Segmentez des clients d'un site e-commerce

Soutenance du projet n°4 : Parcours « Ingénieur Machine Learning »

Luke Duthoit

Plan de la présentation :

- Présentation de la problématique, du *cleaning* effectué, du *feature engineering* et de l'exploration

- Présentation des différentes pistes de modélisation effectuées et du modèle final sélectionné

- Présentation de la simulation pour définir le délai de maintenance du modèle (contrat de maintenance)

Plan de la présentation :

- Présentation de la problématique, du *cleaning* effectué, du *feature engineering* et de l'exploration

- Présentation des différentes pistes de modélisation effectuées et du modèle final sélectionné

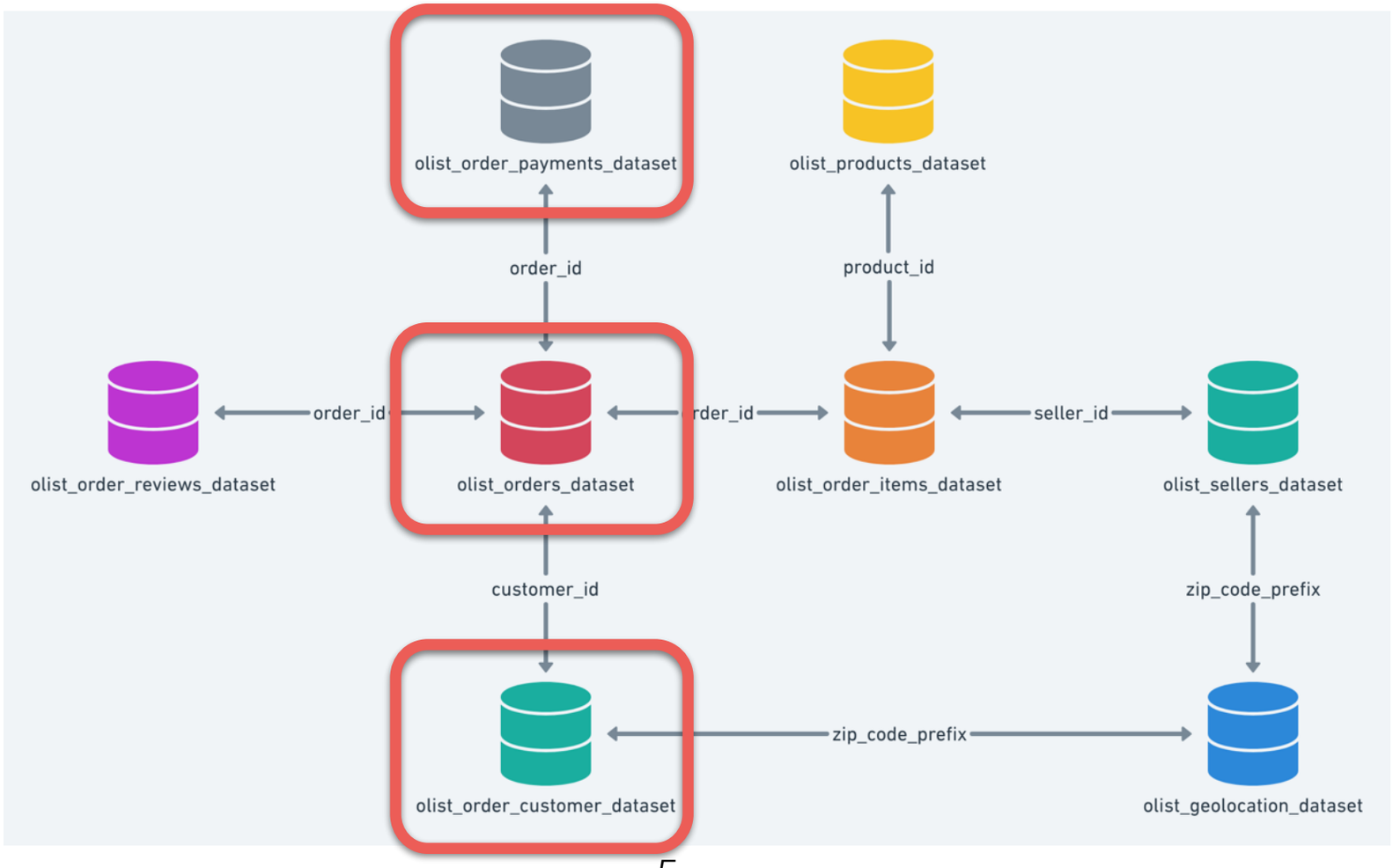
- Présentation de la simulation pour définir le délai de maintenance du modèle (contrat de maintenance)

La problématique du projet :

- *Data set* : données d'un site de e-commerce brésilien, sur ~ 2017-2018
- Paramètres : données de commandes/achats/ventes entre clients
- Segmenter l'ensemble des clients en plusieurs catégories, et en déduire une stratégie marketing
- Évaluer la stabilité dans le temps de ces catégories

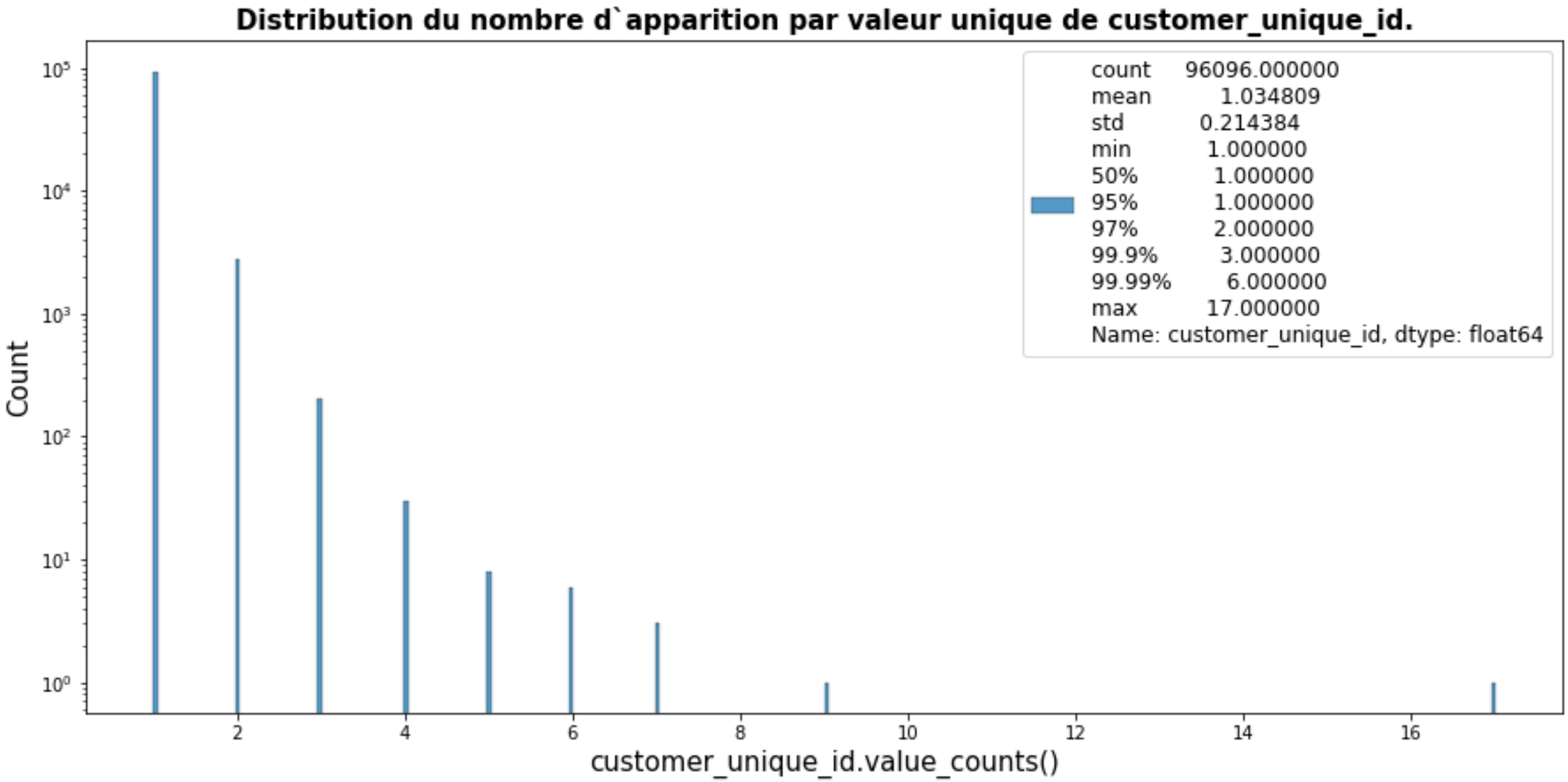
L’exploration de la base de données

- 8 base de données différentes
- Segmentation RFM → 3 bases indispensables pour calculer ces paramètres
- On les fusionne selon leurs paramètres communs



Choix de la fenêtre temporelle pour calculer RFM

- Problème : moins de 3% des clients présents > 1 fois
- Solution : on fait le calcul sur les deux ans, pour être sûr de les inclure.



Création d'une base de données adaptée : obtention des paramètres RFM

Base de 100.000 éléments → TROP

- Sélection aléatoire de 10% des éléments (identifiant des clients « *customer_unique_id* »)

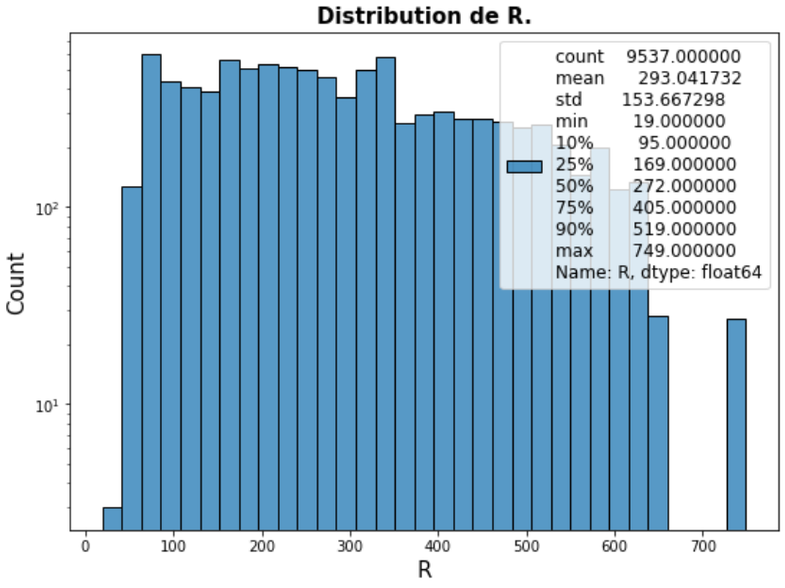
Pour chaque client :

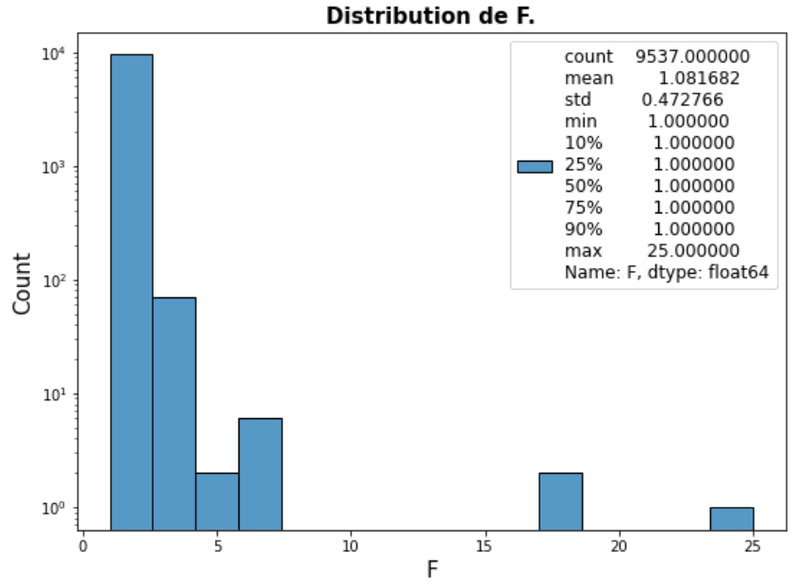
- R = durée écoulée depuis la dernière date d'apparition [jour] (« *order_purchase_timestamp*»)

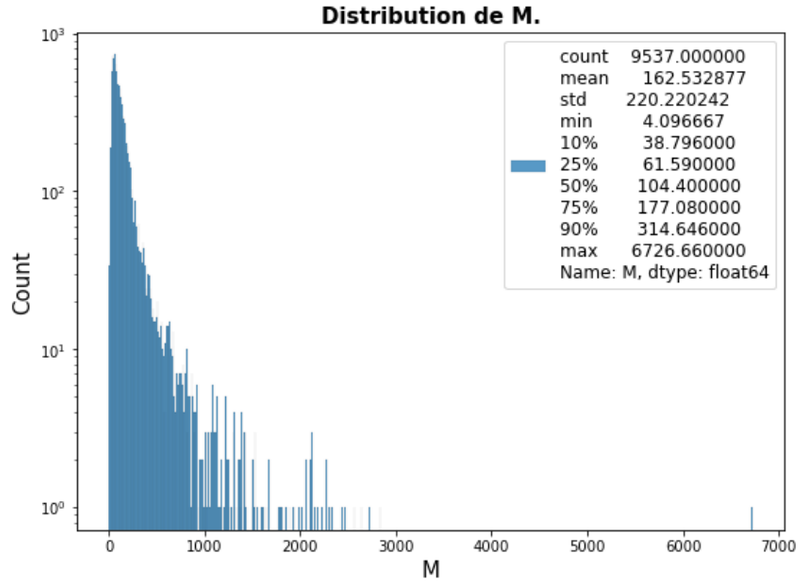
- F = nombre d'apparition dans la fenêtre temporelle (« *customer_unique_id*»)

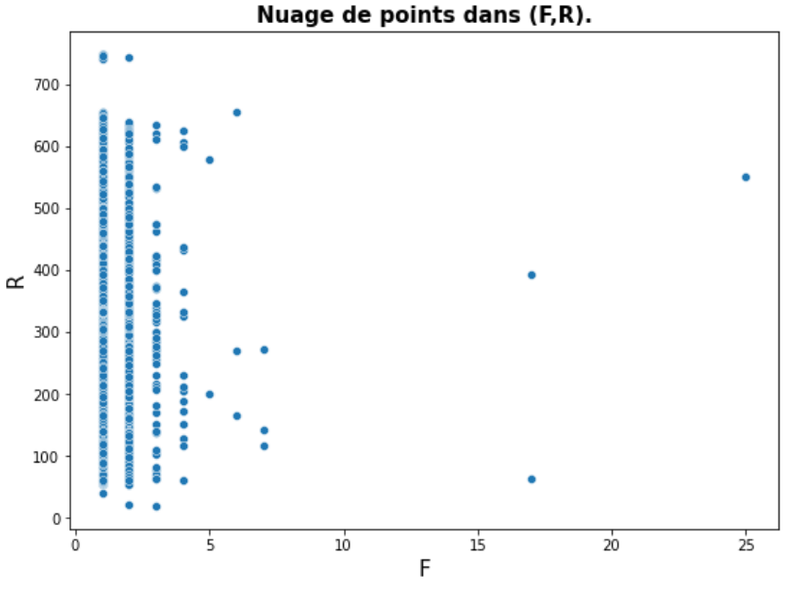
- M = moyenne des montants dépensés sur la fenêtre temporelle [monnaie courante] (« *payment_value*»)

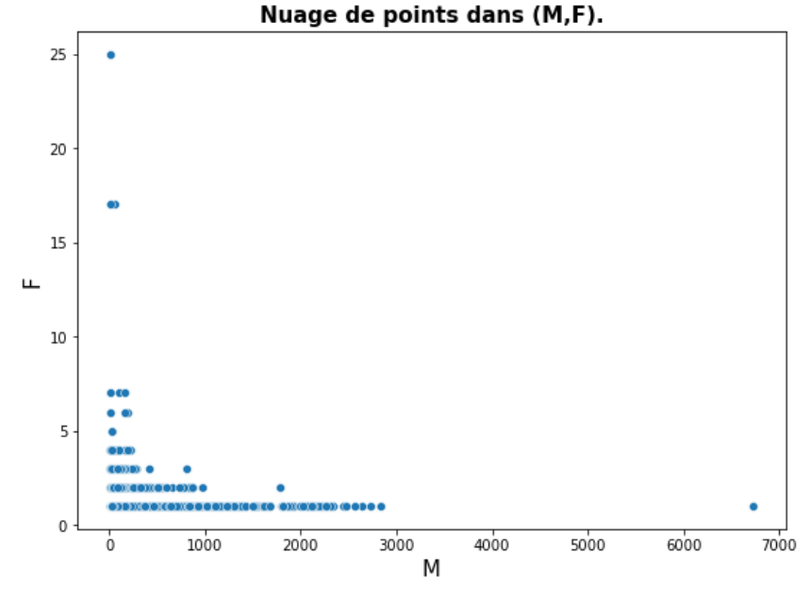
Création d'une base de données adaptée : obtention des paramètres RFM

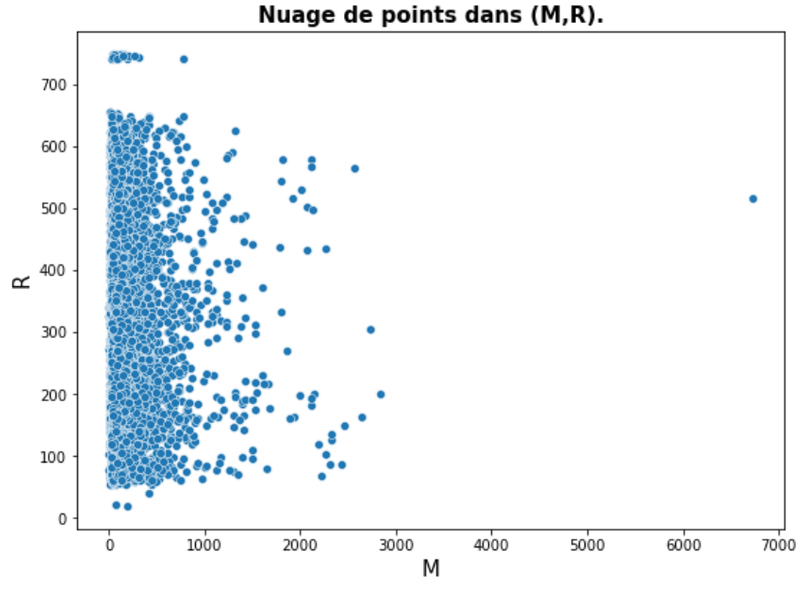








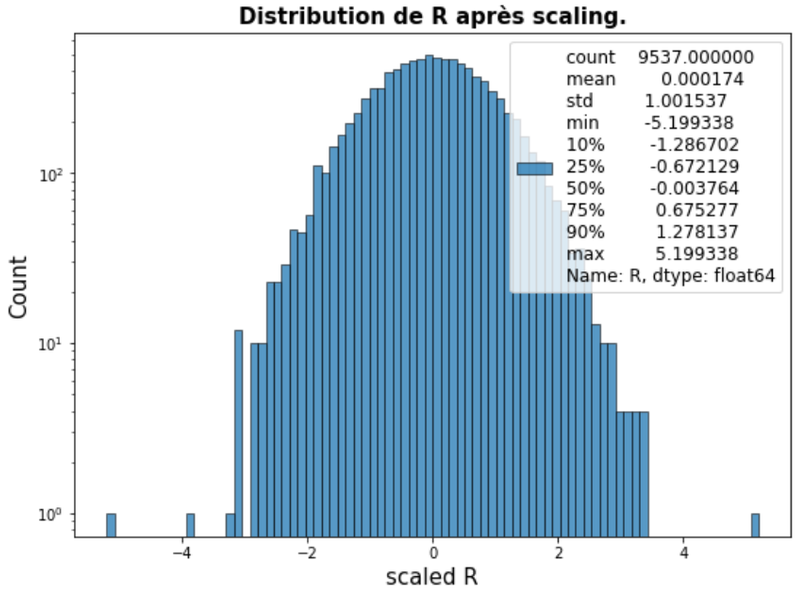


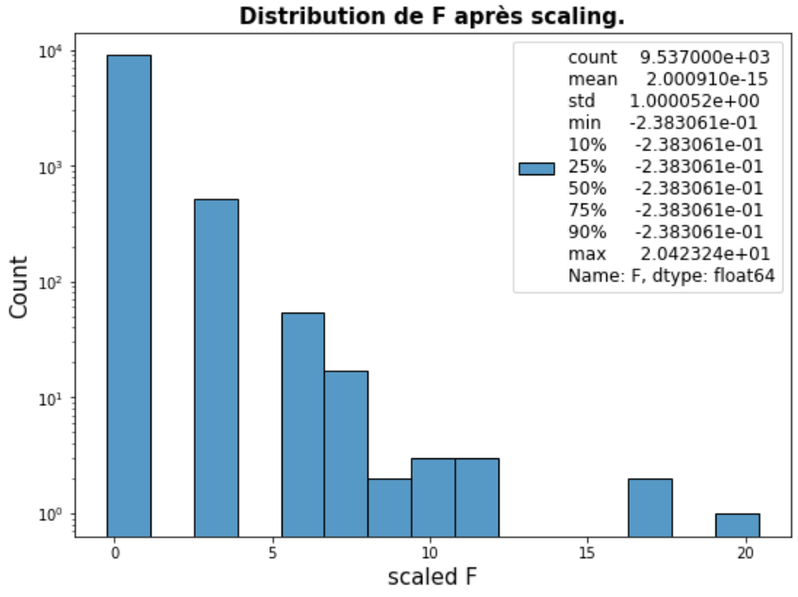


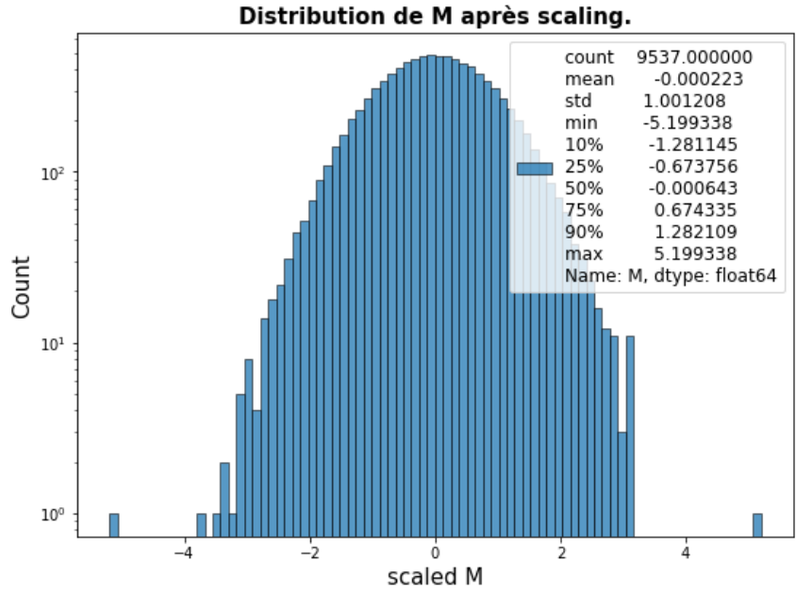
Feature engineering : égaliser les variances entre paramètres

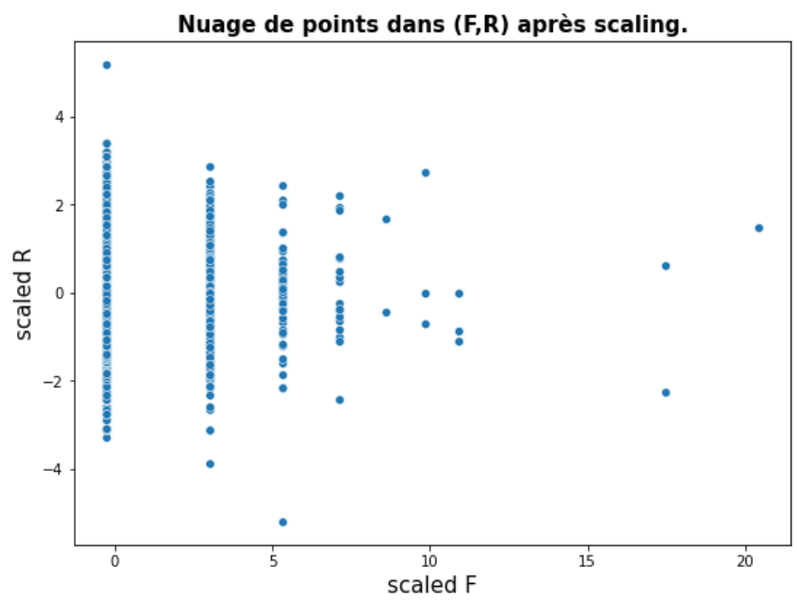
- Passage au log, puis *StandardScaler* ou *QuantileTransformer.*

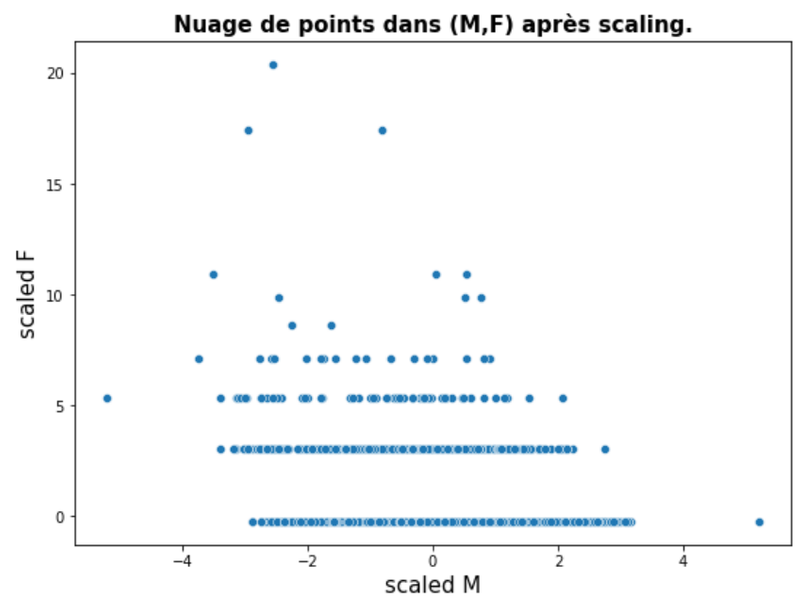
- Malgré tout, la distribution de F pourrait poser des problèmes pour la clusterisation

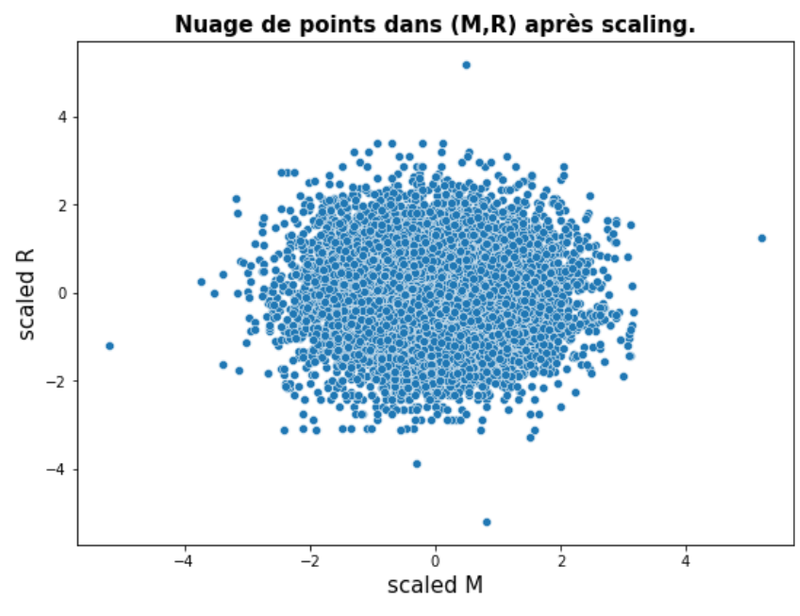








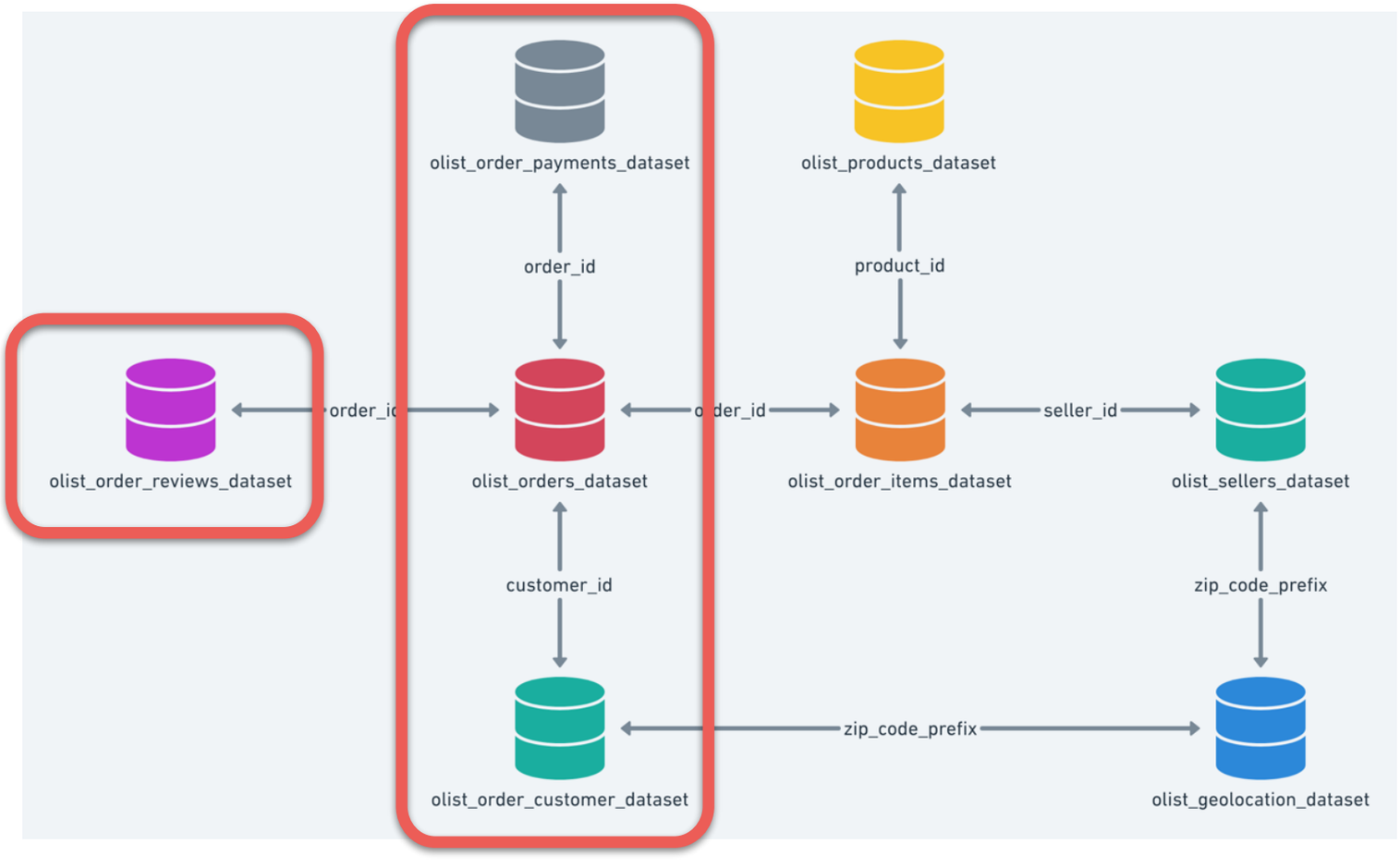




Feature engineering

Nous avons décidé de rajouter quelques *features* aux 3 précédents, pour compléter RFM

- Note de satisfaction (« *review_score* » dans *olist_order_reviews_dataset*)
- Longueur du message d'avis (calculé à partir de *review_comment_message* dans *olist_order_reviews_dataset*)
- Statut de livraison de la commande (issu de « *order_status* » après *one hot encoding*)



Plan de la présentation :

- Présentation de la problématique, du *cleaning* effectué, du *feature engineering* et de l'exploration

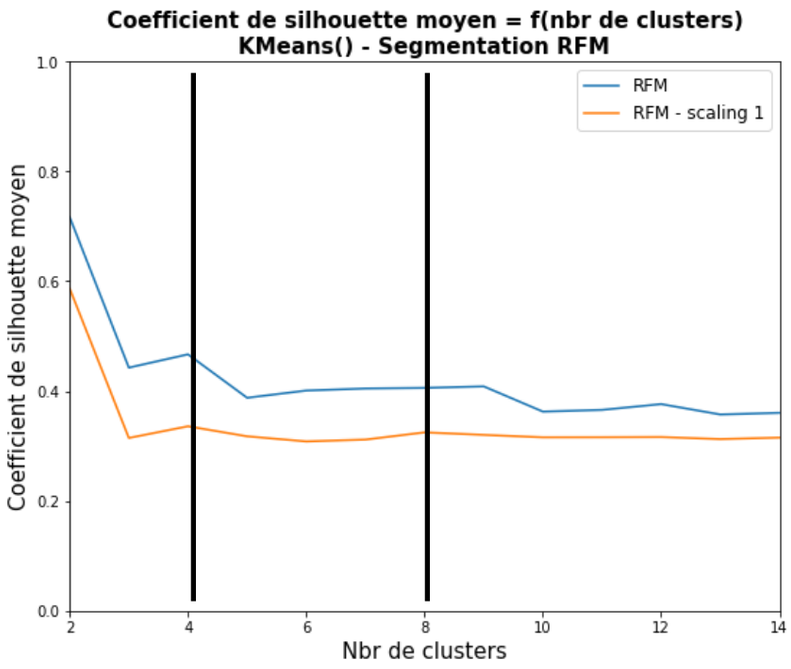
- Présentation des différentes pistes de modélisation effectuées et du modèle final sélectionné

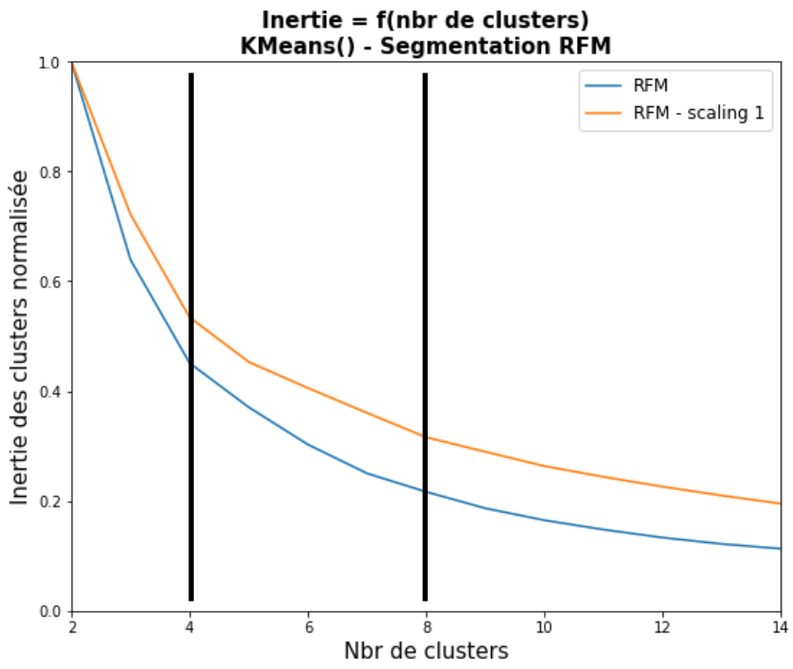
- Présentation de la simulation pour définir le délai de maintenance du modèle (contrat de maintenance)

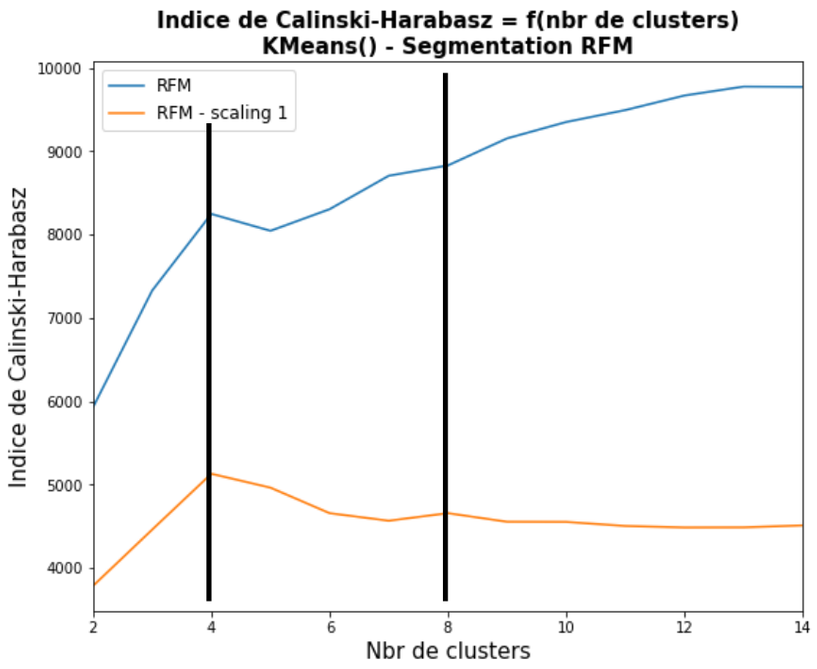
Présentation des différentes pistes de modélisation effectuées : test d'un modèle simple

- On teste les features RFM avec KMeans
- On utilise 3 métriques pour estimer un nombre optimal de clusters

Résultat = entre 4 et 8



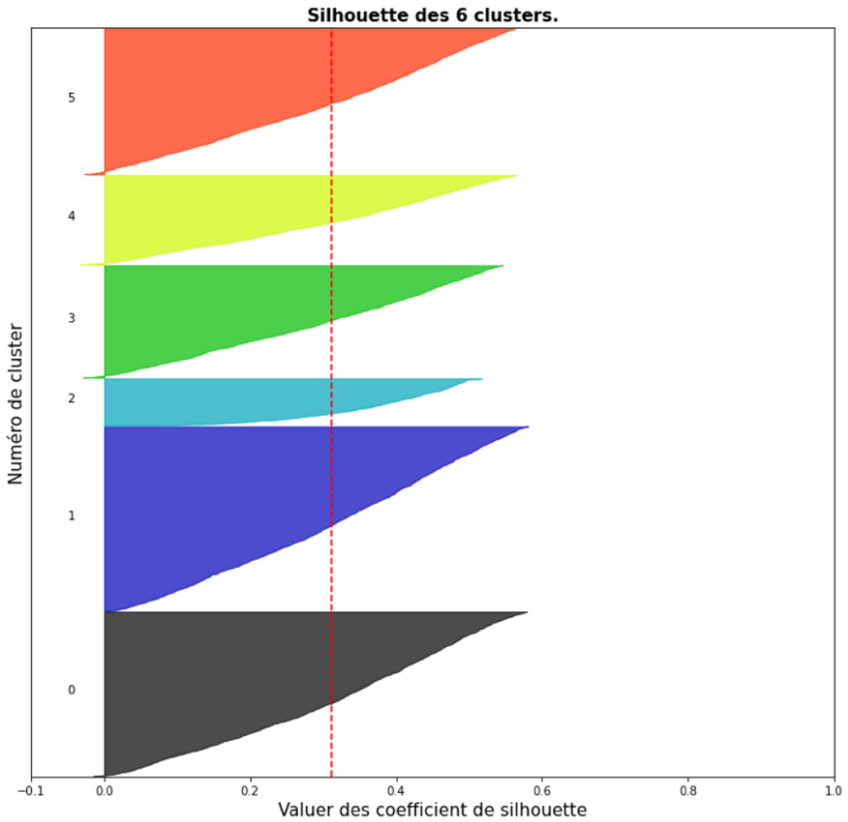


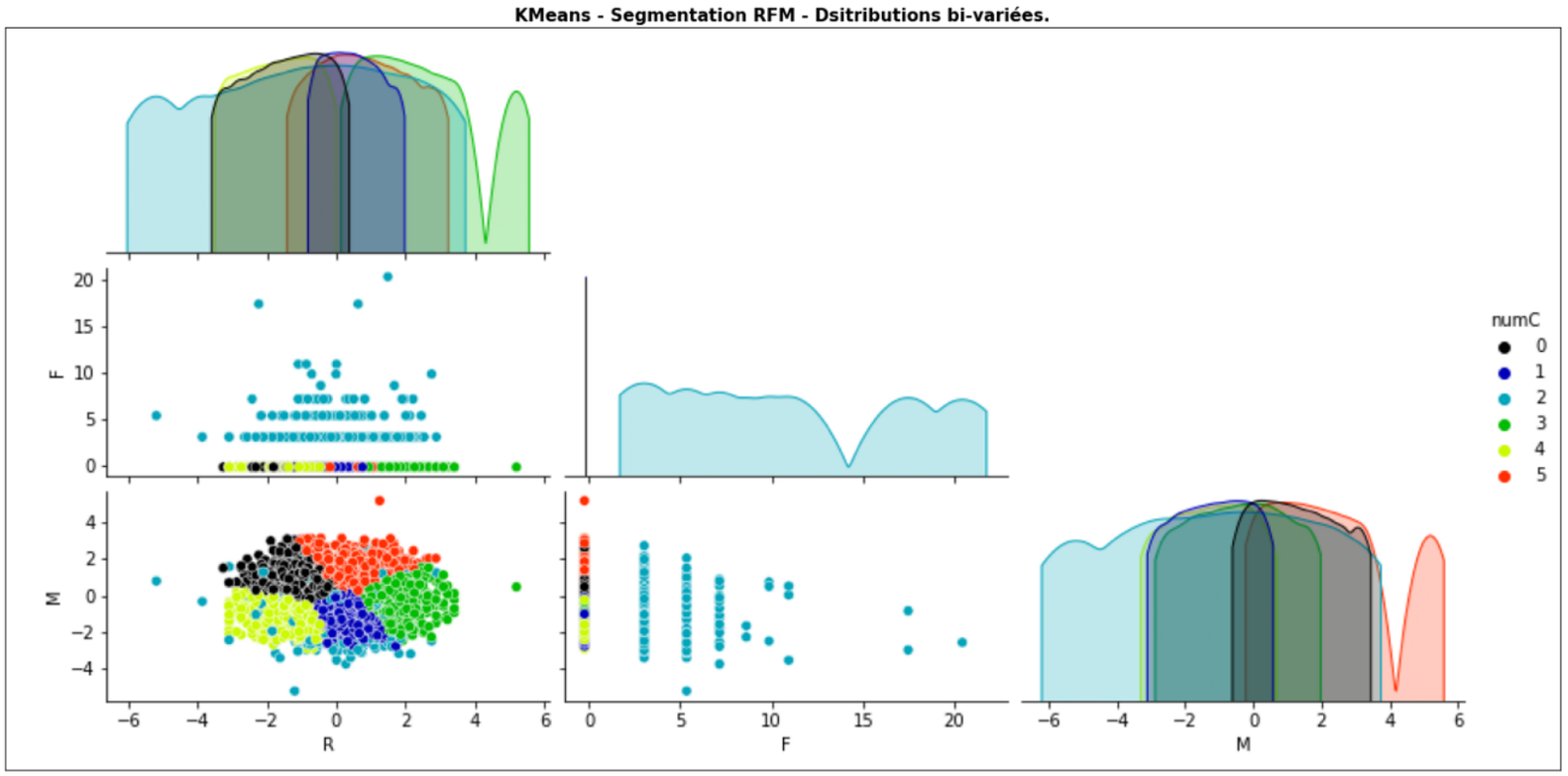


Présentation des différentes pistes de modélisation effectuées : test d'un modèle simple

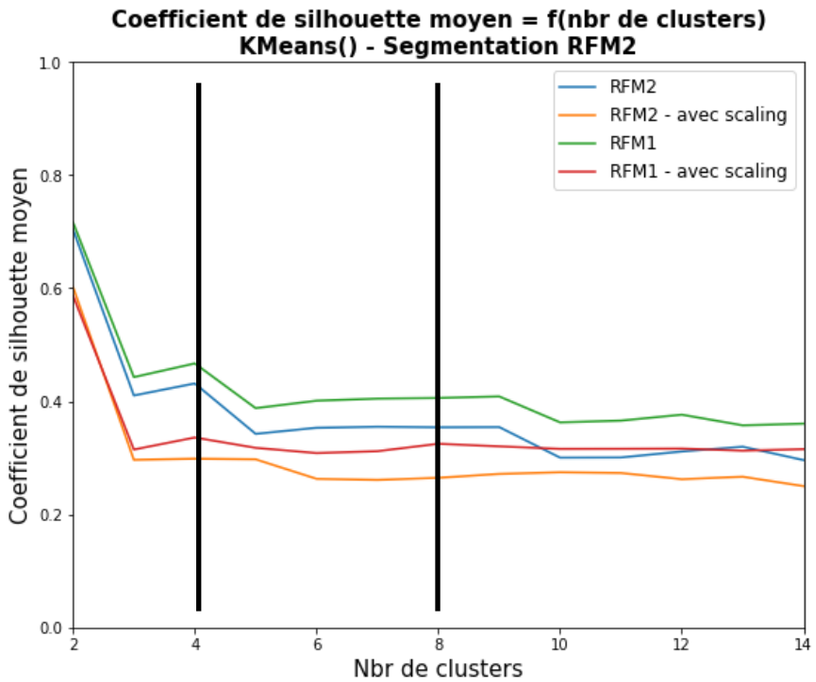
Le *feature scaling* homogénéise un peu plus les silhouette des clusters, MAIS…

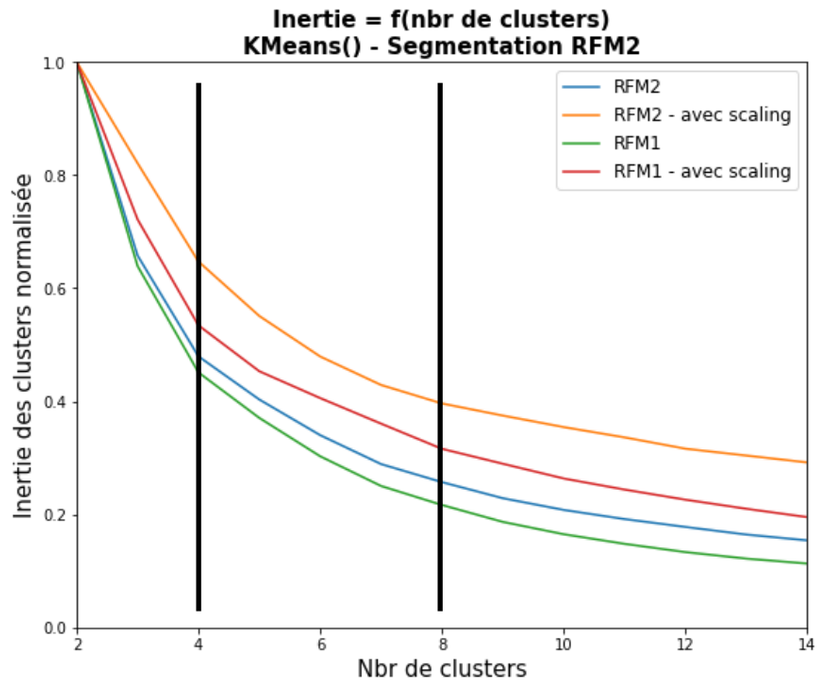
- oblige à une stratégie unique pour les clients récurrents (>1 achat)
- ne nous permet pas de trouver une stratégie simple sauf pour cluster « extrêmes » (récent et fort montant, ou l'inverse)

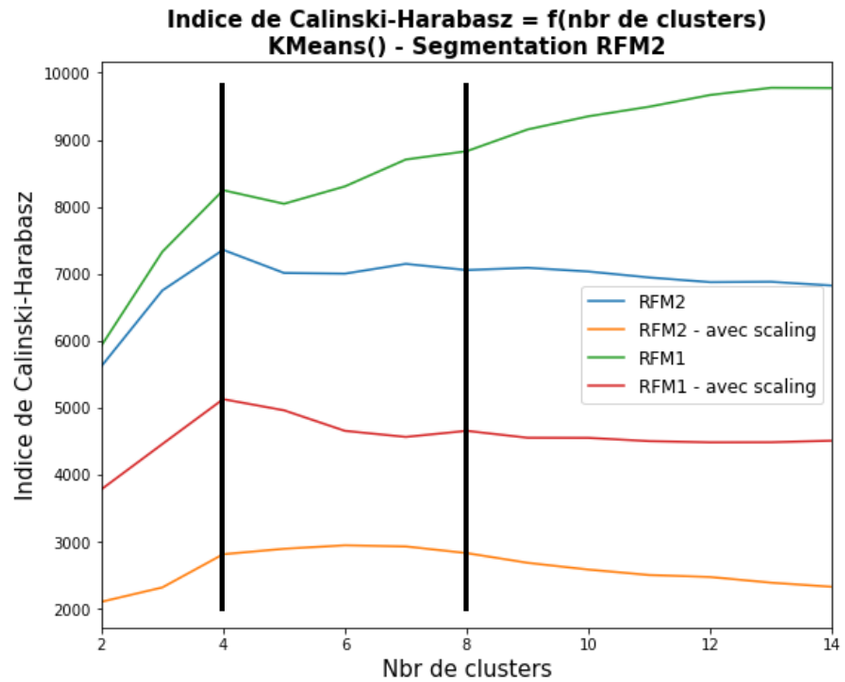




Présentation des différentes pistes de modélisation effectuées : ré-entreinement de KMeans sur RFM + features complémentaires

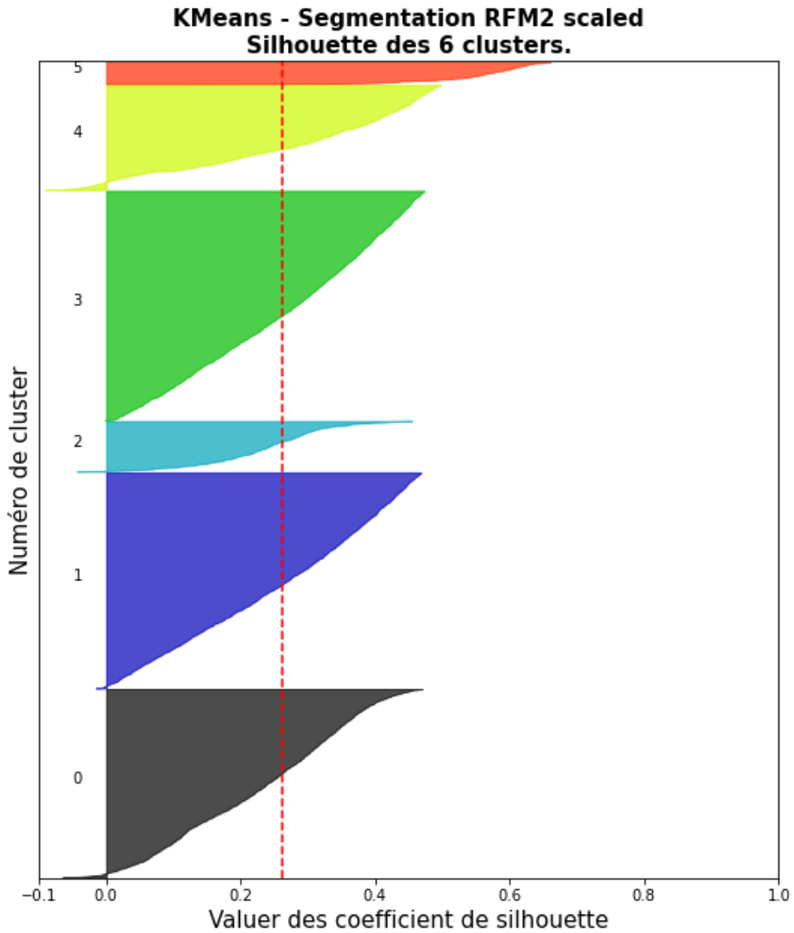






Mêmes considérations que pour RFM simple pour le nombre de clusters

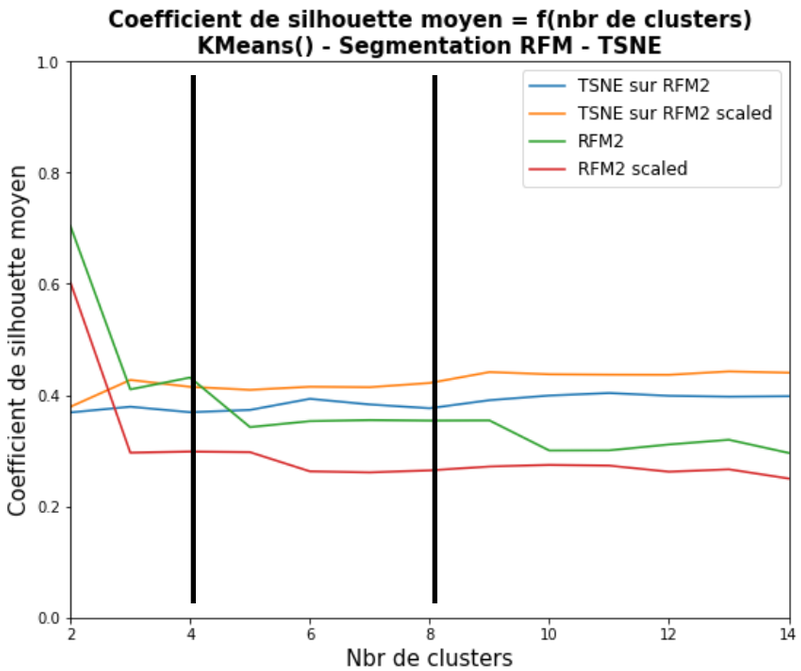
Présentation des différentes pistes de modélisation effectuées : ré-entreinement de KMeans sur RFM + features complémentaires

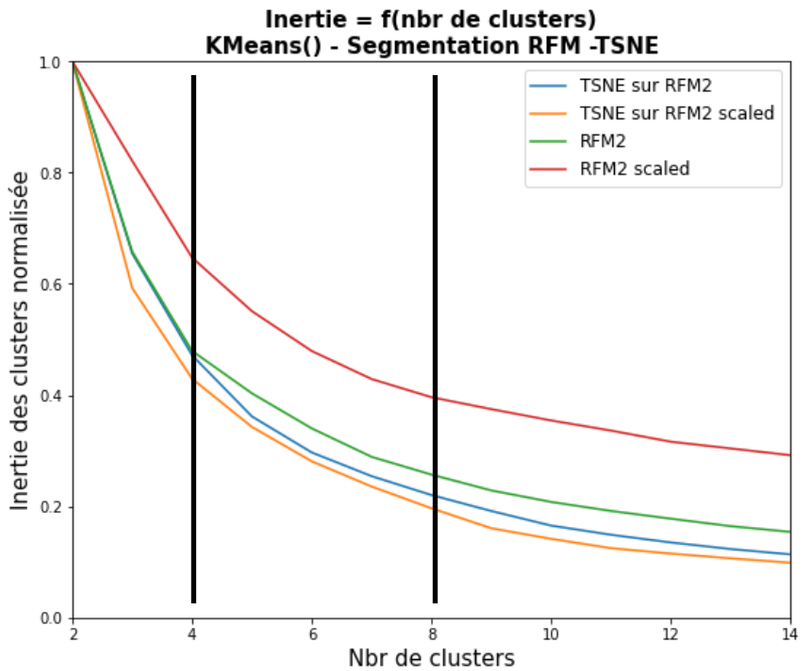


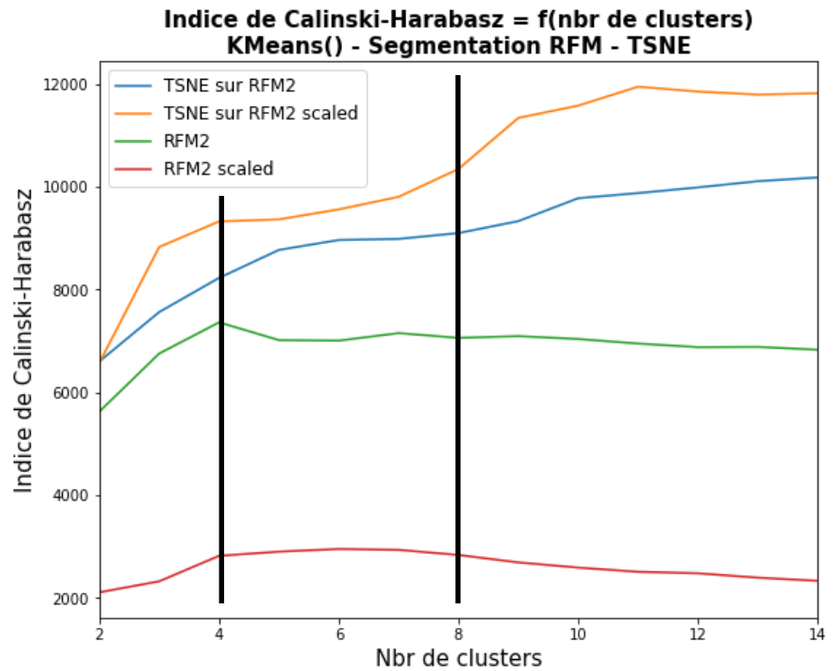
C	count	R mean	F mean	M mean	Score_m mean	Lenght_m mean	Bool_Delv_m mean
0	2214.0	307.820235	1.000000	158.700732	4.642276	62.827913	1.000000
1	2538.0	154.221434	1.000000	155.807277	4.586288	1.029945	1.000000
2	594.0	287.235690	2.289562	118.604221	4.056883	28.909933	0.978337
3	2698.0	413.694959	1.000000	158.932339	4.547813	0.393625	1.000000
4	1236.0	277.961165	1.000000	206.638495	1.426375	83.330097	1.000000
5	257.0	355.968872	1.050584	189.175966	1.661479	63.214008	0.000000

- Un peu de nuance apporté par les nouveaux paramètres, MAIS…
- Perte de variation inter clusters selon M

Présentation des différentes pistes de modélisation effectuées : KMeans sur (RFM + 3 features) réduites par TSNE

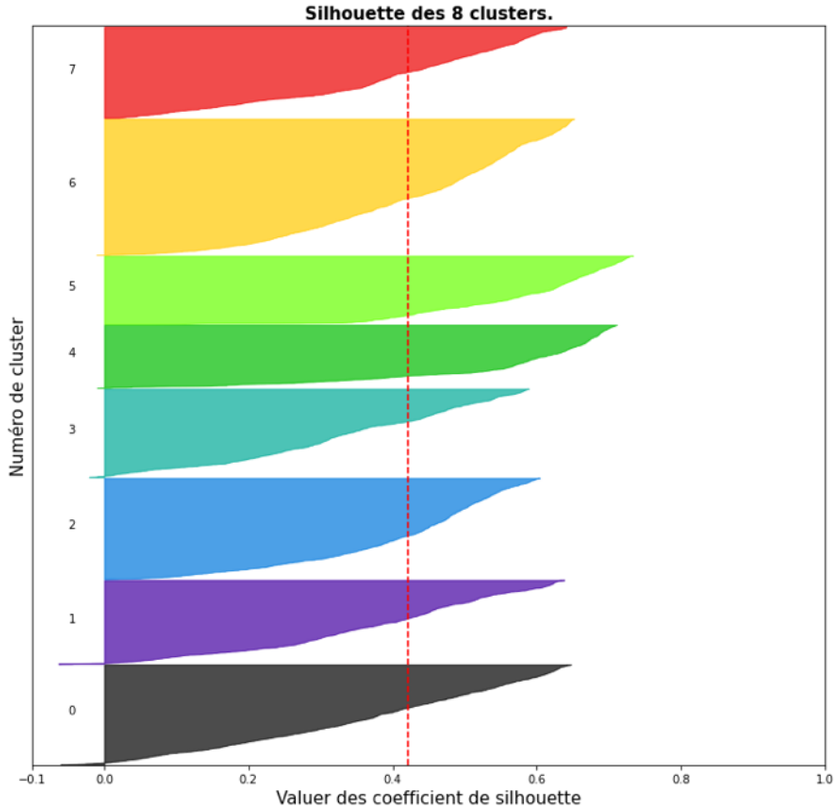


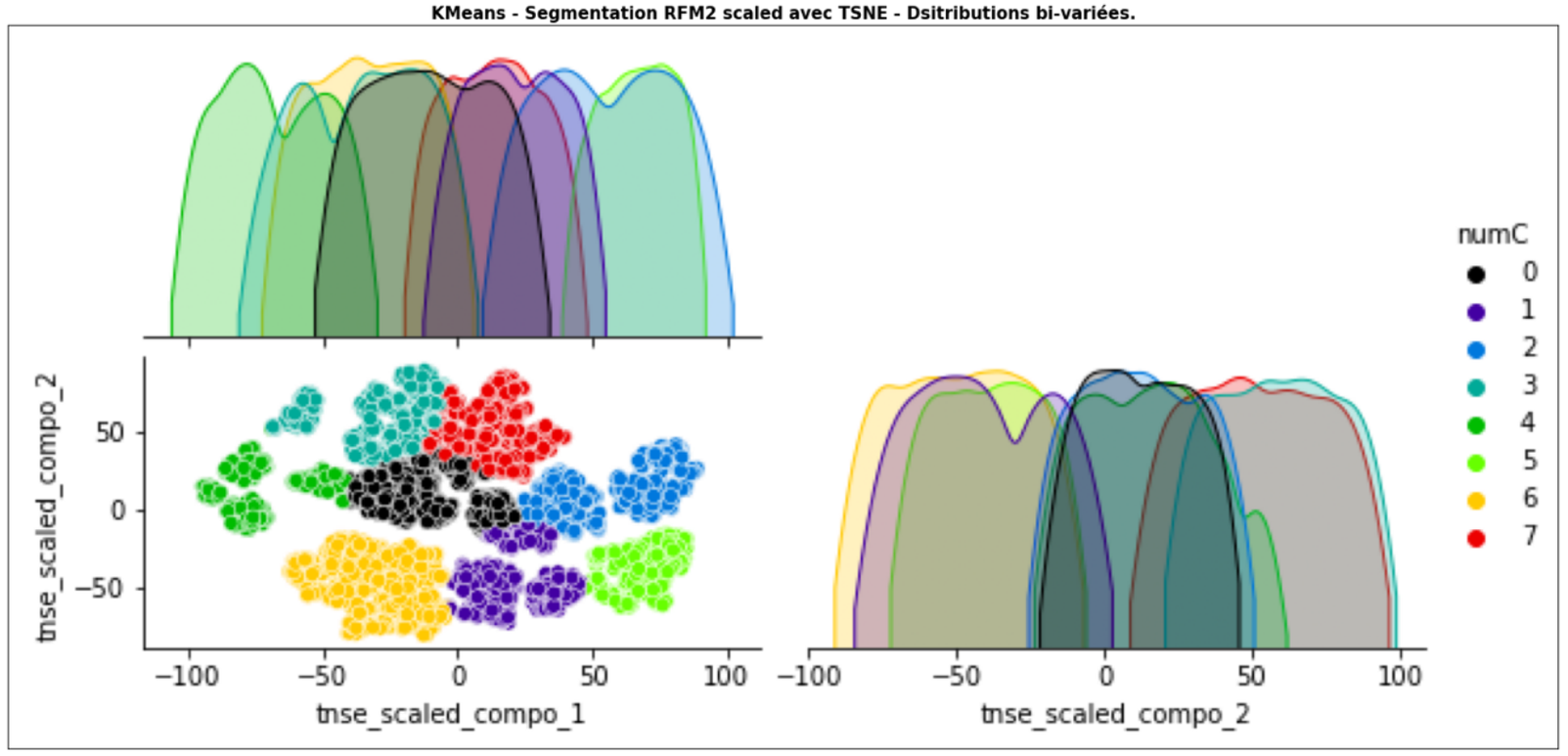




Encore une fois on obtient le même intervalle pour le nombre de clusters

Présentation des différentes pistes de modélisation effectuées : KMeans sur (RFM + 3 features) réduites par TSNE



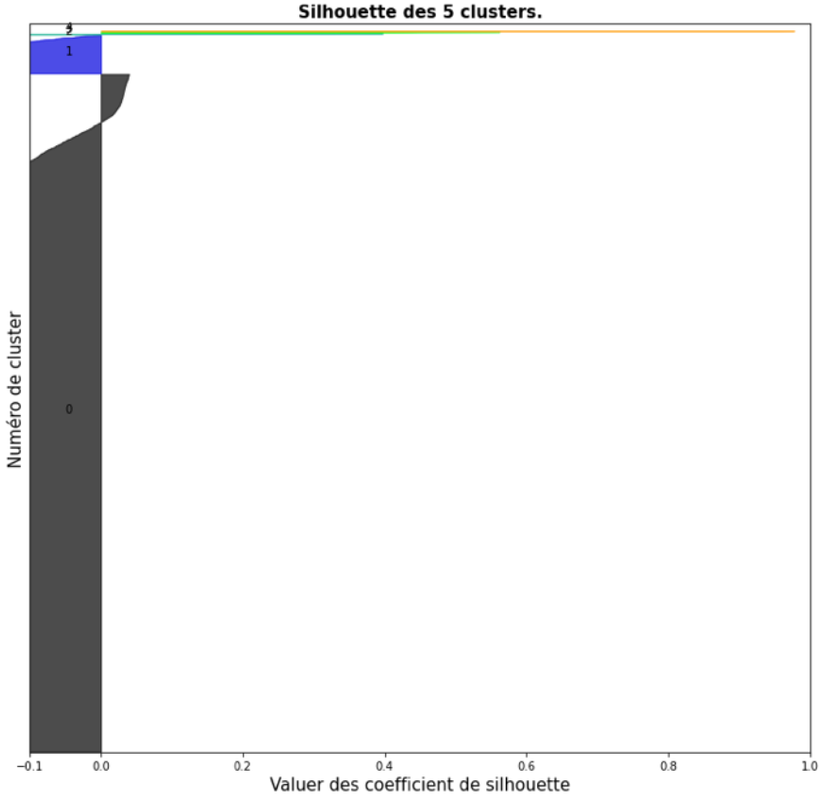


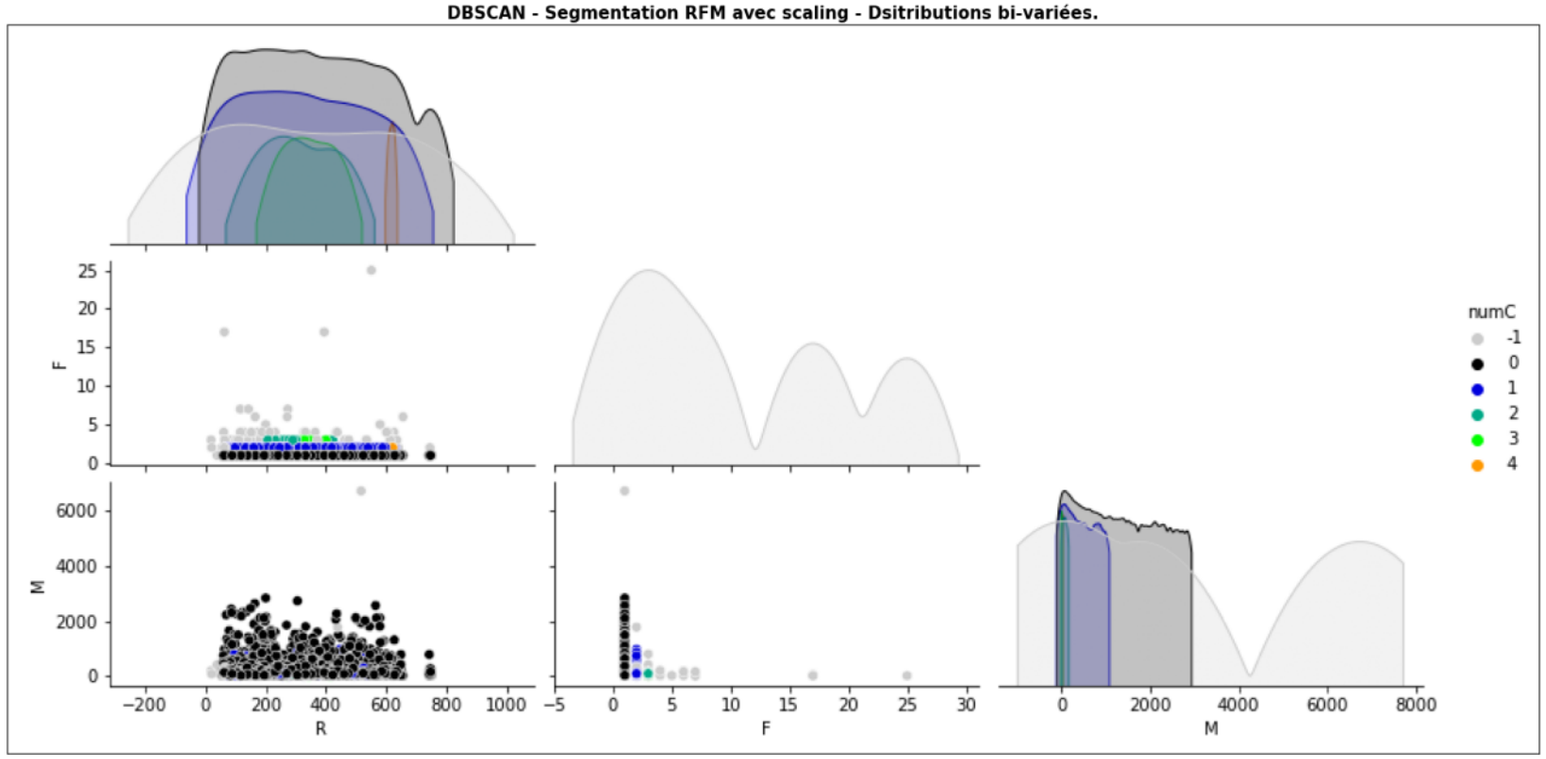
- Les clusters formés sont plus homogènes entre eux (silhouette et répartition des éléments)

- Privilégier 8 clusters plutôt que 6 pour ré-obtenir suffisamment des nuances pour des stratégies

	R	F	M	Score_m	Lenght_m	Bool_Delv_m
	count	mean	mean	mean	mean	mean
C						
0	1300.0	233.855385	1.000000	68.520531	4.789231	0.009231	1.000000
1	1093.0	275.499543	1.000000	132.305947	3.704483	57.137237	1.000000
2	1317.0	268.255885	1.000000	185.182855	3.173121	0.030372	1.000000
3	1153.0	417.982654	1.011275	128.181616	4.255854	14.092801	0.777103
4	822.0	340.801703	1.931873	96.551870	4.318478	20.891119	0.984346
5	890.0	288.088764	1.000000	199.356843	1.220225	101.323596	1.000000
6	1775.0	303.076620	1.000000	168.110158	4.993803	52.444507	1.000000
7	1187.0	235.787700	1.000000	311.306639	4.993260	0.000000	1.000000

Présentation des différentes pistes de modélisation effectuées : DBSCAN + RFM



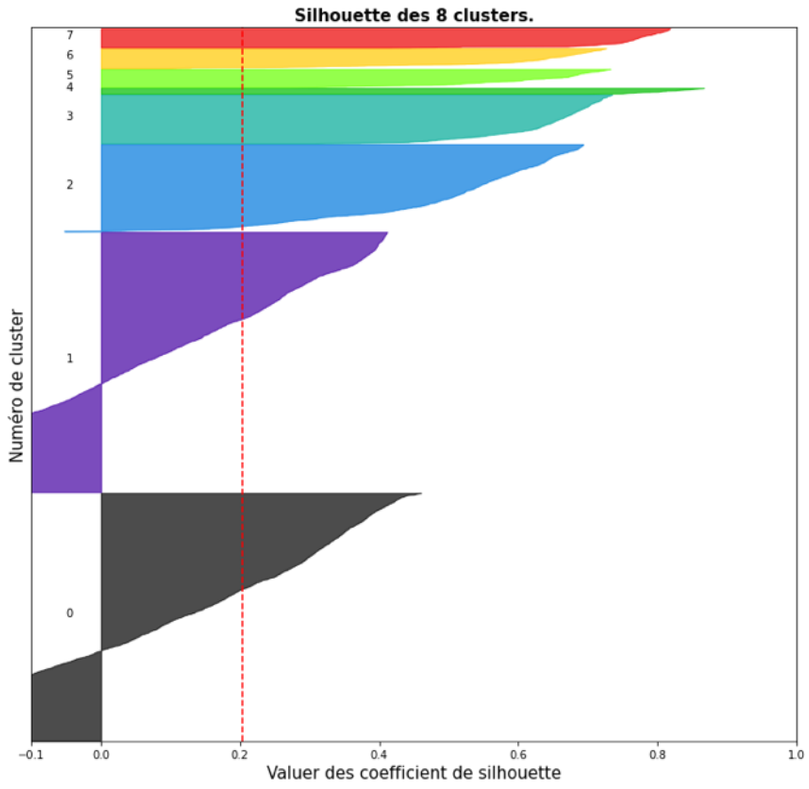


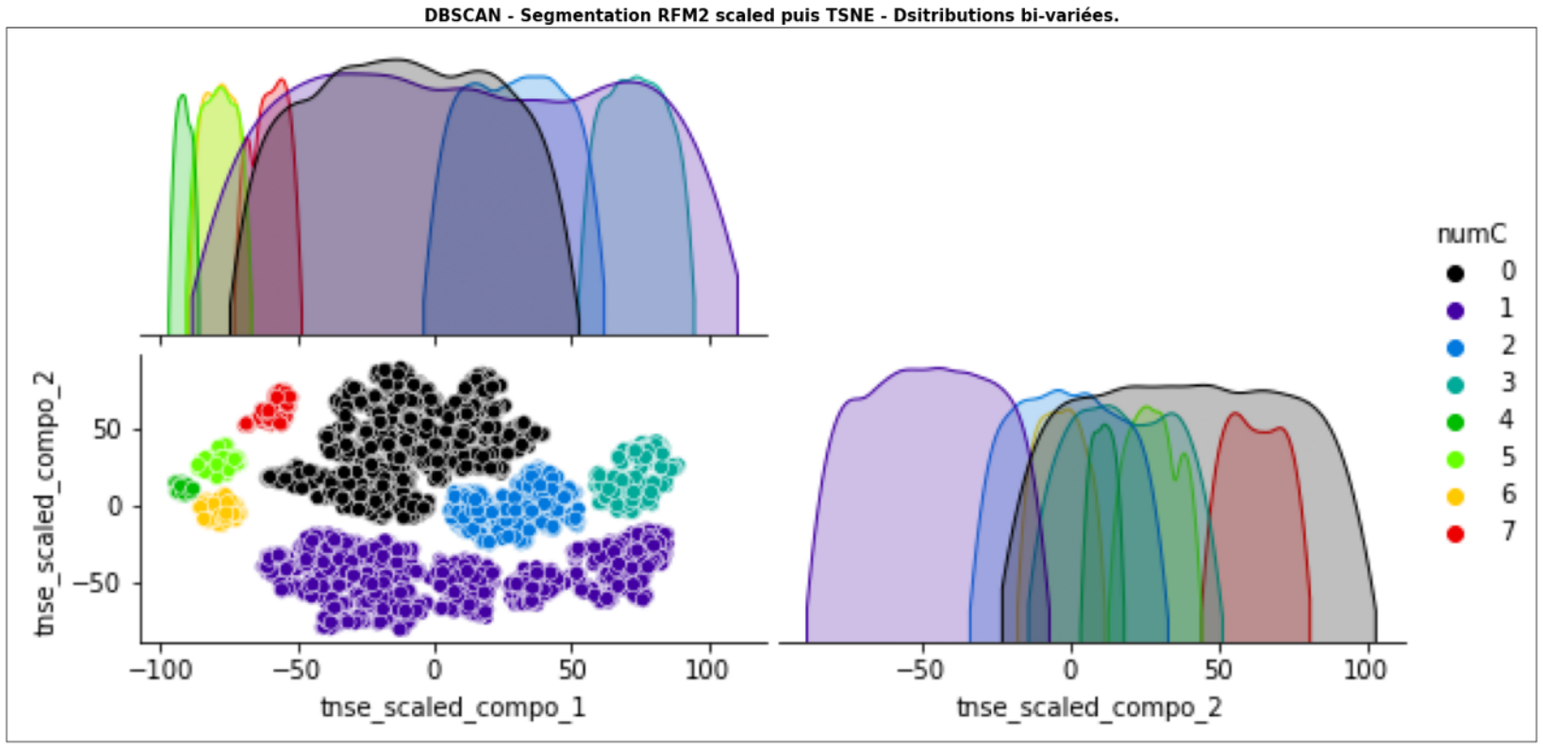
Instable, très peu de marge de manoeuvre pour jouer sur la distance :

- Clusters complètement inhomogènes
- Beaucoup de points assimilés à du bruit

Conclusion : stratégie impossible

Présentation des différentes pistes de modélisation effectuées : DBSCAN + (RFM + 3 features) réduites par TSNE





Plus stable, MAIS :

- Clusters de taille pas comparable
- Permet des stratégies précises mais pour un nombre trop faible d'éléments

Conclusion : stratégie trop couteuse

Modèle final sélectionné :
ré-entreinement de KMeans sur RFM + features complémentaires

Deux meilleures options :

- **KMeans sur RFM + *features* complémentaires**

- KMeans sur (RFM + 3 *features*) réduites par TSNE

On choisit la première car :

- Un peu moins de clusters pour stratégie comparable

- Plus simple à mettre en oeuvre

- Plus directement interprétable dans l'espace initial des données

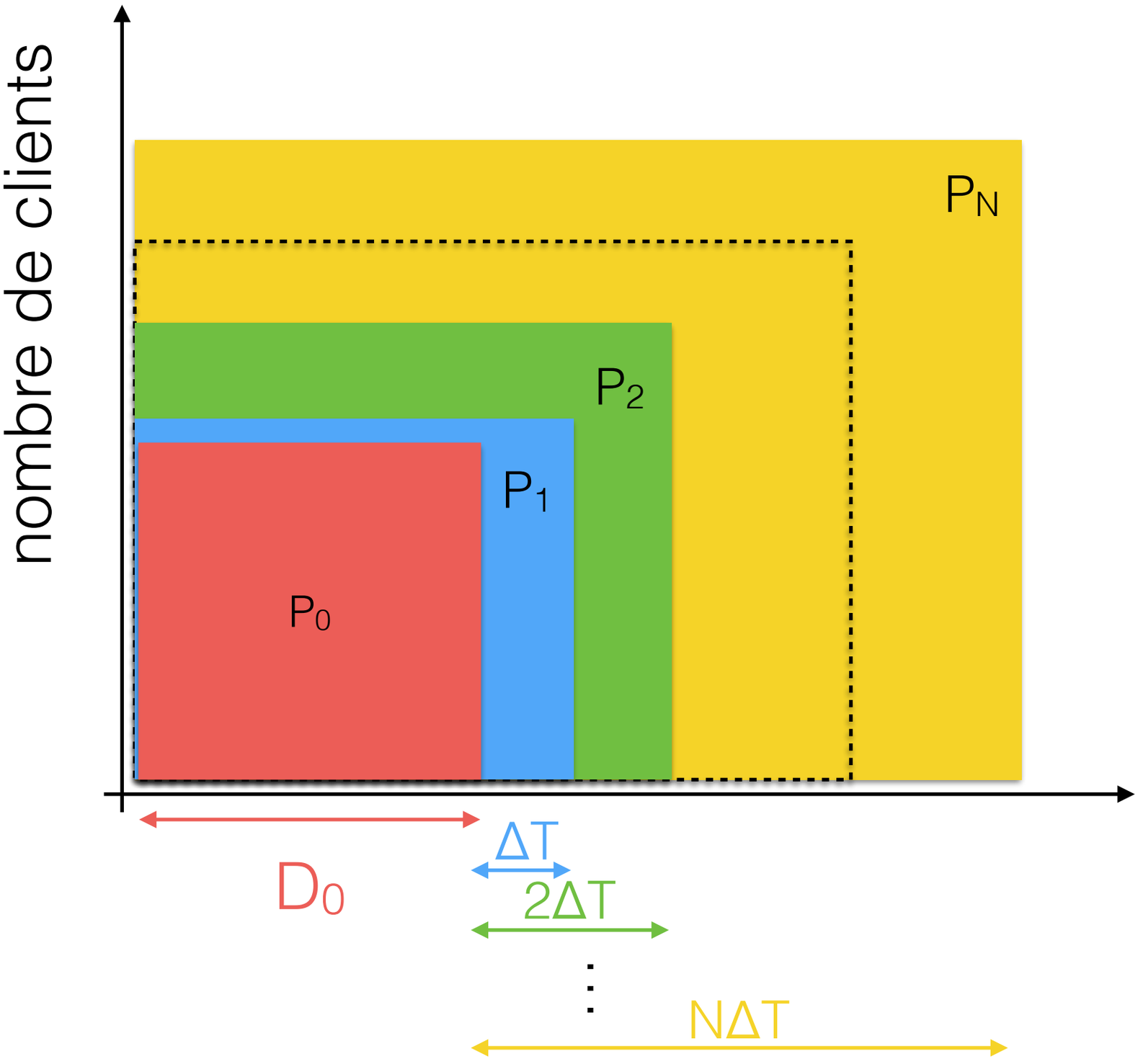
Plan de la présentation :

- Présentation de la problématique, du *cleaning* effectué, du *feature engineering* et de l'exploration

- Présentation des différentes pistes de modélisation effectuées et du modèle final sélectionné

- Présentation de la simulation pour définir le délai de maintenance du modèle (contrat de maintenance)

Schéma du protocole : calculer l'ARI à partir de clusterisations faites sur des *datasets* grandissant avec le temps.

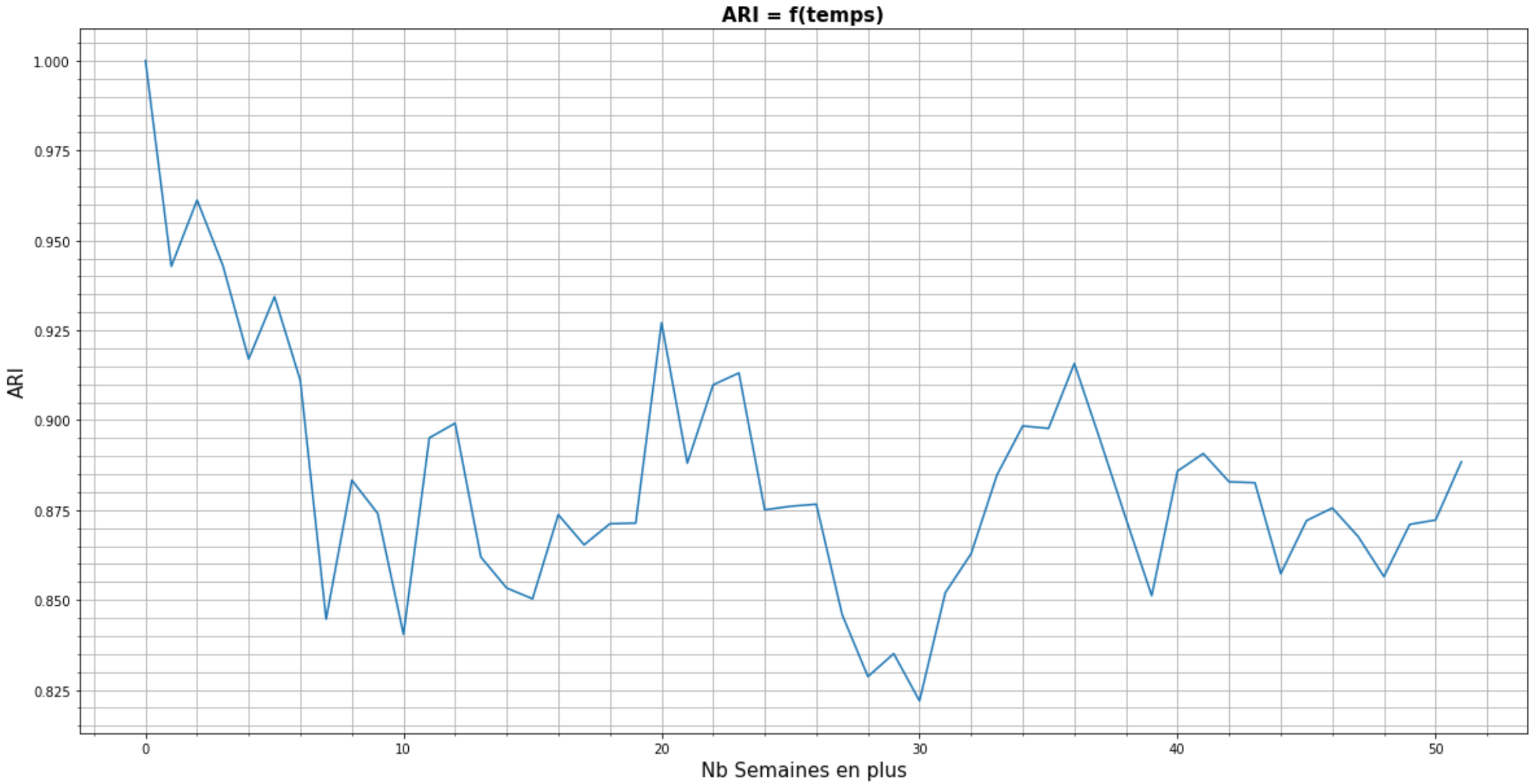


$K_0.fit(P_0)$
$K_N.fit(P_N)$

$C_{0N} = K_0.predict(P_N)$
$C_{NN} = K_N.predict(P_N)$

$ARI(C_{0N}, C_{NN})$

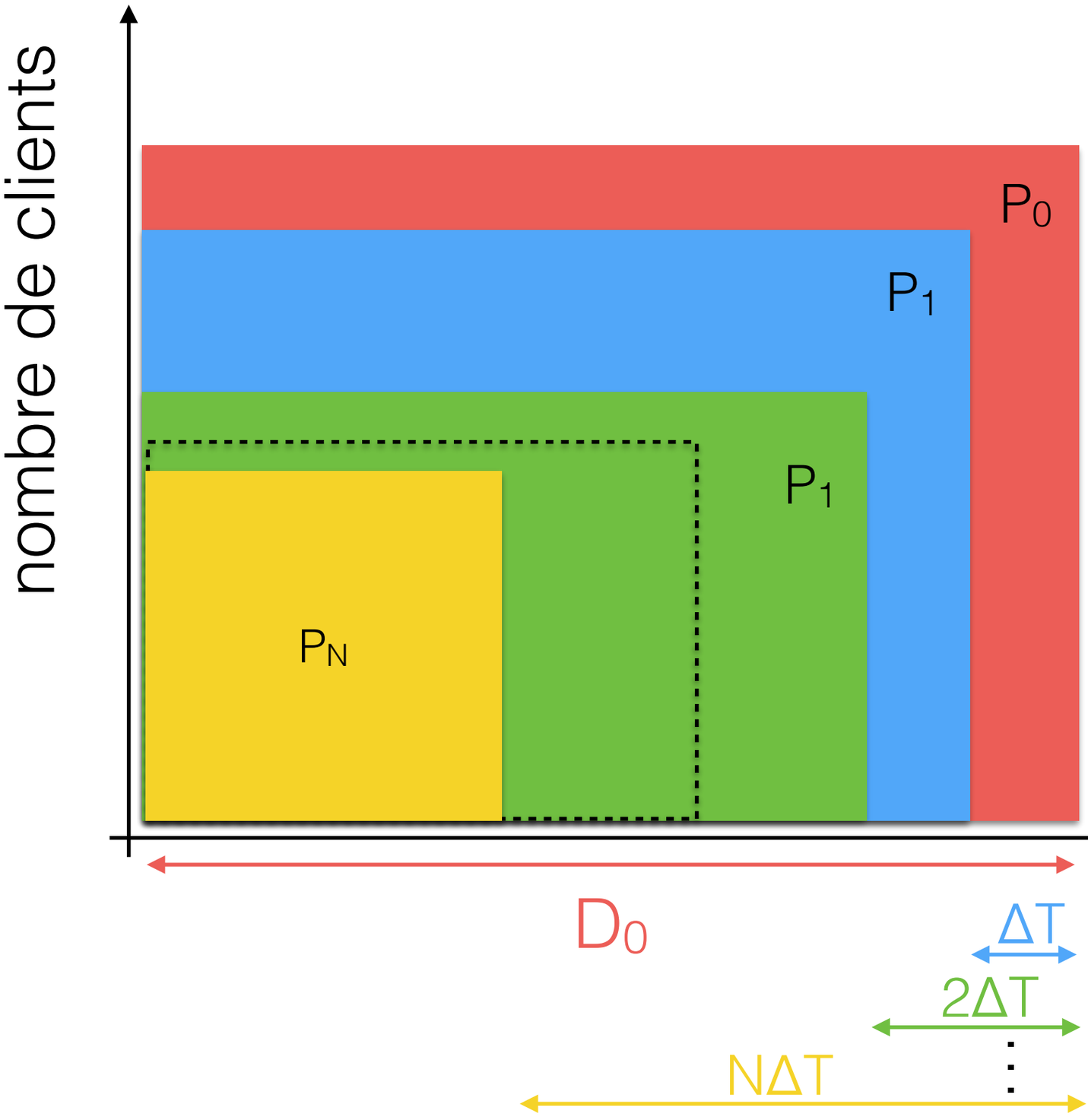
1er essai : (D_0, ΔT) = (1 an, +1 semaine).



- Clusters extrêmement stables (jamais ARI < 0.8)

- Probablement lié à l'évolution lente de la seule récence à chaque semaine.

On inverse le protocole : calculer l'ARI à partir de clusterisations faites sur des *datasets* diminuant avec le temps.

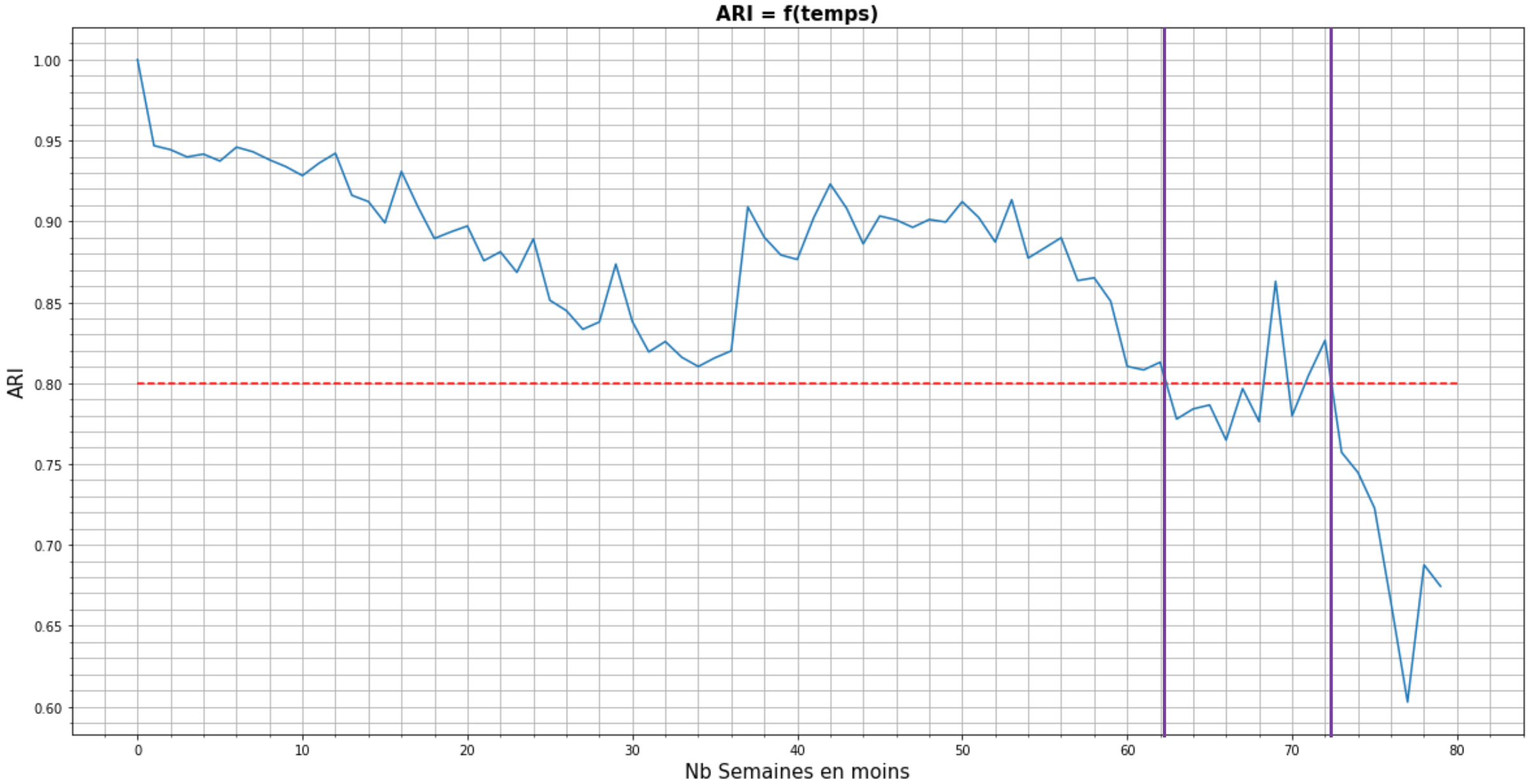


$K_0.fit(P_0)$
$K_N.fit(P_N)$

$C_{0N} = K_0.predict(P_N)$
$C_{NN} = K_N.predict(P_N)$

$ARI(C_{0N}, C_{NN})$

2ème essai : (D_0, ΔT) = (~2 an, -1 semaine).



Ré-entraînement nécessaire à partir de 62 - 72 semaines.

L’instabilité des clusters débute sur très peu de *features*.

On regarde les deux clusterisations au bout de 77 semaines :

- 4 / 6 quasiment identiques.
- les 2 autres diffèrent par :
  - le nombre d’éléments ;
  - la récence R ;
  - la longueur moyenne de commentaire.

*-77 semaines, **avec** ré-entraînement*

		R	F	M	Score_m	Lenght_m	Bool_Delv_m
	count	mean	mean	mean	mean	mean	mean
labels_k77							
0	135.0	0.000000	1.029630	142.308222	4.062963	24.829630	0.985185
2	388.0	50.814433	2.453608	92.039007	4.133862	26.829897	0.974227
3	426.0	73.187793	1.025822	227.944988	1.739437	66.607981	0.000000
4	994.0	54.718310	1.001006	170.210473	1.667002	80.190141	1.000000
1	1917.0	55.883672	1.000000	184.391440	4.782994	58.003652	1.000000
5	3559.0	48.866255	1.000000	158.266940	4.537230	0.107614	1.000000

*-77 semaines, **sans** ré-entraînement*

		R	F	M	Score_m	Lenght_m	Bool_Delv_m
	count	mean	mean	mean	mean	mean	mean
labels_k0							
4	392.0	50.313776	2.448980	97.098316	4.126118	27.313776	0.974490
5	427.0	73.016393	1.025761	228.288396	1.737705	66.669789	0.000000
1	935.0	55.188235	1.000000	171.309754	1.629947	84.771123	1.000000
2	1730.0	17.228902	1.000578	152.834575	4.452601	2.757803	0.999422
0	1772.0	56.146727	1.000000	187.724786	4.784989	61.726862	1.000000
3	2163.0	71.441516	1.000000	159.592469	4.531669	0.386038	1.000000

Conclusion et regard critique

- Segmentation RFM augmentée de 3 *features* interprétables facilement.
- Modèle KMeans() à 6 clusters, appliqué à un *dataset* avec *scaling*.
- Stratégie commerciale simple, mais grande stabilité temporelle des clusters.

Pistes d'améliorations :

- Calculer la fréquence de manière à avoir une distribution de valeurs continues.
- Plus de *features* ET transformées polynomiales pour affiner stratégie, quitte à passer par de la réduction de dimension importante.
- Utiliser *groupby*() plus tôt dans le projet pour accélérer les calculs RFM.