**Liste developpements possibles package**

**Cleaning :**

**-**definir clairement des exceptions sur boris

-definir le format d’output des fichiers (garder le même que la ?).

-definir le format des comportements qu’on veut utiliser. Avantage de segmenter : les différentes étapes sont rendues indépendantes.

-rendre les différentes etaps indépendantes pour la partie filtering, et essayer de voir pour faire liste des fonctions qui ont été appliquées. Tout maintenir dans une classe de données sociales. Définir le format d’output des données.Type d’objet qui fonctionne à partir de ça. Interet : casser un peu l’aspect nested des fonctions.

-au moment de la sauvegarde des données, voir si reconversion nécessaire des données du package.

**Network construction :**

**-**definir les différentes méthodes qui peuvent être utilisées. (scan, comportements affiliatifs, ratio agonistique vs affliatif, etc.).

**-**voir l’interet du scan

**-**commencer à bosser sur multilayer avec Fabrizio

-mettre les affiliative network séparément. On s’en fout de l’optimisaton à ce niveau.

**Net\_viz :**

**-**relier Gephi à Python

-voir si budget temps pertinent

**Gui\_code :**

-Interface graphique à codé plutôt que par ligne de commande

**Stat\_analysis :**

**-**Voir controlabilité etc.

-definir des objets de type null model.

-pour les réseaux, conserver une structure pandas ou passer sur un mode obje qui contient les propriétés ? -> celui la pas mal.

-Partie de Subsampling.

**Aspect dev meta :**

-voir comment sauver les dependances convenablement

-faire un flowchart des méthodes d’utilisation classiques du truc. 1 : cleaning, construction, stat, visualisation.

-tester les options alternatives (Socprog, ANTS) et voir comment ils font ça (format d’entrée, type d’analyses).

-Avec cette organisation, très simple de juste réorganiser les façons de faire.

-discuter avec Axel de la structure générale du code.

-Pour els results / p^reprocessing, faire en sorte qe tout soit sauvé au bon endroit directement. L’utilisateur doit juste rentrer le nom du fichier.

-charger ous les packages dépendants pour tout d’un coup

-qustion sur la definition des fonctions propre au truc ou direct dans méthodes

-dissocier les fonctions qui ont été appliquées

-dissocier les méthodes de construction de réseaux.

**Avancée du projet**

- Une partie claire du projet sur le package, les améliorations et les utilisations effectuées sur les données. Faire un lien sur les développements possibles du package et le résultats permis.

-Autres types d’analyses neuro. La voir avec Strabourg, et noter leurs idées d’une couleur différente, idem pour Fabrizo. Autre partie sur Granger et données IRM.

**Autre**

Charger la table des comportements affiliatifs à part, a prendre comme unve valeur d’input à un moment donné.

Pour ind\_obs\_time, voir si ça peut être utile d’avoir des individus qui ne sont pas du tout dans le jeu de donnée.

Définir les deux colonnes de comportement sociaux.

Faire des checks dans les données : que tout les comportements sociaux aient bien un interactor, que tous les comportements orienés aient bien, etc…..

Méthode de check. Faire des briques de trucs très généraux et des bloques qui synthetisent tout ça pour un pipeline rapide pour l’utilisateur.

Edege\_list : toujours non-symetrique.

**04/04 : Modif du cleaning des données : définir les invalid\_lines en fonction du raw\_data (à faire plus tard)**

**Gérer les histoires de format final + ajuster parser à partir de ça.**

**Les trucs à changer :**

**Cleaning :**

**Import data -> a séparer : soit fusion soit table de base**

**Ordonner données : ok (optionnel si déjà fait)**

**Durée focale : OK (optionnel si déjà fait)**

**Petites corrections : OK (optionnel si déjà fait)**

**Corriger categories comportementale : rendre optionnel (pas obligé d’être fait)**

**New behavioral cat : optionnel -> a adjuster dans visualization. Tres spécifique à notre truc ici.**

**Empty\_col : optionnel, très spécifique à notre truc ici.**

**Ajouter raw\_index : optionnel, nécessaire si on veut effectuer des modifs de filtering.**

**Pour toutes les fonctions du filtering, corriger avec ça.**

**Ajouter lag etc comme valeurs dans la table.**

**Exploration graphique possible n’importe quand.**

**Faire les dernières corrections des valeurs dans la table.**

**Une fois que tout ça est effectué, parsing. Ça c’est presque finito. Avec parsing, chaque matrice d’adjacence.**

**Trier entre ce qui est très specifique à notre cas et ce qui peut être plus general.**

**Thèse : sortir package. Actuellement : pas besoin d’avoir un truc finalisé pour le cleaning. Juste brique de base qui aidera bien pour Saimiri. Pour le cleaning, on a deja les briques de base, et pour l’instant necessite de l’ajustement manuel probablement sous-optimal, on verra pour améliorer ça. format datastream clean, format parsing clean (bientôt), reste plus qu’a voir. Soit dict de matrices d’adjacence, soit liste. Une fois que ça c’est bon, on part sur le multi-couche et les analyses stat.**

**On définit bien la distance, etc tout ce qu’on veut pour faire les stimulis neuro. Allez allez pas d’urgence pour package, on pourra détailler ou on en est dans rapport sans que ça pose pb. On pourra présenter les réseaux qu’on a construit, les analyses effectuées sur ces réseaux, et la construction des stimuli pour IRM fonctionnelle. OK donc maintenant on a format datastream preparé et bientôt format parsing préparé aussi, on aura plus besoin de ce compliqué plus que ça avec le cleaning. Nickel nickel nickel. Jusque la nécessaire, pur avoir format unique propre. On pourra faire le schéma pour expliquer tout ça. Avantage du datastream : représentations associées, stats de base, etc. Derrière, analyses multi-couches avec MuxvizR. Voir pour les méthodes de permut etc. Important pour le cadre stat.**

**Urgence maintenant : terminer le parsing correct, et tracer sur analyses : multi-couche, proximité, seuillage, cadre stat.**