Cartograflow

Filtering Matrices for Thematic Cartography - Flow Mapping

Françoise Bahoken

2019-03-28

Introduction

`Cartograflow` is designed to filter origin-destination (OD) flow values before their thematic cartography.

This Vignette presents a selection of possible functions for filtering flows (see 5. Reference) that can be easily used to prepare the flowdata set. The spatial objects processing are those of `{sf}` (embedded) and the mapping elements are those of `{Cartography}`.

Matrice format available

– List format “L” : is a .csv 3 column flow dataset (origin, destination, flow_value) ;

– Matrice format “M” : is a .csv [n*n] flow dataset.

Use `flowtabmat()` in order to transform a “L” to “M” matrice format and vice versa.

In the case you only have a list of spatial units (code,X,Y), you can also generate an empty square matrice with `flowtabmat()`

Required datasets

– *a statistical dataset* : a .csv “L” or “M” flow dataset ;

– *a geographical dataset* :

- a *.shp* (correctly georeferenced) spatial objet (geometry : polygon) corresponding to the background of the map
- or a *.csv* consisting of a list of spatial spatial units consisting of 3 columns (code, X, Y).

Plan

1. Preparing flow dataset

1-1. Pre-processing

– `flowcarre()` is to transform an un-square to a square matrice from a list of spatial objets ID (code)

– `flowjointure()` is to performs a spatial join between a flow dataset and a map background

– `flowtabmat()` is to transform a “M” matrice format to a “L” format and vice versa

– `flowstructmat()` fixes an unpreviously ID shift in the flow dataset “M” format. If necessary this function is to be used with `flowjointure()` and `flowtabmat`.

1-2. Computing flows

It is to decide firstly to zero or not the diagonal, see `{base::diag}`.

– `flowtype()` is to compute the main types of flows from an asymmetric flow dataset (matrice or list format). The result is a bilateral gross or net flows matrice.

It is also possible to compute the matrice’s margins in order to calculate probabilities of sending and receiving flows or all kinds of indicators. Use for that the R `{base}` or `{dplyr}`

2. Flow analysis

2.1. Concentration

– `flowgini()` performs a concentration analysis of a flow dataset - to be use before `flowanalysis()`

- computes *Gini coefficient*
- plot interactive *Lorenz curve*

– `flowanalysis()` is to be used after `flowgini()` for computing a filter criterion based on *a double criterion for selecting flows* before mapping :

- level of flow thresholding, and the corresponding ;
- desired threshold level of flow’s information significativity (% of total of flow information) ;

or

- desired threshold level of flow’s features density (% of total features).

2.2. Distance travelled

You have two ways to consider the distance travelled by flows : – if you have a matrice distance, go directly to `flowreduct()` at §2.2.3 ;

– if not, you can continue here, and have to choose the type of metric (continous or ordinal)

- if you choose the continous metric, you must first join your flows’ dataset to a spatial shape, using `flowjointure()`, then use `flowdist()` as described below

2.2.1. Compute continuous distances matrices

– `flowjointure()` performs an attribute spatial join - by origin (i) and by destination (j) - between a flow dataset and a spatial shape in order to transfert the origin-destination coordinates (Xi, Yi, Xj, Yj) of the base map to the flow matrice.

– `flowdist()` Computes a *continous distance* matrice choosing metric (only euclidian for this first version)

Computes on the previrous continous distance matrice at least a `{base::summary}` in order to compute a simle distance filter criterion.

– Use `flowreduct()` to reduce the flow data set regarding distances tavelled. For filtering distances, use the *select* criterion and set :

- *dmin* for selecting values up to x km
- *dmax* ifor selecting values less than x km

2.2.2. Compute ordinal distances matrices

– `flowcontig()` compute an *ordinal distance* distance matrice based on a k-contiguity matrice. (k) is the order parameter defined by the number of borders to be crossed between origins and destinations places.

- ordre=**1** : origin-destination places are adjacent ;
- ordre=**2** : origin-destination places are distant from 2 borders ;
- ordre=**4** : origin-destination places are distant from 4 borders.

– Use after that `flowreduct()` function and directly `flowmap()` without applying the filter parameter.

You can also map the neighbourhood spatial graph using `flowmap()` without applying the filter parameter.

2.2.3. Reduces a flow matrice by an external OD matrice

– `flowreduct()` is to perform the reduction of the flow dataset according to another matrice (especially a matrice distance)

3. Flow mapping

`flowmap()` is to create a layer of lines, plot them, using a flow dataset (dataframe format) and a spatial shape (shapefile format).

3.1. Filtering flows (if necessary)

– **filter** is to choose to map all or selected flow values.

- If filter is “FALSE”, no filter will be applied (ie all the N(N-1) theoretical values are drawn).
- else (If filter is “TRUE”) it is possible to applied a filter criterion before plotting flows.

3.2. Filtering flow values

If **filter** is “TRUE”, it is possible to threshold flow values.

– **threshold** is to apply a numerical global (across the whole matrice) filter criterion.

3.3. Filtering flow value (dealing with strongest flows)

- use `{flowanalysis}` after `{flowgini}` and compute the *critflow parameter* to estimate a filter criterion based on flow values.

3.3. Filtering flow features (dealing with density)

- use `{flowanalysis}` after `{flowgini}` and compute the *critlink parameter* to estimate a selection criterion based on the density of the flows' features.

3.4. Setting up the flow features

Use `flowtabmat()` for setting graphic parameters for arrows

– **taille** is to modify the width of the origin-destination lines.

– **a.head** is to add a head of the line (transform it to arrows) to be drawn from places (in, out, in and out).

– **a.length** is to parameter the length of the arrow head in relation to the body of the arrow.

– **a.angle** is to parameter an angle from the shaft of the arrow to the edge of the arrow head.

– **a.col** is to parameter the color of the arrows.

4. Cartograflow : example of applications

4.1 Load packages

```
rm(list=ls())

library(dplyr)
library(cartography)
library(cartograflow)
```

4.2 Data

– Statistical dataset : extraction from the nationale file : "Mobilités professionnelles en 2015 : déplacements domicile - lieu de travail" from the french census (Recensement de la population) - Base flux de mobilité.

-URL : https://www.insee.fr/fr/statistiques/fichier/3566008/rp2015_mobpro_txt.zip

- Citation : INSEE - RP MOBPRO, 2015.

– Geographical dataset :

- municipalities : IGN, GEOFLA 2015 v2.1 Communes France Métropolitaine.
- Territories : APUR.

- geographical ID : INSEE, 2016.
- Citations : IGN, APUR, UMS 2414 RIATE, 2018.

4.2.1 Load Statistical information

```
data<-read.csv2("./data/MOBPRO_ETP.csv",
                header=TRUE,
                sep=";",
                stringsAsFactors=FALSE,
                encoding="UTF-8",
                dec=".",
                check.names=FALSE)
head(data)
```

```
##    i   j    Fij count
## 1 T1  T1 291058   351
## 2 T1 T10   8297    43
## 3 T1 T11   3889    13
## 4 T1 T12  17064    77
## 5 T1  T2  12163    52
## 6 T1  T3  32682    55
```

```
str(data)
```

```
## 'data.frame':    121 obs. of  4 variables:
##  $ i    : chr  "T1" "T1" "T1" "T1" ...
##  $ j    : chr  "T1" "T10" "T11" "T12" ...
##  $ Fij  : int  291058 8297 3889 17064 12163 32682 81129 16622 25317 1962 ...
##  $ count: int  351 43 13 77 52 55 134 63 53 14 ...
```

```
# Variable typing
data$i<-as.character(data$i)
data$j<-as.character(data$j)
data$Fij<-as.numeric(data$Fij)
data$count<-as.numeric(data$count)
```

```
# Loading a list of geo codes
ID_CODE<-read.csv2("./data/COD_GEO_EPT.csv",
                   header=TRUE,
                   sep=";",
                   stringsAsFactors=FALSE,
                   encoding="UTF-8",
                   dec=".",
                   check.names=FALSE)
```

```
head(ID_CODE)
```

```
##   COD_GEO_EPT                    LIBELLE_EPT
## 1          T1                          Paris
## 2          T2          Vallee Sud Grand Paris
## 3          T3 Grand Paris Seine Ouest (GPSO)
## 4          T4          Paris Ouest la Defense
## 5          T5            Boucle nord de Seine
## 6          T6                  Plaine commune
```

```
CODE<-ID_CODE%>%
  select(COD_GEO_EPT)

colnames(CODE)<-c("CODGEO")

head(CODE)
```

```
##   CODGEO
## 1     T1
## 2     T2
## 3     T3
## 4     T4
## 5     T5
## 6     T6
```

4.2.2 Pre-processing flow dataset

```
tabflow<-data%>%
  select(i,j,Fij)

# Change matrice format (if necessary)
matflow <-flowtabmat(tabflow,matlist="M")
```

```
## Using Fij as value column: use value.var to override.
```

```
## great:your matrix is square!
```

```
head(matflow[1:4,1:4])
```

```
##         T1   T10   T11   T12
## T1  291058  8297  3889 17064
## T10  73743 19501 11707  4931
```

```
## T11  22408  9359 12108  6084
## T12  68625  1906  7269 46515
```

```
dim(matflow)
```

```
## [1] 12 12
```

The Warning said that the matrice is square (dimension is : 12*12). If it was not, Use `{flowcarre}`to square it (and to close it)

4.2.3 Computing flow dataset

Zero the diagonal and change matrice format (list to matrice)

```
# Zero the diagonal of matrice format (if necessary)
diag(matflow) <- 0
head(matflow[1:4,1:4])
```

```
##        T1  T10   T11   T12
## T1      0 8297  3889 17064
## T10 73743    0 11707  4931
## T11 22408 9359     0  6084
## T12 68625 1906  7269     0
```

```
# Change matrice to list format
tabflow<-flowtabmat(tab=matflow,
                    matlist="L")
head(tabflow)
```

```
##     i  j ydata
## 1  T1 T1     0
## 2 T10 T1 73743
## 3 T11 T1 22408
## 4 T12 T1 68625
## 5  T2 T1 47427
## 6  T3 T1 45772
```

```
colnames(tabflow)<-c("i","j","Fij")
```

4.2.4 Computing main types of flow dataset

Compute bilateral flow volume and bilateral flow balance from a matrice or a list format of observed flows,

```
# Compute bilateral flow volume - from a "M" format
matflow_vol<-flowtype(matflow,
                      format="M",
                      "bivolum")

# Compute bilateral flow volume - from a "L" format

# FSij will be the gross Fij flow values
tabflow_vol<-flowtype(tabflow,
                     format="L",
                     "bivolum")
head(tabflow_vol)
```

```
##    i   j  FSij
## 1 T1  T1     0
## 2 T1 T10 82040
## 3 T1 T11 26297
## 4 T1 T12 85689
## 5 T1  T2 59590
## 6 T1  T3 78454
```

```
# Compute bilateral flow balance - from a "L" format

# FDij will be the net Fij flow values
tabflow_net<-flowtype(tabflow,
                      format="L",
                      "bisold")
head(tabflow_net)
```

```
##    i   j  FDij
## 1 T1  T1     0
## 2 T1 T10 65446
## 3 T1 T11 18519
## 4 T1 T12 51561
## 5 T1  T2 35264
## 6 T1  T3 13090
```

```
# Compute all types of bilateral flows, in one 6 columns "L"format matrice
tabflow_all<-flowtype(tabflow,
                      format="L",
                      x="all")
head(tabflow_all)
```

```
##    i   j   Fij   Fji  FSij  FDij
## 1 T1  T1     0     0     0     0
```

```
## 2 T1 T10  8297 73743 82040 65446
## 3 T1 T11  3889 22408 26297 18519
## 4 T1 T12 17064 68625 85689 51561
## 5 T1  T2 12163 47427 59590 35264
## 6 T1  T3 32682 45772 78454 13090
```

```
# Compute flow asymetry
tabflow_all$FAsy<-(tabflow_all$FDij / tabflow_all$FDij)*100
```

4.3 Flow mapping

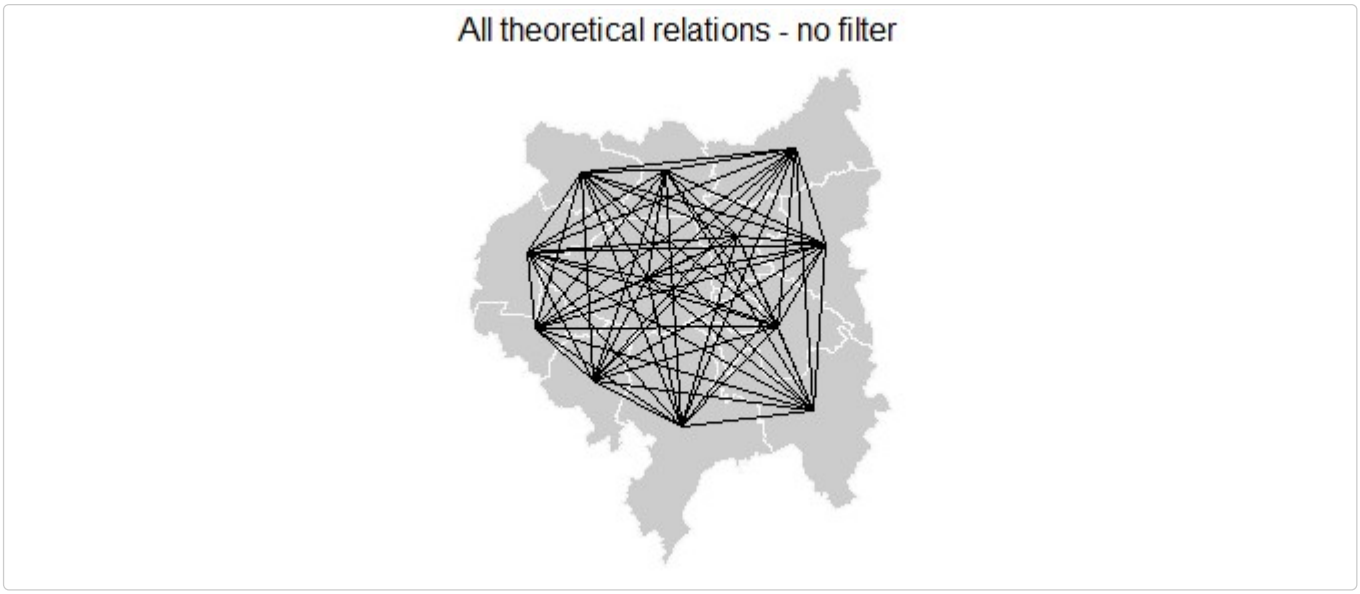
4.3.1 Direct flow mapping

Plot all origin-destination links without any filtering criterion.

```
knitr::opts_chunk$set(fig.width=6, fig.height=6)

par(mar=c(0,0,1,0))
extent <- c(2800000, 1340000, 6400000, 4800000)
resolution<-300

# Plot all theoretical OD links
flowmap(tab = tabflow,
        format="L",
        fdc="./data/MGP_TER.shp",
        code="EPT_NUM",
        filter=FALSE) #no filter criterion
```

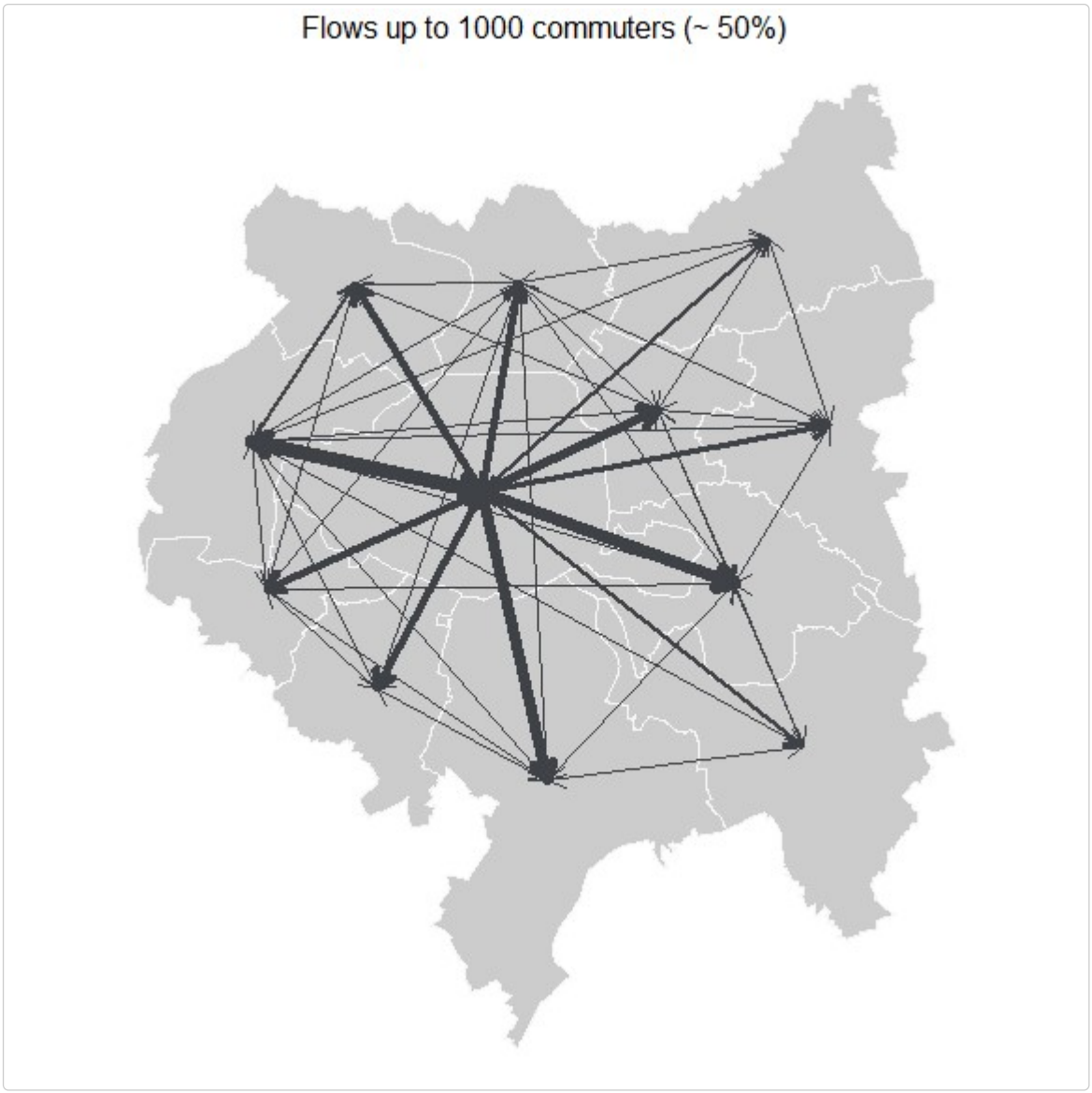
```
## All theorical links are plotted
```

```
mtext("All theoretical relations - no filter",side = 3)
```



Plot valued origin-destination links (> 1000)

```
par(mar=c(0,0,1,0))

#Plot existing relations (up to 1000 commuters)
flowmap(tab = tabflow,
        format="L",
        fdc="./data/MGP_TER.shp",
        code="EPT_NUM",
        filter=TRUE,        #add filter
        a.col="#3f4247",
        threshold=1000,
        taille=8,
        a.head = 1,
        a.length = 0.11)

mtext("Flows up to 1000 commuters (~ 50%)",side = 3)
```

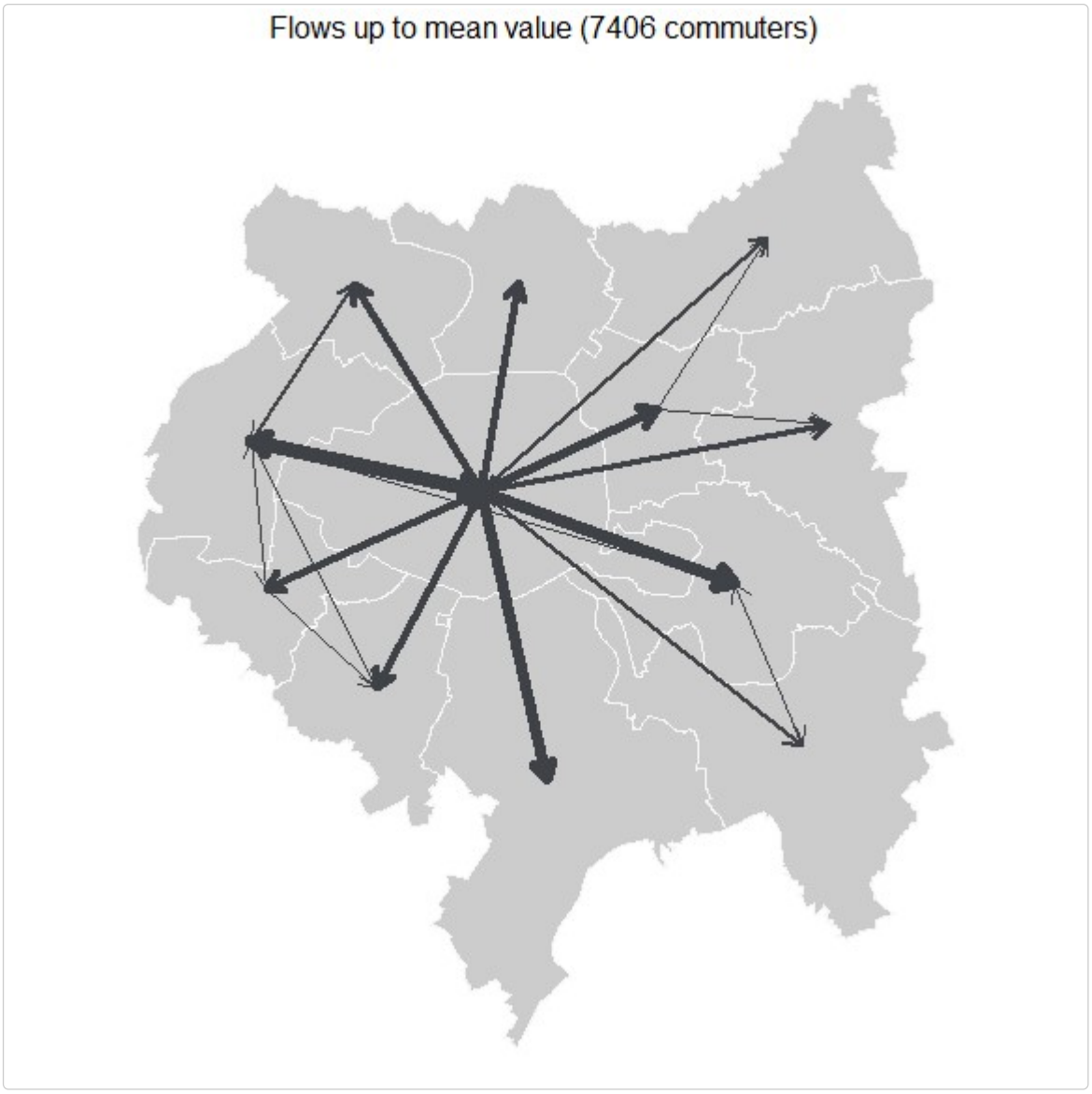


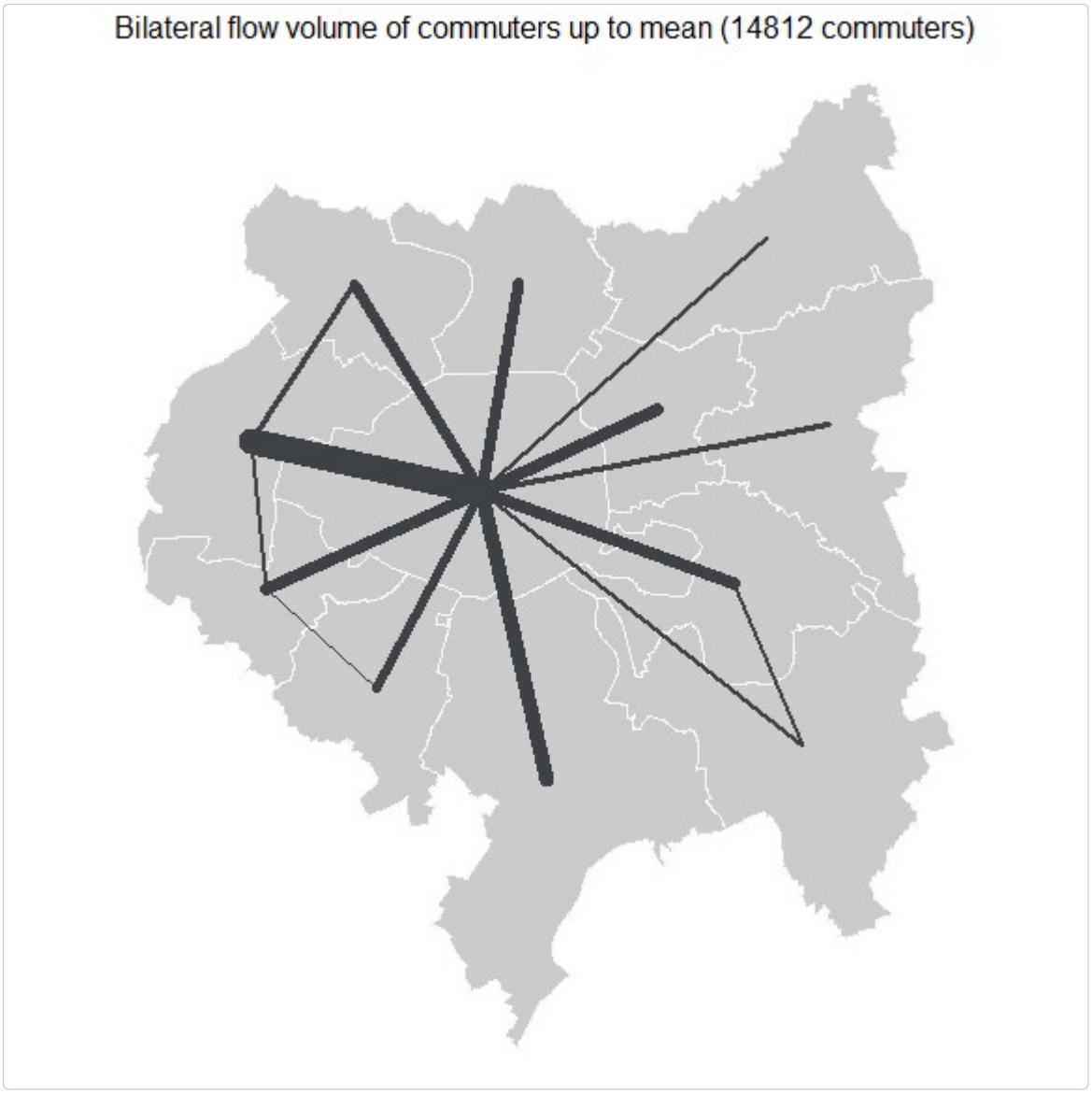
4.5 Mapping a robust flow selection

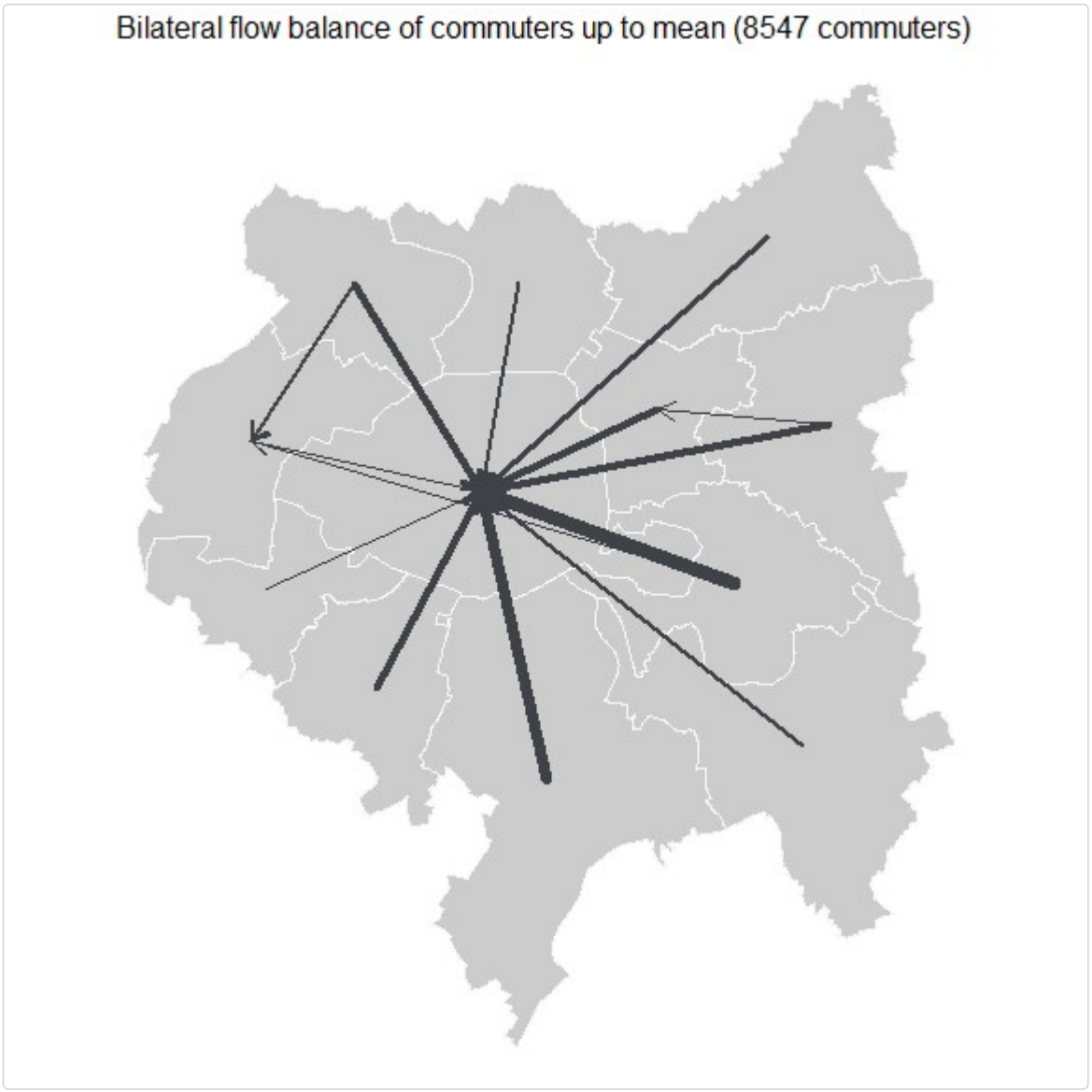
4.5.0 Statistical parameter

```
# Plot flow value up to a global filter criterion

par(mar=c(0,0,1,0))
extent <- c(2800000, 1340000, 6400000, 4800000)
resolution<-300

# Mapping filtered observed commuters
flowmap(tab = tabflow,
              format="L",
              fdc="./data/MGP_TER.shp",
              code="EPT_NUM",
              filter=TRUE,
```

```
                a.col="#3f4247",
                threshold=7406,  # Mean=7406
                taille=8,
                a.head = 1,
                a.length = 0.11)

mtext("Flows up to mean value (7406 commuters)",side = 3)

# Bilateral flow volum of commuters
flowmap(tab = tabflow_vol,
                format="L",
                fdc="./data/MGP_TER.shp",
                code="EPT_NUM",
                filter=TRUE,
                a.col="#3f4247",
                threshold=14812.4,  # Mean=14812.4
                taille=14,
                a.head = 0,
                a.length = 0.11)

mtext("Bilateral flow volume of commuters up to mean (14812 commuters)",side = 3)

# Bilateral flow balance of commuters
flowmap(tab = tabflow_net,
                format="L",
                fdc="./data/MGP_TER.shp",
                code="EPT_NUM",
                filter=TRUE,
                a.col="#3f4247",
                threshold=8547,  # Mean=8547
                taille=8,
                a.head = 1,
                a.length = 0.11)

mtext("Bilateral flow balance of commuters up to mean (8547 commuters)",side = 3)
```


Flows up to mean value (7406 commuters)


Bilateral flow volume of commuters up to mean (14812 commuters)



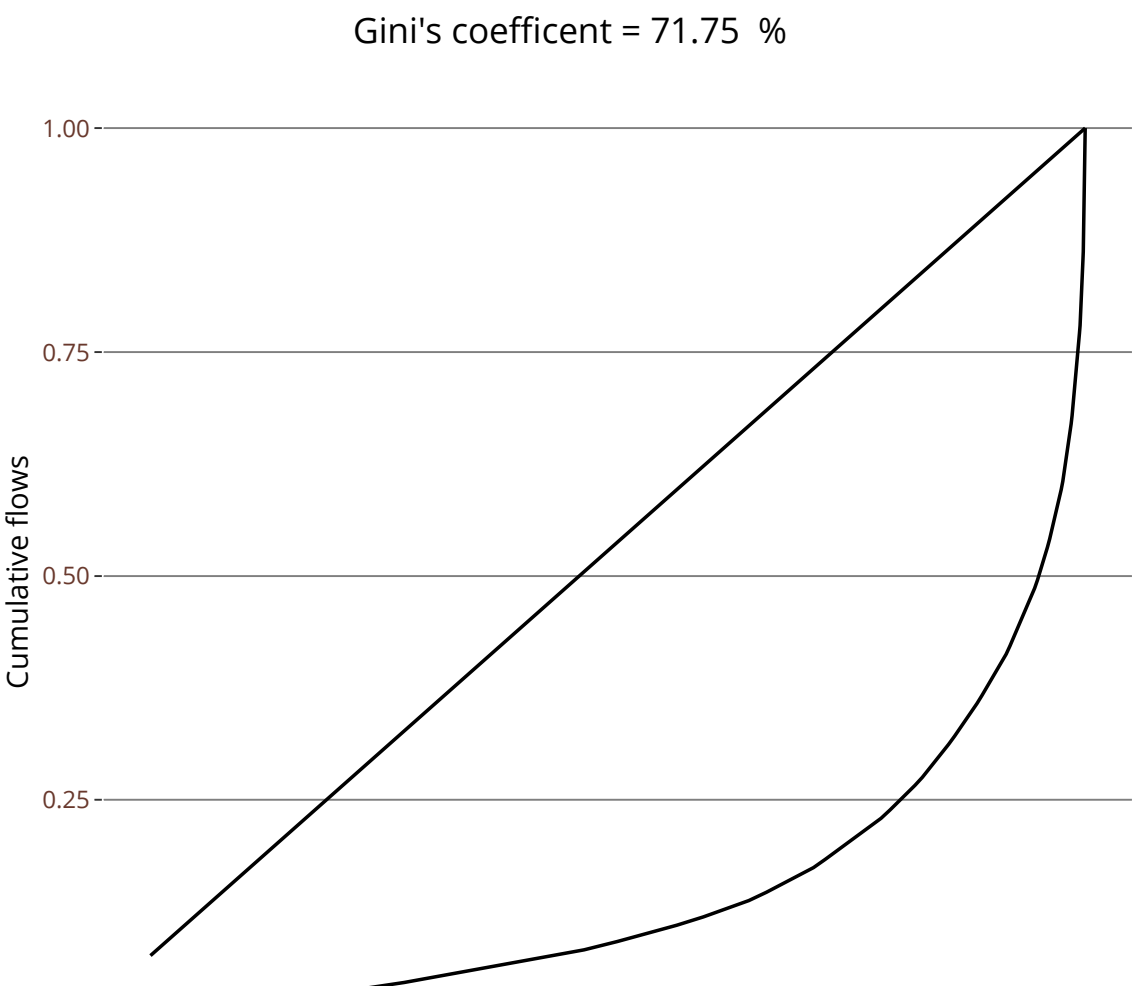
4.5.1 Concentration analysis

```
head(tabflow,3)
```

```
##     i  j   Fij
## 1  T1 T1     0
## 2 T10 T1 73743
## 3 T11 T1 22408
```

```
# 1- Computes Gini's coefficent
#--------------------
tab_gini<-flowgini(tabflow,
                   format="L",
                   origin="i",
                   dest="j",
```

```
                    valflow="Fij",
                    fdc = "./data/MGP_TER.shp",
                    code="EPT_NUM",
                    lorenz.plot = FALSE)


## Gini's coefficent = 71.75 %


#Interpretation ; The flows are quite concentrated on a few links, the Gini coefficent is equal to
71%

# 2- Plot Lorenz curve
#--------------------
#head(tab_gini)

flowgini(tab_gini,
         format="L",
         origin="i",
         dest="j",
         valflow="ydata",
         fdc = "./data/MGP_TER.shp",
         code="EPT_NUM",
         lorenz.plot = TRUE)
```



```
# 3- Compute critflow parameter and flowmap
#------------------------------
#critflow = 0.8
flowanalysis(tab_gini,
             critflow = 0.8,
             result = "signif")
```

```
## [1] "threshold =  11238  ---  flows =  80 % ---  links =  22.94 %"
```

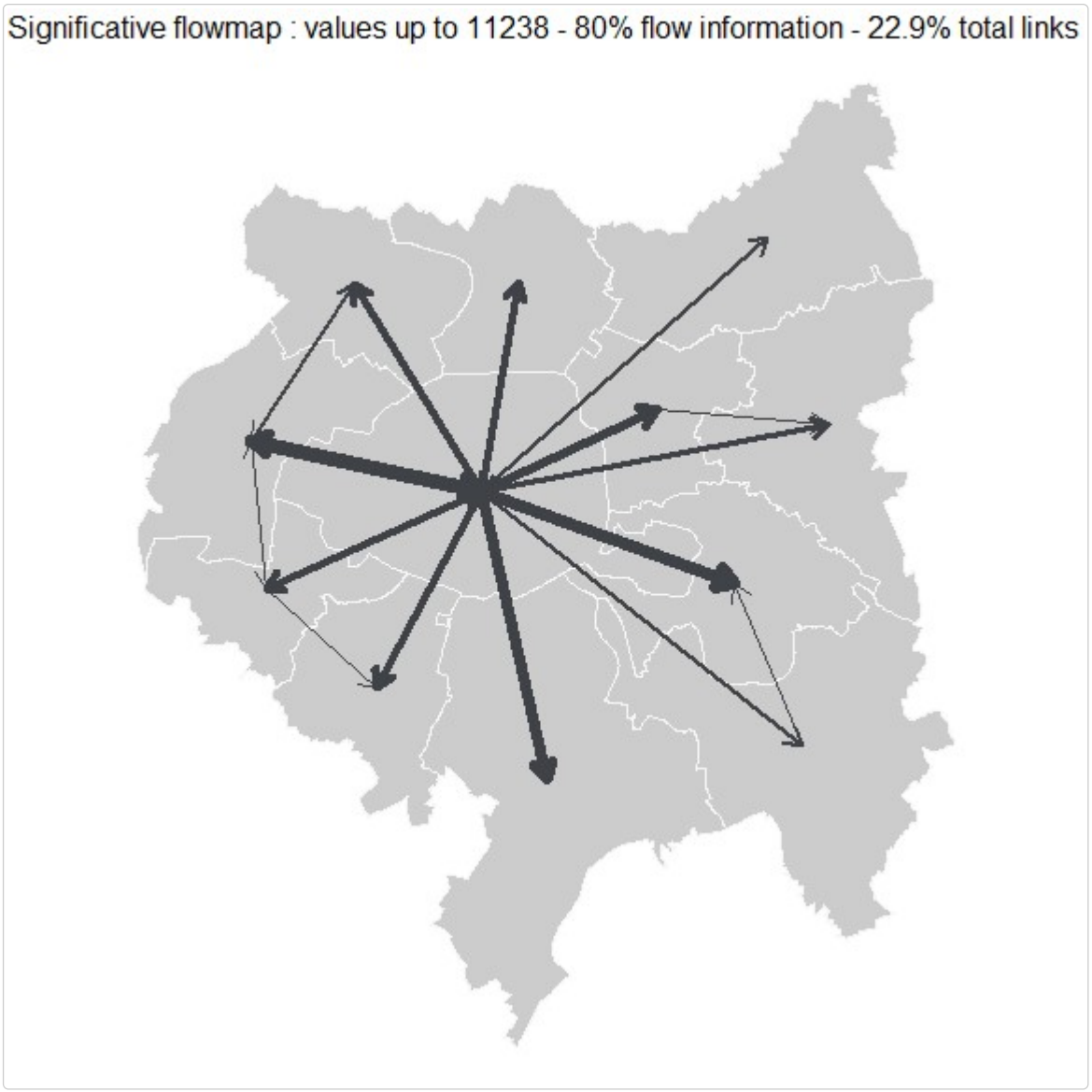
 # Interpretation : Flow values up to 11238 are the 80% largest one corresponding to 22,94% of the total links.

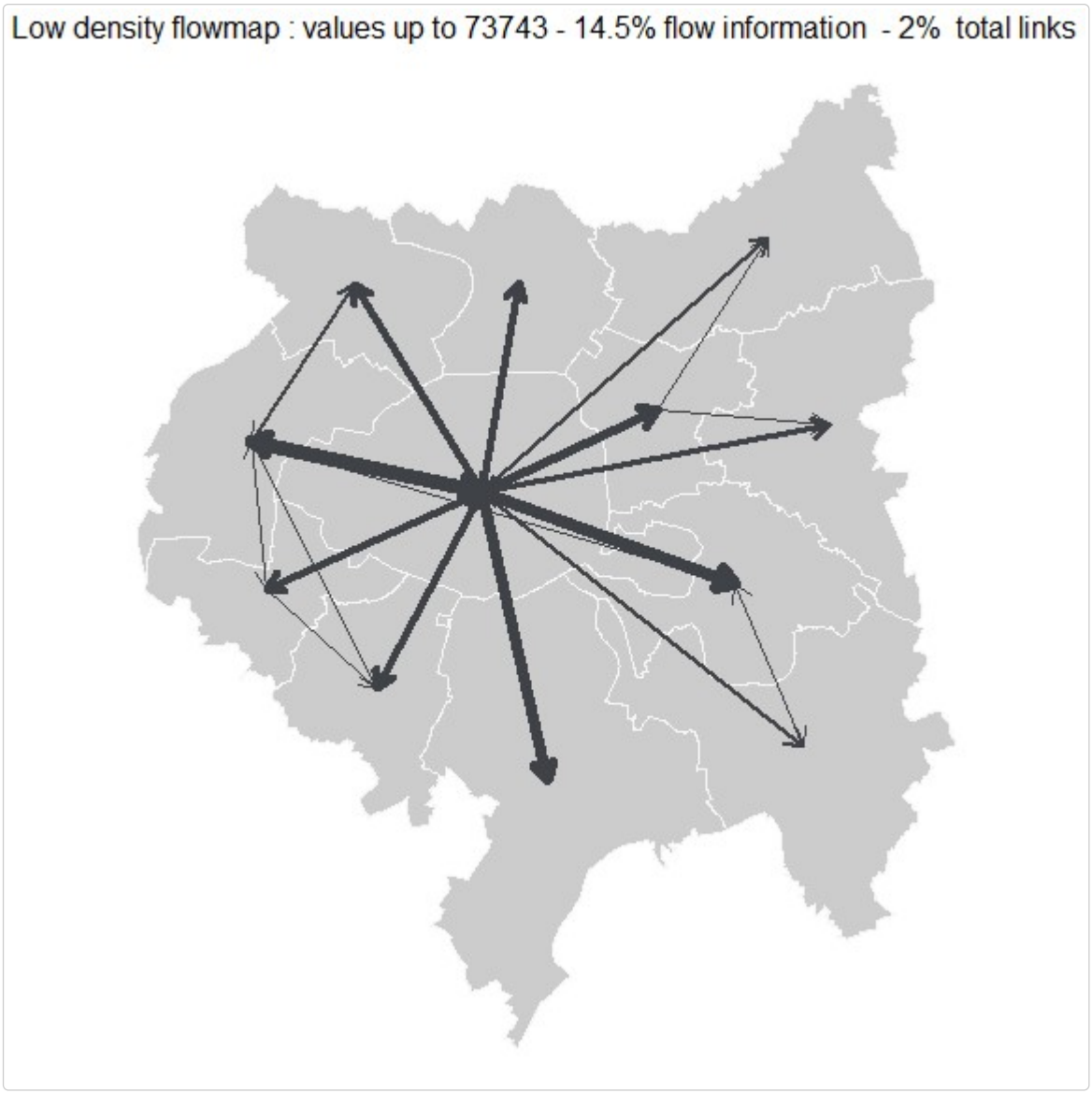
```
#threshold = 11238

par(mar=c(0,0,1,0))
flowmap(tabflow,
        format="L",
        fdc="./data/MGP_TER.shp",
        code="EPT_NUM",
        filter=TRUE,
        threshold=11238,
        taille=8,
        a.head = 1,
        a.length = 0.11,
        a.angle = 30,
        a.col="#3f4247")

mtext("Significative flowmap : values up to 11238 - 80% flow information - 22.9% total links",side
= 3)

# 4- Compute critlink parameter and flowmap
#------------------------------

flowanalysis(tab_gini,
             critlink = 0.02,
             result = "density")
```

```
## [1] "threshold =  73743  ---  flows =  14.52 % ---  links =  2 %"
```

 # Interpretation : Flows up to 73743 are the 14.5% largest one corresponding to 2 % of the total links

```
# Plot 2 % of the total features equals to select flow greater than 73743 commuters
par(mar=c(0,0,1,0))

flowmap(tab = tabflow,
        format="L",
        fdc="./data/MGP_TER.shp",
        code="EPT_NUM",
        filter=TRUE,
        a.col="#3f4247",
        threshold=7343,
        taille=8,
        a.head = 1,
        a.length = 0.11,
        a.angle = 30
        )

mtext("Low density flowmap : values up to 73743 - 14.5% flow information  - 2%  total links",side
= 3)
```

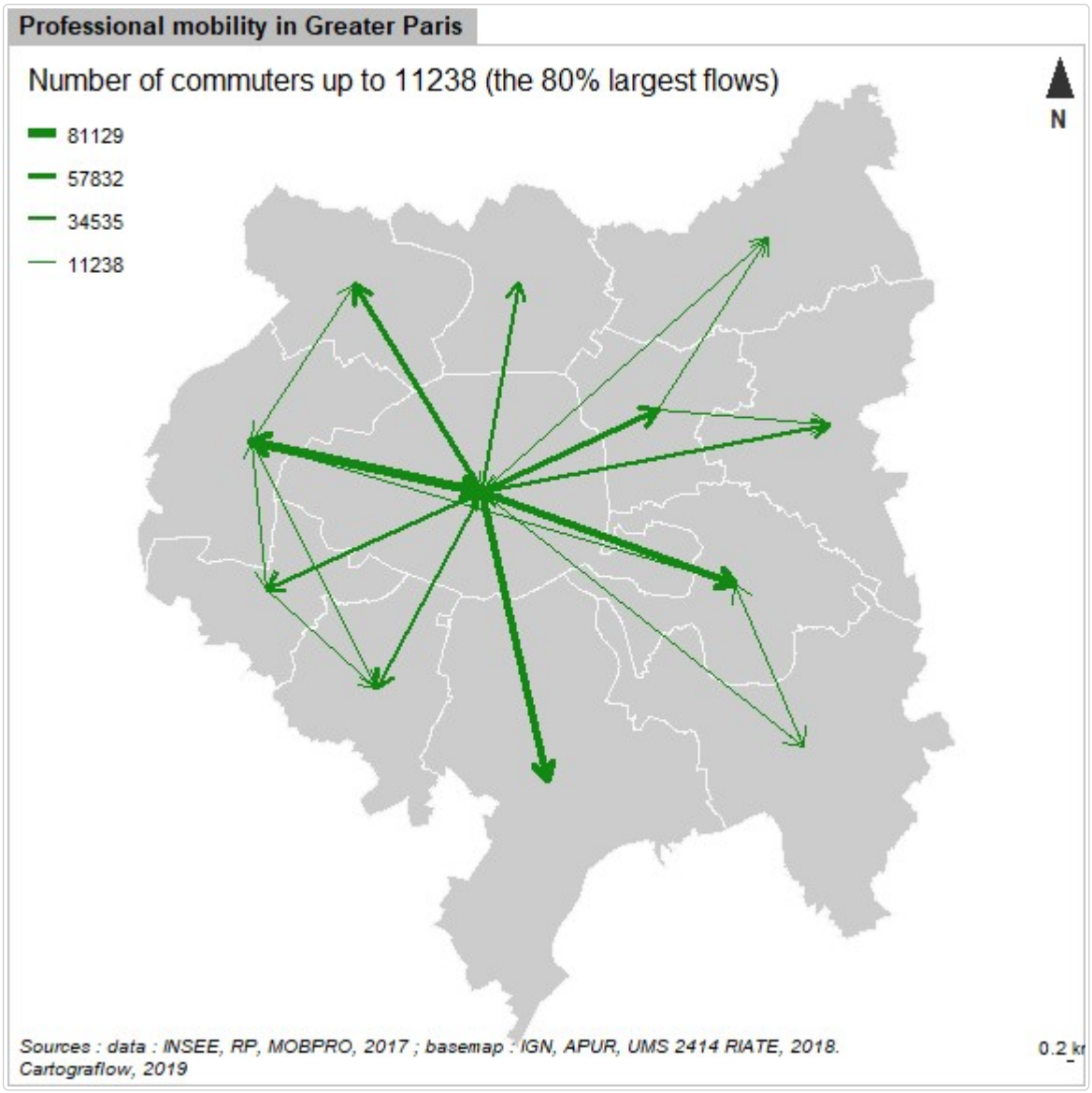

Significative flowmap : values up to 11238 - 80% flow information - 22.9% total links



4.5.3 Concentration's Flowmap

```
par(mar=c(0,0,1,0))
extent <- c(2800000, 1340000, 6400000, 4800000)
resolution<-300

# Final flowmap customized
flowmap(tabflow,
        format="L",
        fdc="./data/MGP_TER.shp",
        code="EPT_NUM",
        filter=TRUE,
        threshold=7343,
        taille=5,
        a.head = 1,
        a.length = 0.11,
```

```
        a.angle = 30,
        a.col="#138913"
        )

# Legend
legendPropLines(pos="topleft",
                title.txt="Number of commuters up to 11238 (the 80% largest flows)",
                title.cex=1,
                cex=0.8,
                values.cex= 0.7,
                var=c(11238,max(tabflow$Fij)),
                lwd=5,
                frame = FALSE,
                col="#138913",
                values.rnd = 0
                )

layoutLayer(title = "Professional mobility in Greater Paris",
            coltitle ="black",
            author = "Cartograflow, 2019",
            sources = "Sources : data : INSEE, RP, MOBPRO, 2017 ; basemap : IGN, APUR, UMS 2414
RIATE, 2018.",
            scale = 0.2,
            tabtitle = TRUE,
            frame = TRUE,
            north(pos = "topright"),
            col = "grey"
                        )
```



4.6 Thresholding flows by distance travelled

4.6.1 Continuous distance

```
head(tabflow)
```

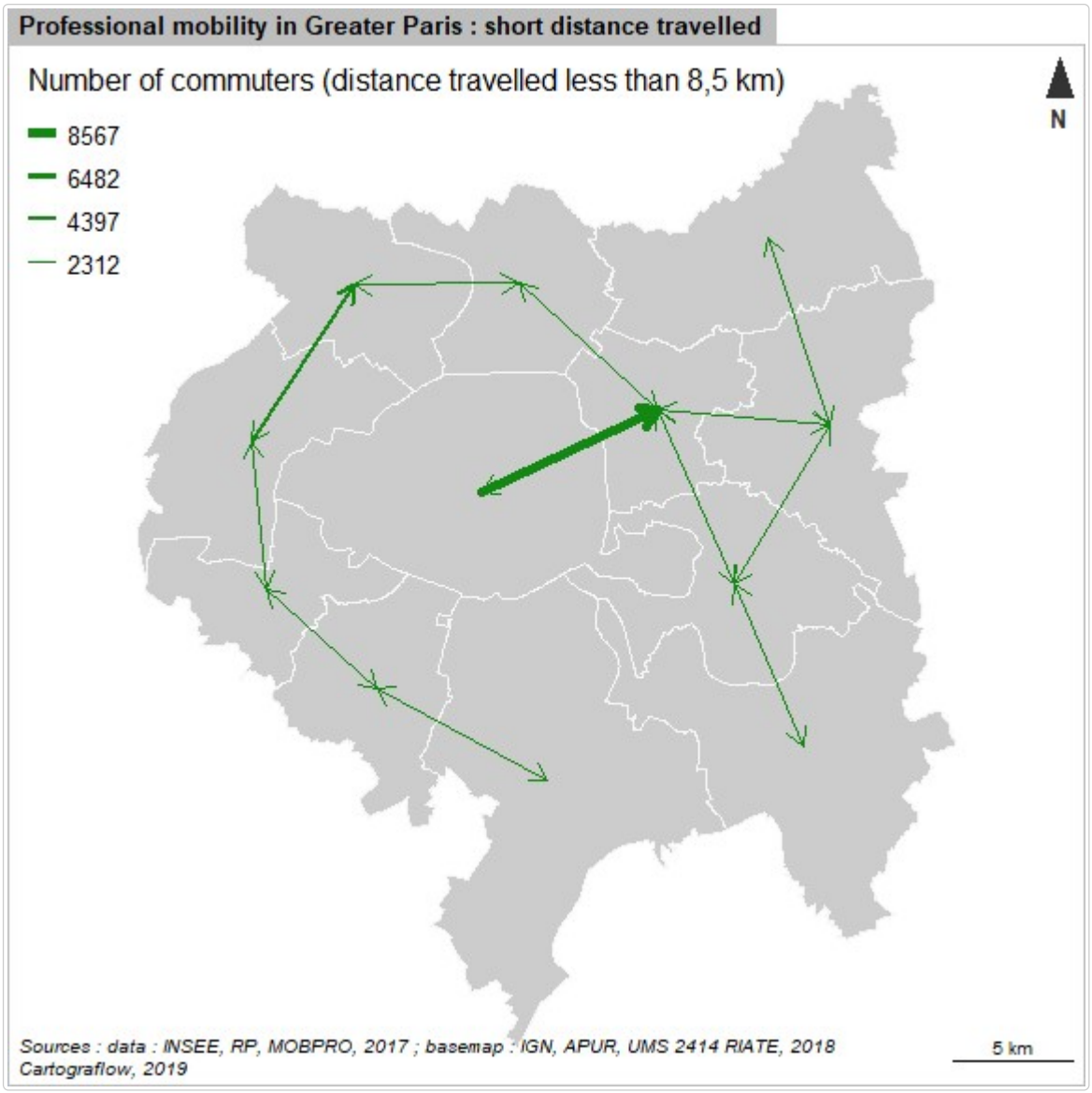
```
##     i  j   Fij
## 1  T1 T1     0
## 2 T10 T1 73743
## 3 T11 T1 22408
## 4 T12 T1 68625
## 5  T2 T1 47427
## 6  T3 T1 45772
```

```
tab<-flowjointure(tabflow,
                  "./data/MGP_TER.shp",
                  "EPT_NUM")

tab.distance<-flowdist(tab,
                       dist.method = "euclidian",
                       result = "dist")
head(tab.distance)
```

```
##     i  j  distance
## 1  T1 T1     0.000
## 2 T10 T1 11367.443
## 3 T11 T1 17285.787
## 4 T12 T1 12460.261
## 5  T2 T1  9367.284
## 6  T3 T1  9951.666
```

```
#reduce the flow dataset from a selected distance travelled < 8.5 km
library(rlang)

tab.flow<-flowreduct(tab,
                     tab.distance,
                     metric = "continous",
                     select = "dmax", #max distance parameter
                     d = 8567)        #max distance value - Q1 : 8567 km


## your dataset must be have three columns : origine, destination and flow or another data


#select for all i,j flow values up to 0
flow.d<-tab.flow%>%
       select(i,j,flowfilter)%>%
        filter(flowfilter !=0)

#Flowmap : flow travelled less than 8.5 km  (Q1)

par(mar=c(0,0,1,0))

extent <- c(2800000, 1340000, 6400000, 4800000)
resolution<-300

flowmap(flow.d,format="L",
       "./data/MGP_TER.shp",
       "EPT_NUM",
        filter = TRUE,
        taille = 5,
        a.col="#138913",
```

```
        a.length = 0.11,
        a.head =1)


## warning : you use the default threshold= 1


legendPropLines(pos="topleft",
                title.txt="Number of commuters (distance travelled less than 8,5 km)",
                title.cex=1,
                cex=0.8,
                values.cex= 0.8,
                var=c(min(flow.d$flowfilter),8567),
                col="#138913",
                lwd=5,
                frame = FALSE,
                values.rnd = 0
                )
# Habillage

layoutLayer(title = "Professional mobility in Greater Paris : short distance travelled",
            author = "Cartograflow, 2019",
            sources = "Sources : data : INSEE, RP, MOBPRO, 2017 ; basemap : IGN, APUR, UMS 2414
RIATE, 2018",
            scale = 5,
            tabtitle = TRUE,
            frame = TRUE,
            north(pos = "topright"),
            col = "grey",
            coltitle ="black")
```



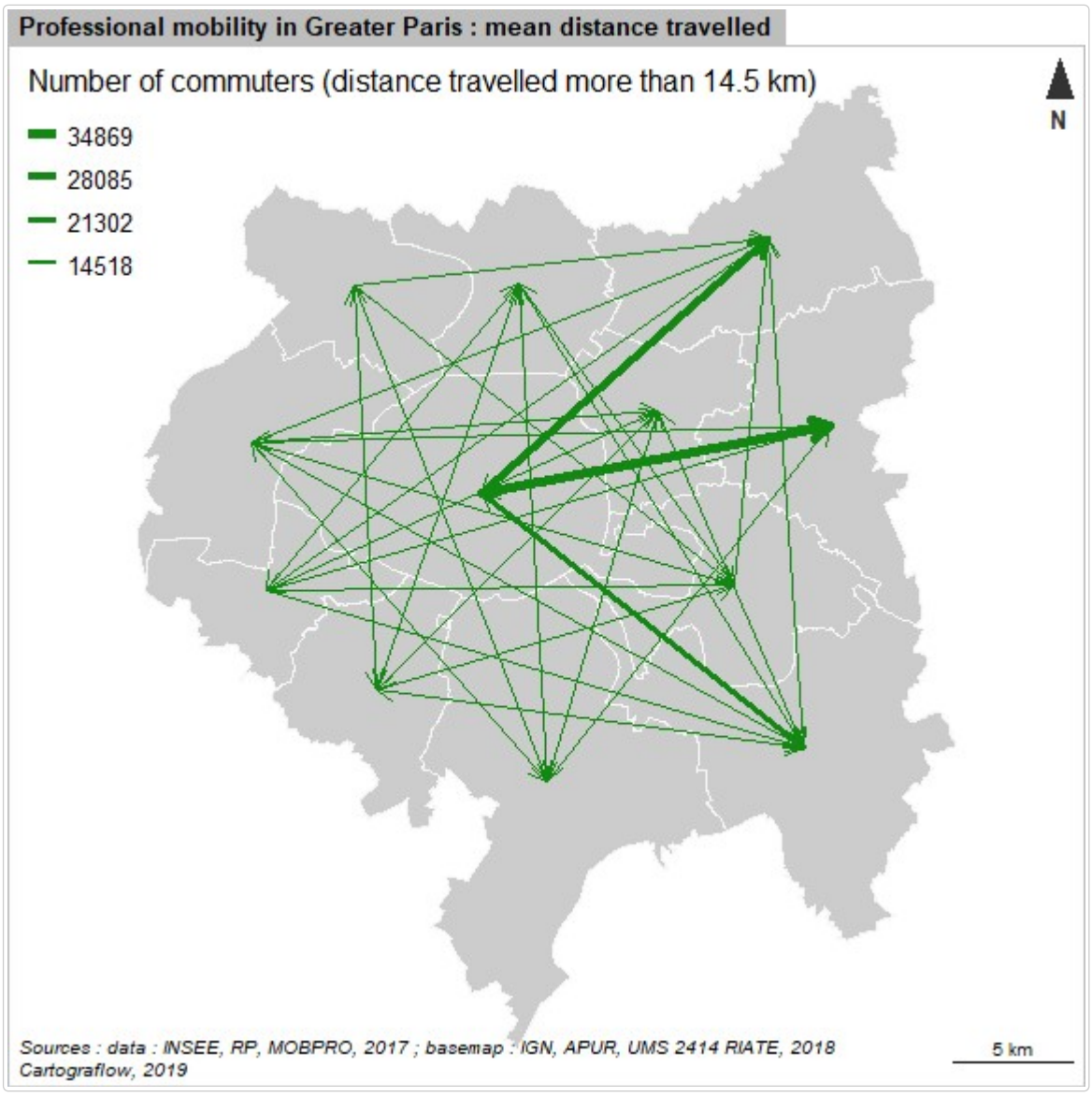

```
head(tabflow)
```

```
##     i  j   Fij
## 1  T1 T1     0
## 2 T10 T1 73743
## 3 T11 T1 22408
## 4 T12 T1 68625
## 5  T2 T1 47427
## 6  T3 T1 45772
```

```
tab<-flowjointure(tabflow,
                  "./data/MGP_TER.shp",
                  "EPT_NUM")

tab.distance<-flowdist(tab,
                       dist.method = "euclidian",
```

```
                      result = "dist")

tab.flow<-flowreduct(tab,
                     tab.distance,
                     metric = "continous",
                     select = "dmin",
                     d = 14518)        #Q2 : 14518 km - Q3:19234 km


## your dataset must be have three columns : origine, destination and flow or another data


#select for all i,j flow values up to 0
flow.d<-tab.flow%>%
       select(i,j,flowfilter)%>%
        filter(flowfilter !=0)


#Flowmap : flow travelled up to (Q3)

par(mar=c(0,0,1,0))

extent <- c(2800000, 1340000, 6400000, 4800000)
resolution<-300

flowmap(flow.d,format="L",
       "./data/MGP_TER.shp",
       "EPT_NUM",
        filter = TRUE,
        taille = 5,
        a.col="#138913",
        a.length = 0.11,
        a.head =1)


## warning : you use the default threshold= 1


legendPropLines(pos="topleft",
                title.txt="Number of commuters (distance travelled more than 14.5 km)",
                title.cex=1,
                cex=0.8,
                values.cex= 0.8,
                var=c(14518, max(flow.d$flowfilter)),
                col="#138913",
                lwd=5,
                frame = FALSE,
                values.rnd = 0
                )
# Habillage
```

```
layoutLayer(title = "Professional mobility in Greater Paris : mean distance travelled",
            author = "Cartograflow, 2019",
            sources = "Sources : data : INSEE, RP, MOBPRO, 2017 ; basemap : IGN, APUR, UMS 2414
RIATE, 2018",
            scale = 5,
            tabtitle = TRUE,
            frame = TRUE,
            north(pos = "topright"),
            col = "grey",
            coltitle ="black")
```

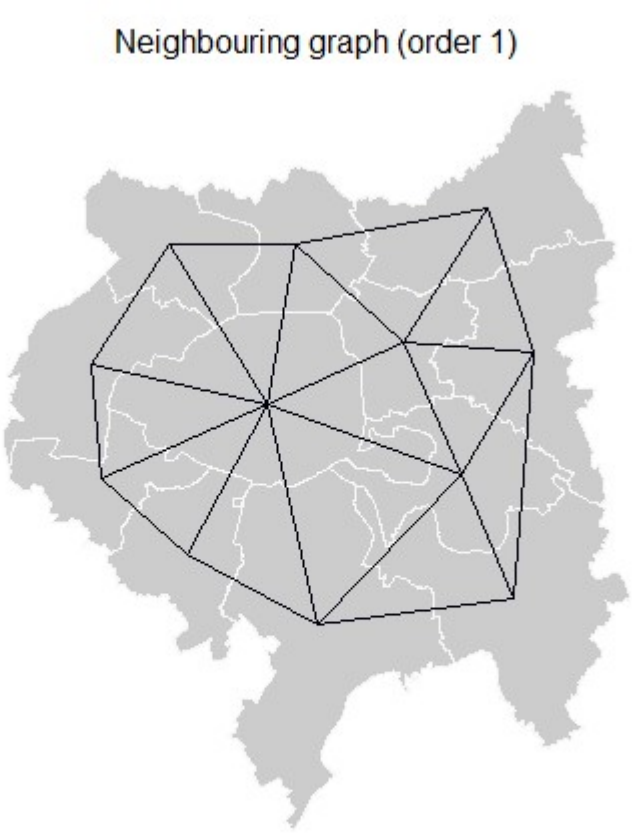


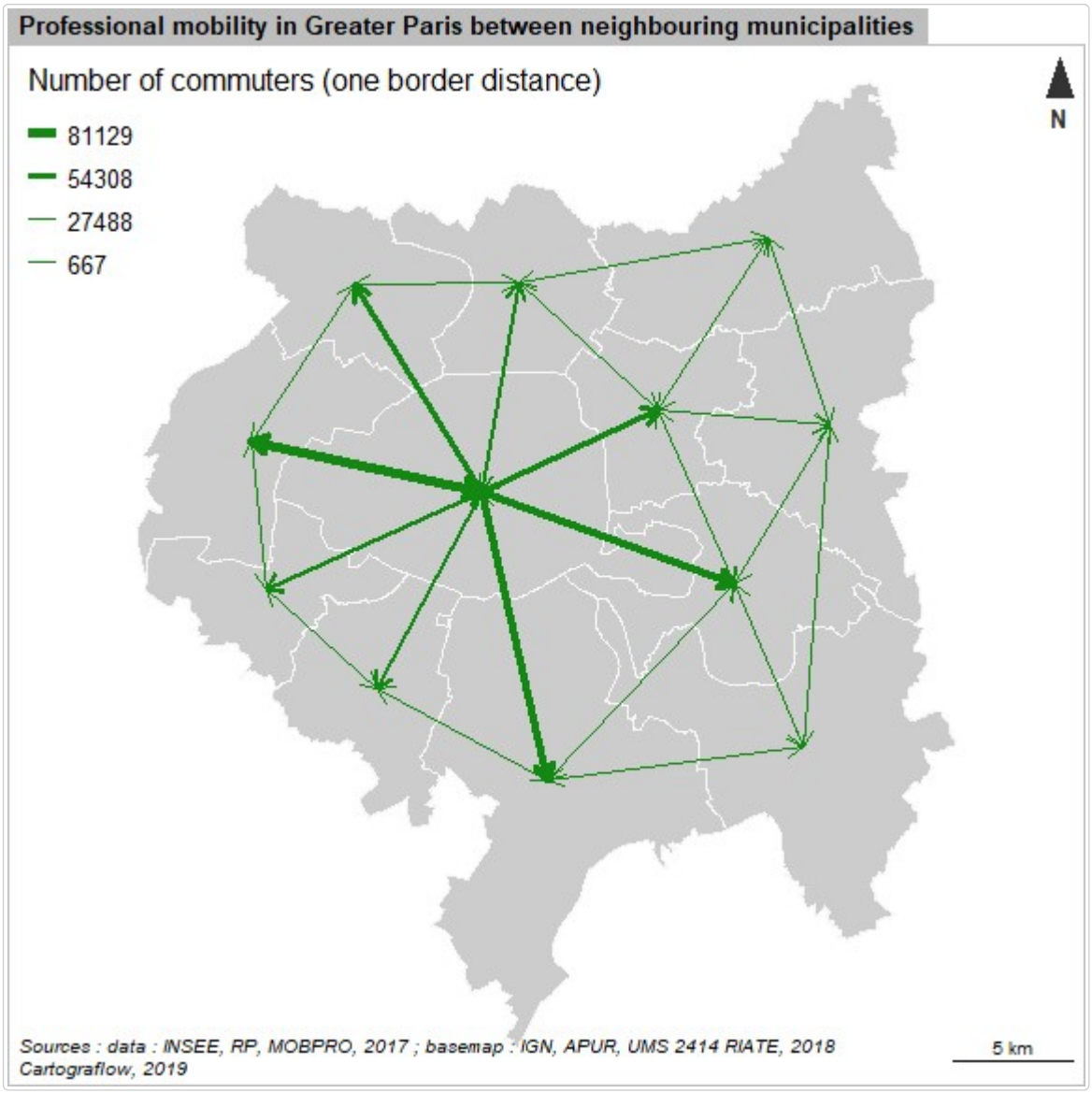
4.6.2 Ordinal distance

```
## Neighbouring graph (ordre 1)
graph_ckij_1<-flowcontig("./data/MGP_TER.shp",
```

```
                        "EPT_NUM",
                        ordre =1)

flowmap(graph_ckij_1,
        format="L",
        "./data/MGP_TER.shp",
        "EPT_NUM",
        filter = TRUE,
        taille = 0.5)


## warning : you use the default threshold= 1


mtext("Neighbouring graph (order 1)",
      side=3)

## Reducing flow matrice by the neighbouring graph (order= 1)
reduc<-flowreduct(tabflow,
                  graph_ckij_1,
                  metric = "ordinal")

flow.c<-reduc %>%
  select(i,j,flux)%>%
  filter(flux!=0)

#Plot adjacent flowmap
par(mar=c(0,0,1,0))
extent <- c(2800000, 1340000, 6400000, 4800000)
resolution<-300

flowmap(flow.c,
        format="L",
        "./data/MGP_TER.shp",
        "EPT_NUM",
        filter = TRUE,
        taille = 5,
        a.col="#138913",
        a.length = 0.1,
        a.head =1)


## warning : you use the default threshold= 1


# Legend
legendPropLines(pos="topleft",
                title.txt="Number of commuters (one border distance)",
                title.cex=1,
                cex=0.8,
```

```
                values.cex= 0.8,
                var=c(min(flow.c$flux),max(flow.c$flux)),
                col="#138913",
                lwd=5,
                frame = FALSE,
                values.rnd = 0
                )
# Habillage

layoutLayer(title = "Professional mobility in Greater Paris between neighbouring municipalities",
            author = "Cartograflow, 2019",
            sources = "Sources : data : INSEE, RP, MOBPRO, 2017 ; basemap : IGN, APUR, UMS 2414
RIATE, 2018",
            scale = 5,
            tabtitle = TRUE,
            frame = TRUE,
            north(pos = "topright"),
            col = "grey",
            coltitle ="black")
```


Neighbouring graph (order 1)



5. Reference

– Bahoken Francoise (2016), Programmes pour R/Rtudio annexés, in : *Contribution à la cartographie d'une matrice de flux*, Thèse de doctorat, Université Paris 7, pp. 325-346. URL : https://halshs.archives-ouvertes.fr/tel-01273776, pp. 480-520.

6. Reproducibility

```
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 15063)
##
```

```
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=French_France.1252  LC_CTYPE=French_France.1252
## [3] LC_MONETARY=French_France.1252 LC_NUMERIC=C
## [5] LC_TIME=French_France.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] rlang_0.3.0.1        bindrcpp_0.2.2       cartograflow_0.0.0.1
## [4] cartography_2.1.2    dplyr_0.7.8
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_0.2.5  xfun_0.4          purrr_0.2.5
##  [4] reshape2_1.4.3    lattice_0.20-38   colorspace_1.3-2
##  [7] htmltools_0.3.6   viridisLite_0.3.0 yaml_2.2.0
## [10] plotly_4.8.0      pillar_1.3.1      later_0.7.5
## [13] foreign_0.8-71    glue_1.3.0        sp_1.3-1
## [16] bindr_0.1.1       plyr_1.8.4        stringr_1.4.0
## [19] rgeos_0.4-2       munsell_0.5.0     gtable_0.2.0
## [22] htmlwidgets_1.3   evaluate_0.12     labeling_0.3
## [25] knitr_1.21        httpuv_1.4.5.1    maptools_0.9-4
## [28] crosstalk_1.0.0   Rcpp_1.0.0        xtable_1.8-3
## [31] promises_1.0.1    scales_1.0.0      jsonlite_1.6
## [34] mime_0.6          ggplot2_3.1.0     digest_0.6.18
## [37] stringi_1.2.4     shiny_1.2.0       grid_3.5.2
## [40] rgdal_1.3-6       tools_3.5.2       magrittr_1.5
## [43] lazyeval_0.2.1    tibble_2.0.0      crayon_1.3.4
## [46] tidyr_0.8.2       pkgconfig_2.0.2   data.table_1.12.0
## [49] assertthat_0.2.0  rmarkdown_1.11    httr_1.4.0
## [52] R6_2.3.0          compiler_3.5.2
```