Chapitre 4

Résultats

4.1 Scripts

Les différents scripts python ont été réalisés en collaboration avec Madame Judith Eck qui dirige le Support Desk de Brain Innovation. Après de nombreuses discussions, tests et modifications, les différentes étapes de traitements ont été codées de la manière suivante.

4.1.1 Compatibilité BIDS

Les deux premiers scripts servent à transformer la structure initiale, le format et le nommage des fichiers vers un projet BIDS compatible. Il est important de noter que ces deux scripts peuvent être exécutés sans utiliser BrainVoyager.

NIIfromDCMfolders.py

Ce script considère que les données sont arrangées selon les fichiers visibles à la Figure 4.1.a. Il peut être aisément adapté pour un autre nommage de fichier, par exemple, le patient pourrait-être identifié par "Patient-4" au lieu de "UCL2_4" dans l'exemple montré ici. En ce qui concerne l'organisation des fichiers, les données sont stockées dans un dossier rawdata ce qui est imposé par BIDS. On remarque également un regroupement des images en fonction de leur nature, à savoir anatomique (anat) ou fonctionnelle (func) et non plus par run. Le numéro du patient est rendu BIDS compatible en le renomant par "sub-04". Pour ce qui est du format, les Dicoms sont maintenant regroupés en un seul NIfTI par anatomie (Voir Figure 4.1.c) et un NIfTI par run dans le dossier func (Voir Figure 4.1.d). Pour traiter plusieurs patients à la fois, il suffit d'entrer leur numéro, dans ce cas-ci le 4 et le script détectera automatiquement le nombre de runs pour chaque patient.

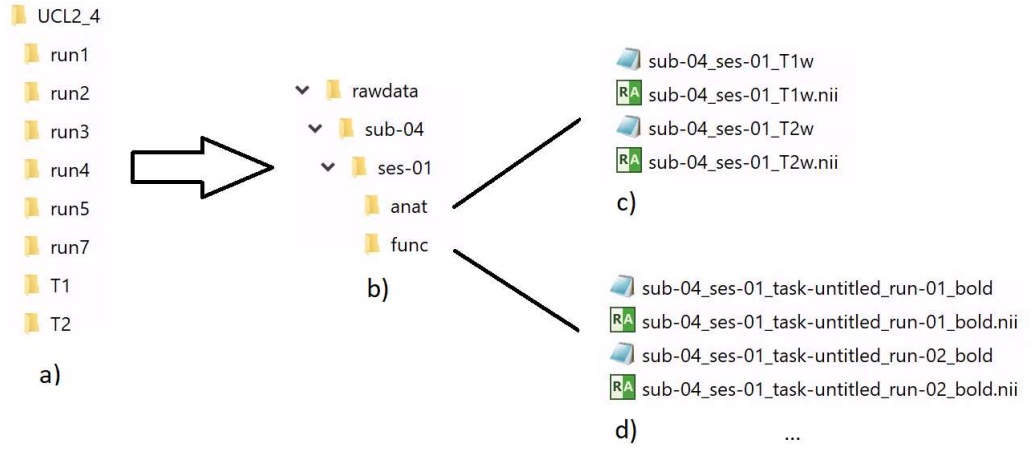

FIGURE 4.1 – Exécution du script *NIIfromDCMfolders.py* sur un sujet type (a). Les sous-dossiers crées (b) sont stockés dans le dossier rawdata. Après la conversion , les fichiers anatomiques (c) et fonctionnels (d) sont maintenant séparés et stockés dans le format NIfTi.

ConvertPRTsInCurrentFoldertoTSV.py

Ce script sert à transformer les protocoles dans le format adéquat du standard BIDS. Le format cible est un "Tab Separated Values" dit tsv. Le nom d'origine est conservé, seul le format est modifié.

4.1.2 Data Analysis Manager

Après avoir assuré la compatibilité BIDS du projet, le traitement des données peut commencer. Afin de se familiariser avec l'organisation du projet, il existe une interface dans BrainVoyager appelée Data Analysis Manager (voir Figure 4.2). Comme on peut le voir, celle-ci est très intuitive et permet aisément de visualiser les sujets, les étapes de traitements ainsi que les liens entre celles-ci. Chaque étape est assimilée à un workflow et est caractérisée par un type, un input et un output. L'avantage est que les workflows peuvent être soit exécutés directement grâce à l'interface soit par un script dans BrainVoyager. Cela permet donc de combiner et compléter l'interface avec des scripts plus complets, plus spécifiques comme ce sera le cas dans ce mémoire. Par exemple, les deux premiers workflows seront exécutés à l'aide de l'interface tandis que les autres seront réalisés par scripts en externe.

Les différents projets ainsi que leur liste de sujets sont visibles sur la gauche de l'image. A l'aide de la commande *Run All* ou *Run Selected* présentent au milieu de l'image, un workflow peut-être exécuté sur tous les sujets ou seulement sur une

sélection. Pour ce qui est des données, les input et les output peuvent être fixés à l'aide de la commande *Connect* sur la droite de l'image. Celle-ci, ouvrira une fenêtre permettant de choisir les données sources du workflow. Par exemple, on peut voir sur la Figure 4.2 que l'input pour l'anatomical préprocessing sera les T1w et que les output seront caractérisés par le suffix T1W auquel sera ajouté un suffix spécifique au préprocessing réalisé (IIHC,...).

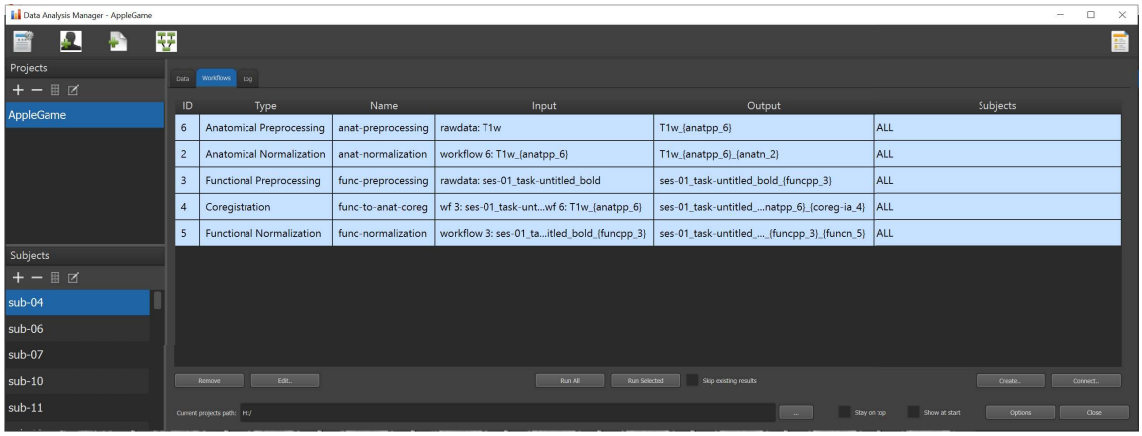

FIGURE 4.2 – Screen Capture of Data Analysis Manager for this thesis.

De plus, le traitement des données anatomiques y est facilement réalisable. Le préprocessing et la normalisation anatomiques seront donc réalisés à l'aide de cette inferface.

Anatomical preprocessing

Comme le montre la figure 4.3, l'interface permet de fixer différents paramètres pour préprocesser les images anatomique. Après les avoir choisis, le workflow peut être exécuté sur un ou plusieurs sujets. Le résultat sera stocké dans un fichier cible respectant le format et le nommage BIDS (Voir Figure 4.4).

Anatomical normalization

Cette étape fonctionne de la même manière que l'étape du preprocessing. Elle permet d'effectuer une normalisation vers l'espace de référence souhaité qui doit être choisi dans les paramètres.

4.1.3 Functionnal Scripts

Cette section regroupe les différents scripts utilisés pour le traitement des données d'imagerie fonctionnelle.

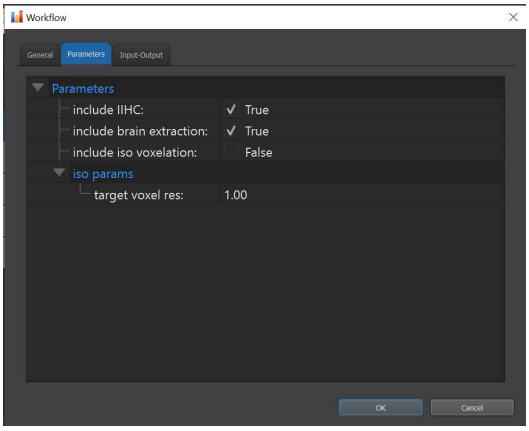

FIGURE 4.3 – Paramètres disponibles pour le preprocessing anatomique. On y retrouve la correction d'inhomogénéité (IIHC), l'extraction du cerveau ainsi que l'iso voxelation qui permet de transformer les voxels en voxels 1x1x1mm. Il est également possible de fixer la taille des voxels finaux via la variable **target voxel res**.[4]

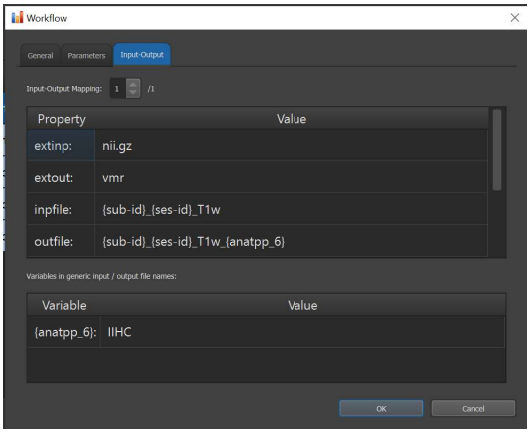

FIGURE 4.4 – Format et nommage des fichiers sources et finaux. Le nommage final(**anatpp_6**) dépend du préprocessing effectué, dans ce cas-ci on peut voir qu'il sera égal à IIHC car une correction d'inhomogénéité sera appliquée.

FMRsfromNII.py

Pour commencer, les fichiers .fmr sont créés à partir des fichiers NIfTis(.nii) pour chaque run de chaque patients. Afin d'assurer le bon fonctionnement du script, plusieurs variables doivent être fixées :

- **project_path** : Le chemin vers les données doit être indiqué dans cette variable.
- **subjects** : Tableau où il faut y indiquer le numéro de chaque sujet à analyser, le script détectera ensuite le nombre de runs effectués par sujet.
- **sessions** : Tableau qui regroupe le numéro de chaque session d'acquisition. Dans ce mémoire, tous les runs ont été acquis au cours d'une même et unique session. Le tableau est donc **sessions** = [1].
- **task_name** : Variable indiquant le nom de la tâche.

Les différents fichiers seront stockés dans des sous-dossiers spécifiques pour chaque patient, chaque session et selon la nature des données. Par exemple, les imageries fonctionnelles seront stockées selon ce chemin : $project\_path + 'derivatives/rawdata\_bv/' + sub\_id + '/' + ses\_id + '/func/' + sub\_id + '\_' + ses\_id + '\_' + task\_name + '\_' + run\_id + '\_bold.fmr'$. On peut remarquer les différents sous-dossiers : derivatives, rawdata_bv et func, ceux-ci sont créés automatiquement s'ils sont inexistants. Le dernier sous-dossier est func pour y stocker les fmr. Pour les images anatomiques, il existe un sous-dossier équivalent appelé anat. La présence des variables des sous-dossiers sub_id et ses_id montre que les images sont stockées en fonction du numéro de patient et de session. Par contre, tous les runs sont stockés dans le même dossier mais ils sont nommés différemment en fonction du numéro de run (run_id).

FMRPreprocessing.py

Afin de procéder au preprocessing des données, il faut définir les différentes étapes que l'on souhaite exécuter sur les images fonctionnelles acquises :

- **Motion correction :** Si l'on souhaite procéder à une correction de mouvement, il faut fixer le booléen **perfom_moco** à True. Si cette correction s'effectue directement sur les images fonctionnelles d'origine(*_bold.fmr*), la variable **input_moco** est fixée à un String vide (' '). Sinon, cette variable doit être modifiée pour correspondre au nommage des images sélectionnées. Pour caractériser le type de motion correction, il faut définir plusieurs références. Pour commencer, le booléen **perform_intrasession** permet d'aligner plusieurs runs acquis lors d'une même session sur une même ré-

férence. En effet, la variable **moco_ref_volume** spécifie le volume et **moco_ref_run** le run de référence. Il est également possible de spécifier une tâche de référence avec la variable **moco_ref_run** si le contexte s'y prête.

— **Slice scan Time correction :** Le booléen **perfom_ssc** doit être égal à True pour exécuter la correction. Lorsque l'on exécute plusieurs corrections à la suite, il faut faire attention d'adapter la variable **input_ssc**. Par exemple, si au préalable, la correction des mouvements a déjà été réalisée, il faut y indiquer *_3DMCTS*. Cela permet d'effectuer la correction de slice sur les images dont les mouvements ont déjà été corrigés.

— **High Pass Filtering :** Le booléen **perfom_hpf** doit être égal à True pour exécuter le filtrage. A nouveau, il faut faire attention d'adapter la variable **input_hpf**. Si les corrections des mouvements ainsi et que de l'ordre des slices ont déja été performées, la variable devient *_3DMCTS_SCCTBL* Par contre, si seule la correction des mouvements a déjà été réalisée, la variable est égale à *_3DMCTS*. Enfin, la variable **n_cycles** permet de spécifier le nombre de cycles

— **Spatial Smoothing :** Comme précédemment, il faut adapter le booléen **perfom_smoothing** ainsi que la variable **input_smoothing** en conséquence. Deux variables supplémentaires **gauss_fwhm** et **fwhm_unit** caractérisent respectivement the full-width at half maximum (fwhm) et ses unités.

GetPlots.py

Ce script permet de générer le graphique représentant les mouvements de la tête. Son analyse est utile pour rejeter les sujets qui ont trop bougé. Les paramètres dépassant les thresholds, fixés au préalable, ne sont pas inclus dans l'analyse. Dans notre cas, un threshold de 4mm et de 4 degrés sont utilisés pour les translations et pour les rotations. Lors de l'exécution de ce script, une fenêtre s'ouvre et il suffit de sélectionner le patient ou le run pour lequel une analyse de mouvement est souhaitée.

4.1.4 Coregistration

Les étapes de coregistration sont divisés en trois scripts. Le premier permet de calculer les transformations nécessaires pour aligner les données fonctionnelles sur les données anatomiques. Le second permet la création des VTCs et le dernier permet, si nécessaire, de réaliser un préprocessing sur les VTCs. Les différents paramètres et variables sont décrites ci-dessous :

Alignment.py

On y retrouve les paramètres classiques déjà développés ci-dessus : project_path, subjects, sessions et task_name. Après cela, on retrouve les chemins vers les différents dossiers où se trouvent les ressources nécessaires :

- **anat_prep :** Chemin vers le dossier contenant les données anatomiques préprocessées.
- **func_prep :** Chemin vers le dossier contenant les données fonctionnelles préprocessées.
- **coreg_out :** Chemin vers le dossier de sortie pour les fichiers créés.

Ensuite, les différents suffix sont définis. Si l'analyse réalisée est semblable à celle de ce mémoire et que le nommage des fichiers a été respecté, il se peut que les variables ne doivent pas être modifiées.

- **anat_suffix :** Suffix représentant les données anatomiques. Par exemple, $\_T1w\_IIHC.vmr$ si les images anatomiques ont été soumis à une correction d'inhomogénéité.
- **func_suffix :** Suffix représentant les données fonctionelles. Par exemple, $\_bold\_3DMCTS\_SCCTBL.fmr$ si la correction des mouvements(3DMCTS) et le slice scan time correction(SCCTBL) ont été réalisés lors du preprocessing fonctionnel.

Pour finir, il reste à définir si BBR l'on souhaite utiliser pour coregistrer. Si c'est le cas, il faut fixé la variable **use_BBR** à True.

Le résultat fourni est la création d'un fichier d'alignement initial (IA.trf) et final (FA.trf). Le nom complet des fichiers reprend le nom de la fonctionnelle ainsi que de l'anatomique sur laquelle elle a été alignée. Un fichier trf permet d'aligner des images anatomiques avec des images fonctionnelles. On y retrouve les informations sur les translations, les rotations et éventuellement les échelles nécessaires pour l'alignement.

VTC_creation.py

Après avoir calculé les alignements, la création des VTCs peut être réalisée. Deux chemins supplémentaires peuvent être nécessaires :

- **vtc_folder :** Chemin vers le dossier cible pour stocker les VTCs en MNI.
- **vtc_native :** Chemin vers le dossier cible pour stocker les VTCs dans l'espace native.

Le choix de l'espace cible se réalise grâce à la variable **mni** qui est fixée à True si l'espace MNI est choisi et false pour le native space. Si un seul des deux espaces est choisi, il n'est donc pas nécessaire de référencer l'autre chemin et cette variable peut-être mise en commentaire.

La caractérisation de la coregistration se fait à l'aide des variables suivantes :

- **res :** Ce paramètre permet de fixer la résolution en mm^2. En général, trois valeurs sont possibles à savoir 1, 2 et 3 mm^2. La valeur adéquate est choisie en fonction de la taille des voxels de la fonctionnelle. Dans notre cas, **res=2**.
- **interpol :** Ce paramètre définit le choix de la technique d'interpolation. La valeur 1 équivaut à une interpolation Nearest neighbor. L'interpolation trilinéaire correspond à la valeur 2. Et enfin, pour une interpolation sinc, il faut choisir la valeur 3 [4].
- **coreg_BBR :** A fixer à True, si l'on souhaite utiliser BBR pour la coregistration, false sinon.
- **use_intrasesscoreg :** La valeur True indique que la coregistration intra session sera réalisée. C'est à dire que les images acquises au sein d'une même session pour un même sujet seront alignées.
- **intensity_thresh :** Option recommandée pour les données d'imagerie fonctionnelle. En fixant ce seuil d'intensité à 100, cela permet de détecter efficacement les valeurs d'intensité des données fonctionnelles dans le tissu cérébral. En effet, celles-ci sont supérieures à 100.

En ce qui concerne les accès aux données, il faut à nouveau indiquer les suffix pour la fmr et l'anatomique (comme ci-dessus pour Alignement.py). Les suffix des alignements doivent être également renseignés. A noter que si la coregistration BBR est utilisée le suffix de l'alignement final passera de $\_FA.trf$ à $\_BBR\_FA.trf$.

L'interface Data Analysis Manager fournit une bonne visualisation générale du projet. En effet, à partir de l'option Data flow, on obtient la figure 4.5 qui illustre les différents liens entre les étapes exécutées.

VTC_preprocessing.py

Il se peut que certaines imperfections doivent être corrigées dans les VTCs. Pour cela, plusieurs étapes de préprocessing peuvent être réalisées :

- **Temporal High pass filtering**
- **Temporal smoothing**
- **Spatial smoothing**

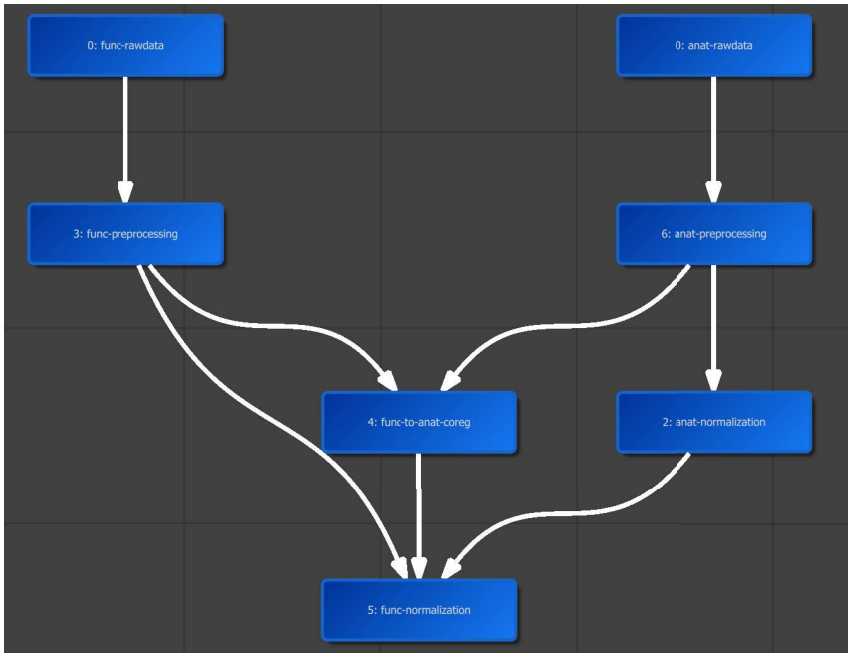

FIGURE 4.5 – **Data Flow Visualization :** Mind map showing the links between the various scripts, from the source data to the creation of the VTCs.

Comme expliqué dans le fmr preprocessing, il faut fixer les différentes variables liées au filtrage et au smoothing.

LinkPrt.py

Après la création des VTCs, ce script permet d'y lier un protocole. A nouveau, il faut spécifier les variables classiques telles que project_path, subjects, sessions et task_name. Afin que le script puisse lier les protocoles aux vtc, il faut spécifier les variables suivantes :

- **file_ending :** Suffix représentant le vtc. Par exemple, pour ce mémoire : $'\_bold\_3DMCTS\_SCCTBL\_256\_sinc\_2x1.0\_MNI.vtc'$.
- **anat_folder :** Chemin vers le dossier contenant la T1.
- **anat_ending :** Suffix représentant la T1. Par exemple, dans le cas d'une T1 en MNI avec IIHC : $'\_T1w\_IIHC\_MNI.vmr'$.
- **vtc_folder :** Chemin vers le dossier contenant les vtc.
- **prt_folder :** Chemin vers le dossier contenant les protocoles (.prt).

4.2 Protocole

Le script crée un fichier Excel intermédiaire où l'on retrouve toutes les données du fichier original castées dans le format adéquat (Voir Figure 4.6). Les variables supplémentaires, comme la vitesse, la variation angulaire y sont également sauvées.

time	dt	v	dyaw	yaw	phasetype	feedback	x	vx	y	vy	z	subtask
0	0,196	0	0	180	0	-1	0	0	0	0	203,91	1
0,196	0,199	0	0	180	0	-1	0	0	0	0	206,474	1
0,395	0,2	0	0	180	0	-1	0	0	0	0	206,474	1
0,595	0,2	0	0	180	0	-1	0	0	0	0	206,474	1
0,795	0,2	0	0	180	0	-1	0	0	0	0	206,474	1
0,995	0,2	0	0	180	0	-1	0	0	0	0	206,474	1
1,195	0,2	0	0	180	0	-1	0	0	0	0	206,474	1
1,395	0,2	0	0	180	0	-1	0	0	0	0	206,474	1
1,595	0,2	0	0	180	0	-1	0	0	0	0	206,474	1
1,795	0,2	0	0	180	0	-1	0	0	0	0	206,474	1
1,995	0,2	0	0	180	0	-1	0	0	0	0	206,474	1

FIGURE 4.6 – Fichier Excel intermédiaire utilisé pour la création d'un fichier protocole. On y retrouve les variables classiques telles que : *time* (temps), v (vitesse), *yaw* (position angulaire). La variable *subtask* permet d'identifier la sous-tâche réalisée à ce temps. Si la valeur est 3, c'est une tâche avec le phare (LPI). En absence de repères (PPI), sa valeur est 1. Cette variable est donc utilisée pour distinguer les translations entre les figures 4.7 et 4.8.

Ensuite, ce fichier Excel est structuré vers un fichier de sortie en fonction de son utilisation future. Par exemple, dans ce mémoire, il doit être reconnu par le programme GridCat et avoir le format illustré à la figure 4.7. Une version plus détaillée est visible à la figure 4.8 car les situations avec (LPI) ou sans phare (PPI) sont dissociées. Il est donc possible d'étudier les différences d'activations des Grids-cells en présence de repères ou non.

```
vnulle;6.795;7.6;180.0
translation;14.395;1.0;290.0
translation;15.395;0.2;285.0
translation;15.595;0.2;280.0
translation;15.795;1.8;275.0
vnulle;17.595;4.2;275.0
translation;21.795;2.0;150.0
translation;23.795;0.8;145.0
translation;24.595;0.2;140.0
translation;24.795;1.401;135.0
vnulle;26.196;7.999;135.0
translation;34.195;5.4;285.0
feedback;39.595;2.0;285.0
```

FIGURE 4.7 – Tableau d'évenements sous forme de Fichier txt compatible avec GridCat.

```
vnulle;6.795;7.6;180.0
translationPPI;14.395;1.0;290.0
translationPPI;15.395;0.2;285.0
translationPPI;15.595;0.2;280.0
translationPPI;15.795;1.8;275.0
vnulle;17.595;4.2;275.0
translationPPI;21.795;2.0;150.0
translationPPI;23.795;0.8;145.0
translationPPI;24.595;0.2;140.0
translationPPI;24.795;1.401;135.0
vnulle;26.196;7.999;135.0
translationPPI;34.195;5.4;285.0
feedback;39.595;2.0;285.0
```

FIGURE 4.8 – Tableau d'évènements sous forme de Fichier txt compatible avec GridCat. Contrairement à la figure de gauche, les évènements de translation sont distinguées en fonction de la présence de repères ou non.

4.3 Grids-cells

Dans cette section, les résultats obtenus avec le programme GridCat sont présentés.

Bibliographie

[1] A. Bierbrauer, L. Kunz, C. A. Gomes, M. Luhmann, L. Deuker, S. Getzmann, E. Wascher, P. D. Gajewski, J. G. Hengstler, M. Fernandez-Alvarez, M. Atienza, D. M. Cammisuli, F. Bonatti, C. Pruneti, A. Percesepe, Y. Bellaali, B. Hanseeuw, B. A. Strange, J. L. Cantero, and N. Axmacher, "Unmasking selective path integration deficits in alzheimer's disease risk carriers," *Science Advances*, vol. 6, no. 35, p. eaba1394, 2020. [Online]. Available : https://www.science.org/doi/abs/10.1126/sciadv.aba1394

[2] L. Colmant, A. Bierbrauer, Y. Bellaali, L. Kunz, J. Van Dongen, K. Sleegers, N. Axmacher, P. Lefèvre, and B. Hanseeuw, "Dissociating effects of aging and genetic risk of sporadic alzheimer's disease on path integration," *Neurobiology of Aging*, 2023. [Online]. Available : https://www.sciencedirect.com/science/article/pii/S0197458023001677

[3] M. F. Folstein, S. E. Folstein, and P. R. McHugh, "Mini-mental state : A practical method for grading the cognitive state of patients for the clinician," *Journal of Psychiatric Research*, vol. 12, no. 3, pp. 189–198, Nov 1975.

[4] BrainVoyager, "BrainVoyager User Guide," "WebPage", 2022. [Online]. Available : https://download.brainvoyager.com/bv/doc/UsersGuide/BrainVoyagerUsersGuide.html

[5] C. Vanden Bulcke, M. Wynen, J. Detobel, F. La Rosa, M. Absinta, L. Dricot, B. Macq, M. Bach Cuadra, and P. Maggi, "Bmat : An open-source bids managing and analysis tool," *NeuroImage : Clinical*, vol. 36, p. 103252, 2022. [Online]. Available : https://www.sciencedirect.com/science/article/pii/S2213158222003175

[6] "Brain imaging data structure (bids)," https://bids.neuroimaging.io/, [Online ; accessed June 1, 2023].

[7] T. Hafting, M. Fyhn, S. Molden, M.-B. Moser, and E. I. Moser, "Microstructure of a spatial map in the entorhinal cortex," *Nature*, vol. 436, pp. 801–6, 09 2005.

[8] A. Maass, D. Berron, L. A. Libby, C. Ranganath, and E. Düzel, "Functional subregions of the human entorhinal cortex," *eLife*, vol. 4, p. e06426, jun 2015. [Online]. Available : https://doi.org/10.7554/eLife.06426

[9] J. J. Knierim, "Neural representations of location outside the hippocampus," *Learning & Memory*, vol. 13, pp. 405–415, 2006.

[10] N. M. van Strien, N. L. M. Cappaert, and M. P. Witter, "The anatomy of memory : an interactive overview of the parahippocampal-hippocampal network," *Nature Reviews. Neuroscience*, vol. 10, pp. 272–282, 2009.

[11] M. P. Witter, C. B. Canto, J. J. Couey, N. Koganezawa, and K. C. O'Reilly, "Architecture of spatial circuits in the hippocampal region," *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, vol. 369, p. 20120515, 2014.

[12] L. Kunz, T. N. Schröder, H. Lee, C. Montag, B. Lachmann, R. Sariyska, M. Reuter, R. Stirnberg, T. Stöcker, P. C. Messing-Floeter, J. Fell, C. F. Doeller, and N. Axmacher, "Reduced grid-cell–like representations in adults at genetic risk for alzheimer's disease," *Science*, vol. 350, no. 6259, pp. 430–433, 2015. [Online]. Available : https://www.science.org/doi/abs/10.1126/science.aac8128

[13] M. Stangl, J. Achtzehn, K. Huber, C. Dietrich, C. Tempelmann, and T. Wolbers, "Compromised grid-cell-like representations in old age as a key mechanism to explain age-related navigational deficits," *Current Biology*, vol. 28, no. 7, pp. 1108–1115.e6, 2018. [Online]. Available : https://www.sciencedirect.com/science/article/pii/S0960982218302239

[14] M. Stangl, J. Shine, and T. Wolbers, "The gridcat : A toolbox for automated analysis of human grid cell codes in fmri," *Frontiers in Neuroinformatics*, vol. 11, 2017. [Online]. Available : https://www.frontiersin.org/articles/10.3389/fninf.2017.00047

[15] D. Hainaut and R. von Sachs, "Lfsab1105 - probabilité et statistiques," École Polytechnique de Louvain, Université Catholique de Louvain, 2019, cours universitaire.

[16] Astropy Collaboration. (2023) Rayleigh test of uniformity. [Online]. Available : https://docs.astropy.org/en/stable/api/astropy.stats.rayleightest.html