Dimension Reduction and Clustering on the Steam Games Dataset

Elizaveta Sokolova

Contents

Dataset

The dataset used in this project is the Steam Games Dataset by FronkonGames, published under a CC-BY-4.0 license. It contains information on over 110,000 games and was collected automatically using the Steam-Games-Scraper — a tool that queries the Steam Web API to retrieve the full app list and then fetches detailed metadata for each title, filtering out DLCs, soundtracks, and utilities so that only actual games are included. The resulting fields cover a wide range of signals: pricing, platform support, review counts, playtime statistics, peak concurrent users, and estimated ownership ranges.

Libraries

```
# Install (run once if packages are missing):
# install.packages(c("FactoMineR","factoextra","Rtsne","umap","psych",
#                    "ggplot2","dplyr","corrplot","gridExtra","RColorBrewer","knitr"))
library(FactoMineR)     # PCA engine (SVD-based)
library(factoextra)     # Scree, biplot, contrib plots (ggplot2-based)
library(psych)          # KMO & Bartlett tests
library(Rtsne)          # t-SNE
library(umap)           # UMAP (pure-R naive method - no Python needed)
library(ggplot2)
library(dplyr)
```

```r
library(corrplot)
library(gridExtra)
library(grid)           # textGrob, gpar (used in grid.arrange titles)
library(RColorBrewer)
library(knitr)
```

Data Loading & Column Diagnosis

```r
df <- read.csv("games.csv", stringsAsFactors = FALSE)

cat("Dimensions:", nrow(df), "rows  ×", ncol(df), "columns\n\n")
```

```
## Dimensions: 122611 rows  × 39 columns
```

```r
cat("Column names as read from file:\n")
```

```
## Column names as read from file:
```

```r
print(summary(df))
```

```
##      AppID               Name           Release.date       Estimated.owners
##  Length:122611      Length:122611      Length:122611      Min.   :0.000e+00
##  Class :character   Class :character   Class :character   1st Qu.:0.000e+00
##  Mode  :character   Mode  :character   Mode  :character   Median :0.000e+00
##                                                           Mean   :5.459e+01
##                                                           3rd Qu.:0.000e+00
##                                                           Max.   :1.014e+06
##
##     Peak.CCU        Required.age          Price        DiscountDLC.count
##  Min.   : 0.0000   Min.   :   0.000   Min.   :  0.00   Min.   :   0.0000
##  1st Qu.: 0.0000   1st Qu.:   0.550   1st Qu.:  0.00   1st Qu.:   0.0000
##  Median : 0.0000   Median :   2.240   Median :  0.00   Median :   0.0000
##  Mean   : 0.1676   Mean   :   4.765   Mean   : 18.35   Mean   :   0.5459
##  3rd Qu.: 0.0000   3rd Qu.:   5.240   3rd Qu.: 40.00   3rd Qu.:   0.0000
##  Max.   :21.0000   Max.   : 999.980   Max.   :100.00   Max.   :3703.0000
##
##  About.the.game     Supported.languages Full.audio.languages   Reviews
##  Length:122611      Length:122611       Length:122611        Length:122611
##  Class :character   Class :character    Class :character     Class :character
##  Mode  :character   Mode  :character    Mode  :character     Mode  :character
##
##
##
##
##  Header.image         Website           Support.url        Support.email
##  Length:122611      Length:122611      Length:122611      Length:122611
##  Class :character   Class :character   Class :character   Class :character
```

```
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    Windows              Mac               Linux           Metacritic.score
##  Length:122611      Length:122611      Length:122611      Min.   : 0.000
##  Class :character   Class :character   Class :character   1st Qu.: 0.000
##  Mode  :character   Mode  :character   Mode  :character   Median : 0.000
##                                                           Mean   : 2.565
##                                                           3rd Qu.: 0.000
##                                                           Max.   :97.000
##
##  Metacritic.url       User.score           Positive          Negative
##  Length:122611      Min.   :  0.00000   Min.   :      0   Min.   :      0.0
##  Class :character   1st Qu.:  0.00000   1st Qu.:      0   1st Qu.:      0.0
##  Mode  :character   Median :  0.00000   Median :      5   Median :      1.0
##                     Mean   :  0.02455   Mean   :   1045   Mean   :    169.2
##                     3rd Qu.:  0.00000   3rd Qu.:     37   3rd Qu.:     10.0
##                     Max.   :100.00000   Max.   :7642084   Max.   :1173003.0
##
##    Score.rank      Achievements     Recommendations        Notes
##  Min.   : 98.00   Min.   :   0.00   Min.   :      0.0   Length:122611
##  1st Qu.: 99.00   1st Qu.:   0.00   1st Qu.:      0.0   Class :character
##  Median : 99.00   Median :   2.00   Median :      0.0   Mode  :character
##  Mean   : 99.17   Mean   :  18.09   Mean   :    961.8
##  3rd Qu.:100.00   3rd Qu.:  19.00   3rd Qu.:      0.0
##  Max.   :100.00   Max.   :9821.00   Max.   :4830455.0
##  NA's   :122571
##  Average.playtime.forever Average.playtime.two.weeks Median.playtime.forever
##  Min.   :      0          Min.   :    0.00           Min.   :      0.0
##  1st Qu.:      0          1st Qu.:    0.00           1st Qu.:      0.0
##  Median :      0          Median :    0.00           Median :      0.0
##  Mean   :    208          Mean   :   13.79           Mean   :    173.6
##  3rd Qu.:      0          3rd Qu.:    0.00           3rd Qu.:      0.0
##  Max.   :3429544          Max.   :20088.00           Max.   :3429544.0
##
##  Median.playtime.two.weeks  Developers         Publishers
##  Min.   :    0.00          Length:122611      Length:122611
##  1st Qu.:    0.00          Class :character   Class :character
##  Median :    0.00          Mode  :character   Mode  :character
##  Mean   :   14.72
##  3rd Qu.:    0.00
##  Max.   :20088.00
##
##   Categories           Genres              Tags           Screenshots
##  Length:122611      Length:122611      Length:122611      Length:122611
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   Movies
```

```
##  Mode:logical
##  NA's:122611
##
##
##
##
##
```

Renaming

```
# Rename the first 8 columns to match what the data truly contains
colnames(df) <- gsub("\\.", "_", colnames(df))
colnames(df)
```

```
##  [1] "AppID"                      "Name"
##  [3] "Release_date"               "Estimated_owners"
##  [5] "Peak_CCU"                   "Required_age"
##  [7] "Price"                      "DiscountDLC_count"
##  [9] "About_the_game"             "Supported_languages"
## [11] "Full_audio_languages"       "Reviews"
## [13] "Header_image"               "Website"
## [15] "Support_url"                "Support_email"
## [17] "Windows"                    "Mac"
## [19] "Linux"                      "Metacritic_score"
## [21] "Metacritic_url"             "User_score"
## [23] "Positive"                   "Negative"
## [25] "Score_rank"                 "Achievements"
## [27] "Recommendations"            "Notes"
## [29] "Average_playtime_forever"   "Average_playtime_two_weeks"
## [31] "Median_playtime_forever"    "Median_playtime_two_weeks"
## [33] "Developers"                 "Publishers"
## [35] "Categories"                 "Genres"
## [37] "Tags"                       "Screenshots"
## [39] "Movies"
```

Data Cleaning & Feature Engineering

Parse string columns

```
# Estimated_owners: "20000 - 50000"  →  midpoint 35000
parse_owners <- function(x) {
  x <- gsub(",", "", as.character(x))

  parts <- strsplit(x, "\\s*[---]\\s*")

  sapply(parts, function(p) {
    nums <- suppressWarnings(as.numeric(p))
    #                   ,
```

```
    if (any(!is.na(nums))) mean(nums, na.rm = TRUE) else NA_real_
  }, USE.NAMES = FALSE)
}


df$Owners_numeric <- parse_owners(df$Estimated_owners)

cat("Owners_numeric - sample values:\n")
```

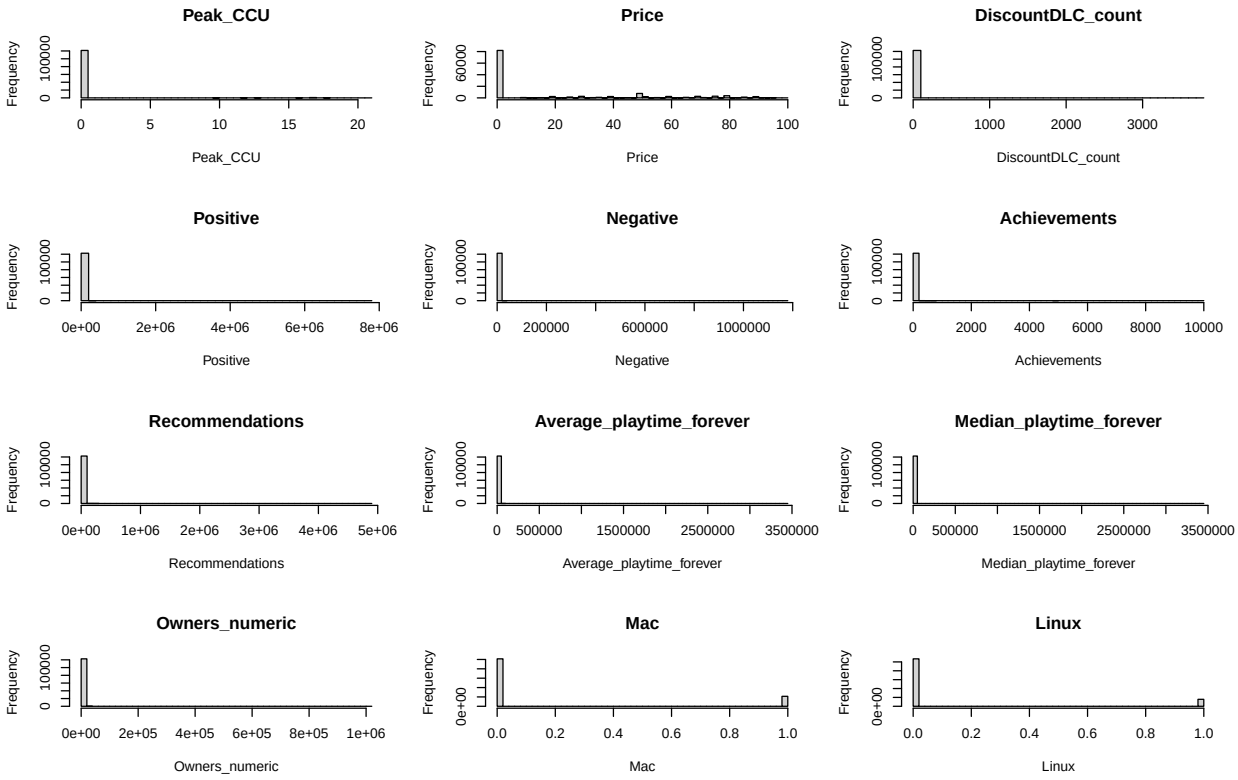
```
## Owners_numeric - sample values:
```

```
df$Owners_numeric <- parse_owners(df$Estimated_owners)
head(df$Owners_numeric, 10)
```

```
##  [1] 0 0 0 1 0 0 8 0 0 1
```

Filter & group genres

```
par(mfrow = c(4, 3))
keep_cols <- c("Peak_CCU", "Price", "DiscountDLC_count",
               "Positive", "Negative", "Achievements", "Recommendations",
               "Average_playtime_forever", "Median_playtime_forever",
               "Owners_numeric", "Mac", "Linux")

feat <- df[, keep_cols]

#   Binary platform flags
feat$Mac   <- as.integer(as.logical(feat$Mac))
feat$Linux <- as.integer(as.logical(feat$Linux))

for(col in keep_cols) {
  hist(feat[[col]], breaks = 50, main = col, xlab = col)
}
```



Feature Selection & Log-Transformation

Why these features?

Feature	Transform	Rationale
Peak CCU	log	Peak concurrent users — extreme right skew
Price	none	Pricing; mild skew
DLC Count	log	Ecosystem depth
Positive Reviews	log	Core engagement signal
Negative Reviews	log	Quality / controversy
Achievements	log	Game-design depth
Recommendations	log	Community endorsement
Avg Playtime	log	Player retention
Med Playtime	log	Retention (outlier-robust)
Est. Owners	log	Market reach
Mac	none	Binary platform flag
Linux	none	Binary platform flag

Dropped: * `Required_age` — 98.6 % zeros (near-zero variance). * `Metacritic score` — 94.7 % zeros. * `Windows` — TRUE for 100 % of games (zero variance).

```r
#   log  on all skewed continuous columns
skewed <- c("Peak_CCU", "Positive", "Negative", "Recommendations",
            "Average_playtime_forever", "Median_playtime_forever",
            "Owners_numeric", "DiscountDLC_count", "Achievements")

feat[, skewed] <- log1p(feat[, skewed])

#   Clean column names for nicer plots
colnames(feat) <- c("Peak CCU", "Price", "DLC Count",
                    "Positive", "Negative", "Achievements", "Recommendations",
                    "Avg Playtime", "Med Playtime", "Est. Owners",
                    "Mac", "Linux")

cat("Final feature matrix:", nrow(feat), "games  ×", ncol(feat), "features\n\n")
```

```
## Final feature matrix: 122611 games  × 12 features
```

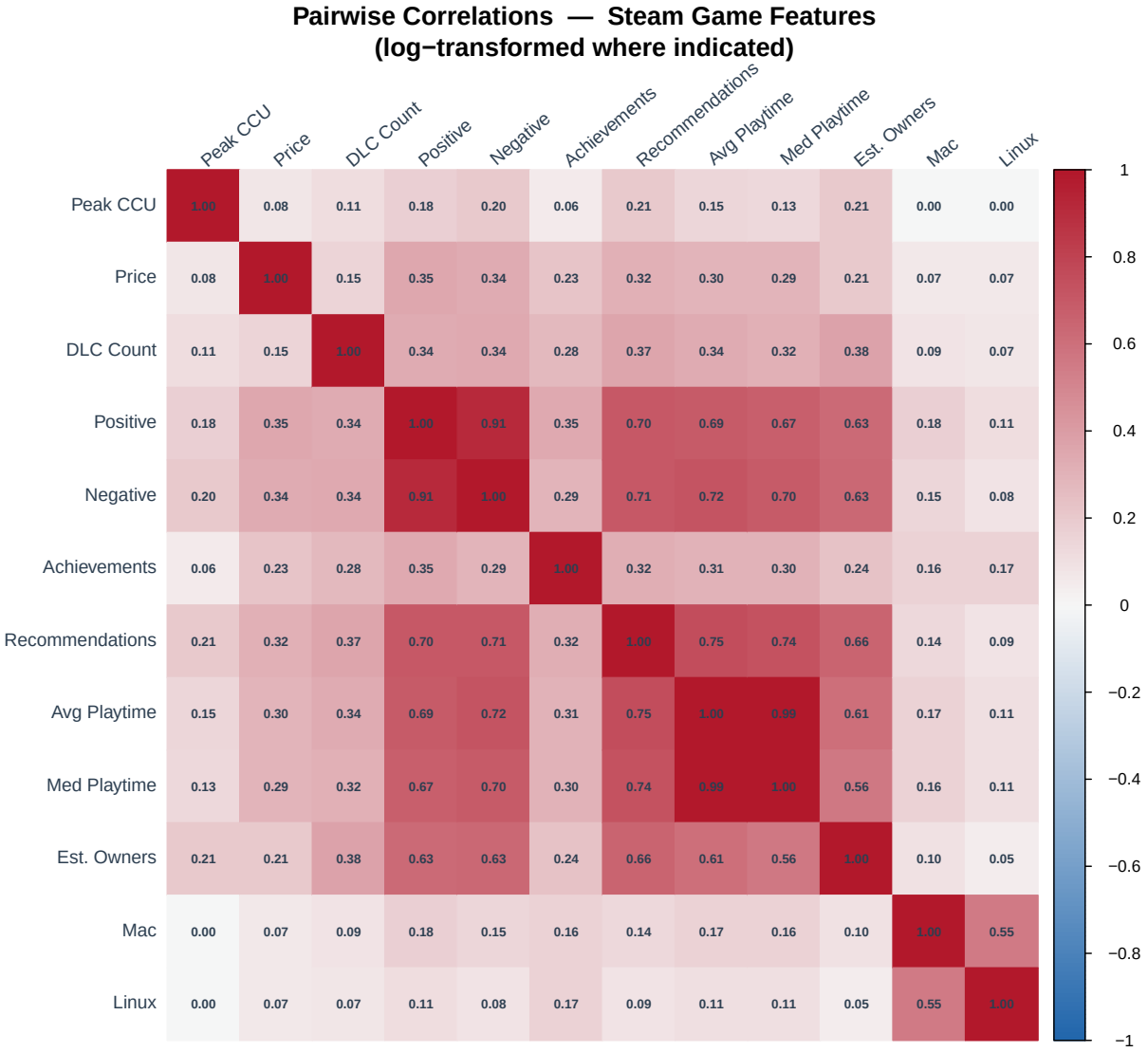
```r
print(summary(feat))
```

```
##     Peak CCU           Price          DLC Count         Positive
##  Min.   :0.00000   Min.   :  0.00   Min.   :0.0000   Min.   : 0.000
##  1st Qu.:0.00000   1st Qu.:  0.00   1st Qu.:0.0000   1st Qu.: 0.000
##  Median :0.00000   Median :  0.00   Median :0.0000   Median : 1.792
##  Mean   :0.02935   Mean   : 18.35   Mean   :0.1485   Mean   : 2.266
##  3rd Qu.:0.00000   3rd Qu.: 40.00   3rd Qu.:0.0000   3rd Qu.: 3.638
##  Max.   :3.09104   Max.   :100.00   Max.   :8.2172   Max.   :15.849
##     Negative        Achievements   Recommendations   Avg Playtime
##  Min.   : 0.0000   Min.   :0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.0000   1st Qu.:0.000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 0.6931   Median :1.099   Median : 0.000   Median : 0.000
##  Mean   : 1.4764   Mean   :1.512   Mean   : 1.099   Mean   : 1.121
##  3rd Qu.: 2.3979   3rd Qu.:2.996   3rd Qu.: 0.000   3rd Qu.: 0.000
##  Max.   :13.9751   Max.   :9.192   Max.   :15.390   Max.   :15.048
##   Med Playtime     Est. Owners          Mac              Linux
##  Min.   : 0.000   Min.   : 0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 0.000   Median : 0.0000   Median :0.0000   Median :0.0000
##  Mean   : 1.093   Mean   : 0.3333   Mean   :0.1737   Mean   :0.1281
##  3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :15.048   Max.   :13.8294   Max.   :1.0000   Max.   :1.0000
```

Correlation Matrix

```r
cor_mat <- cor(feat, use = "pairwise.complete.obs")

corrplot(cor_mat,
         method      = "color",
         col         = colorRampPalette(c("#2166ac", "#f7f7f7", "#b2182b"))(200),
```

```
        tl.cex      = 0.82,
        tl.col      = "#2c3e50",
        tl.srt      = 40,
        addCoef.col = "#2c3e50",
        number.cex  = 0.60,
        number.digits = 2,
        cl.pos      = "r",
        cl.cex      = 0.72,
        title       = "Pairwise Correlations  -  Steam Game Features\n(log-transformed where indicated)",
        mar         = c(0, 0, 2, 0))
```



Several feature groups are strongly correlated — *Positive / Recommendations / Est. Owners / Avg & Med Playtime / Peak CCU* form one tight block; *Avg Playtime* and *Med Playtime* are near-duplicates. This redundancy is exactly what PCA is designed to compress.

PCA Validity Test

Before running PCA we should verify that the correlation structure in the data is actually suitable for factor extraction. Two standard tests exist for this:

- **Kaiser-Meyer-Olkin (KMO)** — measures *sampling adequacy*. It checks whether the pairwise correlations are large relative to the partial correlations (i.e., whether the variables share enough common variance to justify reducing them). KMO > 0.6 is acceptable; > 0.8 is good; > 0.9 is excellent.

```
# KMO needs the raw data (not a cor matrix) to compute per-variable MSA.
kmo_res     <- KMO(feat)
print(kmo_res)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = feat)
## Overall MSA =  0.81
## MSA for each item =
##        Peak CCU           Price       DLC Count        Positive        Negative
##            0.91            0.96            0.94            0.82            0.83
##    Achievements Recommendations    Avg Playtime    Med Playtime     Est. Owners
##            0.89            0.96            0.74            0.73            0.85
##             Mac           Linux
##            0.62            0.57
```

PCA — Principal Component Analysis

Run PCA

```
# FactoMineR's PCA() centres and scales every variable to unit variance automatically.
res.pca <- PCA(feat, graph = FALSE, scale.unit = TRUE)

# Eigenvalue table - as.data.frame so that $ column access works
eigen_tbl <- as.data.frame(get_eigenvalue(res.pca))

cat("Eigenvalues & explained variance (all components):\n\n")
```

```
## Eigenvalues & explained variance (all components):
```

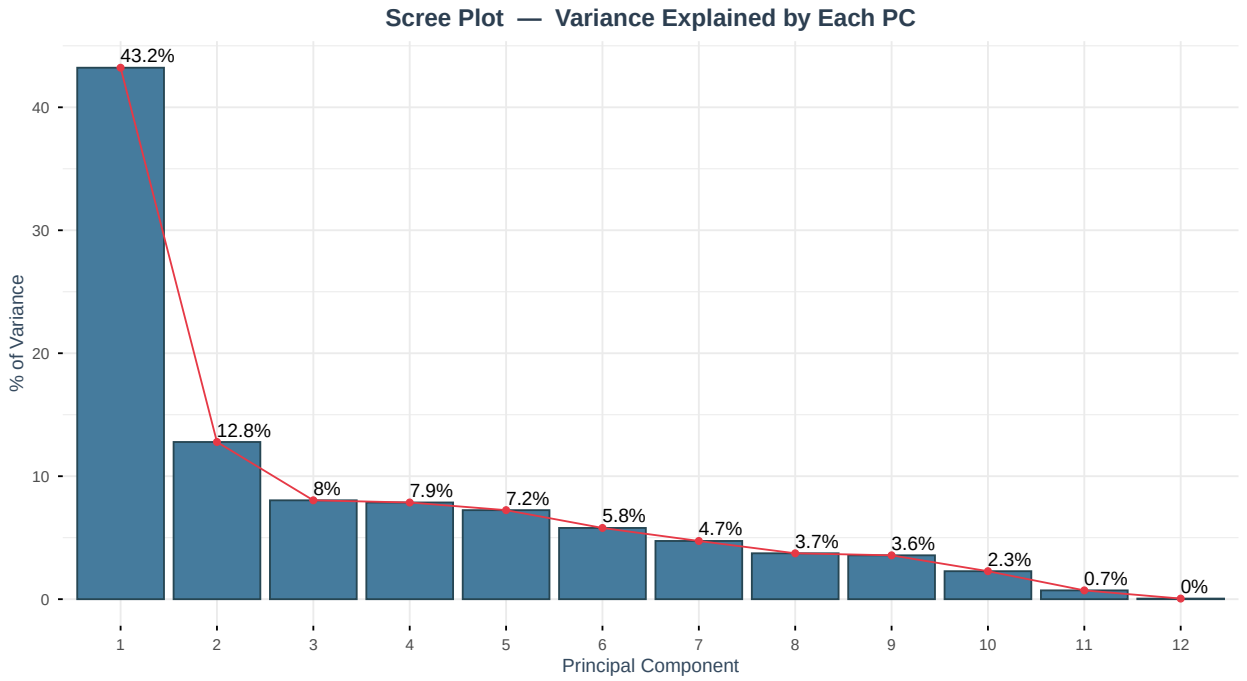
```
print(round(eigen_tbl, 3))
```

```
##        eigenvalue variance.percent cumulative.variance.percent
## Dim.1       5.186           43.217                       43.217
## Dim.2       1.534           12.782                       56.000
## Dim.3       0.965            8.039                       64.039
## Dim.4       0.943            7.862                       71.901
## Dim.5       0.869            7.239                       79.141
## Dim.6       0.695            5.792                       84.932
```

```
## Dim.7       0.568           4.733                  89.666
## Dim.8       0.448           3.731                  93.397
## Dim.9       0.428           3.566                  96.962
## Dim.10      0.273           2.276                  99.238
## Dim.11      0.086           0.714                  99.952
## Dim.12      0.006           0.048                 100.000
```

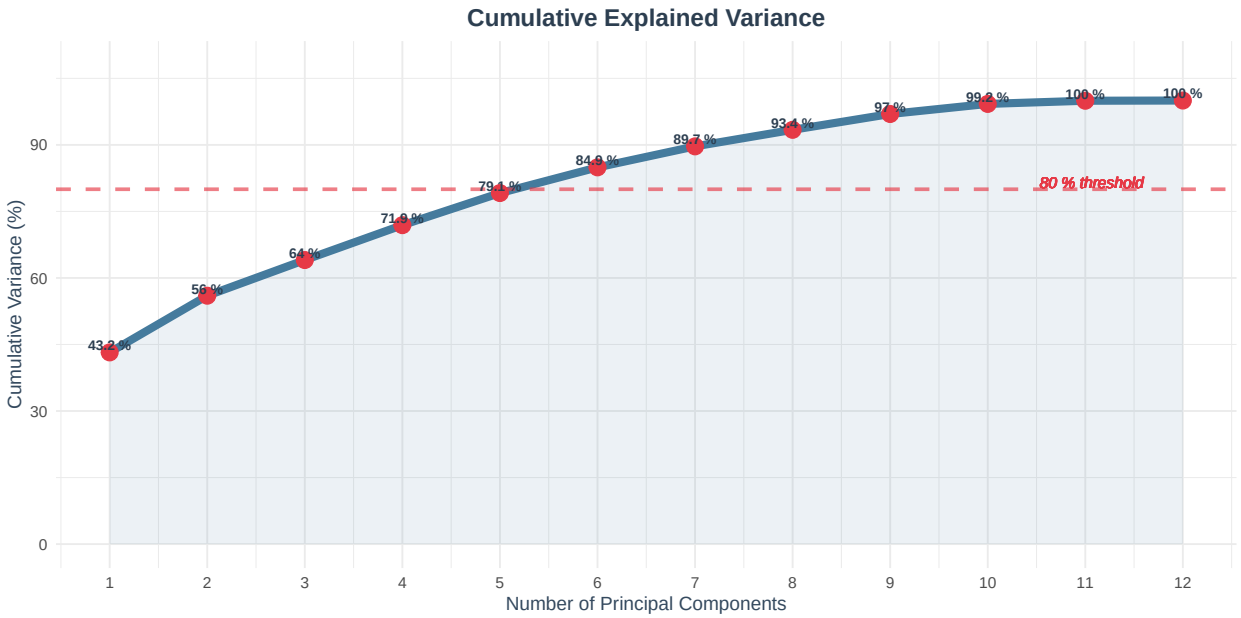
Scree Plot

```
fviz_eig(res.pca,
         addlabels  = TRUE,
         ncp        = ncol(feat),
         barfill    = "#457b9d",
         barcolor   = "#264653",
         linecolor  = "#e63946",
         pointcolor = "#e63946",
         pointsize  = 3.8,
         ggtheme    = theme_minimal()) +
  labs(title = "Scree Plot  -  Variance Explained by Each PC",
       x     = "Principal Component",
       y     = "% of Variance") +
  theme(plot.title = element_text(face  = "bold", size  = 14,
                                  hjust = 0.5,  color = "#2c3e50"),
        axis.title = element_text(size = 11, color = "#34495e"))
```



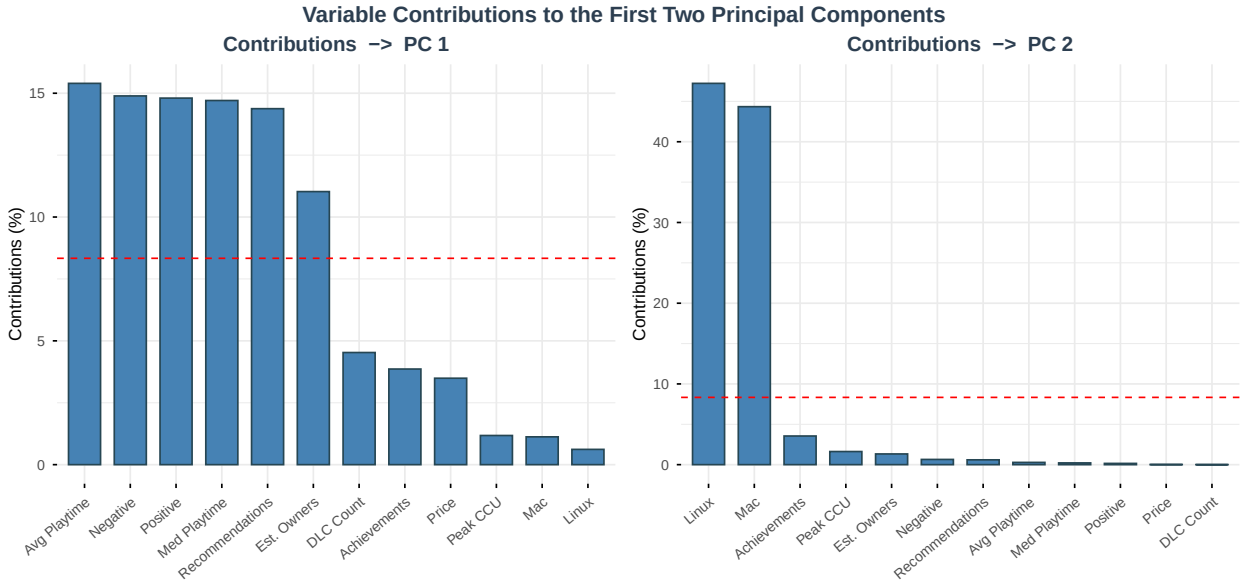
Cumulative Explained Variance

```r
cum_var <- eigen_tbl[, 3]   # col 3 = cumulative percentage of variance
n_pc    <- length(cum_var)

ggplot(data.frame(PC = 1:n_pc, CumVar = cum_var)) +
  geom_area(aes(x = PC, y = CumVar), fill = "#457b9d", alpha = 0.10) +
  geom_line(aes(x = PC, y = CumVar), color = "#457b9d", linewidth = 2.2) +
  geom_point(aes(x = PC, y = CumVar), color = "#e63946", size = 4.2) +
  # 80 % threshold
  geom_hline(yintercept = 80, linetype = "dashed",
             color = "#e63946", linewidth = 1, alpha = 0.65) +
  geom_text(x = n_pc - 0.4, y = 81.5,
            label = "80 % threshold", hjust = 1,
            color = "#e63946", size = 3.1, fontface = "italic") +
  # value labels
  geom_text(aes(x = PC, y = CumVar + 1.6),
            label = paste0(round(cum_var, 1), " %"),
            hjust = 0.5, size = 2.8, color = "#2c3e50", fontface = "bold") +
  scale_x_continuous(breaks = 1:n_pc) +
  scale_y_continuous(limits = c(0, 108)) +
  labs(title = "Cumulative Explained Variance",
       x     = "Number of Principal Components",
       y     = "Cumulative Variance (%)") +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14,
                                  hjust = 0.5, color = "#2c3e50"),
        axis.title = element_text(size = 11, color = "#34495e"))
```



Variable Contributions to PC1 & PC2

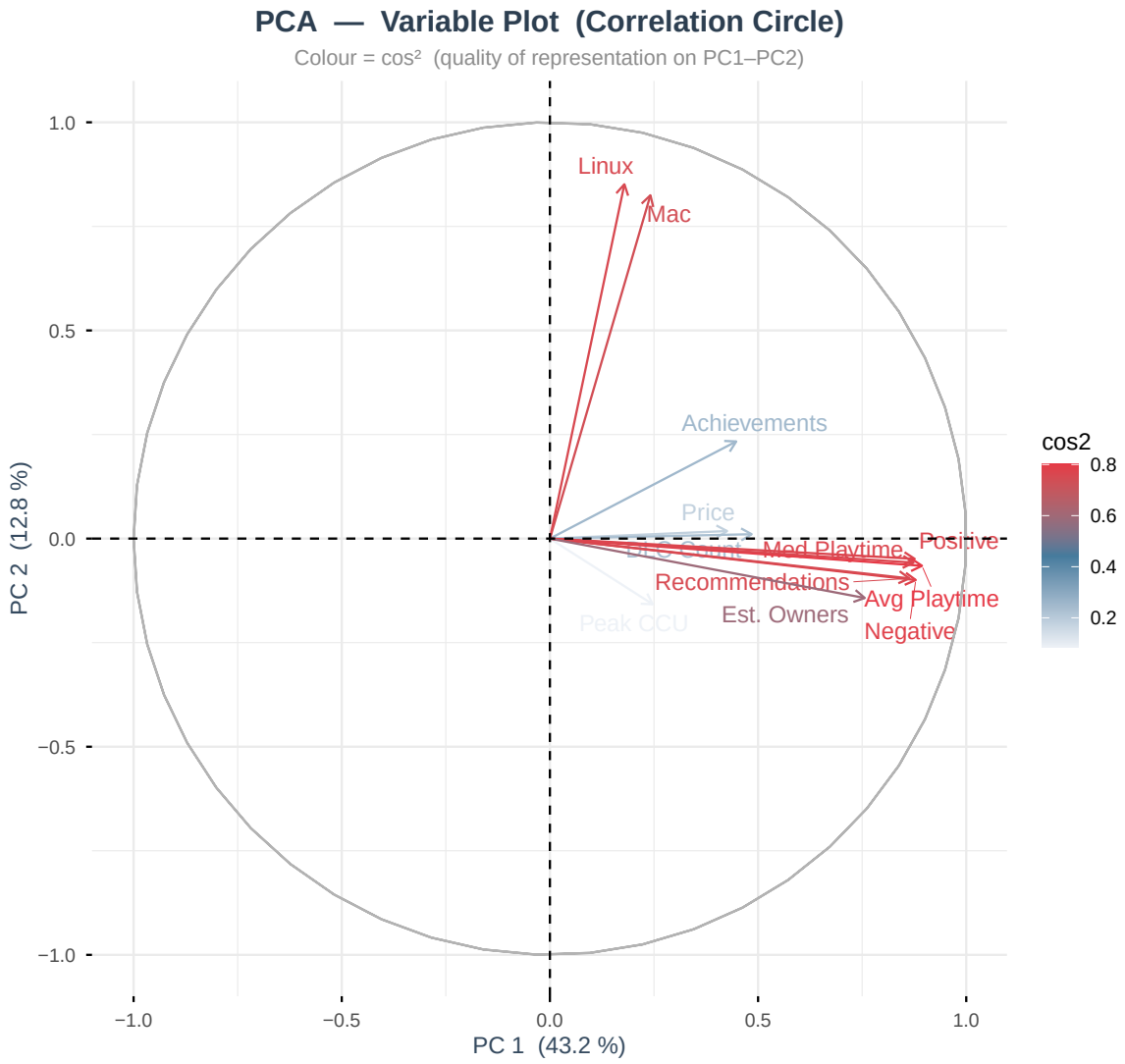
```r
p1 <- fviz_contrib(res.pca, choice = "var", axes = 1,
                   fill.color = "#457b9d", color = "#264653") +
  labs(title = "Contributions  →  PC 1") +
  theme(axis.text.x = element_text(angle = 40, hjust = 1, size = 9),
        plot.title   = element_text(face = "bold", size = 13,
                                    hjust = 0.5, color = "#2c3e50"))

p2 <- fviz_contrib(res.pca, choice = "var", axes = 2,
                   fill.color = "#2a9d8f", color = "#264653") +
  labs(title = "Contributions  →  PC 2") +
  theme(axis.text.x = element_text(angle = 40, hjust = 1, size = 9),
        plot.title   = element_text(face = "bold", size = 13,
                                    hjust = 0.5, color = "#2c3e50"))

grid.arrange(p1, p2, ncol = 2,
             top = textGrob("Variable Contributions to the First Two Principal Components",
                            gp = gpar(fontsize = 14, fontface = "bold", col = "#2c3e50")))
```



Variable Plot (Correlation Circle)

```r
fviz_pca_var(res.pca,
             col.var       = "cos2",
             gradient.cols = list("white" = "#eef2f7",
                                  "blue"  = "#457b9d",
                                  "red"   = "#e63946"),
             repel         = TRUE,
             geom.var      = c("arrow", "text"),
             ggtheme       = theme_minimal(),
             legend.position = "bottom") +
  labs(title    = "PCA  -  Variable Plot  (Correlation Circle)",
       subtitle = "Colour = cos²  (quality of representation on PC1-PC2)",
```

```
      x = paste0("PC 1  (", round(eigen_tbl[1, 2], 1), " %)"),
      y = paste0("PC 2  (", round(eigen_tbl[2, 2], 1), " %)")) +
 theme(plot.title    = element_text(face = "bold", size = 14,
                                    hjust = 0.5, color = "#2c3e50"),
       plot.subtitle = element_text(size = 10, color = "grey55", hjust = 0.5),
       axis.title    = element_text(size = 11, color = "#34495e"))
```

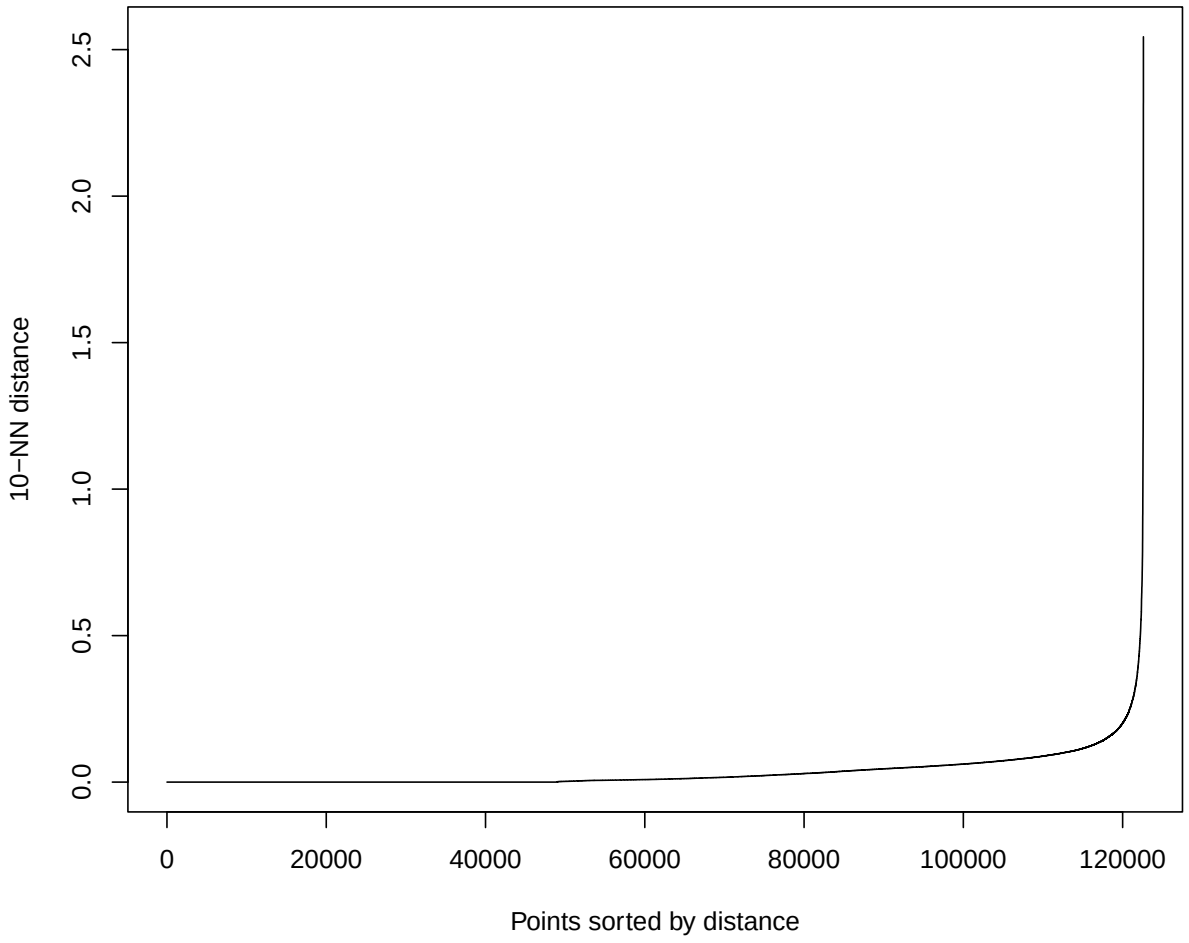


Clustering

After considering K-means, hierarchical, and DBSCAN clustering, DBSCAN was chosen as the final method due to its ability to handle skewed engagement metrics and uneven density distribution without assuming spherical cluster shapes.

```
library(dbscan)
pca_scores <- as.data.frame(res.pca$ind$coord)

# Lets take first 3 components
```

```r
pca_sub <- scale(pca_scores[,1:3])

kNNdistplot(pca_sub, k = 10)
```

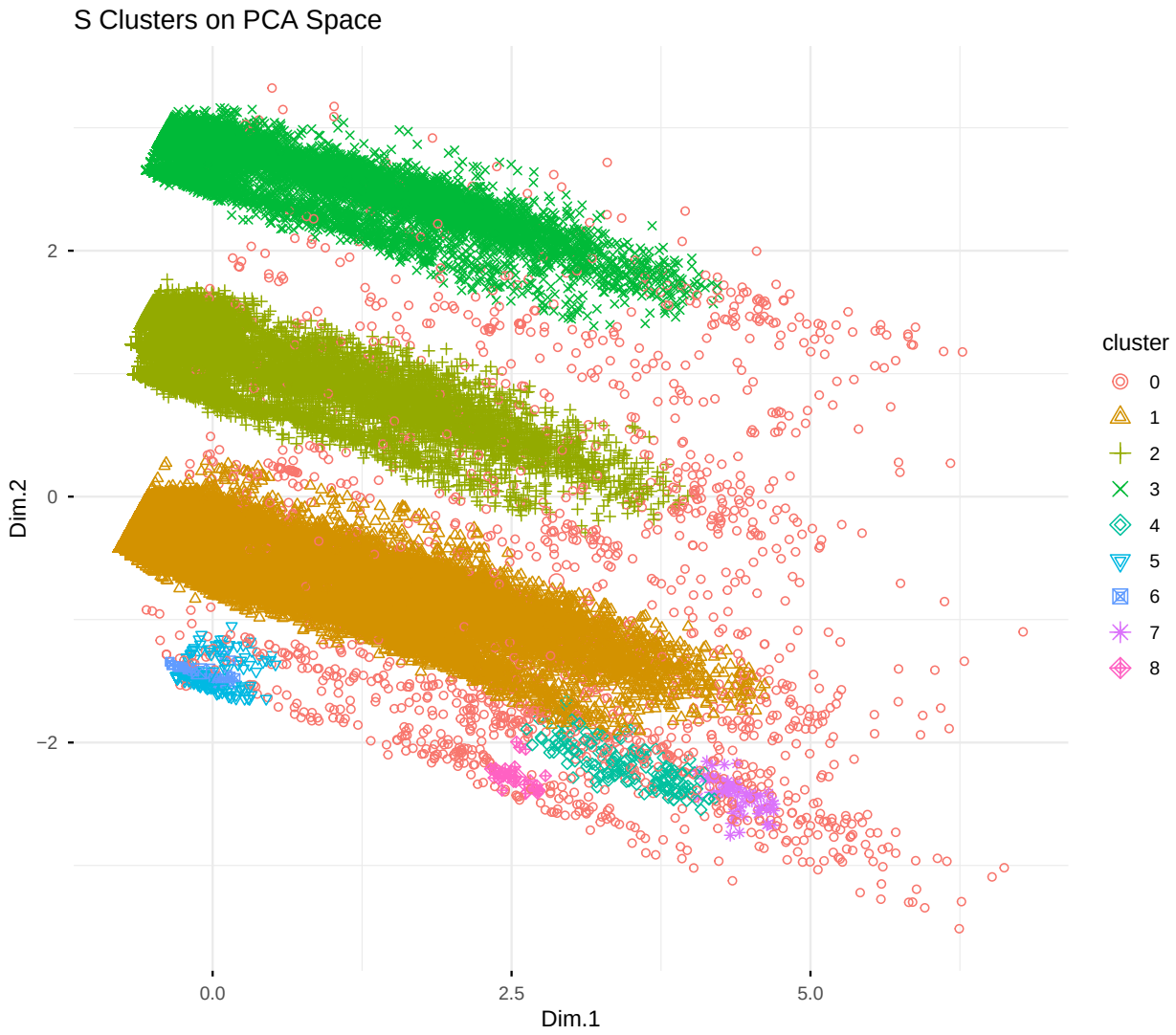



```r
db <- dbscan(pca_sub, eps = 0.28, minPts = 30)

df$Cluster_DBSCAN <- factor(db$cluster)

table(df$Cluster_DBSCAN)
```

```
##
##     0     1     2     3     4     5     6     7     8
##  1614 95513 14022 10994   164   160    47    61    36
```

```r
fviz_cluster(list(data = pca_sub[,1:2],
                  cluster = db$cluster),
             geom = "point",
```

```
            ellipse = FALSE,
            ggtheme = theme_minimal(),
            main = "S Clusters on PCA Space")
```



Conclusions

PCA

Principal Component Analysis revealed a strong dimensional structure in the dataset.

PC1 is mainly driven by popularity / engagement variables — Peak CCU, Positive Reviews, Recommendations, Estimated Owners, Playtime. This component can be interpreted as a "Market Success & Player Engagement" axis.

PC2 is influenced more by game design / ecosystem features

Clustering

Density-based clustering (DBSCAN) revealed natural groupings of games and isolated outliers without assuming spherical cluster shapes like K-means has due to highly skewed engagement metrics and uneven density distribution in the Steam marketplace. Most games ended up in one large group with similar low engagement.