Les principes FAIR adaptés aux logiciels

Pour accéder à la ressource : https://doranum.fr/enjeux-benefices/les-principes-fair-adaptes-aux-logiciels_10_13143_0365-xz76/

Date de publication : 17/05/2024

Sommaire

BIENVENUE

Bienvenue dans cette ressource sur les principes FAIR adaptés aux logiciels et composée de 7 chapitres.

Les chapitres peuvent être suivis de façon linéaire et progressive, mais aussi de manière fragmentée. Vous pouvez consulter uniquement les parties qui vous intéressent.

1. Introduction

« Les principes FAIR ont initialement été pensés pour les données de la recherche. Cependant, dans la plupart des recherches qui produisent des données, les logiciels et les codes sont des résultats de recherche aussi importants que les données.
Il est donc important d'adapter les principes FAIR aux logiciels dans l'objectif d'améliorer leur partage et leur réutilisation. »
Institut Pasteur. Comment rendre son logiciel FAIR ? 15 mars 2023.
https://openscience.pasteur.fr/2023/03/15/comment-rendre-son-logiciel-fair/

Le groupe de travail FAIR for Research Software (FAIR4RS) a adapté les principes FAIR pour la gestion des données scientifiques aux logiciels de recherche. De nombreux principes FAIR peuvent être directement appliqués aux logiciels de recherche en traitant les logiciels et les données comme des objets de recherche numériques similaires. Cependant, les caractéristiques spécifiques des logiciels - telles que leur exécutabilité, leur nature composite, leur évolution et le versionnement continus - rendent nécessaires la révision et l'extension des principes.

Les logiciels de recherche comprennent les fichiers de code source, les algorithmes, les scripts, les flux de calcul et les exécutables qui ont été créés au cours du processus de recherche ou à des fins de recherche.

Dans les chapitres suivants, les questions en bleu correspondent aux 17 questions à se poser pour vérifier l'adaptation aux principes FAIR d'un logiciel. Les numéros (F01 à R17) correspondent à ces 17 questions. Elles font l'objet d'une check-list réutilisable jointe à cette ressource.

Chue Hong Neil, Breitmoser Elena, Antonioletti Mario, Davidson Joy, Garijo Daniel, Gonzalez-Beltran Alejandra, Gruenpeter Morane, Huber Robert, Jonquet Clément, Priddy Mike, Shepeherdson John, Verburg Maaike, Wood Chris. D5.2 - Metrics for automated FAIR software assessment in a disciplinary context (1.0 - DRAFT not yet approved by the European Commission). Zenodo. 27 octobre 2023. https://doi.org/10.5281/zenodo.10047401

2. Principe F – Facile à trouver

Les logiciels et les métadonnées qui leur sont associées sont **faciles à trouver**, tant pour les humains que pour les systèmes informatiques si un **identifiant pérenne** est attribué et que les **métadonnées** sont **riches**.

- **F1. Un identifiant unique et pérenne est-il attribué au logiciel ? (F01)**
  - **F1.1. Des identifiants distincts sont-ils attribués aux différents composants du logiciel ? (F02)**
    Les composants représentent des niveaux de granularité (par exemple une bibliothèque logicielle et une fonction dans cette bibliothèque). La relation entre ces composants doit être intégrée dans les métadonnées associées.
    Par exemple, l'archive universelle Software Heritage attribue des identifiants uniques et pérennes (Software Hash IDentifiers, SWHIDs) aux différents « objets » logiciels.

Logo SWHID

  - **F1.2. Des identifiants distincts sont-ils attribués aux différentes versions du logiciel ? (F03)**
    La relation entre les versions doit être intégrée dans les métadonnées associées. Cela permet de mieux comprendre l'évolution du code, son auteur, sa propriété, sa description et son objectif.

- **F2. Le logiciel est-il décrit à l'aide de métadonnées riches qui aident à définir son objectif ? (F04)**
  Les logiciels nécessitent des métadonnées descriptives pour faciliter l'indexation, la recherche et la découverte. Ces métadonnées doivent respecter les standards communautaires et utiliser des vocabulaires contrôlés.
  Les métadonnées de développement aident-elles à définir son statut ? (F05)
  Le logiciel inclut-il des métadonnées sur les contributeurs et leurs rôles ? (F06)

- **F3. Les métadonnées incluent-elles clairement et explicitement l'identifiant du logiciel qu'elles décrivent ? (F07)**
  Elles décrivent comment le logiciel peut être obtenu.

- **F4. Les métadonnées doivent être FAIR, consultables et indexables.**
  Cela améliore la possibilité de trouver le logiciel en facilitant la recherche et l'indexation par d'autres. Cela permet aux métadonnées d'être publiées ou récoltées dans un registre, un catalogue, un entrepôt ou un moteur de recherche. Les métadonnées FAIR permettent et encouragent également la citation des logiciels de recherche.

3. Principe A - Accessible

Les logiciels et leurs métadonnées peuvent être rendus **accessibles** via des protocoles normalisés.

- **A1. Le logiciel est-il développé dans un entrepôt de code / une forge qui utilise des protocoles de communication normalisés ? (A09)**
  Le logiciel est accessible par son identifiant à l'aide d'un protocole de communication normalisé.
  - **A1.1. Le protocole est ouvert, libre et universellement implémentable**.
    L'accessibilité du logiciel ne doit pas dépendre d'outils ou de méthodes de communication spécialisés ou propriétaires. Le plus souvent, des protocoles techniques de communication normalisés sont utilisés pour accéder aux logiciels, tels que HTTPS.
  - **A1.2. Le protocole permet une procédure d'authentification et d'autorisation, si nécessaire**.

- **A2. Le logiciel dispose-t-il d'une notice de métadonnées publique, librement accessible et pérenne ? (A08)**
  Les métadonnées sont accessibles, même lorsque le logiciel n'est plus disponible. Le dépôt du code dans HAL permet d'obtenir cette « notice de métadonnées ».

4. Principe I - Interopérable

Le logiciel utilise-t-il des API ouvertes documentées de manière à ce que leurs fonctionnalités puissent être comprises par les systèmes informatiques ? (I11)
Les logiciels **interagissent** avec d'autres logiciels en échangeant des données et/ou des métadonnées et/ou en interagissant par le biais d'API (Application Programming Interface), décrites par des standards.

- **I1. Les formats des données traitées ou produites par le logiciel sont-ils ouverts et la documentation fait-elle référence à ces formats ? (I10)**
  Les logiciels lisent, écrivent et échangent des données d'une manière qui respecte les standards communautaires pertinents pour le domaine.

- **I2. Le logiciel inclut-il des références qualifiées à d'autres objets qui soutiennent son utilisation ? (R12)**
  Exemples : fichiers de paramètres nécessaires à l'exécution de certaines applications.

5. Principe R - Réutilisable

Les logiciels sont à la fois **utilisables** (peuvent être exécutés) et **réutilisables** (peuvent être compris, modifiés, construits ou incorporés dans d'autres logiciels).

- **R1. Le logiciel décrit-il ce qui est nécessaire pour l'utiliser ? (R13)**
  Le logiciel est-il accompagné de scénarios de test démontrant son fonctionnement ? (R14)
  Le logiciel est décrit avec une pluralité d'attributs précis et pertinents.
  Exemples : exigences, imports, bibliothèques… nécessaires pour compiler et exécuter le logiciel.
  - **R1.1. Le code source du logiciel inclut-il des informations claires et accessibles sur la licence du logiciel et de tout logiciel externe intégré ? (R15)**
  - **La notice de métadonnées du logiciel contient-elle des informations sur la licence ? (R16)**

- **R1.2. Le logiciel inclut-il des informations détaillées de provenance qui décrivent le développement du logiciel ? (R17)**
  Clarification de pourquoi et comment le logiciel a vu le jour, de qui a contribué à quoi, quand et où.

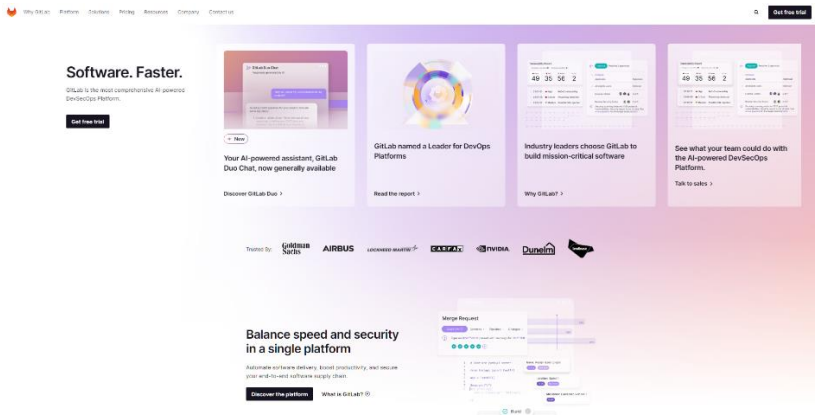
- **R2. Le logiciel inclut-il des références qualifiées à d'autres logiciels ? (R12)**

- **R3. Le logiciel respecte les standards communautaires pertinents pour le domaine.**

- *Chue Hong Neil P., Katz Daniel S., Barker Michelle, Lamprecht Anna-Lena, Martinez Carlos, Psomopoulos Fotis E., Harrow Jen, Castro Leyla Jael, Gruenpeter Morane, Martinez Paula Andrea, Honeyman Tom et al. RDA FAIR4RS WG. FAIR Principles for Research Software (FAIR4RS Principles). Zenodo. 24 mai 2022.* *https://doi.org/10.15497/RDA00068*
- *Institut Pasteur. Comment évaluer la conformité d'un logiciel avec les principes FAIR ? 4 février 2024.* *https://openscience.pasteur.fr/2024/02/14/comment-evaluer-la-conformite-dun-logiciel-avec-les-principes-fair/*

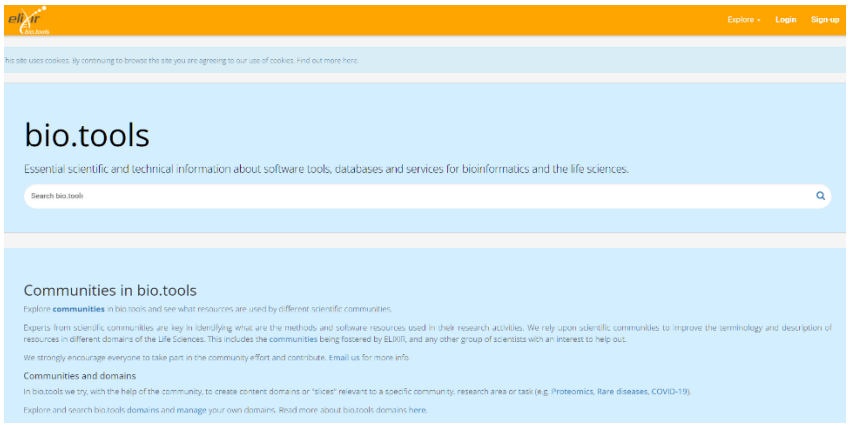
6. Exemples de bonnes pratiques pour rendre son logiciel FAIR

- Durant le projet, pour favoriser les collaborations, utiliser une forge publique reconnue par la communauté, avec un système de contrôle de version (par exemple Github ou Gitlab).



Gitlab : https://about.gitlab.com/

- Enregistrer le logiciel dans un répertoire reconnu par la communauté (par exemple Bio.tools pour les sciences de la vie) et le décrire avec des métadonnées riches.

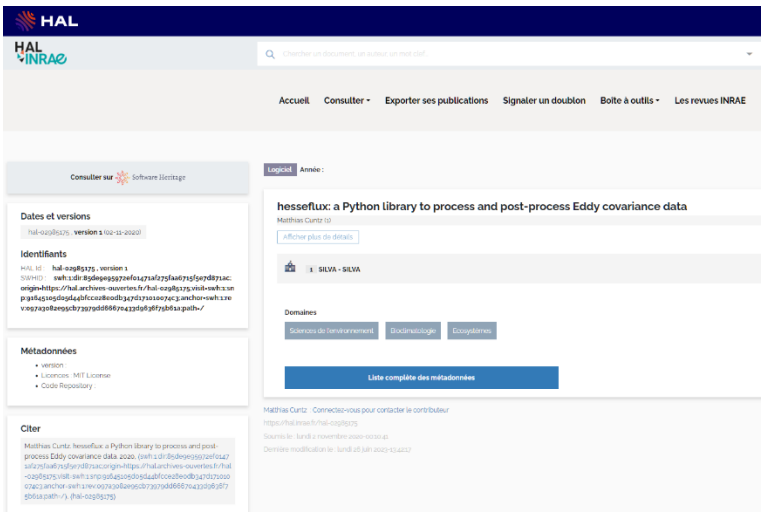


Biotools : https://bio.tools/

- Associer au logiciel une licence ainsi qu'une documentation claire : par exemple, la licence de logiciel libre GNU GPL (GNU General Public License ou licence publique générale GNU) ou la licence de logiciel libre CeCILL-B. Cette dernière a été créée conjointement par le CEA, le CNRS et l'INRIA (en savoir plus).

Logo licence GPL

- Rendre son logiciel citable, par exemple en le déposant dans l'archive ouverte HAL. L'archive universelle Software Heritage et HAL assurent l'accessibilité et la conservation des codes sources en libre accès, tout en gérant les versions.



Exemple de code source déposé dans HAL : https://hal.science/hal-02985175

Institut Pasteur. Comment rendre son logiciel FAIR ? 15 mars 2023. https://openscience.pasteur.fr/2023/03/15/comment-rendre-son-logiciel-fair/

7. Webographie

- *Chue Hong Neil, Breitmoser Elena, Antonioletti Mario, Davidson Joy, Garijo Daniel, Gonzalez-Beltran Alejandra, Gruenpeter Morane, Huber Robert, Jonquet Clément, Priddy Mike, Shepeherdson John, Verburg Maaike, Wood Chris. D5.2 - Metrics for automated FAIR software assessment in a disciplinary context (1.0 - DRAFT not yet approved by the European Commission). Zenodo. 27 octobre 2023. https://doi.org/10.5281/zenodo.10047401*
- *Chue Hong Neil P., Katz Daniel S., Barker Michelle, Lamprecht Anna-Lena, Martinez Carlos, Psomopoulos Fotis E., Harrow Jen, Castro Leyla Jael, Gruenpeter Morane, Martinez Paula Andrea, Honeyman Tom et al. RDA FAIR4RS WG. FAIR Principles for Research Software (FAIR4RS Principles). Zenodo. 24 mai 2022. https://doi.org/10.15497/RDA00068*
- *Institut Pasteur. Comment évaluer la conformité d'un logiciel avec les principes FAIR ? 4 février 2024. https://openscience.pasteur.fr/2024/02/14/comment-evaluer-la-conformite-dun-logiciel-avec-les-principes-fair/*
- *Institut Pasteur. Comment rendre son logiciel FAIR ? 15 mars 2023. https://openscience.pasteur.fr/2023/03/15/comment-rendre-son-logiciel-fair/*