With the support of the Erasmus+ Programme of the European Union

Example presentation

Data-Based Learning Skeleton

Laurent DENIS, Q1 2022

License 1 Economics, Management & Laws

Descriptive Statistics Applied to the Real Estate Market in France

Introduction

When the Covid-19 pandemic started in France in March 2020, analyzing the impact on real estate became an important subject for men in the field.

Almost 2 years later, we now have concrete data that can be analyzed to make, among others, a factual assessment of the real impact of Covid-19 on the French real estate market.

All the graphs and all the data to be used in this course come from the Requests for Land Values made available by the French General Directorate of Public Finances.

Learning outcomes

The objective of the approach is to let students discover how descriptive statistics can be applied to big data so as to show up features and tendencies in order to figure out dependencies between variables and to forecast trends.

The objective is also to make them understand the overall process of managing data collection, data cleaning, data manipulation, before moving on to the analysis itself, all that in a real environment.

A last goal is to let them experiment a most-renowned open-source software that all statisticians and data scientists are used to handling.

Description

This RLV data set as “Requests for Land Values”, published and produced by the Directorate General of Public Finances, provides information on real estate transactions that have taken place over the past five years in France. This data comes from notarial deeds and cadastral information. The last update is as of October 20, 2021 and will be used as a reference to compare our models to reality.

THEME 0 : DATA PREPARATION

Download the R software

Get to the following URL address: "https://www.rstudio.com/products/rstudio/download/#download" and install the free software once downloaded.

Open it, create a file by reaching "File" -> "New File" -> "R Script" and type in as follows:

Upload libraries

```
library(tidyverse)
library(tidymodels)
library(lubridate)
library(ggplot2)
library(here)
library(data.table)
library(kableExtra)
library(gmodels)
library(IndexNumber)
library(optimbase)
library(prophet)
```

Get the data

1. Join the following URL: "https://www.data.gouv.fr/fr/datasets/demandes-de-valeurs-foncieres/#resource-d4e727f0-dff7-4f97-b6f5-d56c0296f453-header"
2. Scroll down to the section called "Fichier CSV agrégé, reformaté, homonégéisé" by Christian Quest then
3. Download the 2GB file by clicking on the green bullet over "csv"
4. Call it "dvf.csv" and load it in the R project folder

```
db_filtered <- read.csv2(here("dvf.csv"),
                         sep =",", header = TRUE)
# First view
glimpse(db_filtered)
```

How many columns are there?

Are the variable types all correct?

Adjust variable types

```
# Date formatting
db_filtered$date_mutation <- as.Date(db_filtered$date_mutation, format="%Y-%m-%d")

# Values formatting
db_filtered <- db_filtered %>%
  mutate_at(c("surface_lot_1", "surface_lot_2", "surface_lot_3",
```

```
              "surface_lot_4", "surface_lot_51"), as.integer) %>%
  mutate_at("valeur_fonciere", as.numeric)

# Factor formatting
db_filtered <- db_filtered %>%
  mutate_at(c("numero_disposition", "nature_mutation", "type_voie", "type_local",
              "code_departement", "code_commune", "section", "nature_culture",
              "nature_culture_speciale"), as.factor)

# Check
glimpse(db_filtered)
```

Deal with missing data

```
#Assess & plot missing vs. observed values
skimr::skim(db_filtered)

# View NA lines to get rid of
cbind("NA's" = dim(db_filtered)-dim(na.exclude(db_filtered)),"Others" =
dim(na.exclude(db_filtered)))

# remove NA's
db_filtered_NA <- na.exclude(db_filtered)
skimr::skim(db_filtered_NA)
```

Data cleanup

```
# Logical variable fields look empty => get rid of them
db_filtered <- db_filtered %>%
  select(-c("code_service_ch", "reference_document", "articles_1", "articles_2",
            "articles_3", "articles_4", "articles_5", "identifiant_local"))

# 1% of real estate valuation field is empty => suppress corresponding records
db_filtered <- db_filtered %>%
  filter(!is.na(valeur_fonciere))

# Check
skimr::skim(db_filtered)
```

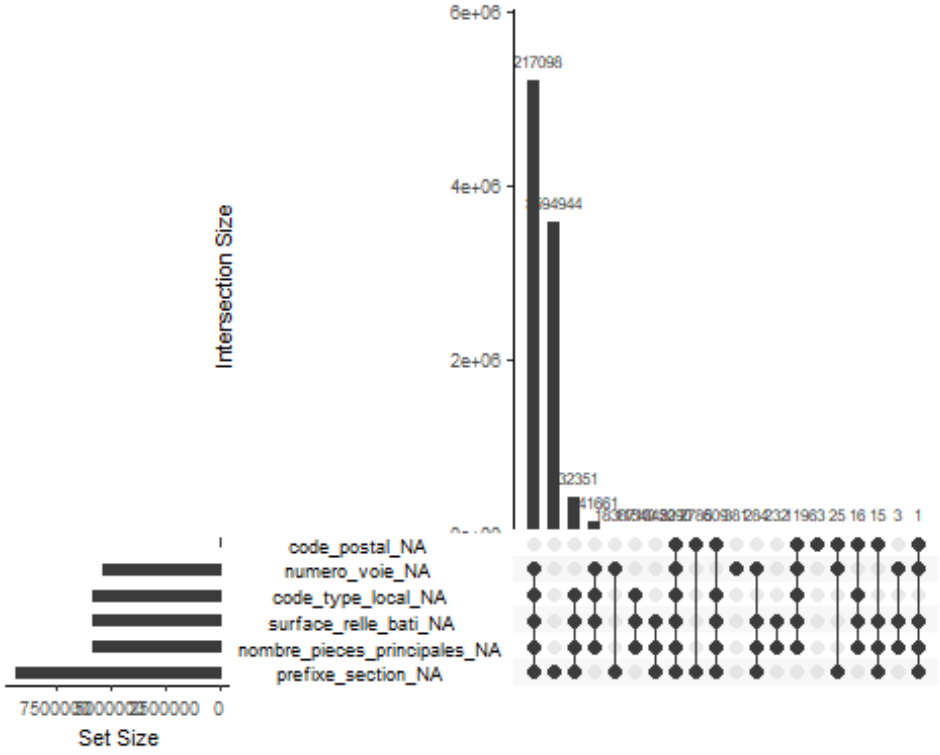
Missing data and relative interdependencies lookup

```
library(naniar)
gg_miss_upset(db_filtered, nsets = 20)
```

Final Selection of fields of interest

```
# Ground surface is an important field so let's remove all NA's in this field
db_filtered <- db_filtered %>%
  filter(!is.na(surface_terrain))
dim(db_filtered)

skimr::skim(db_filtered)
```

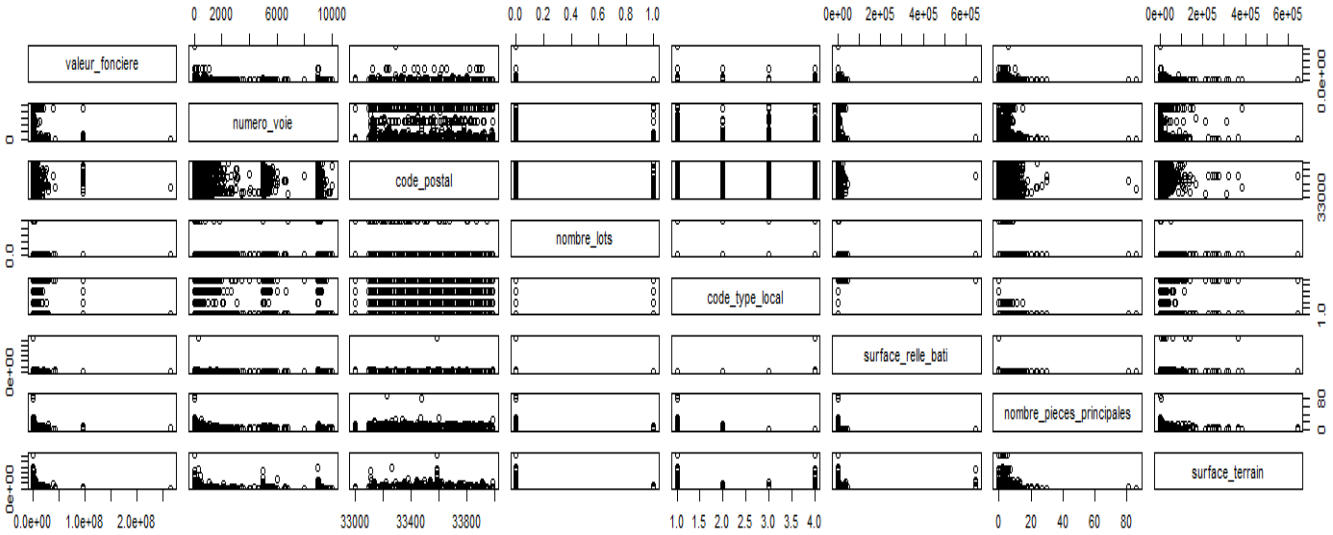
```
# Surface_lots variables can now be excluded
db_filtered <- db_filtered %>%
  select(-c("surface_lot_1", "surface_lot_2", "surface_lot_3",
            "surface_lot_4", "surface_lot_51"))
gg_miss_upset(db_filtered, nsets = 6)
```




```
# Main room number variable is important so let's finish the cleanup with it
db_filtered <- db_filtered %>%
  filter(!is.na(nombre_pieces_principales))
# Assess how many records keeping NA's
cbind("With NAs" = dim(db_filtered)[1],"Without NAs" =
dim(na.exclude(db_filtered))[1])

# As remaining NA's belong to non-critical variables, let's keep it as is
summary(db_filtered)
```

Focus on a specific region and first graphical view to compare each pairs of numeric variables

```
db_ex1 <- db_filtered %>%
  select(-prefixe_section) %>%
  filter(code_departement == 33)

db_ex1 %>% select(where(is.numeric)) %>% na.exclude() %>% pairs()
```



Let's dig in this data by highlighting real estate value versus habitable surface

THEME 1: BI-DIMENSIONAL TABLES & COMMON STATISTICAL METRICS

Create the contingency table

Selection of the 2 dimensions, data gathering and contingency table setup

```
tab_cont <- table(
  "Valeur foncière" = case_when(db_ex1$valeur_fonciere <= 100000 ~ '<100K',
                                db_ex1$valeur_fonciere <= 300000 ~ '101K-300K',
                                db_ex1$valeur_fonciere <= 500000 ~ '301K-500K',
                                db_ex1$valeur_fonciere <= 1000000 ~ '501K-1M',
                                TRUE ~ 'Over 1M'),
  "Surface bâtiment" = case_when(db_ex1$surface_relle_bati <= 50 ~ '<50m²',
                                 db_ex1$surface_relle_bati <= 100 ~ '051m²-100m²',
                                 db_ex1$surface_relle_bati <= 200 ~ '101m²-200m²',
                                 db_ex1$surface_relle_bati <= 500 ~ '201m²-500m²',
                                 TRUE ~ 'Over 500m²'))
tab_cont
```

```
##                 Surface bâtiment
## Valeur foncière <50m² 051m²-100m² 101m²-200m² 201m²-500m² Over 500m²
##       <100K      8032        6922        2449         298        170
##       101K-300K  7372       28846       17832        1111        209
##       301K-500K  3134        6499       11306         981        230
##       501K-1M    3046        2290        4536        1370        352
##       Over 1M    2751        3837        1330        1004        734
```

Table augmentation with conditional sums

```
Somme = sum
tab_cont_sum <- tab_cont %>% addmargins(FUN = Somme, quiet = TRUE)
tab_cont_sum
```

```
##                 Surface bâtiment
## Valeur foncière  <50m² 051m²-100m² 101m²-200m² 201m²-500m² Over 500m²  Somme
##       <100K       8032        6922        2449         298        170  17871
##       101K-300K   7372       28846       17832        1111        209  55370
##       301K-500K   3134        6499       11306         981        230  22150
##       501K-1M     3046        2290        4536        1370        352  11594
##       Over 1M     2751        3837        1330        1004        734   9656
##       Somme      24335       48394       37453        4764       1695 116641
```

Get common metrics

```
# Get marginal means
db_ex1$valeur_fonciere %>% mean()
```

```
## [1] 1861663
```

```
db_ex1$surface_relle_bati %>% mean()
```

```
## [1] 171.7517
```

```
# Get quantiles
db_ex1$valeur_fonciere %>% quantile(0.25)
```

```
##    25%
## 146000
```

```
db_ex1$valeur_fonciere %>% median()
```

```
## [1] 240000
```

```
db_ex1$valeur_fonciere %>% quantile(0.75)
```

```
##    75%
## 399100
```

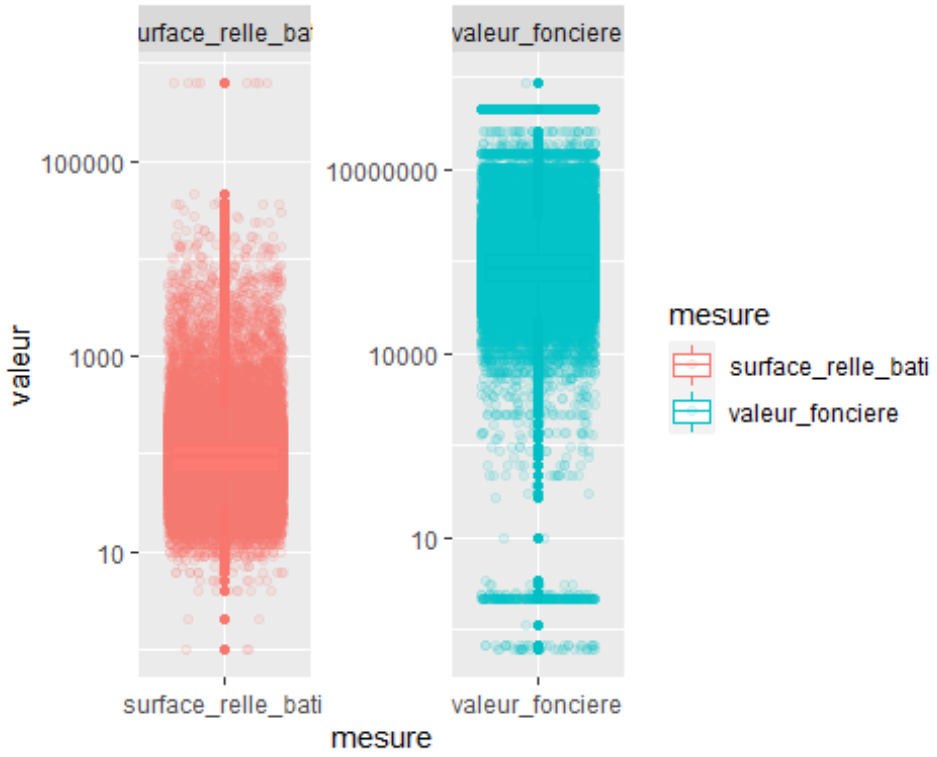
```
# Get all above values
rbind("valeur foncière" = db_ex1$valeur_fonciere %>% summary(),
      "Surface bâtiment" = db_ex1$surface_relle_bati %>% summary())
```

```
##                   Min. 1st Qu. Median         Mean 3rd Qu.      Max.
## valeur foncière      0  146000 240000 1861663.4785  399100 264831008
## Surface bâtiment     0      60     89     171.7517     120    646230
```

View the global distributions for each variable of interest

```
options(scipen = 999)
db <- db_ex1 %>% select_at(c("valeur_fonciere", "surface_relle_bati"))
db %>%
```

```
  gather(key = "mesure", value = "valeur") %>%
  ggplot(aes(x = mesure, y = valeur, color = mesure)) +
  geom_boxplot() +
  geom_jitter(alpha = 0.1) +
  facet_wrap(~mesure, scales = "free") +
  scale_y_log10()
```



Set a frequency table

As there are many different values, you first need to gather the values in groups

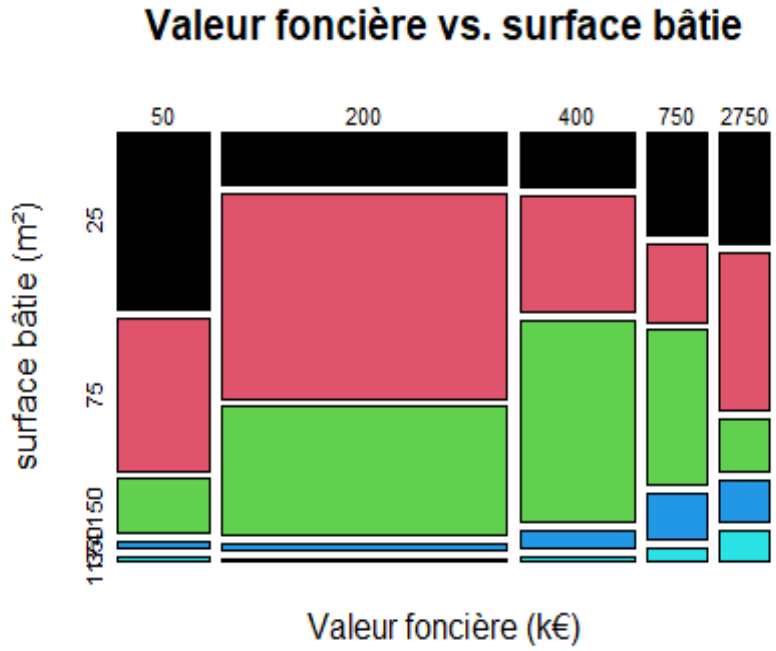
```
db_ex1 <- db_ex1 %>% mutate(
  "groupe_val_fonciere" = case_when(db_ex1$valeur_fonciere <= 100000 ~ 50,
                                    (db_ex1$valeur_fonciere <= 300000 &
db_ex1$valeur_fonciere > 100000) ~ 200,
                                    (db_ex1$valeur_fonciere <= 500000 &
db_ex1$valeur_fonciere > 300000) ~ 400,
                                    (db_ex1$valeur_fonciere <= 1000000 &
db_ex1$valeur_fonciere > 500000) ~ 750,
                                    TRUE ~ 2750))
# 2750 is obtained by getting the median of the upper range of real estate values, as
follows
skimr::skim(db_ex1 %>% filter(valeur_fonciere > 1000000) %>% select(valeur_fonciere))

db_ex1 <- db_ex1 %>% mutate(
  "groupe_surface_bati" = case_when(db_ex1$surface_relle_bati <= 50 ~ 25,
                                    (db_ex1$surface_relle_bati <= 100 &
```

```
db_ex1$surface_relle_bati > 50) ~ 75,
                                    (db_ex1$surface_relle_bati <= 200 &
db_ex1$surface_relle_bati > 100) ~ 150,
                                    (db_ex1$surface_relle_bati <= 500 &
db_ex1$surface_relle_bati > 200) ~ 350,
                                    TRUE ~ 1174)) # Obtained as above, do it yourself!

# Create the "fij" frequency table
prop.table(addmargins(xtabs(~ groupe_surface_bati + groupe_val_fonciere, data =
db_ex1)))

# Improve the appearance
prop.table(xtabs(~ groupe_surface_bati + groupe_val_fonciere, data = db_ex1)) %>%
  addmargins() %>%
  apply(MARGIN=c(1,2), FUN=scales::percent, accuracy=0.1) %>%
  noquote

# Improve a bit more
prop.table(xtabs(~ groupe_surface_bati + groupe_val_fonciere, data = db_ex1)) %>%
  addmargins() %>%
  apply(MARGIN=c(1,2), FUN=scales::percent, accuracy=0.1) %>%
  noquote %>%
  kbl() %>%
  kable_paper(bootstrap_options = "striped", full_width = F, position = "left",
font_size = 12, fixed_thead = T) %>%
  column_spec(1, width = "10", bold = T) %>%
  column_spec(as.numeric(count(as.data.frame(unique(db_ex1$groupe_surface_bati))))+2,
bold = T, color = "white", background = "black") %>%
  row_spec(as.numeric(count(as.data.frame(unique(db_ex1$groupe_val_fonciere))))+1,
bold = T, color = "white", background = "black")

# Create the "fi/j" conditional frequency table
prop.table(xtabs(~ groupe_surface_bati + groupe_val_fonciere, data = db_ex1), margin =
2) %>%
  addmargins(margin = 1) %>%
  apply(MARGIN=2, FUN=scales::percent, accuracy=0.1) %>%
  noquote %>%
  kbl() %>%
  kable_paper(bootstrap_options = "striped", full_width = F, position = "left",
font_size = 12, fixed_thead = T) %>%
  column_spec(1, width = "10", bold = T) %>%
  row_spec(as.numeric(count(as.data.frame(unique(db_ex1$groupe_surface_bati))))+1,
bold = T, color = "white", background = "black")

# Create the "fj/i" conditional frequency table
prop.table(xtabs(~ groupe_surface_bati + groupe_val_fonciere, data = db_ex1), margin =
1) %>%
  addmargins(margin = 2) %>%
  apply(MARGIN=2, FUN=scales::percent, accuracy=0.1) %>%
  noquote %>%
```

```
  kbl() %>%
  kable_paper(bootstrap_options = "striped", full_width = F, position = "left",
font_size = 12, fixed_thead = T) %>%
  column_spec(1, width = "10", bold = T) %>%
  column_spec(as.numeric(count(as.data.frame(unique(db_ex1$groupe_val_fonciere))))+2,
bold = T, color = "white", background = "black")

# To get all the interesting metrics as above in an all-in-one table (discard the Chi-
square contribution on line 2 for now)
CrossTable(db_ex1$groupe_val_fonciere, db_ex1$groupe_surface_bati, prop.t = TRUE,
prop.r = TRUE, prop.c = TRUE)

##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  116641
##
##
##                             | db_ex1$groupe_surface_bati
## db_ex1$groupe_val_fonciere |        25 |        75 |       150 |       350 |
1174 | Row Total |
## ---------------------------|-----------|-----------|-----------|-----------|-------
----|-----------|
##                         50 |      8032 |      6922 |      2449 |       298 |
170 |     17871 |
##                            |  4967.337 |    32.730 |  1885.498 |   255.574 |
30.981 |           |
##                            |     0.449 |     0.387 |     0.137 |     0.017 |
0.010 |     0.153 |
##                            |     0.330 |     0.143 |     0.065 |     0.063 |
0.100 |           |
##                            |     0.069 |     0.059 |     0.021 |     0.003 |
0.001 |           |
## ---------------------------|-----------|-----------|-----------|-----------|-------
----|-----------|
##                        200 |      7372 |     28846 |     17832 |      1111 |
209 |     55370 |
##                            |  1512.460 |  1501.509 |     0.157 |   585.291 |
440.911 |           |
##                            |     0.133 |     0.521 |     0.322 |     0.020 |
0.004 |     0.475 |
##                            |     0.303 |     0.596 |     0.476 |     0.233 |
```

```
0.123 |           |
##                            |     0.063 |     0.247 |     0.153 |     0.010 |
0.002 |           |
## --------------------------|-----------|-----------|-----------|-----------|-------
----|-----------|
##                        400 |      3134 |      6499 |     11306 |       981 |
230 |     22150 |
##                            |   478.607 |   787.958 |  2472.799 |     6.439 |
26.226 |           |
##                            |     0.141 |     0.293 |     0.510 |     0.044 |
0.010 |     0.190 |
##                            |     0.129 |     0.134 |     0.302 |     0.206 |
0.136 |           |
##                            |     0.027 |     0.056 |     0.097 |     0.008 |
0.002 |           |
## --------------------------|-----------|-----------|-----------|-----------|-------
----|-----------|
##                        750 |      3046 |      2290 |      4536 |      1370 |
352 |     11594 |
##                            |   162.590 |  1320.494 |   177.638 |  1697.114 |
199.898 |           |
##                            |     0.263 |     0.198 |     0.391 |     0.118 |
0.030 |     0.099 |
##                            |     0.125 |     0.047 |     0.121 |     0.288 |
0.208 |           |
##                            |     0.026 |     0.020 |     0.039 |     0.012 |
0.003 |           |
## --------------------------|-----------|-----------|-----------|-----------|-------
----|-----------|
##                       2750 |      2751 |      3837 |      1330 |      1004 |
734 |      9656 |
##                            |   269.223 |     7.150 |  1011.026 |   942.317 |
2511.834 |           |
##                            |     0.285 |     0.397 |     0.138 |     0.104 |
0.076 |     0.083 |
##                            |     0.113 |     0.079 |     0.036 |     0.211 |
0.433 |           |
##                            |     0.024 |     0.033 |     0.011 |     0.009 |
0.006 |           |
## --------------------------|-----------|-----------|-----------|-----------|-------
----|-----------|
##               Column Total |     24335 |     48394 |     37453 |      4764 |
1695 |    116641 |
##                            |     0.209 |     0.415 |     0.321 |     0.041 |
0.015 |           |
## --------------------------|-----------|-----------|-----------|-----------|-------
----|-----------|
##
##
```

```
# You can also view the bi-distribution graphically
mosaicplot(table(db_ex1$groupe_val_fonciere, db_ex1$groupe_surface_bati),
           main = "Valeur foncière vs. surface bâtie", xlab = "Valeur foncière (k€)",
ylab = "surface bâtie (m²)",
           color = 1:5)
```



Verification of the relation between frequencies

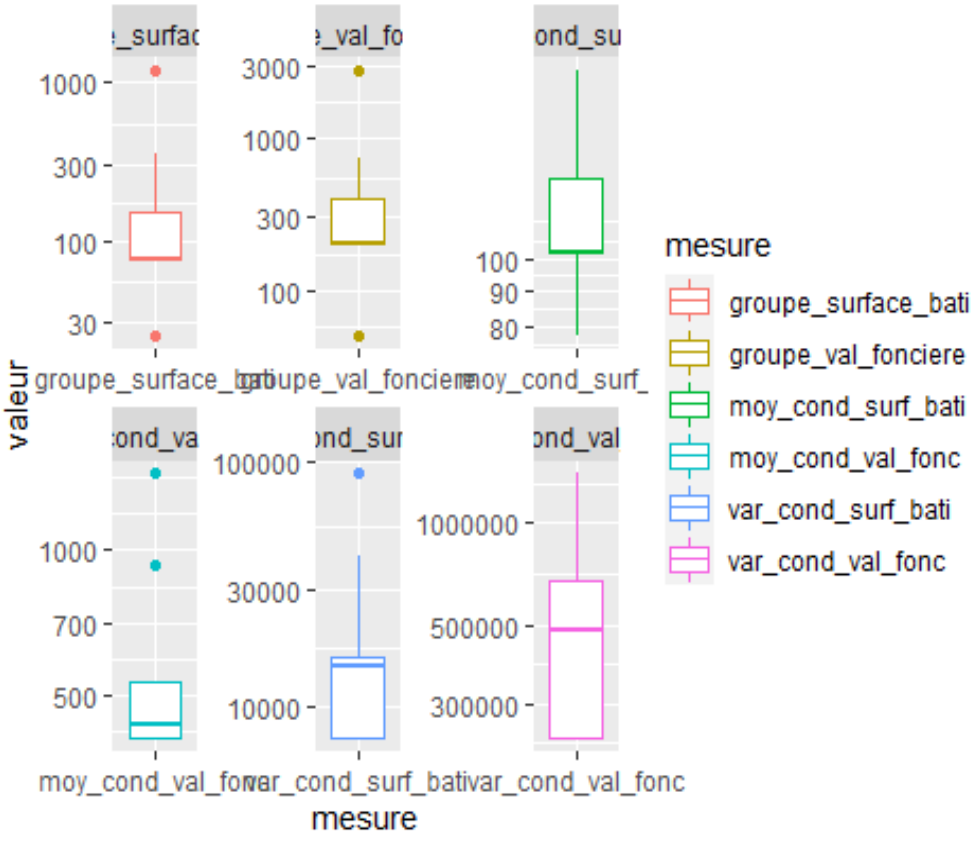
```
# Table definition (as seen above)
tab_fij <- prop.table(xtabs(~ groupe_surface_bati + groupe_val_fonciere, data =
db_ex1)) %>%
  addmargins()
tab_f.j <- prop.table(xtabs(~ groupe_surface_bati + groupe_val_fonciere, data =
db_ex1), margin = 2) %>%
  addmargins(margin = 1)
tab_fi. <- prop.table(xtabs(~ groupe_surface_bati + groupe_val_fonciere, data =
db_ex1), margin = 1) %>%
  addmargins(margin = 2)

# formula: fi/j = fij / f.j -> for instance: i = 4 and j = 3?
fi_j <- tab_f.j[4,3]
fverif <- tab_fij[4,3] / tab_fij[nrow(tab_f.j),3]
fi_j - fverif

## [1] 0

# formula: fj/i = fij / fi. -> for instance: i = 4 and j = 3?
fj_i <- tab_fi.[4,3]
fverif <- tab_fij[4,3] / tab_fij[4,ncol(tab_fi.)]
fj_i - fverif
```

```
## [1] 0.000000000000000002775558
```

Conditional means and variances

```
# Calculate conditional means and variances by selecting the interesting variables and
group by values for each
db <- db_ex1 %>%
  select_at(c("groupe_surface_bati", "groupe_val_fonciere")) %>%
  group_by(groupe_surface_bati) %>%
  mutate(moy_cond_val_fonc = mean(groupe_val_fonciere)) %>%
  mutate(var_cond_val_fonc = var(groupe_val_fonciere)) %>%
  ungroup() %>%
  group_by(groupe_val_fonciere) %>%
  mutate(moy_cond_surf_bati = mean(groupe_surface_bati)) %>%
  mutate(var_cond_surf_bati = var(groupe_surface_bati)) %>%
  ungroup()

# Create the various tables
db %>%
  group_by(groupe_surface_bati) %>%
  summarise(moy_cond_val_fonc = mean(groupe_val_fonciere)) %>%
  kbl() %>%
  kable_paper(bootstrap_options = "striped", full_width = F, position = "left",
font_size = 12, fixed_thead = T)

db %>%
  group_by(groupe_surface_bati) %>%
  summarise(var_cond_val_fonc = var(groupe_val_fonciere)) %>%
  kbl() %>%
  kable_paper(bootstrap_options = "striped", full_width = F, position = "left",
font_size = 12, fixed_thead = T)

db %>%
  group_by(groupe_val_fonciere) %>%
  summarise(moy_cond_surf_bati = mean(groupe_surface_bati)) %>%
  kbl() %>%
  kable_paper(bootstrap_options = "striped", full_width = F, position = "left",
font_size = 12, fixed_thead = T)

db %>%
  group_by(groupe_val_fonciere) %>%
  summarise(var_cond_surf_bati = var(groupe_surface_bati)) %>%
  kbl() %>%
  kable_paper(bootstrap_options = "striped", full_width = F, position = "left",
font_size = 12, fixed_thead = T)

db %>%
  gather(key = "mesure", value = "valeur") %>%
  ggplot(aes(x = mesure, y = valeur, color = mesure)) +
  geom_boxplot() +
```

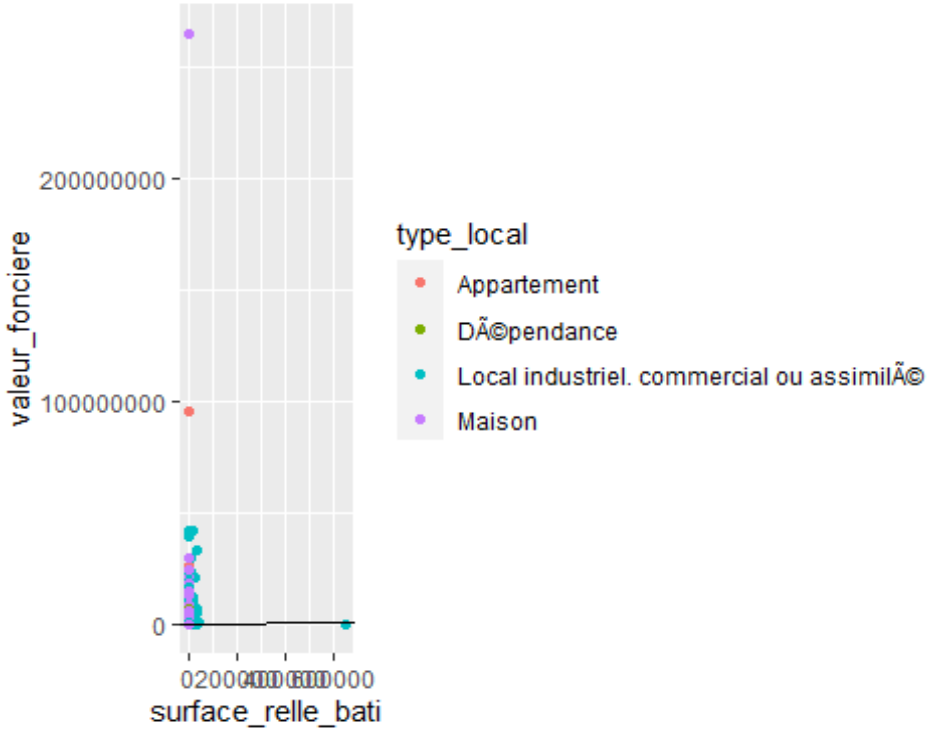
```
  facet_wrap(~mesure, scales = "free") +
  scale_y_log10()
```



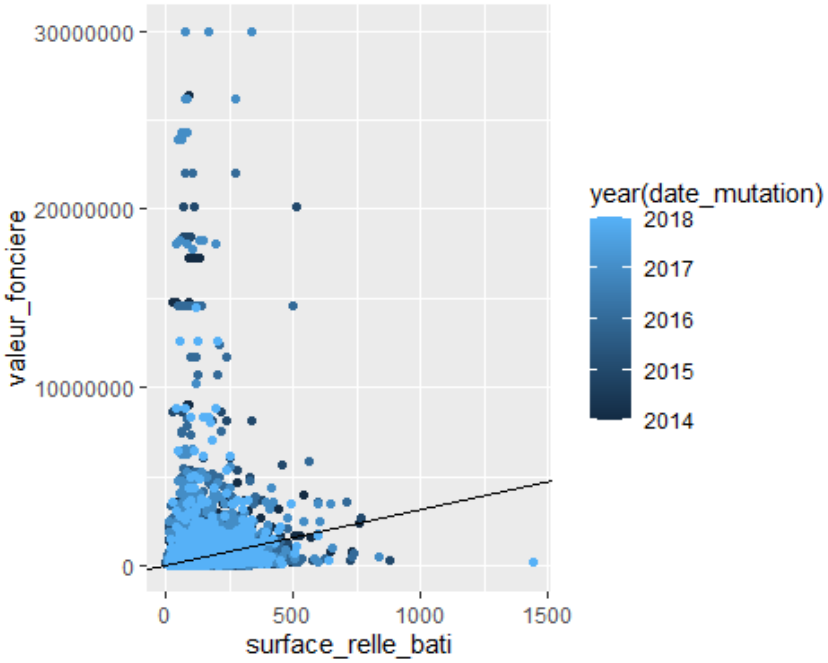
THEME 2: REGRESSION, CORRELATION AND LINEAR ADJUSTMENT

Compare habitation types and refine data selection

```
db_ex1 %>%
  ggplot(aes(x = surface_relle_bati, y = valeur_fonciere, color = type_local)) +
  geom_point() +
  geom_abline(slope = 1)

db_ex2 <- db_ex1 %>%
  filter(surface_relle_bati < 200000) %>%
  filter(valeur_fonciere < 50000000) %>%
  filter(type_local == "Maison")
```

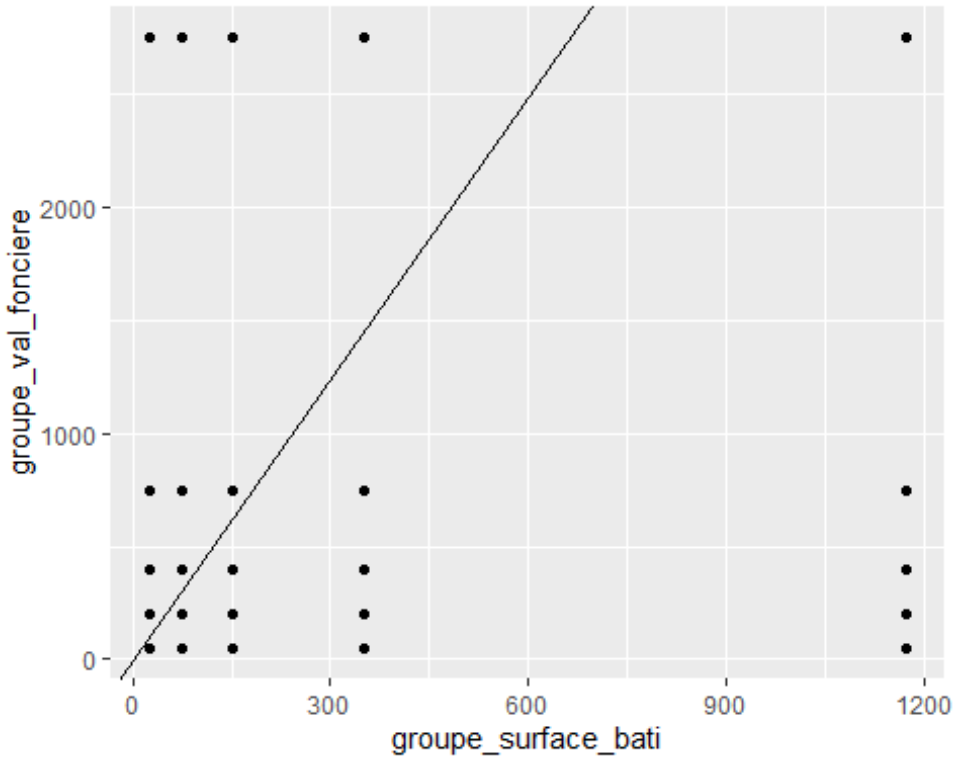



```
X1 <- db_ex2$valeur_fonciere %>% mean()
X2 <- db_ex2$surface_relle_bati %>% mean()
pente <- X1/X2

db_ex2 %>%
  ggplot(aes(x = surface_relle_bati, y = valeur_fonciere, color =
year(date_mutation))) +
  geom_point() +
  geom_abline(slope = pente) # Assuming a constant rate based on marginal mean ratio
```



Recalculate the slope based on the "db" resource

```
X1 <- db$groupe_val_fonciere %>% mean()
X2 <- db$groupe_surface_bati %>% mean()
pente2 <- X1/X2

db %>%
  ggplot(aes(x = groupe_surface_bati, y = groupe_val_fonciere)) +
  geom_point() +
  #geom_jitter(alpha = 0.1) +
  geom_abline(slope = pente2)
```



Calculation of linear regression parameters

```
# Covariance
cov(db$groupe_surface_bati, db$groupe_val_fonciere)

## [1] 17877.49

cov(db_ex2$surface_relle_bati, db_ex2$valeur_fonciere)

## [1] 6160150

# slope "a"
cov(db$groupe_surface_bati, db$groupe_val_fonciere)/var(db$groupe_surface_bati)

## [1] 0.8393495

cov(db_ex2$surface_relle_bati, db_ex2$valeur_fonciere)/var(db_ex2$surface_relle_bati)

## [1] 2343.1

# Correlation
cor(db$groupe_surface_bati, db$groupe_val_fonciere)

## [1] 0.1731347

cor(db_ex2$surface_relle_bati, db_ex2$valeur_fonciere)

## [1] 0.1509169
```

```
# Get covariance among others
lm(db$groupe_val_fonciere ~ db$groupe_surface_bati)

## Coefficients:
##            (Intercept)  db$groupe_surface_bati
##               383.5251                  0.8393

lm(db_ex2$valeur_fonciere ~ db_ex2$surface_relle_bati)

## Coefficients:
##               (Intercept)  db_ex2$surface_relle_bati
##                     84200                       2343
```

Back to the overall data: main numeric variables affecting the real estate valuation <- "FEATURE ENGINEERING"

```
# 2 additional libraries are needed
library(tidymodels)
library(vip)

# Filter the numeric variables
db_ex2 <- db_ex1 %>%
  select_if(is.numeric)

# Use set.seed() to ensure your results are reproducible
set.seed(2022)

# Create a split object
db_split <- initial_split(db_ex2, prop = 0.75, strata = nombre_pieces_principales)

# Build training & testing data sets
db_train <- training(db_split)
db_test  <- testing(db_split)

# Create a linear regression model object
lm_model <- linear_reg() %>%
  set_engine('lm') %>%
  set_mode('regression')

# View object properties
lm_model

# Assign the model to the data
lm_fit <- lm_model %>%
          fit(valeur_fonciere ~ ., data = db_train)

# View lm_fit properties
lm_fit
```

```
## Coefficients:
##                (Intercept)                 numero_voie
##           -87553807.04347                   -229.84922
##               code_postal                 nombre_lots
##                2675.79838               -664963.58261
##           code_type_local           surface_relle_bati
##            -1384149.17427                    15.46313
## nombre_pieces_principales              surface_terrain
##             -346260.59420                    -0.09404
##       groupe_val_fonciere          groupe_surface_bati
##                8125.20283                 -5869.18200
```

Obtain the detailed results in a data frame

```
# Takes a linear regression object and return model coefficients + associated metrics.
tidy(lm_fit)
```

```
##    term                       estimate   std.error statistic   p.value
##    <chr>                         <dbl>       <dbl>     <dbl>     <dbl>
##  1 (Intercept)               -8.76e+7 4508333.      -19.4   7.81e- 84
##  2 numero_voie               -2.30e+2      20.7     -11.1   1.35e- 28
##  3 code_postal                2.68e+3     135.       19.9   9.46e- 88
##  4 nombre_lots               -6.65e+5  677743.       -0.981 3.27e-  1
##  5 code_type_local           -1.38e+6   60751.      -22.8   1.44e-114
##  6 surface_relle_bati         1.55e+1       6.84      2.26  2.37e-  2
##  7 nombre_pieces_principales -3.46e+5   25565.      -13.5   9.43e- 42
##  8 surface_terrain           -9.40e-2       0.508    -0.185 8.53e-  1
##  9 groupe_val_fonciere        8.13e+3      50.4     161.    0
## 10 groupe_surface_bati       -5.87e+3     256.      -23.0   2.15e-116
```

```
# Obtain performance metrics from the training data
glance(lm_fit)
```

```
##   r.squared adj.r.squared    sigma statistic p.value    df  logLik    AIC    BIC
##       <dbl>         <dbl>    <dbl>     <dbl>   <dbl> <dbl>   <dbl>  <dbl>  <dbl>
## 1     0.234         0.234 9877143.     2949.       0     9 -1.52e6 3.05e6 3.05e6
```

Training result exploration

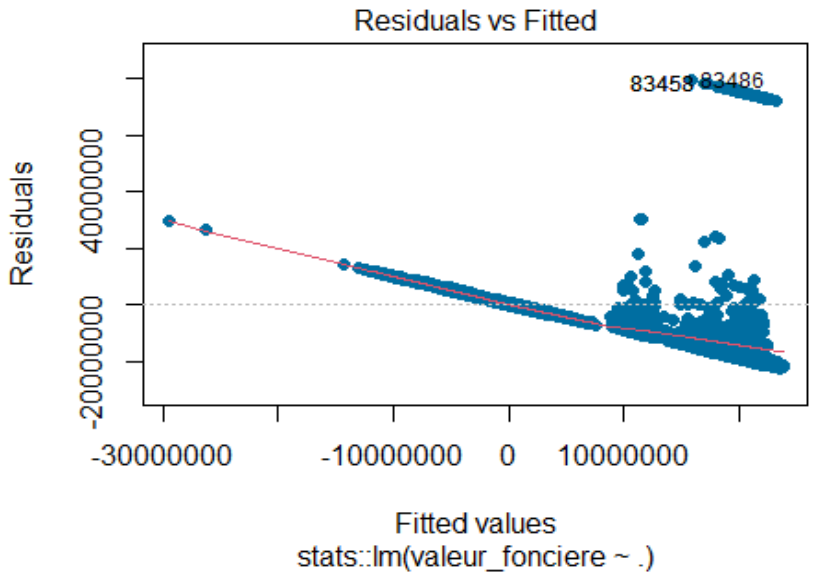
```
# View the named objects that are stored within lm_fit
names(lm_fit)
```

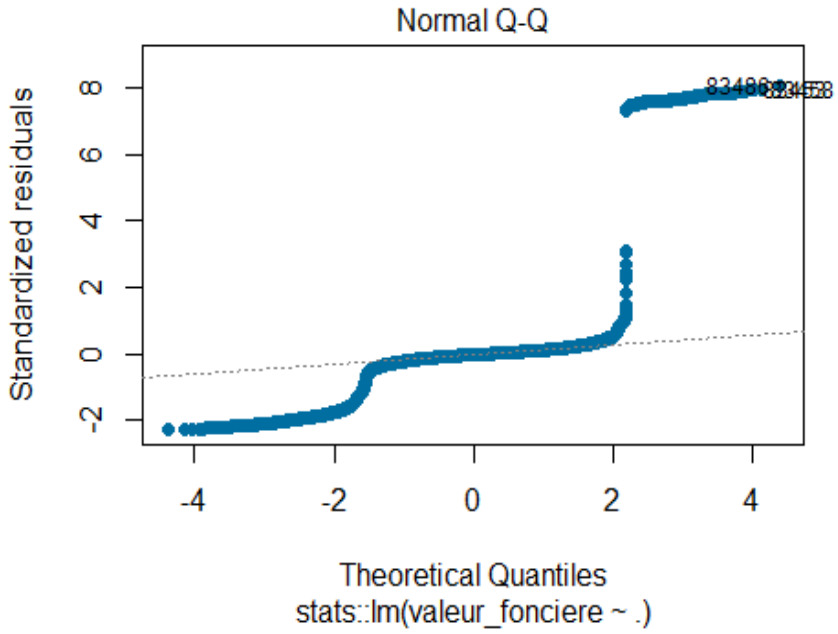
```
# Explore the estimated coefficients
summary(lm_fit$fit)
```

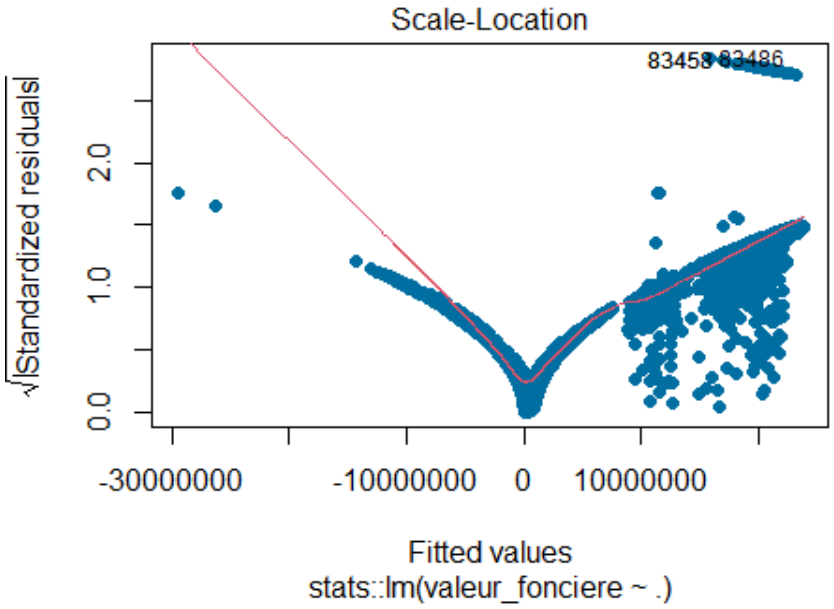
```
## Coefficients:
##                              Estimate      Std. Error t value
## (Intercept)              -87553807.04347   4508332.69446 -19.420
## numero_voie                   -229.84922        20.71192 -11.097
## code_postal                   2675.79838       134.59579  19.880
## nombre_lots                 -664963.58261    677743.13409  -0.981
## code_type_local            -1384149.17427     60751.31011 -22.784
## surface_relle_bati               15.46313         6.83772   2.261
```

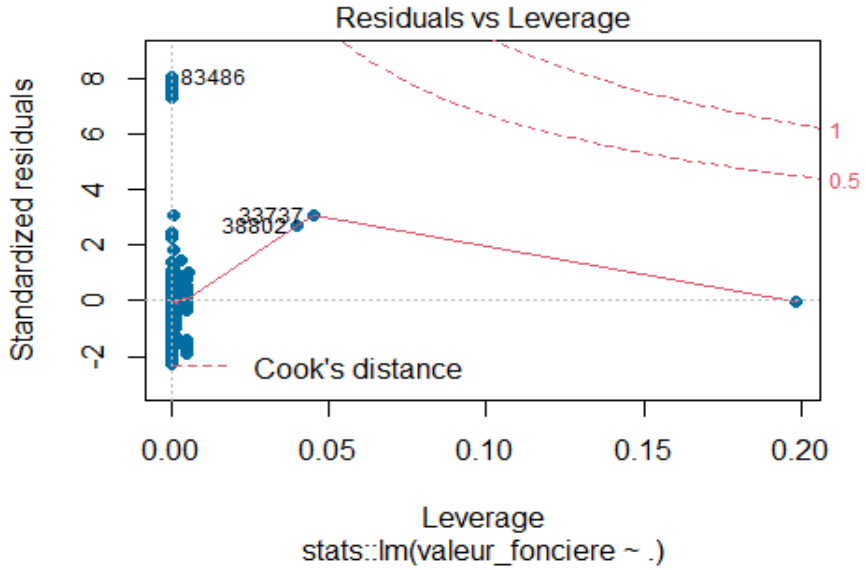
```
## nombre_pieces_principales   -346260.59420     25564.82819 -13.544
## surface_terrain                 -0.09404         0.50841  -0.185
## groupe_val_fonciere           8125.20283        50.37017 161.310
## groupe_surface_bati          -5869.18200       255.53320 -22.968
##                                            Pr(>|t|)
## (Intercept)                <0.0000000000000002 ***
## numero_voie                <0.0000000000000002 ***
## code_postal                <0.0000000000000002 ***
## nombre_lots                                0.3265
## code_type_local            <0.0000000000000002 ***
## surface_relle_bati                         0.0237 *
## nombre_pieces_principales  <0.0000000000000002 ***
## surface_terrain                            0.8532
## groupe_val_fonciere        <0.0000000000000002 ***
## groupe_surface_bati        <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9877000 on 86924 degrees of freedom
##   (545 observations effacées parce que manquantes)
## Multiple R-squared:  0.2339, Adjusted R-squared:  0.2339
## F-statistic:  2949 on 9 and 86924 DF,  p-value: < 0.00000000000000022
```

```
# Use the plot() function to obtain diagnostic plots
plot(lm_fit$fit,
     pch = 16,
     col = '#006EA1')
```

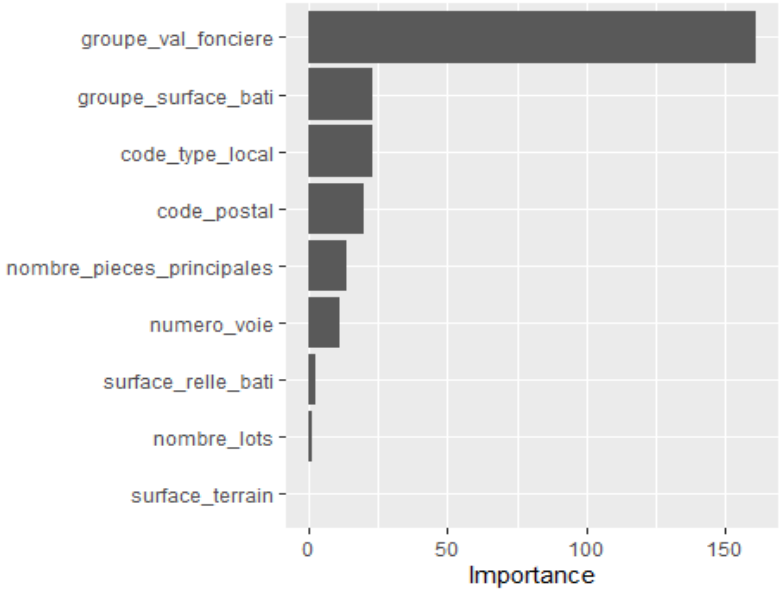



Normal Q-Q
Standardized residuals
-2
0
2
4
6
8
-4
-2
0
2
4
Theoretical Quantiles
stats::lm(valeur_fonciere ~ .)


Scale-Location
√|Standardized residuals|
0.0
1.0
2.0
-30000000
-10000000
0
10000000
Fitted values
stats::lm(valeur_fonciere ~ .)




```
# Plot the variable importance for each predictor in the model
vip(lm_fit)
```



Assess the accuracy of the trained model

```
db_test_res <- predict(lm_fit, new_data = db_test) %>% bind_cols(db_test)
# View results
db_test_res
```

```
## # A tibble: 29,162 x 11
##        .pred valeur_fonciere numero_voie code_postal nombre_lots code_type_local
##        <dbl>           <dbl>       <int>       <int>       <int>           <int>
##  1  -415272.          233000          34       33160           0               1
##  2  1229597.          306070          24       33460           0               1
```

```
##  3    20118.          280000          2       33320           0               1
##  4 -1195650.          300000         17       33290           0               1
##  5     2756.          180000         78       33320           0               1
##  6   430324.          318000          6       33160           0               1
##  7 19190251.         1550000          6       33000           0               2
##  8 18791278.         1550000          6       33000           0               3
##  9 18404575.         1550000          6       33000           0               2
## 10  1324859.          277000         24       33680           0               1
```

Assess the validity of the model

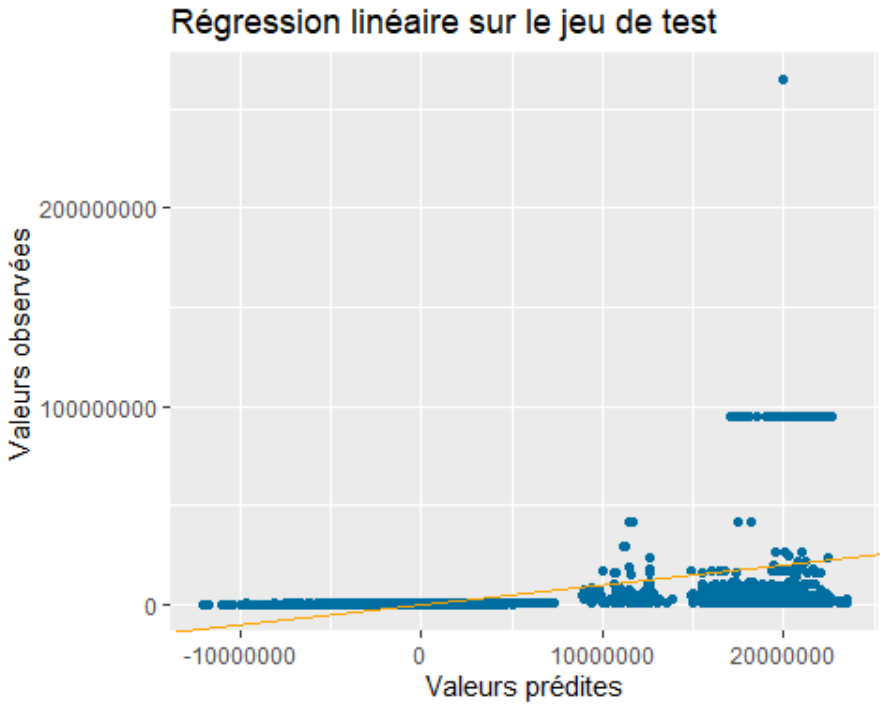
```
# RMSE on test set
rmse(db_test_res,
     truth = valeur_fonciere,
     estimate = .pred)
```

```
## 1 rmse    standard    9828373.
```

```
# R2 on test set
rsq(db_test_res,
    truth = valeur_fonciere,
    estimate = .pred)
```

```
## 1 rsq     standard       0.226
```

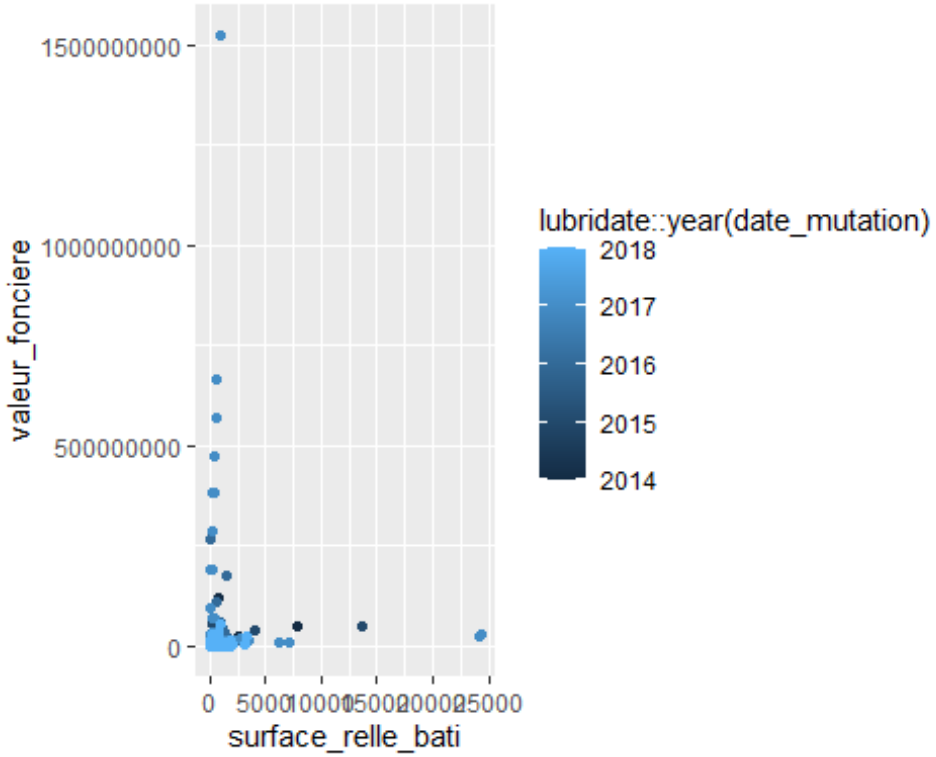
```
# Plot the actual values versus the model predictions
ggplot(data = db_test_res,
       mapping = aes(x = .pred, y = valeur_fonciere)) +
  geom_point(color = '#006EA1') +
  geom_abline(intercept = 0, slope = 1, color = 'orange') +
  labs(title = 'Régression linéaire sur le jeu de test',
       x = 'Valeurs prédites',
       y = 'Valeurs observées')
```



Please interpret, decide on possible changes and repeat steps if needed

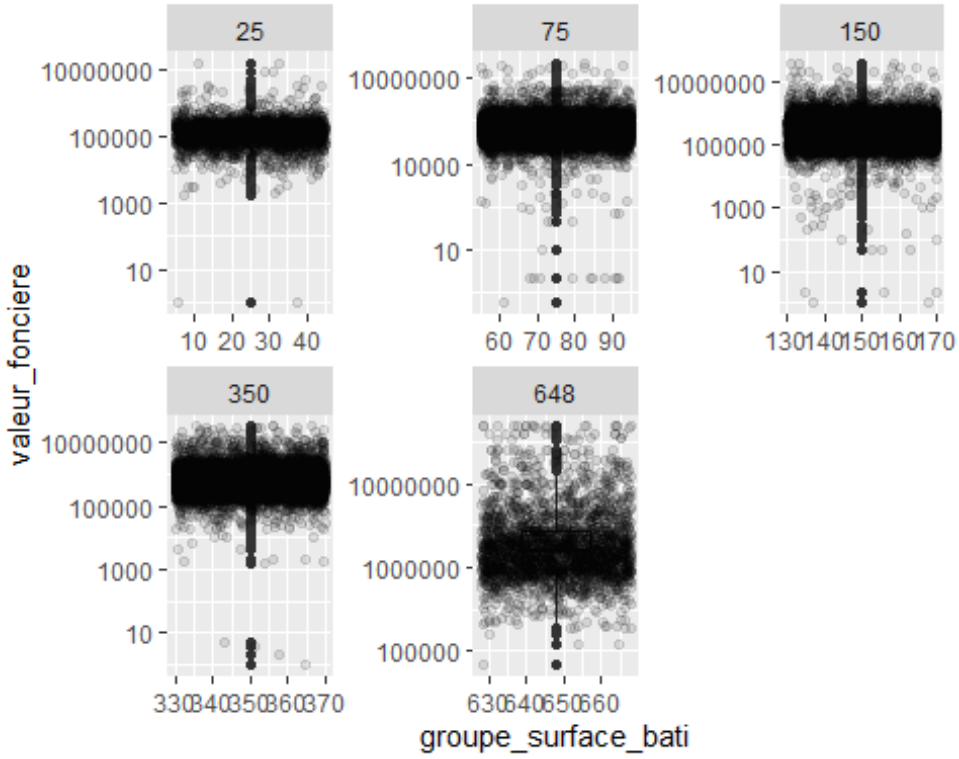
Grouping data after outlier exclusion

```
# Group data by plan ID and calculate resulting habitable surfaces
db_ex2 <- db_ex2 %>%
  select(-c("groupe_val_fonciere", "groupe_surface_bati")) %>%
  group_by(numero_plan) %>%
  mutate(valeur_fonciere = sum(valeur_fonciere)) %>%
  mutate(surface_relle_bati = sum(surface_relle_bati)) %>%
  mutate(nombre_pieces_principales = sum(nombre_pieces_principales)) %>%
  ungroup() %>%
  distinct()
db_ex2$valeur_fonciere %>% sort(decreasing = TRUE) %>% head(100)

# Check
db_ex2 %>%
  ggplot(aes(x = surface_relle_bati, y = valeur_fonciere, color =
lubridate::year(date_mutation))) +
  geom_point()
```

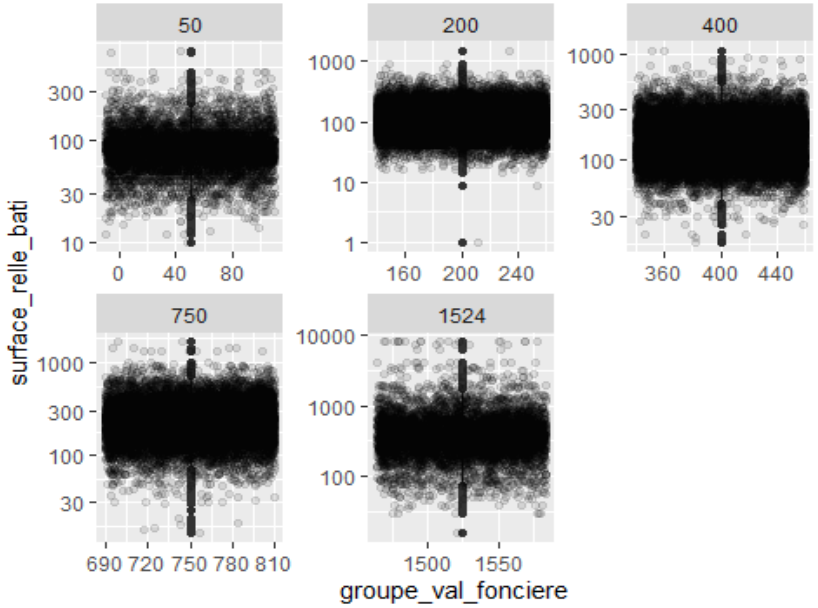



```
# Exceptions exist for high valuations, let's exclude it for the model
db_ex2 <- db_ex2 %>%
  filter(valeur_fonciere < 5.e+07) %>%
  filter(surface_relle_bati < 10000)

# Put the variables into groups
skimr::skim(db_ex2 %>% filter(valeur_fonciere > 1000000) %>% select(valeur_fonciere))

db_ex2 <- db_ex2 %>% mutate(
  "groupe_val_fonciere" = case_when(db_ex2$valeur_fonciere <= 100000 ~ 50,
                                    (db_ex2$valeur_fonciere <= 300000 &
db_ex2$valeur_fonciere > 100000) ~ 200,
                                    (db_ex2$valeur_fonciere <= 500000 &
db_ex2$valeur_fonciere > 300000) ~ 400,
                                    (db_ex2$valeur_fonciere <= 1000000 &
db_ex2$valeur_fonciere > 500000) ~ 750,
                                    TRUE ~ 1524))

skimr::skim(db_ex2 %>% filter(surface_relle_bati > 500) %>%
select(surface_relle_bati))

db_ex2 <- db_ex2 %>% mutate(
  "groupe_surface_bati" = case_when(db_ex2$surface_relle_bati <= 50 ~ 25,
                                    (db_ex2$surface_relle_bati <= 100 &
db_ex2$surface_relle_bati > 50) ~ 75,
                                    (db_ex2$surface_relle_bati <= 200 &
```

```
db_ex2$surface_relle_bati > 100) ~ 150,
                                    (db_ex2$surface_relle_bati <= 500 &
db_ex2$surface_relle_bati > 200) ~ 350,
                                    TRUE ~ 648))
# Visual check
options(scipen = 999)
db_ex2 %>%
  ggplot(aes(x = groupe_surface_bati, y = valeur_fonciere)) +
  geom_boxplot() +
  geom_jitter(alpha = 0.1) +
  facet_wrap(~ groupe_surface_bati, scales = "free") +
  scale_y_log10()
```




```
db_ex2 %>%
  ggplot(aes(x = groupe_val_fonciere, y = surface_relle_bati)) +
  geom_boxplot() +
  geom_jitter(alpha = 0.1) +
  facet_wrap(~ groupe_val_fonciere, scales = "free") +
  scale_y_log10()
```




```
# So the distribution of habitable surfaces seems independent from the real estate valuation, let's check it
```

Linear relationship modeling

```
# Covariance
cov(db_filter$groupe_surface_bati, db_filter$groupe_val_fonciere)
```

```
## [1] 8045.79
```

```
# slope "a"
cov(db_filter$groupe_surface_bati,
db_filter$groupe_val_fonciere)/var(db_filter$groupe_surface_bati)
```

```
## [1] 0.4548799
```

```
# Correlation coefficient
cor(db_filter$groupe_surface_bati, db_filter$groupe_val_fonciere)
```

```
## [1] 0.2697895
```

```
# Determination coefficient
cor(db_filter$groupe_surface_bati, db_filter$groupe_val_fonciere)**2
```

```
## [1] 0.07278637
```

Comment on the advantage of using linear fitting lines in the case studied

Rework the overall process by focusing on each above-created subgroups

THEME 3: ECONOMICAL INDEXES AND TIME SERIES MODELING

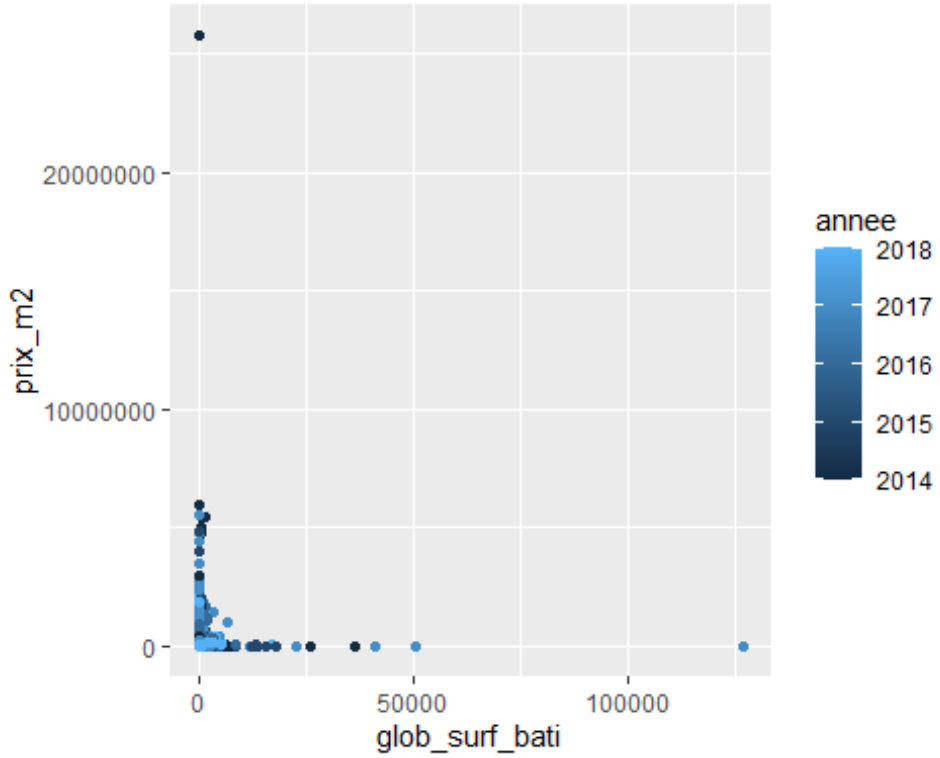
Evolution of averaged yearly real estate values by geographical area

Create a new matrix that focuses on Houses & Add a Price/Squared-Meter field

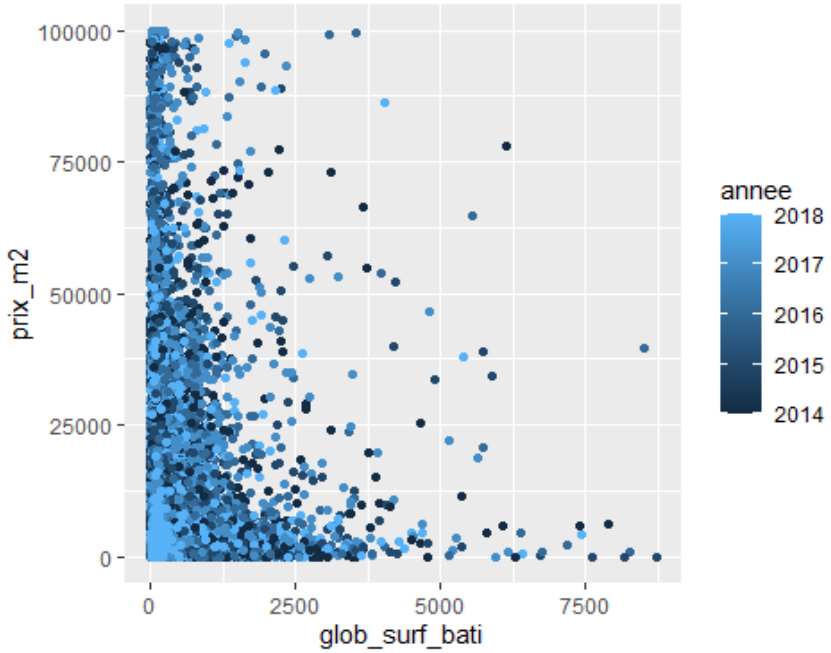
```
# Set up the matrix
db_yearly <- db_filtered %>%
  filter(type_local == "Maison") %>%
  group_by("annee" = year(date_mutation), code_departement, numero_plan) %>%
  mutate(glob_val_fonc = sum(valeur_fonciere)) %>%
  mutate(glob_surf_bati = sum(surface_relle_bati)) %>%
  mutate(nb_main_rooms = sum(nombre_pieces_principales)) %>%
  distinct() %>%
  ungroup()

# Add the Price-per-Squared-Meter field
options(scipen = 999)
db_yearly <- db_yearly %>%
  mutate(prix_m2 = round(glob_val_fonc/glob_surf_bati, 2))
```

Finalize data preparation

```
# Select the variables
tableau_annuel <- db_yearly %>%
  select(code_departement, annee, prix_m2, glob_surf_bati, glob_val_fonc,
nb_main_rooms) %>%
  arrange(code_departement)

# Plot it
options(scipen = 999)
tableau_annuel %>%
  ggplot(aes(x = glob_surf_bati, y = prix_m2, color = annee)) +
  geom_point()
```




```r
# Eliminate outliers
tableau_annuel <- tableau_annuel %>%
  filter(prix_m2 < 1.e+05) %>%
  filter(glob_surf_bati < 1.e+04)
tableau_annuel %>%
  ggplot(aes(x = glob_surf_bati, y = prix_m2, color = annee)) +
  geom_point()
```



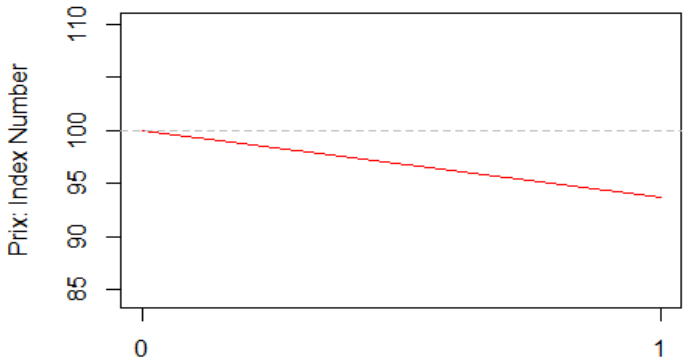

```
# Group by year, by geographical department
tab_ann_dept <- tableau_annuel %>%
  group_by(code_departement, annee) %>%
  mutate(prix_m2 = round(sum(prix_m2 * glob_surf_bati)/sum(glob_surf_bati),2)) %>%
  mutate(glob_surf_bati = sum(glob_surf_bati)) %>%
  select(-c(glob_val_fonc, nb_main_rooms)) %>%
  ungroup() %>%
  distinct()
# Slight adjustment on dept 47 (2018 missing)
tab_ann_dept <- rbind(tab_ann_dept, data.frame(code_departement = "47", annee = 2018,
prix_m2 = 0.0, glob_surf_bati = 0)) %>%
  arrange(code_departement, annee)
```

Calculate the Laspeyres, Paasche or Fisher Quantity/Price index

Laspeyres price index

```
library(IndexNumber)

# Laspeyres price index based on last year versus first year
annee_fin <- max(tab_ann_dept$annee)
annee_base <- min(tab_ann_dept$annee)

# Refine the matrix with only data showing useful information
prix <- rbind(
  tab_ann_dept %>% filter(annee == annee_base) %>% select(prix_annee_base = prix_m2)
%>% as.matrix() %>% transpose(),
  tab_ann_dept %>% filter(annee == annee_fin) %>% select(prix_annee_fin = prix_m2) %>%
as.matrix() %>% transpose())
poids <- tab_ann_dept %>% filter(annee == annee_base) %>% select("m²_annee_base" =
```

```
glob_surf_bati) %>% as.matrix() %>% transpose()
P_laspeyres <- laspeyres.index.number(prix, poids, "Prix", opt.plot=TRUE,
opt.summary=FALSE)
```

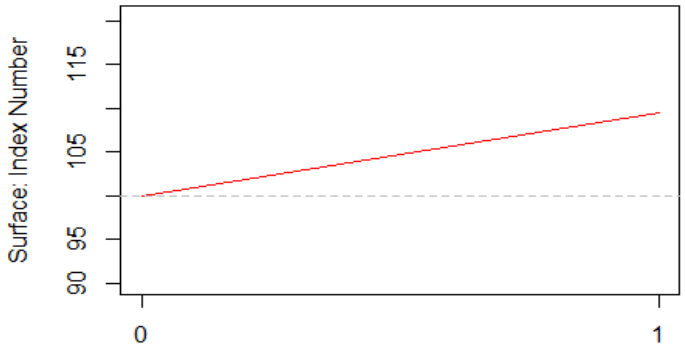



```
# Exhibit the result
paste('Indice de Laspeyres des prix :',round(P_laspeyres[2,ncol(P_laspeyres)],2),
"pour", annee_fin, "sur la base de", annee_base)

## [1] "Indice de Laspeyres des prix : 93.72 pour 2018 sur la base de 2014"
```

Proceed similarly to get the Laspeyres Quantity Index

Paasche quantity index

```
library(IndexNumber)
library(optimbase)

# Paasche quantity index based on last year versus first year
annee_fin <- max(tab_ann_dept$annee)
annee_base <- min(tab_ann_dept$annee)

# Refine the matrix with only data showing useful information
surface <- rbind(
  tab_ann_dept %>% filter(annee == annee_base) %>% select(surf_annee_base =
glob_surf_bati) %>% as.matrix() %>% transpose(),
  tab_ann_dept %>% filter(annee == annee_fin) %>% select(surf_annee_fin =
glob_surf_bati) %>% as.matrix() %>% transpose())
poids <- rbind(ones(1,tab_ann_dept %>% filter(annee == annee_base) %>%
nrow()),tab_ann_dept %>% filter(annee == annee_fin) %>% select("prix_annee_fin" =
prix_m2) %>% as.matrix() %>% transpose())
Q_paasche <- paasche.index.number(surface, poids, "Surface", opt.plot=TRUE,
opt.summary=FALSE)
```

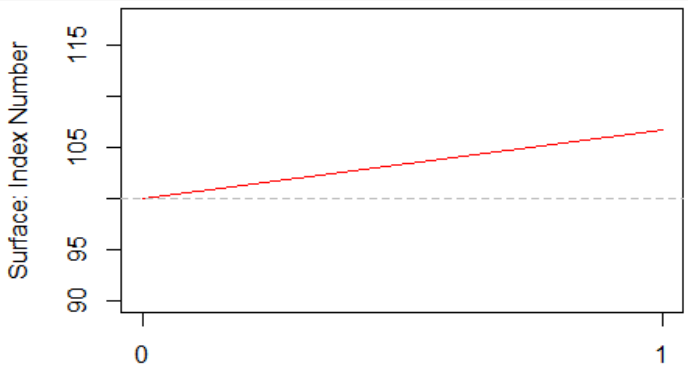



```
# Exhibit the result
paste('Indice de Paasche des quantités :',round(Q_paasche[2,ncol(Q_paasche)],2),
"pour", annee_fin, "sur la base de", annee_base)

## [1] "Indice de Paasche des quantités : 109.48 pour 2018 sur la base de 2014"
```

Proceed similarly to get the Paasche Price Index

Fisher Quantity Index

```
library(IndexNumber)

# Fisher quantity index based on last year versus first year
annee_fin <- max(tab_ann_dept$annee)
annee_base <- min(tab_ann_dept$annee)

# Refine the matrix with only data showing useful information
surface <- rbind(
  tab_ann_dept %>% filter(annee == annee_base) %>% select(surf_annee_base =
glob_surf_bati) %>% as.matrix() %>% transpose(),
  tab_ann_dept %>% filter(annee == annee_fin) %>% select(surf_annee_fin =
glob_surf_bati) %>% as.matrix() %>% transpose())
poids <- rbind(ones(1,97),tab_ann_dept %>% filter(annee == annee_fin) %>%
select("prix_annee_base" = prix_m2) %>% as.matrix() %>% transpose())
Q_fisher <- fisher.index.number(surface, poids, "Surface", opt.plot=TRUE,
opt.summary=FALSE)
```



```
# Exhibit the result
paste('Indice de Fisher des quantités :',round(Q_fisher[2,ncol(Q_fisher)],2), "pour",
annee_fin, "sur la base de", annee_base)
```

```
## [1] "Indice de Fisher des quantités : 106.8 pour 2018 sur la base de 2014"
```

```
# Verification of the relation between the 3 indices
Q_fisher_verif <- sqrt(Q_laspeyres * Q_paasche)
cbind(paste('Indice de Fisher des quantités =',(Q_fisher[,ncol(Q_fisher)])),
      paste('Racine_carrée de (Paasche des quantités x Laspeyres des quantités)
=',Q_fisher_verif[,ncol(Q_fisher_verif)]))[2,] %>%
  as.data.table() %>%
  kbl() %>%
  kable_paper(bootstrap_options = "striped", full_width = F, position = "left",
font_size = 12, fixed_thead = T)
```

Indice de Fisher des quantités = 106.803528017719

Racine_carrée de (Paasche des quantités x Laspeyres des quantités) = 106.803528017719

Proceed similarly to get the Fisher Price Index

Evolution of averaged monthly real estate values by geographical area

The objective is to determine possible seasonality effects, estimate trends and forecast values for the future

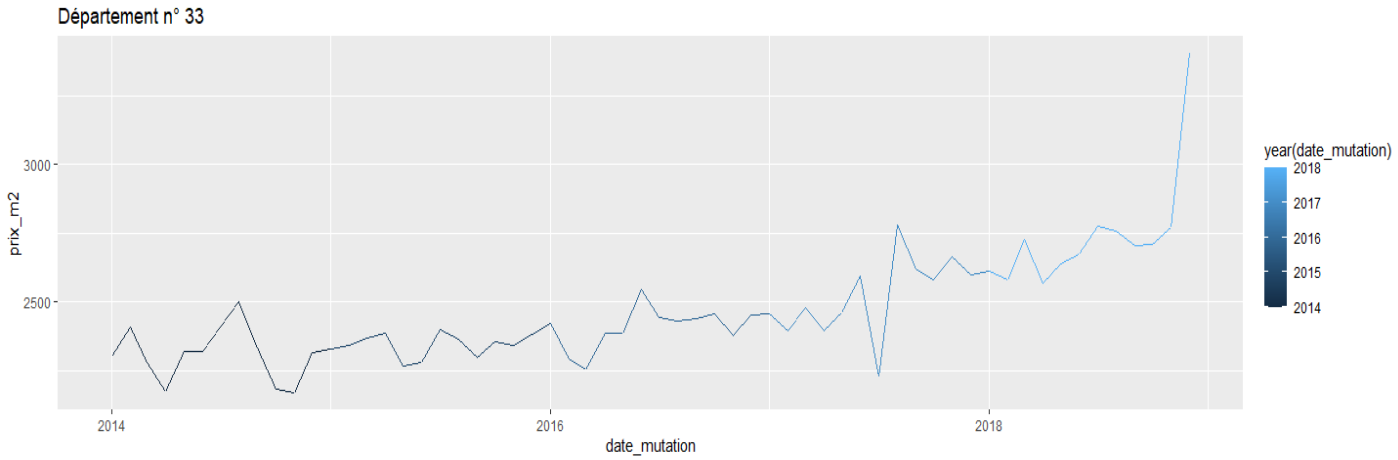
```
library(prophet)

# Prepare the only variables we need for this investigation: months, geographical
areas and (for instance) the averaged price per squared meter
options(scipen = 999)
db_proj <- db_filtered %>%
  filter(type_local == "Maison") %>%
  filter(surface_relle_bati < 10000) %>%
  filter(valeur_fonciere < 1.e+06) %>%
  select(-c(numero_disposition, nature_mutation, numero_voie, suffixe_numero,
type_voie,
            code_voie, voie, code_postal, commune, code_commune, prefixe_section,
            section, numero_volume, numero_plan, lot_1, lot_2, lot_3, lot_4, lot_5,
nombre_lots,
            code_type_local, type_local, nature_culture, nature_culture_speciale,
surface_terrain)) %>%
  mutate(date_mutation = floor_date(date_mutation, "month")) %>%
  group_by(date_mutation, code_departement) %>%
  mutate(valeur_fonciere = sum(valeur_fonciere)) %>%
  mutate(surface_relle_bati = sum(surface_relle_bati)) %>%
  mutate(nombre_pieces_principales = sum(nombre_pieces_principales))%>%
  mutate("prix_m2" = round(valeur_fonciere/surface_relle_bati, 2)) %>%
```

```
  ungroup() %>%
  distinct() %>%
  select(-c(valeur_fonciere, surface_relle_bati, nombre_pieces_principales)) %>%
  arrange(code_departement, date_mutation)
summary(db_proj)
```

```
##  date_mutation        valeur_fonciere      code_departement surface_relle_bati
##  Min.   :2014-01-01   Min.   :         0   01     :  60     Min.   :    37
##  1st Qu.:2015-03-01   1st Qu.:  28162164   02     :  60     1st Qu.: 20336
##  Median :2016-06-01   Median :  61353186   03     :  60     Median : 36918
##  Mean   :2016-06-07   Mean   :  87607275   04     :  60     Mean   : 44799
##  3rd Qu.:2017-09-01   3rd Qu.: 117818930   05     :  60     3rd Qu.: 59792
##  Max.   :2018-12-01   Max.   :1064974130   06     :  60     Max.   :478287
##                                            (Other):5396
##  nombre_pieces_principales    prix_m2
##  Min.   :    2.0           Min.   :    0
##  1st Qu.:  820.8           1st Qu.: 1332
##  Median : 1495.5           Median : 1634
##  Mean   : 1845.1           Mean   : 1959
##  3rd Qu.: 2449.2           3rd Qu.: 2207
##  Max.   :24895.0           Max.   :16347
```

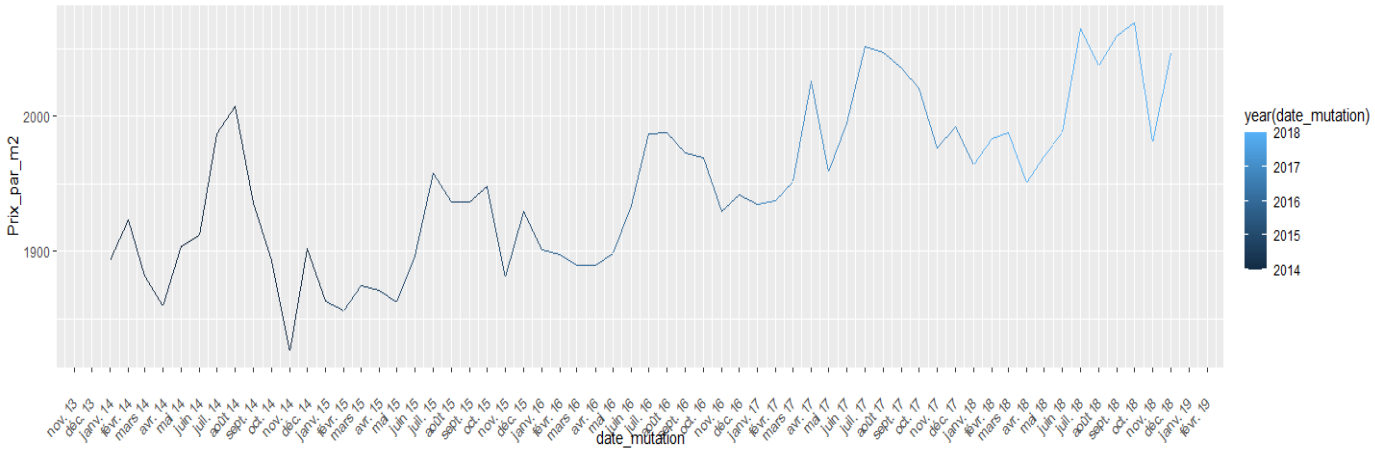
```
# Plot it
db_proj %>%
  gather(key = "Variable", value = "Valeur", -c(code_departement, date_mutation)) %>%
  ggplot(aes(x = date_mutation, y = Valeur, color = year(date_mutation))) +
  geom_line() +
  facet_wrap(~code_departement, scales = "free")
```

```
# Plot each department individually
plot_list <- map(.x = unique(db_proj$code_departement), ~ db_proj %>%
              filter(code_departement == .x) %>%
              ggplot(aes(x = date_mutation, y = prix_m2, color = year(date_mutation)))
+
              geom_line() +
              ggtitle(paste("Département n°",.x)))
# example with the Gironde:
plot_list[[33]]
```



Let's merge the information all together to represent the overall tendency within France

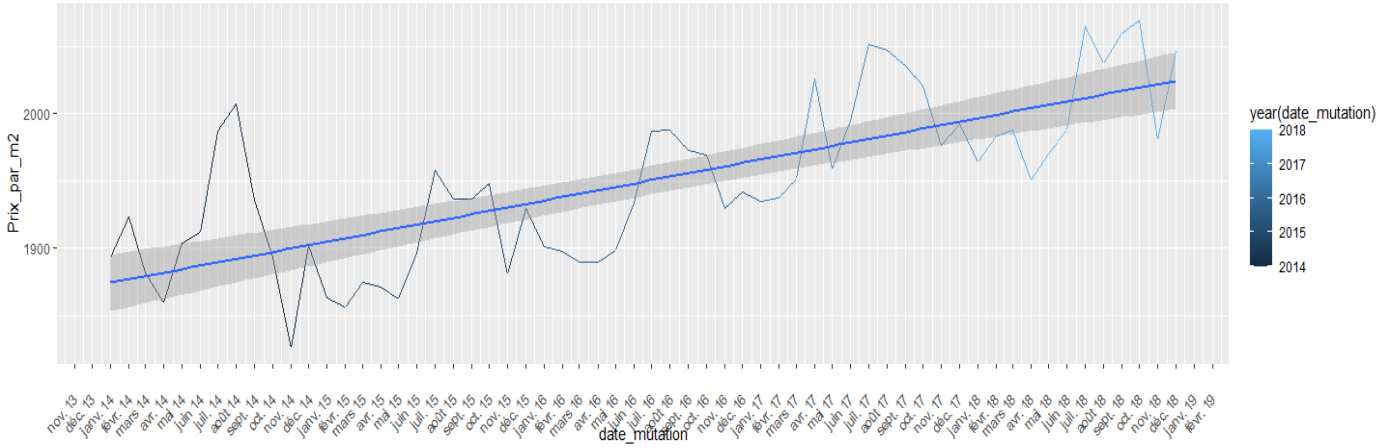
```
options(scipen = 999)
db_proj <- db_filtered %>%
  filter(type_local == "Maison") %>%
  filter(surface_relle_bati < 10000) %>%
  filter(valeur_fonciere < 1.e+06) %>%
  select(-c(numero_disposition, nature_mutation, numero_voie, suffixe_numero,
type_voie,
            code_voie, voie, code_postal, commune, code_commune, prefixe_section,
code_departement,
            section, numero_volume, numero_plan, lot_1, lot_2, lot_3, lot_4, lot_5,
nombre_lots,
            code_type_local, type_local, nature_culture, nature_culture_speciale,
surface_terrain)) %>%
  mutate(date_mutation = floor_date(date_mutation, "month")) %>%
  group_by(date_mutation) %>%
  mutate(valeur_fonciere = sum(valeur_fonciere)) %>%
  mutate(surface_relle_bati = sum(surface_relle_bati)) %>%
  mutate(nombre_pieces_principales = sum(nombre_pieces_principales))%>%
  mutate("prix_m2" = round(valeur_fonciere/surface_relle_bati, 2)) %>%
  ungroup() %>%
  distinct() %>%
  select(-c(valeur_fonciere, surface_relle_bati, nombre_pieces_principales)) %>%
  arrange(date_mutation) %>%
  gather(key = "Variable", value = "Prix_par_m2", -date_mutation)

# Plot it
db_proj %>%
  ggplot(aes(x = date_mutation, y = Prix_par_m2, color = year(date_mutation))) +
  geom_line() +
  scale_x_date(date_labels="%b %y", date_breaks = "1 month") +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=1))
```



Add statistical information to the data

Add a linear trend

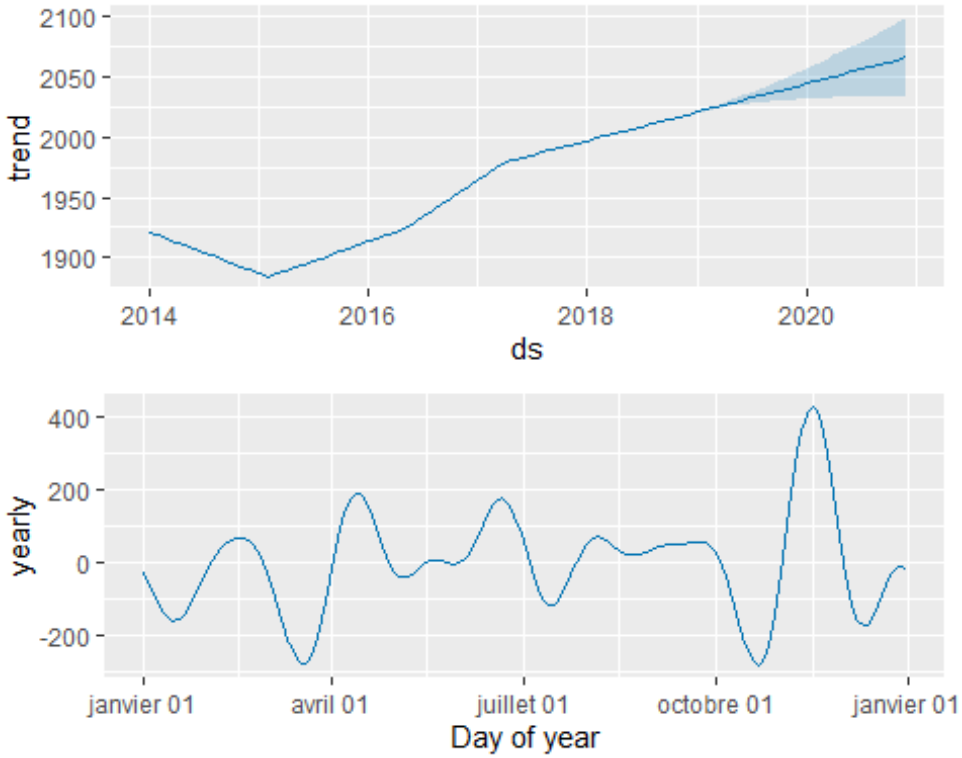
```
db_proj %>%
  ggplot(aes(x = date_mutation, y = Prix_par_m2, color = year(date_mutation))) +
  geom_line() +
  scale_x_date(date_labels="%b %y", date_breaks = "1 month") +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=1)) +
  geom_smooth(method = "lm")
```



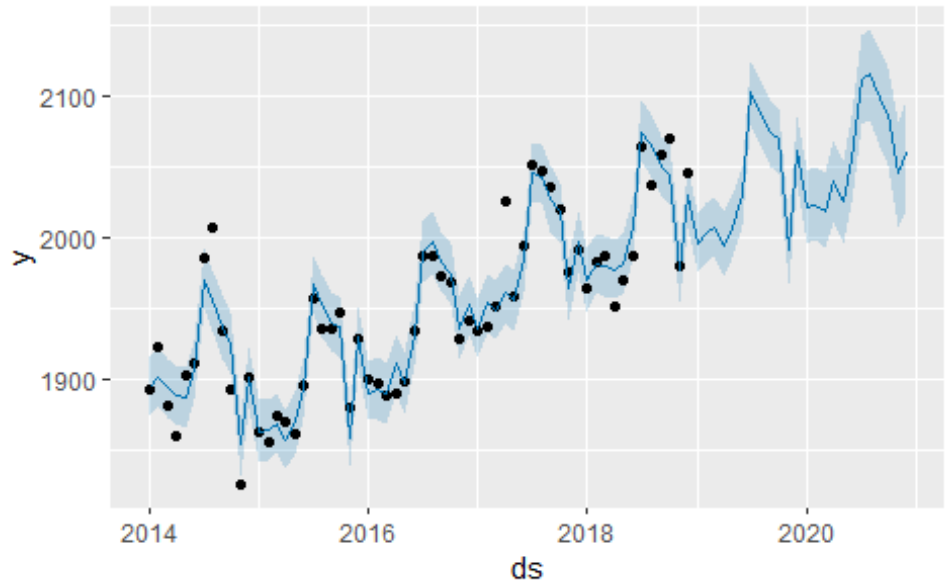
Project to next time intervals

```
m <- prophet(df)

## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override
this.

## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override
this.

futur <- make_future_dataframe(m, periods = 24, freq = "month", include_history =
TRUE)
```

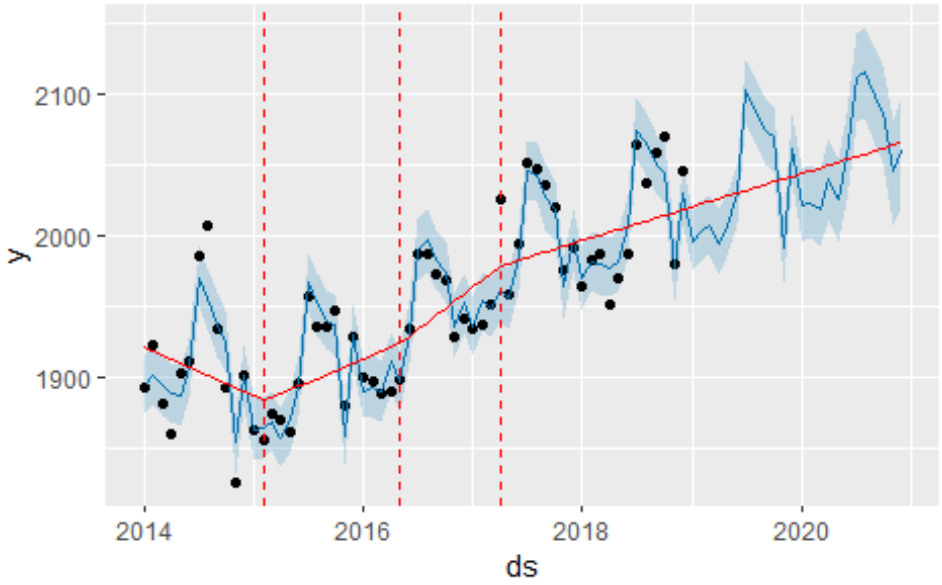
```
prevision <- predict(m, futur)
prevision

# Separate trend prevision from seasonality
prophet_plot_components(m, prevision)
```




```
# View projections to next 2 years
plot(m, prevision)
```



```
# Add to the view possible change points if any
plot(m, prevision) +
  add_changepoints_to_plot(m)
```

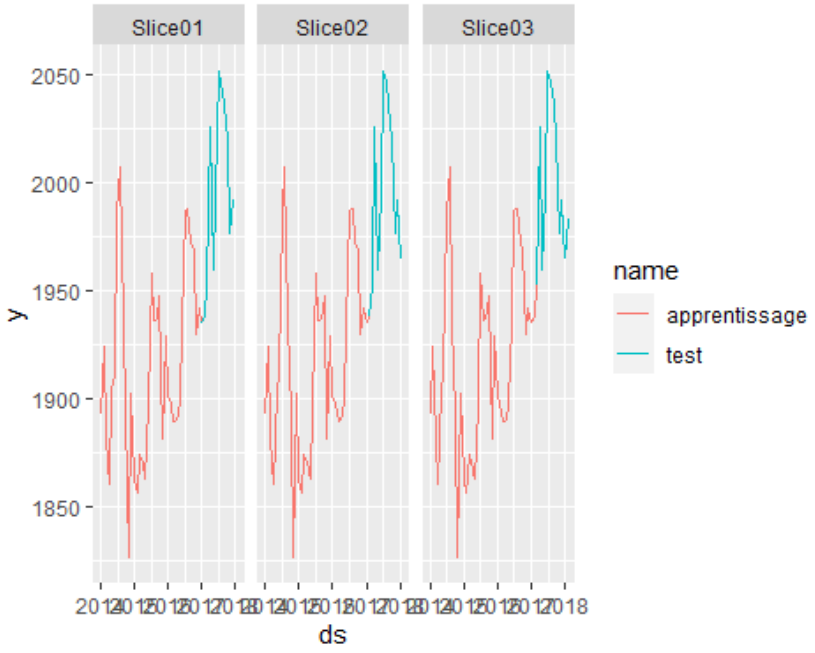



```
# Use plotly for advanced view if necessary
pred_plot <- plot(m, prevision) %>% plotly::ggplotly()
```

Investigate to link those change points to economical events that occurred over past years

Tune the model to find a best adjustment

```
# Define the reference and the forecast periods
rolling_origin(df, initial = 36, assess = 12) %>%
  mutate(apprentissage = map(splits, analysis),
         test = map(splits, assessment)) %>%
  select(id, apprentissage, test) %>%
  pivot_longer(-id) %>%
  unnest(value) %>%
  #count(id)
  filter(id %in% c("Slice01", "Slice02", "Slice03")) %>%
  ggplot(aes(x = ds, y = y, color = name, group = id)) +
  geom_line() +
  facet_wrap(~id, scales = "fixed")
```



Define a function to run both options: additive or multiplicative

```
tune_prophet <- function(splits){
  donnees_apprentissage <- analysis(splits)
  donnees_test <- assessment(splits)
  m1 <- prophet(df = donnees_apprentissage, seasonality.mode = "additive")
  m2 <- prophet(df = donnees_apprentissage, seasonality.mode = "multiplicative")
  futur1 <- make_future_dataframe(m1, periods = nrow(donnees_test), freq = "month",
include_history = FALSE)
  futur2 <- make_future_dataframe(m2, periods = nrow(donnees_test), freq = "month",
include_history = FALSE)
  bind_rows(
  predict(m1, futur1) %>% select(ds, yhat) %>% mutate(type = "additive"),
  predict(m2, futur2) %>% select(ds, yhat) %>% mutate(type = "multiplicative")
  ) %>%
    left_join(donnees_test, by = "ds")
}

st_tune <- serie_temp %>% mutate(res = map(splits, tune_prophet))
```

Identify the best model among all existing ones

```
st_tune %>%
  select(id, res) %>%
  unnest(res) %>%
  group_by(id, type) %>%
  arrange(ds) %>%
  mutate(prevision = paste0("prévision_n°", row_number())) %>%
  ungroup() %>%
  select(prevision, type, ds, yhat, y) %>%
  group_by(prevision, type) %>%
```

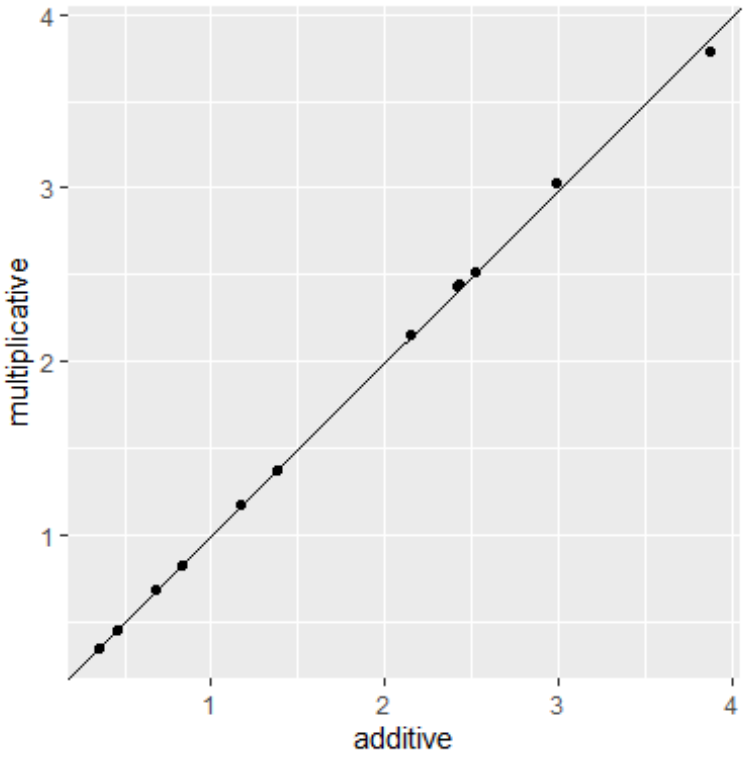
```
  rmse(truth = y, estimate = yhat) %>%
  ungroup() %>%
  group_by(prevision) %>%
  slice_min(.estimate) %>%
  ungroup() %>%
  count(type)
```

```
## 1 additive           7
## 2 multiplicative     5
```

What conclusion to take from it?

Define how the forecast is effective as compared to the naive forecast i.e. taking N-1 value for period N

```
st_tune %>%
  select(id, res) %>%
  unnest(res) %>%
  group_by(id, type) %>%
  arrange(ds) %>%
  mutate(prevision = paste0("prévision_n°", row_number())) %>%
  ungroup() %>%
  select(prevision, type, ds, yhat, y) %>%
  left_join(df %>%
              mutate(naive = lag(y, n = 1, order_by = ds)) %>%
              drop_na() %>%
              select(ds, naive),
            by = "ds") %>%
  group_by(prevision, type) %>%
  summarise(mase = mean(abs(yhat - y)) / mean(abs(naive - y))) %>%
  ungroup() %>%
  pivot_wider(names_from = type, values_from = mase) %>%
  ggplot(aes(x = additive, y = multiplicative)) +
  geom_abline() +
  geom_point() +
  coord_obs_pred()
```



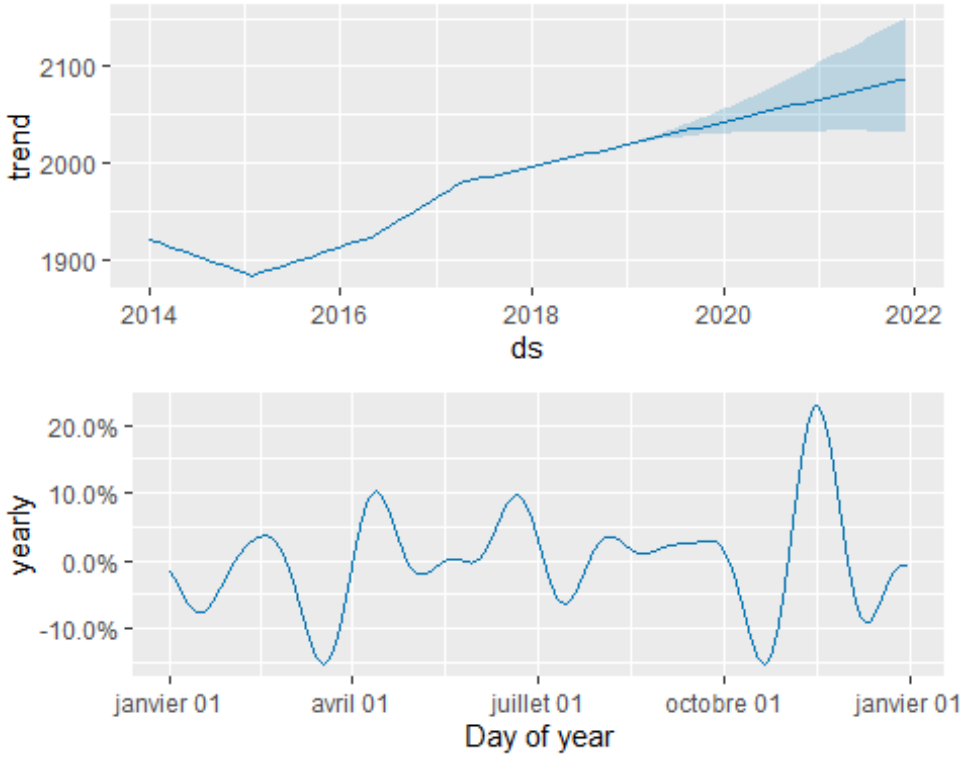

```
st_tune %>%
  select(id, res) %>%
  unnest(res) %>%
  group_by(id, type) %>%
  arrange(ds) %>%
  mutate(prevision = paste0("prévision_n°", row_number())) %>%
  ungroup() %>%
  select(prevision, type, ds, yhat, y) %>%
  left_join(df %>%
              mutate(naive = lag(y, n = 1, order_by = ds)) %>%
              drop_na() %>%
              select(ds, naive),
            by = "ds") %>%
  group_by(prevision, type) %>%
  summarise(mase = mean(abs(yhat - y)) / mean(abs(naive - y))) %>%
  ungroup() %>%
  group_by(type) %>%
  summarise(median = median(mase))

## 1 additive         1.77
## 2 multiplicative   1.77

st_tune %>%
  select(id, res) %>%
  unnest(res) %>%
  group_by(id, type) %>%
```

```
  arrange(ds) %>%
  mutate(prevision = paste0("prévision_n°", row_number())) %>%
  ungroup() %>%
  select(id, prevision, type, ds, yhat, y) %>%
  group_by(id, type) %>%
  mutate(naive_date = min(ds) - months(1)) %>%
  ungroup() %>%
  left_join(df %>%
              mutate(naive = lag(y, n = 1, order_by = ds)) %>%
              drop_na() %>%
              select(ds, naive),
            by = c("naive_date" = "ds")) %>%
  group_by(prevision, type) %>%
  summarise(mase = mean(abs(yhat - y)) / mean(abs(naive - y))) %>%
  ungroup() %>%
  group_by(type) %>%
  summarise(median = median(mase))
```

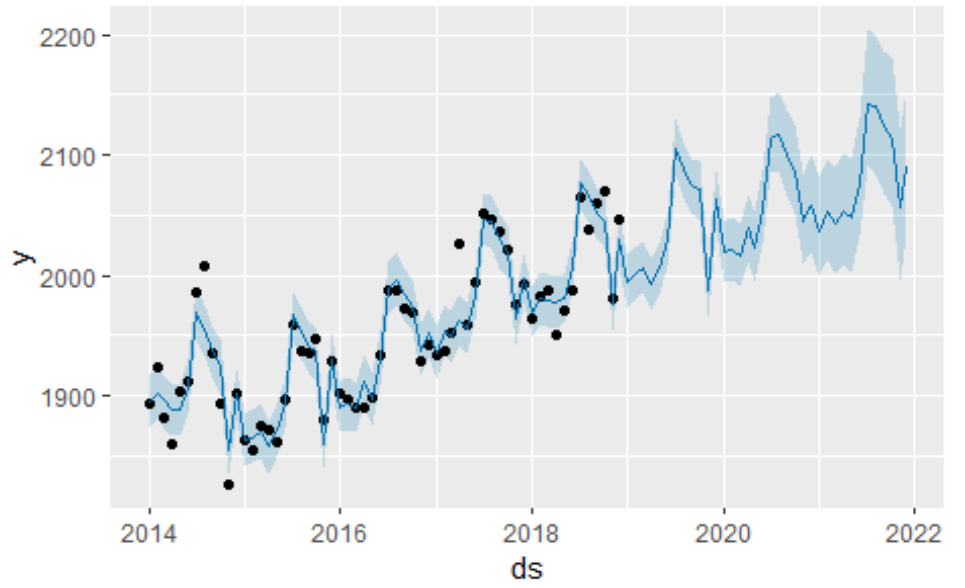
```
## 1 additive        0.611
## 2 multiplicative  0.605
```

Finalizing the best applicable model

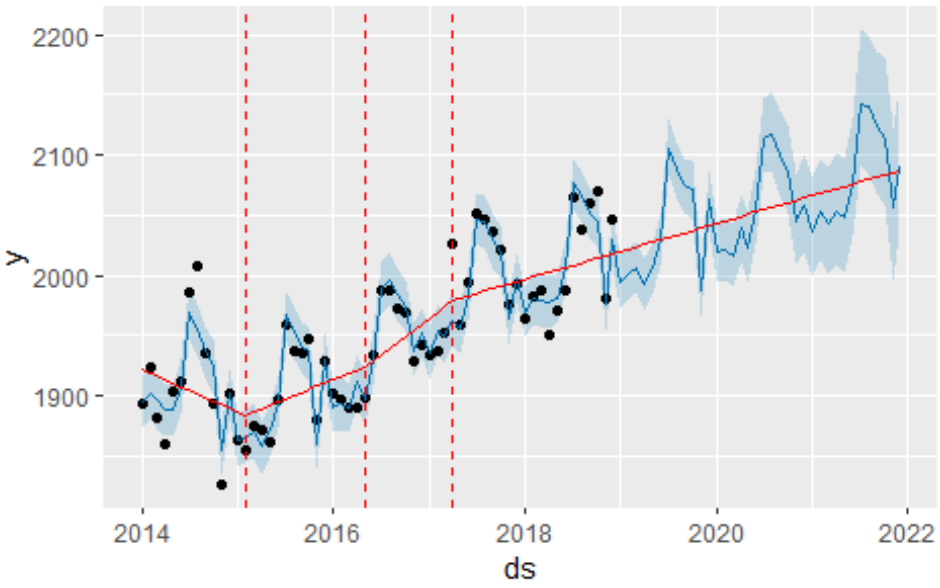
```
m <- prophet(df, seasonality.mode = "multiplicative")

futur <- make_future_dataframe(m, periods = 36, freq = "month", include_history =
TRUE)
prevision <- predict(m, futur)

# Separate trend prevision from seasonality
prophet_plot_components(m, prevision)
```




```
# View projections to next 2 years
plot(m, prevision)
```




```
# Add to the view possible change points if any
plot(m, prevision) +
  add_changepoints_to_plot(m)
```



Take the model to compare to real values as in 2021 => measure the impact of COVID-19 emergence

Get the 2021 data and adjust it

```
db_2021 <- read.csv2(file("https://static.data.gouv.fr/resources/demandes-de-valeurs-foncieres/20211020-111744/valeursfoncieres-2021-s1.txt", encoding="UTF-8"),
                     sep ="|", header = TRUE)
# First view
glimpse(db_2021)
```

Select and reformat the fields to accord the data to the previous database

CONCLUSION: COVID-19 CREATED A SIGNIFICANT INCREASE IN REAL ESTATE VALUATION IN FRANCE

Select a "preferred" department then perform a similar analysis on the volume of transactions and conclude on any impact due to the COVID-19