APEX Enfant
Compte rendu construction base BIDS

François Ramon & Belen Azofra

Objectif : Création d'une base de données IRM respectant le formalisme BIDS pour le groupe enfant de l'étude APEX.

Outils :
BIDS Website : https://bids.neuroimaging.io/
dcm2niix: https://github.com/rordenlab/dcm2niix
dicm2nii: https://github.com/xiangruili/dicm2nii
bidscoin : https://bidscoin.readthedocs.io/en/latest/preparation.html

Les codes de conversion : https://github.com/FRramon/bids_apex_enf

Versions logiciels/paquets:
- bidscoin 4.4.0
- dcm2niix :Chris Rorden's dcm2niiX version v1.0.20240202 (JP2:OpenJPEG) (JP-LS:CharLS) Clang14.0.3 ARM (64-bit MacOS) v1.0.20240202
- dicm2nii: v2023.02.23
- python 3.11.5
- Matlab R2024b

Liste des scripts python
- 1_sort_sequences.py
- 2_create_raw_structure.py
- 3_copysource_data.py
- 4_convert_dot_stop.py
- 5_correct_runs_enrich_json.py
- 6_rename_participants.py
- bidsmap.yaml

Table des matières

DESCRIPTION GENERALE

Conversion d'une base de données IRM brute avec acquisition au format Philips REC/PAR, en base BIDS avec images au format nii.gz / json.

I. Définition des sujets : 1_sort_sequences.py

DESCRIPTION

Récupération d'un **dossier "sub-enf"** contenant des sous dossiers appelé par les noms apex (adoci002, etc), et contenant une acquisition particulière ex:

sub-enfca101_ses-post_mri-1457529310-1-01VALAL-3DT1-3-1
|--- 1-01VALAL-3DT1.PAR
|--- 1-01VALAL-3DT1.REC

Nature des données : Fichiers REC/PAR.

Pour chaque fichier PAR dans sub-enf, lecture des métadonnées du fichier et écriture dans un csv de :

- *protocol name*
- *patient name*
- *StudyDate*
- *StudyTime*

Les patients names sont par exemple : 1-01VALAL, 2-08MORJU …

On n'utilise à aucun moment le nom du dossier ou le nom de fichier. Une vérification préalable des métadonnées DICOM a montré que les DICOM présents

dans /source/sub-enf existent au format REC/PAR dans le même dossier. **La conversion de la base de données ne se fait donc qu'à partir des données REC/PAR**.

LISTE DES VARIABLES

source_dir : pointe vers source/sub-enf
docs_dir : pointe vers un dossier recevant les csv

OPTIONS

Bool option	Default	Description
sortsequences	True	Identifier si un id de participant est associé à plusieurs "patient_name"
delete_change	True	Supprime les données copiées en 2024 du répertoire sub-enf (pare feu copies)

II. Création d'une structure brute 2_create_raw_structure.py

DESCRIPTION

La première étape est la **correction des noms** : mauvaise orthographe, majuscule, point, de manière à suivre cette syntaxe :

- Premier caractère = appartenance au groupe i
- tiret
- deux chiffres : numéro de sujet au sein du groupe
- Cinq lettres : identifiant sujet

Ex : 1.01 VALAL devient : 1-01VALAL

Cas particulier : quelques renommages manuels dans le code

Nom incorrect	Nom correct
1.06gcema	1-06GUEMA
1-43HIUEMA	1-43HUEMA
2-010CHAPA	2-10CHAPA
apex027	2-09MORJU
4-06MALAX	4-06HALAX

Pour chaque sujet unique identifié dans le champ “patient name” :

- **Création d’un dossier** portant le nom du patient.
- Dans ce dossier, création d’un **fichier csv** regroupant l’adresse de chaque fichier PAR/REC pour lequel “patient name” correspond au nom du patient, ainsi que les colonnes suivantes : protocol name, StudyDate, et StudyTime.

Dans le csv ainsi généré pour chaque dossier patient :

- Identification des dates uniques et **comptage des acquisitions** pour chaque date.
- En général, le **nombre de dates uniques correspond au nombre de sessions** (pré, post, post diff). Ces sessions sont temporairement nommées 1, 2, et 3.
- Un cas particulier a été rencontré : un sujet avait 4 sessions uniques (4VALMA) car une acquisition T1 post avait été refaite (confirmation dans le cahier d’acquisition). D’autres sujets présentaient des incohérences, comme des sessions inexistantes (par exemple, 3 dates uniques, mais une seule acquisition pour l’une des dates). Ces problèmes ont été résolus individuellement.

Une fois les “vraies” sessions identifiées, elles ont été nommées **ses-00, session ses-01, et ses-02, par ordre de date croissante**.

*Règles de renommage des sessions **1,2, 3 en pré, post, postdiff:***

- ➔ La session 0 est toujours pré
- ➔ La session 1 peut être post, ou postdiff, cela à été décidé s' il y avait un écart >45j entre ses-00 et ses-01.
- ➔ Si il y a trois sessions c’est forcément pré/post/post/diff
- ➔ Si il n’y a qu’une session : c’est forcément pré

Dans chaque dossier sujet, **on trouve les sous-dossiers ses-pre, ses-post, et ses-postdiff** si les acquisitions correspondantes ont eu lieu. Un **fichier csv** est également créé dans chaque dossier de session, contenant les fichiers REC/PAR associés à ce sujet et à cette session, ainsi que le nom de chaque fichier.

Un fichier **CSV récapitulatif** est généré dans le répertoire /docs, résumant les acquisitions trouvées pour chaque participant et chaque session.

LISTE DES VARIABLES

raw_structure_dir : pointe vers un dossier recevant la structure brute
docs_dir : pointe vers le dossier contenant les csv de séquences

OPTIONS

Bool option	Default	Description
checkdoubles	False	Identifier si un id de participant est associé à plusieurs “patient_name”
correctNames	True	Corriger les noms de sujets. Pour la nomenclature type 1-01VALAL
getSequences_persubject	True	identifier le nombre de sessions
createSes	True	Attribuer un nom de session (pre/post/postdiff) à la place de ses-00,ses-01,ses-02
rename_session	True	Copier bidsmap depuis code dir dans rawdata/code/bidsoin et exécution de bidscoin
createSes_seq	True	Renommer sequence_ses-.csv en suivant le nouveau nom de session
createSummary	True	Création d’un csv regroupant les aquisitions REC/PAR présentes pour chaque sujet/session

III. Création de source_data et conversion BIDS : 3_copysource_data.py

DESCRIPTION

La première étape consiste à **copier les données REC/PAR** dans un nouveau répertoire source_data. La structure de ce répertoire est créée en suivant la structure brute précédemment définie. Les fichiers REC/PAR identifiés dans chaque **fichier sequences.csv** sont ensuite copiés dans les dossiers correspondants.

Exemple de répertoire source :
subject_id/
|-- ses-id/
|-- 3DT1.PAR
|-- DWI.PAR
|-- 3DT1.REC
|-- DWI.REC

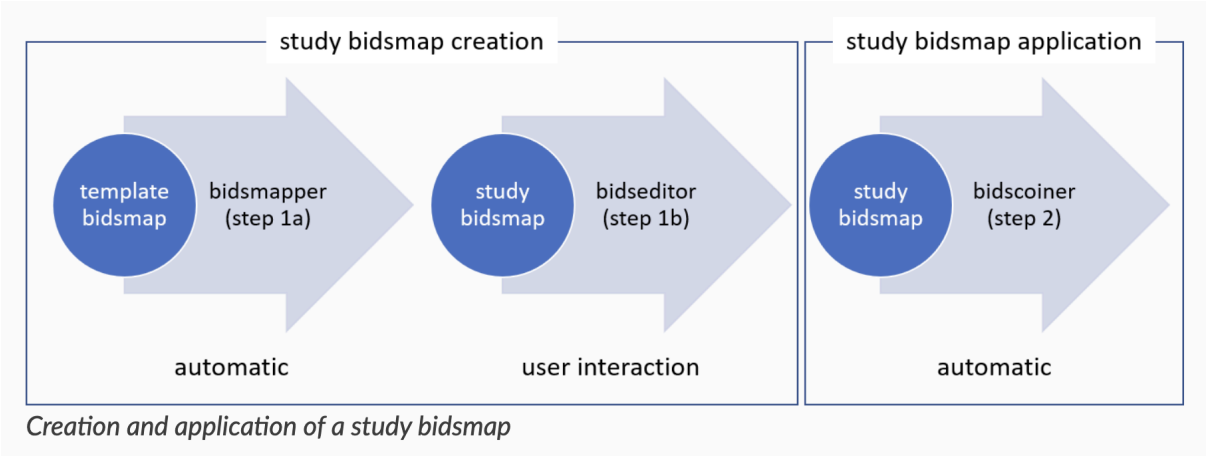
Deuxième étape : Renommer les participants. **Ajout du préfixe "sub-"** et suppression du tiret dans leur identifiant.

Exemple : 1-01VALAL devient sub-101VALAL.

Si deux fichiers ayant le même nom sont destinés au même répertoire. Alors soit:

- les deux fichiers REC/PAR ont la même taille et donc ce sont des doublons : on en copie un des deux
- Les fichiers n'ont pas la même taille. Après vérification, le fichier le plus léger n'est pas convertible avec dcm2niix (sûrement erreur de transfert) : On copie le fichier le plus lourd.
- Cas particulier à 434 et 431, des fichiers DBIEX.PAR/REC correspondent en terme de meta données et donc sont ajouté à source data, mais ce sont des fichiers dédoublés. Donc ils ne sont pas copiés dans source data.
- Cas particulier pour 204HEBTO. Des fichiers DBIEX existent mais ne sont pas tous des doublons. Une copie de ces données à été faite dans le script à part.

Troisième étape : Conversion BIDS. La conversion des données REC/PAR est effectuée avec le logiciel bidscoin. De la même manière que heudiconv, ce logiciel interface dcm2niix (version = 20242202). **Bidscoin, contrairement à heudiconv, est adapté pour la conversion de données REC/PAR en nifti.**



Creation and application of a study bidsmap

source **:** **https://bidscoin.readthedocs.io/en/latest/workflow.html**

Utilisation : *bidsmapper, bidseditor, bidscoiner*

L'équivalent de l'heuristique d'heudiconv est la bidsmap avec bidscoin. elle est écrite au format yaml, en suivant les règles suivantes :

Une T1 peut s'appeler :
- 3DT1-*-1.PAR
- 3DT1-*-1_run-*.PAR

Et DOIT contenir les champs :
- protocol_name: 3DT1
- tech: T1TFE
- scan_mode: 3D

NOM BIDS : anat/sub-*_ses-*_T1w.nii.gz

Une T2* peut s'appeler:
- T2GREph-SENSE-*-1.PAR
- T2GREph-*-1.PAR

Et DOIT contenir les champs :
- protocol_name: T2GREph
- series_type: Image MRSERIES
- tech: FFE

NOM BIDS : anat/sub-*_ses-*_T2starw.[nii.gz/json] & sub-*_ses-*_part-phase_T2starw.[nii.gz/json]

Un resting state multi echo peut s'appeler :
- rs-multi-echo-SENSE-*-1.PAR
- rs-multi-echo-*-1.PAR

et DOIT contenir les champs :

- protocol_name: rs_multi_echo SENSE
- tech: FEEPI
- diffusion: '0'
- scan_mode: MS

NOM BIDS : func/sub-*_ses-*_task-rest_echo-*_bold.[nii.gz/json]

Une DWI PA peut s'appeler :

- WIP-DTI2-3-SENSE-*-1_run-*.PAR
- WIP-DTI2-3-SENSE-*-1.PAR
- DTI2-3-ok-*-1.PAR
- DTI2-3-alt-*-1.PAR

et DOIT contenir les champs :

- protocol_name: DTI2-3-alt
- tech: DwiSE
- scan_mode: MS
- flow_compensation: '0'
- max_echoes: '1'

NOM BIDS : dwi/sub-*_ses-*_acq-64dirs_dir-PA_dwi.[nii.gz/json/bval/bvec]

Une carte de champs peut s'appeler

- WIP-B0MAP-*-1_run-*.PAR,
- WIP-B0MAP-*-1.PAR

Et DOIT contenir les champs :

- protocol_name: WIP B0MAP
- tech: T1FFE
- diffusion: '0'
- scan_mode: MS
- max_slices: '35'
- flow_compensation: '1'
- max_echoes: '1'

NOM BIDS : fmap/sub-*_ses-*_[phase1/magnitude1].[nii.gz/json]

Une DWI AP peut s'appeler :

DTI2-3-alt-topup-*-1.PAR

et DOIT contenir les champs :

- protocol_name: DTI2-3-alt-topup
- tech: DwiSE
- scan_resolution: '[128 128]'
- scan_mode: MS
- flow_compensation: '0'

NOM BIDS : fmapi/sub-*_ses-*_acq-6dirs_dir-AP_epi.[nii.gz/json/bval/bvec]

LISTE DES VARIABLES DE 3_copysource_data.py

code_dir : pointe vers le dossier code où sont les scripts 1,2, …, 7 et bidsmap.yaml
base_dir : pointe vers le dossier apex_enf (parent de rawdata, sub-enf etc)
source_data_dir : pointe vers le dossier source_data (fils de base_dir)
raw_patient_dir : pointe vers le dossier raw_structure
raw_source_dir : pointe vers le dossier sub-enf

OPTIONS DE 3_copysource_data.py

Bool option	Default	Description
copysource	True	copie des source data dans un répertoire source_data
correct_name	True	Correction des noms en sub- etc
get_unique_sequences	False	Génère les noms uniques de séquences, pour la définition de bidsmap.yaml
add_run_label	True	Ajout d'un suffixe run dans les REC/PAR
run_bidscoin	True	Copie de bidsmap depuis code dir dans rawdata/code/bidsoin et exécution de bidscoin

IV. Conversion des DOT et STOP 4_convert_dot_stop

DESCRIPTION

La conversion échoue dans la plupart des cas pour les données IRM d'activation sur les tâches dot et stop.

Identification du problème
La séquence décrite dans le protocole dure 277 secondes. Cependant, **ce paramètre n'est pas ajusté à l'enregistrement, car la séquence est généralement arrêtée avant la fin**. D'après les cahiers d'acquisition, les participants mettent en effet moins de 277 secondes pour exécuter la tâche.

Les outils dcm2niix, dicomifier, et mrconvert échouent tous à convertir les données, car **le nombre d'images n'est pas proportionnel au nombre de coupe**s. Un autre problème identifié est l'arrêt brutal de la séquence, entraînant un dernier volume tronqué. En revanche, dicm2nii, implémenté en matlab, parvient à convertir ces données. Après conversion, on obtient n-1 volumes par rapport au cahier d'acquisition, car **dicm2nii exclut le volume tronqué**.
Une interface MATLAB avec Python (**Matlab Engine**) a été mise en place pour exécuter **dicm2nii sur les tâches dot et stop** :

Étape 1 : Conversion des données PAR en NIfTI
Pour chaque tâche (DOT et STOP), on identifie si le participant a reçu un ou plusieurs fichiers IRM liés à la tâche.

- Si deux fichiers PAR sont présents, les deux sont convertis et nommés run-1 et run-2.
- Sinon, l'unique fichier IRM est converti avec dicm2nii.

Nom BIDS
Les fichiers convertis respectent la nomenclature suivante :

- func/sub-/ses-/sub-_ses-_task-dot_bold.nii.gz
- func/sub-/ses-/sub-_ses-_task-stop_bold.nii.gz

Création du fichier compagnon au format JSON
dicm2nii génère un fichier .mat contenant les métadonnées de l'acquisition. Ce fichier est ensuite converti au format JSON. Finalement, tous les fichiers NIfTI et JSON sont copiés dans les dossiers func

La présence de deux acquisitions IRM pour une même tâche **s'explique toujours par l'une des deux acquisitions étant fortement tronquée** (arrêt prématuré ou redémarrage de la séquence). En cas de doublons, **l'image NIfTI contenant le plus grand nombre de volumes 3D est conservée**, et l'autre acquisition est supprimée.

Note : Matlab et dicm2nii sont donc requis pour cette étape.
Note 2 : pour 2-03VALMA/post, erreur de conversion du rs fMRI. dicm2nii peut convertir, mais alors il crée un nifti, au lieu de trois (un par echo). C'est aussi le cas pour 204HEBTO/postdiff.

LISTE DES VARIABLES

source_data_dir: pointe vers le dossier source_data
rawdata_dir : pointe vers le dossier rawdata

OPTIONS

Bool option	Default	Description
convert_dot_stop	True	Conversion des IRM d'activation dot/stop REC/PAR au format nifti + json
correct_runs	True	Choix d'un des deux IRM d'activation selon le nombre de volumes 3D.

V. Définition des champs Json - "json enrichment" 5_correct_runs_enrich_json.py

DESCRIPTION

La conversion bidscoin, malgré les définitions de la bidsmap, échoue parfois à identifier **lorsque le suffixe run doit être ajouté**. Ainsi, pour la majorité de T2starw qui sont présentes deux fois, la deuxième conversion possède le suffixe run-2, mais le premier fichier s'appelle toujours sub-*_ses-*_T2starw.nii.gz. Pour chaque cas de figure où un fichier contenant 'run-2' est trouvé, et un fichier sans run, alors ce fichier est renommé avec le suffixe run-1. Cette anomalie s'est aussi retrouvée dans d'autres types d'acquisition.

La conversion avec bidscoin génère les **fichiers json** attendus par le format bids, mais peu de champs sont présents. Bidscoin génère un excel bidscoiner.tsv, contenant l'historique des commandes (conversion source → target)
Pour chaque "target", on identifie la source REC/PAR et avec un lecteur txt on crée un nouveau fichier json. On concatène ensuite le json issu de bidscoin et ce json, créant un json final complet.

Gestion des autres problèmes avec les indices "run"

Une fois que les run 1 et run 2 ont été proprement renommés, il a fallu choisir de supprimer un des deux, ou de garder les deux.

Concernant les run-1/run-2 **DWI AP et DWI PA** . Cela concernent certains sujets pour lesquels il y a eu des erreurs de nommages (ex 1-06GUEMA s'appelait 1.06guema, ou 1 06GUEMA). Ce sont des doublons. On a supprimé un, et gardé l'autre sans le suffixe "run".

Concernant les cartes de champs **b=0 (fmap)** : Nous avons, après inspection visuelle pour run-1_magnitude/run-2_magnitude, renommer run-1 en magnitude, run-2 en phase

Une grande partie des sujets ont deux T2*, bien que les cahiers d'acquisition n'y fassent pas référence. Les enfants font parfois les acquisitions en deux temps, lorsqu'ils sont petits ou agités. La T2* dédoublée est un indice disant que **le participant à réalisé ses acquisitions en deux temps (avec 30min de pause)**. Il y a deux T2* pour recaler les images de la première partie des acquisitions avec les images de la deuxième partie. Il faut donc garder les deux. L'index run-1/run-2 a été défini en fonction de l'heure de passage.

Il n'y avait pas de répétition du resting state multi echo. Les répétitions des IRMf d'activation ont été traitées dans la partie précédente.

Harmonisation des clés d'identification entre les métadonnées issues des REC/PAR, dcm2niix et dicm2nii. Au cours de ce projet, différents convertisseurs et lecteurs de métadonnées ont été utilisés. Nous avons converti tous les champs DICOM pour que les clefs correspondent au formalisme BIDS.

La conversion des fieldmap avec bidscoin **ne crée pas de fichiers bval et bvec** (alors que si ces derniers sont compris dans /dwi si). Les fichiers sont donc créés avec dcm2niix puis copiés dans les bons répertoires.

Vérification finale

Création d'un csv (check_conversion.csv) retraçant toutes les acquisitions trouvées dans sourcedata, et leur conversion dans rawdata. Si un fichier existe dans sourcedata mais pas dans raw data ⇒ conversion error = 1.

Après vérification, nous avons utilisé dicm2nii pour convertir les fichiers sur lesquels dcm2niix échouait. Les resting state euvent être convertis mais les trois échos sont concaténés dans une seule image (vol 1 echo 1, vol 2 echo 2, vol 3 echo 3, vol 4 echo 1 etc …)
Concernant:

- l'image DWI de sub-401GUERA/postdiff, dicm2nii échoue aussi. → Problème nb slices et dimensions images (not integer). **Pas de conversion**
- l'image DWI AP de sub-523GUYPA/post, dicm2nii échoue aussi. → Problème nb slices et dimensions images (not integer). **Pas de conversion**

LISTE DES VARIABLES

source_data_dir: pointe vers le dossier source_data

rawdata_dir : pointe vers le dossier rawdata

OPTIONS

Bool option	Default	Description
rename_target	True	Identifier des run-2 dans le fichier bidscoiner.tsv, et créer un run-1
enrich_json	True	Récupération des métadonnées REC/PAR et copie dans les json
correct_run_time	True	Corrige les identifiants de run pour les séquences sauf T2*
correct_run_time_T2	True	Renommer les numéro de run T2* en fonction de l'heure/date d'acquisition
change_field_json	True	Harmoniser les clés d'identification entre les métadonnées issues des REC/PAR, dcm2niix et dicm2nii
correct_fmap_epi	True	Ajouter les bvec et bval des fmap epi
generate_conversion_table	True	Créer un tableau avec les résultats de conversion
check_conversion	True	Check visuel pour chaque erreur de conversion trouvé dans conversion_table.

VI. Renommer les participants : 6_rename_participants.py

DESCRIPTION

Les noms des participants sont renommés pour correspondre à la table de correspondance ci-dessous :

Préfixe original	Nouveau préfixe

1	ca1
2	ci2
3	mt3
4	pc4
5	prema5

Par exemple : sub-107MAPSA devient sub-ca107

Chacune des apparitions du terme sub-x est remplacé par son nouveau nom dans le répertoire rawdata, c'est à dire dans les noms de fichiers json, et nifti, ainsi que dans les fichiers scans.tsv, participants.tsv

LISTE DES VARIABLES

source_data_dir: pointe vers le dossier source_data
rawdata_dir : pointe vers le dossier rawdata

OPTIONS

Bool option	Default	Description
rename_participants	True	Renommer toutes les occurences de l'ancienne nomenclature en nouvelle nomenclature.
rename_tsv	True	Renommer les occurences dans les fichiers tsv de la base bids
rename_summary_file	True	add a column subject_id_bids in the check_conversion.csv file

En bref :
De la structure brute ++ en entrée, le téléchargement de la bidsmap (bidsmap.yaml) et l'exécution un par un des codes contenue dans le repo github https://github.com/FRramon/bids_apex_enf permet d'obtenir la structure bids.