(\mathbf{p} + \mathbf{h}) - \mathcal{Z}(\mathbf{p})\} = \langle \mathbf{a}, \mathbf{h} \rangle & \forall \mathbf{p} \in \mathbb{D} \\ \frac{1}{2} V\{\mathcal{Z}(\mathbf{p} + \mathbf{h}) - \mathcal{Z}(\mathbf{p})\} = \gamma(\mathbf{h}) & \forall \mathbf{p} \in \mathbb{D}, \forall \mathbf{p} + \mathbf{h} \in \mathbb{D} \end{cases}$$

- $\langle \mathbf{a}, \mathbf{h} \rangle$  est la **fonction de tendance** (ou *drift*) de  $\mathcal{Y}_{\mathbf{h}}$ . C'est nécessairement une fonction linéaire de  $\mathbf{h}$ .

- $\gamma : \mathbb{D} \rightarrow \mathbb{R}^+$  est la **fonction de variogramme** (cf slide 415).

## Fonction aléatoire intrinsèquement stationnaire (suite)

- Dans le cas où  $\mathcal{Z}$  est une IRF avec une **tendance nulle**  $\mu(\mathbf{p}) = 0$  ( $\mathbf{a} = \mathbf{0}$ ), nous avons les propriétés équivalentes suivantes :

$$\begin{cases} E\{\mathcal{Z}(\mathbf{p} + \mathbf{h}) - \mathcal{Z}(\mathbf{p})\} = 0 & \forall \mathbf{p} \in \mathbb{D} \\ \frac{1}{2}E\{(\mathcal{Z}(\mathbf{p} + \mathbf{h}) - \mathcal{Z}(\mathbf{p}))^2\} = \gamma(\mathbf{h}) & \forall \mathbf{p} \in \mathbb{D}, \forall \mathbf{p} + \mathbf{h} \in \mathbb{D} \end{cases}$$

- La stationnarité intrinsèque ne fait pas d'hypothèses sur l'existence des moments de  $\mathcal{Z}(\mathbf{p})$  mais uniquement sur l'existence des moments des incrément  $\mathcal{Y}_{\mathbf{h}}(\mathbf{p}) = \mathcal{Z}(\mathbf{p} + \mathbf{h}) - \mathcal{Z}(\mathbf{p})$  ce qui est moins restrictif. Toute fonction aléatoire SRF est IRF mais l'inverse n'est pas vraie.
- La relation bijective  $\gamma(\mathbf{h}) = C(\mathbf{0}) - C(\mathbf{h})$  permettant de passer d'une fct. de variogramme à une fct. de covariance et vice versa n'est valide que sous l'hypothèse de stationnarité au sens faible.

## Fonction aléatoire Gaussienne

- Une fonction aléatoire  $\mathcal{Z}$  est dite Gaussienne si toutes ses fonctions de distribution de dimensions finies (distribution spatiale de  $\mathcal{Z}$ ) sont Gaussiennes.
- Si  $\mathcal{Z}$  est une fonction aléatoire Gaussienne, alors la connaissance de la fonction moyenne  $\mu(\mathbf{p})$  et de la fonction de covariance  $C(\mathbf{p}, \mathbf{p}')$  suffisent pour déterminer la distribution spatiale de  $\mathcal{Z}$ .
- Si  $\mathcal{Z}$  est une fonction aléatoire Gaussienne alors stationnarité au sens faible est équivalente à stationnarité au sens fort.
- Si  $\mathcal{Z}$  est une fonction aléatoire Gaussienne IRF alors les incrément  $\mathcal{Y}_{\mathbf{h}}$  sont des fonctions aléatoires Gaussiennes.
- Attention !** Si on suppose que seules les distributions marginales de  $\mathcal{Z}$  sont Gaussiennes alors cela n'implique pas nécessairement que  $\mathcal{Z}$  est une fonction aléatoire Gaussienne.

## Fonction aléatoire intrinsèquement stationnaire (suite)

- Si  $\mathcal{Z}$  est une IRF, la fct. variogramme ne permet pas, **en général**, de retrouver la fct. covariance.
- Toutefois**, si  $\mathcal{Z}$  est une IRF dont la fct. variogramme est bornée,  $\lim_{\|\mathbf{h}\| \rightarrow \infty} \gamma(\mathbf{h}) = \gamma(\infty) < \infty$ , alors  $\mathcal{Z}$  est une **SRF** et on a :

$$C(\mathbf{h}) = \underbrace{\gamma(\infty)}_{<\infty} - \gamma(\mathbf{h})$$

- Si  $\mathcal{Z}$  est une IRF dont la fct. variogramme est bornée alors on peut passer de  $\gamma$  à  $C$  et vice versa.
- Si  $\gamma(\infty) < \infty$  alors la valeur de stabilisation de  $\gamma$  est appelée le **palier (sill)**. En pratique, si le variogramme a un **palier** alors il faudra prendre  $\gamma(\infty)$  plus grand ou égale à cette valeur. L'existence d'un palier indique qu'il existe une distance appelée la **portée (range)** au-dessus de laquelle, on suppose qu'il y a indépendance.

## Palier et portée et effet pépite

- Palier (sill)** : valeur maximale atteinte par le variogramme à grande distance. Il représente la variance totale des données lorsque les points ne sont plus spatialement corrélés.
  - ▶ Interprétation : il traduit la limite de l'hétérogénéité spatiale.
  - ▶ Importance : le palier est crucial pour déterminer la variabilité globale des données, influençant directement les prédictions spatiales et les incertitudes associées.
- Portée (range)** : distance à laquelle le variogramme atteint environ 95% du palier. Elle délimite la zone d'influence spatiale, c.à.d. la dist. max. sur laquelle deux points restent corrélés.
  - ▶ Interprétation : elle indique jusqu'à quelle dist. les valeurs des échantillons sont liées. Au-delà de cette distance, les valeurs sont considérées comme indépendantes.
  - ▶ Importance : la portée détermine le rayon d'action des modèles géostatistiques. Elle influence aussi le choix de la taille des fenêtres pour l'interpolation spatiale.

## Effet pépite

- **Effet pépite (nugget)** : il correspond à l'ordonnée à l'origine du variogramme (discontinuité à distance zéro). Il traduit une variabilité à très petite échelle ou des erreurs de mesure.

- ▶ Interprétation : il reflète les variations non résolues par le maillage d'échantillonnage ou les incertitudes des mesures.
- ▶ Importance : négliger un effet pépite peut fausser les modèles en sous-estimant la variabilité spatiale à petite échelle, tandis qu'une surestimation conduit à introduire des incertitudes inutiles.

## Variogrammes bornés et non bornés

- **Variogrammes bornés** :  $\gamma(\infty) < \infty$  ce qui implique une fct. de covariance définie qui a la forme  $C(\mathbf{h}) = \gamma(\infty) - \gamma(\mathbf{h})$ .

- ▶ Le palier existe et vaut  $\gamma(\infty)$ .
- ▶ La portée existe.
- ▶ L'effet pépite est, si il y a lieu, la discontinuité de  $\gamma$  en **0**.

- **Variogrammes non bornés** :  $\gamma(\infty) = \infty$  ce qui implique une fct. de covariance qui n'est pas formalisée sous les hypothèses de stationnarité précédentes.

- ▶ Il n'y a pas de palier.
- ▶ Il n'y a pas de portée.
- ▶ L'effet pépite est, si il y a lieu, la discontinuité de  $\gamma$  en **0**.
- ▶ La pente traduit une certaine non stationnarité et donc l'incompatibilité des données avec les hypothèses précédentes et la nécessité d'utiliser des outils plus avancés que ceux de la géostatistique standard.

## Exemples de variogrammes bornés et non bornés

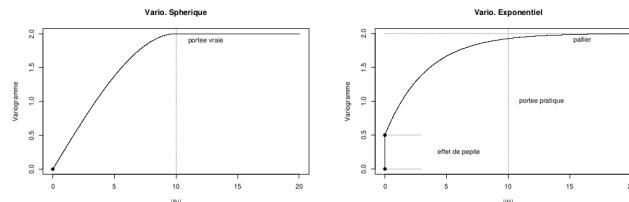


FIGURE 2 – Deux exemples de variogrammes bornés.

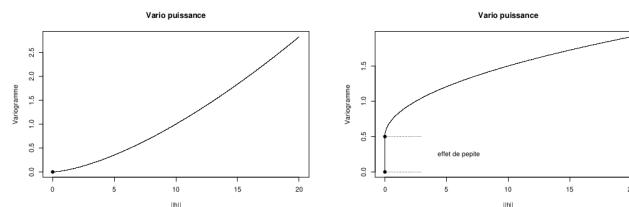


FIGURE 3 – Deux exemples de variogrammes puissances.

## Rappel du Sommaire

### 6 Eléments de géostatistique

- Les champs aléatoires
- Variogramme
- Les méthodes de Krigeage

## Estimation du variogramme : motivations

- On suppose que  $\mathcal{Z}$  est SRF ou IRF avec  $\gamma(\infty) < \infty$ . Il est alors possible de raisonner soit avec  $C(\mathbf{h})$  soit avec  $\gamma(\mathbf{h})$ . En géostatistique, on privilégie l'usage de  $\gamma(\mathbf{h})$  pour les raisons suivantes :
  - Hypothèses moins restrictives :
    - Le variogramme a une classe de fonctions plus vaste que la fonction de covariance. Ainsi, estimer un variogramme permet de poser des hypothèses moins contraignantes.
    - La présence d'un palier dans le variogramme permet de vérifier si  $\mathcal{Z}$  est SRF.
  - Moins de biais dans l'estimation :
    - L'estimation de  $C$  nécessite la connaissance de  $\mu$  qui est souvent inconnue. Or  $\gamma$  n'emploie pas  $\mu$ .
    - Remplacer  $\mu$  par une estimation  $\hat{\mu}$  introduit un biais irréductible dans l'estimation de  $C$ . Ce pb. est donc évité en estimant  $\gamma$ .

## Variogramme expérimental

- Le variogramme expérimental est au variogramme théorique ce que l'histogramme est à la fonction de densité : une représentation graphique, nécessairement imparfaite, construite à partir des données.
- On suppose  $n$  points  $(\mathbf{p}_1, \dots, \mathbf{p}_n)$  de  $\mathbb{D} \subset \mathbb{R}^p$  et on note  $\mathcal{Z}(\mathbf{p}_1), \dots, \mathcal{Z}(\mathbf{p}_n)$  les var. aléatoires associées à  $\mathcal{Z}$  en ces  $n$  points.
- Pour alléger les notations on définit  $\mathbf{h}_{ii'} \equiv \mathbf{p}_{i'} - \mathbf{p}_i$ .
- Le **variogramme expérimental** de  $\mathcal{Z}$  déterminé à partir de  $(\mathcal{Z}(\mathbf{p}_1), \dots, \mathcal{Z}(\mathbf{p}_n))$  est la fonction  $\hat{\gamma}$  définie par :

$$\hat{\gamma}(\mathbf{h}) = \frac{1}{2N(\mathbf{h})} \sum_{i,i':\mathbf{h}_{ii'} \approx \mathbf{h}} (\mathcal{Z}(\mathbf{p}_i) - \mathcal{Z}(\mathbf{p}_{i'}))^2$$

où  $N(\mathbf{h})$  est le nombre de paires  $(\mathbf{p}_i, \mathbf{p}_{i'})$  telles que  $\mathbf{h}_{ii'} \approx \mathbf{h}$ .

## Variogramme isotropique et anisotropique

- Précisons la notation  $\mathbf{h}_{ii'} \approx \mathbf{h}$ . On suppose alors deux cas :
  - Variogramme **isotropique** : il n'y a pas d'effet directionnel, ce qui compte c'est la distance entre les points, la fonction de variogramme dépend de  $\|\mathbf{h}\|$  uniquement :  $\gamma(\|\mathbf{h}\|)$ .
  - Variogramme **anisotropique** : il y a des effets directionnels, en plus de la distance  $\|\mathbf{h}\|$ , l'orientation de  $\mathbf{h}$  fait varier les valeurs de la fonction de variogramme :  $\gamma(\mathbf{h})$ .

### Cas isotropique :

- Notons  $r = \|\mathbf{h}\|$ , on a alors :

$$\hat{\gamma}(r) = \frac{1}{2N(r)} \sum_{i,i':\mathbf{h}_{ii'} \in [r-\Delta r, r+\Delta r]} (\mathcal{Z}(\mathbf{p}_i) - \mathcal{Z}(\mathbf{p}_{i'}))^2$$

où  $N(r)$  est le nb. de paires  $(\mathbf{p}_i, \mathbf{p}_{i'})$  tel que  $\mathbf{h}_{ii'} \in [r-\Delta r, r+\Delta r]$ .

## Variogramme isotropique et anisotropique (suite)

### Cas isotropique (suite) :

- En pratique, on **partitionne le domaine des dist. en des intervalles** de même amplitude  $\alpha$ . Avec  $K$  intervalles on a :
  - $r_k, k = 1, \dots, K$ ,
  - $r_k - r_{k-1} = \alpha, \forall k = 2, \dots, K$ ,
  - $\Delta r_k = \alpha/2, \forall k = 1, \dots, K$ .
- Le variogramme expérimental s'écrit alors,  $\forall k \in 1, \dots, K$  :

$$\hat{\gamma}(r_k) = \frac{1}{2N(r_k)} \sum_{i,i':\mathbf{h}_{ii'} \in [r_k-\alpha/2, r_k+\alpha/2]} (\mathcal{Z}(\mathbf{p}_i) - \mathcal{Z}(\mathbf{p}_{i'}))^2$$

- $\alpha$  est également appelé le **pas** et  $k$  le **nombre de pas**. Si  $\alpha$  est trop petit, on aura bcp. de variations, si  $\alpha$  est trop grand, on lisse trop et on aura des erreurs.

## Variogramme isotropique et anisotropique (suite)

- Dans le **cas anisotropique** :

- Aux **classes de distances**, s'ajoutent des **classes de directions**.
- Dans  $\mathbb{R}^2$ ,  $\mathbf{h}$  en fonction des coordonnées polaires  $(r, \theta)$  s'écrit  $r(\cos(\theta), \sin(\theta))$ . En pratique, comme précédemment, on prendra  $r$  et  $\theta$  appartenant resp. à plusieurs intervalles de même amplitude indicés par  $k$  et centrés en  $r_k$  et  $\theta_k$ ,  $\forall k = 1, \dots, K$  :
  - $[r_k - \alpha/2, r_k + \alpha/2]$ .
  - $[\theta_k - \beta/2, \theta_k + \beta/2]$ .
- Les directions  $\theta_k$  sont en général choisies de façon régulière et en nb. restreint. En pratique, un expert pourra aider à choisir le nb. et les différentes valeurs  $r_k$  et  $\theta_k$ . Les directions choisies en général sont (en degré) : 0, 45, 90, 135.
- Une anisotropie se traduit par des  $\hat{\gamma}$  dont les portées varient selon les directions. Souvent les directions de plus longue et de plus courte portée sont orthogonales.

## Propriétés des fonctions covariance et de variogramme

- Soit  $\lambda = (\lambda_1, \dots, \lambda_n)$  un vecteur de  $\mathbb{R}^n$ . Nous avons la propriété suivante de l'opérateur variance :

$$V \left\{ \sum_{i=1}^n \lambda_i Z(\mathbf{p}_i) \right\} = \sum_{i=1}^n \sum_{i'=1}^n \lambda_i \lambda_{i'} C \{ Z(\mathbf{p}_i), Z(\mathbf{p}_{i'}) \}$$

- Comme une variance est positive ou nulle nous avons nécessairement :

$$\sum_{i=1}^n \sum_{i'=1}^n \lambda_i \lambda_{i'} \overbrace{C(Z(\mathbf{p}_i), Z(\mathbf{p}_{i'}))}^{C(\mathbf{p}_i, \mathbf{p}_{i'})} \geq 0$$

- Une fonction  $C$  quelconque t.q.  $\sum_i \sum_{i'} \lambda_i \lambda_{i'} C(\mathbf{p}_i, \mathbf{p}_{i'}) \geq 0, \forall \lambda \in \mathbb{R}^n$ , est dite **semi-définie positive (s.d.p.)**.
- La fonction de covariance est donc (et doit forcément être) s.d.p.

## Estimation paramétrique d'un modèle de variogramme

- A partir du variogramme expérimental, il est nécessaire d'inférer un variogramme théorique représenté par un modèle paramétrique pour les raisons suivantes :

- Pour prédire avec les techniques de Krigeage, il est nécessaire de connaître les valeurs du variogrammes pour des distances quelconques ne faisant pas forcément partie des distances représentées à partir des données et du variogramme empirique.
- Le variogramme doit être une fonction conditionnellement définie négative (cf ci-après), il faut donc une fonction qui vérifie cette propriété. L'estimation consistera à déterminer une instance dans un ens. de fonctions vérifiant cette propriété.

## Propriétés des fcts covariance et de variogramme (suite)

- Dans le cas de la fonction de variogramme, nous avons une propriété équivalente mais qui nécessite l'introduction de la notion de **combinaison linéaire autorisée (CLA)** :

$$\lambda = (\lambda_1, \dots, \lambda_n) \text{ est une CLAssi } \sum_{i=1}^n \lambda_i = 0$$

- On montre que si  $Z$  est une IRF alors  $\sum_i \lambda_i Z(\mathbf{p}_i)$  possède un premier et second momentssi  $\lambda$  est une CLA.
- Plus précisément si  $Z$  est une IRF et si  $\lambda$  est une CLA alors :

$$E \left\{ \sum_{i=1}^n \lambda_i Z(\mathbf{p}_i) \right\} = 0$$

$$V \left\{ \sum_{i=1}^n \lambda_i Z(\mathbf{p}_i) \right\} = - \sum_{i=1}^n \sum_{i'=1}^n \lambda_i \lambda_{i'} \gamma(\mathbf{p}_i - \mathbf{p}_{i'})$$

## Propriétés des fcts covariance et de variogramme (suite)

- A nouveau du fait qu'une variance est positive ou nulle on a :

$$-\sum_{i=1}^n \sum_{i'=1}^n \lambda_i \lambda_{i'} \gamma(\mathbf{p}_i - \mathbf{p}_{i'}) \geq 0$$

- Une fonction  $\gamma$  qui vérifie  $\sum_{i=1}^n \sum_{i'=1}^n \lambda_i \lambda_{i'} \gamma(\mathbf{p}_i - \mathbf{p}_{i'}) \leq 0$  pour toute CLA  $\lambda$  est dite **conditionnellement définie négative** (c.d.n.). La fonction de variogramme est donc c.d.n.
- Nous avons en fait le résultat théorique suivant : une fonction  $\gamma$  définie sur  $\mathbb{R}^p$  est un variogramme ssi  $\gamma$  est c.d.n.
- De plus, on montre que la classe des fonctions c.d.n est plus grande que celle des fonctions s.d.p. Par conséquent, la classe des fonctions de variogramme est plus grande que celle des fonctions covariance.

## Estimation des paramètres

- Notons de façon générale  $\mathbf{c} \in \mathbb{R}^q$  le vecteur des paramètres et  $k$  le nb. de classes de distances (et d'orientations, le cas échéant) retenu.
- Estimation de  $\mathbf{c}$  par **MCO** :

$$\hat{\mathbf{c}}_{mco} = \arg \min_{\mathbf{c} \in \mathbb{R}^q} \sum_{k=1}^k (\hat{\gamma}(r_k) - \gamma(r_k; \mathbf{c}))^2$$

- Estimation de  $\mathbf{c}$  par **MCP (Moindres Carrés Pondérés)** :

$$\hat{\mathbf{c}}_{mcp} = \arg \min_{\mathbf{c} \in \mathbb{R}^q} \sum_{k=1}^k \frac{N(r_k)}{\gamma(r_k; \mathbf{c})^2} (\hat{\gamma}(r_k) - \gamma(r_k; \mathbf{c}))^2$$

- Si on suppose un modèle probabiliste, par ex.  $Z$  est une SRF Gaussienne (cf slide 431), alors on peut estimer  $\mathbf{c}$  par **MV** (maximum de vraisemblance) (cf [Chiles and Delfiner, 2012, Allard, 2012]).
- MCP donnent en général de meilleurs résultats que MCO et MV.

## Modèles paramétriques de variogramme

- Ex. de modèles paramétriques de variogrammes isotropiques.
- $r = \|\mathbf{h}\|$ ,  $a$  le palier,  $c_0$  la variance de l'effet pépite,  $c_0 + c_1$  la portée.
- Modèle isotrope **sphérique** avec effet pépite :

$$\gamma(r; \mathbf{c}) = \begin{cases} 0 & r = 0 \\ c_0 + c_1 \left( \frac{3r}{2a} - \frac{1}{2} \left( \frac{r}{a} \right)^3 \right) & 0 < r < a \\ c_0 + c_1 & r \geq a \end{cases}$$

- Modèle isotrope **exponentiel** avec effet pépite :

$$\gamma(r; \mathbf{c}) = c_0 + c_1 \left( 1 - \exp \left( -\frac{r}{a} \right) \right)$$

- Modèle isotrope **Gaussien** avec effet pépite :

$$\gamma(r; \mathbf{c}) = c_0 + c_1 \left( 1 - \exp \left( -\frac{r^2}{a^2} \right) \right)$$

## Validation d'un modèle de variogramme

- Pour valider un modèle théorique ou comparer plusieurs modèles théoriques, voici en bref qques méthodes :
  - ▶ Validation visuelle :
    - On compare graphiquement le variogramme expérimental et la courbe prédictive par le modèle estimé.
    - On vérifie si le modèle suit correctement la tendance des points et la forme générale (palier, portée, effet pépite).
  - ▶ Analyse des résidus :
    - On calcule l'erreur quadratique moyenne des ajustements :

$$Mse(\hat{\gamma}, \gamma(\cdot; \hat{\mathbf{c}})) = \frac{1}{k} \sum_{k=1}^k (\hat{\gamma}(r_k) - \gamma(r_k; \hat{\mathbf{c}}))^2$$

- On vérifie la répartition aléatoire des résidus autour de 0.
- ▶ Validation croisée (LOOCV) sur modèle de prédiction (Krigage) :
  - On retire un point de l'ens. et on apprend le modèle sans lui.
  - On estime la valeur de ce point à partir du modèle estimé.
  - On compare la valeur réelle et la valeur prédictive.
  - On itère sur tous les points et on moyennise l'erreur.

## \*\*Code\*\* : Variogramme expérimental, AirBnB

- Il existe plus. librairies de géostat. en Python. Nous utiliserons gstoools.
- Le code suivant permet de déterminer le variogramme expérimental de la variable log\_price de AirBnB.

```

1 # Import necessary libraries
2 import numpy as np
3 import pandas as pd
4 import geopandas as gpd
5 import gstoools as gs
6 import matplotlib.pyplot as plt
7
8 ## Variogram estimation of log_price ++++++
9 db = gpd.read_file("regression_db.geojson")
10 x = db['geometry'].x
11 y = db['geometry'].y
12 z = db["log_price"].values
13
14 # Estimate the variogram of the field and plot the result.
15 bin_center, gamma = gs.vario_estimate((x, y), z, sampling_size=1000, sampling_seed=42)
16
17 # plot the estimated variogram
18 plt.scatter(bin_center, gamma, color="k", label="data")

```

## \*\*Code\*\* : Variogramme expérimental, AirBnB

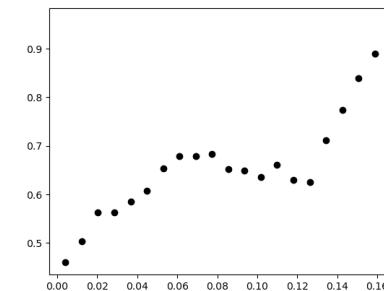
- Le code suivant permet de comparer plusieurs ajustements de modèles théoriques du variogramme.

```

1 # Define a set of models to test.
2 models = {
3     "Gaussian": gs.Gaussian,
4     "Exponential": gs.Exponential,
5     "Matern": gs.Matern,
6     "Stable": gs.Stable,
7     "Rational": gs.Rational,
8     "Circular": gs.Circular,
9     "Spherical": gs.Spherical,
10    "SuperSpherical": gs.SuperSpherical,
11    "JBessel": gs.JBessel,
12 }
13 scores = {}
14 ax = plt.gca()
15 # fit all models to the estimated variogram
16 for model in models:
17     fit_model = models[model](dim=2)
18     para, pcov, r2 = fit_model.fit_variogram(bin_center, gamma, return_r2=True)
19     fit_model.plot(x_max=0.2, ax=ax)
20     scores[model] = r2
21 # Create a ranking based on the score and determine the best models
22 ranking = sorted(scores.items(), key=lambda item: item[1], reverse=True)
23 print("RANKING by Pseudo-r2 score")
24 for i, (model, score) in enumerate(ranking, 1):
25     print(f"{i}>6}. {model}: {score:.5f}")

```

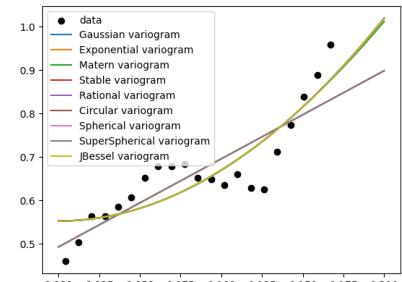
## \*\*Code\*\* : Variogramme expérimental, AirBnB (suite)



- Le variogramme est non borné ce qui correspond à une remise en cause des hypothèses de stationnarité.
- Essayons néanmoins de représenter ce graphique par un modèle théorique.

## \*\*Code\*\* : Modélisation du variogramme, AirBnB (suite)

RANKING by Pseudo-r2 score	
1.	Gaussian: 0.76302
2.	Stable: 0.76302
3.	Rational: 0.76285
4.	JBessel: 0.76275
5.	Matern: 0.76106
6.	Spherical: 0.73839
7.	Circular: 0.73839
8.	SuperSpherical: 0.73838
9.	Exponential: 0.73838



## Rappel du Sommaire

### 6 Eléments de géostatistique

- Les champs aléatoires
- Variogramme
- Les méthodes de Krigeage

## Régression linéaire et Krigeage Simple

- On cherche à **estimer la valeur non observée** d'une fct. aléatoire  $\mathcal{X}$  en un point  $\mathbf{p}_0 \in \mathbb{D}$ , **par une combinaison linéaire des valeurs observées** d'une fct. aléatoire  $\mathcal{Y}$  en les points  $\mathbf{p}_1, \dots, \mathbf{p}_n$ .
- On traitera en particulier le cas où les deux fcts représentent la même variable et on note  $\mathcal{X} = \mathcal{Y} = \mathcal{Z}$ . On parle alors de **Krigeage**.
- Le cas où  $\mathcal{X} \neq \mathcal{Y}$  est appelé **Co-Krigeage**. Du point de vue procédure, il n'y a pas de différence entre le Krigeage et le Co-Krigeage. La distinction s'opère en amont lors de l'estimation de la fct. de covariance ou de variogramme.
- Notons par  $\hat{z}_0$  la valeur estimée de  $\mathcal{Z}(\mathbf{p}_0)$  et par  $z_1, \dots, z_n$  les valeurs observées de  $\mathcal{Z}(\mathbf{p}_1), \dots, \mathcal{Z}(\mathbf{p}_n)$ . On a donc le modèle linéaire suivant :

$$\hat{z}_0 = \lambda_0 + \sum_{i=1}^n \lambda_i z_i$$

- Il s'agit donc de déterminer les poids  $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_n)$ .

## Régression linéaire et Krigeage Simple (suite)

- Raisonnons en théorie et prenons les v.a.r. associées. L'estimateur linéaire de  $\mathcal{Z}(\mathbf{p}_0)$  est dénoté par  $\widehat{\mathcal{Z}}(\mathbf{p}_0)$  et on a :

$$\widehat{\mathcal{Z}}(\mathbf{p}_0) = \lambda_0 + \sum_{i=1}^n \lambda_i \mathcal{Z}(\mathbf{p}_i)$$

- Supposons dans un premier temps que les **espérances**  $E\{\mathcal{Z}(\mathbf{p}_i)\} = \mu_i$  avec  $i = 0, 1, \dots, n$  sont **connues** et potentiellement différentes.
- L'espérance de la v.a.r. de l'erreur de  $\mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0)$  s'écrit :

$$\begin{aligned} E\{\mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0)\} &= E\{\mathcal{Z}(\mathbf{p}_0)\} - E\{\widehat{\mathcal{Z}}(\mathbf{p}_0)\} \\ &= E\{\mathcal{Z}(\mathbf{p}_0)\} - \lambda_0 - \sum_{i=1}^n \lambda_i E\{\mathcal{Z}(\mathbf{p}_i)\} \\ &= \mu_0 - \lambda_0 - \sum_{i=1}^n \lambda_i \mu_i \end{aligned}$$

## Régression linéaire et Krigeage Simple (suite)

- On souhaite avoir un **estimateur sans biais**, par conséquent :

$$\begin{aligned} E\{\mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0)\} &= 0 \Leftrightarrow \mu_0 - \lambda_0 - \sum_{i=1}^n \lambda_i \mu_i = 0 \\ &\Leftrightarrow \lambda_0 = \mu_0 - \sum_{i=1}^n \lambda_i \mu_i \end{aligned}$$

- En injectant cette relation dans la définition de l'estimateur, il vient :

$$\widehat{\mathcal{Z}}(\mathbf{p}_0) = \mu_0 + \sum_{i=1}^n \lambda_i (\mathcal{Z}(\mathbf{p}_i) - \mu_i)$$

## Régression linéaire et Krigeage Simple (suite)

- L'estimateur linéaire sans biais  $\widehat{\mathcal{Z}}(\mathbf{p}_0)$  est en lien direct avec l'estimateur linéaire sans biais de l'erreur  $\mathcal{Z}(\mathbf{p}_0) - \mu_0$  à partir des erreurs stochastiques  $\{\mathcal{Z}(\mathbf{p}_i) - \mu_i\}_{i=1,\dots,n}$  puisque :

$$(\widehat{\mathcal{Z}}(\mathbf{p}_0) - \mu_0) = (\widehat{\mathcal{Z}}(\mathbf{p}_0) - \mu_0) = \sum_{i=1}^n \lambda_i (\mathcal{Z}(\mathbf{p}_i) - \mu_i)$$

- Revenons sur notre v.a.r. de l'erreur  $\mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0)$ , on a :

$$\begin{aligned} \mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0) &= (\mathcal{Z}(\mathbf{p}_0) - \mu_0) - (\widehat{\mathcal{Z}}(\mathbf{p}_0) - \mu_0) \\ &= (\mathcal{Z}(\mathbf{p}_0) - \mu_0) - \sum_{i=1}^n \lambda_i (\mathcal{Z}(\mathbf{p}_i) - \mu_i) \\ &= \sum_{i=0}^n \color{red}{a_i} (\mathcal{Z}(\mathbf{p}_i) - \mu_i) \end{aligned}$$

où  $a_0 = 1$  et  $a_i = -\lambda_i$ ,  $\forall i = 1, \dots, n$ .

## Régression linéaire et Krigeage Simple (suite)

- Introduisons les notations suivantes :

- ▶  $\mathbf{K}$  la matrice des var-cov. pour chq. paire  $\{(\mathcal{Z}(\mathbf{p}_i), \mathcal{Z}(\mathbf{p}_i))\}_{i,i'=1,\dots,n}$  :

$$\mathbf{K} = (C(\mathbf{p}_i, \mathbf{p}_{i'}))_{i,i'=1,\dots,n}$$

- ▶  $\mathbf{k}$  le vecteur des var-cov. de  $\mathcal{Z}(\mathbf{p}_0)$  avec chq. v.a.r. de  $\{\mathcal{Z}(\mathbf{p}_i)\}_{i=1,\dots,n}$  :

$$\mathbf{k} = (C(\mathbf{p}_i, \mathbf{p}_0))_{i=1,\dots,n}$$

- Le système normal du KS s'écrit alors comme suit :

$$\mathbf{K}\boldsymbol{\lambda} = \mathbf{k}$$

- La **solution analytique**  $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_n)$  est donnée par :

$$\boldsymbol{\lambda} = \mathbf{K}^{-1}\mathbf{k}$$

## Régression linéaire et Krigeage Simple (suite)

- On veut l'erreur  $\mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0)$  de **variance minimale**. On a :

$$V \left\{ \mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0) \right\} = \sum_{i=0}^n \sum_{i'=0}^n a_i a_{i'} C(\mathbf{p}_i, \mathbf{p}_{i'})$$

- On cherche les  $\{a_i\}_{i=0,\dots,n}$  qui minimisent la variance : pb. d'optim. convexe non contraint dont les CNPO sont,  $\forall i = 1, \dots, n$  :

$$\begin{aligned} \frac{\partial V}{\partial a_i}(\cdot) = 0 &\Leftrightarrow \sum_{i'=0}^n a_{i'} C(\mathbf{p}_i, \mathbf{p}_{i'}) = 0 \\ &\Leftrightarrow a_0 C(\mathbf{p}_i, \mathbf{p}_0) + \sum_{i'=1}^n a_{i'} C(\mathbf{p}_i, \mathbf{p}_{i'}) = 0 \\ &\Leftrightarrow \sum_{i'=1}^n \lambda_{i'} C(\mathbf{p}_i, \mathbf{p}_{i'}) = C(\mathbf{p}_i, \mathbf{p}_0) \quad (\text{car } a_0 = 1 \text{ et } a_i = -\lambda_i) \end{aligned}$$

On a un ens. de n équations linéaires appelé **système normal du KS**.

## Estimateur de Krigeage Simple et cas particulier d'une SRF

- L'estimateur du Krigeage Simple de  $\mathcal{Z}(\mathbf{p}_0)$  est donc donné par :

$$\begin{aligned} \widehat{\mathcal{Z}}(\mathbf{p}_0) &= \mu_0 + \sum_{i=1}^n \lambda_i (\mathcal{Z}(\mathbf{p}_i) - \mu_i) \\ &= \mu_0 + [\underbrace{\mathbf{K}^{-1}\mathbf{k}}_{\boldsymbol{\lambda}}]^\top (\mathbf{Z} - \boldsymbol{\mu}) \end{aligned}$$

où  $\mathbf{Z} = (\mathcal{Z}(\mathbf{p}_1), \dots, \mathcal{Z}(\mathbf{p}_n))$  et  $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)$ .

- L'estimateur précédent est défini dans un cadre assez général où on a supposé que les moyennes pouvaient varier selon les points. Dans le **cas d'une SRF** la moyenne de  $\mathcal{Z}$  est constante (cf slide 425), on a  $\mu_0 = \mu_1 = \dots = \mu_n = \mu$ , ce qui implique :

$$\widehat{\mathcal{Z}}(\mathbf{p}_0) = \mu + [\mathbf{K}^{-1}\mathbf{k}]^\top (\mathbf{Z} - \mu \mathbf{1}_n)$$

où  $\mathbf{1}_n$  est le vecteur de taille n rempli de 1.

## Propriétés de l'estimateur de Krigeage Simple (KS)

- Si la moyenne et la fct. de covariance sont connues alors l'**estimateur linéaire sans biais de variance minimale** est celui de KS.
- La **variance** de l'estimateur de KS de l'erreur vaut [Journel, 1989] :

$$\begin{aligned}\sigma_{KS}^2 &= V \left\{ Z(\mathbf{p}_0) - \widehat{Z}(\mathbf{p}_0) \right\} \\ &= C \{ \mathbf{p}_0, \mathbf{p}_0 \} - \sum_{i=1}^n \lambda_i C \{ \mathbf{p}_i, \mathbf{p}_0 \} \\ &= C(\mathbf{0}_p) - \sum_{i=1}^n \lambda_i C(\mathbf{p}_i - \mathbf{p}_0)\end{aligned}$$

- Le KS est un **interpolateur exacte** : si on prend  $\mathbf{p}_0 = \mathbf{p}_i$  un élément de l'ens. des points d'observation de la fonction aléatoire alors  $\widehat{Z}(\mathbf{p}_i) = Z(\mathbf{p}_i)$  (ou  $\widehat{z}_i = z_i$ ) (*exactitude property [Journel, 1989]*).
- Si de plus  $Z$  est une **fonction aléatoire Gaussienne** (cf slide 431) alors on a la propriété :  $\widehat{Z}(\mathbf{p}_0) = E \{ Z(\mathbf{p}_0) | Z(\mathbf{p}_1), \dots, Z(\mathbf{p}_n) \}$ .

## Krigeage Ordinaire (suite)

- Comme  $\mu$  est inconnue, pour satisfaire à la condition précédente "sans biais"  $-\lambda_0 + \mu(1 - \sum_{i=1}^n \lambda_i) = 0$ , on impose les contraintes :  $\lambda_0 = 0$  et  $\sum_{i=1}^n \lambda_i = 1$ .
- On a donc l'estimateur linéaire sans biais de  $Z(\mathbf{p}_0)$  suivant :

$$\widehat{Z}(\mathbf{p}_0) = \sum_{i=1}^n \lambda_i Z(\mathbf{p}_i) \quad \text{s.l.c. } \sum_{i=1}^n \lambda_i = 1$$

- On veut l'erreur  $Z(\mathbf{p}_0) - \widehat{Z}(\mathbf{p}_0)$  de **variance minimale**. On a :

$$\begin{aligned}V \left\{ Z(\mathbf{p}_0) - \widehat{Z}(\mathbf{p}_0) \right\} &= \sum_{i=0}^n \sum_{i'=0}^n a_i a_{i'} C(\mathbf{p}_i, \mathbf{p}_{i'}) \\ &= \sum_{i=0}^n \sum_{i'=0}^n a_i a_{i'} C(\mathbf{p}_i - \mathbf{p}_{i'}) \quad (\text{car SRF})\end{aligned}$$

où  $a_0 = 1$  et  $a_i = -\lambda_i$ ,  $\forall i = 1, \dots, n$  (cf slide 457).

## Krigeage Ordinaire

- Nous nous plaçons dans le contexte où  $Z$  est une **SRF**, que sa moyenne  $\mu$  (**constante**) est **inconnue** et nous souhaitons définir un estimateur linéaire sans biais de variance minimale de  $Z$  en un point  $\mathbf{p}_0 \in \mathbb{D}$  non observé à partir des valeurs observées en  $\mathbf{p}_1, \dots, \mathbf{p}_n$ .
- L'estimateur linéaire est de la forme :

$$\widehat{Z}(\mathbf{p}_0) = \lambda_0 + \sum_{i=1}^n \lambda_i Z(\mathbf{p}_i)$$

- Pour un estimateur linéaire sans biais on a :

$$\begin{aligned}E \left\{ Z(\mathbf{p}_0) - \widehat{Z}(\mathbf{p}_0) \right\} &= 0 \Leftrightarrow \mu - \lambda_0 - \sum_{i=1}^n \lambda_i \mu = 0 \\ &\Leftrightarrow -\lambda_0 + \mu \left( 1 - \sum_{i=1}^n \lambda_i \right) = 0\end{aligned}$$

## Krigeage Ordinaire (suite)

- La variance de l'erreur en fonction de  $\lambda$  s'écrit comme suit :

$$\begin{aligned}V \left\{ Z(\mathbf{p}_0) - \widehat{Z}(\mathbf{p}_0) \right\} &= C(\mathbf{0}_p) - 2 \sum_{i=1}^n \lambda_i C(\mathbf{p}_i - \mathbf{p}_0) \\ &\quad + \sum_{i=1}^n \sum_{i'=1}^n \lambda_i \lambda_{i'} C(\mathbf{p}_i - \mathbf{p}_{i'})\end{aligned}$$

- On veut l'erreur  $Z(\mathbf{p}_0) - \widehat{Z}(\mathbf{p}_0)$  de **variance minimale** : pb. de minimisation convexe sous la contrainte linéaire  $\sum_{i=1}^n \lambda_i = 1$ .
- Soit  $\text{Lag}(\cdot)$  et  $\nu \in \mathbb{R}$  la fonction et le multiplicateur de Lagrange respectivement. Les CNPO (théorème de Lagrange) donnent :

$$\begin{cases} \frac{\partial \text{Lag}}{\partial \lambda_i}(\cdot) = 0, \forall i \\ \frac{\partial \text{Lag}}{\partial \nu}(\cdot) = 0 \end{cases} \Rightarrow \begin{cases} \sum_{i'=1}^n \lambda_{i'} C(\mathbf{p}_i - \mathbf{p}_{i'}) + \nu = C(\mathbf{p}_i - \mathbf{p}_0), \forall i \\ \sum_{i=1}^n \lambda_i = 1 \end{cases}$$

Cet ens. de  $n+1$  éq. linéaires est appelé **système normal du KO**.

## Propriétés de l'estimateur de Krigeage Ordinaire (KO)

- Si la fonction de covariance est connue mais la **moyenne est inconnue**, et si les combinaisons linéaires sont telles que  $\sum_{i=1}^n \lambda_i = 1$  alors l'estimateur de KO est :
  - l'estimateur linéaire sans biais de variance minimale.
  - un **interpolateur exacte** (*exactitude property [Journel, 1989]*)
- La **variance** de l'estimateur de KO de l'erreur vaut [Journel, 1989] :

$$\begin{aligned}\sigma_{KO}^2 &= V\left\{\mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0)\right\} \\ &= C(\mathbf{0}_p) - \sum_{i=1}^n \lambda_i C(\mathbf{p}_i - \mathbf{p}_0) - \nu\end{aligned}$$

## Krigeage Universel (suite)

- Pour un estimateur  $\widehat{\mathcal{Z}}(\mathbf{p}_0)$  sans biais,  $E\left\{\mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0)\right\} = 0$ . Les hypothèses précédentes impliquent les relations suivantes :

$$E\{\mathcal{Z}(\mathbf{p}_0)\} = E\left\{\widehat{\mathcal{Z}}(\mathbf{p}_0)\right\} = \mu(\mathbf{p}_0) = \sum_{l=0}^1 a_l f_l(\mathbf{p}_0)$$

- Considérons l'estimateur linéaire suivant :

$$\widehat{\mathcal{Z}}(\mathbf{p}_0) = \sum_{i=1}^n \lambda_i \mathcal{Z}(\mathbf{p}_i)$$

- Si on injecte ce modèle linéaire dans l'hypothèse "sans biais", il vient :

$$\begin{aligned}E\left\{\mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0)\right\} = 0 &\Leftrightarrow \mu(\mathbf{p}_0) - \sum_{i=1}^n \lambda_i \mu(\mathbf{p}_i) = 0 \\ &\Leftrightarrow \sum_{l=0}^1 a_l \left( f_l(\mathbf{p}_0) - \sum_{i=1}^n \lambda_i f_l(\mathbf{p}_i) \right) = 0\end{aligned}$$

## Krigeage Universel

- Nous considérons tjs que  $\mathcal{Z}$  est une **SRF** de moyenne inconnue mais nous supposons le cas plus général où cette dernière est une **fonction**  $\mu$  (non constante) définie sur  $\mathbb{D} \subset \mathbb{R}^p$ . On fait l'hypothèse que  $\mu$  est différentiable en tout  $\mathbf{p} \in \mathbb{P}$  et qu'elle prend la forme suivante :

$$\mu(\mathbf{p}) = \sum_{l=0}^1 a_l f_l(\mathbf{p})$$

où les  $\{f_l\}_{l=0,\dots,1}$  sont des fonctions différentiables définies sur  $\mathbb{D}$  qui sont connues mais les  $\{a_l\}_{l=0,\dots,1}$  sont des coefficients réels inconnus et qu'il faut donc estimer.

- On prendra par convention  $f_0(\mathbf{p}) = 1, \forall \mathbf{p} \in \mathbb{D}$  (fonction constante). Ainsi, le cas  $l = 0$  correspond au KO.
- Le Krigeage Universel est aussi connu sous les noms de Krigeage avec tendance externe ou Régression-Krigeage (à qques nuances près).

## Krigeage Universel (suite)

- Les  $\{a_l\}_l$  étant des coefficients inconnus, pour avoir la propriété "sans biais" on impose les  $l + 1$  contraintes suivantes :

$$\sum_{i=1}^n \lambda_i f_l(\mathbf{p}_i) = f_l(\mathbf{p}_0), \quad \forall l = 0, \dots, 1$$

- On cherche ensuite à minimiser la variance de l'erreur  $\mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0)$ . On a alors un pb. de minimisation convexe sous les contraintes linéaires ci-dessus.
- Notons par  $\nu_l, l = 0, \dots, 1$ , les multiplicateurs de Lagrange de ces contraintes. Le théorème de Lagrange permet comme pour le KO de déterminer un ens. de contraintes caractérisant la solution unique. Cet ensemble est constitué de  $n + l + 1$  équations linéaires.

## Krigeage Universel (suite)

- Le **système normal de KU** est donné par [Journel, 1989] :

$$\begin{cases} \sum_{i'=1}^n \lambda_{i'} C(\mathbf{p}_i - \mathbf{p}_{i'}) + \sum_{l=0}^1 \nu_l f_l(\mathbf{p}_i) = C(\mathbf{p}_i - \mathbf{p}_0) & \forall i = 1, \dots, n \\ \sum_{i'=1}^n \lambda_{i'} f_l(\mathbf{p}_{i'}) = f_l(\mathbf{p}_0) & \forall l = 0, \dots, 1 \end{cases}$$

- La **variance** de l'estimateur de KU de l'erreur vaut [Journel, 1989] :

$$\begin{aligned} \sigma_{KU}^2 &= V \left\{ \mathcal{Z}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0) \right\} \\ &= E \left\{ (\widehat{\mathcal{Z}}(\mathbf{p}_0) - \widehat{\mathcal{Z}}(\mathbf{p}_0))^2 \right\} \\ &= C(\mathbf{0}_p) - \sum_{i=1}^n \lambda_i C(\mathbf{p}_i - \mathbf{p}_0) - \sum_{l=0}^1 \nu_l f_l(\mathbf{p}_i) \end{aligned}$$

## Krigeage Universel (suite)

- Les paramètres du KU se déterminent en résolvant le système KU.
- La formulation explicite de la prédiction du KU est donnée par :

$$\widehat{\mathcal{Z}}(\mathbf{p}_0) = [\mathbf{f}(\mathbf{p}_0)]^\top \mathbf{a} + [\mathbf{K}^{-1} \mathbf{k}]^\top (\mathbf{Z} - \mathbf{F}\mathbf{a}), \text{ où :}$$

- $\mathbf{F} = ([\mathbf{f}(\mathbf{p}_1)]^\top, \dots, [\mathbf{f}(\mathbf{p}_n)]^\top) \in \mathbb{R}^{n \times 1}$  est la mat. des variables explicatives en chaque localisation.
- Le vecteur  $\mathbf{a}$  de la régression modélisant la moyenne sous forme de tendance externe est donné par l'estimateur des moindres carrés généralisés suivant :

$$\mathbf{a} = (\mathbf{F}^\top \mathbf{K}^{-1} \mathbf{F})^{-1} \mathbf{F}^\top \mathbf{K}^{-1} \mathbf{Z}$$

- Nous reviendrons sur le modèle KU ultérieurement pour identifier ses limites et présenter un modèle plus général fondé sur les réseaux de neurones appelé DeepKriging [Chen et al., 2024].

## Krigeage Universel (suite)

- Le Krigeage Universel revient à supposer le modèle suivant :

$$\mathcal{Z}(\mathbf{p}) = [\mathbf{f}(\mathbf{p})]^\top \mathbf{a} + \epsilon(\mathbf{p}), \text{ où :}$$

- $\mathbf{f} : \mathbb{D} \subset \mathbb{R}^p \rightarrow \mathbb{R}^1$  est une fonction vectorielle pouvant encoder des variables explicatives exogènes à  $\mathcal{Z}$  supposées différentiables sur  $\mathbb{D}$ .
- $\epsilon$  est une SRF avec  $E\{\epsilon(\mathbf{p})\} = 0, \forall \mathbf{p} \in \mathbb{D}$  et de covariance telle que :

$$\begin{aligned} C\{\epsilon(\mathbf{p} + \mathbf{h}), \epsilon(\mathbf{p})\} &= E\{\epsilon(\mathbf{p} + \mathbf{h})\epsilon(\mathbf{p})\} \\ &= C\{\mathcal{Z}(\mathbf{p} + \mathbf{h}), \mathcal{Z}(\mathbf{p})\} \\ &= C(\mathbf{h}) \end{aligned}$$

- On pourrait ajouter un effet pépite avec le modèle étendu suivant :

$$\mathcal{Z}(\mathbf{p}) = [\mathbf{f}(\mathbf{p})]^\top \mathbf{a} + \epsilon(\mathbf{p}) + \eta(\mathbf{p})$$

où  $\eta(\mathbf{p})$  est t.q.  $E\{\eta(\mathbf{p})\} = 0$  et  $C\{\eta(\mathbf{p}), \eta(\mathbf{p}')\} = \delta_{\mathbf{p}\mathbf{p}'}\sigma^2(\mathbf{p})$ ,  $\forall \mathbf{p}, \mathbf{p}' \in \mathbb{D}$ . Mais nous ne le ferons pas pour alléger le propos.

## \*\*Code\*\* : Krigeage Universel, AirBnB

- Nous commençons par préparer les données comme suit.

```

1 db["longitude"] = x
2 db["latitude"] = y
3 variable_names = [
4     "accommodates", # Number of people it accommodates
5     "bathrooms", # Number of bathrooms
6     "bedrooms", # Number of bedrooms
7     "beds", # Number of beds
8     # Below are binary variables, 1 True, 0 False
9     "rt_Private_room", # Room type: private room
10    "rt_Shared_room", # Room type: shared room
11    "pg_Condominium", # Property group: condo
12    "pg_House", # Property group: house
13    "pg_Other", # Property group: other
14    "pg_Townhouse", # Property group: townhouse
15    # Below are spatial variables
16    "longitude",
17    "latitude"
18]
19 X = db[variable_names]
20 from sklearn.model_selection import train_test_split
21 # Split the data into training and test sets
22 X_train, X_test, z_train, z_test = train_test_split(X, z, test_size=0.2, random_state=42)
23 coords_train = (X_train['longitude'].values, X_train['latitude'].values)
24 coords_test = (X_test['longitude'].values, X_test['latitude'].values)
25 X_train.drop(columns=['longitude', 'latitude'])
26 X_test.drop(columns=['longitude', 'latitude'])

```

## \*\*Code\*\* : Krigeage Universel, AirBnB (suite)

- Le variogramme expérimental est ajusté par un modèle Gaussien.
- Nous apprenons les coefficients du KU sur les données d'entraînement puis testons sur les données test.

```

1 # Select the Gaussian model and assume there is an external trend -> KU
2 model = gs.Gaussian(dim=2)
3 # Fit the Gaussian model using training data
4 model.fit_variogram(bin_center, gamma, return_r2=True)
5
6 # Learn the linear trend using LinearRegression on training datasets
7 from sklearn.linear_model import LinearRegression # Linear model
8 lr = LinearRegression()
9 lr.fit(X_train, z_train) # Fit the model to the training data
10
11 # Estimate the KU
12 ku = gs.krige.ExtDrift(model=model, cond_pos=coords_train, cond_val=z_train, ext_drift
13 =lr.predict(X_train))
14
15 # Predict the KU model on test set
16 ku_pred = ku(pos=coords_test, ext_drift=lr.predict(X_test))
17
18 from sklearn.metrics import mean_squared_error # For calculating RMSE
19 # Print the Root Mean Squared Error (RMSE) for Linear Regression
20 print('KU RMSE:', np.sqrt(mean_squared_error(z_test, ku_pred[0])))

```

KU RMSE: 0.43742089806302925

## Rappel du Sommaire

### 7 Méthodes de DL et extensions spatiales

- Introduction aux réseaux de neurones
- Le perceptron : l'élément de base d'un NN
- MLP : le perceptron multicouche
- Le modèle DeepKriging

## Rappel du Sommaire

### 1 Introduction et motivations générales

### 2 Concepts de base et outils Python associés

### 3 Analyse de la dépendance spatiale

### 4 Méthodes de régression spatiale

### 5 Méthodes d'ensemble en ML et extensions spatiales

### 6 Eléments de géostatistique

### 7 Méthodes de DL et extensions spatiales

## Introduction

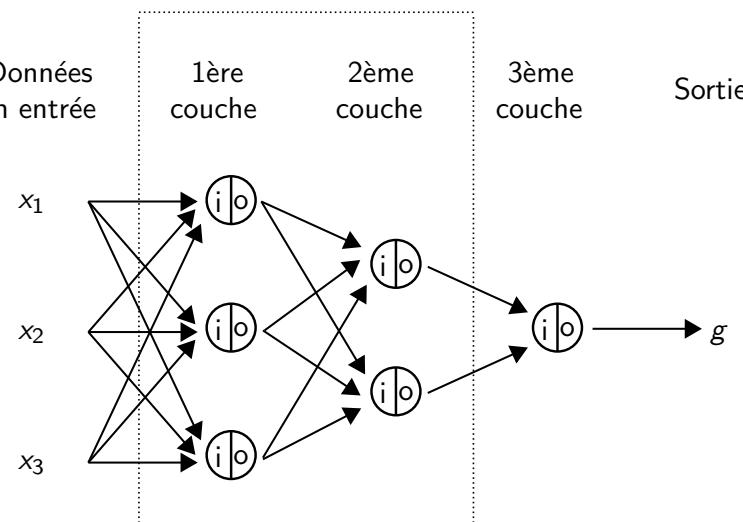
- Les Réseaux de Neurones Artificiels/*Artificial Neural Networks/NN* sont des modèles qui s'inspirent du fonctionnement du cerveau et de travaux provenant des sciences cognitives et des neurosciences.
- Le cerveau peut être vu comme un système de traitement d'informations qui possède des capacités extraordinaires dépassant bien évidemment celles d'un ordinateur.
- Le cerveau effectue par exemple des tâches de reconnaissance de formes visuelles ou sonores, ou est capable d'apprendre à partir d'exemples ou d'un enseignant.
- Ces quelques tâches que nous venons de citer sont les problèmes que traite l'intelligence artificielle. Puisque le cerveau est capable de résoudre ces tâches alors il est tentant de modéliser le fonctionnement du cerveau, de le programmer et de demander à une machine de reproduire les capacités du cerveau.

## Introduction (suite)

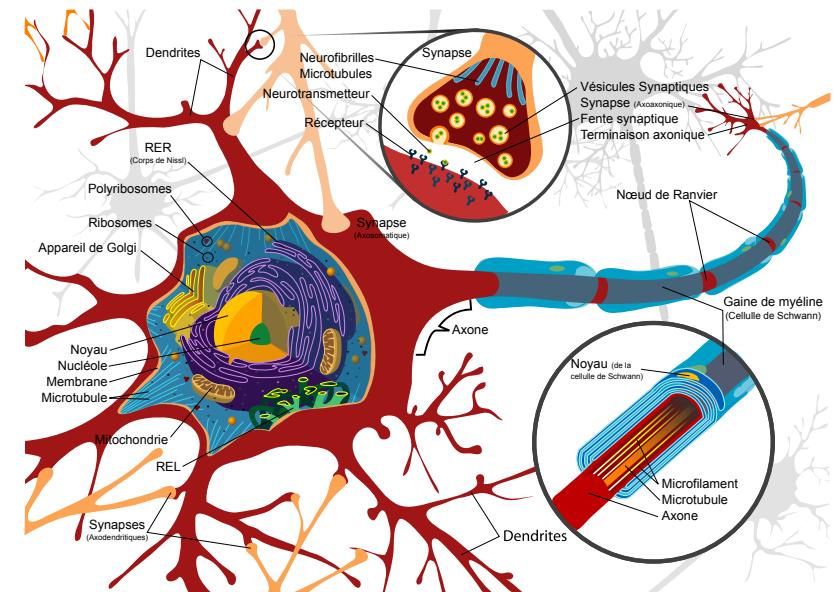
- Le cerveau est à bien des égards différent d'un ordinateur ! Mais si l'on devait appréhender le cerveau comme un système de traitement d'informations, on voit qu'un ordinateur à un (ou quelques) processeur alors que le cerveau est composé d'un très large nb. de "processeurs" que sont les **neurones** (le cerveau humain en compte environ  $10^{11}$ ).
- Toutefois, le neurone en tant qu'unité de traitement est plus "simple" qu'un processeur. Ce qui fait la particularité du cerveau est sa capacité à **traiter de manière parallèle l'information**. Ceci est possible par la très grande connectivité entre neurones reliés entre eux par les **synapses**.
- Les synapses sont les éléments du cerveau qui permettent la **transmission (en parallèle) d'information** entre neurones. On compte en moyenne 10000 synapses par neurone !

## Schéma d'un réseau de neurones artificiel (NN)

Couches cachées



## Schéma d'un neurone biologique



## Rappel du Sommaire

### 7 Méthodes de DL et extensions spatiales

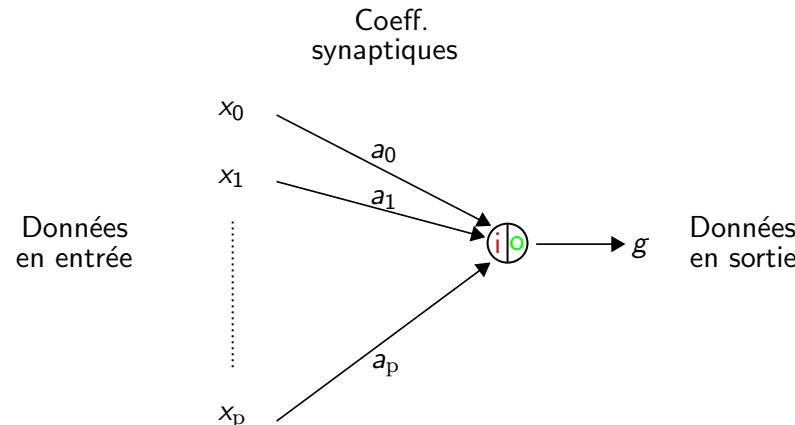
- Introduction aux réseaux de neurones
- Le perceptron : l'élément de base d'un NN**
- MLP : le perceptron multicouche
- Le modèle DeepKriging

## Le perceptron + Notations

- Le **perceptron** est l'élément de base d'un NN : il reçoit des données en entrée provenant de signaux internes (d'autres perceptrons) ou externes (données du problème) et donne en sortie un signal qui peut être communiqué en interne (à d'autres perceptrons) ou en externe (prédiction du perceptron). Par abus de langage nous utiliserons également neurone ou parfois noeud d'un NN.
- Soit un vecteur  $(x_1, \dots, x_p)$  appartenant à  $\mathbb{X} \subseteq \mathbb{R}^p$  qui est le signal d'entrée transmis à un perceptron.
- On ajoute en général une constante  $x_0 = 1$  appelée **biais** et on utilise alors les notations  $x_{\setminus 0} = (x_1, \dots, x_p)$  et  $\mathbf{x} = (1, x_1, \dots, x_p) \in \mathbb{R}^{p+1}$ .
- A chaque var.  $X^j$  on lui associe un poids  $a_j$ . On a ainsi un vecteur  $\mathbf{a} = (a_0, a_1, \dots, a_p)$  que l'on appelle les **coefficients synaptiques**.
- Le signal **post-synaptique** est défini de la façon suivante :

$$s(\mathbf{x}) = a_0 + \sum_{i=1}^p a_i x_i$$

## Le perceptron : illustration

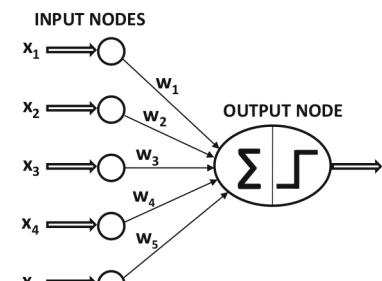


- Dans la suite on notera  $\mathbf{x} = (1, x_1, \dots, x_p)$  et  $\mathbf{a} = (a_0, a_1, \dots, a_p)$  si bien que le signal post-synaptique s'écrit :

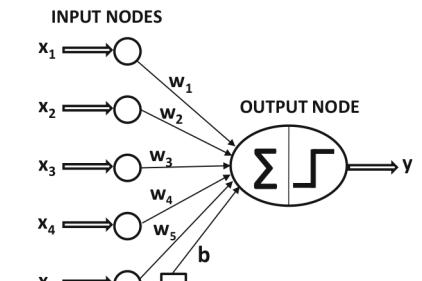
$$s(\mathbf{x}) = i = \mathbf{a}^\top \mathbf{x}$$

## Schéma d'un perceptron sans et avec biais

Extrait de [Aggarwal, 2018].



(a) Perceptron without bias



(b) Perceptron with bias

## Le perceptron : signal post-synaptique et fct. d'activation

- Le signal post-synaptique est ainsi une combinaison linéaire des données en entrée :  $s(\mathbf{x}) = \mathbf{a}^\top \mathbf{x}$ .
- Le neurone qui reçoit ce signal le transforme par le biais d'une **fonction d'activation**,  $h(s(\mathbf{x})) = o$  et qui peut prendre plusieurs formes :

► La fonction identité :

$$h(s(\mathbf{x})) = h(i) = o = \mathbf{a}^\top \mathbf{x}$$

► Une fonction à seuil comme la fonction d'Heaviside :

$$h(s(\mathbf{x})) = o = \begin{cases} 1 & \text{si } \mathbf{a}^\top \mathbf{x} > 0 \\ 0 & \text{sinon} \end{cases}$$

► La fonction sigmoïde (fonction inverse de logit) :

$$h(s(\mathbf{x})) = o = \frac{1}{1 + \exp(-\mathbf{a}^\top \mathbf{x})}$$

$$\text{Rq : } \frac{1}{1 + \exp(-\mathbf{a}^\top \mathbf{x})} = \frac{\exp(\mathbf{a}^\top \mathbf{x})}{1 + \exp(\mathbf{a}^\top \mathbf{x})}.$$

## Le perceptron : liens avec les modèles linéaires

- Nous voyons que le perceptron est une forme d'allégorie permettant de retrouver les modèles linéaires vus précédemment !
  - ▶ Le perceptron associé à la fct d'activation identité revient à un modèle linéaire pour le problème de régression.
  - ▶ Le perceptron associé à des fcts à seuil détermine des frontières de décision linéaires (hyperplans) dans  $\mathbb{X}$ , ce qui est équivalent à des modèles linéaires en catégo. telle que l'Analyse Factorielle Discriminante.
  - ▶ Le perceptron associé à la fct sigmoïde donne en sortie des valeurs entre 0 et 1 ce qui permet une interprétation en terme de probabilité pour le pb. de catégo. binaire. On remarquera par ailleurs (rég log. binomiale) :

$$P(C_1|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{a}^\top \mathbf{x})} \Rightarrow \underbrace{1 - P(C_1|\mathbf{x})}_{P(C_2|\mathbf{x})} = \frac{\exp(-\mathbf{a}^\top \mathbf{x})}{1 + \exp(-\mathbf{a}^\top \mathbf{x})}$$

- Un perceptron permet donc de modéliser le pb. de rég. et de catégo. binaire. Dans ce dernier cas, quand est-il du pb. multiclasse ?
- On utilise en parallèle q perceptrons !

## Problème multiclasse : q perceptrons en parallèle (suite)

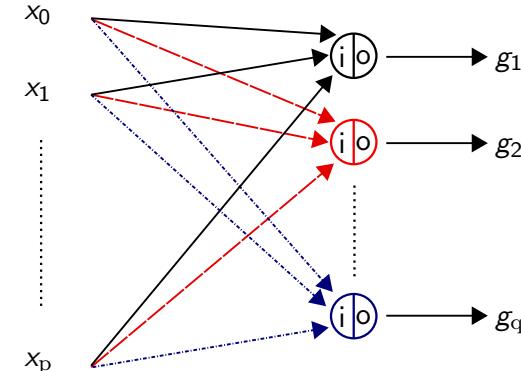
- Dans ce cas, nous avons q systèmes de coefficients synaptiques ce qui nous conduit à définir la matrice  $\mathbf{A}$  de taille  $((p + 1) \times q)$  dont les colonnes  $\mathbf{a}_l$  sont les coefficients synaptiques du  $l$ -ème perceptron.
- Chaque perceptron reçoit un message post-synaptique défini par  $s_l(\mathbf{x}) = \mathbf{a}_l^\top \mathbf{x}$  et chaque perceptron est indépendant des autres.
- Globalement, on obtient une fonction à valeurs vectorielles  $\mathbf{s} : \mathbb{X} \rightarrow \mathbb{R}^q$  qui, étant donné un signal en entrée  $\mathbf{x}$ , calcule les signaux post-synaptiques de chaque perceptron  $l$  de la façon suivante :

$$\mathbf{s}(\mathbf{x}) = \mathbf{A}^\top \mathbf{x} \text{ avec } s_l(\mathbf{x}) = \mathbf{a}_l^\top \mathbf{x}$$

- Les neurones appliquent ensuite la fonction d'activation et on note :
- $\mathbf{g}(\mathbf{x}) = h(\mathbf{A}^\top \mathbf{x})$  (par abus de notations) avec :  $g_l(\mathbf{x}) = h(\mathbf{a}_l^\top \mathbf{x})$
- La **règle de décision** est alors la suivante :
- $f(\mathbf{x}) = C_l \Leftrightarrow \forall l' \neq l : g_l(\mathbf{x}) \geq g_{l'}(\mathbf{x})$

## Problème multiclasse : q perceptrons en parallèle

Données  
en entrée



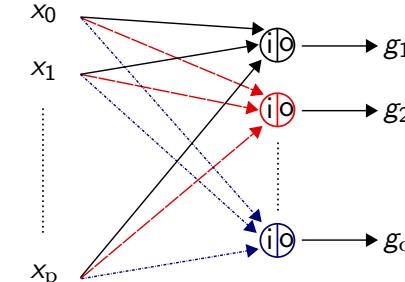
Données  
en sortie

## q perceptrons en parallèle et régression logistique

- La régression logistique polytomique est équivalente à un NN avec q perceptrons en parallèle et des fonctions d'activation softmax :

$$g_k(\mathbf{x}) = h(\mathbf{a}_k^\top \mathbf{x}) = \frac{\exp(\mathbf{a}_k^\top \mathbf{x})}{\sum_{l=1}^q \exp(\mathbf{a}_l^\top \mathbf{x})}$$

Données  
en entrée



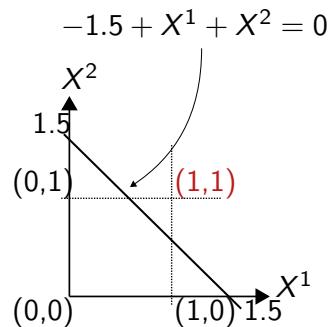
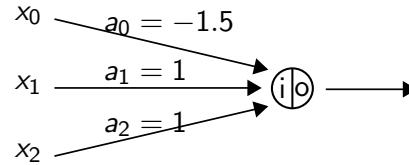
Données  
en sortie

## Le perceptron : modélisation du “AND”

Signal post-synaptique et fonction d'Heaviside :

$X^1$	$X^2$	AND
1	1	1
1	0	0
0	1	0
0	0	0

$$\begin{aligned} a_0 + a_1 X^1 + a_2 X^2 &\leq 0 \\ \left\{ \begin{array}{l} a_0 + a_1 + a_2 > 0 \\ a_0 + a_1 \leq 0 \\ a_0 + a_2 \leq 0 \\ a_0 \leq 0 \end{array} \right. \end{aligned}$$



## Rappel du Sommaire

- 7 Méthodes de DL et extensions spatiales
  - Introduction aux réseaux de neurones
  - Le perceptron : l'élément de base d'un NN
  - MLP : le perceptron multicouche
  - Le modèle DeepKriging

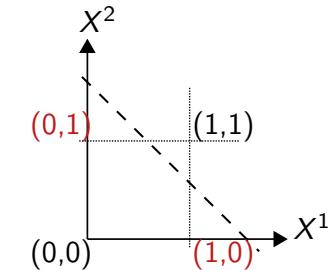
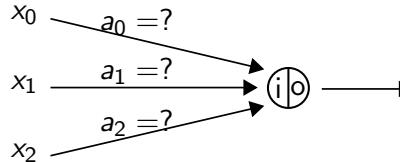
## Le perceptron : échec de la modélisation du “XOR”

Signal post-synaptique et fonction d'Heaviside :

$X^1$	$X^2$	XOR
1	1	0
1	0	1
0	1	1
0	0	0

$$\begin{aligned} a_0 + a_1 X^1 + a_2 X^2 &\leq 0 \\ \left\{ \begin{array}{l} a_0 + a_1 + a_2 \leq 0 \\ a_0 + a_1 > 0 \\ a_0 + a_2 > 0 \\ a_0 \leq 0 \end{array} \right. \end{aligned}$$

Ensemble de contraintes **incompatibles**.



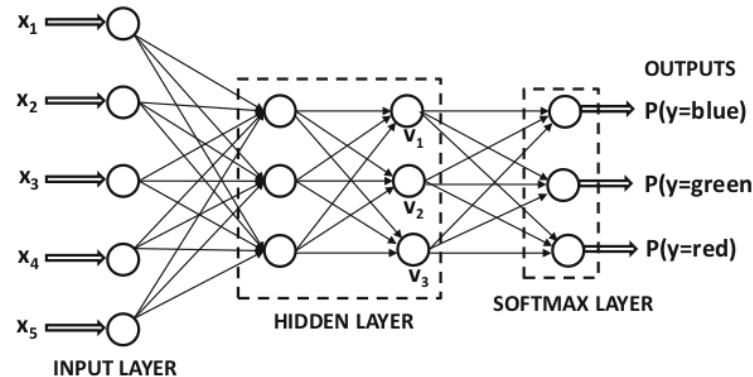
## MLP : le perceptron multicouche

## MLP : le perceptron multicouche

- Les modèles précédents définissent des modèles linéaires avec certaines limites.
- Le **perceptron multicouche/Feed Forward Network/MultiLayer Perceptron/MLP** est une généralisation de ces modèles :
  - ▶ En régression il permet de traiter les cas **non-linéaires** de régression.
  - ▶ En catégo., il permet de déterminer des fcts de décision **non-linéaires** permettant de résoudre le problème “XOR” précédent par exemple.
- Il consiste en l'ajout de **couches de neurones dites cachées** entre les données en entrée et les données en sortie. C'est en fait l'utilisation de fcts d'activation **non-linéaires** dans la couche cachée qui permet d'étendre les méthodes linéaires précédentes. Cela permet de définir une classe d'hypothèses  $\mathbb{H}$  de très grande complexité.

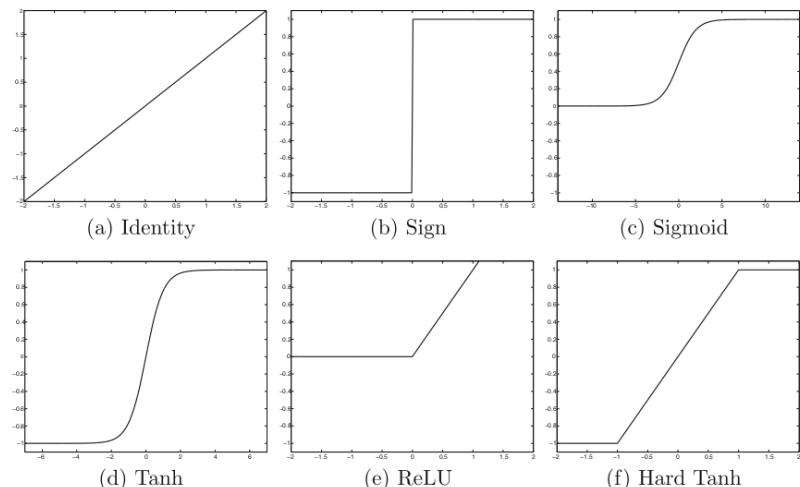
## Schéma d'un perceptron multicouche

Extrait de [Aggarwal, 2018].



## Exemples de fonctions d'activation

Extrait de [Aggarwal, 2018].



## MLP : 1 couche cachée

- Dans ce qui suit nous supposerons dans un premier temps une seule couche cachée afin d'introduire de façon formelle les MLP (mais nous verrons plus loin des configurations/architectures avec plusieurs couches cachées qui se succèdent).
- Nous supposerons :
  - p données en entrée plus un biais  $x_0 : \mathbf{x} = (x_0, x_1, \dots, x_p)$ .
  - Une 1ère couche de m perceptrons (la couche cachée).
  - Une 2ème couche de q perceptrons ( $q = 1$  dans le cas de la régression, couche de sortie).
  - La 1ère couche contient  $(p + 1) \times m$  coefficients synaptiques.
  - La 2ème couche contient  $(m + 1) \times q$  coefficients synaptiques.
- Rq : dans un MLP on suppose par défaut que toutes les variables initiales sont connectées à tous les neurones de la couche cachée et que tous les neurones de la couche cachée sont connectés à tous les neurones de la couche de sortie.

## MLP : 1 couche cachée (suite)

- Les données en entrée  $\mathbf{x}$  sont envoyées à chaque perceptron  $k$  de la 1ère couche qui combine linéairement celles-ci en un signal post-synaptique :  $s_k(\mathbf{x}) = \mathbf{a}_k^\top \mathbf{x}$ .
- Le perceptron  $k$  de la 1ère couche applique ensuite la fonction d'activation  $h_1 : z_k = h_1(s_k(\mathbf{x}))$ .
- Les données du vecteur  $\mathbf{z}_{\setminus 0} = (z_1, \dots, z_m)$  sont transmis à chaque perceptron  $l$ , représenté par la fct.  $g_l$ , de la 2ème couche (c.à.d. de la couche de sortie) qui le combine linéairement avec l'ajout d'un biais :

$$s_l(\mathbf{z}) = \mathbf{b}_l^\top \mathbf{z}$$

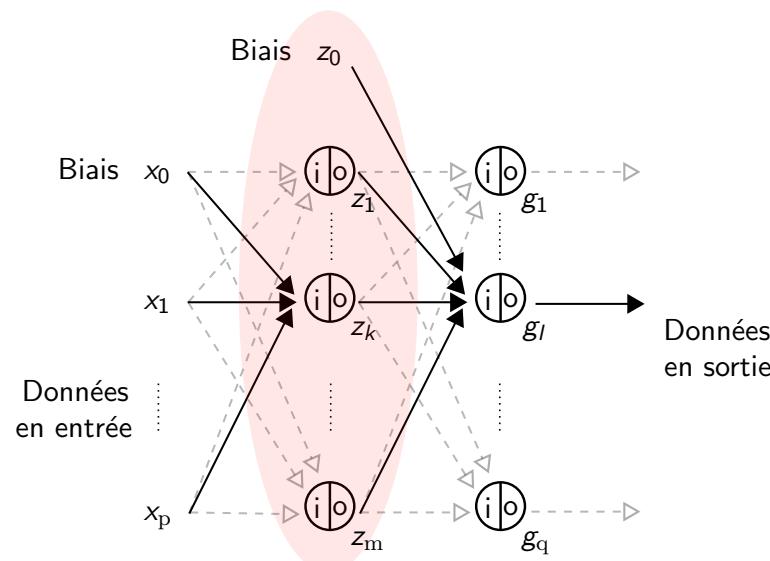
avec :  $\mathbf{z} = (1, \mathbf{z}_{\setminus 0})$  et  $\mathbf{b}_l = (b_{l,0}, \dots, b_{l,m}) \in \mathbb{R}^{m+1}$ .

- Le perceptron  $l$  de la 2ème couche applique ensuite la fonction d'activation  $h_2$  pour obtenir les données en sortie :

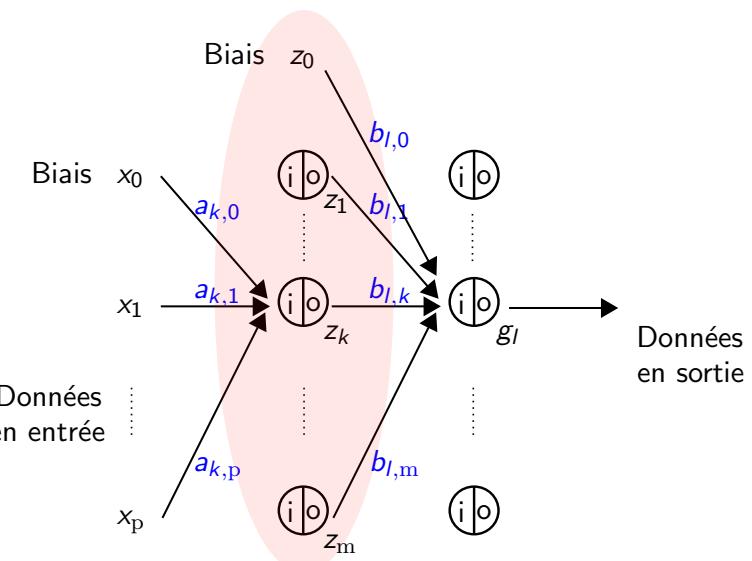
$$g_l = h_2(s_l(\mathbf{z}))$$

- Rq : les fcts d'activation  $h_1$  et  $h_2$  peuvent être différentes.

## Schéma d'un MLP avec 1 couche cachée



## Schéma d'un MLP avec 1 couche cachée + coefficients



## MLP : fonction d'activation de la couche de sortie

- Fonction d'activation de la couche de sortie (deuxième couche) :
  - Pour la rég. on utilise un perceptron de sortie et la fct. identité :

$$h_2(s_l(\mathbf{z})) = s_l(\mathbf{z}) = \mathbf{b}_l^\top \mathbf{z}$$

- Pour la catégo. binaire, un perceptron de sortie et la fct. sigmoïde :

$$h_2(s_l(\mathbf{z})) = \frac{1}{1 + \exp(-\mathbf{b}_l^\top \mathbf{z})}$$

- Pour la catégo. en  $q > 2$  classes,  $q$  perceptrons de sortie et la fct. softmax :

$$h_2(s_l(\mathbf{z})) = \frac{\exp(\mathbf{b}_l^\top \mathbf{z})}{\sum_{l=1}^q \exp(\mathbf{b}_l^\top \mathbf{z})}$$

- Pour des pbs de catégo., la var. en sortie  $g_l$  peut être interprétée telle la prob. d'appartenir à  $C_l$ .

## MLP : décision dans un espace issu d'une projection non-linéaire

- Le perceptron à deux couches peut être vu comme un modèle comportant une phase de **projection non-linéaire** :
  - La 1ère couche produit une projection de  $\mathbb{X}$  de dimension  $p + 1$  vers un nouvel espace de dimension  $m + 1$ .
  - La 2ème couche peut être vue comme  $q$  neurones en parallèle mais prenant en *input* les données issues de l'espace intermédiaire.
- Cet espace caché est découvert lors de l'apprentissage. Cette capacité des NN avec une couche cachée (ou plus) et une activation non-linéaire, d'apprendre une représentation des données facilitant la résolution d'une tâche donnée, est une spécificité qui explique le succès de ce type de modèle vis-à-vis des autres techniques.
- En statistique, des modèles proches appelés "**projection pursuit**" ont été proposés [Hastie et al., 2011] :  $f(\mathbf{x}) = \sum_{k=1}^m g_k(\mathbf{a}_k^\top \mathbf{x})$  où les  $g_k$  et les  $\mathbf{a}_k$  sont des fcts et vect. à estimer. Mais les NN ne se limitent pas à une seule couche cachée ce qui en fait des modèles plus flexibles.

## MLP : Résolution du “XOR”

- Définition logique de l'opération binaire “XOR”

$X^1$	$X^2$	XOR
1	1	0
1	0	1
0	1	1
0	0	0

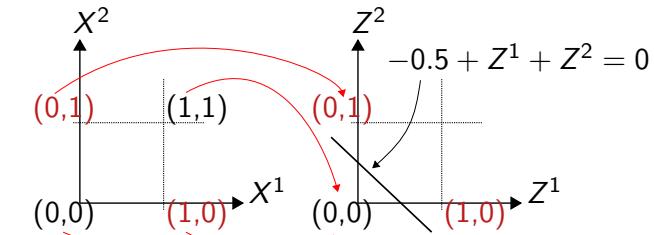
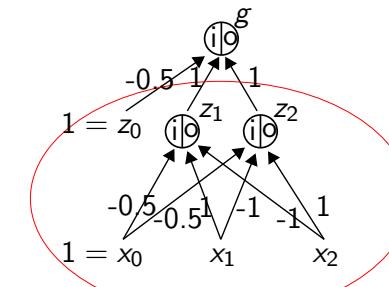
- Contraintes linéaires incompatibles (dans l'espace initial) :

$$\begin{cases} a_0 \leq 0 \\ a_0 + a_1 + a_2 \leq 0 \\ a_0 + a_1 > 0 \\ a_0 + a_2 > 0 \end{cases}$$

- Écriture sous la forme normale disjonctive :

$$X^1 \text{ XOR } X^2 \Leftrightarrow (X^1 \text{ AND } \neg X^2) \text{ OR } (\neg X^1 \text{ AND } X^2)$$

- Pour résoudre le problème avec un perceptron à 2 couches on prend  $m = 2$  et  $h$  la fonction d'Heaviside.



## MLP : formalisation avec 1 couche cachée

- L'estimation d'un MLP consiste à inférer l'ensemble des paramètres

$$\mathbb{P} = (\mathbf{a}_1, \dots, \mathbf{a}_m, \mathbf{b}_1, \dots, \mathbf{b}_q)$$

- $\forall k = 1, \dots, m$ ;  $\mathbf{a}_k$  est de taille  $(p+1) \times 1$
- $\forall l = 1, \dots, q$ ;  $\mathbf{b}_l$  est de taille  $(m+1) \times 1$

- Le vecteur  $\mathbf{z}$  obtenu à l'issue de la 1ère couche est de terme général :

$$z_k = h_1(\underbrace{\mathbf{a}_k^\top \mathbf{x}}_{s_k(\mathbf{x})}) = h_1(\langle \mathbf{a}_k, \mathbf{x} \rangle)$$

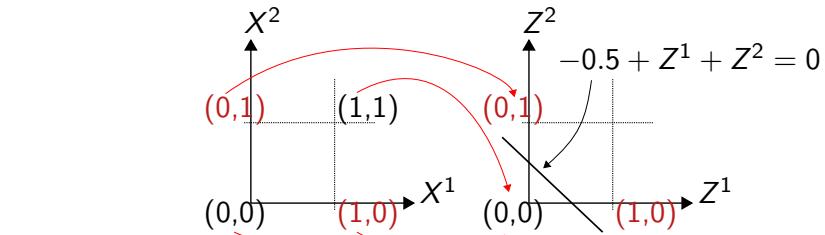
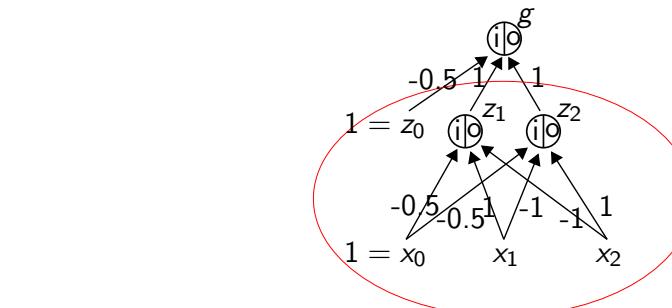
- Le vecteur  $\mathbf{g}$  obtenu à l'issue de la 2ème couche est de terme général :

$$g_l = h_2(\underbrace{\mathbf{b}_l^\top \mathbf{z}}_{s_l(\mathbf{z})}) = h_2(\langle \mathbf{b}_l, \mathbf{z} \rangle)$$

- Ainsi la fonction de décision vectorielle  $\mathbf{g}(\mathbf{x})$  donnée en sortie du perceptron à 2 couches est de terme général :

$$g_l(\mathbf{x}) = h_2(\underbrace{\mathbf{b}_l^\top \mathbf{z}}_{\mathbf{b}_l}, \underbrace{(z_0, h_1(\langle \mathbf{a}_1, \mathbf{x} \rangle), \dots, h_1(\langle \mathbf{a}_m, \mathbf{x} \rangle))}_{\mathbf{z}})$$

## MLP : Résolution du “XOR” (suite)



## MLP : fonctions objectifs régression et catégorisation

- La **fonction objectif** est de deux types selon le problème traité :

- Pour la **régression**, on utilise la somme des carrés des résidus :

$$\text{Err}(g) = \text{Scr}(g) = \sum_{i=1}^n \underbrace{(y_i - g(\mathbf{x}_i))^2}_{\text{Err}_i} \text{ où } \forall i : y_i \in \mathbb{R}$$

- Pour la **catégorisation**, on utilise la cross-entropie définie par :

$$\text{Err}(\mathbf{g}) = \text{Ce}(\mathbf{g}) = \sum_{i=1}^n \sum_{l=1}^q \underbrace{-y_{il} \log(g_l(\mathbf{x}_i))}_{\text{Err}_i}$$

où  $\forall i, l : y_{il} = 1$  si  $y_i = C_l$  et  $y_{il} = 0$  sinon.

- Ce sont des pbs de min. non-contraints mais non-convexes et dans ces cas, chercher un mininiseur pose des pbs de sur-apprent. (cf plus loin).

## MLP : 1 couche cachée, écriture matricielle

- Nous avons détaillé le modèle du MLP couche après couche et en utilisant le perceptron comme unité de base.
- Nous allons ci-dessous formalisé ces éléments en prenant une couche (donc un ens. de neurones) comme unité de base. Les dif. coef. synap. en entrée d'un perceptron sont représentés par un vect. Ainsi, **l'ens. des coef. synap. d'une couche sera représenté par une matrice.**
- Pour la 1ère couche (couche cachée) il y a  $m$  perceptrons dont les coef. synap. sont  $\{\mathbf{a}_k\}_{k=1,\dots,m}$  avec  $\mathbf{a}_k \in \mathbb{R}^{(m+1) \times 1}$ . Notons  $\mathbf{A} = (\mathbf{a}_1 \ \dots \ \mathbf{a}_m) \in \mathbb{R}^{(p+1) \times m}$ . Nous avons alors :

$$\mathbf{z}_{\setminus 0} = h_1(\mathbf{A}^T \mathbf{x})$$

où  $\mathbf{z}_{\setminus 0} = (z_1, \dots, z_m) \in \mathbb{R}^m$  et par abus de notation  $h_1(\mathbf{A}^T \mathbf{x})$  signifie que  $h_1$  est appliquée à chaque élément du vecteur  $\mathbf{A}^T \mathbf{x} \in \mathbb{R}^m$ .

## Approximateur universel de propositions logiques

- Le perceptron multicouche est un **approximateur universel** de toute proposition logique.
- Toute proposition logique peut être représentée par une disjonction de conjonctions (Disjunctive Normal Form). Ex. :

$$A \vee B \Leftrightarrow (A \wedge B) \vee (A \wedge \neg B) \vee (\neg A \wedge B)$$

- Le perceptron à 2 couches peut alors représenter toute prop. logique :
  - chq. conjonction est représentée par un perceptron de la 1ère couche,
  - la disjonction est représentée par la 2ème couche.

## MLP : 1 couche cachée, écriture matricielle (suite)

- Pour la 2ème couche (couche de sortie) de  $q$  neurones avec les coef. synap.  $\{\mathbf{b}_l\}_{l=1,\dots,q}$  dans  $\mathbb{R}^q$ , notons  $\mathbf{B} = (\mathbf{b}_1 \ \dots \ \mathbf{b}_q) \in \mathbb{R}^{(m+1) \times q}$  :

$$\mathbf{g} = h_2(\mathbf{B}^T \mathbf{z})$$

où  $\mathbf{z} = (\overbrace{1}^{z_0}, \mathbf{z}_{\setminus 0}) \in \mathbb{R}^{m+1}$  et  $\mathbf{g} = (g_1, \dots, g_q) \in \mathbb{R}^q$  (pas de biais dans la couche de sortie).

- Le MLP avec une couche cachée de  $m$  perceptrons et  $q$  perceptrons en parallèle en couche de sortie est une fct. vectorielle  $\mathbf{g}$  à valeurs dans  $\mathbb{R}^q$  qui peut donc s'écrire :

$$\mathbf{g}(\mathbf{x}) = h_2(\mathbf{B}^T (\underbrace{1, h_1(\mathbf{A}^T \mathbf{x})}_{\mathbf{z}}))$$

## Approximateur universel de fonctions

- Le MLP avec comme fcts d'activation des *squashing function* (sigmoïde par ex.) [Hornik et al., 1989] ou la Relu [Leshno et al., 1993], est un approximateur universel de toute fct. continue définie et à valeurs dans des ens. compacts de dim. finie (ss-ens. fermés et bornés de  $\mathbb{R}^p$  et de  $\mathbb{R}^q$  resp. par ex.).
- En bref, en ce qui nous concerne, ce résultat indique qu'un MLP avec
  - 1 couche cachée de dimension  $m$  suffisamment large,
  - des fct. d'activation sigmoïde (par ex.) pour la couche cachée et,
  - la fct. identité pour la couche de sortie de dimension 1 ;
 peut approximer toute fct. de plusieurs variables et à valeurs réelles, qui soit continue et suffisamment lisse.
- Un MLP permet donc de définir des classes d'hypothèses très larges permettant de représenter de façon très flexible :
  - des courbes de régression non-linéaires ou des courbes de frontières de décision non-linéaires pour les pbs supervisés.
- Cette extraordinaire complexité ne garantie pas que l'on soit capable de trouver la bonne fct. pour autant (pbs d'inf. et de sur-apprent.) !

## MLP : formalisation avec d couches cachées

- Nous avons jusqu'à présent considéré un MLP avec 2 couches : 1 couche cachée et une couche de sortie.
- Cette architecture peut être étendue en utilisant  $d \geq 2$  couches cachées. Dans ce cas, le MLP suppose que tous les neurones de la couche cachée  $c$  sont connectés aux neurones de la couche  $c + 1$ . On parle également de **fully connected NN** ou de **dense NN**.
- Notons par  $m_c$  le nb. de perceptrons dans la couche cachée avec  $c = 1, \dots, d$  et par  $\mathbf{W}^{[c]} \in \mathbb{R}^{(m_{c-1}+1) \times m_c}$  la matrice des poids des perceptrons de la couche  $c$ . Rq : si  $c = 1$  alors la couche  $c - 1$  est la couche d'entrée (ou des inputs) composée des variables initiales de  $\mathbf{x}$ .
- Le signal sortant de la couche  $c$  et transmis à la couche  $c + 1$  est :

$$\mathbf{z}_{\setminus 0}^{[c]} = h^{[c]}(\mathbf{W}^{[c]}\top(1, \mathbf{z}_{\setminus 0}^{[c-1]}))$$

- Le MLP est alors une fct. composée de type :

$$\mathbf{g}(\mathbf{x}) = h_d \left( \dots [\mathbf{W}^{[3]}]\top(1, h_2([\mathbf{W}^{[2]}]\top(1, h_1([\mathbf{W}^{[1]}\top \mathbf{x})))) \right)$$

## MLP : formalisation avec d couches cachées (suite)

- Par ailleurs, le choix d'un NN avec plus. couches cachées définit une classe d'hypothèses dont les éléments sont des fcts obtenues par composition de fcts plus simples. L'hypothèse ou biais inductif que nous faisons peut être exprimé comme suit :
  - on suppose qu'il est nécessaire de **découvrir un ens. de facteurs latents** (unités dans l'avant dernière couche) qui peuvent expliquer, de manière structurée, les variations importantes des données dans le but de résoudre la tâche,
  - on suppose que ces facteurs de "haut-niveau" peuvent être, à leur tour et en cascade, être décrits par des facteurs de plus "bas-niveau".
- Ce biais inductif est cohérent si les données possèdent des régularités ("motifs qui se répètent") et que détecter celles-ci contribuent à résoudre la tâche. Dans ce contexte :
  - un *deep NN* peut capter ces régularités avec "peu" de neurones avec la structure hiérarchique donnée par les compositions successives de fcts,
  - un *shallow NN* traite toute l'info. "au même niveau" (1 cc.) et aura besoin d'un nb. plus élevé d'unités pour capturer l'ens. des régularités.

## MLP : formalisation avec d couches cachées (suite)

- Nous avons vu slide 508, qu'en théorie, un MLP avec 1 couche cachée suffisamment large et des fonctions d'activation non-linéaires pouvait représenter tout type de fct. continue. En pratique, le nb. de neurones/unités dans la couche cachée peut être très grand. De plus, l'insuffisance des données (*overfitting*) et les imperfections des algo. d'optimisation (*no free lunch*, cf cours de ML) pourraient ne pas permettre d'estimer correctement la bonne fonction.
- Dans ce contexte, une idée centrale de l'apprentissage profond est qu'une architecture *deep* de NN permettrait de réduire le nb. de neurones à utiliser pour estimer la fonction visée et réduire également les pbs de sur-apprentissage. En théorie, l'article [Montufar et al., 2014] montre que des fonctions représentables par un *deep NN* peuvent au contraire nécessiter un nb. exponentiellement grand de neurones pour être représentées par des *shallow NN* (NN avec 1 cc.).

## MLP : formalisation avec d couches cachées (suite)

- Prenons une illustration : supposons une fct. répliquant un même motif 1024 fois le long d'un axe. Dans ce cas :
  - un *shallow NN* aura besoin de 1024 unités pour représenter la fonction.
  - un *deep NN* aura besoin de 10 couches cachées à deux unités ( $2^{10} = 1024$ ), soit 20 neurones, pour représenter la fonction.

Extrait de [Aggarwal, 2018].

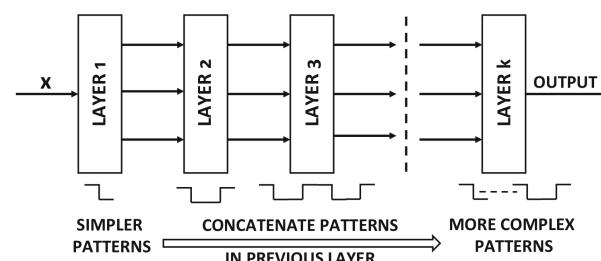


Figure 1.15: Deeper networks can learn more complex functions by composing the functions learned in earlier layers.

## MLP : Régularisation

- Les MLP (le DL en général) est très sujet au **sur-apprentissage** : l'erreur sur les données d'entraînement est très petite alors que l'erreur sur des données de test est très grande.
- Le sur-apprentissage indique que le modèle s'ajuste trop fortement aux données d'entraînement et n'est **pas capable de généraliser**.
- Pour palier ce pb. des **techniques de régularisation** sont utilisées. Dans le cas du DL, voici quelques techniques classiques :
  - Early stopping* : on arrête l'apprentissage afin de converger.
  - Pénalisation : on ajoute une fct. de pénalisation dans la fct. de perte qui constraint la variance du modèle.
  - Dropout* : on masque aléatoirement des neurones au cours de l'apprentissage par SGD ce qui est équivalent à une approche ensembliste comparable au *bagging*.
- La régularisation est un aspect important des réseaux de neurones afin que ces modèles conduisent à de bonnes performances en pratique mais leur étude approfondie dépasse le cadre de ce cours.

## Limites des méthodes de Krigeage

- Nous avons vu précédemment, les méthodes de Krigeage notamment le Krigage dit Universel qui suppose une tendance externe dépendant de variables exogènes notées  $\mathbf{f}(\mathbf{p}) = (f_1(\mathbf{p}_0), \dots, f_l(\mathbf{p})) \in \mathbb{R}^l, \forall \mathbf{p} \in \mathbb{D}$ .
- Les modèles de Krigeage présentent les inconvénients suivants :
  - Ils passent par l'estimation de la fonction de variogramme ou celle de covariance et supposent que le champ aléatoire est stationnaire. Cette hypothèse n'est pas toujours vérifiée en pratique.
  - Pour prédire il est nécessaire d'inverser la matrice variance-covariance ce qui dans le cas général a une complexité de calcul en  $O(n^3)$ . Ce coût est prohibitif dans le contexte de données volumineuses.
  - Ce sont des modèles linéaires : la prédiction en une position géographique nouvelle est donnée par une combinaison linéaire des observations de l'échantillon. Or certains phénomènes spatiaux complexes impliquent des dépendances non linéaires.
  - Les données analysées sont nécessairement réelles. On fait d'ailleurs souvent l'hypothèse d'une fonction aléatoire Gaussienne. Les méthodes de Krigeage ne traitent pas des données discrètes géoréférencées.

## Rappel du Sommaire

7

### Méthodes de DL et extensions spatiales

- Introduction aux réseaux de neurones
- Le perceptron : l'élément de base d'un NN
- MLP : le perceptron multicouche
- Le modèle DeepKriging

## Le modèle DeepKriging (DK)

- Nous étudions le modèle **DeepKriging (DK)** [Chen et al., 2024].
- DK emploie des **réseaux de neurones** et surmonte les inconvénients du KU en permettant **plus de non linéarité**, une **complexité réduite**, la généralisation à la **prédiction de variables discrètes**.
- Les caractéristiques de DK que nous discuterons par la suite sont :
  - Une modélisation des dépendances spatiales par le biais de fonctions de base spatiales.
  - L'absence de calculs matriciels lourds et une scalabilité accrue pour des données en masse.
  - Un prédicteur permettant des dépendances non linéaires avec les covariables et les observations.
  - La possibilité de modéliser des données non Gaussiennes et/ou non stationnaires.
  - Un lien direct avec les méthodes de Krigeage traditionnelles.
- Un résultat théorique sur lequel DK se fonde est le **théorème de Karhunen-Loève** que nous présentons ci-après.

## Backgrd : Théorème de Karhunen-Loève (KL)

- On présente le **théorème de Karhunen-Loève** dans le cas classique des processus stochastiques mais celui-ci se généralise au cas des fonctions aléatoires telles qu'introduites en slide 421.
- Soit  $\mathbb{D} = [a, b]$  un intervalle de  $\mathbb{R}$  et  $(\Omega, \Sigma, P)$  un espace probabilisé.
- $\mathcal{X}$  est un processus stochastique défini sur  $\mathbb{D} \times \Omega$ .
- On suppose que :
  - Le processus est centré :  $E\{\mathcal{X}(p)\} = \mu_{\mathcal{X}}(p) = 0, \forall p \in [a, b]$ .
  - Le processus est de carré intégrale :  $E\{\mathcal{X}^2(p)\} < \infty, \forall p \in [a, b]$  ( $\mathcal{X}(p) \in L_2(\Omega, \Sigma, P)$ ,  $\forall p \in [a, b]$ ).
  - La fonction de covariance  $C$  définie sur  $\mathbb{D} \times \mathbb{D}$  est donnée par :

$$\begin{aligned} C(p, p') &= C\{\mathcal{X}(p), \mathcal{X}(p')\} \\ &= E\{\mathcal{X}(p)\mathcal{X}(p')\} \end{aligned}$$

## Backgrd : Théorème de Karhunen-Loève (KL) (suite)

- Soit  $\mathcal{X}$  un processus stochastique défini sur  $(\Omega, \Sigma, P)$  et indiqué sur  $\mathbb{D} = [a, b]$ , qui soit de carré intégrable, de fonction moyenne nulle et de fonction de covariance  $C$  continu sur  $\mathbb{D} \times \mathbb{D}$ . Alors  $C$  est un noyau de Mercer et  $\mathcal{X}$  admet la représentation suivante :

$$\mathcal{X}(p) = \sum_{m=1}^{\infty} \mathcal{Y}_m \phi_m(p), \text{ avec } \mathcal{Y}_m = \int_a^b \mathcal{X}(p) \phi_m(p) dp, \text{ et où :}$$

- $\{\phi_m\}_{m \geq 1}$  est l'ens. des **fonctions propres de l'opérateur intégral**  $T_C$ , qui forme une **base orthonormée de  $L_2([a, b])$** .
- $\{\mathcal{Y}_m\}_{m \geq 1}$  sont des v.a.r. d'espérances nulles, mutuellement non corrélées et de variances  $\{\lambda_m\}_{m \geq 0}$  l'ens. des valeurs propres de  $T_C$  :

$$\begin{aligned} E\{\mathcal{Y}_m\} &= 0 & \forall m \geq 0 \\ E\{\mathcal{Y}_m \mathcal{Y}_{m'}\} &= \delta_{mm'} \lambda_m & \forall m, m' \geq 0 \end{aligned}$$

## Backgrd : Théorème de Karhunen-Loève (KL) (suite)

- On associe à  $C$  l'**opérateur intégral**  $T_C$  suivant :

$$\begin{aligned} T_C : L_2([a, b]) &\rightarrow L_2([a, b]) \\ f &\rightarrow T_C(f)(.) = \int_a^b C(p, .)f(p) dp \end{aligned}$$

où  $L_2([a, b])$  est l'espace de Hilbert des fonctions  $f$  de carrés intégrables c.à.d. t.q.  $\int_a^b |f(p)|^2 dp < \infty$ .

- $T_C$  est un opérateur linéaire il existe alors des couples (**valeur propre**, **fonction propre**) dénotés  $(\lambda_m, \phi_m)$  qui sont solutions de l'équation :

$$T_C(\phi_m) = \lambda_m \phi_m \Leftrightarrow \int_a^b C(p, p') \phi_m(p) dp = \lambda_m \phi_m(p')$$

- Le théorème de Karhunen-Loève fournit une décomposition optimale d'un processus stochastique en une série de fonctions orthogonales pondérées par des coefficients qui sont des v.a.r. Ce théorème fondamental est énoncé ci-après.

## Retour sur le Krigeage Universel

- Revenons sur le modèle avec tendance externe du KU :

$$\mathcal{Z}(\mathbf{p}) = [\mathbf{f}(\mathbf{p})]^\top \mathbf{a} + \epsilon(\mathbf{p})$$

- Nous avons vu en slide 471 la formulation du prédicteur du KU :

$$\widehat{\mathcal{Z}(\mathbf{p}_0)} = [\mathbf{f}(\mathbf{p})]^\top \mathbf{a} + [\mathbf{K}^{-1} \mathbf{k}]^\top (\mathbf{Z} - \mathbf{F}\mathbf{a}), \text{ où :}$$

- $\mathbf{a} = (\mathbf{F}^\top \mathbf{K}^{-1} \mathbf{F})^{-1} \mathbf{F}^\top \mathbf{K}^{-1} \mathbf{Z}$
- $\mathbf{K} = (k_{ii'})_{i,i'=1,\dots,n}$  avec  $k_{i,i'} = C\{\mathcal{Z}(\mathbf{p}_i), \mathcal{Z}(\mathbf{p}_{i'})\} = C(\mathbf{p}_i, \mathbf{p}_{i'})$
- L'élément central dans les modèles de Krigeage est la fonction de variance-covariance  $C$  (dont  $\mathbf{K}$  est issue) :
  - $\epsilon(\mathbf{p})$  est supposé être une SRF (ce qui implique que  $\mathcal{Z}$  est une SRF) ce qui permet de simplifier la modélisation de  $C$  mais réduit la flexibilité du modèle.
  - La prédiction des modèles de Krigeage implique nécessairement l'inversion de la matrice  $\mathbf{K}$  ce qui est coûteux en temps de traitement quand  $n$  grand.

## DK : représentation alternative de $\epsilon$

- Les auteurs de [Chen et al., 2024] invoquent le théorème de Karhunen-Loève pour proposer une représentation alternative de  $\epsilon(\mathbf{p})$ .
- En effet, on suppose que  $\epsilon(\mathbf{p})$  est une fonction aléatoire de fonction moyenne nulle et de fonction de covariance  $C$  continue sur  $\mathbb{D} \times \mathbb{D}$ . On sait alors qu'on peut écrire (théorème de KL) :

$$\epsilon(\mathbf{p}) = \sum_{m=1}^{\infty} \gamma_m \phi_m(\mathbf{p}), \text{ avec } \gamma_m = \int_a^b \phi_m(\mathbf{p}) \epsilon(\mathbf{p}) d\mathbf{p}, \text{ et où :}$$

$\{\phi_m\}_{m \geq 1}$  est l'ens. des **fonctions propres de l'op. intégral  $T_C$** .

- Ne connaissant pas  $C$ , il n'est pas possible de déterminer  $\{\phi_m\}_{m \geq 1}$ .
- Dans ce contexte, l'idée centrale de DK est de prendre un **ens. fini de fonctions prédéfinies définies sur  $\mathbb{D}$**  que l'on notera  $\{\psi_m\}_{m=1,\dots,s}$ . Cet ens. est utilisé comme base d'un espace Euclidien (car dimension  $s$  finie) de fonctions. La fonction aléatoire  $\epsilon$  **est alors estimée dans cette base de fonctions**.

## DK : approche multirésolution

- Les auteurs de [Chen et al., 2024] reprennent l'idée présentée dans [Nychka et al., 2015] qui utilise un modèle de multirésolution pour la prédiction spatiale dans le cas de données volumineuses :
  - plusieurs maillages réguliers de  $\mathbb{D}$  avec des niveaux de granularité variables sont utilisés,
  - chaque maillage est associé à un ens. de fonctions à base radiale,
  - en particulier une fonction de Wendland (fct. de base radiale à support compact) est utilisée.
- Soit  $\{\mathbf{u}_j\}_{j=1,\dots,r}$  les points d'un maillage quelconque. Soit  $\theta$  un paramètre réel d'échelle. La fonction de Wendland utilisée est :

$$\psi_j(\mathbf{p}) = W_{3,2} \left( \frac{\|\mathbf{p} - \mathbf{u}_j\|}{\theta} \right)$$

$$\text{avec } W_{3,2}(d) = \begin{cases} (1-d)^6(35d^2 + 18d + 3)/3 & \text{si } d \in [0, 1] \\ 0 & \text{sinon} \end{cases}$$

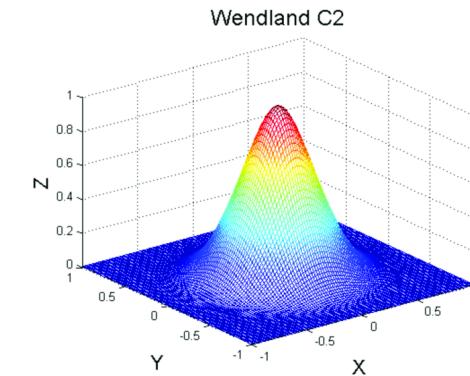
Le paramètre d'échelle  $\theta$  est fixé à  $2.5\delta$  où  $\delta$  est l'écart entre deux noeuds du maillage.

## DK : base de fonctions

- Si on reste strictement dans le cadre du théorème de Karhunen-Loève alors les fonctions de base doivent satisfaire :
  - $\psi_m \in L_2(\mathbb{D})$ ,  $\forall m = 1, \dots, s$ .
  - $\langle \psi_m, \psi_{m'} \rangle_{L_2(\mathbb{D})} = 0$ ,  $\forall m, m' = 1, \dots, s$ ;  $m \neq m'$ .
- Toutefois, le type de fonctions de base est moins important que le nombre de fonctions de base  $s$  [Chen et al., 2024]. Pour approximer correctement  $\epsilon$  dans la base de fonctions  $\{\psi_m\}_{m=1,\dots,s}$  il faut que la dimension de la base  $s$  soit plus grande que la taille de l'échantillon  $n$  :  $s \geq n$ .
- Plusieurs types de fonctions de base peuvent être employés :
  - Splines.
  - Ondelettes.
  - Fonctions de base radiale (*Radial Basis Functions -RBF-*).
  - ...
- DK emploie des réseaux de neurones.

## Illustration d'une fonction de Wendland

- Illustration d'une fonction de Wendland :  $W_{3,1}(d) = (1-d)_+^4(4d+1)$ .



Extrait de [Feng et al., 2021]

## DK : approche multirésolution (suite)

- Plusieurs niveaux de résolution  $h = 1, \dots, h$  sont utilisés.
- D'un niveau à un autre, les maillages s'affinent : le nb. de points double à chq. fois pour chaque dimension.
- Le nombre de noeuds du niveau  $h$  est donné par :

$$s_h = (9 \times 2^{h-1} + 1)^p \text{ où } \text{Dim}(\mathbb{D}) = p$$

- Pour avoir un nb. total de fonctions de base  $s$  t.q.  $s \geq n$  il faut :

$$h = \lceil \log_2(n^{1/p}/10) \rceil$$

- Exemple : si  $n = 7000$  localisations dans  $\mathbb{D} \subset \mathbb{R}^2$  alors il faudra 4 niveaux et on a  $s = s_1 + s_2 + s_3 + s_4 = 10^2 + 19^2 + 37^2 + 73^2 = 7159$  noeuds et donc fonctions de base.
- L'approche multirésolution hiéarchiques avec la fonction de Wenland centrés en un nombre de points  $k \geq n$  permet de modéliser des **fonctions de covariance non stationnaires capables de capturer des dépendances spatiales locales et globales** [Nychka et al., 2015].

## DK : Réseaux de neurones

- DK considère les couples  $\{\mathbf{x}(\mathbf{p}_i), z_i\}_{i=1,\dots,n}$  comme ens. d'entraînement  $\mathbb{E}$ .
- DK propose d'apprendre à prédire spatialement  $\mathcal{Z}(\mathbf{p}_0)$  pour un quelconque  $\mathbf{p}_0 \in \mathbb{D}$  à partir d'un modèle neuronal MLP appris sur  $\mathbb{E}$ .
- Reprenons les notations du slide 512 à propos des MLP à  $d$  couches. Le modèle DK peut alors être formalisé comme suit :

- ▶ Données en entrée de DK :  $\mathbf{x}(\mathbf{p})$ .
- ▶ Sortie de la 1ère couche cachée :  $\mathbf{z}^{[1]}(\mathbf{p}) = h^{[1]}([\mathbf{W}^{[1]}]^\top(1, \mathbf{x}(\mathbf{p})))$
- ▶ Sortie de la 2ème couche cachée :  $\mathbf{z}^{[2]}(\mathbf{p}) = h^{[2]}([\mathbf{W}^{[2]}]^\top(1, \mathbf{z}^{[1]}(\mathbf{p})))$
- ▶ ...
- ▶ Données en sortie (dème couche) :  
 $\mathbf{z}^{[d]}(\mathbf{p}) = h^{[d]}([\mathbf{W}^{[d]}]^\top(1, \mathbf{z}^{[d-1]}(\mathbf{p})))$

où,  $\forall c = 1, \dots, d$  :

- ▶  $h^{[c]}$  est la fonction d'activation de la couche  $c$ .
- ▶  $\mathbf{W}^{[c]} \in \mathbb{R}^{(m_{c-1}+1) \times m_c}$  est la mat. des poids des perceptrons de la couche  $c$ .

## DK : Expansion de bases

- Pour toute position géographique  $\mathbf{p} \in \mathbb{D}$ , on peut évaluer les  $s$  fonctions de base prédéfinies (tout maillage confondu) et employer le vecteur  $\psi \in \mathbb{R}^s$  ci-après comme outils de représentation riche de l'information spatiale dans le voisinage de  $\mathbf{p}$  :

$$\psi(\mathbf{p}) = (\psi_1(\mathbf{p}), \dots, \psi_s(\mathbf{p}))$$

- DK procède à une expansion de base en concaténant le vecteur des données des covariables  $\mathbf{f}$  et le vecteur des fonctions de base  $\psi$ . En chaque localisation  $\mathbf{p}$  on considère le vecteur étendu :

$$\begin{aligned} \mathbf{x}(\mathbf{p}) &= (\mathbf{f}(\mathbf{p}), \psi(\mathbf{p})) \\ &= (\underbrace{f_1(\mathbf{p}), \dots, f_l(\mathbf{p})}_{\text{Var. explicatives}}, \underbrace{\psi_1(\mathbf{p}), \dots, \psi_s(\mathbf{p})}_{\text{Fonctions de base}}) \end{aligned}$$

## DK : Illustration d'un modèle DK à 1 couche cachée

Extrait de [Chen et al., 2024]

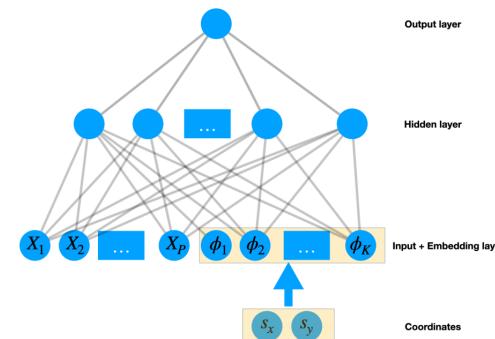


Figure 1: Visualization of the DeepKriging structure in 2D spatial prediction based on a three-layer DNN

- Remarque : quand le nb. de neurones dans la couche cachée est très grand on parle de réseau de neurones large en opposition à un réseau de neurones profonds qui lui comporte bcp. de couches cachées.

## DK : Régression et catégorisation

- Notons la prédiction de DK pour une donnée en entrée  $\mathbf{x}(\mathbf{p})$  par :

$$\mathbf{g}(\mathbf{p}) = \mathbf{z}^{[d]}(\mathbf{p}) = h^{[d]} \left( [\mathbf{W}^{[d]}]^\top (1, \mathbf{z}^{[d-1]}(\mathbf{p})) \right)$$

- Le modèle DK étant un réseau de neurones il permet de traiter plusieurs types de problèmes supervisés (cf slides 499 et 504) :

► Cas de la **régression** :

- Problème typique de prédiction spatiale.
- Couche de sortie :  $m_d = 1$  et  $h^{[d]} = \text{Id}$ .
- Fonction de perte :  $\text{Scr}(g) = \sum_{i=1}^n (z_i - g(\mathbf{x}(\mathbf{p}_i)))^2$ .

► Cas de la **catégorisation** multiclasse :

- Problème atypique de prédiction spatiale.
- La variable cible est encodée dans une matrice  $\mathbf{Z} = (z_{il})_{i=1,\dots,n; l=1,\dots,1}$  où  $z_{il} = 1$  si  $z(\mathbf{p}_i) = C_l$  et  $z_{il} = 0$  sinon.
- Couche de sortie :  $m_d = q$  et  $h^{[d]} = \text{Softmax}$ .
- Fonction de perte :  $\text{Ce}(g) = \sum_{i=1}^n \sum_{l=1}^q (-z_{il} \log(g_l(\mathbf{x}(\mathbf{p}_i))))$ .

## DK : Liens avec la méthode de Krigeage FRK (suite)

- Sous les hypothèses précédentes, la mat. de variance-covariance  $\mathbf{K}$  de  $\epsilon$  (ou  $\mathcal{Z}$ ) calculée en chq. paire  $(\epsilon(\mathbf{p}_i), \epsilon(\mathbf{p}_{i'}))$ ,  $i, i' = 1, \dots, n$ , est t.q. :

$$\begin{aligned} [\mathbf{K}]_{ii'} &= C \{ \mathcal{Z}(\mathbf{p}_i), \mathcal{Z}(\mathbf{p}_{i'}) \} \\ &= C \{ \epsilon(\mathbf{p}_i), \epsilon(\mathbf{p}_{i'}) \} \\ &= C \{ \phi(\mathbf{p}_i)^\top \mathbf{Y}, \phi(\mathbf{p}_i)^\top \mathbf{Y} \} \\ &= \phi(\mathbf{p}_i)^\top C \{ \mathbf{Y}, \mathbf{Y} \} \phi(\mathbf{p}_i) \\ &= \phi(\mathbf{p}_i)^\top \Sigma_Y \phi(\mathbf{p}_i) \end{aligned}$$

- Si on considère la matrice complète on a :

$$\mathbf{K} = \Phi \Sigma_Y \Phi^\top \text{ avec } \Phi = (\phi(\mathbf{p}_1)^\top, \dots, \phi(\mathbf{p}_n)^\top) \in \mathbb{R}^{n \times s}$$

- $\mathbf{K}$  est de rang  $s$ , la prédiction (linéaire) de FRK en  $\mathbf{p}_0$  est donnée par :

$$\widehat{\mathcal{Z}(\mathbf{p}_0)} = [\mathbf{f}(\mathbf{p}_0)]^\top \mathbf{a} + [\mathbf{K}^{-1} \Phi \Sigma_Y \phi(\mathbf{p}_0)]^\top (\mathbf{Z} - \mathbf{Fa})$$

## DK : Liens avec la méthode de Krigeage FRK

- La méthode *Fixed Rank Kriging* (FRK) définie dans [Cressie and Johannesson, 2008] est une technique de Krigeage qui a des liens avec DK [Chen et al., 2024]. Reprenons le modèle de base :

$$\mathcal{Z}(\mathbf{p}) = [\mathbf{f}(\mathbf{p})]^\top \mathbf{a} + \epsilon(\mathbf{p})$$

- Dans FRK,  $\epsilon$  est également représentée dans une base de fonctions prédéfinies  $\{\phi_j\}_{j=1,\dots,s}$ . Toutefois, on suppose  $s < n$  ce qui implique que la matrice de variance-covariance  $\mathbf{K}$  est de faible rang  $s$ . Ceci a l'avantage de simplifier les calculs impliquant son inversion.
- Plus formellement, dans FRK on suppose :

$$\epsilon(\mathbf{p}) = \sum_{m=1}^s Y_m \phi_m(\mathbf{p}) = \phi(\mathbf{p})^\top Y$$

où  $\mathbf{Y} = (Y_1, \dots, Y_s)$  est un vecteur aléatoire Gaussien de taille  $s$  de matrice de variance-covariance  $\Sigma_Y = (C \{ Y_m, Y_{m'} \})_{m,m'=1,\dots,s}$ .

## DK : Autres propriétés

- D'autres arguments de nature théorique sont évoqués dans [Chen et al., 2024] :
  - DK en théorie de la décision : DK approxime le prédicteur spatial optimal en décomposant l'erreur totale de prédiction en plus. types d'erreur avec plus. prédicteurs de référence. Cela permet d'indiquer que la structure du réseau neuronal de DK offre une capacité de modélisation supérieure à celle du Krigeage, lui permettant de mieux capturer les dépendances spatiales complexes et de minimiser l'erreur d'approximation.
  - DK et processus Gaussien : lorsque le nb. de neurones dans la couche cachée est infini, DK est équivalent à un processus Gaussien. Cela permet d'indiquer que DK peut induire des fonctions de covariance non stationnaires, offrant une flexibilité supplémentaire pour modéliser des processus spatiaux complexes par rapport aux approches classiques de Krigeage.

## DK : Architecture par défaut

- En pratique, dans [Chen et al., 2024] les auteurs proposent l'architecture par défaut suivante pour DK (MLP à 3 couches cachées) :
  - (1) Normalize (min-max normalization) the observed covariates  $\mathbf{f}$  ;
  - (2) Build the embedding layer  $\psi$  using a 3 to 5 level multi-resolution radial basis functions with the corresponding basis matrix  $\Psi = (\psi(\mathbf{p}_1)^\top, \dots, \psi(\mathbf{p}_n)^\top)$ ;
  - (3) Remove the all-zero columns of the basis matrix  $\Psi$  ;
  - (4) Add the 1st dense layer with 100 hidden neurons and ReLu activation ;
  - (5) Add the 1st dropout layer with 0.5 dropout rate (cf slide 513) ;
  - (6) Add the 1st batch-normalization layer (pb. de gradient évanescence) ;
  - (7) Add the 2nd dense layer with 100 hidden neurons and ReLu activation ;
  - (8) Add the 2nd dropout layer with 0.5 dropout rate ;
  - (9) Add the 3rd dense layer with 100 hidden neurons and ReLu activation ;
  - (10) Add the 2nd batch-normalization layer ;
  - (11) Add the output layer.

## Quelques références I

-  Aggarwal, C. C. (2018). Neural networks and deep learning. Springer, 10 :978–3.
-  Allard, D. (2012). Statistiques spatiales : introduction à la géostatistique. [https://biosp.mathnum.inrae.fr/sites/default/files/2021-03/geostat\\_janv12.pdf](https://biosp.mathnum.inrae.fr/sites/default/files/2021-03/geostat_janv12.pdf). Accessed November 2024.
-  Anselin, L. (1988). Spatial econometrics : methods and models, volume 4. Springer Science & Business Media.
-  Anselin, L. (1995). Local indicators of spatial association—lisa. Geographical analysis, 27(2) :93–115.
-  Borcard, D. and Legendre, P. (2002). All-scale spatial analysis of ecological data by means of principal coordinates of neighbour matrices. Ecological modelling, 153(1-2) :51–68.
-  Breiman, L. (1996). Bagging predictors. Machine learning, 24(2) :123–140.
-  Breiman, L. (2001). Random forests. Machine learning, 45(1) :5–32.

## \*\*Code\*\* : DK, PM25

- Nous reprenons le code et l'exemple proposé sur le github des auteurs de [Chen et al., 2024] : ici.

## Quelques références II

-  Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). Classification and Regression Trees. Chapman & Hall, New York, NY.
-  Brunsdon, C., Fotheringham, A. S., and Charlton, M. E. (1996). Geographically weighted regression : a method for exploring spatial nonstationarity. Geographical analysis, 28(4) :281–298.
-  Chen, T. and Guestrin, C. (2016). Xgboost : A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 785–794.
-  Chen, W., Li, Y., Reich, B. J., and Sun, Y. (2024). Deepkriging : Spatially dependent deep neural networks for spatial prediction.
-  Chiles, J.-P. and Delfiner, P. (2012). Geostatistics : modeling spatial uncertainty, volume 713. John Wiley & Sons.
-  Cressie, N. and Johannesson, G. (2008). Fixed rank kriging for very large spatial data sets. Journal of the Royal Statistical Society Series B : Statistical Methodology, 70(1) :209–226.
-  Cutler, A., Cutler, D. R., and Stevens, J. R. (2012). Random forests. In Ensemble machine learning, pages 157–175. Springer.

## Quelques références III

-  Dawson, T., Sandoval, J. O., Sagan, V., and Crawford, T. (2018).  
A spatial analysis of the relationship between vegetation and poverty.  
*ISPRS International Journal of Geo-Information*, 7(3) :83.
-  de Jong, P., Sprenger, C., and van Veen, F. (1984).  
On extreme values of moran's i and geary's c.  
*Geographical Analysis*, 16(1) :17–24.
-  Doob, J. (1990).  
*Stochastic Processes*.  
A Wiley-interscience publication. Wiley.
-  Dray, S., Legendre, P., and Peres-Neto, P. R. (2006).  
Spatial modelling : a comprehensive framework for principal coordinate analysis of neighbour matrices (pcnm).  
*Ecological modelling*, 196(3-4) :483–493.
-  Duque, J. C., Anselin, L., and Rey, S. J. (2012).  
The max-p-regions problem.  
*Journal of Regional Science*, 52(3) :397–419.
-  Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996).  
A density-based algorithm for discovering clusters in large spatial databases with noise.  
In *kdd*, volume 96, pages 226–231.
-  Feng, B., Zhan, C., Liu, Z., Cheng, X., and Chang, H. (2021).  
Application of basis functions for hull form surface modification.  
*Journal of Marine Science and Engineering*, 9(9) :1005.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

3 / 8

## Quelques références V

-  Goodchild, M. F. and Janelle, D. G. (2004).  
*Spatially integrated social science*.  
Oxford University Press.
-  Gower, J. C. (1966).  
Some distance properties of latent root and vector methods used in multivariate analysis.  
*Biometrika*, 53(3-4) :325–338.
-  Hastie, T., Tibshirani, R., and Friedman, J. (2011).  
*The Elements of Statistical Learning*.  
Springer.
-  Hornik, K., Stinchcombe, M., and White, H. (1989).  
Multilayer feedforward networks are universal approximators.  
*Neural networks*, 2(5) :359–366.
-  Janelle, D. G. and Goodchild, M. F. (2011).  
Concepts, principles, tools, and challenges in spatially integrated social science.  
*The SAGE handbook of GIS and society*, pages 27–45.
-  Janowicz, K., Gao, S., McKenzie, G., Hu, Y., and Bhaduri, B. (2020).  
Geoai : spatially explicit artificial intelligence techniques for geographic knowledge discovery and beyond.
-  Jenks, G. F. and Caspall, F. C. (1971).  
Error on choroplethic maps : definition, measurement, reduction.  
*Annals of the Association of American Geographers*, 61(2) :217–244.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

5 / 8

## Quelques références IV

-  Fotheringham, A. S., Brunsdon, C., and Charlton, M. (2009).  
Geographically weighted regression.  
*The Sage handbook of spatial analysis*, 1 :243–254.
-  Freedman, D. and Diaconis, P. (1981).  
On the histogram as a density estimator : L 2 theory.  
*Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4) :453–476.
-  Freund, Y., Schapire, R. E., et al. (1996).  
Experiments with a new boosting algorithm.  
In *icml*, volume 96, pages 148–156.
-  Friedman, J. H. (2001).  
Greedy function approximation : a gradient boosting machine.  
*Annals of statistics*, pages 1189–1232.
-  Georganos, S., Grippa, T., Niang Gadiaga, A., Linard, C., Lennert, M., Vanhuysse, S., Mboga, N., Wolff, E., and Kalogirou, S. (2021).  
Geographical random forests : a spatial extension of the random forest algorithm to address spatial heterogeneity in remote sensing and population modelling.  
*Geocarto International*, 36(2) :121–136.
-  Goodchild, M. (2001).  
Issues in spatially explicit modeling.  
*Agent-based models of land-use and land-cover change*, pages 13–17.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

4 / 8

## Quelques références VI

-  Jiang, B. (2013).  
Head/tail breaks : A new classification scheme for data with a heavy-tailed distribution.  
*The Professional Geographer*, 65(3) :482–494.
-  Journel, A. G. (1989).  
*Fundamentals of geostatistics in five lessons*, volume 8.  
American Geophysical Union Washington, DC.
-  Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993).  
Multilayer feedforward networks with a nonpolynomial activation function can approximate any function.  
*Neural networks*, 6(6) :861–867.
-  Louppe, G., Wehenkel, L., Sutera, A., and Geurts, P. (2013).  
Understanding variable importances in forests of randomized trees.  
*Advances in neural information processing systems*, 26.
-  Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014).  
On the number of linear regions of deep neural networks.  
*Advances in neural information processing systems*, 27.
-  Müller, S., Schüler, L., Zech, A., and Heße, F. (2022).  
Gtools v1. 3 : a toolbox for geostatistical modelling in python.  
*Geoscientific Model Development*, 15(7) :3161–3182.
-  Nychka, D., Bandyopadhyay, S., Hammerling, D., Lindgren, F., and Sain, S. (2015).  
A multiresolution gaussian process model for the analysis of large spatial datasets.  
*Journal of Computational and Graphical Statistics*, 24(2) :579–599.

J. Ah-Pine

Masterclass IHEDD-CERDI-GDN GeoAI 24

6 / 8

## Quelques références VII

-  Oliver, M. and Webster, R. (2014).  
A tutorial guide to geostatistics : Computing and modelling variograms and kriging.  
*Catena*, 113 :56–69.
-  Quinlan, J. (1986).  
Induction of decision trees.  
*Machine Learning*, 1(1) :81–106.
-  Quinlan, J. R. (1993).  
C4.5 : programs for machine learning.  
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
-  Rey, S. J., Arribas-Bel, D., and Wolf, L. J. (2020).  
Geographic Data Science with Python.  
<https://geographicdata.science/book/intro.html>.  
Accessed October 2022.
-  Ripley, B. D. (1988).  
Statistical inference for spatial processes.  
Cambridge university press.
-  Segal, M. and Xiao, Y. (2011).  
Multivariate random forests.  
*Wiley interdisciplinary reviews : Data mining and knowledge discovery*, 1(1) :80–87.

## Quelques références VIII

-  Silverman, B. W. (2018).  
*Density estimation for statistics and data analysis*.  
Routledge.
-  Sun, K., Zhou, R. Z., Kim, J., and Hu, Y. (2024).  
Pygrf : An improved python geographical random forest model and case studies in public health and natural disasters.  
*Transactions in GIS*.
-  Torgerson, W. (1958).  
Theory and methods of scaling.
-  Wackernagel, H. (2003).  
*Multivariate geostatistics : an introduction with applications*.  
Springer Science & Business Media.
-  Wikipedia (2022).  
Carte choroplète.  
[https://fr.wikipedia.org/wiki/Carte\\_choropl%C3%A8the](https://fr.wikipedia.org/wiki/Carte_choropl%C3%A8the).  
Accessed November 2022.
-  Zhang, X., Rao, H., Wu, Y., Huang, Y., and Dai, H. (2020).  
Comparison of spatiotemporal characteristics of the covid-19 and sars outbreaks in mainland china.  
*BMC infectious diseases*, 20(1) :1–7.