Lab 1: Vizualization

This plot is a bit complicated, in particular the confidence intervals. A first thing could be to plot the mean over year.

The different steps are:

- stack the dataframes with `rbind`
- filter a subtable without extreme sales and without the row having missing values in `valeur_fonciere`
- create a colunm year, for this you can use the package `lubridate`, but you can also do it alone with a package extracting the first four characters of a string.
- then use the function `group_by`, to group by year the observations and compute the mean and the standard deviation
- then plot the average over years
- the 95% confidence intervals are obtained from the standard deviation

Again, don't worry if you don't succeed to finish everything or understand all the lines. Please note your questions for the next sessions.

```
knitr::opts_chunk$set(echo = TRUE)

# We could have done a loop to avoid copy paste, but let's keep it like this for the first class
# Be careful to have all the data in a folder named data
df_2016 <- read.csv("./data/2016.csv")
df_2017 <- read.csv("./data/2017.csv")
df_2018 <- read.csv("./data/2018.csv")
df_2019 <- read.csv("./data/2019.csv")
df_2020 <- read.csv("./data/2020.csv")
df_2021 <- read.csv("./data/2021.csv")
immo <- rbind(df_2016, df_2017, df_2018, df_2019, df_2020,df_2021)

# at the beginning of your notebook
VALEUR_FONCIERE_MAX = 2000000
VALEUR_FONCIERE_MIN = 50000

# Libraries used
library(ggplot2)
library(dplyr) # group_by
```

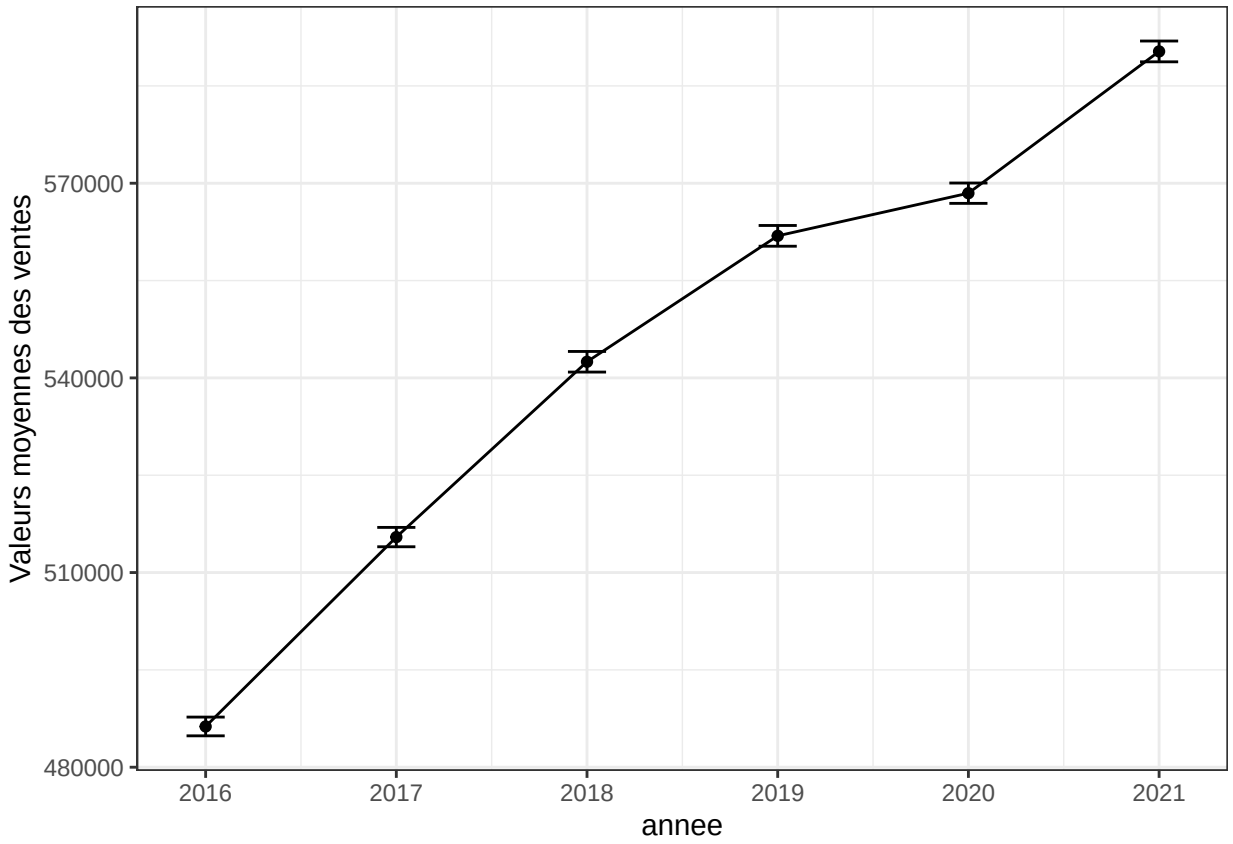
```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(lubridate) # function to treat automatically year
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
# create a column year
immo$annee <- year(immo$date_mutation)

# take subtable
immo <- immo[immo$valeur_fonciere < VALEUR_FONCIERE_MAX,]
immo <- immo[immo$valeur_fonciere > VALEUR_FONCIERE_MIN,]

# keep only observations for which valeur fonciere is observed
immo <- immo[!is.na(immo$valeur_fonciere),] # ! indicates "opposite to" so that we keep only rows for w

summary <- immo  %>%
  group_by(annee) %>%
  summarise(mean = mean(valeur_fonciere),
            sd = sd(valeur_fonciere))


summary$se = summary$sd / sqrt(nrow(immo))

ggplot(summary, aes(x = annee, y = mean)) +
  geom_line() +
  geom_point()+
  geom_errorbar(aes(ymin=mean-1.96*se, ymax=mean+1.96*se), width=.2,
                position=position_dodge(0.05)) +
  theme_bw() +
  ylab("Valeurs moyennes des ventes")
```

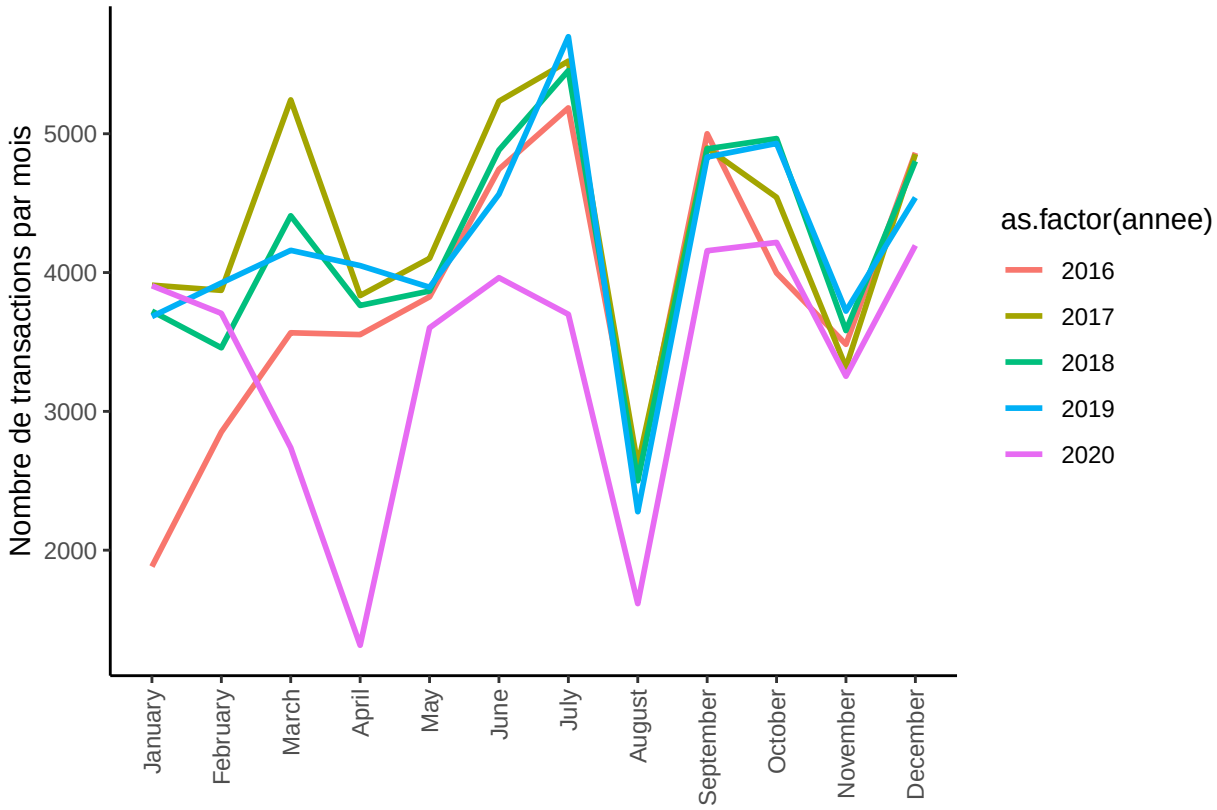


Then, we can observe the number of transactions per month over year to understand what happened in 2020.

```
# function from package lubridate
immo$month <- month(immo$date_mutation, label=TRUE, abbr=FALSE)
```

```
# 2021 is only first semester so we remove it from the plot
immo[immo$annee != 2021,] %>%
  group_by(annee, month) %>%
  summarise(count = n()) %>%
  ggplot(aes(x = month, y = count, color = as.factor(annee), group = as.factor(annee))) +
  geom_line(size = 1) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  ylab("Nombre de transactions par mois") +
  xlab("")
```

```
## `summarise()` has grouped output by 'annee'. You can override using the `.groups` argument.
```



You can further improve plots, customizing colors, title, font size, and so on…!