| **Type de données** | **Fournisseur direct** | **Source réelle** | **Ownership** | **Remarques** |
| --- | --- | --- | --- | --- |
| **Prépaiements** | DFIN | SGRF DAT / ALM | SGRF/ALM | Fournies pour RACER |
| **Origination volume** | DFIN | SGRF DAT / ALM | SGRF/ALM | Fournies pour RACER |
| **Pertes (PES, PED)** | RISQ CRE / ERM | RISQ CRE / ERM | RISQ | Données certifiées par RISQ |
| **Encours sains/défaut** | DFIN (extraction SAFIR) | SAFIR (RISQ) | RISQ | DFIN n’est **pas propriétaire** de ces données |
| **Provisions** | DFIN (extraction SAFIR) | SAFIR (RISQ) | RISQ | À valider avec RISQ CRE/ERM |

# Processus de production des datapacks

Le nouveau processus de production des datapacks est automatisé et simplifié, grâce à une solution **Python** qui intègre :

* **Collecte des données** : automatisation de la collecte des données nécessaires à la production des datapacks, en utilisant les données préparées par SAS.
* **Traitement des données** : traitement des données collectées directement dans l’environnement Python.
* **Analyse de la qualité des données** : une phase d’analyse est réalisée afin de détecter les valeurs manquantes, les doublons, les outliers, ainsi que d’appliquer des sanity checks.
* **Calcul des indicateurs** : utilisation de Python pour le calcul des indicateurs, remplaçant ainsi le besoin d'un programme R.
* **Automatisation des gap analyses** : mise en place d’un système pour automatiser l’analyse des écarts entre les transactions N et N-1, facilitant l’évaluation de la performance et des changements au fil du temps.

## Diagramme des classes

La procédure de production est modélisée à l’aide d’un **diagramme des classes**, afin de structurer les différents objets manipulés (Datapack, Indicateur, Donnée, Qualité des données, etc.) et leurs relations.

Ce modèle objet permet de clarifier les flux, d’assurer la traçabilité des calculs, et de faciliter les évolutions futures de l’outil.

Les principales classes sont les suivantes :

1. **Classe Data** :  
   Gère la collecte et le pré-traitement des données brutes nécessaires au calcul des indicateurs. Elle lit les sources (SAS, fichiers plats, etc.) et prépare les jeux de données.
2. **Classe DataQuality** :  
   Analyse les données pour détecter les valeurs manquantes, doublons et outliers. Elle applique des règles de nettoyage et produit un rapport de qualité des données utilisé en audit ou en supervision.
3. **Classe Indicator** :  
   Classe générique pour le calcul des indicateurs (PD, LGD, etc.). Elle fournit une structure commune à tous les types d’indicateurs.
4. **Classes filles de Indicator** (ex. : MigrationMatrix, DefaultRate) :  
   Implémentent les logiques de calcul spécifiques à chaque indicateur, en héritant de la classe Indicator.
5. **Classe Datapack** :  
   Regroupe les indicateurs calculés et génère les datapacks au format investisseur, ainsi que la gap analysis entre N-1 et N.
6. **Classe Process** :  
   Orchestre l’ensemble du processus en exécutant les étapes dans l’ordre : collecte, traitement des données, contrôle qualité, calcul, puis génération des livrables.

## Collecte des données

La **création des bases de données dans SAS** ainsi que l’**export des bases SAS vers STARK via un bucket S3** sont décrits dans la note de procédure disponible ici : *Lien note de procédure (à insérer ou à compléter).*

Les données nécessaires à la production des datapacks sont ensuite récupérées automatiquement par l’outil Python, conformément aux paramètres définis dans la partie **Inputs** (voir section 4). La collecte est orchestrée par la **classe Data**, qui interagit avec les fichiers de configuration (*variable\_mapping*) et les répertoires d’entrée (*inputs/pd, inputs/lgd,* etc.).

Cette étape garantit que seules les données correctement préparées et attendues sont intégrées au pipeline d’automatisation.

**Classe Data**

La classe Data est utilisée pour collecter toutes les données nécessaires au calcul des indicateurs de risque du datapack. Ces données sont localisées dans un bucket Amazon S3.

Elle contient trois méthodes principales :

* **load\_data\_from\_folder**  
  Cette méthode permet de charger les données, soit depuis un dossier géré, soit depuis des datasets importés dans S3. Elle retourne un dictionnaire de DataFrames, où chaque clé correspond au nom d’un fichier de données, et chaque valeur à son DataFrame associé.  
  *Arguments* :
  + *source\_type* : "folder" ou "dataset"
  + *source\_names* : None si source\_type = "folder", ou une liste de noms de datasets
  + *folder\_id* : identifiant du dossier source (fourni par Dataiku)
  + *mapping\_file\_name* : nom du fichier de mapping Excel
* **collect\_data**  
  Cette méthode extrait les données du dictionnaire produit par load\_data\_from\_folder, et construit des DataFrames PySpark en utilisant les règles définies dans le fichier de mapping.  
  *Argument* :
  + *mapping\_file\_name* : nom du fichier Excel contenant les correspondances colonnes/bases.
* **retrieve\_data**  
  Cette méthode permet de récupérer tous les fichiers CSV ou Excel d’un dossier spécifié.  
  *Argument :*
  + *folder\_path* : chemin du dossier cible

*Retour :*  
Un dictionnaire avec en clé le nom du fichier (sans extension), et en valeur le DataFrame Spark correspondant.

## Traitement des données

Le traitement des données collectées s’effectue directement dans l’environnement **Python** à l’aide de **PySpark**, après leur récupération depuis le S3 ou les dossiers SAS exportés.

Les principales étapes de traitement sont :

* **Filtrage du périmètre** : seules les lignes correspondant au périmètre titrisé sont conservées (filtrage par code portefeuille bâlois, CTR, rating system, etc.).
* **Normalisation et renommage des colonnes** : les noms des variables sont harmonisés conformément au fichier variable\_mapping, afin de garantir une cohérence des données en entrée des calculs.
* **Jointures entre les différentes bases** : les données issues des différentes sources (PD, LGD, etc.) sont jointes entre elles selon les clés définies dans le mapping.
* **Pré-traitement spécifique** : des règles métiers sont appliquées pour certaines variables (ex. : gestion des dates, ajustements spécifiques à la titrisation, etc.).
* **Préparation à l’analyse qualité et au calcul** : les données sont préparées dans un format structuré pour être utilisées dans les modules de contrôle qualité (DataQuality) puis dans le calcul des indicateurs (Indicator).

Ce traitement vise à obtenir des jeux de données propres, alignés avec les exigences du reporting titrisation, et compatibles avec les templates de production.