Ingénierie de la fouille et de la visualisation de données massives (RCP216)

Approches sans distribution

Michel Crucianu

(prenom.nom@cnam.fr)

http://cedric.cnam.fr/vertigo/Cours/RCP216/

Département Informatique

Conservatoire National des Arts & Métiers, Paris, France

22 septembre 2026

Le sujet de cette séance

- Comment traiter de grands volumes de données sans recourir à une plateforme distribuée (*cluster*) ?

- Deux approches
  1. **Réduction du volume de données** : échantillonnage (moins d'observations), réduction de dimension (moins de variables)
  2. **Calcul centralisé plus efficace** : par ex. bibliothèques Python Polars et Dask

→ Repousser significativement les limites d'une architecture centralisée (sans toutefois remplacer le calcul distribué pour des volumes réellement massifs)

Plan du cours

Réduction du volume de données

- Travailler sur un sous-ensemble représentatif des données originales
  - Diminuer le nombre d'observations → échantillonnage
  - Diminuer le nombre de variables → réduction de dimension
- Les résultats obtenus sont en général des approximations de ceux que l'on obtiendrait sur les données complètes !
- Risque majeur si faible densité en information : un échantillon trop petit ou un espace de représentation trop restreint peuvent rendre les régularités recherchées indétectables

Échantillonnage

- Objectif : inférer des propriétés de toute la « population » de N données à partir d'un échantillon de $n \ll N$ données
- Méthodes non aléatoires (choix d'expert, volontariat, etc.) : pragmatiques mais représentativité difficile à qualifier
- Méthodes aléatoires (voir par ex. [2]) : garanties statistiques
  1. **Échantillonnage simple** : tirages indépendants, en général sans remise, chaque observation a la même probabilité $p_s = \frac{n}{N}$ d'être sélectionnée
  2. **Échantillonnage stratifié** : sous-ensembles (strates) d'une certaine homogénéité interne ; échantillonnage simple appliqué dans chaque strate
  3. **Échantillonnage en grappes** : sous-ensembles (grappes) où l'hétérogénéité intra-grappe est plus forte que l'hétérogénéité inter-grappes ; on choisit un échantillon de grappes dont on retient toutes les observations ; utile pour faciliter la collecte, pas pour réduire le volume de données

Échantillonnage stratifié : exemple

- Étude des pratiques des clients du commerce en ligne : une certaine homogénéité à l'intérieur de chaque tranche de revenus, mais effectifs des tranches très déséquilibrés

→ Un échantillonnage simple global ne conserverait pas très bien la proportion relative de chaque tranche

→ Échantillonnage stratifié (1 strate = 1 tranche de revenus) avec un même taux de sélection p_s dans chaque strate : règle ce problème et augmente la précision pour un même n (ou conserve la précision avec n plus faible)

Échantillonnage en grappes : exemple

- Étude des pratiques sportives des élèves de troisième en zone urbaine : autant de diversité à l'intérieur d'un même collège qu'entre collèges

→ Plutôt que d'interroger quelques élèves dans chaque collège (échantillonnage simple), on sélectionne quelques collèges et on interroge tous leurs élèves de troisième

  - Solution bien plus facile à mettre en œuvre, mais sans intérêt particulier pour réduire le volume de données déjà disponibles

→ Pour réduire le volume, on utilisera l'échantillonnage simple ou stratifié

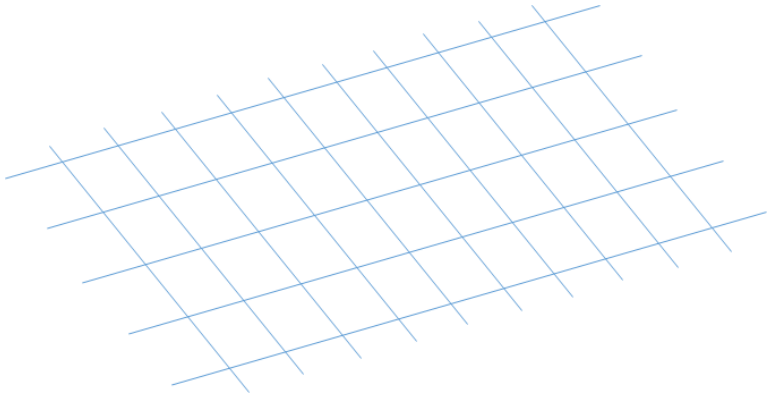
Réduction de dimension : objectifs et approches

N observations dans $\mathbb{R}^m \rightarrow N$ observations dans $\mathbb{R}^k$, $k \ll m$

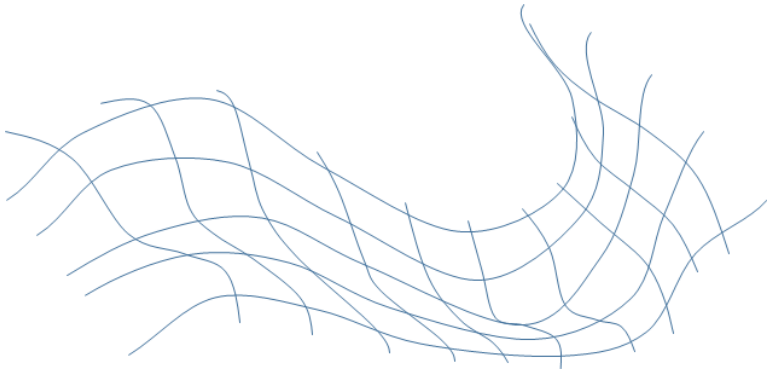
- Objectifs : réduire le volume en conservant l'information utile, améliorer le rapport signal/bruit, faciliter la visualisation, atténuer la malédiction de la dimension

1. **Sélection de variables** (*feature selection*) : choisir k variables (qui gardent leur signification) parmi les m initiales ; recherche combinatoire (C_m^k) $\rightarrow$ heuristiques (tri par critère individuel, ex. test du χ^2 ; sélection incrémentale ou décrémentale)
2. **Transformation de variables** (*feature extraction*) : construire de nouvelles variables, rarement interprétables, mais solution plus générale que la sélection
   - **Linéaire** : sous-espace linéaire de dimension k (ACP, AFD, ACM)
   - **Non linéaire** : variété (*manifold*) de dimension k (t-SNE, UMAP, autoencodeurs)

Sous-espaces linéaires et non linéaires



Sous-espace linéaire



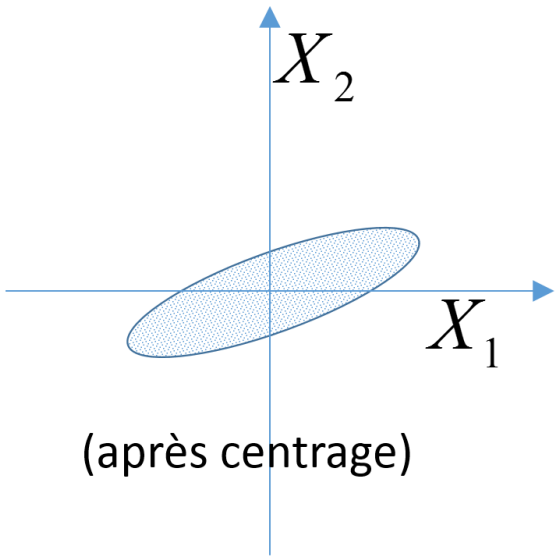
Sous-espace non linéaire

ACP : données, objectifs

- Données : N observations décrites par m variables quantitatives (matrice $\mathbf{R}$)
- Méthode exploratoire : cherche $k < m$ nouvelles variables (les **composantes principales**), combinaisons linéaires des variables initiales, en conservant le maximum de variance
- Utilisations pour les données massives
  - Visualiser en faible dimension l'organisation prépondérante des données
  - Éliminer des sous-espaces de faible variance (souvent assimilés à du bruit, parfois à tort) avant méthodes décisionnelles
  - Comprendre les relations entre (groupes de) variables initiales

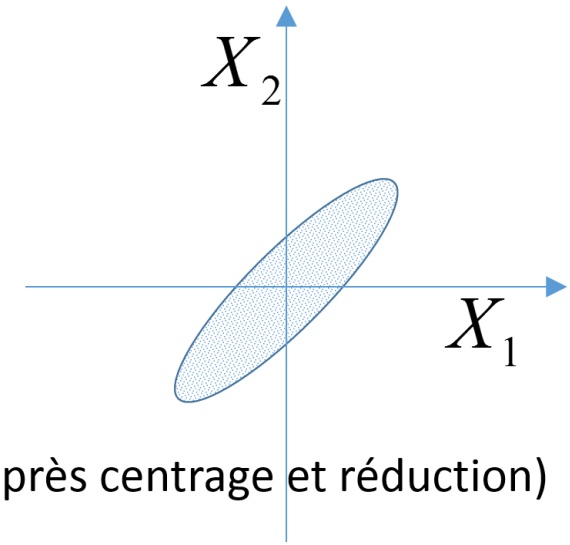
ACP centrée et ACP normée

- **ACP centrée** : centrage préalable des variables ($\rightarrow$ moyenne nulle) ; pour variables directement comparables, de même nature
- **ACP normée** (la plus courante) : centrage et réduction ($\rightarrow$ moyenne nulle, écart-type unitaire) ; pour variables de nature différente ou intervalles de variation très différents



(après centrage)

ACP centrée



(après centrage et réduction)

ACP normée

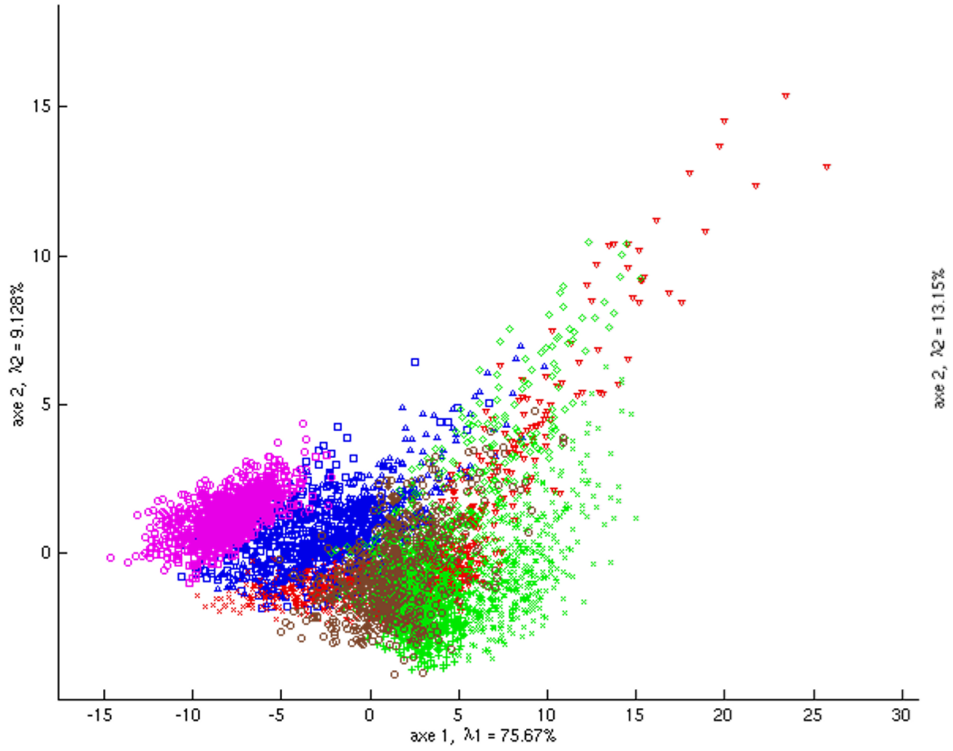
ACP : calcul

- Après centrage et éventuelle réduction, matrice $\mathbf{R} \rightarrow$ matrice $\mathbf{X}$
- Analyse nuage observations : les k vecteurs propres $\mathbf{u}_\alpha$ associés aux k plus grandes valeurs propres λ_α de $\mathbf{X}^T\mathbf{X}$: solutions de $\mathbf{X}^T\mathbf{X}\,\mathbf{u}_\alpha = \lambda_\alpha\,\mathbf{u}_\alpha,\ \alpha \in \{1, \ldots, k\}$
- Analyse nuage variables : les k vecteurs propres $\mathbf{v}_\alpha$ associés aux k plus grandes valeurs propres μ_α de $\mathbf{X}\mathbf{X}^T$: solutions de $\mathbf{X}\mathbf{X}^T\,\mathbf{v}_\alpha = \mu_\alpha\,\mathbf{v}_\alpha$
  - ACP centrée : $\mathbf{X}^T\mathbf{X}$ matrice des covariances empiriques
  - ACP normée : $\mathbf{X}^T\mathbf{X}$ matrice des corrélations empiriques
- $N \gg m$ en général $\Rightarrow$ travailler sur $\mathbf{X}^T\mathbf{X}$ $(m \times m)$ plutôt que $\mathbf{X}\mathbf{X}^T$ $(N \times N)$, on emploie les relations de transition pour les résultats de l'analyse des variables
- La matrice $\mathbf{X}^T\mathbf{X}$ étant symétrique, les axes principaux sont orthogonaux
- Complexité algorithmique :
  - Toutes les valeurs propres : $O(m^3)$
  - Les k plus grandes (cas courant en données massives) : algorithme itératif $O(Nmk)$
  - Données creuses (chaque variable $\neq 0$ pour au plus $p \ll N$ observations) : $O(pmk)$

ACP : exemple – nuage des observations

Données *textures* (LTIRF-INPG, projet ESPRIT III ELENA (No. 6891), groupe de travail ESPRIT ATHOS (No. 6620)

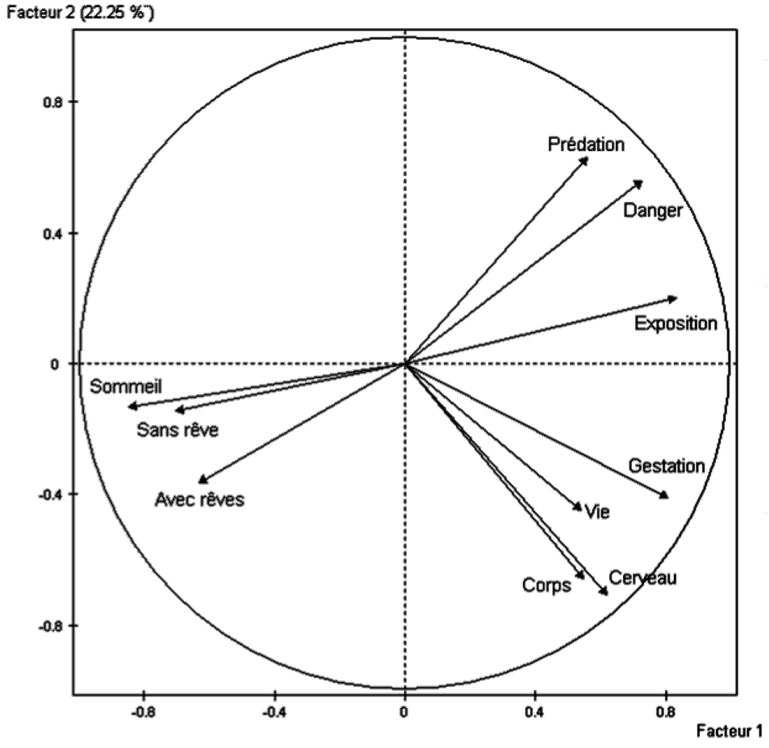
- $N = 5500$ descripteurs visuels, $m = 40$ variables, 11 classes
- Les couleurs montrent dans quelle mesure la projection sépare les classes (l'ACP ne tient pas compte des classes !)



ACP : exemple – nuage des variables

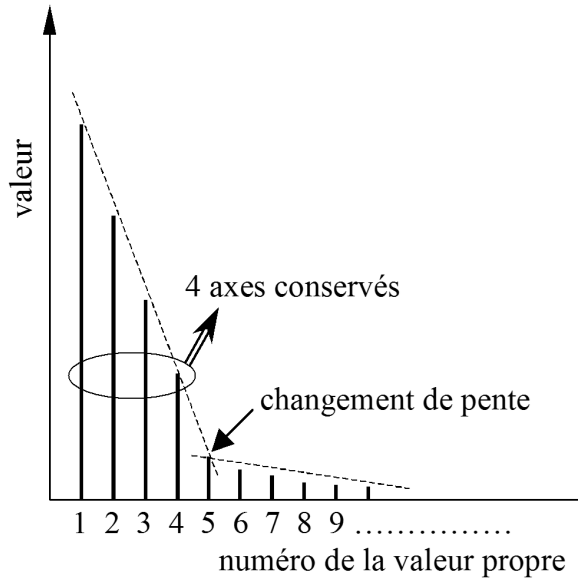
Données *Allison T., Cicchetti D., Sleep in mammals : ecological and constitutional correlates, Science, vol. 194, pp. 732-734.*

- $N = 62$ espèces, $m = 10$ variables (poids corps et cerveau, durée de vie et de gestation, durées sommeil, indices de prédation, etc.) ; ACP normée
- Trois groupes : « sommeil », « danger », « corps, cerveau, vie, gestation » (CCVG) ; forte anti-corrélation « sommeil » / « danger »



ACP : choix du nombre de composantes k

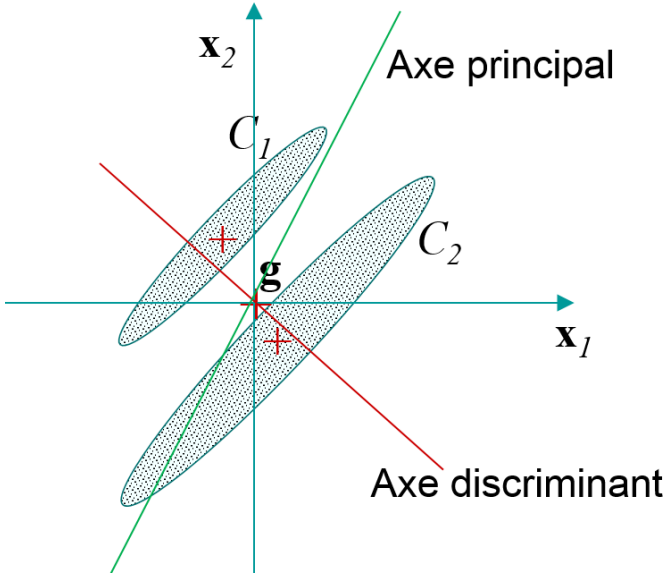
- **Analyse descriptive** : méthode du « coude » c.à.d. valeurs propres triées par ordre décroissant, on retient celles qui précèdent le premier coude marqué
- **Réduction du volume** : taux minimum imposé (ex. 85 %) d'inertie expliquée
- **Prétraitement décisionnel** : bon conditionnement de la matrice des covariances après réduction ou k traité comme hyperparamètre à optimiser



AFD : données, objectifs

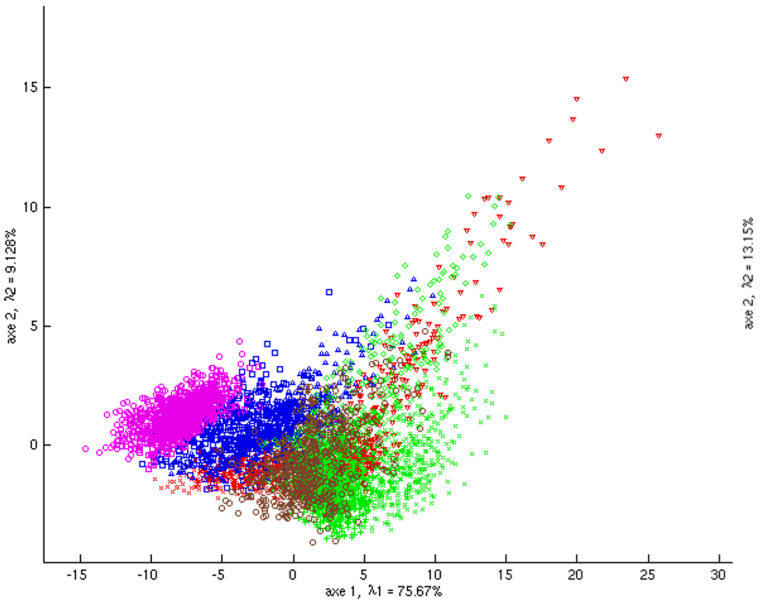
- Méthode descriptive et décisionnelle : en plus des m variables quantitatives, une variable nominale de classe $Y \in \{1, \ldots, q\}$
- Recherche de $k < q, k < m$ **facteurs discriminants** : sous-espace linéaire qui maximise la séparation entre classes
- Pour la réduction de dimension on s'intéresse à la composante descriptive

- ACP maximise la variance des projections, AFD maximise la séparation entre classes

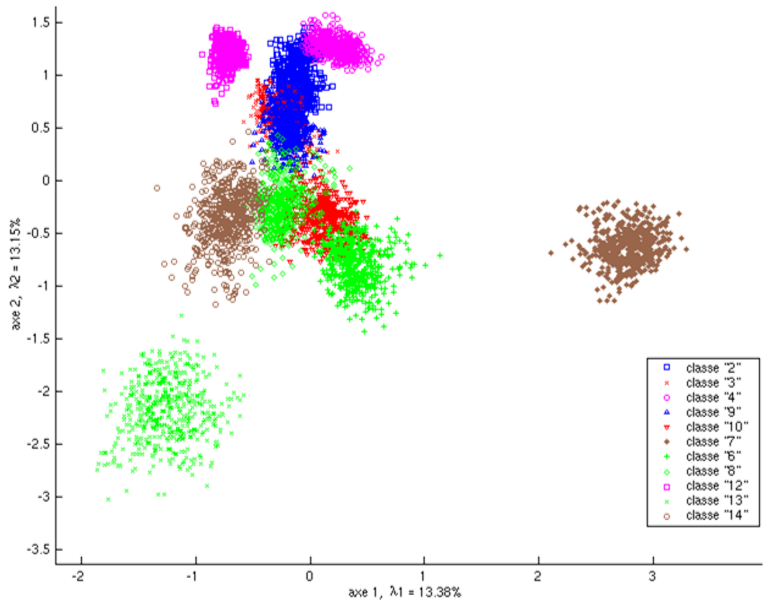


AFD *versus* ACP : illustration

Données *textures* ($N = 5500$, $m = 40$, $q = 11$) : la projection sur les axes discriminants sépare nettement mieux les classes que la projection sur les axes principaux



ACP



AFD

AFD : calcul

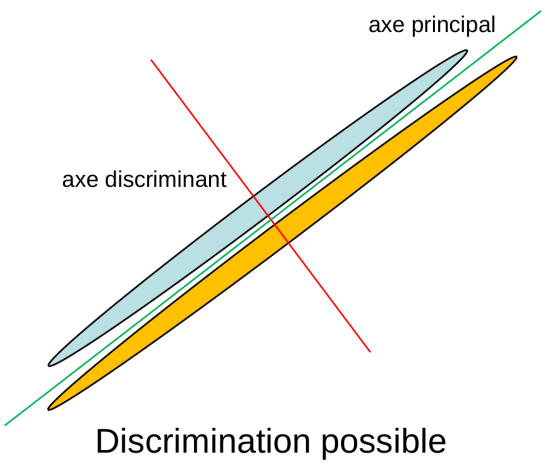
- Trois matrices de covariance
  - $\mathbf{E}$: covariance **inter-classes** (centres de gravité des q classes)
  - $\mathbf{D}$: covariance **intra-classes** (chaque classe centrée sur son centre de gravité)
  - $\mathbf{S}$: covariance **totale**, relation de Huygens $\mathbf{S} = \mathbf{E} + \mathbf{D}$
- Facteurs (axes) discriminants : vecteurs propres $\mathbf{u}_\alpha$ de l'équation généralisée

$$\mathbf{E}\,\mathbf{u}_\alpha = \lambda_\alpha\,\mathbf{S}\,\mathbf{u}_\alpha$$

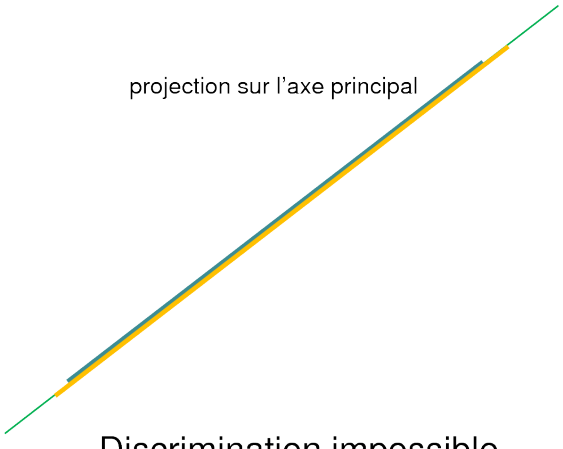
  si $\mathbf{S}$ inversible, résoudre plutôt l'équation simple $\mathbf{S}^{-1}\mathbf{E}\,\mathbf{u}_\alpha = \lambda_\alpha\,\mathbf{u}_\alpha$
- Plus la valeur propre associée est grande, plus le facteur est discriminant
- La matrice $\mathbf{S}^{-1}\mathbf{E}$ n'étant pas symétrique, les axes discriminants ne sont pas orthogonaux

AFD : limites

- Nombre de facteurs réduit : $k < q$ (car $\mathbf{E}$ est de rang $\leq q - 1$)
  - Avec seulement 2 classes : un seul facteur discriminant
- Si $\mathbf{S}$ n'est pas inversible ou est inversible mais mal conditionnée, employer l'ACP comme prétraitement de l'AFD (pour réduire la dimension et améliorer le conditionnement de $\mathbf{S}$) est tentant mais risqué
  - Les directions de faible variance pour l'ACP peuvent être très discriminantes pour l'AFD



Discrimination possible

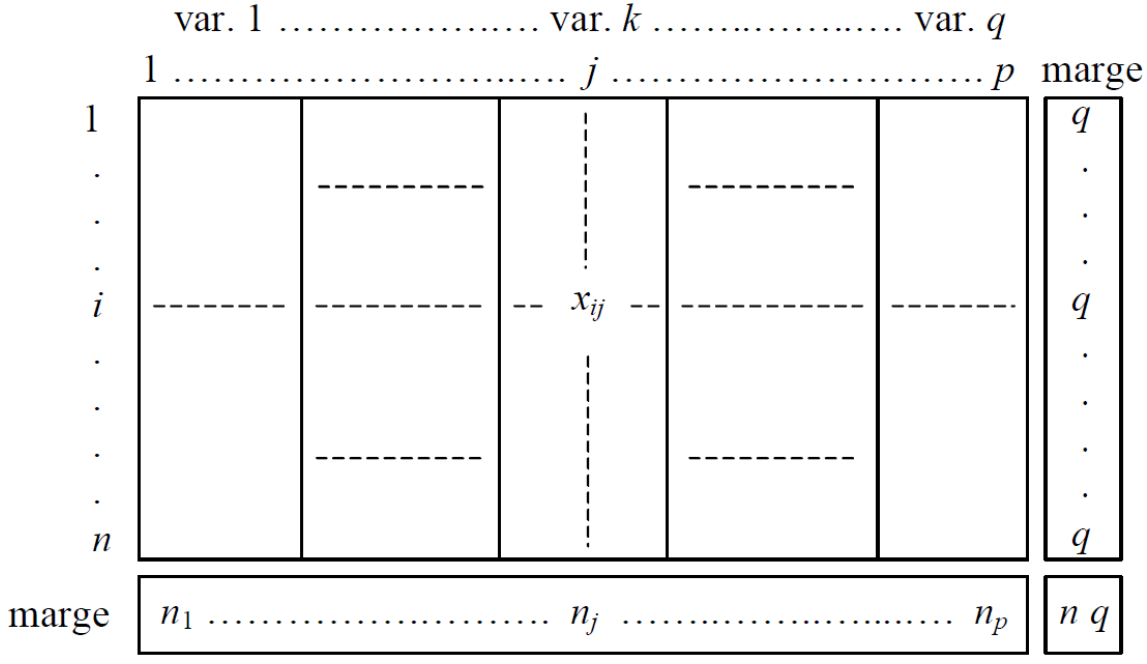


Discrimination impossible

$\rightarrow$ Préférer une régularisation : $\mathbf{S} \leftarrow \mathbf{S} + r\mathbf{I}_m$, avec r suffisamment élevé

ACM : données, objectifs

- Méthode exploratoire, données décrites par des variables **nominales** (à modalités)
- N observations, $q > 2$ variables nominales, représentées par un **tableau disjonctif complet** (TDC) : chaque modalité correspond à une colonne binaire



ACM : principe, utilisations

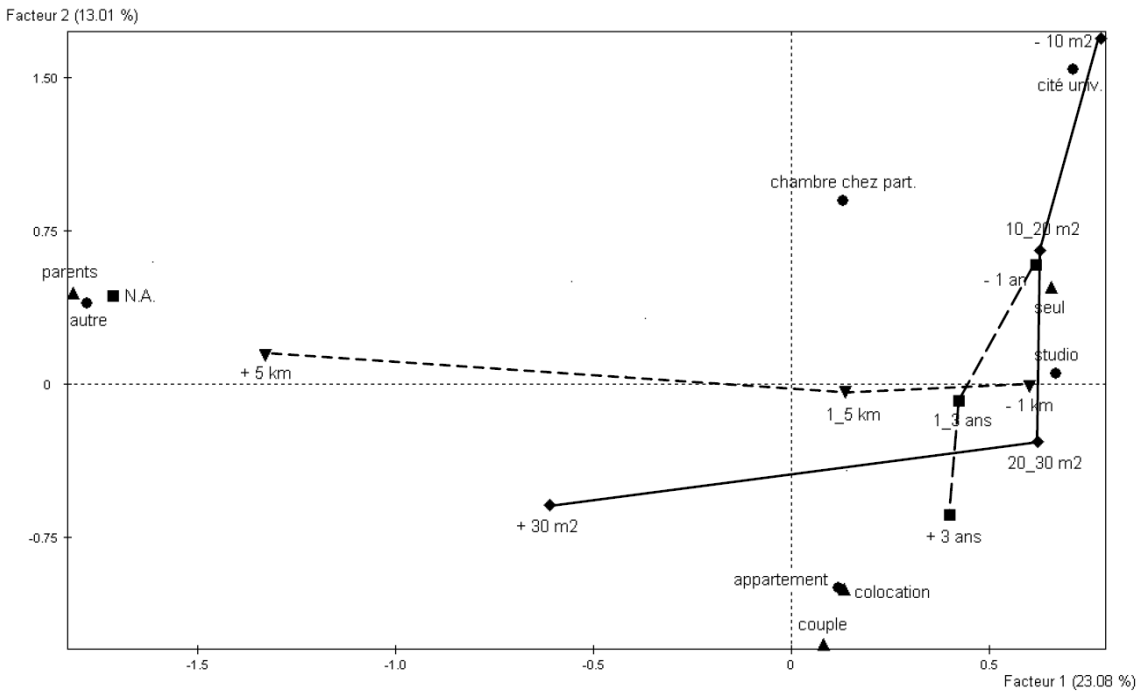
- Données massives : N élevé, on s'intéresse surtout à l'analyse des modalités
- Construction de k nouvelles variables quantitatives qui conservent un maximum de variance du nuage des modalités
  - Pondération de chaque modalité (colonne du TDC) par sa fréquence relative $\Rightarrow$ les modalités rares ont un faible impact sur l'analyse

- Utilisations
  - Résumer un grand nombre q de variables qualitatives par un faible nombre $k \ll q$ de variables quantitatives
  - Mettre en évidence les relations dominantes entre modalités
  - Intégrer des variables quantitatives (après discrétisation) pour détecter des relations non linéaires

ACM : exemple

Données issues de l'enquête « Les étudiants et la ville » réalisée en 2001 sous la direction de S. Denèfle à l'Université de Tours, voir [1]

- Cinq variables nominales (ou quantitatives discrétisées) : mode d'habitation, type de logement, ancienneté, distance à l'université, surface habitable
- Similarités entre projections de modalités :
  - De variables différentes : mêmes populations d'observations
  - D'une même variable (donc mutuellement exclusives) : populations similaires par rapport aux autres variables
- Oppositions fortes : « seul » *vs* « parents », « +5 km » *vs* « -1 km »



Plan du cours

Limites de `Pandas` à grande échelle

- `Pandas` : bibliothèque *DataFrame* de référence en Python, meilleur choix pour des données de taille modeste

- Limites au-delà de quelques dizaines (centaines ?) de Mo
  - **Exécution mono-thread** : n'exploite qu'un seul cœur
  - **Tout en mémoire** : le *DataFrame* entier doit tenir en RAM
  - **Copies mémoire coûteuses** : de nombreuses opérations créent des copies intermédiaires
  - **Format en lignes** (*row-major*) : peu efficace pour les opérations columnaires (agrégations, filtres) très fréquentes

→ À partir de ∼1 Go : temps d'exécution pénalisants, gestion mémoire difficile ⇒ intérêt de `Polars` et `Dask`

`Polars` : architecture

- `Polars` : bibliothèque *DataFrame* écrite en Rust, publiée à partir de 2020, adoption très rapide
- **Format en colonnes** (Apache Arrow) : adapté aux agrégations/filtres sur colonnes entières ; échange efficace avec d'autres outils Arrow (Spark, DuckDB)
- **Parallélisme multi-cœur automatique**, sans configuration par l'utilisateur
- **Exécution paresseuse** (*lazy evaluation*)
  - *Eager* : exécution immédiate (comme `Pandas`), pour l'exploration
  - *Lazy* : opérations empilées dans un **graphe de requête**, exécutées à l'appel de `collect()`, après passage par un optimiseur (*predicate/projection pushdown*)

Format Apache Arrow

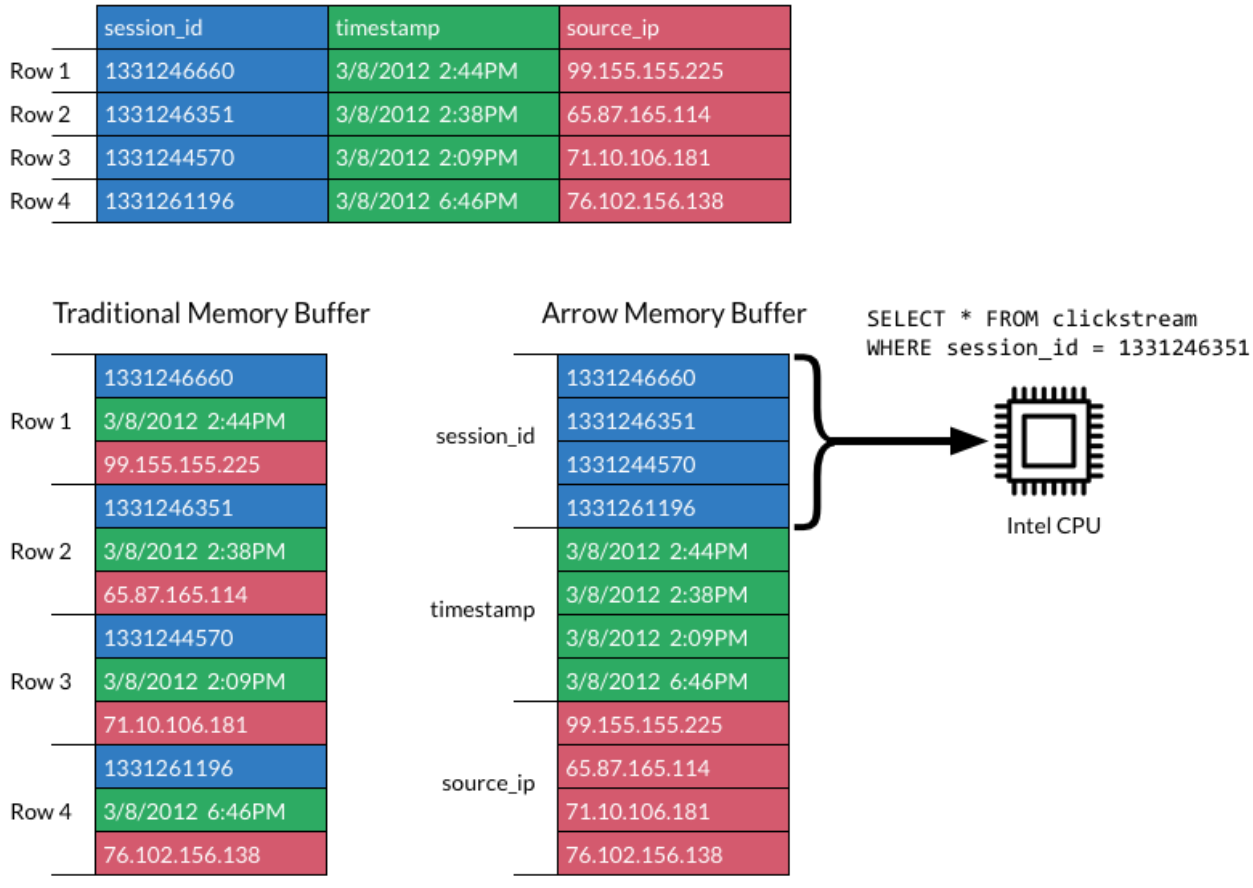


FIG. – Apache Arrow : plus facile de lire seulement les colonnes nécessaires, d'appliquer des opérations vectorielles, etc.

Polars : API principale

- Point d'entrée : `pl.DataFrame` (*eager*) ou `pl.LazyFrame` (*lazy*)

```
import polars as pl

# Mode eager : lecture immédiate en mémoire
df = pl.read_csv("data.csv")

# Mode lazy : lecture différée, optimisée
lf = pl.scan_csv("data.csv")

result = (
    lf
    .filter(pl.col("age") > 30)
    .select(["nom", "age", "revenu"])
    .collect()
)
```

Polars : comparaison syntaxique avec Pandas

Pandas	Polars (lazy)
`df[df["age"] > 30]`	`lf.filter(pl.col("age") > 30)`
`df[["nom", "age"]]`	`lf.select(["nom", "age"])`
`df.groupby("cat")["rev"].mean()`	`lf.group_by("cat").agg(pl.col("rev").mean`
`df.merge(df2, on="id")`	`lf.join(lf2, on="id", how="inner")`
`df.sort_values("rev", ascending=False)`	`lf.sort("rev", descending=True)`

- Sur de nombreuses tâches analytiques (filtrage, agrégation, jointure, tri) Polars est 5 à 20 fois plus rapide que Pandas selon les opérations et la taille des données

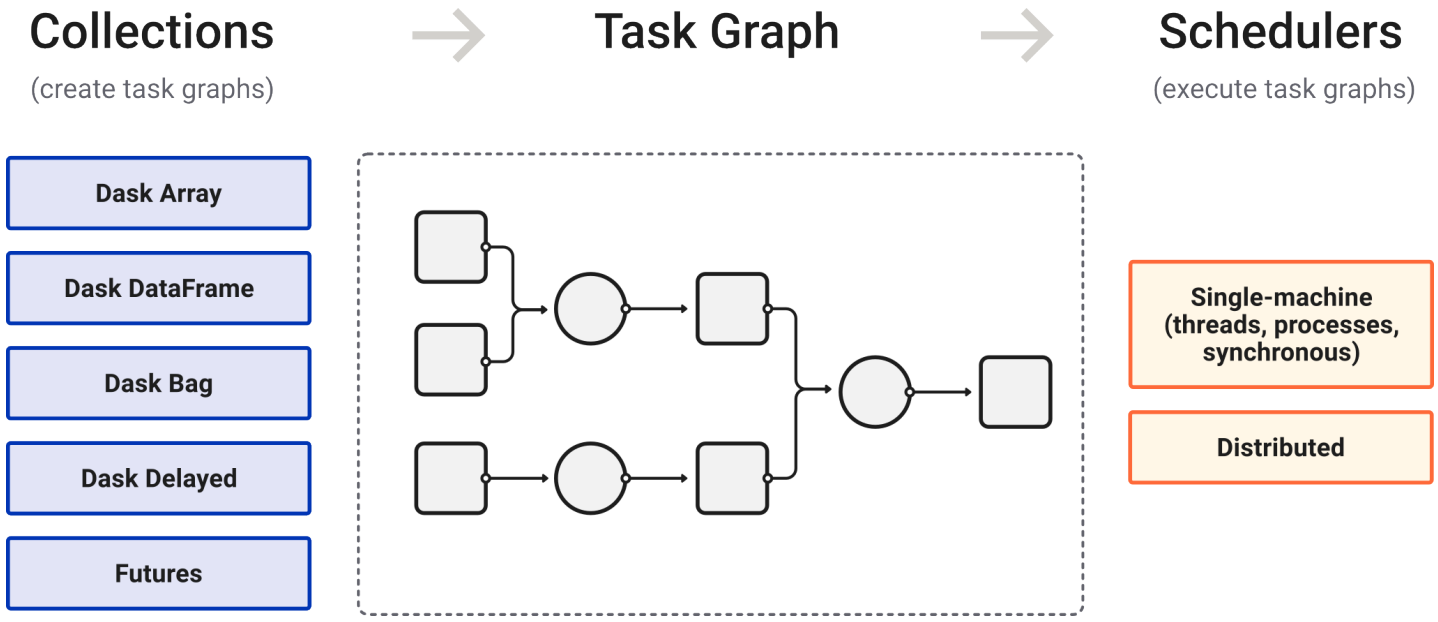
`Polars` : cas d'usage typiques

- Les données tiennent en RAM (de quelques Go à quelques dizaines de Go)
- Traitements analytiques fortement columnaires : filtrage, agrégation, jointure
- Priorité à la rapidité d'exécution sur une seule machine
- Pas nécessaire d'apprendre un nouveau paradigme de programmation distribué

→ `Polars` ne propose pas nativement de distribution sur plusieurs machines : pour ce besoin employer plutôt `Dask`, `Spark`, `Ray`, etc.

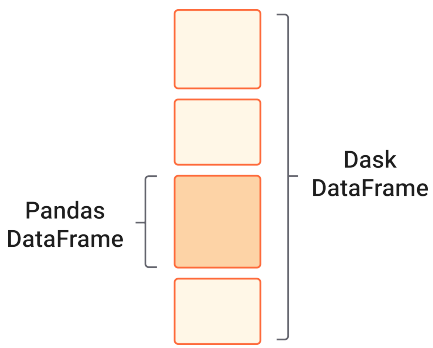
Dask : architecture

- Dask : bibliothèque Python native (depuis 2014) ; paralléliser et potentiellement distribuer des calculs utilisant les API NumPy/Pandas familières
- Collections partitionnées : traitées en parallèle sur les cœurs d'une machine, ou successivement si trop volumineuses pour la RAM, ou sur les machines d'un *cluster*
- Traitements représentés par un **graphe acyclique dirigé** (DAG), exécutés en mode *lazy* sous le contrôle d'un *scheduler* centralisé



`Dask` : DataFrame et *schedulers*

- Une `dask.DataFrame` est une collection de partitions, chaque partition étant une `pandas.DataFrame` ordinaire
- Partitionnement par plage d'index (*range*) ou par hachage (*hash*, nécessaire pour `join`/`groupby` efficaces)



- **Schedulers** : synchrone (débogage), *threads* (intra-machine, GIL [1] libéré), processus (intra-machine, code Python pur), distribué (cluster, tolérant aux pannes)

[1] Global Interpreter Lock

Dask : API

```
import dask.dataframe as dd

ddf = dd.read_csv("data_large_*.csv")   # lecture lazy
result1 = ddf[ddf["age"] > 30]["revenu"].mean()
print(result.compute())                 # déclenche l'exécution

result2 = (ddf.groupby("categorie")["revenu"]
            .agg(["mean", "max", "count"]).compute())

result3 = ddf.merge(ddf2, on="id_client").compute()
```

- `dask.array` : API NumPy distribuée pour tableaux qui ne tiennent pas en mémoire
- `dask.bag` : collections Python semi-structurées (opérations `map`, `filter`, `groupby`)

Dask en mode distribué

- Sur une seule machine : scheduler multi-threads ou multi-processus par défaut
- Sur *cluster* : `dask.distributed` fournit un **scheduler centralisé** (`Client`) et des **workers** distribués

```
from dask.distributed import Client

client = Client("scheduler-address:8786")  # cluster existant
# ou : client = Client()  # workers locaux automatiques

ddf = dd.read_csv("hdfs://data/*.csv")
result = ddf.groupby("cat")["val"].mean().compute()
```

- Tableau de bord (*dashboard*) accessible via navigateur web : suivi des tâches, mémoire, transferts réseau

Choisir entre `Dask` et `Polars`

- **Polars** préférable quand les données tiennent en RAM et la priorité est la vitesse d'exécution sur une seule machine
- **Dask** préférable quand
  - les données dépassent la RAM mais on ne dispose pas d'un *cluster* : données partitionnées, partitions traitées successivement, conservées sur disque entre étapes
  - déploiement souhaité sur *cluster* avec la même API que sur une seule machine (paralléliser du code `NumPy`/`Pandas` existant sans le réécrire)

Plan du cours

Synthèse : `Pandas`, `Polars`, `Dask`

	Pandas	**Polars**	**Dask**
Langage interne	C / Python	Rust	Python
Format mémoire	Lignes (NumPy)	Colonnes (Arrow)	Partitions Pandas
Multi-cœur	Non	Oui (automatique)	Oui (configurable)
Données $>$ RAM	Non	Partiel (scan *lazy*)	Oui
Distribution cluster	Non	Non	Oui (`dask.distributed`)
Exécution paresseuse	Non	Oui (LazyFrame)	Oui (`compute()`)
Compatibilité Pandas	–	Partielle	Très forte
Vitesse (données en RAM)	⋆⋆	⋆ ⋆ ⋆ ⋆ ⋆	⋆ ⋆ ⋆
Prise en main	⋆ ⋆ ⋆ ⋆ ⋆	⋆ ⋆ ⋆⋆	⋆ ⋆ ⋆⋆

- Cas d'usage principal : `Pandas` $\rightarrow$ volumes modérés, exploration ; `Polars` $\rightarrow$ analyses rapides, quelques Go en RAM ; `Dask` $\rightarrow$ données $>$ RAM, parallélisation de code `Pandas` existant
- Ces bibliothèques évoluent constamment $\Rightarrow$ les écarts relatifs de performance peuvent changer significativement !

Références I


M. Crucianu, J.-P. Asselin de Beauville, and R. Boné.
Méthodes d'analyse factorielle des données : méthodes linéaires et non linéaires.
Hermès, Paris, 2004.


Y. Tillé.
Théorie des sondages.
Dunod, Paris, 2001.