

Introduction à R - partie 5 (analyse exploratoire)

Aurélie Labbe

Analyse exploratoire

Nous allons ici vous donner un exemple de code **R** qui permet de faire l'analyse exploratoire d'un jeu de données. Je vous rappelle que l'étape d'exploration est cruciale qu'elle que soit l'analyse que vous effectuez et est trop souvent négligée.

Les données que nous allons analyser proviennent du site Kaggle et contiennent les prix de vente de 21 613 maisons vendues dans l'État de Washington entre mai 2014 et mai 2015 dans le comté de King, qui inclut la ville de Seattle. Ce jeu de données contient aussi 19 variables mesurant différentes caractéristiques des habitations. Ces variables sont détaillées ci-dessous:

- **id**: numéro d'identifiant de la propriété
- **date**: date de vente de la propriété
- **price**: prix de vente de la propriété (variable d'intérêt)
- **bedrooms**: nombre de chambres
- **bathrooms**: nombre de salles de bain
- **sqft_living**: superficie en pieds carrés (intérieur)
- **sqft_lot**: superficie en pieds carrés (lot au complet)
- **floors**: nombre d'étages
- **waterfront**: vue sur la mer (1=où, 0=non)
- **view**: nombre de vues de la propriété dans l'agence immobilière (note: aucune explication n'est donnée sur la signification de cette variable)
- **condition**: condition de la propriété (score de 1 à 5) en fonction de son âge et de son état (1=mauvaise, 2=acceptable, 3=moyen, 4=bon, 5=très bon). Ce score est calculé selon le système de notation du comté de King.
- **grade**: score global selon la qualité des rénovations et des matériaux de construction (score de 1 à 13 - un score élevé correspond à des matériaux utilisés de meilleure qualité). Ce score est calculé selon le système de notation du comté de King.
- **sqft_above**: superficie de la propriété excluant le sous-sol
- **sqft_basement**: superficie du sous-sol

- yr_built: année de construction
- yr_renovated: année des dernières rénovations
- zip: code postal
- lat: latitude
- long: longitude

Voici le code qui permet de lire les données:

```
# install.packages("MAVE")
# Extraire directement les données Kaggle d'un paquet
data(kc_house_data, package = "MAVE")
maisons <- kc_house_data # renommer pour faciliter l'analyse
str(maisons)
```

```
tibble [21,613 x 21] (S3: tbl_df/tbl/data.frame)
 $ id          : chr [1:21613] "7129300520" "6414100192" "5631500400" "2487200875" ...
 $ date        : POSIXct[1:21613], format: "2014-10-13" "2014-12-09" ...
 $ price       : num [1:21613] 221900 538000 180000 604000 510000 ...
 $ bedrooms    : int [1:21613] 3 3 2 4 3 4 3 3 3 3 ...
 $ bathrooms   : num [1:21613] 1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
 $ sqft_living : int [1:21613] 1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
 $ sqft_lot    : int [1:21613] 5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
 $ floors      : num [1:21613] 1 2 1 1 1 1 2 1 1 2 ...
 $ waterfront  : int [1:21613] 0 0 0 0 0 0 0 0 0 0 ...
 $ view        : int [1:21613] 0 0 0 0 0 0 0 0 0 0 ...
 $ condition   : int [1:21613] 3 3 3 5 3 3 3 3 3 3 ...
 $ grade       : int [1:21613] 7 7 6 7 8 11 7 7 7 7 ...
 $ sqft_above  : int [1:21613] 1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
 $ sqft_basement: int [1:21613] 0 400 0 910 0 1530 0 0 730 0 ...
 $ yr_built    : int [1:21613] 1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
 $ yr_renovated : int [1:21613] 0 1991 0 0 0 0 0 0 0 0 ...
 $ zipcode     : int [1:21613] 98178 98125 98028 98136 98074 98053 98003 98198 98146 98038
 $ lat         : num [1:21613] 47.5 47.7 47.7 47.5 47.6 ...
 $ long        : num [1:21613] -122 -122 -122 -122 -122 ...
 $ sqft_living15: int [1:21613] 1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
 $ sqft_lot15  : int [1:21613] 5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
 - attr(*, "spec")=
 .. cols(
 ..   id = col_character(),
 ..   date = col_datetime(format = ""),
 ..   price = col_double(),
 ..   bedrooms = col_integer(),
```

```

..   bathrooms = col_double(),
..   sqft_living = col_integer(),
..   sqft_lot = col_integer(),
..   floors = col_double(),
..   waterfront = col_integer(),
..   view = col_integer(),
..   condition = col_integer(),
..   grade = col_integer(),
..   sqft_above = col_integer(),
..   sqft_basement = col_integer(),
..   yr_built = col_integer(),
..   yr_renovated = col_integer(),
..   zipcode = col_integer(),
..   lat = col_double(),
..   long = col_double(),
..   sqft_living15 = col_integer(),
..   sqft_lot15 = col_integer()
.. )

```

Exploration des données: une variable à la fois

Nous pouvons commencer par faire des résumés numériques de toutes les variables pour identifier les valeurs manquantes, les valeurs aberrantes (en regardant le min et max) et évaluer rapidement leur distribution en regardant les quartiles. Ceci peut se faire à l'aide de la fonction `summary`, comme dans l'exemple ci-dessous:

```
summary(maisons$price)
```

Exercice 0.1. À l'aide de la fonction `summary`, évaluez le profile des variables quantitatives. Quels sont les éléments qui retiennent particulièrement votre attention? Y-a-t-il des données aberrantes ou extrêmes à considérer plus attentivement?

Des histogrammes et des boîtes à moustaches des variables quantitatives permettent de visualiser la distribution des données. Ces graphiques sont un bon moyen d'évaluer la normalité et symmétrie des variables.

```

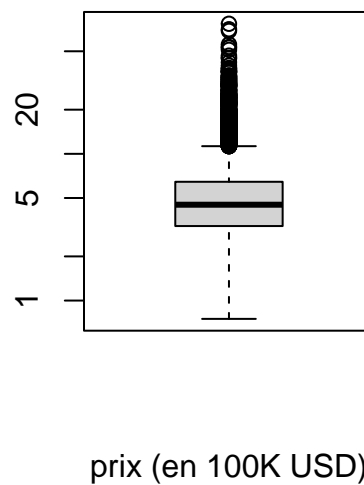
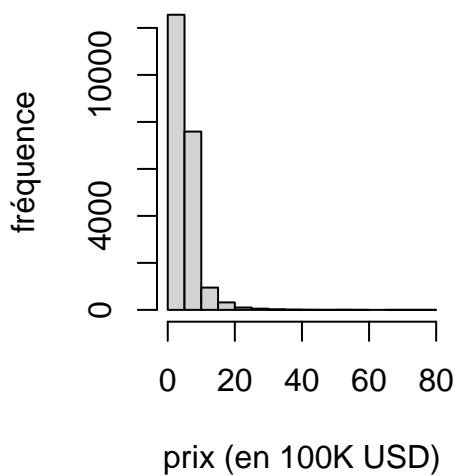
# Variable "price"
par(mfrow = c(1, 2))
prix <- maisons$price # prix
hist(prix / 1e5, # modifier échelle pour clarté des étiquettes
     main = "",

```

```

ylab = "fréquence",
xlab = "prix (en 100K USD)")
# Boîte à moustache - pas adéquat vu le volume
boxplot(prix / 1e5,
        horizontal = FALSE,
        main = "", # libellé du titre
        xlab = "prix (en 100K USD)",
        log = "y") #libellé de l'axe des x

```

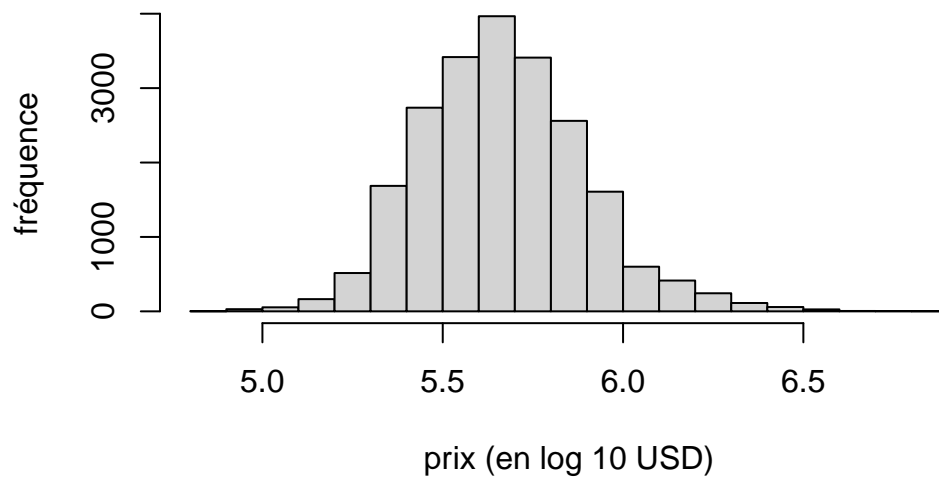


Nous voyons par exemple que la distribution de la variable `prix` est très asymétrique vers la droite. Si nous voulons par la suite utiliser un modèle de régression linéaire pour modéliser le prix des maisons, la transformation `log` sera très probablement nécessaire pour éviter l'impact des extrêmes.

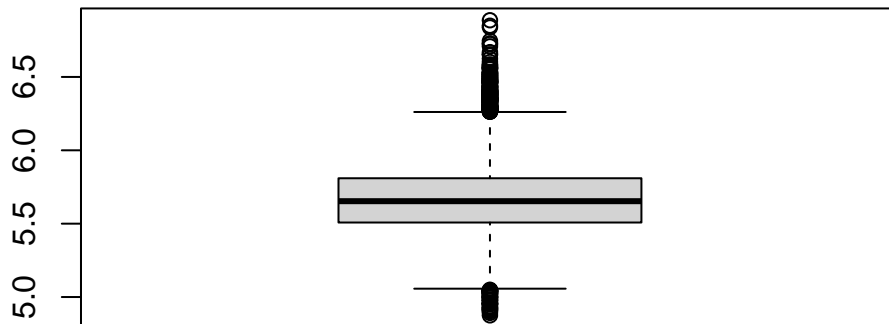
```

hist(log(prix, base = 10),
     main = "",
     ylab = "fréquence",
     xlab = "prix (en log 10 USD)")

```



```
boxplot(log(prix, base = 10),  
        horizontal = FALSE,  
        # log = "y" # mettre un axe logarithmique avec données brutes  
        main = "", # libellé du titre  
        xlab = "prix (en log(10) USD)")
```



prix (en log(10) USD)

Comprendre et explorer les données implique l'exploration un peu plus approfondie de certaines variables afin d'avoir une idée précise du profil de nos données. Par exemple, vous devriez pouvoir répondre à des questions de base comme par exemple de connaître le nombre de maisons par catégorie de prix. Voici un exemple de code qui pourrait répondre à ces questions

```
# Pourcentage des observations qui excèdent 200K, 300K, 400K
apply(c(2e5, 3e5, 4e5), function(val){mean(prix > val)})
```

```
[1] 0.9610882 0.7885532 0.5910332
```

La distribution des variables numériques catégorielles, telles que le nombre de chambres, salle de bain, etc. peut être calculée à l'aide de la fonction `table`:

```
table(maisons$bedrooms)
```

0	1	2	3	4	5	6	7	8	9	10	11	33
13	199	2760	9824	6882	1601	272	38	13	6	3	1	1

Exercice 0.2. À l'aide de la fonction `table`, évaluez le profil des variables numériques catégorielles. Quels sont les éléments qui retiennent particulièrement votre attention?

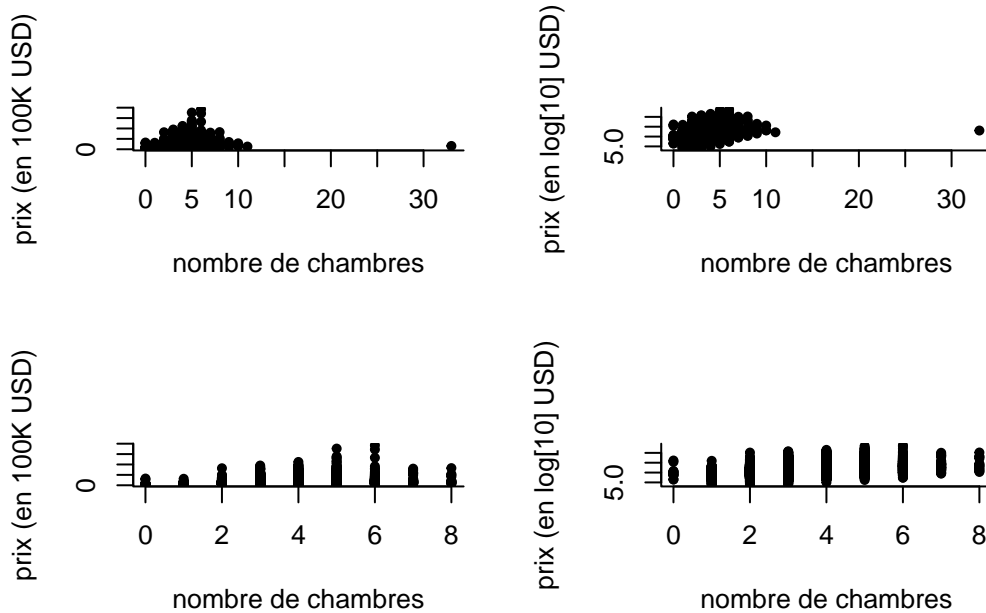
Exercice 0.3. Combien y a-t-il de codes postaux différents dans la base de données?

Exercice 0.4. Faites toujours quelques petites vérifications de base sur vos données (soyez curieux et créatifs!) Par exemple, est-ce que la somme des superficies à l'étage et au sous-sol est bien égale à la superficie totale ? Vous pouvez vérifier aussi par exemple que lorsque `sqft_living > sqft_lot`, c'est que la propriété a plusieurs étages.

Exploration des données: croiser les variables

Lorsque l'on utilise un modèle linéaire pour prédire une variable d'intérêt Y , il est important de vérifier que la relation entre Y et chacun des prédicteurs X est belle et bien linéaire. Si un modèle linéaire est ajusté sur des données montrant une relation quadratique ou log-linéaire par exemple, la qualité de prédiction peut être grandement affectée. Il est donc important de regarder au préalable quel type de relation (linéaire ou nonlinéaire) semble exister entre chacun des prédicteurs et la variable d'intérêt. Voici maintenant une série de commandes qui vont vous permettre d'évaluer la relation entre le prix des maisons et leurs caractéristiques. N'hésitez pas à enlever les valeurs extrêmes de vos graphiques (pas du jeu de données !) pour avoir une meilleure idée des relations. Vous pouvez aussi regarder la relation avec le logarithme des variables, pour évaluer quelle transformation donne la meilleure relation linéaire entre les variables.

```
# Bedrooms
par(mfrow = c(2, 2), pch = 20, bty = "l")
plot(I(price / 1e5) ~ bedrooms,
     data = maisons,
     xlab = "nombre de chambres",
     ylab = "prix (en 100K USD)")
plot(log(price, base = 10) ~ bedrooms,
     data = maisons,
     xlab = "nombre de chambres",
     ylab = "prix (en log[10] USD)")
with(maisons |> dplyr::filter(bedrooms < 9),
     plot(x = bedrooms,
          y = price / 1e5,
          xlab = "nombre de chambres",
          ylab = "prix (en 100K USD)"))
with(maisons |> dplyr::filter(bedrooms < 9),
     plot(x = bedrooms,
          y = log(price, base = 10), # base 10 pour interprétabilité
          xlab = "nombre de chambres",
          ylab = expression("prix (en log[10] USD)"))))
```



Exercice 0.5. Répétez cette opération pour les autres variables. Quelles tendances se dégagent?

Exploration des données: ggplot

La librairie `ggplot2` permet de réaliser des graphiques plus complexes et de qualité bien supérieure à ceux de la fonction de base `plot`. Vous pouvez consulter la [galerie de graphiques produits avec ggplot](#) en ligne.

Nous n'allons pas faire une introduction complète à `ggplot2` ici (ce serait bien trop long, il y a des livres entiers sur le sujet), mais vous donner quelques exemples de graphiques de visualisation de données. La syntaxe de `ggplot2` est par contre différente des autres fonctions R classiques, et cela demande un peu d'adaptation. Un graphique consiste en effet en une succession de commandes séparée avec le symbole `+`.

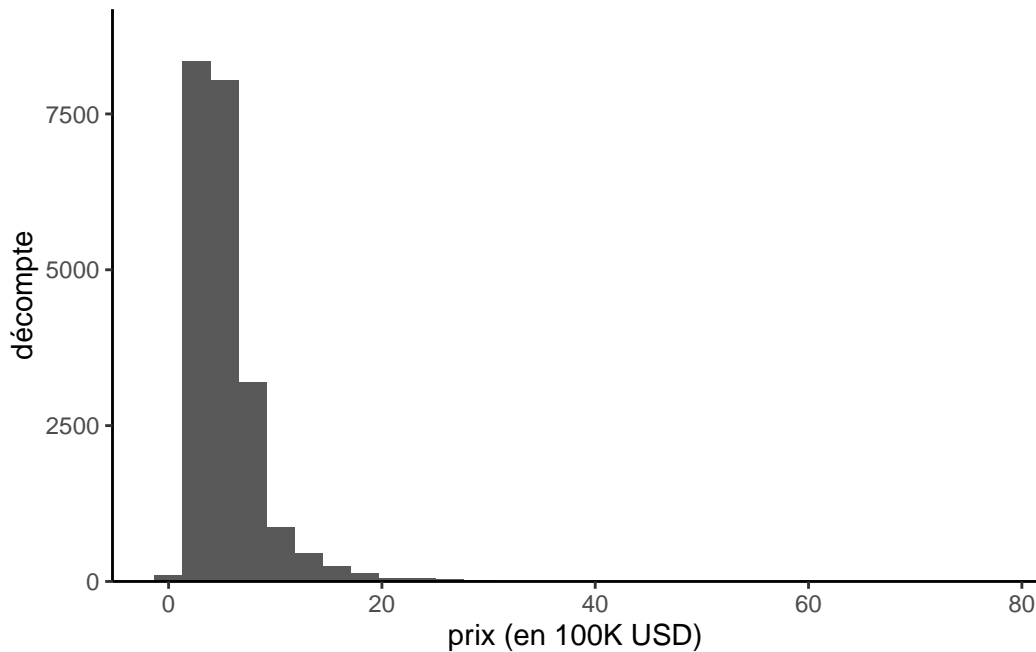
Avant de commencer, assurez-vous que vous avez installé les paquets `ggplot2`, `patchwork` et `ggridges`.

Commençons par effectuer un histogramme et un diagramme à bandes pour la variable `price`. La fonction `ggplot` permet de définir le cadre du graphique de base et de définir quelles sont les variables qui vont être incluses.


```
library(ggplot2) # grammaire des graphiques
library(patchwork) # combiner des objets ggplot
theme_set(theme_classic()) # choisir le thème graphique
```

En combinant la fonction de base `ggplot` avec d'autres fonctions spécifiques, toutes sortes de graphiques peuvent être réalisés, comme un histogramme par exemple:

```
ggplot(data = maisons,
       mapping = aes(x = price / 1e5)) +
  geom_histogram() + # choix de la représentation graphique
  scale_y_continuous(limits = c(0, NA),
                    expand = expansion(mult = c(0, 0.1))) +
  labs(x = "prix (en 100K USD)", # étiquettes des axes, titre, sous-titre, etc.
       y = "décompte")
```



Il est possible de contrôler le nombre de séparations de l'histogramme avec l'option `binwidth` (largeur des boîtes) ou `bins` (nombre de catégories), par exemple ici en incréments de 50K dollars.

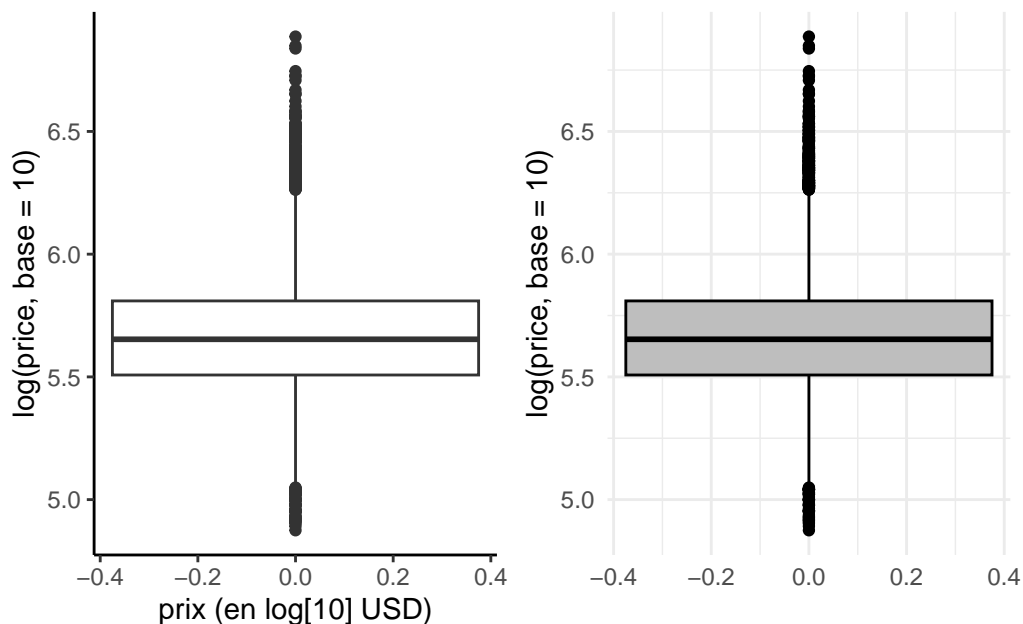
```
g1 <- ggplot(maisons,
             aes(x = log(price, base = 10))) +
  geom_histogram(binwidth = 0.5) +
```

```
labs(x = "prix (log base 10 USD)",
     y = "décompte",
     caption = "largeur de boîte de 1/2")
```

Voici maintenant comment faire un diagramme en boîte de la variable `prix` à l'échelle logarithmique. Il y a plusieurs options de thèmes et la possibilité, en voici deux exemples:

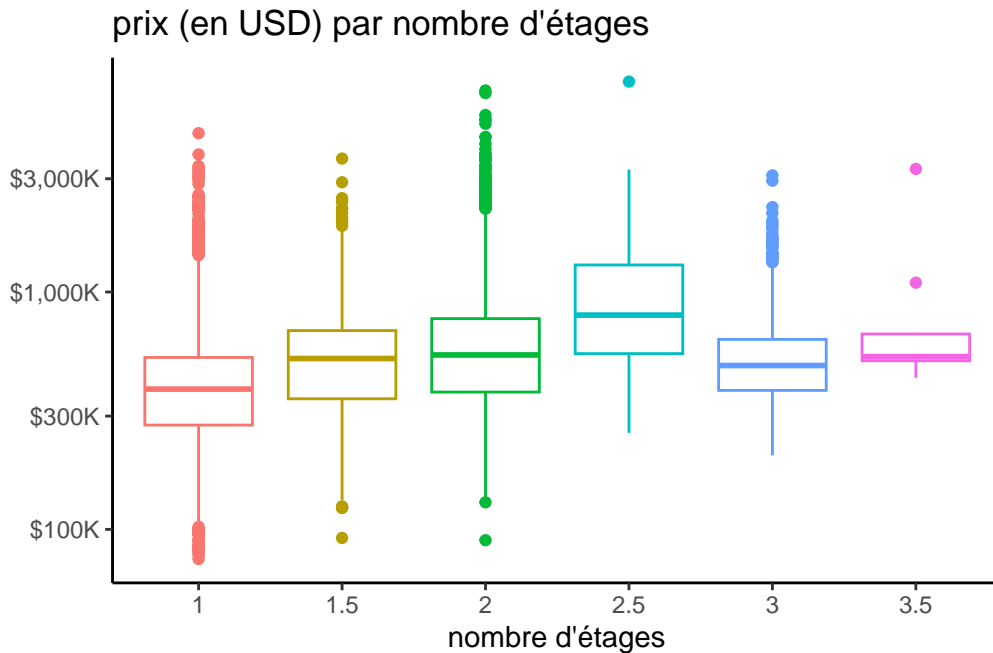
```
g1 <- ggplot(maisons,
             aes(y = log(price, base = 10))) +
  geom_boxplot() +
  labs(x = "prix (en log[10] USD)")

g2 <- ggplot(maisons,
             aes(y = log(price, base = 10))) +
  geom_boxplot(fill = "gray",
               color = "black") +
  theme_minimal()
# Combiner des graphiques avec "patchwork"
# g1 + g2 pour côte à côte, g1 / g2 pour haut/bas
g1 + g2
```



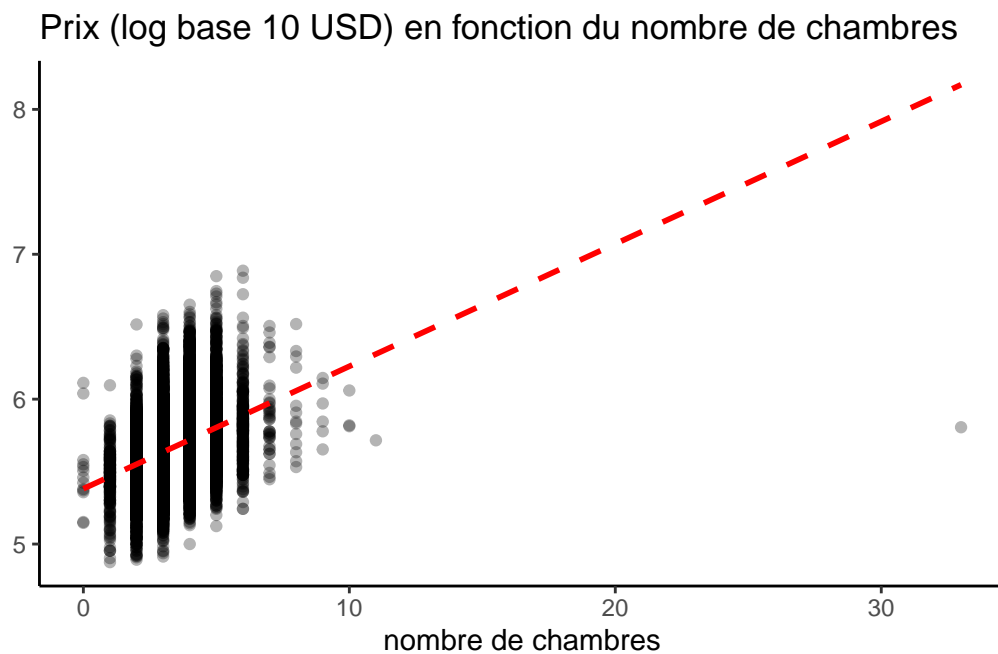
On peut aussi produire des boîtes à moustaches de la variable `logprix` en fonction du nombre d'étages de la maison, par exemple. Notez qu'il faut pour cela que la variable `floors` soit convertie en facteur. Différentes options vous sont montrées ci-dessous.

```
maisons |>
  dplyr::mutate(
    floors_fact = factor(floors)) |>
  ggplot(mapping = aes(x = floors_fact,
                        y = prix,
                        color = floors_fact)) +
  geom_boxplot() +
  scale_y_continuous(transform = "log10",
                     labels = scales::dollar_format(
                       scale = .001,
                       accuracy = 1,
                       suffix = "K")) +
  labs(title = "prix (en USD) par nombre d'étages",
       x = "nombre d'étages",
       y = "") +
  theme(legend.position = "none")
```



Nous pouvons aussi regarder comment le prix des maisons (échelle log) varie en fonction du nombre de chambres par exemple:

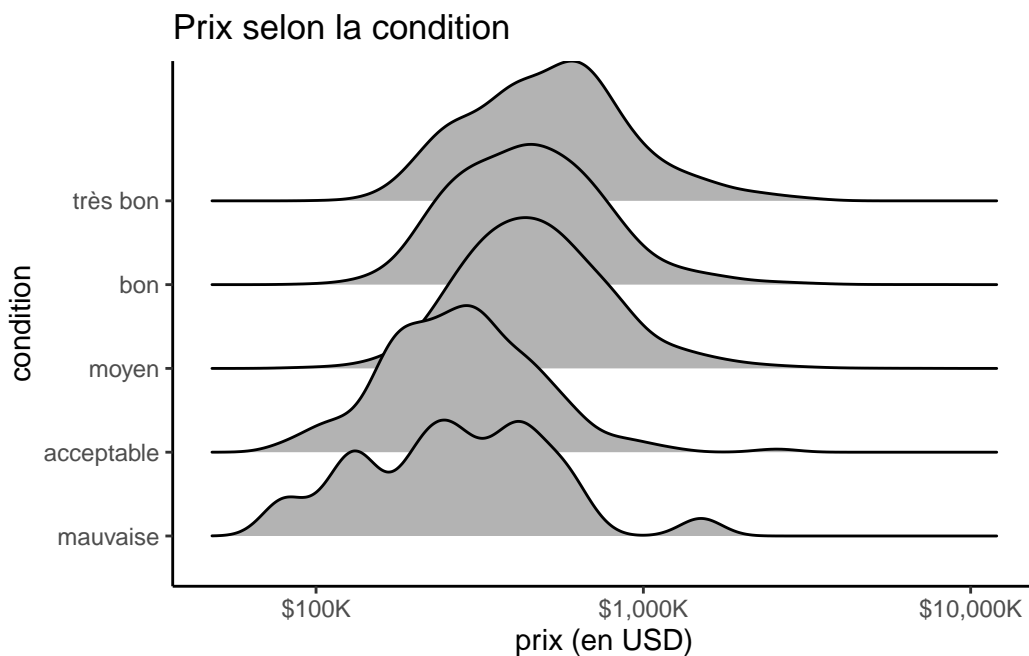
```
ggplot(maisons,
       aes(x = bedrooms,
           y = log(prix, base = 10))) +
  geom_point(alpha = .3) +
  geom_smooth(
    method = "lm",
    se = FALSE,
    color = "red",
    linetype = "dashed") +
  labs(title = "Prix (log base 10 USD) en fonction du nombre de chambres",
       x = "nombre de chambres",
       y = "")
```



Voici un dernier exemple de graphique un peu plus sophistiqué, qui nécessite l'installation du paquet `ggridge`. Ce graphique permet de voir la distribution du prix des maisons selon la condition.

```
maisons |>
  dplyr::mutate(fcondition = factor(
    condition,
    labels = c("mauvaise", "acceptable", "moyen", "bon", "très bon")))
|>
```

```
ggplot(aes(x = price,
           y = fcondition)) +
  ggribes::geom_density_ridges() +
  scale_x_continuous(transform = "log10",
                    labels = scales::dollar_format(
                      scale = .001,
                      accuracy = 1,
                      suffix = "K")) +
  labs(title = "Prix selon la condition",
       x = "prix (en USD)",
       y = "condition") +
  theme_classic()
```



Références

Les possibilités avec le paquet `ggplot` et la grammaire des graphiques sont extrêmement nombreuses. Si vous souhaitez explorer un peu plus le sujet, je vous réfère au livre disponible en ligne [ggplot2: Elegant Graphics for Data Analysis](#)