# Beyond The Border

## Lissage spatial avec R

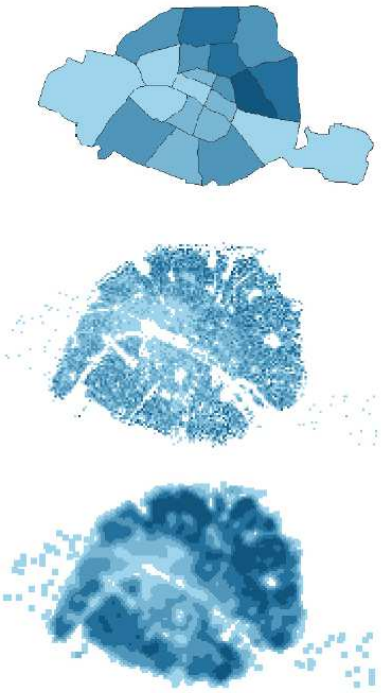

Rencontres R, Avignon 2023

Insee

Mesurer pour comprendre

# 01 Lissage spatial avec R

# 3 manières de cartographier les données ponctuelles



1. **Agrégation sur territoire administratif** : une partition irrégulière de l'espace. Plusieurs difficultés : essentiellement l'effet **MAUP** (*Modifiable areal unit problem*) ;

2. **Agrégation sur grille carroyée** : un découpage régulier de l'espace en carreaux. Par construction, ces données peuvent être très erratiques ;

3. **Lissage spatial** : une extension du carroyage consistant à décrire l'environnement d'une population dans un rayon donné.

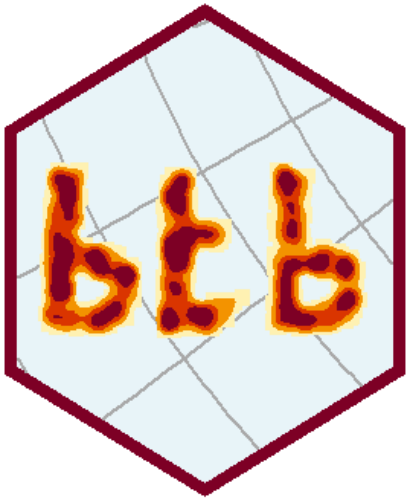
## Pourquoi avoir développé BTB ?

Il existe d'autres packages de lissage :

- `KernSmooth`
- `spatstat`...

Mais...

- Souhait de lissage **conservatif** ;
- Souhait de pouvoir gérer les **effets de bord** ;
- Souhait de pouvoir **imposer une zone de lissage**
  - pas de taux de pauvreté dans la mer ! ;
  - résolution choisie
- Souhait de faire du « lissage quantile » ;
- Souhait de disposer d'un algorithme rapide.




## btb

- Package R
- Développé depuis 2018, par l'Insee
- Mis à jour en 2022 (versions 0.2.0)
- Cœur écrit en C++ (Rcpp)

➡️ Répond aux contraintes de l'analyse urbaine

## Les paramétrages du lissage

- Un **noyau** (*kernel*) indiquant la manière d'approcher le voisinage ;
- Le **rayon de lissage**, décrivant la taille du voisinage (arbitrage biais/variance) ;
- La **résolution** ou le nombre de points sur lesquels des valeurs lissées seront estimées ;
- La **gestion des effets de bord**, pour expliciter la manière dont les frontières géographiques et la zone d'observation sont prises en compte dans le lissage.

## Dans btb...

- Méthode d'estimatation par ***noyau quadratique***
- Un paramètre de rayon de lissage
- Un paramètre de ***taille des carreaux***
- **Grille de lissage** paramétrable

# 02 Exemple : les prix immobiliers à Paris en 2021

## « Demandes de Valeurs Foncières »,

- Base de données des transactions immobilières (maisons et appartements)
- En 2021
- Région parisienne
- Géolocalisée (1 transaction = 1 point géographique)

Variables utilisées dans notre exemple :

- `id_mutation` : identifiant de chaque transaction
- `valeur_fonciere` : prix en euros
- `surface_reelle_bati` : surface en m²
- `x` : longitude (**projection Lambert 93**)
- `y` : latitude (**projection Lambert 93**)

## Chargement des données

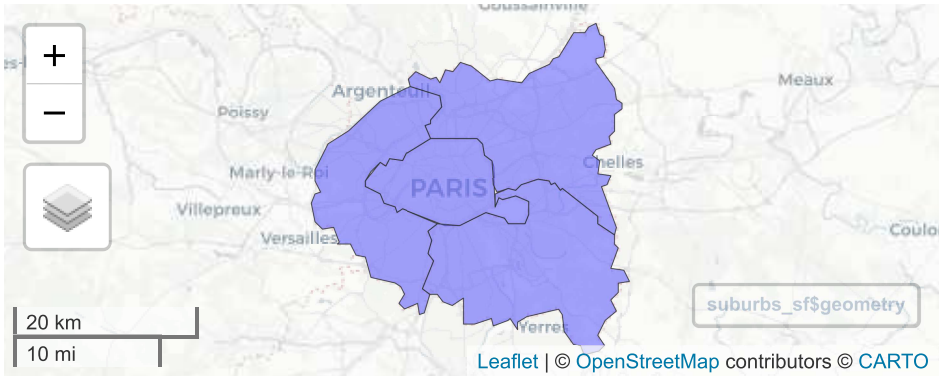
```
url_file <- url("https://minio.lab.sspcloud.fr/
dfBase <- readRDS(url_file)
dfBase <- dfBase[,c("id_mutation",
                    "valeur_fonciere",
                    "surface_reelle_bati",
                    "x","y")]
```

| id_mutation | valeur_fonciere | surface_r |
|---|---|---|
| 2021-447023 | 480000 | |
| 2021-447024 | 345000 | |
| 2021-447025 | 384000 | |
| 2021-447027 | 261900 | |
| 2021-447029 | 407200 | |

## Chargement des fonds de carte

```
url_suburbs <-"https://minio.lab.sspcloud.fr/projet-f
suburbs_sf <- sf::st_read(url_suburbs)
```

```
suburbs_sf <- suburbs_sf %>%
  rename(geometry=geom)
```



## 1. Transformation des observations en points géométriques

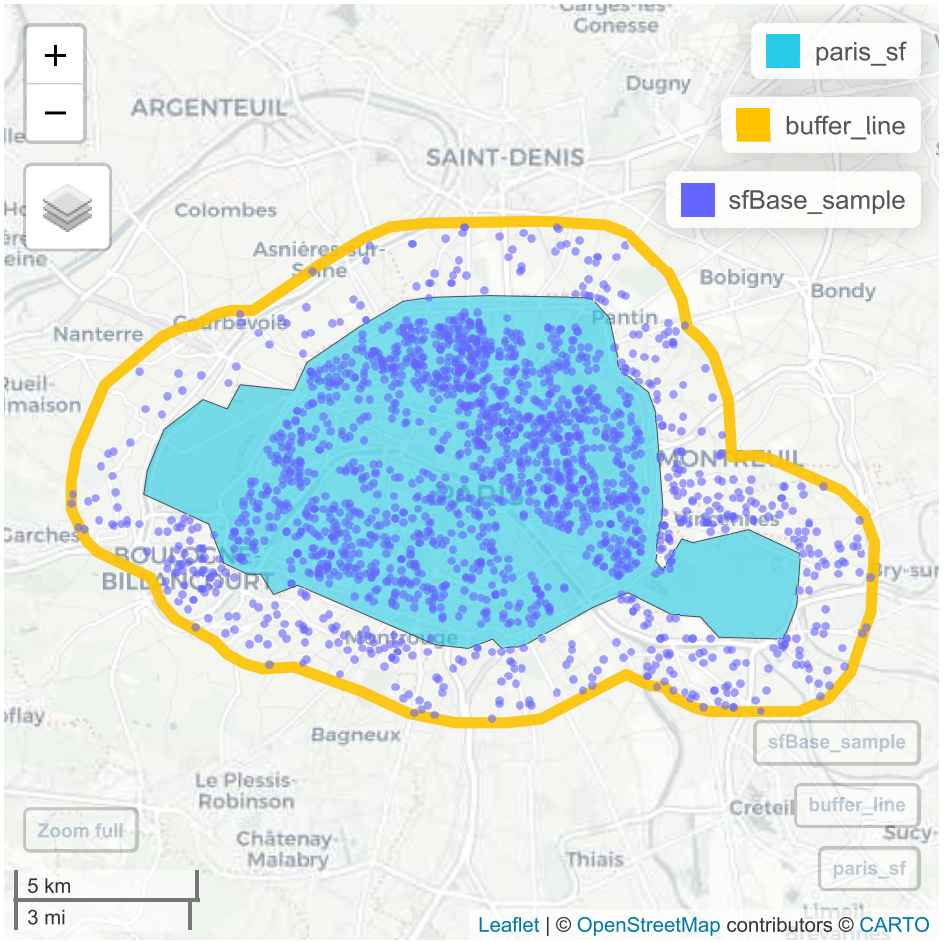
```
sfBase <- sf::st_as_sf(dfBase,
                       coords = c("x", "y"),
                       crs = 2154)
```

## 2. Zone tampon autour de Paris

```
paris_sf <- suburbs_sf %>% filter(code=="75")
buffer_sf <- st_buffer(paris_sf,dist = 2000)
```

## 3. Sélection par intersection géographique

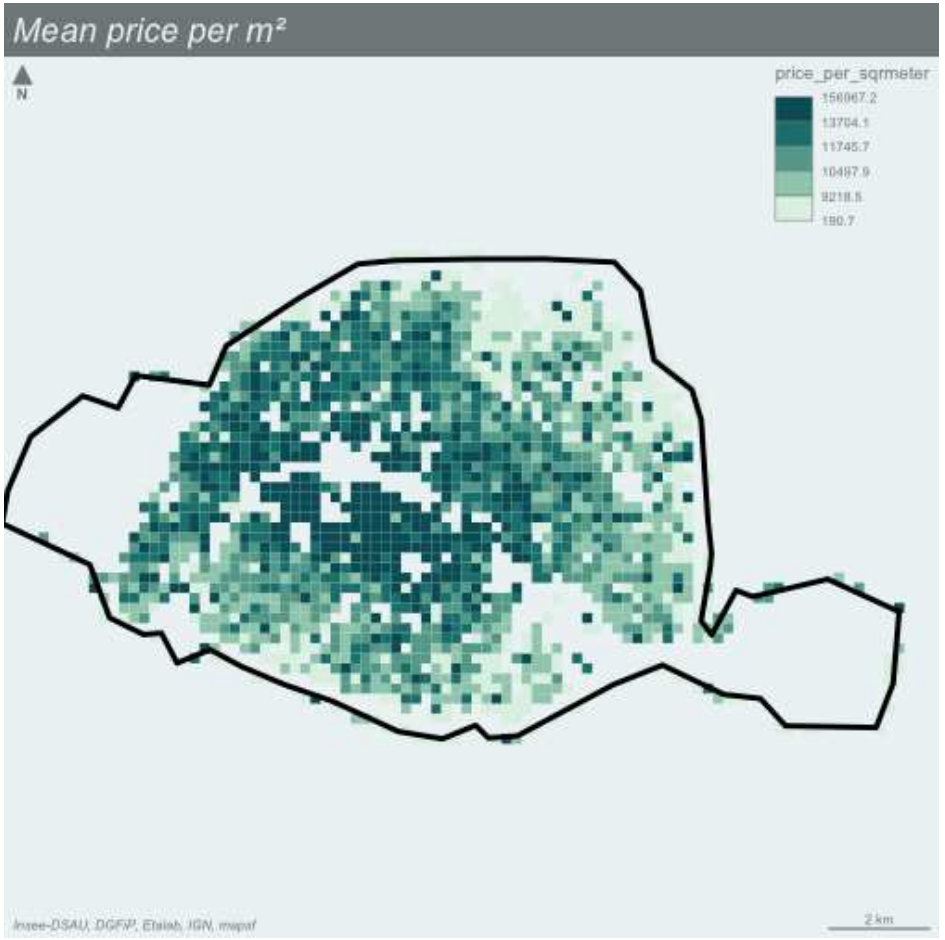
```
sfBase_buffer <- st_join(sfBase, buffer_sf,
                         left=FALSE)
```



- Pour bien maîtriser son jeu de données avant lissage : commencer par **carroyer les données** !

- Très facile avec les fonctions intégrées à `btb` :
  - `btb_add_centroids`
  - `btb_ptsToGrid`

➡ Calculons les prix moyens au mètre-carré (parmis les transactions de 2021) sur des carreaux de 50m de côté



# Lissage spatial avec `btb::btb_smooth`

- `pts` : table des points avec uniquement des variables numériques (objet geometrique ou non) ;
- `iCellSize` : Taille des carreaux en mètres (granularité de la grille des résultats) ;
- `iBandwidth` : Rayon de lissage en mètres.

```
pts <- sfBase_buffer[,c("valeur_fonciere","surface_reelle_bati")]
smooth_result <- btb::btb_smooth(pts = pts,
                                 iCellSize = 50,
                                 iBandwidth = 800)
```

| x | y | valeur_fonciere | surface_reelle_bati | geometry |
|---|---|---|---|---|
| 651475 | 6867325 | 67810.0992 | 8.3208764 | POLYGON ((651450 6867350, 6... |
| 646525 | 6862175 | 610034.2982 | 49.2174889 | POLYGON ((646500 6862200, 6... |
| 660925 | 6860375 | 144876.4983 | 17.6343084 | POLYGON ((660900 6860400, 6... |
| 642575 | 6864375 | 8850.0240 | 1.0743428 | POLYGON ((642550 6864400, 6... |
| 641675 | 6862975 | 8379.2341 | 1.2727656 | POLYGON ((641650 6863000, 6... |
| 653675 | 6859775 | 40044.2777 | 4.4192896 | POLYGON ((653650 6859800, 6... |

## Obtenir les prix au m² lissés

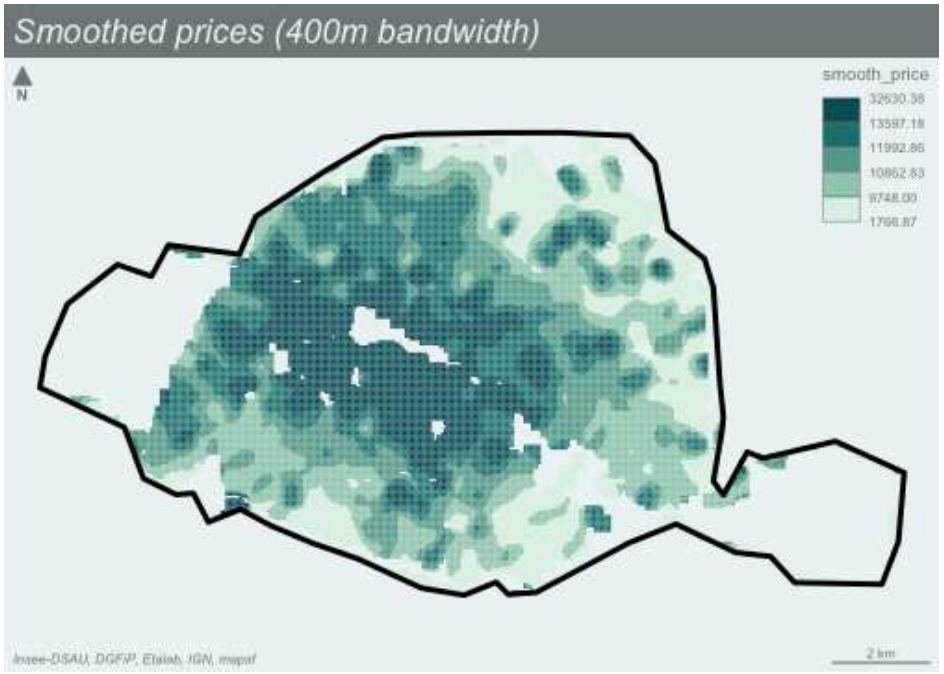
```
smooth_result <- smooth_result %>%
  mutate(smooth_price=
    valeur_fonciere / surface_reelle_bati)
```

## Ne conserver que les estimations lissées à l'intérieur de Paris

```
smooth_result <- smooth_result %>%
  st_join(paris_sf[,"geometry"],left=F)
```

## Cartographier le résultat

# Avec différents rayons de lissage (arbitrage biais-variance)





Spécificités locales
Informations structurelles
btb
STATISTICIEN-CARTOGRAPHE
imgflip.com
JAKE-CLARK.TUMBLR

## `btb` disponible sur le CRAN et sur Github

```
install.packages("btb")
```

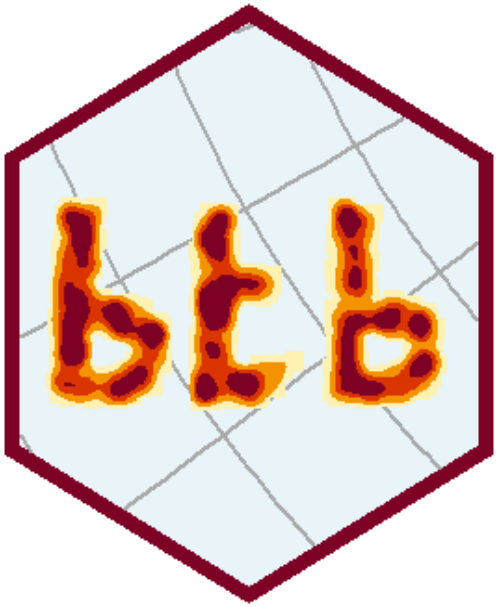
## Trouver toute la documentation :

- Site web de `btb`
- Manuel d'analyse spatiale de l'Insee
- Auto-formation

## Et n'hésitez pas à nous contacter !

- analyse-urbaine@insee.fr

## Merci pour votre attention !



## Retrouvez-nous sur :

insee.fr

Kim Antunez
Julien Pramil
Insee, France
analyse-urbaine@insee.fr

Insee
Mesurer pour comprendre