Modélisation du prix d'un bien immobilier à Paris intramuros

Mais dis-moi chérie, le prix du m^2 a vachement augmenté !

Maxime PHILIPPOT Pierre LEPAGNOL

2020-01-14

Résumé

Nous avons voulu modéliser le prix d'un bien immobilier dans Paris. Pour pouvoir répondre à la question que beaucoup de monde se pose : "Pourquoi est-ce donc si cher ?". Ainsi pouvoir justifier un prix ou bien prédire pour des futurs biens pour dénicher les offres intéressantes à l'aide d'un algorithme reposant sur nos modèles. Pour ce faire nous avons du créer notre propre dataset en récoltant nos données sur différents sites web.

La seconde partie de notre travail fût la modélisation. Nous avons sélectionné nos variables de manière naïve, calculer le plus d'indicateurs (des distances, par exemple) qui soient indépendants les uns des autres. Puis en mettant en oeuvre notre cours d'Econométrie nous avons simulé avec nos données. Nous avons pu remarquer au long de l'étude que nous avons menée, la difficulté d'acquérir les données et parmi lesquelles : la position géographique des biens.

Table des matières

1 Création du dataset

1.1 Récuperation des Biens Immobiliers Bruts sous Python

1.1.1 Problème et Résolution

Il n’y avait aucun jeu de donnée déjà formaté. Ainsi nous avons récupéré nos données en scrappant le site PAP [1]. Ce site présente l’avantage d’être un site d’annonces n’appartenant pas à une agence donc le prix affiché n’est pas surévalué d’une marge d’une agence. Nous nous sommes restreints aux biens en vente seule (pas de bien en viager, pas de bien location) et nous avons exclus les fonds de commerce, garage, péniche, chalet, mobil-homes, locaux en tout genre.

Nous avons ainsi 99 appartements, 4 maisons, 21 studios, 2 chambres et 3 pièces.

1. (« PAP: Particulier à Particulier » 2020)

Nous avons récolté, pour chaque bien immobilier, le `type`, le `prix`, le nombre de photo (`nb_photo`), les trois transports les plus proches selon PAP (`transport_#`), le nombre de pièces (`nb_pieces`), le nombre de chambre (`nb_chambre`), la surface habitable (`surface`), le code postal (`code_postal`), la latitude `lat`, La longitude `lon`, l'url pour accéder à l'annonce (`url`).

Pour réaliser le scrapping nous avons utilisé les modules suivants :

- `requests` : Pour les requêtes HTTP.
- `unidecode` : Pour la gestion des caractères spéciaux.
- `datetime` : Pour dater nos fichiers.
- `ast` : Pour parser une chaîne de caractère en dictionnaire Python.
- `json` : Pour gérer les fichiers json
- `re` : Pour la gestion des expression.
- `bs4 (BeautifulSoup)` : Pour scrapper les pages web.

Pour accéder aux données nous avons effectué une requête HTTP à l'adresse :

`https://www.pap.fr/annonce/vente-immobiliere-paris-75-g439`

Nous avons réparti le code en plusieurs fonctions:

```
# Initiation de la racine du site web
site_main='https://www.pap.fr'

# Création d'un set() contenant toutes les urls des biens
URLset=GetURLSET(site_main,20)
# Nettoyage du set() pour enlever les types de biens "intéréssants"
# (Fond de commerces, locaux, péniches, etc).
URLset=CleanIDset(URLset)

#Création du jeu de données brut
data=GetDetails(site_main,URLset)

#Exportation de l'objet data (dict) dans un fichier .json
exportdata(data)
```

Vous trouverez en annexe, le code complet.

1.2 Création du `data.frame station` sous R

1.2.1 Lecture du jeu de données et sélection

Le fichier "positions-geographiques-des-stations-du-reseau-ratp.csv" provient du jeu de données de la RATP [2]. Ce fichier contient les coordonnées GPS et l'addresse des stations de bus, métro, RER, Tramway de la RATP.

Vous trouverez à Références l'addresse pour télécharger le jeu de donnée.

```
station<-read.csv("../../Datasets/positions-geographiques-des-stations-du-reseau-ratp.csv",header= T,se

# Ajout du code postal au data.frame et ordonner par code_postal

station<-cbind(station,code_postal=str_sub(as.character(station[,3]),-5))
station<-station[order(station$code_postal),]
station$code_postal<-as.numeric(as.character(station$code_postal))
```

2. (« RATP Open Data » 2020)

```r
# Mise en forme UTF-8 pour rectifier l'erreur d'écriture des stations avec des accents (les stations de

fwrite(station,"../../Datasets/station_UTF-8.csv")
station <- fread("../../Datasets/station_UTF-8.csv",encoding = "UTF-8")

# Séparation de Longitude et Latitude en deux colonnes

setDT(station)[, c("Latitude","Longitude") := tstrsplit(Coordinates, ",")]

#Sélection uniquement des stations de Paris et ordonner par nom de station

station<- station[station$code_postal < 75121,]
station<-station[order(station$Name),]
```

1.2.2 Nettoyage du jeu de données

On a remarqué que certaines stations avaient des addresses identiques, on a donc comparé les données Longitude/Latitude. Ces données ce sont avérées redondantes également. On a décidé de supprimer les stations qui possédaient plusieurs points GPS identique pour une même addresse, autrement dit une addresse est maintenant associée à un point GPS unique.

```r
# Suppression des doublons Longitude/Latitude dans le data.frame

station<-station[!duplicated(station[,c(3,6:7)]),]
```

Notre jeu de données qui provient de la RATP n'écrit pas de la même façon une station de métro et une station de bus. "Alésia" est une station de métro. "ALESIA - DIDOT" est un arrêt de bus. Pour uniformiser le nom des stations il faut donc supprimer les accents, puis toutes les mettre en majuscule.

```r
# Supprime les accents des stations de métro de notre jeu de données

for (i in seq(1, length(station$Name))){
  station$Name[i]<-iconv(station$Name[i],from="UTF-8",to="ASCII//TRANSLIT")
}

# Mise en forme de toutes les stations (en majuscule)

station$Name<-str_to_upper(station$Name)
```

On a supprimé également les lignes pour lesquelles les addresses étaient identiques pour ne garder qu'un point GPS par addresse.

On a remarqué qu'une station de métro pouvait avoir des données identiques d'addresse mais un point GPS différent. Une explication serait que les bouches de métro sont référencées et donc pour une station de métro et une addresse nous obtenons plusieurs points GPS différents (ce n'est qu'une hypothèse). On a décidé de garder qu'une information par station de métro.

```r
# Sélection d'une coordonnées GPS par addresse.

station<-station[!duplicated(station[,3]),]
station<-station[,-4] #on enlève "Coordinates" qui est maintenant en deux colonnes "Longitude" et "Lati
```

```
# On garde une ligne par station de métro en faisant la moyenne de Longitude/Latitude lorsqu'il y en a

station<-station%>%
  group_by(Name)%>%summarise(longitude=mean(as.numeric(Longitude)),latitude=mean(as.numeric(Latitude)))
```

1.3 Création dudata.frame biens sous R

1.3.1 Lecture du jeu de données et Sélection

le fichier "result_python_annonces.csv" provient du scrapping des annonces effectué à l'aide de Python. Pour chaque bien (ou presque), nous avons de l'informations sur trois stations à proximité (métro,RER)

```
biens<-read.csv("../../Datasets/result_python_annonces.csv",header= T,sep=",")

# Mise en forme UTF-8 pour rectifier l'erreur d'écriture des stations avec des accents

fwrite(biens,"../../Datasets/biens_UTF-8.csv")
biens <- fread("../../Datasets/biens_UTF-8.csv",encoding = "UTF-8")

# Supprime les accents des stations de chaque biens pour uniformiser avec la même écriture que le data.

for (i in seq(1, length(biens$transport.0))){
  biens$transport.0[i]<-iconv(biens$transport.0[i],from="UTF-8",to="ASCII//TRANSLIT")
  biens$transport.1[i]<-iconv(biens$transport.1[i],from="UTF-8",to="ASCII//TRANSLIT")
  biens$transport.2[i]<-iconv(biens$transport.2[i],from="UTF-8",to="ASCII//TRANSLIT")
}


# Mise en forme de toutes les stations (en majuscule)

biens$transport.0<-str_to_upper(biens$transport.0)
biens$transport.1<-str_to_upper(biens$transport.1)
biens$transport.2<-str_to_upper(biens$transport.2)

# Donne l'ensemble des stations présentes dans le jeu de données biens

transport<-unique(c(biens$transport.0,biens$transport.1,biens$transport.2))

# On ordonne par le nom de la première station renseignée, puis on a décidé de supprimer les biens pour

biens<-biens[order(biens$transport.0),]
biens<-biens[-c(1,2,3,4,5),]
```

On a remarqué un bien sans description du nombre de chambre. A l'aide de l'url renseigné dans le `data.frame` nous avons visité la page `https://www.pap.fr/annonces/appartement-paris-16e-75016-r430400259` correspondant au bien.

Si on regarde la photo du plan sur l'annonce on remarque que le bien peut avoir 1 ou 2 chambres selon les besoins/envies de l'acheteur, on a décidé que ce bien possédais qu'une chambre à l'achat et que la salle à manger pouvait se transformer en chambre par la suite.

```r
#url du bien

subset(biens,is.na(biens$nb_chambres))$url
```

```
## [1] "/annonces/appartement-paris-16e-75016-r430400259"
```

```r
# On associe 1 chambre à ce bien

biens$nb_chambres[is.na(biens$nb_chambres)]<-1
```

1.3.2 Prise en compte des stations sans référencement

Nous avons vu très rapidement que certaines stations référencées dans l'annonce ne correspondaient à aucune station du `data.frame station`.

```r
# On cherche donc les stations du data.frame biens qui n'ont pas de valeurs dans le data.frame "station

sort(setdiff(transport,unique(station$Name)))#on a 30 stations sur 198 qui n'ont pas de correspondance
```

```
##  [1] ""
##  [2] "ALEXANDRE DUMAS"
##  [3] "AVENUE HENRI MARTIN"
##  [4] "BUTTES CHAUMONT"
##  [5] "CHAMPS-ELYSEES - CLEMENCEAU (GRAND PALAIS)"
##  [6] "CHATEAU-LANDON"
##  [7] "CHATELET - LES HALLES"
##  [8] "CLUNY - LA SORBONNE"
##  [9] "CORENTIN CARIOU"
## [10] "FRANKLIN D. ROOSEVELT"
## [11] "GONCOURT (HOPITAL-SAINT-LOUIS)"
## [12] "HOPITAL ROBERT-DEBRE"
## [13] "JAVEL - ANDRE CITROEN"
## [14] "LA MOTTE-PICQUET - GRENELLE"
## [15] "MAGENTA"
## [16] "MARX DORMOY"
## [17] "NEUILLY - PORTE MAILLOT"
## [18] "NOTRE-DAME-DE-LORETTE"
## [19] "PEREIRE - LEVALLOIS"
## [20] "PEREIRE (MARECHAL JUIN)"
## [21] "PLACE MONGE (JARDIN DES PLANTES - ARENES DE LUTECE)"
## [22] "PONT DE L'ALMA"
## [23] "PONT-CARDINET"
## [24] "PORTE DE PANTIN (PARC DE LA VILLETTE)"
## [25] "PORTE DE SAINT-CLOUD (PARC DES PRINCES)"
## [26] "PORTE DE VERSAILLES (PARC DES EXPOSITIONS DE PARIS)"
## [27] "PORTE MAILLOT (PALAIS DES CONGRES)"
## [28] "SAINT-SEBASTIEN - FROISSART"
## [29] "SOLFERINO (MUSEE D'ORSAY)"
## [30] "STADE CHARLETY"
## [31] "TELEGRAPHE"
```

```
#length(transport) #transport qui est l'ensemble des stations référencées dans les 124 biens
```

Pour corriger ce problème, nous avons créé le fichier `result_r_station.csv` pour connaître le nom de toutes les stations de Paris ordonnées. On a associé manuellement les nouveaux noms de station en mettant en évidence deux causes de ce problème.

La première cause venait d'une écriture différente dans les deux `data.frame`. Par exemple,"JAVEL - ANDRE CITROEN" est devenu "JAVEL-ANDRE-CITROEN", "FRANKLIN D. ROOSEVELT" est devenu "FRANKLIN-ROOSEVELT" etc...

Ensuite, dans l'autre cas la cause venait d'un manque d'information sur les stations RER du jeu de données de la RATP. On a décidé d'associer la station de bus la plus proche de la station de RER pour garder une bonne information. Par exemple, "PONT DE L'ALMA" est devenu "BOSQUET - RAPP", "TELEGRAPHE" est devenu "PELLEPORT - BELLEVILLE", "STADE CHARLETY" est devenu "STADE CHARLETY - PORTE DE GENTILLY" etc...

```
# Création du fichier des stations pour trouver le nom exact des stations de métro, ou dans l'autre cas

#write.csv(station, file = "result_r_station.csv")

biens[biens=="ALEXANDRE DUMAS"]<-"ALEXANDRE-DUMAS"
biens[biens=="AVENUE HENRI MARTIN"]<-"OCTAVE FEUILLET"
biens[biens=="BUTTES CHAUMONT"]<-"BUTTES-CHAUMONT"
biens[biens=="CHAMPS-ELYSEES - CLEMENCEAU (GRAND PALAIS)"]<-"CHAMPS-ELYSEES - CLEMENCEAU"
biens[biens=="CHATEAU-LANDON"]<-"CHATEAU LANDON"
biens[biens=="CHATELET - LES HALLES"]<-"CHATELET-LES HALLES"
biens[biens=="CLUNY - LA SORBONNE"]<-"CLUNY-LA SORBONNE"
biens[biens=="CORENTIN CARIOU"]<-"CORENTIN-CARIOU"
biens[biens=="FRANKLIN D. ROOSEVELT"]<-"FRANKLIN-ROOSEVELT"
biens[biens=="GONCOURT (HOPITAL-SAINT-LOUIS)"]<-"GONCOURT (HOPITAL SAINT-LOUIS)"
biens[biens=="HOPITAL ROBERT-DEBRE"]<-"HOPITAL ROBERT DEBRE"
biens[biens=="JAVEL - ANDRE CITROEN"]<-"JAVEL-ANDRE-CITROEN"
biens[biens=="LA MOTTE-PICQUET - GRENELLE"]<-"LA MOTTE-PICQUET-GRENELLE"
biens[biens=="MAGENTA"]<-"LA FAYETTE - DUNKERQUE"
biens[biens=="MARX DORMOY"]<-"MARX-DORMOY"
biens[biens=="NEUILLY - PORTE MAILLOT"]<-"PORTE MAILLOT - PALAIS DES CONGRES"
biens[biens=="NOTRE-DAME-DE-LORETTE"]<-"NOTRE-DAME DE LORETTE"
biens[biens=="PEREIRE - LEVALLOIS"]<-"PEREIRE"
biens[biens=="PEREIRE (MARECHAL JUIN)"]<-"PEREIRE - MARECHAL JUIN"
biens[biens=="PLACE MONGE (JARDIN DES PLANTES - ARENES DE LUTECE)"]<-"PLACE MONGE (JARDIN DES PLANTES)"
biens[biens=="PONT-CARDINET"]<-"PONT CARDINET"
biens[biens=="PONT DE L'ALMA"]<-"BOSQUET - RAPP"
biens[biens=="PORTE DE PANTIN (PARC DE LA VILLETTE)"]<-"PORTE DE PANTIN - PARC DE LA VILLETTE"
biens[biens=="PORTE DE SAINT-CLOUD (PARC DES PRINCES)"]<-"PORTE DE SAINT-CLOUD"
biens[biens=="PORTE DE VERSAILLES (PARC DES EXPOSITIONS DE PARIS)"]<-"PORTE DE VERSAILLES - PARC DES EX
biens[biens=="PORTE MAILLOT (PALAIS DES CONGRES)"]<-"PORTE MAILLOT - PALAIS DES CONGRES"
biens[biens=="SAINT-SEBASTIEN - FROISSART"]<-"SAINT-SEBASTIEN-FROISSART"
biens[biens=="SOLFERINO (MUSEE D'ORSAY)"]<-"SOLFERINO - BELLECHASSE"
biens[biens=="STADE CHARLETY"]<-"STADE CHARLETY - PORTE DE GENTILLY"
biens[biens=="TELEGRAPHE"]<-"PELLEPORT - BELLEVILLE"

transport<-unique(c(biens$transport.0,biens$transport.1,biens$transport.2))
sort(setdiff(transport,unique(station$Name)))
```

```
## character(0)
```

On remarque bien que maintenant toutes les stations référencées dans les annonces possèdent une association dans le `data.frame station`

1.3.3 Ajout des coordonnées GPS des stations au `data.frame biens`

On peut commencer par associer à chaque bien les coordonnées des stations les plus proches. Par exemple, `Latitude_transport.0` est la latitude de la première station de métro/RER référencée dans l'annonce, `Longitude_transport.1` est la longitude de la deuxième station de métro/RER référencée dans l'annonce, etc...

```
# On intègre les coordonnées GPS des stations de l'annonce à notre data.frame biens

colnames(station)[1]<-"transport.0"
transport0<-merge(biens,station, by="transport.0")
transport0<-transport0[order(transport0$transport.0),]

colnames(station)[1]<-"transport.1"
transport1<-merge(biens,station, by="transport.1")
transport1<-transport1[order(transport1$transport.0),]

colnames(station)[1]<-"transport.2"
transport2<-merge(biens,station, by="transport.2")
transport2<-transport2[order(transport2$transport.0),]

biens<-biens[order(biens$transport.0),]
biens<-cbind(biens,Latitude_transport.0=transport0$latitude,Longitude_transport.0=transport0$longitude,
```

1.3.4 Ajout de la distance des biens aux stations au `data.frame biens`

On a décidé de calculer la distance du bien aux stations de métro/RER pour déterminer si le fait que les stations de métro/RER soit éloignées ou proches à une incidence sur le prix. On peut déjà émettre l'hypothèse que cette distance ne devrait pas influée sur le prix puisqu'il existe énormément de station de métro/RER à Paris. Par conséquent, pour chaque bien il va forcément y avoir une station proche.

Cela sera à prouver lors de notre modélisation lorsqu'on décidera ou non de garder ces variables dans notre modèle final.

```
#On intègre les distances des biens aux stations de métro/RER référencées dans l'annonce

dist_metro0=c()
dist_metro1=c()
dist_metro2=c()

for (i in seq(1, length(biens$code.postal))){
    dist_metro0=c(dist_metro0,distHaversine(c(biens$lon[i],biens$lat[i]),c(biens$Longitude_transport.0[
}



distHaversine(c(2.270395,48.86509),c(2.269568,48.86309))
```

```
## [1] 230.7291
```

```
for (i in seq(1, length(biens$code.postal))){
    dist_metro1=c(dist_metro1,distHaversine(c(biens$lon[i],biens$lat[i]),c(biens$Longitude_transport.1[
}

for (i in seq(1, length(biens$code.postal))){
    dist_metro2=c(dist_metro2,distHaversine(c(biens$lon[i],biens$lat[i]),c(biens$Longitude_transport.2[
}

dist_metro<-data.frame(dist_metro0,dist_metro1,dist_metro2)
colnames(dist_metro)<-c("distance_station_0","distance_station_1","distance_station_2")

biens<- cbind(biens,dist_metro)
```

1.3.5 Ajout de la distance des biens aux monuments les plus visités de Paris

On enrichit le`data.frame biens`à l'aide des coordonnées GPS [3] des monuments de Paris [4] récoltées sur internet.

Pour obtenir ces données il suffit de se rendre à l'addresse en Références et de rentrer le nom du monument pour obtenir la longitude et latitude.

```
# On calcule la distance des biens à ces 11 monuments.

monument<-read.csv("../../Datasets/monuments_paris.csv",header= T,sep=",",encoding = "UTF-8")

mon=c()
dist_monu=data.frame()
for (i in seq(1, length(biens$code.postal))){
    mon=c()
for (c in seq(1, length(monument$Nom))){

    mon=c(mon,distHaversine(c(biens$lon[i],biens$lat[i]), c(monument$Longitude[c],monument$Latitude[c]))

}
 dist_monu<-rbind(dist_monu,mon)
}
colnames(dist_monu)<-monument$Nom

biens<- cbind(biens,dist_monu)
```

1.3.6 Ajout de la distance des biens à l'Université la plus proche au `data.frame biens`

On enrichit le`data.frame biens`à l'aide des coordonnées GPS des principales universités de Paris [5] récoltées sur internet.

Pour obtenir ces données, il faut appliquer la même procédure que pour les monuments.

Dans ce cas, on ne calcule pas la distance des biens à toutes les universités mais la distance du biens à l'université la plus proche. On ajoute donc une seule colonne au `data.frame`

3. (« Coordonnées GPS » 2020)
4. (« Monuments de Paris » 2020)
5. (« Université de Paris » 2020)

```r
# Lecture du fichier
universite<-read.csv("../../Datasets/université_paris.csv",header= T,sep=",",encoding = "UTF-8")

# Calcule de la distance du bien pour chaque université puis sélection du minimum pour garder la distan
dist_uni=rep(0,124)
for (i in seq(1, length(biens$code.postal))){
    v=rep(0,7)
for (c in seq(1, length(universite$Nom))){

    v[c]=distHaversine(c(biens$lon[i],biens$lat[i]), c(universite$Longitude[c],universite$Latitude[c]),
}
 dist_uni[i]=min(v)
}

biens<-cbind(biens,distance_université_plus_près=dist_uni)
```

1.3.7 Transformation de la variable qualitative "type"

On a une variable qualitative qui donne le type de bien de l'annonce: `appartement`,`maison`,`chambre`,`pièce`,`studio`.

```r
#On considère que les types "pièce" et "chambre" reflètent la même information, on associe "pièce" à "c

biens[biens=="pièce"]<-"chambre"

#On rajoute quatres colonnes à gauche pour mettre la variable "type" qui est qualitative en 4 variable

app=c()
stu=c()
mai=c()
cha=c()
for (i in seq(1, length(biens$code.postal))){
  if(biens$type[i]=="appartement") {app=c(app,1)} else {app=c(app,0)}
  if(biens$type[i]=="studio") {stu=c(stu,1)} else {stu=c(stu,0)}
  if(biens$type[i]=="maison") {mai=c(mai,1)} else {mai=c(mai,0)}
  if(biens$type[i]=="chambre") {cha=c(cha,1)} else {cha=c(cha,0)}
}

biens<-cbind(appartement=app,studio=stu,maison=mai,chambre=cha,biens)
```

1.3.8 Ajout d'une variable binaire sur les arrondissements

On a choisit de retranscrire en variable binaire le fait que le bien est soit dans la première coronne de Paris (arrondissement de 1 à 11) ou dans la deuxième couronne (arrondissement de 12 à 20).

```r
#On choisit de mettre 0 lorsque le bien est dans l'hyper-centre et 1 sinon (i.e. 0 du 1er au onzième ar

arr=c()
for (i in seq(1, length(biens$code.postal))){
  if(as.numeric(str_sub(biens$code.postal[i],-2))<12) {arr=c(arr,0)} else {arr=c(arr,1)}
}

biens<-cbind(biens,arrondissement=arr)
```

2 Statistique Descriptive

Dans cette partie, on a étudier les variables explicatives que l'ont trouvait les plus judicieuses à analyser. Notre jeu de données "biens" possède une multitude de valeurs inutiles à analyser (lat,lon,type,transport.0,Longitude_transport.2...). On a dans un premier temps réduire ce jeu de données aux variables intéressantes.

2.0.1 Résumé statistique du type de bien et nombre de photos

A l'aide de ce résumé on peut analyser nos différentes variables avant d'en faire un modèle. On remarque beaucoup d'appartement et studio dans notre jeu de données, ils représentent presque 90% de nos biens. On constate que 50% des annonces ont moins de 10 photos avec certaines annonces sans photos.

```
# Comparaison de deux distances avec et sans prise en compte de la sphère de la Terre

#distHaversine(c(biens$lon[1],biens$lat[1]), c(monument$Longitude[1],monument$Latitude[1]), r=6378137)
#distGeo(c(biens$lon[1],biens$lat[1]), c(monument$Longitude[1],monument$Latitude[1]))

# Sélection des variables

biens_stat_des<-biens[,c(1,2,3,4,6,7,12,13,14,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39)]

#Transformation des variables qualitatives en facteur de niveaux

biens_stat_des$appartement<-as.factor(biens_stat_des$appartement)
biens_stat_des$studio<-as.factor(biens_stat_des$studio)
biens_stat_des$maison<-as.factor(biens_stat_des$maison)
biens_stat_des$chambre<-as.factor(biens_stat_des$chambre)
biens_stat_des$arrondissement<-as.factor(biens_stat_des$arrondissement)

#Résumé statistique

summary(biens_stat_des[,c(1,2,3,4,5)])
```

```
##  appartement studio  maison  chambre    nb_photo
##  0:30        0:103   0:120   0:119   Min.   : 0.000
##  1:94        1: 21   1:  4   1:  5   1st Qu.: 6.000
##                                      Median :10.000
##                                      Mean   : 8.935
##                                      3rd Qu.:12.000
##                                      Max.   :18.000
```

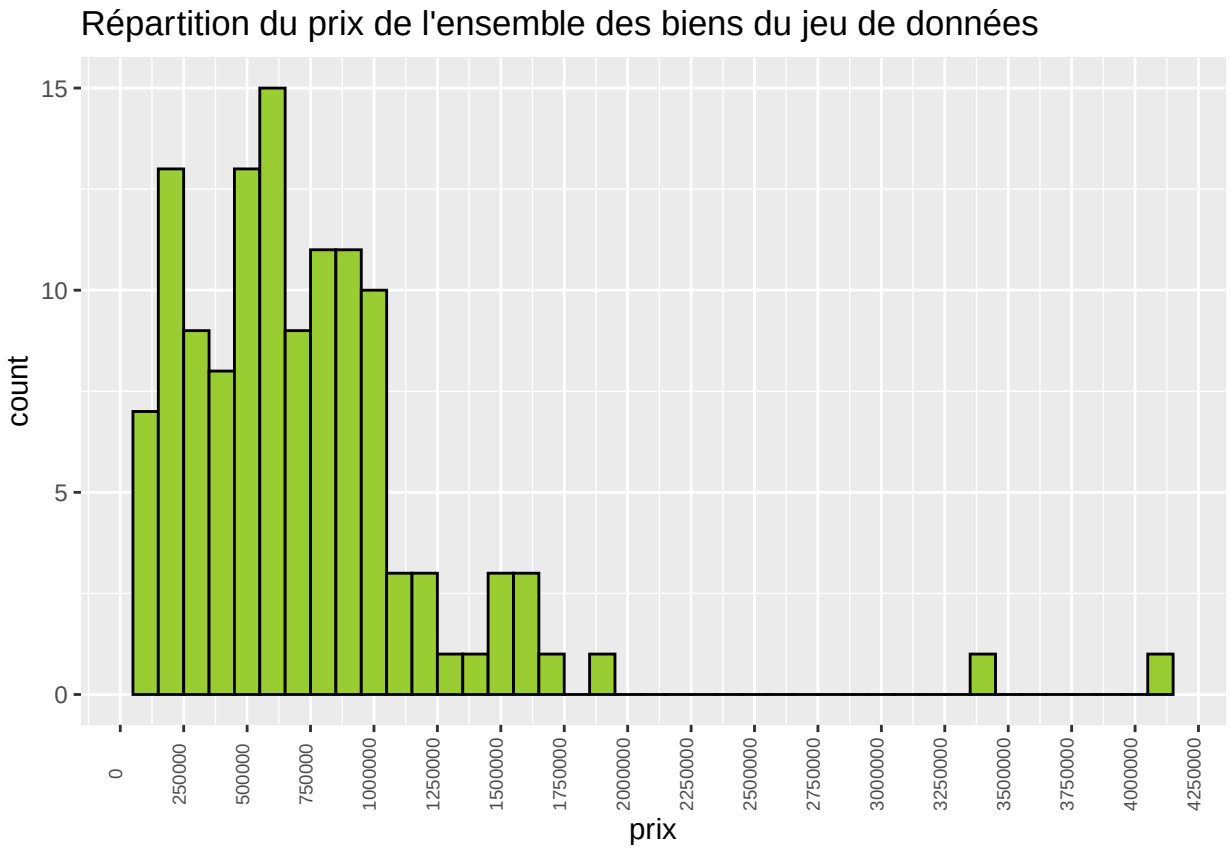
2.0.2 Prix des biens de notre jeu de données

On remarque que bon nombre de biens en un prix inférieur à 1 million d'euros. On a tout de même deux maisons a plus de 3 millions d'euros. On a beaucoup de biens autour de 500000euros dans notre jeu de données.

```
# Résumé statistique

summary(biens_stat_des[,6])
```

```
##       prix
##  Min.   :  91000
##  1st Qu.: 388750
##  Median : 650000
##  Mean   : 728296
##  3rd Qu.: 926250
##  Max.   :4113500
```

```r
# Répartition du prix de l'ensemble des biens de notre jeu de données

prix_biens <- ggplot(biens_stat_des, aes(x = prix)) +
        geom_histogram(aes(y = ..count..),binwidth = 100000,fill="olivedrab3",col="black") +
        scale_x_continuous(breaks=seq(0,4500000,250000))+theme(axis.text.x = element_text(angle=90, size
prix_biens
```



2.0.3 Distance des biens aux stations

Si on regarde les distances aux stations on remarque qu'en moyenne les stations sont rangées dans l'ordre dans les annonces. Autrement dit, la station la plus proche est nommer en premier dans l'annonce, suivi de la deuxième station la plus proche pour finir par la troisième station de métro/RER la plus proche. A noter que la distance Haversine est exprimée en mètres et prend en compte l'ellipsoïde de la Terre. Cela n'a pas vraiment d'incidence dans notre cas puisque les distances sont relativement proche. Pour les distances des bies aux monuments par exemple cela varie de simplement quelques mètres lorsqu'on prend en compte ou non la sphère de la Terre. On remarque qu'en moyenne la première station renseignée est à 228.03m du bien. 25% des biens ont la première station à moins de 152.25m. On constate également une répartion plutôt équitablement répartie avec 50% des biens ayant la première station à moins de 229.32m et 50% des autres

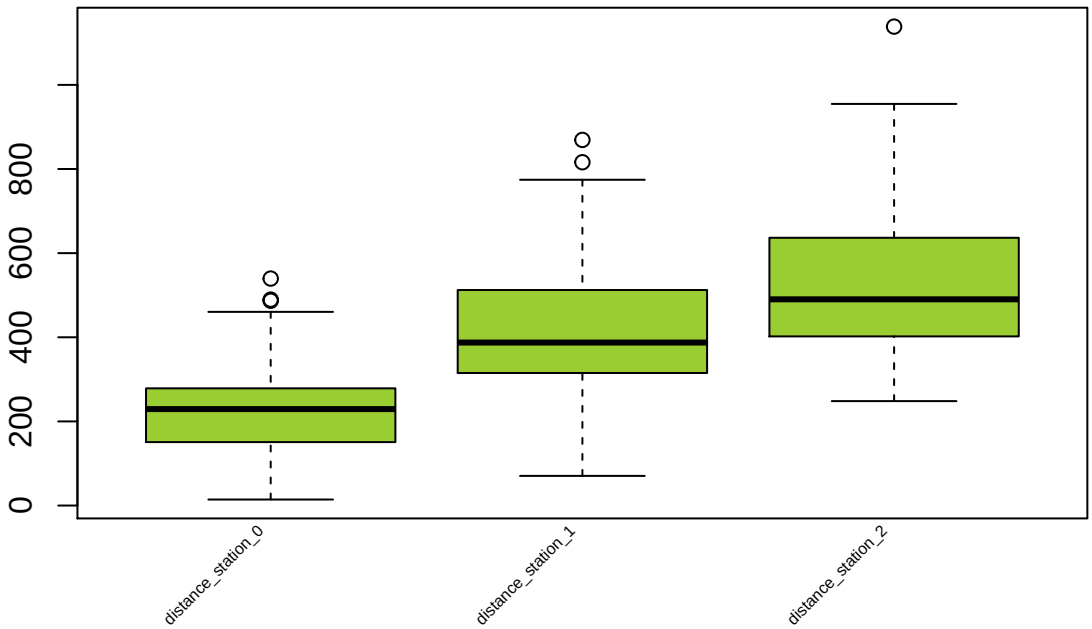
ayant une station comprise entre 229.32m et 539.63m. Pour ce qui est de la deuxième et troisième station ont obtient également des répartions équitables puisqu'on a 50% des biens à moins de 387.49m de la station (maximum à 869.20m) dans le premier cas, et 50% des biens à moins de 490.3m de la station (maximum à 1138.5m) dans le second cas. On peut ainsi confirmer que le nombre important de stations de métros/RER dans Paris rend des distances faibles puisque pratiquement tous les biens ont une station de métro/RER à moins de 1km.

```
# Résumé statistique

summary(biens_stat_des[,c(10,11,12)])
```

```
##  distance_station_0 distance_station_1 distance_station_2
##  Min.   : 14.37     Min.   : 70.54     Min.   : 248.3
##  1st Qu.:152.25     1st Qu.:315.16     1st Qu.: 403.4
##  Median :229.32     Median :387.49     Median : 490.3
##  Mean   :228.03     Mean   :415.35     Mean   : 534.9
##  3rd Qu.:278.32     3rd Qu.:510.29     3rd Qu.: 635.5
##  Max.   :539.63     Max.   :869.20     Max.   :1138.5
```

```
# Représentation de la répartion des distances selon les stations
par(xaxt="n")
boxplot(biens_stat_des$distance_station_0,biens_stat_des$distance_station_1,biens_stat_des$distance_sta


axis(1, at=seq(1, 3, by=1), labels = FALSE,las=2)
text(seq(1, 3, by=1), par("usr")[3] - 0.2, labels = colnames(biens_stat_des[,c(10,11,12)]),adj= 1.1, sr
```

Représentation de la répartion des distances selon les stations



2.0.4 Distance des biens aux monuments

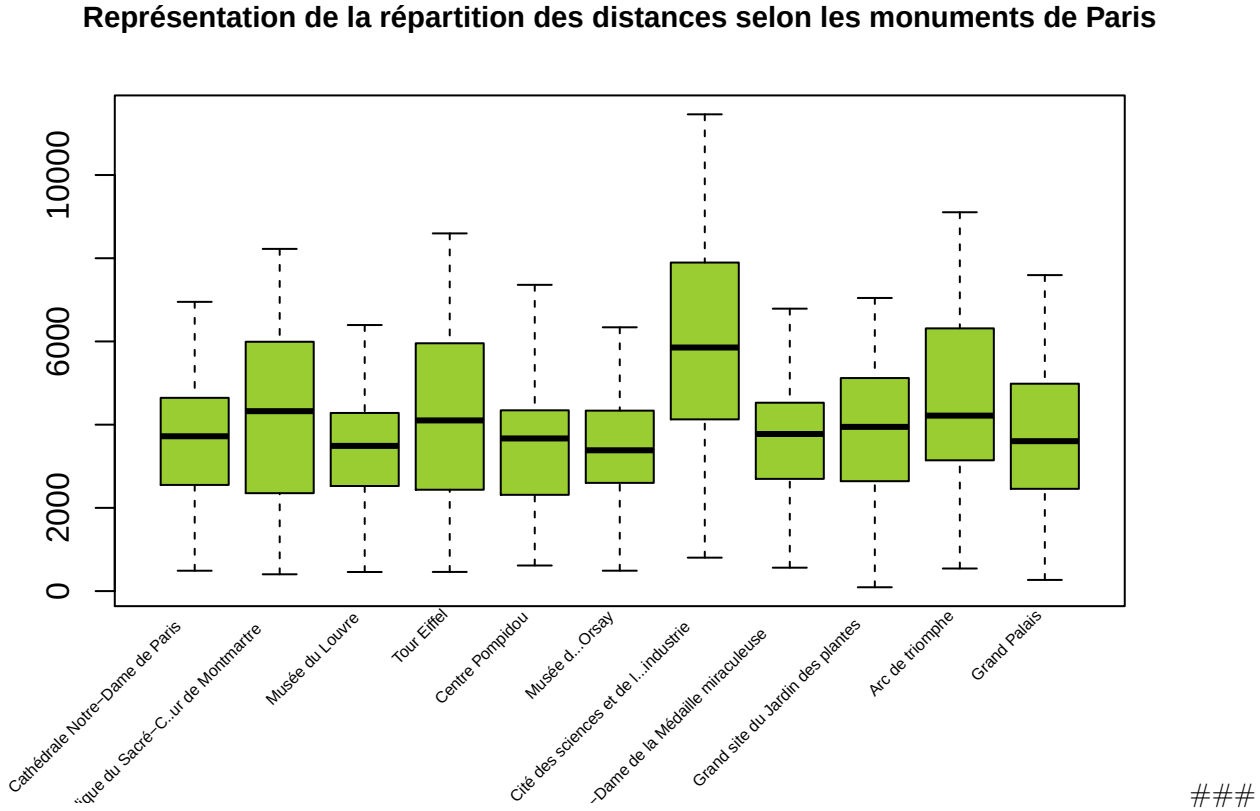
Si on regarde les distances des biens aux monuments les plus visités de Paris on remarque qu'en moyenne tous nos biens sont plus proches du Musée du Louvre, ils sont en moyenne à 3415.7m. On remarque que la Cité des sciences et de l'industrie possède une distribution très étaléa allant d'environ 500m à plus de 10km. Le fait est que ce monument est à l'extrémité nord-est de Paris et donc un bien à l'extrémité sud-ouest est à plus de 10km du monument.

```
# Résumé statistique

summary(biens_stat_des[,c(13,14,15,16,17,18,19,20,21,22,23)])
```

```
##  Cathédrale Notre-Dame de Paris Basilique du Sacré-Cœur de Montmartre
##  Min.   : 489.1                 Min.   : 403.4
##  1st Qu.:2565.8                 1st Qu.:2355.5
##  Median :3720.9                 Median :4324.0
##  Mean   :3623.1                 Mean   :4168.5
##  3rd Qu.:4644.3                 3rd Qu.:5943.1
##  Max.   :6951.4                 Max.   :8224.4
##  Musée du Louvre   Tour Eiffel     Centre Pompidou  Musée d'Orsay
##  Min.   : 458.5   Min.   : 460.5   Min.   : 615.3   Min.   : 490.3
##  1st Qu.:2543.3   1st Qu.:2439.6   1st Qu.:2329.2   1st Qu.:2623.7
##  Median :3489.6   Median :4101.6   Median :3668.3   Median :3383.7
##  Mean   :3415.7   Mean   :4325.9   Mean   :3520.1   Mean   :3495.0
##  3rd Qu.:4274.6   3rd Qu.:5953.8   3rd Qu.:4320.4   3rd Qu.:4326.2
##  Max.   :6394.3   Max.   :8596.4   Max.   :7360.5   Max.   :6339.9
##  Cité des sciences et de l'industrie
##  Min.   :  803.3
##  1st Qu.: 4126.9
##  Median : 5854.1
##  Mean   : 5991.3
##  3rd Qu.: 7890.6
##  Max.   :11458.3
##  Chapelle Notre-Dame de la Médaille miraculeuse
##  Min.   : 562.3
##  1st Qu.:2697.2
##  Median :3775.5
##  Mean   :3699.0
##  3rd Qu.:4519.3
##  Max.   :6787.9
##  Grand site du Jardin des plantes Arc de triomphe   Grand Palais
##  Min.   :  92.22                  Min.   : 542.7   Min.   : 268.1
##  1st Qu.:2645.93                  1st Qu.:3146.8   1st Qu.:2461.0
##  Median :3946.69                  Median :4217.3   Median :3602.9
##  Mean   :3875.11                  Mean   :4466.5   Mean   :3754.1
##  3rd Qu.:5110.92                  3rd Qu.:6302.6   3rd Qu.:4963.6
##  Max.   :7044.63                  Max.   :9105.0   Max.   :7594.0
```

```
# Représentation de la répartition des distances selon les monuments de Paris

par(xaxt="n")
boxplot(biens_stat_des$`Cathédrale Notre-Dame de Paris`,biens_stat_des$`Basilique du Sacré-Cœur de Montr
```

```
axis(1, at=seq(1, 11, by=1), labels = FALSE,las=2)
text(seq(1, 11, by=1), par("usr")[3] - 0.2, labels = colnames(biens_stat_des[,c(13,14,15,16,17,18,19,20
```

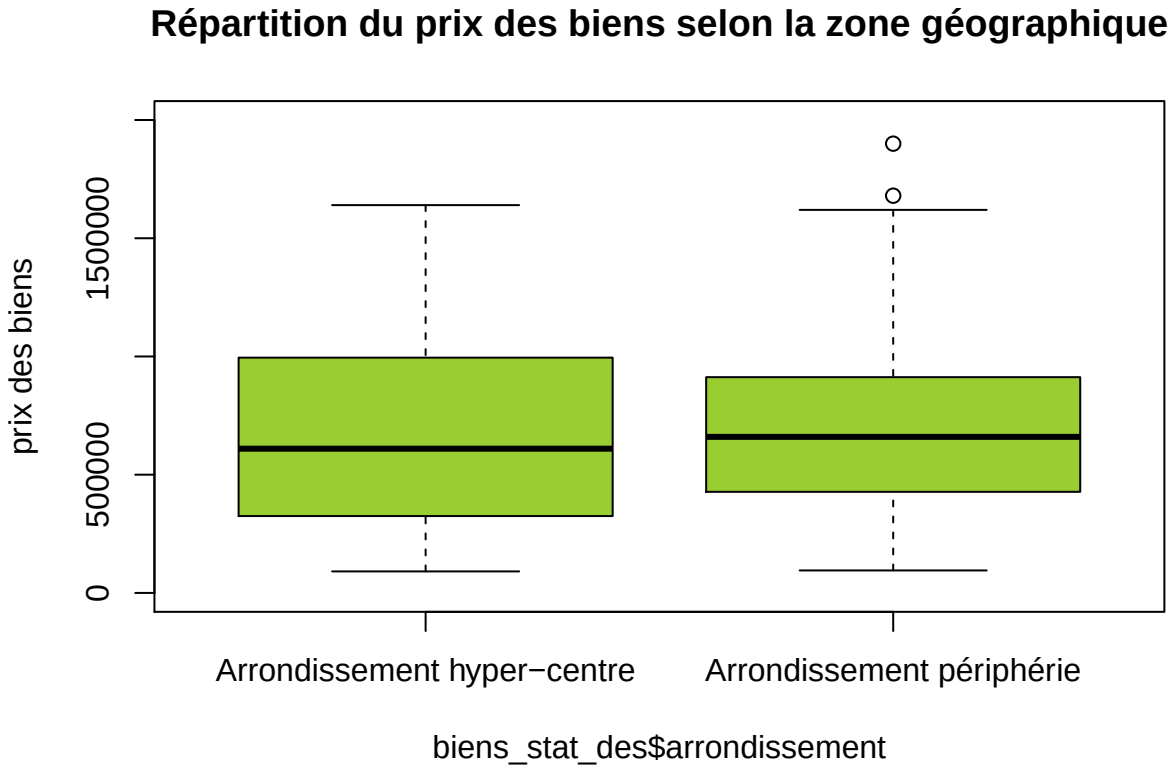


###

Prix des biens en fonction de la zone géographique dans Paris

On remarque que le fait que les biens soient dans l'hyper-centre ou non de Paris n'a pas d'influence sur les prix. Les deux médianes sont autour de 600000euros avec une concentration légèrement plus forte pour les biens dans les arrondissement périphériques de Paris (i. e. du 12e au 20e)

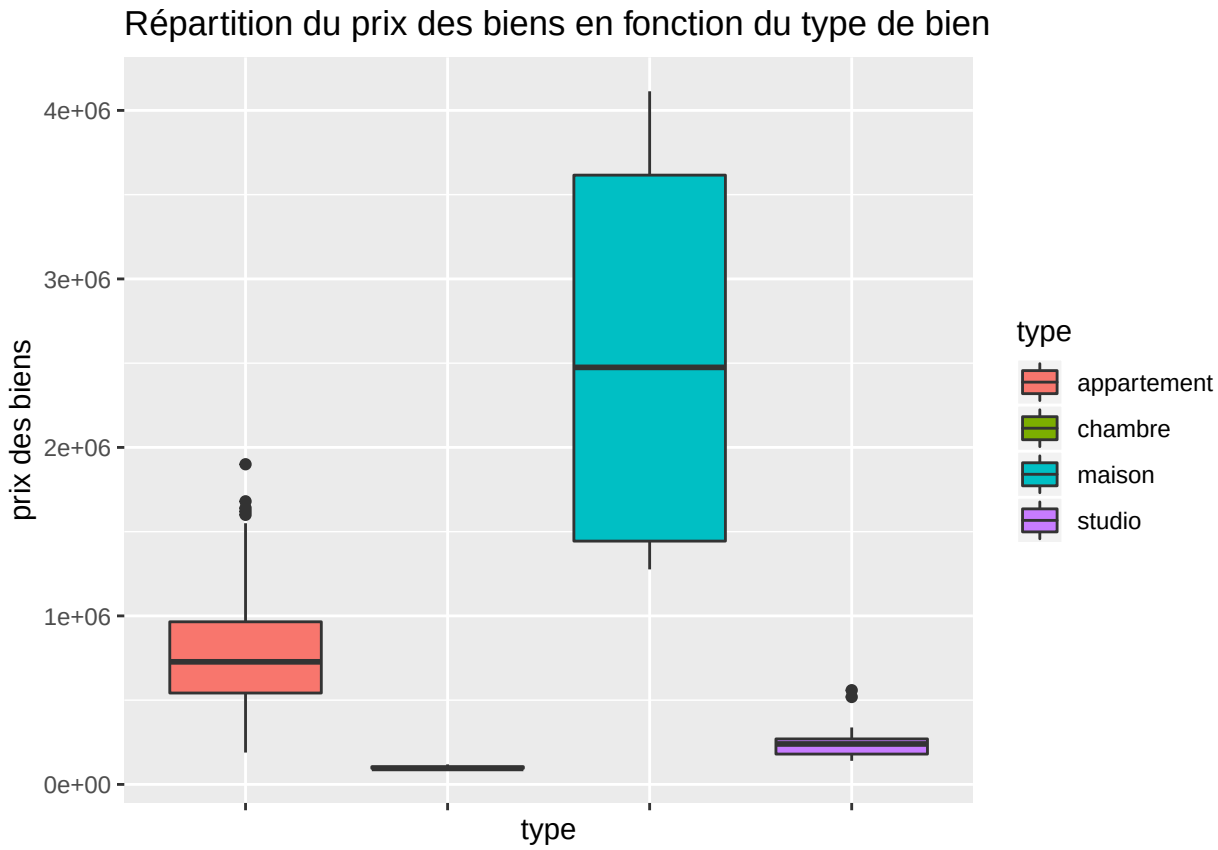
```
# Répartition du prix des biens en fonction de la zone géographique

boxplot(biens_stat_des$prix~biens_stat_des$arrondissement,names=c("Arrondissement hyper-centre","Arrond
```



2.0.5 Prix des biens en fonction du type de biens

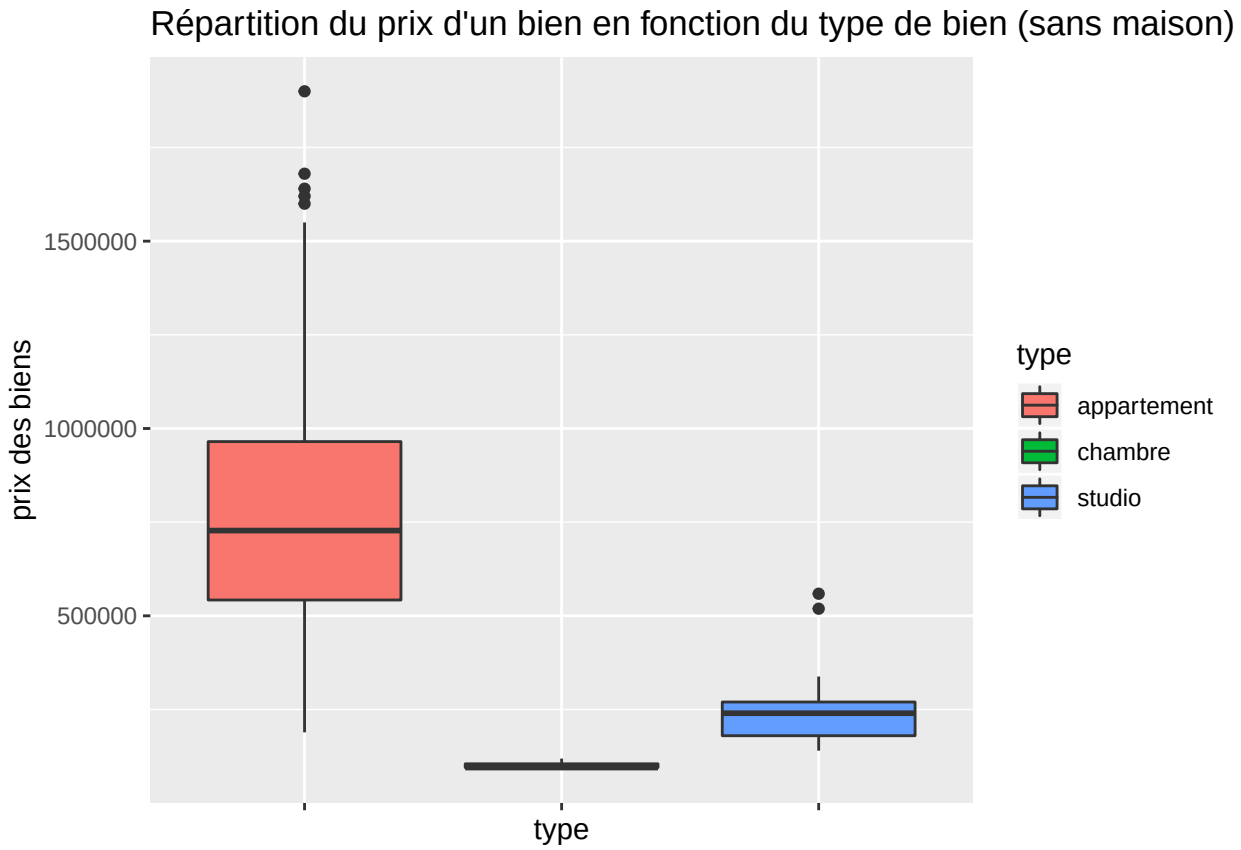
On a choisit de séparer les biens selon leur type pour essayer de mettre en évidence le répartition du prix selon le type de bien. On remarque que le prix des maisons est nettement supérieur au prix des appartement, 75% des maisons (3 en réalité...) ont un prix supérieur à pratiquement 100% des appartement lorsqu'on se place au prix de 1500000euros.

```
#Répartition du prix des bien en fonction de tous les types de biens

rep_biens <- ggplot(data=biens, aes(x=type, y=prix,fill=type))
rep_biens <- rep_biens +ggtitle("Répartition du prix des biens en fonction du type de bien")+ geom_boxpl
rep_biens
```



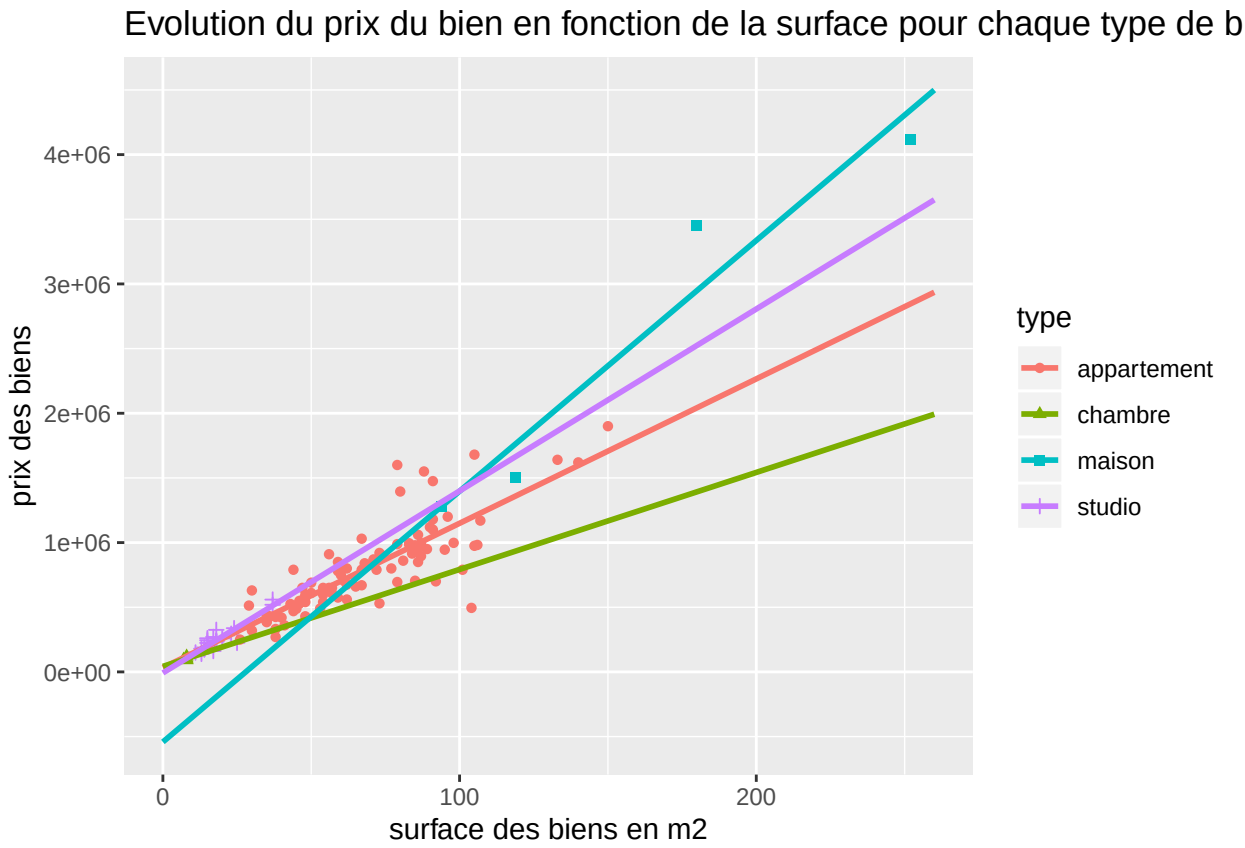
On a choisit de supprimer les maisons dans la représentation suivante pour limiter l'écrasement des répartitions des prix des chambres et studio. On constate que 100% des chambres sont moins chères que les studio et appartements. Une chambre est associée à l'absence d'une cuisine, WC et douches dans le logement. On remarque aussi que le prix des appartements varie de 250000euros à plus de 1500000euros (soit plus 6 fois son prix minimum qui est pourtant élevé...) avec un prix médian à 750000euros selon notre jeu de données. Le prix médian d'un studio est de 250000euros à Paris selon notre jeu de données.

```
# On élimine les maisons pour voir plus en détails la répartition des autres type de biens

biens_sans_mai<-biens[order(biens$type),][-c(100,101,102,103),]


# On obtient une nouvelle représentation

rep_biens_sans_mai <- ggplot(data=biens_sans_mai, aes(x=type, y=prix,fill=type))
rep_biens_sans_mai<- rep_biens_sans_mai +ggtitle("Répartition du prix d'un bien en fonction du type de l
rep_biens_sans_mai
```



On a choisit d'effectuer une régression du prix des biens en fonction de la surface et du type de bien. Les données sont intéressantes à analyser puisqu'on constate avec notre jeu de données que le prix d'une maison de 75m2 équivaut à celui d'un appartement de 75m2. Cependant, toutes les analyses ne sont pas correctes. Il serait faux de dire que le prix d'une chambre de 50m2 équivaut au prix d'une maison de 50m2 puisque par définition une chambre en vente dépasse rarement les 15m2 dans Paris. Pourtant, c'est ce que nous dit ce graphique. Pour finir, on remarque que la surface à plus d'incidence sur le prix d'une maison que sur le prix d'un bien, autrement dit 1m2 en plus va faire augmenter plus fortement le prix d'une maison que le prix d'un appartement.

```
# régression du prix des biens en fonction du type de bien

ggplot(biens,aes(x=surface,y=prix,color=type, shape=type))+geom_point() +scale_y_continuous(name="prix
```



2.0.6 Matrice de variance-covariance

La matrice de variance-covariance nous sert à sélectionner dans notre modèle final que les variables indépendantes entre-elles. Sans surprise, on remarque que la surface d'un bien est fortement corrélé avec le nombre de pièces et le nombre de chambre. A l'aide de cette matrice nous remarquons également que les monuments proches sont fortement corrélés entre eux. Par exemple, le centre Pompidou et Cathédrale Notre-Dame de Paris ont une covariance de 0.94. Lorsqu'on sélectionnera le modèle final il faudra s'assurer que nous avons pas garder des monuments proches pour avoir des variables décorrélées entre elles.

```
mat_cor <- round(cor(biens[,c(6,12,13,14,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38)]),2)
rownames(mat_cor) <- colnames(biens[,c(6,12,13,14,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38)])

colnames(mat_cor) <-colnames(biens[,c(6,12,13,14,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38)])

mat_cor<-data.frame(mat_cor)
head(mat_cor)
```

```
##                    nb_photo nb_pieces nb_chambres surface distance_station_0
## nb_photo               1.00      0.21        0.23    0.22               0.05
## nb_pieces              0.21      1.00        0.94    0.91               0.15
## nb_chambres            0.23      0.94        1.00    0.84               0.17
## surface                0.22      0.91        0.84    1.00               0.09
## distance_station_0     0.05      0.15        0.17    0.09               1.00
## distance_station_1     0.08      0.24        0.26    0.21               0.31
##                    distance_station_1 distance_station_2
```

```
## nb_photo                        0.08              0.01
## nb_pieces                       0.24             -0.05
## nb_chambres                     0.26             -0.03
## surface                         0.21             -0.05
## distance_station_0              0.31              0.25
## distance_station_1              1.00              0.35
##                   Cathédrale.Notre.Dame.de.Paris
## nb_photo                                    0.12
## nb_pieces                                   0.18
## nb_chambres                                 0.23
## surface                                     0.17
## distance_station_0                          0.10
## distance_station_1                          0.18
##                   Basilique.du.Sacré.Cœur.de.Montmartre Musée.du.Louvre
## nb_photo                                           0.08            0.11
## nb_pieces                                          0.29            0.27
## nb_chambres                                        0.32            0.32
## surface                                            0.25            0.20
## distance_station_0                                 0.12            0.14
## distance_station_1                                 0.12            0.26
##                   Tour.Eiffel Centre.Pompidou Musée.d.Orsay
## nb_photo                -0.02            0.12          0.08
## nb_pieces                0.10            0.23          0.23
## nb_chambres              0.10            0.28          0.28
## surface                 -0.01            0.22          0.13
## distance_station_0       0.03            0.13          0.12
## distance_station_1       0.11            0.18          0.26
##                   Cité.des.sciences.et.de.l.industrie
## nb_photo                                         0.06
## nb_pieces                                        0.16
## nb_chambres                                      0.18
## surface                                          0.20
## distance_station_0                               0.08
## distance_station_1                               0.02
##                   Chapelle.Notre.Dame.de.la.Médaille.miraculeuse
## nb_photo                                                    0.04
## nb_pieces                                                   0.13
## nb_chambres                                                 0.16
## surface                                                     0.05
## distance_station_0                                          0.04
## distance_station_1                                          0.20
##                   Grand.site.du.Jardin.des.plantes Arc.de.triomphe
## nb_photo                                      0.11            0.00
## nb_pieces                                     0.11            0.18
## nb_chambres                                   0.14            0.20
## surface                                       0.12            0.05
## distance_station_0                            0.06            0.06
## distance_station_1                            0.13            0.14
##                   Grand.Palais distance_université_plus_près
## nb_photo                  0.03                         0.01
## nb_pieces                 0.21                         0.03
## nb_chambres               0.24                         0.05
## surface                   0.08                        -0.09
## distance_station_0        0.09                         0.05
```

```
## distance_station_1         0.20                          0.04
```

3 Modèles Econométrique

3.1 Construction du Modèle "Optimal".

Durant cette partie nous utiliserons le `data.frame biens_stat_des`

Extrayons dans un premier temps notre Y et notre X: le prix d'un bien (Y) et Les variables exogènes (X) en prenant soin de retirer la variable maison (par soucis du rang de la matrice de design).

3.1.1 Premier modèle "Naïf"

Nous créeons un premier modèle avec toutes les variables explicatives.

```
lm_1=lm(Y~.,data=X)
summary(lm_1)
```

```
##
## Call:
## lm(formula = Y ~ ., data = X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -597981  -56713     216   60918  709974
##
## Coefficients:
##                                                 Estimate Std. Error t value
## (Intercept)                                    3.593e+05  2.916e+05   1.232
## appartement                                   -4.702e+05  1.187e+05  -3.961
## studio                                        -4.259e+05  1.438e+05  -2.961
## chambre                                       -4.698e+05  1.611e+05  -2.917
## nb_photo                                       2.352e+03  4.250e+03   0.553
## nb_pieces                                     -7.815e+04  4.668e+04  -1.674
## nb_chambres                                    3.930e+04  4.561e+04   0.862
## surface                                        1.394e+04  1.164e+03  11.975
## distance_station_0                             1.235e+01  1.511e+02   0.082
## distance_station_1                            -1.640e+01  1.277e+02  -0.128
## distance_station_2                            -1.383e+02  1.065e+02  -1.299
## `Cathédrale Notre-Dame de Paris`              -2.686e+02  3.936e+02  -0.682
## `Basilique du Sacré-Cœur de Montmartre`        2.365e+00  4.309e+01   0.055
## `Musée du Louvre`                              1.044e+03  3.715e+02   2.809
## `Tour Eiffel`                                 -5.735e+01  7.395e+01  -0.775
## `Centre Pompidou`                             -8.073e+01  1.950e+02  -0.414
## `Musée d'Orsay`                               -1.870e+03  5.430e+02  -3.444
## `Cité des sciences et de l'industrie`          2.013e+01  3.327e+01   0.605
## `Chapelle Notre-Dame de la Médaille miraculeuse`  5.701e+02  1.935e+02   2.945
## `Grand site du Jardin des plantes`             1.424e+02  2.194e+02   0.649
## `Arc de triomphe`                              2.482e+01  7.499e+01   0.331
## `Grand Palais`                                 4.776e+02  2.056e+02   2.323
## distance_université_plus_près                 -1.481e+01  3.567e+01  -0.415
```

```
## arrondissement                                    1.648e+04  6.360e+04   0.259
##                                                  Pr(>|t|)
## (Intercept)                                      0.220662
## appartement                                      0.000140 ***
## studio                                           0.003831 **
## chambre                                          0.004366 **
## nb_photo                                         0.581229
## nb_pieces                                        0.097239 .
## nb_chambres                                      0.390977
## surface                                           < 2e-16 ***
## distance_station_0                               0.935013
## distance_station_1                               0.898107
## distance_station_2                               0.197026
## `Cathédrale Notre-Dame de Paris`                 0.496566
## `Basilique du Sacré-Cœur de Montmartre`          0.956341
## `Musée du Louvre`                                0.005970 **
## `Tour Eiffel`                                    0.439881
## `Centre Pompidou`                                0.679710
## `Musée d'Orsay`                                  0.000839 ***
## `Cité des sciences et de l'industrie`            0.546473
## `Chapelle Notre-Dame de la Médaille miraculeuse` 0.004012 **
## `Grand site du Jardin des plantes`               0.517733
## `Arc de triomphe`                                0.741337
## `Grand Palais`                                   0.022192 *
## distance_université_plus_près                    0.678884
## arrondissement                                   0.796075
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 164400 on 100 degrees of freedom
## Multiple R-squared:  0.9284, Adjusted R-squared:  0.912
## F-statistic: 56.39 on 23 and 100 DF,  p-value: < 2.2e-16
```

Nous pouvons noter que la variable la plus signicative est la `surface`. Nous nous y attendions. Nous remarquons que les variables `Chapelle Notre-Dame de la Médaille miraculeuse` et `Musée d'Orsay` sont toutes deux significatives (au seuil de 5%), elles correspondent aux distances à ces deux monuments qui se trouvent dans le même arrondissement et tout deux plutôt au centre de Paris. Ainsi nous pouvons allons créer 3 sous-modèles l'un avec la moyenne des distance au monuments, un deuxième avec seulement `Musée d'Orsay` et un dernier avec seulement `Chapelle Notre-Dame de la Médaille miraculeuse` (toutes spécifications/modification égales par ailleurs).

La présence d'une université proche (`distance_université_plus_près`) d'un bien ne semble pas avoir d'impact significatif (au seuil de 5%) sur le prix. Pareillement pour `distance_station_0`. Nous remarquons aussi que les variables `arrondissement` (qui décrit la zone, `nb_photo`,`Centre Pompidou`,`Cathédrale Notre-Dame de Paris`, `Cité des sciences et de l'industrie`, `Grand site du Jardin des plantes` et ne sont pas non plus significatives (au seuil de 5%).

De plus les variables `surface`, `nb_pieces` et `nb_chambres` sont toutes les trois très corrélées positivements et portent une certaine redondance d'information (en effet, un bien avec une grande surface aura tendance à avoir plus de pièces et le nombre de pièce contient déjà le nombre de chambre).
Ainsi nous prenons allons remplacer ces trois variables par le rapport $\frac{surface}{nb_piece}$.

```r
# Création nouvelle variable Rapport
biens_stat_des=cbind(biens_stat_des,Rapport=biens_stat_des$surface/biens_stat_des$nb_pieces)
```

```r
# Suppression de surface, nb_pieces, nb_chambres
biens_stat_des = select(biens_stat_des, -c(surface,nb_pieces,nb_chambres))
```

#Création des 3 datasets pour les 3 sous-modèles

```r
## Dataset avec les moyennes ==> Ajout d'une variable Moyenne_ORSAY_CHAPELLE_var
Moyenne_ORSAY_CHAPELLE_var=rowMeans(biens_stat_des[,c(9,8)])

## Ajout au dataset adéquat: biens_stat_des_Moyenne
biens_stat_des_Moyenne=cbind(biens_stat_des,Moyenne_ORSAY_CHAPELLE=Moyenne_ORSAY_CHAPELLE_var)

## Suppression des autres variables
biens_stat_des_Moyenne=select(biens_stat_des_Moyenne,-c(`Musée d'Orsay`,`Chapelle Notre-Dame de la Méda
```

```r
## Dataset avec la distance au Musée d'Orsay ==> Suppression de Chapelle Notre-Dame de la Médaille mira
biens_stat_des_ORSAY=select(biens_stat_des,-c(`Chapelle Notre-Dame de la Médaille miraculeuse`))

## Dataset avec la distance à la Chapelle Notre-Dame de la Médaille miraculeuse ==> Suppression de Musé
biens_stat_des_Chapelle=select(biens_stat_des,-c(`Musée d'Orsay`))
```

3.2 Selection du meilleur 2ème Modèle

3.2.1 Procédure

Nous allons donc fitter 3 sous-modèle et selectionner le sous-modèle ayant la vairable la plus significative.
Fit Sous-modèle 1 : `Moyenne_ORSAY_CHAPELLE`

```r
X=biens_stat_des_Moyenne
lm_2_1=lm(Y~.,data=X)
summary(lm_2_1)
```

```
##
## Call:
## lm(formula = Y ~ ., data = X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1185160  -146467    -5171   140384  1503281
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             1.673e+06  2.891e+05   5.788 6.61e-08 ***
## appartement            -1.648e+06  1.728e+05  -9.540 3.81e-16 ***
## studio                 -2.036e+06  1.870e+05 -10.886  < 2e-16 ***
## chambre                -1.894e+06  2.373e+05  -7.983 1.36e-12 ***
## distance_station_1      1.765e+02  2.191e+02   0.806   0.4221
## distance_station_2     -3.910e+02  1.839e+02  -2.126   0.0357 *
## `Musée du Louvre`       1.421e+02  1.124e+02   1.265   0.2085
## `Tour Eiffel`           2.218e+01  9.150e+01   0.242   0.8089
## `Arc de triomphe`       1.480e+02  1.128e+02   1.312   0.1921
## `Grand Palais`         -2.744e+02  1.776e+02  -1.545   0.1252
```

```
## Rapport                 3.352e+04  5.312e+03   6.310 5.73e-09 ***
## Moyenne_ORSAY_CHAPELLE -1.738e+01  1.448e+02  -0.120   0.9047
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 317400 on 112 degrees of freedom
## Multiple R-squared:  0.7012, Adjusted R-squared:  0.6718
## F-statistic: 23.89 on 11 and 112 DF,  p-value: < 2.2e-16
```

3.2.2 Fit Sous-modèle 2 : `Musée d'Orsay`

```
X=biens_stat_des_ORSAY
lm_2_2=lm(Y~.,data=X)
summary(lm_2_2)
```

```
##
## Call:
## lm(formula = Y ~ ., data = X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1149989  -151546   -11194   140499  1513207
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          1.609e+06  2.957e+05   5.441 3.15e-07 ***
## appartement         -1.601e+06  1.770e+05  -9.042 5.35e-15 ***
## studio              -1.995e+06  1.902e+05 -10.491  < 2e-16 ***
## chambre             -1.855e+06  2.386e+05  -7.774 4.00e-12 ***
## distance_station_1   2.077e+02  2.189e+02   0.949   0.3448
## distance_station_2  -4.238e+02  1.826e+02  -2.321   0.0221 *
## `Musée du Louvre`    2.995e+02  1.892e+02   1.582   0.1164
## `Tour Eiffel`        7.842e+01  8.054e+01   0.974   0.3323
## `Musée d'Orsay`     -2.875e+02  2.978e+02  -0.965   0.3364
## `Arc de triomphe`    8.850e+01  1.136e+02   0.779   0.4376
## `Grand Palais`      -1.507e+02  2.161e+02  -0.697   0.4871
## Rapport              3.293e+04  5.315e+03   6.196 9.87e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 316100 on 112 degrees of freedom
## Multiple R-squared:  0.7036, Adjusted R-squared:  0.6745
## F-statistic: 24.17 on 11 and 112 DF,  p-value: < 2.2e-16
```

3.2.3 Fit Sous-modèle 3 : `Chapelle Notre-Dame de la Médaille miraculeuse`

```
X=biens_stat_des_Chapelle
lm_2_3=lm(Y~.,data=X)
summary(lm_2_3)
```

```
##
## Call:
## lm(formula = Y ~ ., data = X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1188751  -148612    -6562   142844  1505489
##
## Coefficients:
##                                                  Estimate Std. Error t value
## (Intercept)                                     1.674e+06  2.889e+05   5.795
## appartement                                    -1.656e+06  1.716e+05  -9.654
## studio                                         -2.042e+06  1.862e+05 -10.963
## chambre                                        -1.901e+06  2.367e+05  -8.034
## distance_station_1                              1.685e+02  2.188e+02   0.770
## distance_station_2                             -3.800e+02  1.838e+02  -2.067
## `Musée du Louvre`                               1.248e+02  9.239e+01   1.350
## `Tour Eiffel`                                   4.520e-01  9.322e+01   0.005
## `Chapelle Notre-Dame de la Médaille miraculeuse`  1.354e+01  9.326e+01   0.145
## `Arc de triomphe`                               1.658e+02  1.114e+02   1.488
## `Grand Palais`                                 -2.842e+02  1.716e+02  -1.656
## Rapport                                         3.362e+04  5.306e+03   6.337
##                                                 Pr(>|t|)
## (Intercept)                                     6.38e-08 ***
## appartement                                      < 2e-16 ***
## studio                                           < 2e-16 ***
## chambre                                         1.05e-12 ***
## distance_station_1                                0.4429
## distance_station_2                                0.0411 *
## `Musée du Louvre`                                 0.1797
## `Tour Eiffel`                                     0.9961
## `Chapelle Notre-Dame de la Médaille miraculeuse`  0.8848
## `Arc de triomphe`                                 0.1396
## `Grand Palais`                                    0.1004
## Rapport                                         5.04e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 317400 on 112 degrees of freedom
## Multiple R-squared:  0.7012, Adjusted R-squared:  0.6718
## F-statistic: 23.89 on 11 and 112 DF,  p-value: < 2.2e-16
```

On remarque à l'issu du fitting de ces sous-modèles que la variable la plus pertinante est `Musée d'Orsay`. Nous sélectionnons donc ce sous-modèle.

4 Raffinons le modèle

Nous créeons un 3 ème modèle, en retirant les variables `distance_station_1`,`Arc de triomphe`,`Grand Palais`,`Tour Eiffel`.

```
X=select(X,-c(distance_station_1,`Arc de triomphe`,`Grand Palais`,`Tour Eiffel`))
lm_3=lm(Y~.,data=X)
summary(lm_3)
```

```
##
## Call:
## lm(formula = Y ~ ., data = X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1197086  -156654    -8191   128865  1590217
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          1.960e+06  2.247e+05   8.724 2.28e-14 ***
## appartement         -1.701e+06  1.628e+05 -10.452  < 2e-16 ***
## studio              -2.119e+06  1.748e+05 -12.121  < 2e-16 ***
## chambre             -1.953e+06  2.297e+05  -8.504 7.36e-14 ***
## distance_station_2 -3.263e+02  1.669e+02  -1.955   0.0530 .
## `Musée du Louvre`   8.728e+01  4.834e+01   1.806   0.0736 .
## `Musée d'Orsay`    -8.656e+01  4.788e+01  -1.808   0.0732 .
## Rapport              3.178e+04  5.098e+03   6.234 7.60e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 315400 on 116 degrees of freedom
## Multiple R-squared:  0.6944, Adjusted R-squared:  0.6759
## F-statistic: 37.65 on 7 and 116 DF,  p-value: < 2.2e-16
```

4.1 Batteries de Tests

4.1.1 Test d'Homosédasticité

4.1.1.1 Test de Breusch–Pagan

```
##
##  studentized Breusch-Pagan test
##
## data:  lm_3
## BP = 84.729, df = 7, p-value = 1.489e-15
```

```
gqtest(lm_3)
```

```
##
##  Goldfeld-Quandt test
##
## data:  lm_3
## GQ = 2.6846, df1 = 54, df2 = 54, p-value = 0.0001983
## alternative hypothesis: variance increases from segment 1 to 2
```

4.1.2 Test d'autocorrélation d'ordre 1 : Drubin-Watson

```r
dwtest(lm_3)
```

```
##
##  Durbin-Watson test
##
## data:  lm_3
## DW = 2.1121, p-value = 0.6942
## alternative hypothesis: true autocorrelation is greater than 0
```

4.1.3 Test de spécification du modèle : Test de Ramsey

```r
resettest(lm_3,power = 2:3,type = 'regressor',data=biens_2)
```

```
##
##  RESET test
##
## data:  lm_3
## RESET = 1.1053, df1 = 14, df2 = 102, p-value = 0.3623
```

5 Annexes

5.1 Code Python

```python
# Initiation de la racine du site web
site_main='https://www.pap.fr'

# Création d'un set() contenant toutes les urls des biens
URLset=GetURLSET(site_main,20)
# Nettoyage du set() pour enlever les types de biens "intéréssants"
# (Fond de commerces, locaux, péniches, etc).
URLset=CleanIDset(URLset)

#Création du jeu de données brut
data=GetDetails(site_main,URLset)

#Exportation de l'objet data (dict) dans un fichier .json
exportdata(data)
# On importe les différents packages & bibliothèques
import requests
from bs4 import BeautifulSoup
from unidecode import unidecode
import json
import re
import datetime
import ast
```

```
# Fonction prenant en paramètre une URL et retournant l'Objet BeautifulSoup associé.
def GetHTMLPage(url_str):
    try:
        requete = requests.get(url_str,headers={'User-Agent':'Mozilla/5.0'})
        html_file  = requete.content
        soup = BeautifulSoup(html_file,'html.parser')
        return soup
    except :
        print("Erreur")

#Fonction :
# paramètres : URL root +  nmax : nombre page à parser.

def GetURLSET(site_main,nmax):
    res=set()
    for number in range(1,nmax+1):
        url_str=site_main+'/annonce/vente-immobiliere-paris-75-g439-'+str(number)
        soup=GetHTMLPage(url_str)
        temp=GetSet_URL_Bien(soup)
        res=res|temp
    return res


def GetSet_URL_Bien(soup):
    liste_hrefs=set()
    spans_bien=soup.find_all('div',class_="search-list-item")
    for item in spans_bien:
        str_href=item.find('a').get('href')
        liste_hrefs.add(str_href)
    return liste_hrefs

#Fonction de filtre
def condition_keep(string):
    substring_list = ("/annonces/appartement-paris","/annonces/maison-paris")
    if (string.startswith(substring_list)):
        return True
    else:
        return False
#Fonction de nettoyage de URLset.
def CleanIDset(URLset):
    keep_set=set()
    for item in URLset:
        if(condition_keep(item)):
            keep_set.add(item)
    return keep_set

#Fonction de scrapping des informations pour un bien immobilier particulier
def GetDetails(site_main,URLset):
    res=[]
    for url_detail in URLset:
        url_str=site_main+url_detail
        soup_ID = GetHTMLPage(url_str)
        to_append=ScrapDetail(soup_ID)
```

```
        to_append['url']=url_detail;
        res.append(to_append)
    return res



def Cleandataset(data):
    keys=['.','EUR le m']
    for key in keys:
        if key in data.keys():
            del data[key]
        else :continue
    return data

def GetDetailsBien(div_desc,res):
    patterns= [r'\D+']

    list_item_tag=div_desc.find(class_='item-tags')
    list_item_tag=[unidecode(item.strong.text) for item in list_item_tag.find_all('li')]
    for item in list_item_tag:
        for p in patterns:
            matches= re.findall(p, item)
        for match in matches:
            res[unidecode(match).strip()]=format_list_tag(item)
    res=Cleandataset(res)

    return res


def format_list_tag(item):
    item=re.sub('m2','',item)
    item=re.sub('[^\d]','',item)
    return int(item)

def ScrapDetail(soup):
    res={}
    type_bien=soup.find(class_='item-title').text
    res['type']=type_bien.split()[1]
    nb_photo=len(soup.find_all(class_='img-liquid owl-thumb-item'))
    res['nb_photo']=nb_photo

    div_desc=soup.find('div',class_="item-description")
#   Collecte du prix du bien
    tarif=soup.find(class_='item-price').text
    tarif=re.sub('[^\d]','',tarif)
    res['prix']=tarif

#   Collecte du code postal
    zipcode=div_desc.find('h2').text
    zipcode= re.findall(r"(\b\d{5}\b)", zipcode)[0]
    res['code postal']=zipcode
##    Collecte des Stations de métro
    list_station=div_desc.find_all(class_='item-transports')
```

```
    list_name_station=[]
    for item in list_station :
        to_append=item.find(class_='label')
        if(type(to_append)!=type(None)):
                list_name_station.append(to_append.text)

    res['transport']=list_name_station

##    Collecte des détails
    GetDetailsBien(div_desc,res)
##    Collecte des coordonées
    GetCoord(soup,res)
    return res

def GetCoord(soup,res):
#    print(carte_item)
    carte_item=ast.literal_eval(carte_item['data-mappy'])
    res['lat']=float(carte_item['center'][0])
    res['lon']=float(carte_item['center'][1])
    return res


#Fonction d'exportation du dictionnaire : dict_data
def exportdata(dict_data):
    x = datetime.datetime.now()
    x=x.strftime("%H-%M-%S")
    with open('result-'+x+'.json', 'w') as fp:
        json.dump(dict_data, fp)
    return x
```

Références

« Coordonnées GPS ». 2020. Coordonnées GPS. https://www.coordonnees-gps.fr/.

« Monuments de Paris ». 2020. Monuments de Paris. http://www.quotidiendutourisme.com/france/quels-sont-les-monuments-les-plus-visites-de-paris/170372.

« PAP: Particulier à Particulier ». 2020. Entreprise de presse immobilier français. https://pap.fr.

« RATP Open Data ». 2020. Régie autonome des transports parisiens. https://dataratp2.opendatasoft.com/explore/dataset/positions-geographiques-des-stations-du-reseau-ratp/table/?disjunctive.stop_name.

« Université de Paris ». 2020. Université de Paris. https://www.sorbonne.fr/toutes-les-universites/.