

# Analyse de données

## Optimisation des karts sur Mario Kart 8 Deluxe

Justine Simon Clarisse Le Philippe

Université de Strasbourg

12 janvier 2026

# Contexte du projet

Sur Mario Kart 8 Deluxe, le joueur peut choisir entre plusieurs composants pour son kart :

- le personnage
- le kart
- les roues,
- le planeur,

Chaque composant a des statistiques spécifiques :

- la vitesse,
- l'accélération,
- la maniabilité,
- le poids...

Le choix d'un véhicule repose sur des arbitrages entre ces statistiques, qui dépendent du style de jeu du joueur et du type de circuit sélectionné.

## L'objectif global :

Automatiser le choix de la meilleure combinaison possible en fonction de ces éléments, afin de proposer une aide à la décision personnalisée.

## Sous objectifs :

- Intégrer des préférences utilisateur via un questionnaire interactif.
- Mettre en place un système de pondération contextuelle.
- Automatiser le classement et la sélection de solutions optimales.
- Proposer une aide à la décision personnalisée pour le joueur.

# Explication des scores

Création de bases de données à partir des statistiques du site Mario Wiki.

Attribution de points à chaque composant du kart et à chaque personnage pour chaque statistiques prises en compte :

- vitesse (sol, eau, air, antigravité)
- maniabilité / manutention (sol, eau, air, antigravité)
- poids
- accélération
- mini-turbo

## Principe des points

Le score total est exprimé en point et correspond à la somme des points du personnage, du kart, des roues et du planeur, qui est compris entre 0 et 20. L'objectif étant d'obtenir la valeur la plus élevée possible afin d'être le plus performant.

# Construction et préparation des bases de données

Construction de quatre bases de données : personnages, karts, roues, planeurs.

	A	B	C	D	E	F	G	H	I	J	K	L	M
	Kart_Performance_Scores												
1	nom du kart	vitesse	vitesse (sol)	vitesse (eau)	vitesse (air)	besse (antigrav)	accélération	po	poils	maniabilité	manutention (sol)	manutention (eau)	manutention (air)
2	Kart Standard	3	3	3	3	3	4	2	2,75	3	2	3	3
3	Kart Retro	1,75	2	3	1	1	6	1	3,75	5	4	2	4
4	Proto B	3,75	3	3	4	5	3	3	2,5	2	2	2	4
5	Nautomobile	2,75	4	5	0	2	1	4	2	1	5	1	1
6	Chabotlet	2,75	2	2	4	3	5	2	3,25	4	2	4	3
7	Machopelle	3	5	1	2	4	1	3	1	1	1	0	2
8	Tubul B3	2,75	4	5	0	2	1	4	2	1	5	1	1
9	Boat bolide	2,75	5	2	1	3	0	4	0,5	0	1	0	1
10	Cavalhart	3,25	4	3	3	3	2	1	2,75	3	3	3	2
11	Parasolcorneille	1	0	1	1	2	7	0	4,5	5	4	4	5
12	Canardfloc / Batou roulet	2,25	2	5	2	0	6	0	3,5	4	5	3	2
13	Cavalhart / Espalhart	3	4	2	3	3	2	2	2,5	3	2	2	3
14	Proxoluar	3,75	3	3	4	5	3	3	2,5	2	2	2	4
15	Oz	3	4	2	3	3	2	2	2,5	3	2	2	3
16	GLB	2,75	5	2	1	3	0	4	0,5	0	1	0	1
17	W25 Flèche d'argent	2,75	2	2	3	4	5	1	3,5	4	3	3	4
18	300 SL Roadster	3	3	3	3	3	4	2	2,75	3	2	3	3

- Gestion du séparateur décimal.
- Nettoyage automatique des noms de colonnes.
- Correction des valeurs manquantes (variable taille des personnages).
- Calcul de statistiques moyennes de vitesse et de maniabilité selon l'environnement.

# Classification des circuits

Regroupement des circuits en 4 catégories :

- EAU
- VOL
- VITESSE
- TECHNIQUE

Chaque circuit appartient à une seule catégorie basée sur l'environnement dominant.

Pondérations spécifiques selon le type de circuit.

Exemple : Circuit aquatique :

sol : 40 %  
eau : 45 %  
air : 5 %  
anti gravité : 10 %

## Étape 1 — Choix du personnage

- L'utilisateur saisit le nom de son personnage.
- > Le programme transforme automatiquement la saisie et la base en minuscules et supprime les espaces inutiles.
- Une boucle vérifie si le personnage existe réellement dans la base.
- > Si le personnage est introuvable, la liste complète est réaffichée et une nouvelle saisie est demandée.

## Étape 2 — Choix du circuit

- L'utilisateur saisit le nom du circuit.
- Le programme nettoie également cette saisie (minuscules et suppression des espaces).
- Une boucle vérifie l'existence du circuit.
- > En cas d'erreur, la liste des circuits est réaffichée et la saisie est redemandée.

# Questionnaire : préférences et normalisation

- L'utilisateur note 5 critères entre 0 et 10 : vitesse, mini-turbo, maniabilité, accélération et poids.
- **Pourquoi normaliser ?** Pour interpréter les notes comme des **poids relatifs** comparables entre utilisateurs.
- Exemple : 10,10,10,10,10  $\Rightarrow$  chaque critère vaut **20%**  $\Rightarrow$  il veut un kart polyvalent.

```
prefs <- c(vitesse = p_vitesse,  
          mini_turbo = p_drift,  
          maniabilite = p_mania,  
          acceleration = p_accel,  
          poids = p_poids)
```

## Principe de la normalisation

La normalisation consiste à diviser chaque préférence par la somme totale afin d'obtenir un vecteur dont la somme vaut 1.



# Algorithme (1) : combinaisons et addition des statistiques

- Génération de toutes les combinaisons **kart** × **roue** × **planeur** (produit cartésien).
- > Problème : colonnes dupliquées (vitesse, poids, etc.) dans chaque table.
- > Solution : ajout de suffixes (**\_kart**, **\_roue**, **\_planeur**) pour distinguer l'origine.
- Addition finale : **statistiques pièces** + **statistiques personnage**.

```
combos <- crossing(  
  kart %>%  
    select(nom_kart, all_of(stats_cols)) %>%  
    rename_with(~ paste0(.x, "_kart"), all_of(stats_cols)),
```

## Algorithme (2) : adaptation au circuit

- Les circuits sont classés en catégories :Eau, Vol, Vitesse, Technique.
- Chaque catégorie fournit des pondérations  
 $w = (w_{sol}, w_{eau}, w_{air}, w_{anti})$ .
- On calcule ensuite :
  - **vitesse contextuelle**  $v_{ctx}$  (pondérée selon l'environnement)
  - **maniabilité contextuelle**  $m_{ctx}$

# Algorithme (3) : score global et classement

## Score final

Somme pondérée des critères, selon les préférences normalisées.

- Le programme affiche :
  - le **meilleur combo** (score maximal),
  - un **Top 10** des meilleures alternatives.
- Intérêt : proposer un choix optimal mais aussi des options proches.

```
cat("\n=== MEILLEUR COMBO ===\n")
print(resultats %>% slice(1)
      %>% select(nom_kart, nom_roue, nom_planeur, score))
```

```
cat("\n=== TOP ", top_n, " ===\n", sep="")
print(resultats %>% slice_head(n = top_n)
      %>% select(nom_kart, nom_roue, nom_planeur, score))
```