KAZMIERCZAK Elisa - POLLET Tom

# Compte rendu Business Décision

## Table des matières

KAZMIERCZAK Elisa - POLLET Tom

# 1. Présentation des datasets utilisés

## 1. Bored Ape Yacht Club

Bored Ape Yacht Club est l'une des collections de NFT les plus connues. Elle est composée de 10 000 Bored Ape NFT - des objets de collection numériques uniques vivant sur la blockchain Ethereum. Il existe 170 attributs (appelés aussi traits ou caractéristiques) possibles qui agissent sur l'expression, la coiffure, l'arrière-plan… pour former des NFT uniques. Voici ci -dessous quelques exemples de Bored Ape NFT :

## 2. Nos datasets

### 1 - Dataset des transactions de vente des NFTs de la collection Bored Ape Yacht Club

Le dataset que nous allons utiliser au cours de cette étude vient de Transpose data, et contient une liste des transactions réalisées pour la collection de NFT Bored Ape Yacht Club (cf annexe 1).
Il contient les informations suivantes :

- prix_vente : Valeur totale de cette vente en dollars américain.
- prix_eth : Valeur de l'ETH en dollars américain au moment de la vente (l'ETH est une cryptomonnaie qui sert d'échange pour les NFTs).
- volume_opensea : le volume total des échanges en dollars américain de la marketplace permettant de vendre et acheter des NFTs (Opensea est le nom de cette plateforme).
- nb_traits: Le nombre de traits du NFT
- frais_royalty : Les frais liés aux royalties
- frais_platforme : Les frais liés de transaction à la plateforme

## 2 - Datasets tableau de contingence des caractéristiques des NFTs

Comme dit précédemment, chaque NFT au sein de la collection Bored Ape Yacht Club possède une combinaison d'attributs spécifiques. Par exemple, le NFT ci-dessous a pour attributs :
Background : New Punk Blue
Clothes : Prison Jumpsuit
Earring : None
Eyes : Bored
Fur : Brown
Hat : Horns
Mouth : Rage

Alors que celui-ci a pour attributs :
Background : Yellow
Clothes : Tie Dye
Earring : Gold Hoop
Eyes : Bored
Fur : Red
Hat : None
Mouth :  Phoneme L

Nous avons donc réalisé 7 tableaux de contingence (un par type d'attribut) entre les attributs et la catégorie de prix de la vente de ces NFTs. La catégorie de prix de vente a été créée en discrétisant la variable du prix de vente (qui est une variable quantitative continue) en 4 classes :

- price_category_1 qui correspond aux prix inférieurs ou égaux à 3293.835$.
- price_category_2 qui correspond aux prix strictement supérieurs à 3293.835$ et inférieurs ou égaux à 18356.33$.
- price_category_3 qui correspond aux prix strictement supérieurs à 18356.33$ et inférieurs ou égaux à 124216.6$.
- price_category_4 qui correspond aux prix strictement supérieurs à 124216.6$.

Les valeurs ont été choisies à partir des quantiles des prix de ventes.

Voici par exemple à quoi ressemble le tableau de contingence du type d'attribut "Fur" :

| | price_category_1 | price_category_2 | price_category_3 | price_category_4 |
|---|---|---|---|---|
| **Black** | 1156 | 1089 | 1109 | 1169 |
| **Blue** | 428 | 395 | 293 | 355 |
| **Brown** | 1230 | 1164 | 1327 | 1238 |
| **Cheetah** | 318 | 250 | 188 | 292 |
| **Cream** | 588 | 592 | 744 | 599 |
| **Dark Brown** | 1179 | 1184 | 1211 | 1135 |
| **Death Bot** | 96 | 119 | 78 | 117 |
| **Dmt** | 108 | 166 | 105 | 140 |
| **Golden Brown** | 689 | 668 | 734 | 698 |
| **Gray** | 419 | 436 | 413 | 440 |
| **Noise** | 94 | 112 | 75 | 84 |
| **Pink** | 410 | 423 | 415 | 406 |
| **Red** | 397 | 403 | 404 | 375 |
| **Robot** | 163 | 191 | 112 | 145 |
| **Solid Gold** | 1 | 15 | 25 | 27 |
| **Tan** | 564 | 614 | 613 | 535 |
| **Trippy** | 4 | 45 | 35 | 41 |
| **White** | 330 | 294 | 295 | 337 |
| **Zombie** | 197 | 210 | 195 | 238 |

## 3 - Dataset brut des transactions de vente de NFTs

Ce dataset nommé BAYC_3.csv est le dataset d'origine que nous avons récupéré depuis Transpose (API permettant de faire des requêtes SQL sur les transactions de vente des NFTs).

# 2. Analyse en composantes principales

Dans cette partie nous utiliserons le dataset 1 des transactions de vente des NFTs de la collection Bored Ape Yacht Club.
Le but de cette première analyse est de réduire le nombre de variables en utilisant des composantes principales. Ces composantes principales, qui sont des combinaisons linéaires des variables d'origine, captent le maximum de variation dans le dataset. Cela nous permettra de simplifier l'analyse et la visualisation des données pour la suite de l'étude.

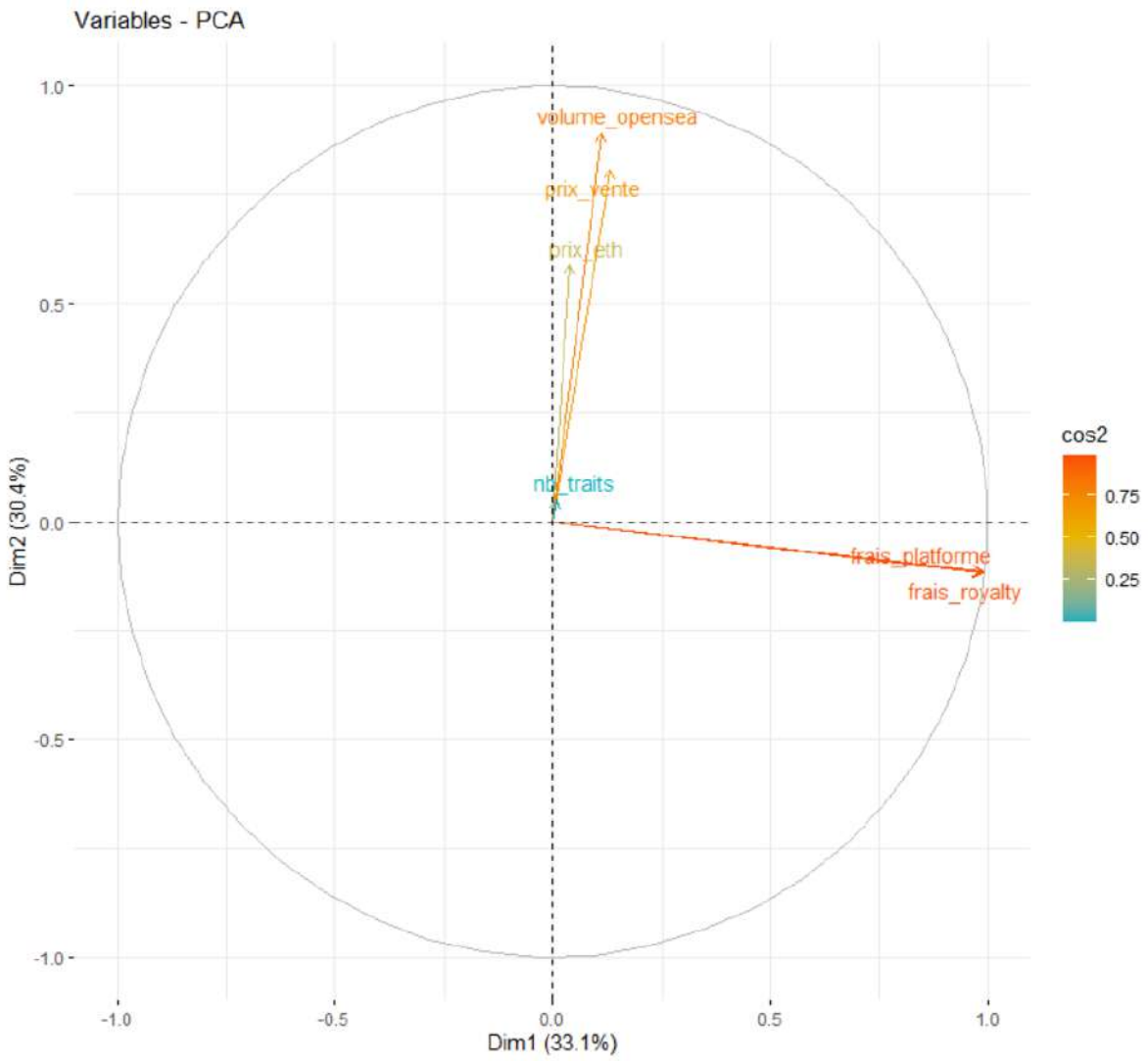
## 1. Corrélation entre les axes

Nous allons commencer par observer s'il existe une corrélation ou non entre les différentes données du dataset. Pour cela, nous allons réaliser le graphique de corrélation des variables. Nous pourrons l'interpréter de la façon suivante : les variables positivement corrélées sont regroupées alors que celles corrélées négativement sont positionnées sur les côtés opposés du graphique. De plus, plus l'axe est grand, mieux il est représenté par l'ACP.

Code utilisé :

```
# Cercle de corrélation coloré en fonction du cos2

res.pca <- PCA(data, graph = FALSE)

fviz_pca_var(res.pca, col.var = "cos2",

             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),

             repel = TRUE # Évite le chevauchement de texte

             )
```

Résultat graphique :




On peut donc déduire de ce graphique les informations suivantes :
Les frais plateforme et les royalties sont corrélés positivement entre eux. De même pour le prix de vente et le volume opensea. D'autre part, on remarque que les frais plateforme et les royalties ne sont pas du tout corrélés au prix. Ainsi, les frais de plateforme et les royalties ne sont pas proportionnels au prix de la NFT. Enfin, le nombre de traits n'est pas bien représenté par l'ACP car sa distance à l'origine du graphique est très petite. Il n'influe donc pas sur le prix non plus.

## 2. Variance des composantes principales

L'objectif ici est de définir les composantes principales, puis de comprendre quels sont les anciens axes qui contribuent le plus à ces composantes principales. Ceci nous permettra de garder les composantes les plus importantes pour expliquer la variabilité de notre dataset.

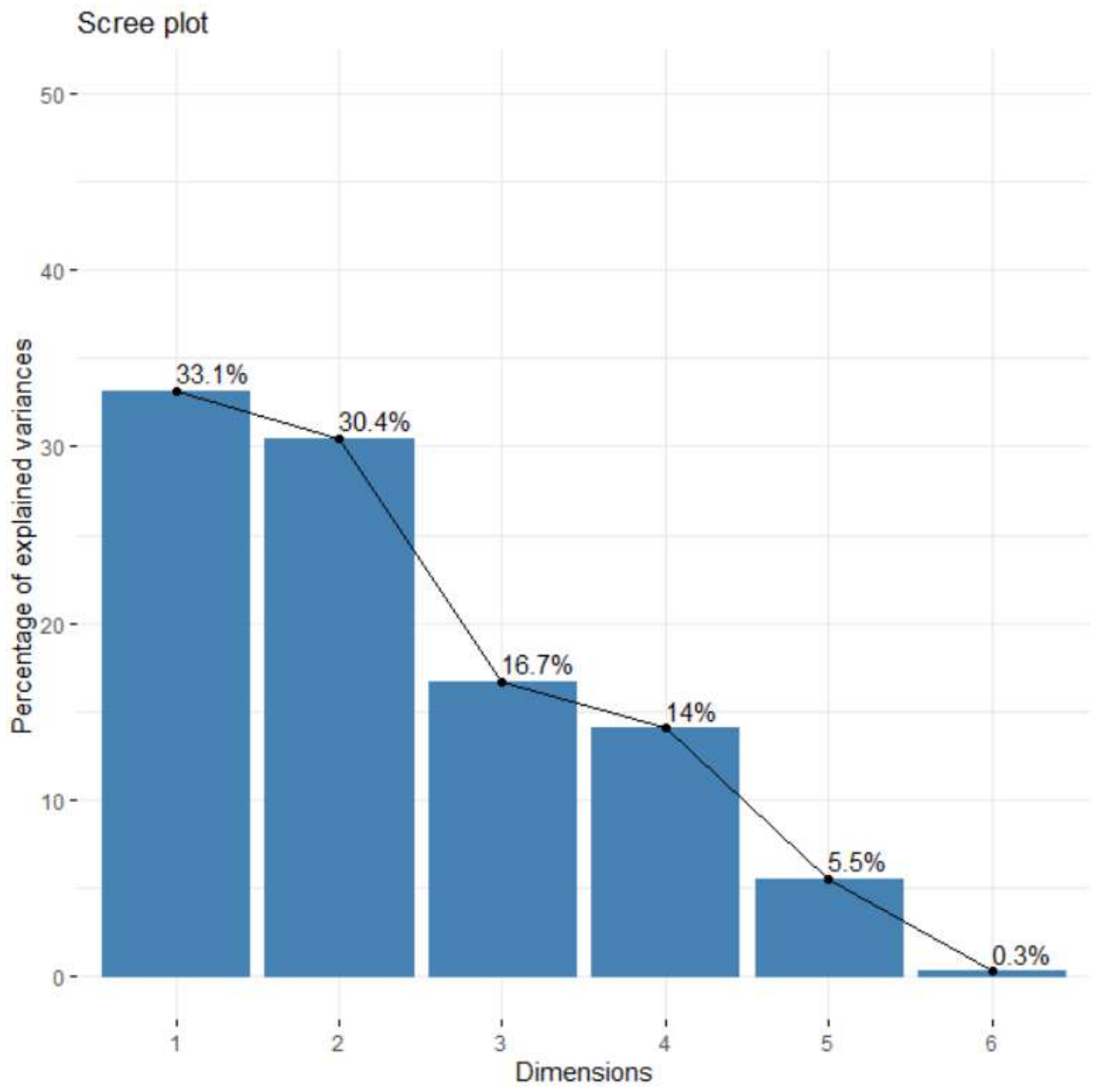
KAZMIERCZAK Elisa - POLLET Tom

## Histogramme des composantes principales

Code utilisé :

```
fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 50))
```

Résultat graphique :



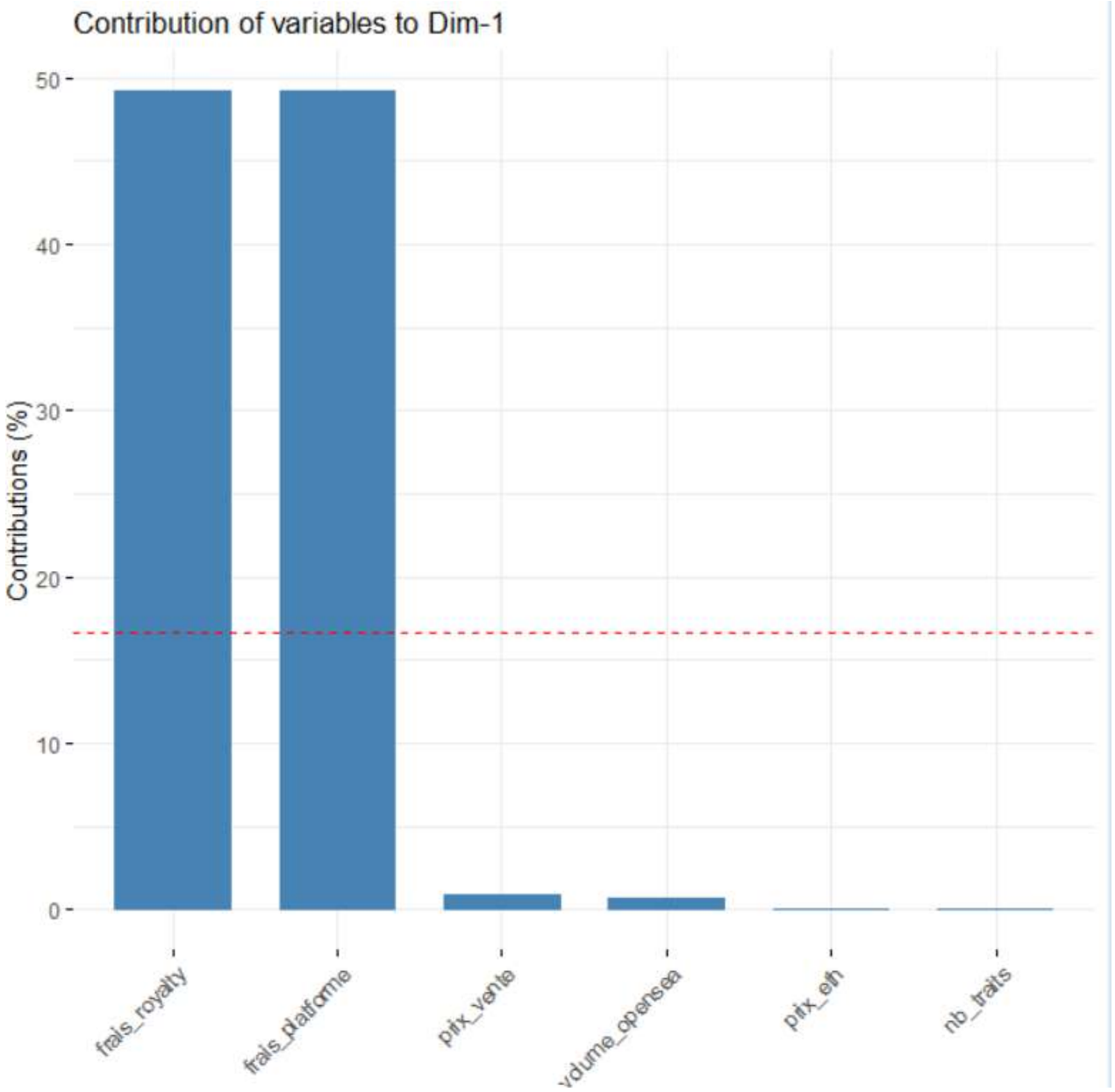

A partir du graphique ci-dessus, on peut voir que 63.4% des variances contenues dans les données sont conservées par les 2 premières composantes principales, ce qui est un pourcentage acceptable.

## Histogramme des contributions sur l'axe 1

Code utilisé :

```
fviz_contrib(res.pca, choice="var", axes = 1 )
```

Résultat graphique :



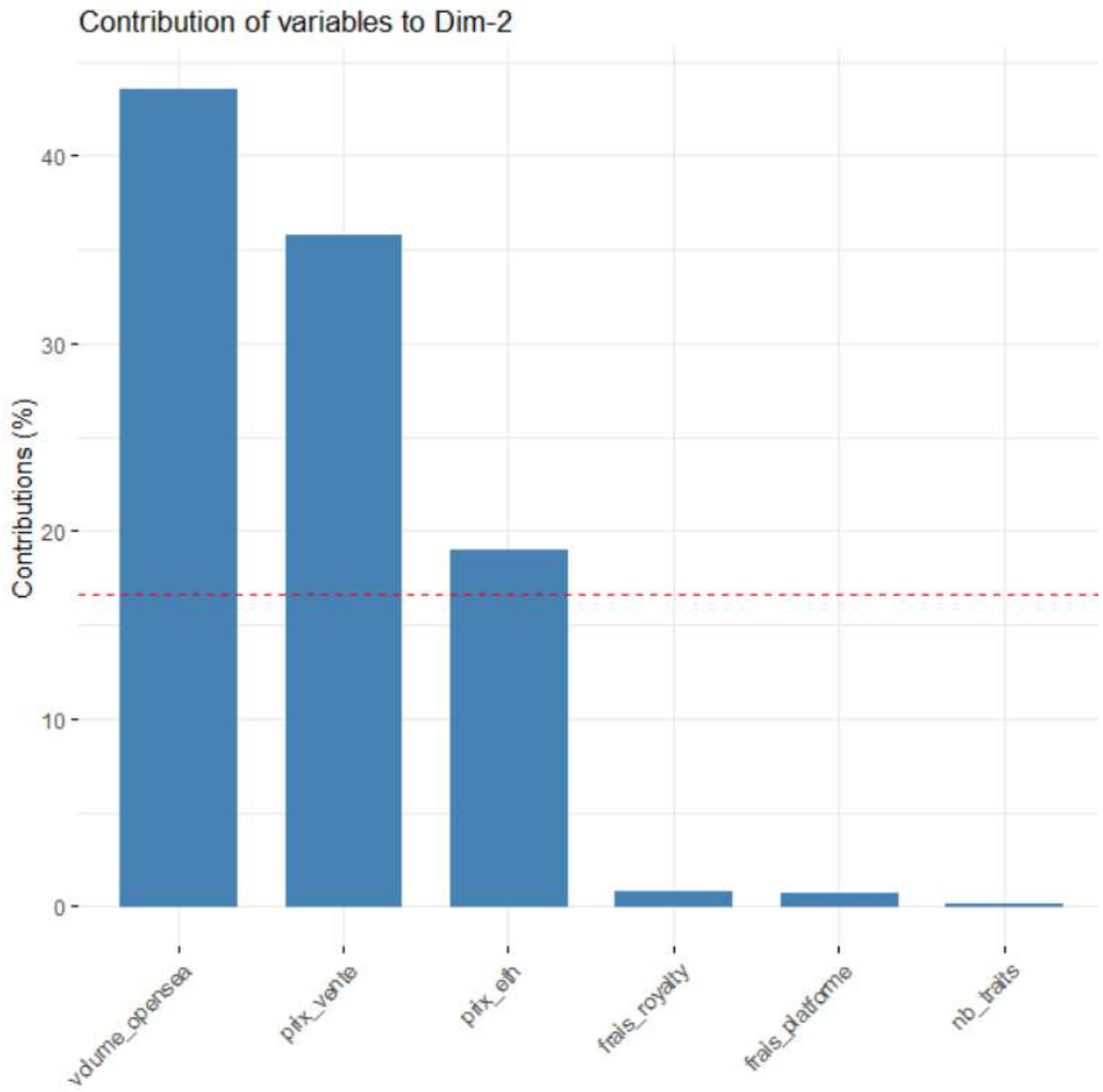

Sur le graphique ci-dessus, on peut voir que la contribution des variables frais_royalty et frais_platforme se situe bien au-dessus de la contribution moyenne attendue (en pointillés rouge sur le graphique). On considère donc que ces deux variables contribuent de manière importante à la composante principale 1.

## Histogramme des contributions sur l'axe 2

Code utilisé :

```
fviz_contrib(res.pca, choice="var", axes = 2 )
```

Résultat graphique :




Sur l'histogramme ci-dessus, on peut voir que la contribution des variables volume_opensea, prix_vente et prix_eth se situe bien de la contribution moyenne attendue (en pointillés rouge sur le graphique). On considère donc que ces trois données contribuent de manière importante à la composante principale 2.

Grâce à ces deux graphiques, on remarque que le nombre de traits n'a pas d'impact significatif sur les deux premières composantes principales. Étant donné qu'elles représentent plus de 60% des informations importantes sur la variance de nos données, on va considérer que le nombre de traits n'est pas une donnée pertinente pour bien décrire les variance entre nos données.

# 3. Classification

Dans cette partie nous utiliserons le dataset 1 des transactions de vente des NFTs de la collection Bored Ape Yacht Club. L'objectif de cette classification est de mieux comprendre la structure des données en la divisant en plusieurs groupes.

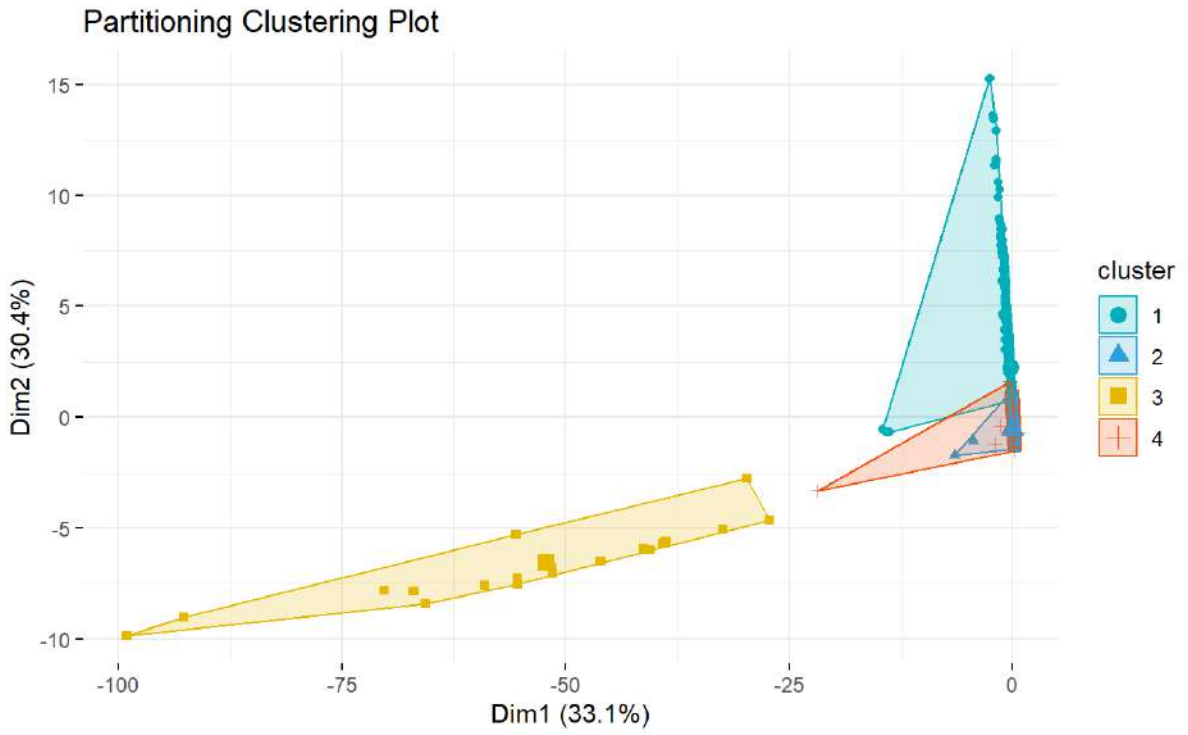
## 1. Classification non hiérarchique

Ici nous allons diviser notre dataset en plusieurs groupes de données en fonction des deux premières composantes principales identifiées précédemment.

Code utilisé :

```
fviz_cluster(res.hcpc,
             repel = TRUE,             # Evite le chevauchement des textes
             show.clust.cent = TRUE, # Montre le centre des clusters
                    palette = "jco",             # Palette de couleurs, voir
?ggpubr::ggpar
             ggtheme = theme_minimal(),
             main = "Factor map"
             )
```

Résultat graphique :




Cette représentation graphique nous permet d'identifier clairement 4 clusters à partir de nos données. Pour comprendre les différences entre ces groupes, il nous faudrait étudier plus en détail les caractéristiques des transactions elles-mêmes.

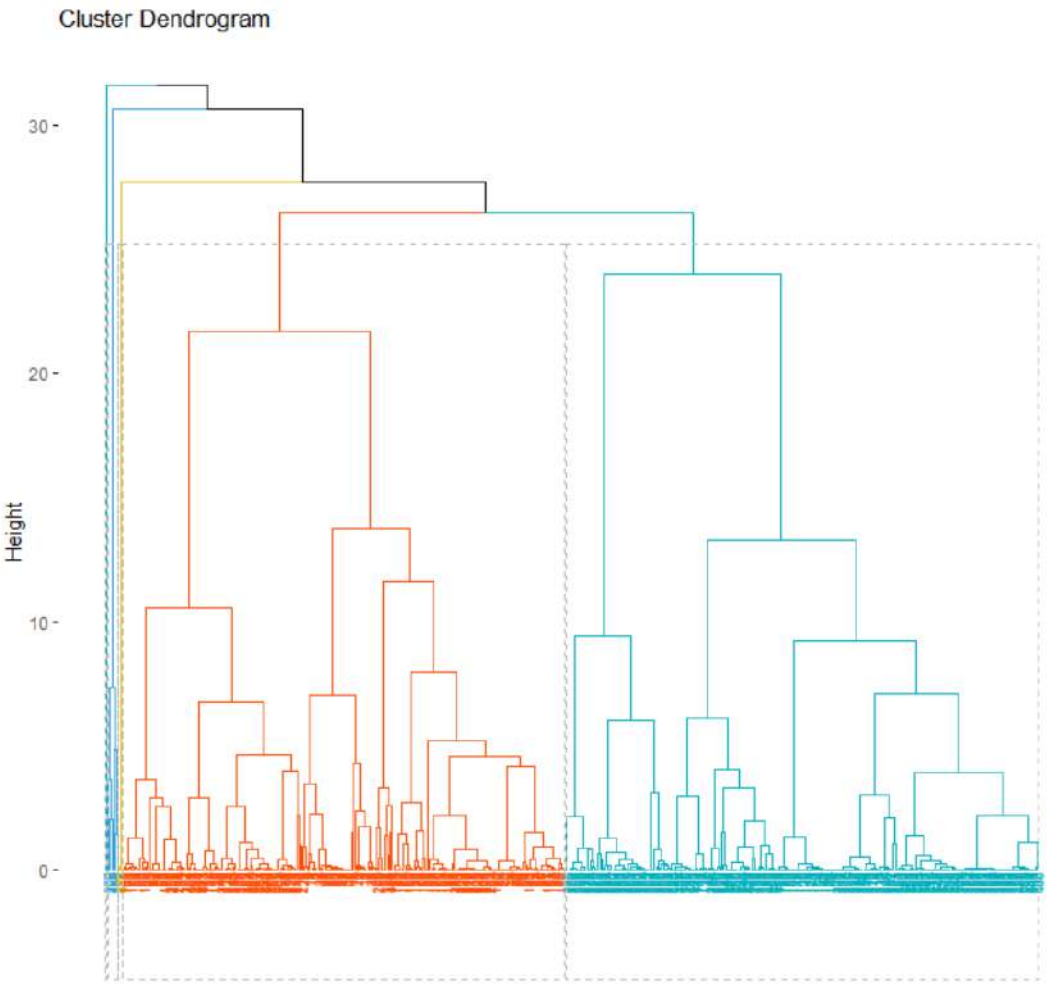
## 2. Classification hiérarchique

Le but de la classification hiérarchique est d'identifier des groupes d'observations similaires dans notre dataset pour mieux comprendre comment les données sont structurées.

Code utilisé :

```
res <- hcut(data, k = 5, stand = TRUE)

fviz_dend(res, rect = TRUE, cex = 0.5,

      k_colors = c("#00AFBB","#2E9FDF", "#E7B800", "#FC4E07"))
```

Résultat graphique :




Ce graphique nous permet d'identifier clairement deux groupes principaux, et trois autres groupes plus petits.

# 4. Analyse Discriminante

Dans cette partie nous utiliserons le dataset des transactions de vente des NFTs de la collection Bored Ape Yacht Club.

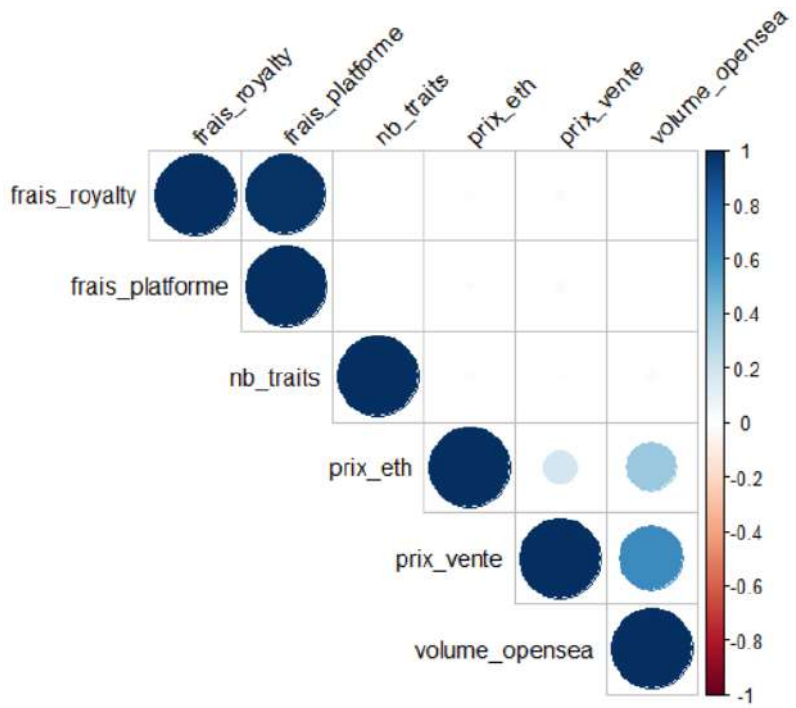
## 1. Matrice de corrélation

Nous avons utilisé la matrice de corrélation afin de mettre en évidence les variables les plus corrélées d'une autre manière.

Code utilisé :

```
mcor <- cor(data)
library(corrplot)
corrplot(mcor, type="upper", order="hclust", tl.col="black", tl.srt=45)
```

Résultat graphique :




Grâce à cette représentation graphique, on peut voir que comme précédemment les frais liés aux royalties et ceux liés à la plateforme sont étroitement corrélés. Le prix de vente et le volume sur la plateforme Opensea sont également corrélés (le prix de vente et le prix d'ETH aussi dans une moindre mesure). Ces résultats sont cohérents avec ceux obtenus lors de l'analyse en composantes principales.

# 5. Test d'ajustement du Khi-Deux

Dans cette partie, nous utiliserons les 7 tableaux de contingences "catégorie de prix / attributs".

Le but de test du Khi-Deux est ici de déterminer s'il existe une association ou une dépendance entre les variables catégorielles de nos données. Dans notre cas, une dépendance signifierait que le prix de vente d'un NFT dépend de ses attributs / caractéristiques.

Code utilisé :

```r
chisq.test(tableau_contingence_background)
chisq.test(tableau_contingence_clothes)
chisq.test(tableau_contingence_earring)
chisq.test(tableau_contingence_eyes)
chisq.test(tableau_contingence_fur)
chisq.test(tableau_contingence_hat)
chisq.test(tableau_contingence_mouth)
```

Résultat :

```
        Pearson's Chi-squared test

data:  tableau_contingence_background
X-squared = 34.93, df = 21, p-value = 0.02874


        Pearson's Chi-squared test

data:  tableau_contingence_clothes
X-squared = 476.65, df = 126, p-value < 2.2e-16


        Pearson's Chi-squared test

data:  tableau_contingence_earring
X-squared = 41.599, df = 15, p-value = 0.000259


        Pearson's Chi-squared test

data:  tableau_contingence_eyes
X-squared = 421.46, df = 66, p-value < 2.2e-16


        Pearson's Chi-squared test

data:  tableau_contingence_fur
X-squared = 251.64, df = 54, p-value < 2.2e-16


        Pearson's Chi-squared test

data:  tableau_contingence_hat
X-squared = 589.6, df = 105, p-value < 2.2e-16


        Pearson's Chi-squared test

data:  tableau_contingence_mouth
X-squared = 421.87, df = 96, p-value < 2.2e-16
```

Interprétation :

LOI DU KHI-DEUX AVEC $k$ DEGRÉS DE LIBERTÉ
QUANTILES D'ORDRE $1-\gamma$

| | $\gamma$ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | 0.995 | 0.990 | 0.975 | 0.950 | 0.900 | 0.500 | 0.100 | 0.050 | 0.025 | 0.010 | 0.005 |
| 1 | 0.00 | 0.00 | 0.00 | 0.00 | 0.02 | 0.45 | 2.71 | 3.84 | 5.02 | 6.63 | 7.88 |
| 2 | 0.01 | 0.02 | 0.05 | 0.10 | 0.21 | 1.39 | 4.61 | 5.99 | 7.38 | 9.21 | 10.60 |
| 3 | 0.07 | 0.11 | 0.22 | 0.35 | 0.58 | 2.37 | 6.25 | 7.81 | 9.35 | 11.34 | 12.84 |
| 4 | 0.21 | 0.30 | 0.48 | 0.71 | 1.06 | 3.36 | 7.78 | 9.94 | 11.14 | 13.28 | 14.86 |
| 5 | 0.41 | 0.55 | 0.83 | 1.15 | 1.61 | 4.35 | 9.24 | 11.07 | 12.83 | 15.09 | 16.75 |
| 6 | 0.68 | 0.87 | 1.24 | 1.64 | 2.20 | 5.35 | 10.65 | 12.59 | 14.45 | 16.81 | 18.55 |
| 7 | 0.99 | 1.24 | 1.69 | 2.17 | 2.83 | 6.35 | 12.02 | 14.07 | 16.01 | 18.48 | 20.28 |
| 8 | 1.34 | 1.65 | 2.18 | 2.73 | 3.49 | 7.34 | 13.36 | 15.51 | 17.53 | 20.09 | 21.96 |
| 9 | 1.73 | 2.09 | 2.70 | 3.33 | 4.17 | 8.34 | 14.68 | 16.92 | 19.02 | 21.67 | 23.59 |
| 10 | 2.16 | 2.56 | 3.25 | 3.94 | 4.87 | 9.34 | 15.99 | 18.31 | 20.48 | 23.21 | 25.19 |
| 11 | 2.60 | 3.05 | 3.82 | 4.57 | 5.58 | 10.34 | 17.28 | 19.68 | 21.92 | 24.72 | 26.76 |
| 12 | 3.07 | 3.57 | 4.40 | 5.23 | 6.30 | 11.34 | 18.55 | 21.03 | 23.34 | 26.22 | 28.30 |
| 13 | 3.57 | 4.11 | 5.01 | 5.89 | 7.04 | 12.34 | 19.81 | 22.36 | 24.74 | 27.69 | 29.82 |
| 14 | 4.07 | 4.66 | 5.63 | 6.57 | 7.79 | 13.34 | 21.06 | 23.68 | 26.12 | 29.14 | 31.32 |
| 15 | 4.60 | 5.23 | 6.27 | 7.26 | 8.55 | 14.34 | 22.31 | 25.00 | 27.49 | 30.58 | 32.80 |
| 16 | 5.14 | 5.81 | 6.91 | 7.96 | 9.31 | 15.34 | 23.54 | 26.30 | 28.85 | 32.00 | 34.27 |
| 17 | 5.70 | 6.41 | 7.56 | 8.67 | 10.09 | 16.34 | 24.77 | 27.59 | 30.19 | 33.41 | 35.72 |
| 18 | 6.26 | 7.01 | 8.23 | 9.39 | 10.87 | 17.34 | 25.99 | 28.87 | 31.53 | 34.81 | 37.16 |
| 19 | 6.84 | 7.63 | 8.81 | 10.12 | 11.65 | 18.34 | 27.20 | 30.14 | 32.85 | 36.19 | 38.58 |
| 20 | 7.43 | 8.26 | 9.59 | 10.85 | 12.44 | 19.34 | 28.41 | 31.41 | 34.17 | 37.57 | 40.00 |
| 21 | 8.03 | 8.90 | 10.28 | 11.59 | 13.24 | 20.34 | 29.62 | 32.67 | 35.48 | 38.93 | 41.40 |
| 22 | 8.64 | 9.54 | 10.98 | 12.34 | 14.04 | 21.34 | 30.81 | 33.92 | 36.78 | 40.29 | 42.80 |
| 23 | 9.26 | 10.20 | 11.69 | 13.09 | 14.85 | 22.34 | 32.01 | 35.17 | 38.08 | 41.64 | 44.18 |
| 24 | 9.89 | 10.86 | 12.40 | 13.85 | 15.66 | 23.34 | 33.20 | 36.42 | 39.36 | 42.98 | 45.56 |
| 25 | 10.52 | 11.52 | 13.12 | 14.61 | 16.47 | 24.34 | 34.28 | 37.65 | 40.65 | 44.31 | 46.93 |
| 26 | 11.16 | 12.20 | 13.84 | 15.38 | 17.29 | 25.34 | 35.56 | 38.89 | 41.92 | 45.64 | 48.29 |
| 27 | 11.81 | 12.88 | 14.57 | 16.15 | 18.11 | 26.34 | 36.74 | 40.11 | 43.19 | 46.96 | 49.65 |
| 28 | 12.46 | 13.57 | 15.31 | 16.93 | 18.94 | 27.34 | 37.92 | 41.34 | 44.46 | 48.28 | 50.99 |
| 29 | 13.12 | 14.26 | 16.05 | 17.71 | 19.77 | 28.34 | 39.09 | 42.56 | 45.72 | 49.59 | 52.34 |
| 30 | 13.79 | 14.95 | 16.79 | 18.49 | 20.60 | 29.34 | 40.26 | 43.77 | 46.98 | 50.89 | 53.67 |
| 40 | 20.71 | 22.16 | 24.43 | 26.51 | 29.05 | 39.34 | 51.81 | 55.76 | 59.34 | 63.69 | 66.77 |
| 50 | 27.99 | 29.71 | 32.36 | 34.76 | 37.69 | 49.33 | 63.17 | 67.50 | 71.42 | 76.15 | 79.49 |
| 60 | 35.53 | 37.48 | 40.48 | 43.19 | 46.46 | 59.33 | 74.40 | 79.08 | 83.30 | 88.38 | 91.95 |
| 70 | 43.28 | 45.44 | 48.76 | 51.74 | 55.33 | 69.33 | 85.53 | 90.53 | 95.02 | 100.42 | 104.22 |
| 80 | 51.17 | 53.54 | 57.15 | 60.39 | 64.28 | 79.33 | 96.58 | 101.88 | 106.63 | 112.33 | 116.32 |
| 90 | 59.20 | 61.75 | 65.65 | 69.13 | 73.29 | 89.33 | 107.57 | 113.14 | 118.14 | 124.12 | 128.30 |
| 100 | 67.33 | 70.06 | 74.22 | 77.93 | 82.36 | 99.33 | 118.50 | 124.34 | 129.56 | 135.81 | 140.17 |

Pour le tableau de contingence “background”, le seuil alpha 5% à 21 degrés de liberté = 32,67. X-squared = 34,93 > 32,67, on peut donc en conclure au risque 5% que les données sont dépendantes. Donc, le prix d’un NFT dépend de sa caractéristique “background”.

Pour le tableau de contingence “clothes”, le seuil alpha 5% à 126 degrés de liberté = 233,99 (X-squared le plus proche à 200 ddl). X-squared = 476,65 > 233,99, on peut donc en conclure au risque 5% que les données sont dépendantes. Donc, le prix d’un NFT dépend de sa caractéristique “clothes”.

Pour le tableau de contingence “earring”, le seuil alpha 5% à 15 degrés de liberté = 25. X-squared = 41,599 > 25, on peut donc en conclure au risque 5% que les données sont dépendantes. Donc, le prix d’un NFT dépend de sa caractéristique “earring”.

Pour le tableau de contingence “eyes”, le seuil alpha 5% à 66 degrés de liberté = 90,53. X-squared = 421,46 > 90,53, on peut donc en conclure au risque 5% que les données sont dépendantes. Donc, le prix d’un NFT dépend de sa caractéristique “eyes”.

Pour le tableau de contingence “fur”, le seuil alpha 5% à 54 degrés de liberté = 79,08. X-squared = 251,64 > 79,08, on peut donc en conclure au risque 5% que les données sont dépendantes. Donc, le prix d’un NFT dépend de sa caractéristique “fur”.

Pour le tableau de contingence “hat”, le seuil alpha 5% à 105 degrés de liberté = 233,99 (X-squared le plus proche à 200 ddl). X-squared = 589,6 > 233,99, on peut donc en conclure au risque 5% que les données sont dépendantes. Donc, le prix d’un NFT dépend de sa caractéristique “hat”.

Pour le tableau de contingence “mouth”, le seuil alpha 5% à 96 degrés de liberté = 124,34. X-squared = 421,87 > 124,34, on peut donc en conclure au risque 5% que les données sont dépendantes. Donc, le prix d’un NFT dépend de sa caractéristique “mouth”.

On peut finalement conclure de ces tests du Khi 2 que le prix d’un NFT dépend des caractéristiques de ce NFT. On peut néanmoins avoir un doute pour la caractéristique “bachground” étant donné que sa statistique de test est proche de son seuil.

# 6. Anova

Le test ANOVA est une méthode statistique utilisée pour comparer les moyennes de trois groupes ou plus. Il permet de déterminer s'il existe des différences significatives entre les moyennes des groupes.

Dans cette partie, nous utiliserons le jeu de données intégré à R “PlantGrowth” car nos datasets ne se prêtent pas à une analyse de l’égalité entre des moyennes. Ce jeu de données contient le poids de plantes sous différents traitements différents. Le but ici est de déterminer s' il y a une différence significative entre ces traitements sur le poids des plantes.

Code utilisé :

```r
res.aov <- aov(weight ~ group, data = data)

summary(res.aov)
```

Résultat :

```
             Df Sum Sq Mean Sq F value Pr(>F)
group         2  3.766  1.8832   4.846 0.0159 *
Residuals    27 10.492  0.3886
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Interprétation :
On observe ici que la p-value (0,0159) est strictement inférieure au seuil 0,05, nous pouvons donc conclure qu'il existe des différences significatives entre les groupes. Les traitements ont donc un impact sur le poids des plantes.

# 7. AFC

Dans cette partie, nous utiliserons les 7 tableaux de contingences “catégorie de prix / caractéristiques”.

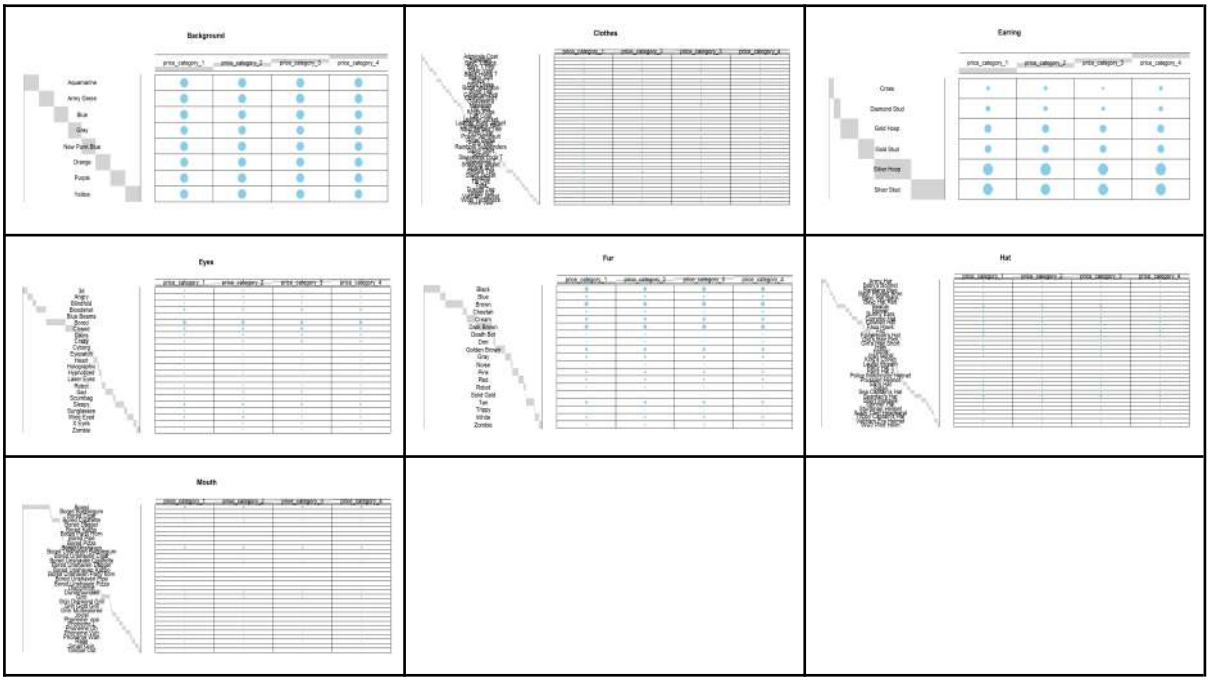
L’analyse factorielle des correspondances est une extension de l’analyse en composantes principales pour analyser l’association entre deux variables qualitatives. Ici nos variables sont en effet qualitatives car les caractéristiques et les catégories de prix de vente sont des classes.

## 1. Graphique des tableaux de contingence

Code utilisé :

```r
library("gplots")


# 1. convertir les données en tant que table
dt.background <- as.table(as.matrix (tableau_contingence_background))
dt.clothes <- as.table(as.matrix (tableau_contingence_clothes))
dt.earring <- as.table(as.matrix (tableau_contingence_earring))
dt.eyes <- as.table(as.matrix (tableau_contingence_eyes))
dt.fur <- as.table(as.matrix (tableau_contingence_fur))
dt.hat <- as.table(as.matrix (tableau_contingence_hat))
dt.mouth <- as.table(as.matrix (tableau_contingence_mouth))


# 2. Graphique
balloonplot(t (dt.background), main = "housetasks", xlab = "", ylab = "",
            label = FALSE, show.margins = FALSE)
balloonplot(t (dt.clothes), main = "housetasks", xlab = "", ylab = "",
            label = FALSE, show.margins = FALSE)
balloonplot(t (dt.earring), main = "housetasks", xlab = "", ylab = "",
            label = FALSE, show.margins = FALSE)
balloonplot(t (dt.eyes), main = "housetasks", xlab = "", ylab = "",
            label = FALSE, show.margins = FALSE)
balloonplot(t (dt.fur), main = "housetasks", xlab = "", ylab = "",
            label = FALSE, show.margins = FALSE)
balloonplot(t (dt.hat), main = "housetasks", xlab = "", ylab = "",
            label = FALSE, show.margins = FALSE)
balloonplot(t (dt.mouth), main = "housetasks", xlab = "", ylab = "",
            label = FALSE, show.margins = FALSE)
```

Résultats :




Interprétation :
Ces graphiques permettent déjà de distinguer visuellement des différences dans la répartition des individus au sein d'une même ligne ou d'une même colonne.

## 2. AFC

Code utilisé :

```r
```{r}
res.ca.background <- CA (tableau_contingence_background, graph = FALSE)
res.ca.clothes <- CA (tableau_contingence_clothes, graph = FALSE)
res.ca.earring <- CA (tableau_contingence_earring, graph = FALSE)
res.ca.eyes <- CA (tableau_contingence_eyes, graph = FALSE)
res.ca.fur <- CA (tableau_contingence_fur, graph = FALSE)
res.ca.hat <- CA (tableau_contingence_hat, graph = FALSE)
res.ca.mouth <- CA (tableau_contingence_mouth, graph = FALSE)
```
```

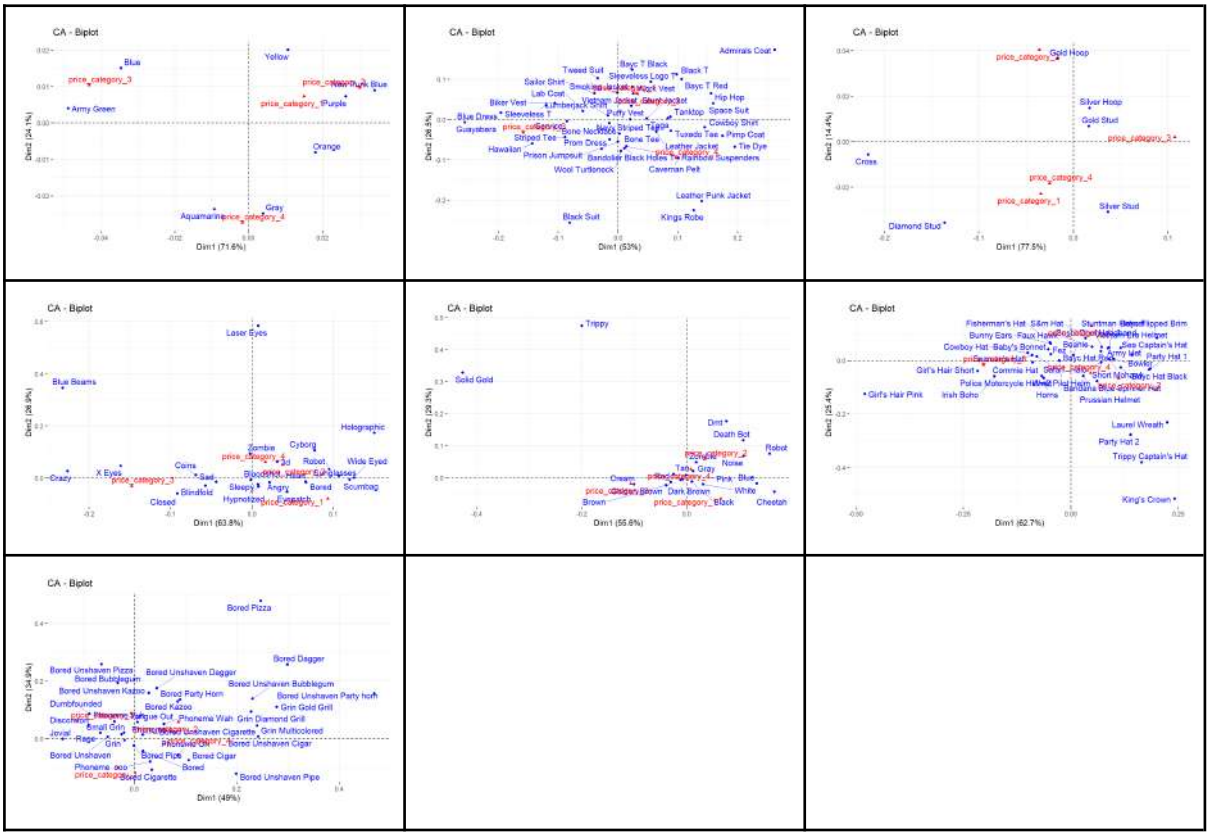
## 3. Biplot

Code utilisé :

```r
```{r}
fviz_ca_biplot(res.ca.background, repel = TRUE)
fviz_ca_biplot(res.ca.clothes, repel = TRUE)
fviz_ca_biplot(res.ca.earring, repel = TRUE)
fviz_ca_biplot(res.ca.eyes, repel = TRUE)
fviz_ca_biplot(res.ca.fur, repel = TRUE)
fviz_ca_biplot(res.ca.hat, repel = TRUE)
fviz_ca_biplot(res.ca.mouth, repel = TRUE)
```
```

KAZMIERCZAK Elisa - POLLET Tom

Résultats :




Interprétation :
Dans le graphique ci-dessus, les lignes sont représentées par des points bleus et des colonnes par des triangles rouges. La distance entre les points lignes (ici les attributs) et la distance entre les points colonnes (ici les catégories de prix) donne une mesure de leur similitude (si proches) ou de leur dissemblance (si éloignés). Les points lignes avec un profil similaire sont proches sur le graphique (c'est la même chose pour les points colonnes).

Le premier graphique (en haut à gauche) correspond au biplot du tableau de contingence du type d'attribut "background". On observe que les attributs "Army Green" et "Blue" ont un profil similaire car ils sont proches dans le graphique, il en va de même pour les attributs "Aquamarine" et "Gray".

Le 3e graphique (en haut à droite) correspond au biplot du tableau de contingence du type d'attribut "earring". On observe que les attributs "Cross" et "Diamond Stud" sont très éloignés des autres attributs.
Le 4e graphique (au milieu à gauche) correspond au biplot du tableau de contingence du type d'attribut "eyes". On observe que les attributs "Blue Beams" et "Laser Eyes" sont très éloignés des autres attributs.
Le 5e graphique (au milieu) correspond au biplot du tableau de contingence du type d'attribut "fur". On observe que les attributs "Solid Gold" et "Trippy" sont très éloignés des autres attributs.

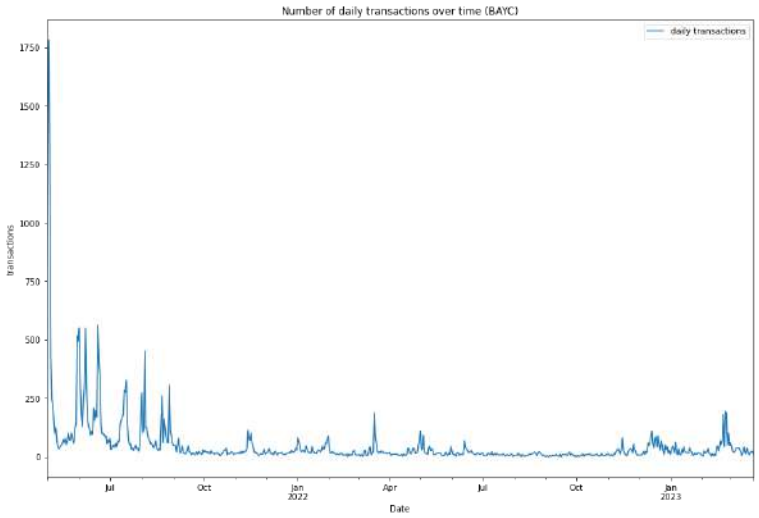
Cela est cohérent car l'ensemble de ces attributs sont particulièrement rares et ont donc un profil différent des autres.

# 8. Analyse avec Python

Code utilisé :

```
Entrée [9]:
BAYC_dates = BAYC[["timestamp"]]
BAYC_dates['Date'] = pd.to_datetime(BAYC_dates['timestamp'], format='%d/%m/%Y %H:%M')
BAYC_dates['Date'] = BAYC_dates['Date'].dt.strftime('%d/%m/%Y')
BAYC_dates = BAYC_dates[['Date']]
BAYC_2 = pd.concat([BAYC_dates, BAYC[["usd_price"]]], axis=1)
BAYC_2.index = BAYC_2["Date"]
BAYC_2.index = pd.to_datetime(BAYC_2.index, format="%d/%m/%Y")
BAYC_daily_transaction = BAYC_2.resample('d').count()
BAYC_daily_transaction = BAYC_daily_transaction['usd_price'].to_frame()
BAYC_daily_transaction = BAYC_daily_transaction.rename(columns={'usd_price': "daily transactions"})
BAYC_daily_transaction.plot(title="Number of daily transactions over time (BAYC)", figsize=(15,10), color="C0",
ylabel="transactions")
```

Résultat :




Interprétation :
Ce graphique permet d'observer l'évolution temporelle du nombre de ventes journalières des NFTs de la collection Bored Ape Yacht Club. On observe un fort engouement pour la collection avant octobre 2021 ainsi qu'un regain de l'activité à partir de décembre 2022.

Code utilisé :

Entrée [10]:

```
BAYC = pd.read_csv("BAYC_3.csv", low_memory=False)

BAYC_Background = BAYC[['Background']]
BAYC_Clothes = BAYC[['Clothes']]
BAYC_Earring = BAYC[['Earring']]
BAYC_Eyes = BAYC[['Eyes']]
BAYC_Fur = BAYC[['Fur']]
BAYC_Hat = BAYC[['Hat']]
BAYC_Mouth = BAYC[['Mouth']]

BAYC_Background_count = BAYC_Background['Background'].value_counts()
BAYC_Clothes_count = BAYC_Clothes['Clothes'].value_counts()
BAYC_Earring_count = BAYC_Earring['Earring'].value_counts()
BAYC_Eyes_count = BAYC_Eyes['Eyes'].value_counts()
BAYC_Fur_count = BAYC_Fur['Fur'].value_counts()
BAYC_Hat_count = BAYC_Hat['Hat'].value_counts()
BAYC_Mouth_count = BAYC_Mouth['Mouth'].value_counts()

BAYC_Background_count.index = 'Background ' + BAYC_Background_count.index
BAYC_Clothes_count.index = 'Clothes ' + BAYC_Clothes_count.index
BAYC_Earring_count.index = 'Earring ' + BAYC_Earring_count.index
BAYC_Eyes_count.index = 'Eyes ' + BAYC_Eyes_count.index
BAYC_Fur_count.index = 'Fur ' + BAYC_Fur_count.index
BAYC_Hat_count.index = 'Hat ' + BAYC_Hat_count.index
BAYC_Mouth_count.index = 'Mouth ' + BAYC_Mouth_count.index

BAYC_count = pd.concat([BAYC_Background_count, BAYC_Clothes_count, BAYC_Earring_count, BAYC_Eyes_count,
BAYC_Fur_count, BAYC_Hat_count, BAYC_Mouth_count], axis=0).to_frame()

BAYC_unique = BAYC.drop_duplicates(subset='token_id', keep='first')

BAYC_unique_Background_count = BAYC_unique['Background'].value_counts()
BAYC_unique_Clothes_count = BAYC_unique['Clothes'].value_counts()
BAYC_unique_Earring_count = BAYC_unique['Earring'].value_counts()
BAYC_unique_Eyes_count = BAYC_unique['Eyes'].value_counts()
BAYC_unique_Fur_count = BAYC_unique['Fur'].value_counts()
BAYC_unique_Hat_count = BAYC_unique['Hat'].value_counts()
BAYC_unique_Mouth_count = BAYC_unique['Mouth'].value_counts()

BAYC_unique_Background_count.index = 'Background ' + BAYC_unique_Background_count.index
BAYC_unique_Clothes_count.index = 'Clothes ' + BAYC_unique_Clothes_count.index
BAYC_unique_Earring_count.index = 'Earring ' + BAYC_unique_Earring_count.index
BAYC_unique_Eyes_count.index = 'Eyes ' + BAYC_unique_Eyes_count.index
BAYC_unique_Fur_count.index = 'Fur ' + BAYC_unique_Fur_count.index
BAYC_unique_Hat_count.index = 'Hat ' + BAYC_unique_Hat_count.index
BAYC_unique_Mouth_count.index = 'Mouth ' + BAYC_unique_Mouth_count.index

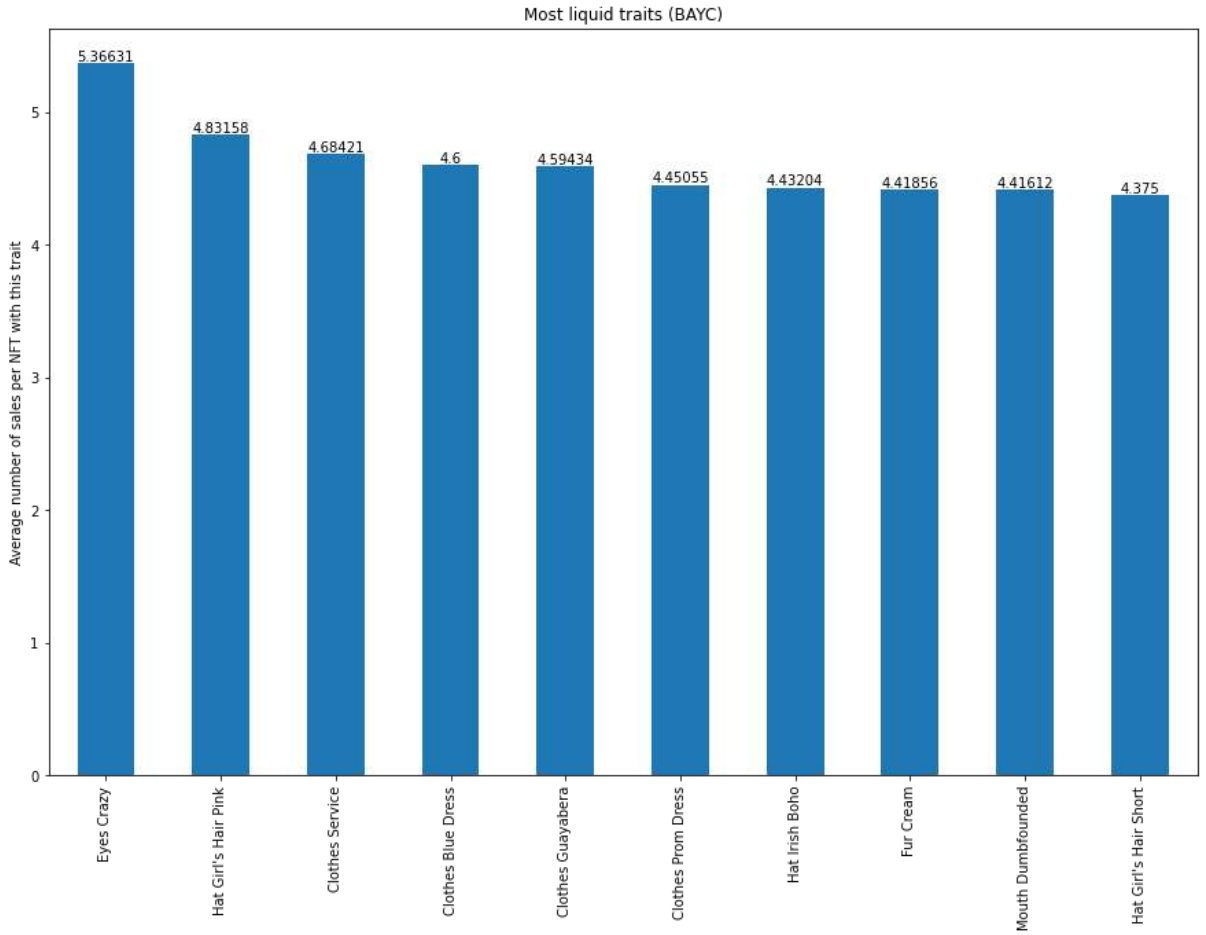
BAYC_unique_count = pd.concat([BAYC_unique_Background_count, BAYC_unique_Clothes_count, BAYC_unique_Earring_count,
BAYC_unique_Eyes_count, BAYC_unique_Fur_count, BAYC_unique_Hat_count, BAYC_unique_Mouth_count], axis=0).to_frame()

BAYC_liquid_traits = BAYC_count[0].sort_index() / BAYC_unique_count[0].sort_index()
BAYC_liquid_traits = BAYC_liquid_traits.sort_values(ascending=False)

ax = BAYC_liquid_traits[:10].plot(kind='bar', title='Most liquid traits (BAYC)', figsize=(15,10),
ylabel="Average number of sales per NFT with this trait", color='C0')

for i in ax.containers:
    ax.bar_label(i)
plt.show()
```

Résultat :




Interprétation :
Ce graphique permet d'obtenir les caractéristiques les plus liquides de la collection, c'est -à -dire les caractéristiques ayant le plus de vente. On observe que les caractéristiques les plus liquides sont celles les plus communes (les moins rares).

Code utilisé :

Entrée [80]:
```
fig, ax1 = plt.subplots(figsize=(15,10))
ax2 = ax1.twinx()

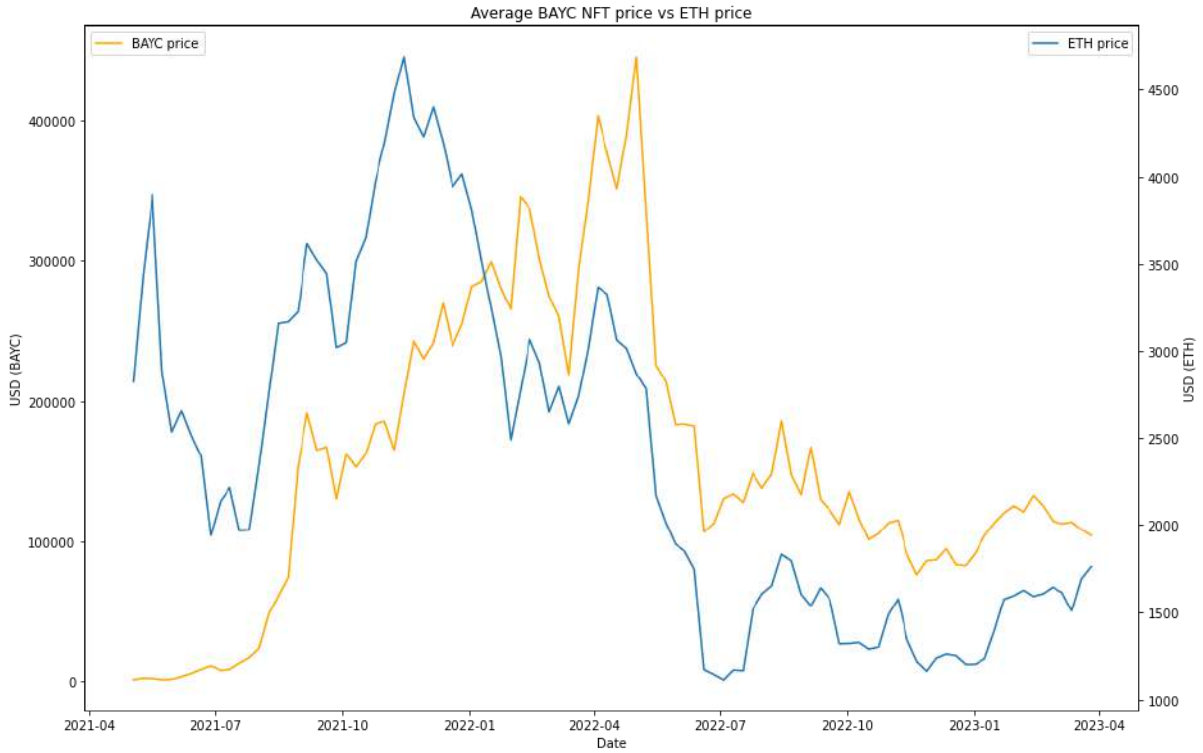
ax1.plot(BAYC_eth_2.index, BAYC_eth_2['usd_price'], label="BAYC price", color="orange")
ax2.plot(BAYC_eth_2.index, BAYC_eth_2['Open'], label='ETH price')

ax1.set_title('Average BAYC NFT price vs ETH price')
ax1.set_ylabel('USD (BAYC)')
ax1.set_xlabel('Date')
ax2.set_ylabel('USD (ETH)')
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

plt.show()
```

Résultat :

KAZMIERCZAK Elisa - POLLET Tom




Interprétation :
Ce graphique permet de se rendre compte de la corrélation positive entre le prix moyen de vente hebdomadaire des NFTs de la collection et le prix moyen hebdomadaire d'ETH.

Code utilisé :

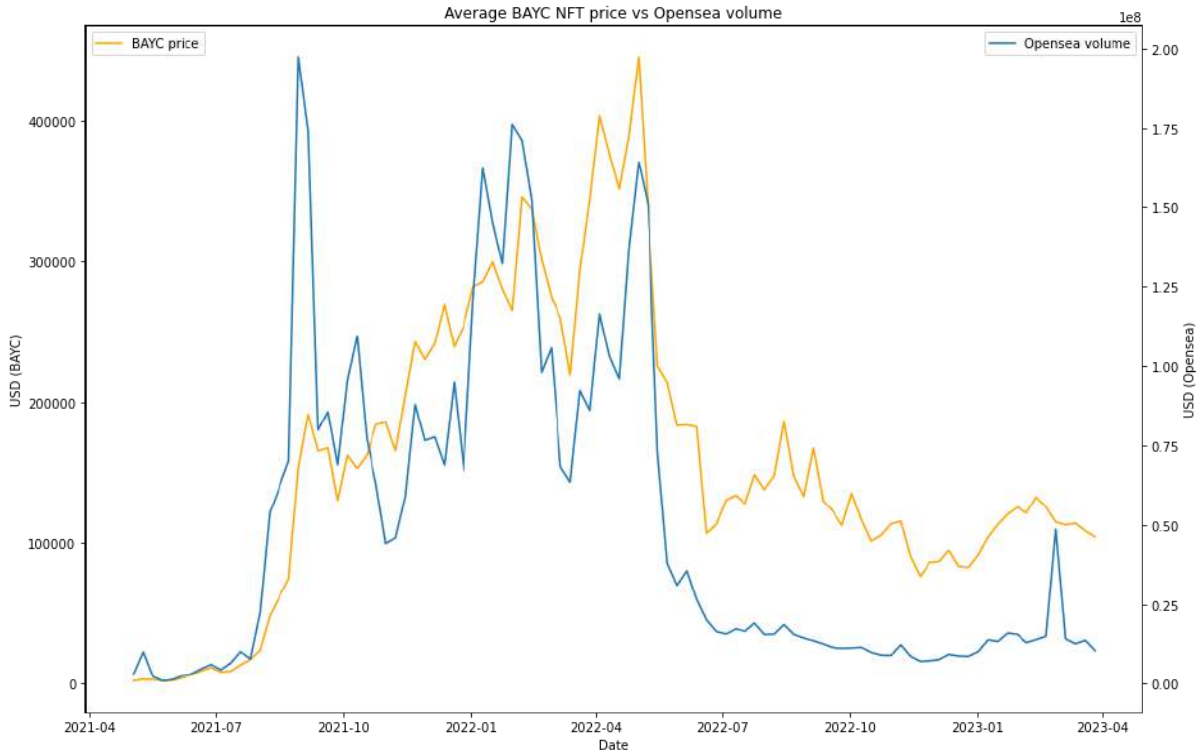
Entrée [78]:
```
fig, ax1 = plt.subplots(figsize=(15,10))
ax2 = ax1.twinx()

ax1.plot(BAYC_eth_2.index, BAYC_eth_2['usd_price'], label="BAYC price", color="orange")
ax2.plot(BAYC_eth_2.index, BAYC_eth_2['vol_usd'], label='Opensea volume')

ax1.set_title('Average BAYC NFT price vs Opensea volume')
ax1.set_ylabel('USD (BAYC)')
ax1.set_xlabel('Date')
ax2.set_ylabel('USD (Opensea)')
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

plt.show()
```

Résultat :




Interprétation :
Ce graphique permet de se rendre compte de la corrélation positive entre le prix moyen de vente hebdomadaire des NFTs de la collection et le volume hebdomadaire des ventes de NFTs sur la plateforme de vente de NFTs (donc le volume moyen de vente de toutes les collections de NFTs).

# 9. Machine learning avec Python

Fort de nos précédentes analyses sur R et Python, nous pouvons désormais construire un modèle de machine learning permettant de prédire le prix d'un NFTs en se basant sur ses

caractéristiques (dont nous avons vu l'impact sur le prix de vente) ainsi que sur le volume des ventes de NFTs sur la plateforme Opensea (dont nous avons constater la corrélation avec le prix de vente).

# 1. Entraînement du modèle

Code utilisé :

Entrée [5]:
```
BAYC = pd.read_csv("BAYC_3.csv", low_memory=False)

BAYC_2 = BAYC.loc[BAYC['usd_price'] > 100]
BAYC_2 = BAYC_2.reset_index(drop=True)

traits =  BAYC_2[["Background", "Clothes", "Earring", "Eyes", "Fur", "Hat", "Mouth"]]

one_hot = pd.get_dummies(traits, columns=["Background", "Clothes", "Earring", "Eyes", "Fur", "Hat", "Mouth"])

dates = BAYC_2[["timestamp"]]
dates['Date'] = pd.to_datetime(dates['timestamp'], format='%d/%m/%Y %H:%M')
dates['Date'] = dates['Date'].dt.strftime('%d/%m/%Y')
dates = dates[["Date"]]

volume = pd.read_csv('opensea_volume.csv')
volume['time'] = pd.to_datetime(volume['time'], format='%Y-%m-%d %H:%M:%S.%f UTC')
volume['Date'] = volume['time'].dt.strftime('%d/%m/%Y')
vol_dates = pd.merge(dates, volume, on='Date', how='left')
vol_dates = vol_dates[["Date", 'vol_usd']]

X = pd.concat([vol_dates[["vol_usd"]], one_hot], axis=1)
Y = BAYC_2[["usd_price"]]
Y = Y.to_numpy().ravel()

X_train, X_test, y_train, y_test = train_test_split(X, Y,test_size=.01,random_state =11)

model = RandomForestRegressor()
model.fit(X_train, y_train)
```

Résultat :

```
Out[5]: RandomForestRegressor()
```

# 2. Evaluation du modèle

Interprétation :
Grâce à ce script, nous avons entraîné notre modèle de Random Forest Regressor à prédire le prix de vente en se basant sur les caractéristiques des NFTs et sur le volume d'échange de la plateforme Opensea. Nous avons dû au préalable encoder les données dans le bon format à l'aide de l'encodage "one hot" permettant de passer de variables quantitatives à qualitatives et ainsi d'entraîner correctement le modèle.

Code utilisé :

Entrée [6]:
```
from sklearn.metrics import mean_absolute_error

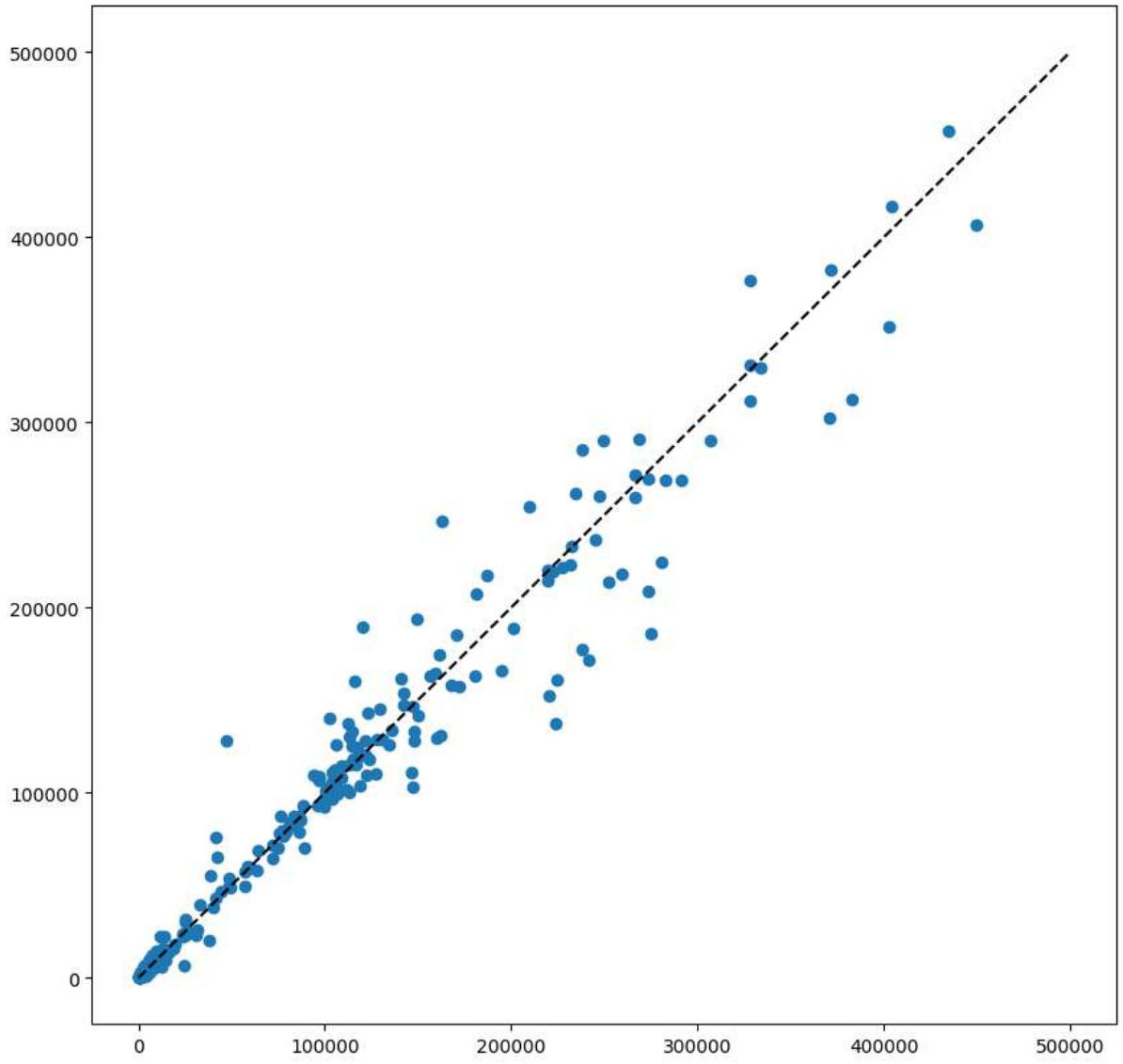
R_squared = model.score(X_test, y_test)
mae = mean_absolute_error(y_test, model.predict(X_test))
mape = np.mean(np.abs(y_test - model.predict(X_test)) / y_test) * 100

print(f"R²: {R_squared:.2f}")
print(f"Mean Absolute Error: {mae:.2f}")
print(f"Mean Absolute Percentage Error: {mape:.2f}%")
```

```
Entrée [7]:  plt.figure(figsize=(10,10))
             plt.plot(y_test, model.predict(X_test), 'o')
             plt.plot([0, 500000], [0, 500000], 'k--')
             plt.show()
```

Résultat :

```
R²: 0.96
Mean Absolute Error: 8440.87
Mean Absolute Percentage Error: 19.92%
```




Interprétation :

Nous observons ici que notre modèle prédit très bien les prix de vente des NFTs de la collection Bored Ape Yacht Club. En effet, les 3 métriques $R^2$, erreur moyenne absolue moyenne (MAE) et erreur moyenne absolue en pourcentage (MAPE) montrent que notre

modèle est très bon. Le $R^2$ de 0.96 nous indique que le modèle explique 96% de la variance des données ce qui est une très bonne performance. Le MAE de 8440,87 USD peut sembler être une erreur élevée, mais nous devons nous rappeler que le prix de vente moyen des NFTs de la collection est de 83 641 dollars, l'erreur n'est donc pas négligeable, mais pas mauvaise non plus. Le MAPE de 19,92% signifie qu'en moyenne, les prédictions du modèle se situent à 19,92% des valeurs réelles, ce qui est correct pour nos données qui présentent une grande variabilité. Enfin, le dernier graphique nous montre que le modèle est proche d'un modèle parfait, en effet, les couples (prix de vente réel, prix de vente prédit par le modèle) sont proches de la droite d'équation y=x (les points se situeraient tous sur cette droite si le modèle était parfait).