UCLouvain

Institut de Statistique, Biostatistique et Sciences Actuarielles

Extension du package `limpca` à l'analyse des modèles linéaires mixtes.

Membres du jury :

Co-Promoteur : Bernadette Govaerts
Lecteur : Céline Bugli

Mémoire présenté en vue de
l'obtention du master
en science des données
(orientation statistique)
par :

Antoine Visschers

Louvain-La-Neuve
Janvier 2024

Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidé et soutenu durant la rédaction de ce mémoire.

Tout d'abord, je voudrais remercier ma promotrice Madame Bernadette Govaerts pour avoir pris beaucoup de son temps pour m'expliquer, me conseiller et me corriger dans l'élaboration de celui-ci.

Je tiens à témoigner toute ma reconnaissance aux personnes suivantes, pour leur aide dans la réalisation de ce mémoire :

Madame Manon Martin pour son code informatique sur lequel je me suis basé pour l'extension de `limpca`. Celui-ci m'a beaucoup aidé.

Madame Céline Bugli pour avoir accepté d'être ma lectrice et pour le temps qu'elle a accordé à ce mémoire.

Monsieur Jean-Louis Claes pour les corrections et les conseils apportés durant la rédaction.

Ma famille et mes amis pour leurs corrections du mémoire mais aussi pour leur soutien et leurs conseils avisés tout au long de celui-ci.

Table des matières

Chapitre 1

Introduction

Il est très difficile de créer un modèle et de visualiser les facteurs sur des données *-omiques* car celles-ci ont souvent une structure particulière qui rend difficile leur analyse. C'est pour pallier à cette difficulté que le package `limpca` expliqué dans Thiel et al.(2023)[1] a été créé. Le package `limpca` permet d'analyser des ensembles de données *-omiques* qui émanant d'un design d'expérience multifactoriel non balancé et sont surtout réputées d'avoir un très grand nombre de variables de réponse.

Pour pouvoir interpréter ce type de données, il faut choisir une bonne méthode. L'analyse de la variance (ANOVA) est une méthode commune qui permet d'analyser les effets des facteurs pour une variable de réponse qui décompose la variance des données en des variances liées à chaque effet du modèle. Cependant, elle ne suffit pas pour la majorité des données *-omiques* car il y a une multitude de variables de réponse et cette méthode n'analyse qu'une seule d'entre elles. Le MANOVA (multivariate ANOVA) n'est pas non plus la méthode adéquate lorsqu'il y a plus de variables de réponse que d'observations. Quant à l'Analyse en composante principale (ACP), elle ne permet pas d'analyser ce type de données car elle ne prend pas en compte la structure d'un design d'expérience. L'ASCA (ANOVA-Simultaneous Component Analysis) et APCA (ANOVA-Principal Component Analysis) sont des méthodes qui combinent l'ANOVA et l'ACP. Elles sont utilisées pour la modélisation et pour visualiser et interpréter les résultats du modèle dans un espace réduit par la méthode ACP. L'ASCA et l'APCA fonctionnent pour ce type de données émanant mais seulement lorsque le plan est balancé. Pour gérer les données non balancées, le package `limpca` utilise les méthodes ASCA+ et APCA+.

Les méthodes ASCA+ et APCA+ sont des méthodes qui s'inspirent des méthodes ASCA et APCA mais utilisent des principes des LM (General Linear Models) plutôt que la méthode ANOVA. Ces deux méthodes apportent une solution générale pour analyser des données émanant d'un plan d'expériences. De plus, elles fonctionnent correctement pour les plans non balancés en corrigeant le biais des estimations des paramètres par l'utilisation de l'estimateur OLS (Ordinary Least Squares). L'approche LM englobe tous les modèles analysables en ANOVA avec des effets catégoriels fixes mais pourra aussi être étendue à

des modèles incluant des facteurs quantitatifs ou encore des effets aléatoires.

Le package `limpca` permet d'explorer les données d'un ensemble d'observations avec une structure qui comporte une matrice de réponse de grande dimension ainsi qu'un plan d'expérience. Il permet aussi d'ajuster le modèle choisi sur ces données, d'analyser et de visualiser les résultats avec une variété de statistiques, graphiques et tables. Ce package a l'avantage de pouvoir traiter des plans d'expérience balancés et non balancés pour des facteurs catégoriels fixes. Il calcule l'importance des effets et leurs significativités. Il permet de visualiser les données sur un grand nombre de graphiques personnalisables et peut être étendu à des modèles plus sophistiqués du point de vue statistique. Le package `limpca` ne peut gérer que des modèles linéaires simples avec seulement des effets fixes.

L'objectif de ce mémoire est d'étendre le package `limpca` à la gestion de modèles mixtes. A cette fin, trois ensembles de données différents seront utilisés dans le but de créer, pour chacun d'entre eux, un modèle mixte et d'être capable d'analyser ces modèles avec le package `limpca`. Comme le package `limpca` ne gère pas encore les modèles mixtes, les fonctions du package sont modifiées pour y ajouter la gestion des effets aléatoires en utilisant et généralisant la méthode de Martin & Govaerts LiMM-PCA (2020)[2]. Toutes les ressources créées pour la réalisation du mémoire peuvent être visualisées dans le Github d'Antoine Visschers : `https://github.com/AntoineVisschers/limpca_LMM`. Lorsque, dans le mémoire, nous faisons référence à Github, il s'agira donc de celui-ci. Le GitHub est actuellement en mode Privé mais, si des personnes sont intéressées par l'extension du package `limpca`, elles peuvent toujours contacter les auteurs de ce dernier.

Par convention, les mots en italique dans ce texte représentent les mots en anglais qui ne sont pas traduits en français.

1.1 Structure du mémoire

Ce mémoire est structuré comme suit :

— la Section 1.2 présente les quatre ensembles de données utilisées : Candies, Répétabilité/Reproductibilité, CHOO et UCH.

— Le Chapitre 2 explique le package existant `limpca`, la méthodologie sur lequel il se base : ASCA+ et APCA+, le modèle linéaire mixte (LMM) et la méthodologie sur laquelle l'extension du package `limpca` se base : LiMM-PCA.

— Le Chapitre 3 compare les différentes sorties de Candies avec le package `limpca`, la méthode LiMM-PCA de M.Martin et l'extension du package `limpca`.

— Le Chapitre 4 présente l'implémentation de l'extension du package `limpca` par la méthode LiMM-PCA. Celui-ci est découpé en sous-sections représentant chaque étape de `limpca`. Chacune de celles-ci comprend une présentation des fonctions modifiées et/ou ajoutées, ainsi que des problèmes rencontrés durant l'implémentation de l'extension pour cette étape. Le chapitre 4 contient aussi une section avec

les améliorations qui pourraient encore être apportées de même que les problèmes restant à gérer.
— Le Chapitre 5 est consacré aux conclusions de ce mémoire.

1.2 Ensembles de données utilisées

Les trois ensembles de données utilisés pour tester l'extension du package `limpca` sont les données Candies, les données Répétabilité/Reproductibilité qui seront nommées les données Sérum dans la suite du texte et les données CHOO. Chaque ensemble de données comportera un plan d'expérience et une matrice réponse.

L'ensemble de données Candies a été publié pour la première fois par Luciano & Næs (2009)[6] et Liland et al.(2018)[5] et repris par Martin & Govaerts(2020) dans leur article qui introduit la méthode LiMM-PCA[2].

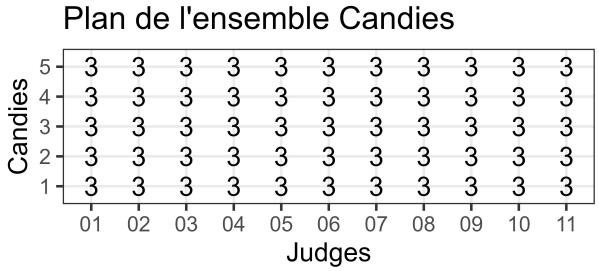


FIGURE 1.2.1 – Plan d'expérience de Candies

Dans les données Candies, le plan d'expérience 1.2.1 comporte 3 facteurs catégoriels : les bonbons, les juges et les répétitions. 11 juges notent 5 bonbons, 3 fois chacun sur 9 critères différents : la transparence, l'acidité, le goût sucré, l'arôme de framboise, la texture de l'enrobage du sucre testée avec une cuillère, la force de morsure dans la bouche, la dureté, l'élasticité en bouche et l'adhérence aux dents lorsque le bonbon est en bouche. Les notes pour chaque critère sont sur une échelle entre 1 et 15 et ne sont pas obligatoirement des entiers. Les données sont balancées, donc il n'en manque aucune, ce qui donne 165 = (11 x 5 x 3) observations et 9 variables. La structure du code pour identifier une observation est la concaténation du numéro du juge, du numéro du bonbon et du numéro de la répétition. Par exemple, l'observation 0913 correspond au juge numéro 09, au bonbon numéro 1 et à la 3ème répétition. Le modèle mixte ne prendra pas en compte le facteur *Repetition*. Le modèle construit sera un modèle mixte avec un effet aléatoire principal $Judges$ et un effet aléatoire d'interaction $Candies : Judges$ entre l'effet aléatoire $Judges$ et l'effet fixe $Candies$. La formule du modèle sera :

$$outcome \sim Candies + (1|Judges) + (1|Candies : Judges) \tag{1.2.1}$$

L'*outcome* est l'une des 9 colonnes de la matrice réponse. Le tableau récapitulatif des données Candies est le suivant :

TABLE 1.2.1 – Description des données Candies pour notre modèle

Nom	**Valeur**
Nom de l'ensemble	Candies
Nombre d'observations	165
Nombre de réponses	9
Facteurs utilisés dans le modèle	$Candies$ (5 niveaux) $Judges$ (11 niveaux)
Effets utilisés dans le modèle	effet fixe $Candies$ (4 paramètres) effet aléatoire $Judges$ (11 paramètres) effet aléatoire $Candies : Judges$ (55 paramètres)
Formule du modèle	$outcome \sim Candies + (1\|Judges) + (1\|Candies : Judges)$

Les données Répétabilité/Reproductibilité appelées les données Sérum ont été, au départ, utilisées dans les articles de Rousseau (2011)[7] et Martin & Govaerts(2020)[2]. Cette base de données a été créée avec des échantillons de sang prélevé sur des volontaires. Les échantillons sont mesurés puis mis dans un spectromètre, ce qui permet d'avoir, pour chaque observation, un spectre qui est enregistré sous forme d'un vecteur numérique de réponse de taille 750. Le plan d'expérience 1.2.2 comporte 4 facteurs catégoriels : les volontaires, les échantillons, les tubes et les mesures. Par la suite, le facteur des tubes ne sera plus pris en compte dans le modèle mixte et n'est pas pris en compte dans le plan d'expérience suivant :

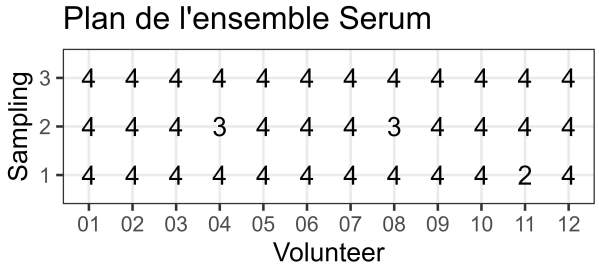


FIGURE 1.2.2 – Plan d'expérience de Serum sans le facteur *Tubes*

Il y a 12 volontaires en bonne santé. Chaque volontaire a donné 3 échantillons de sang prélevés sur 3 jours différents non consécutifs. Pour chaque échantillon, il y a 2 tubes collectés et chaque tube est mesuré 2 fois (4 observations au total). Si la matrice de réponse était complète, elle comporterait 144 = (12 x 3 x 2 x 2) observations avec 750 variables. Chaque observation est nommée avec un code à 5 chiffres, c'est la concaténation du numéro du volontaire, du numéro de l'échantillon, du numéro du tube et du numéro de la mesure. Par exemple, l'observation 04321 représente le volontaire numéro 04, l'échantillon numéro

3, le tube numéro 2 et la mesure numéro 1. Les quatre observations 04221, 08222, 11121 et 11122 ne sont pas reprises dans la matrice de réponse et dans le plan d'expérience à cause d'un problème d'acquisition. Le plan d'expérience n'est donc pas balancé. Le modèle mixte testé sur ces données ne prendra pas en compte l'effets $Tubes$. Le modèle construit avec les données Sérum est un modèle mixte hiérarchique ne comportant aucune variable fixe et deux variables aléatoires : les volontaires $Volunteer$ et les échantillons des volontaires $Volunteer : Sampling$. La formule du modèle sera :

$$outcome \sim (1|Volunteer) + (1|Volunteer : Sampling) \tag{1.2.2}$$

L'*outcome* est l'une des 750 colonnes de la matrice réponse. Le tableau récapitulatif des données Serum est le suivant :

TABLE 1.2.2 – Description des données Serum pour notre modèle

Nom	**Valeur**
Nom de l'ensemble	Serum (Répétabilité/Reproductibilité)
Nombre d'observations	140
Nombre de réponses	750
Facteurs utilisés dans le modèle	$Volunteer$ (12 niveaux) $Sampling$ (3 niveaux)
Effets utilisés dans le modèle	effet aléatoire $Volunteer$ (12 paramètres) effet aléatoire $Volunteer : Sampling$ (36 paramètres)
Formule du modèle	$outcome \sim (1\|Volunteer) + (1\|Volunteer : Sampling)$

L'ensemble de données CHOO provient de l'article Choo & al. (2017)[8] et a été utilisé dans le mémoire de Tournay (2020)[9] pour tester un modèle longitudinal avec la méthode LiMM-PCA. Les données ont pour but de caractériser l'impact de deux antibiotiques cliniquement importants, la vancomycine-imipénem et le ciprofloxacine en affectant les bactéries et les substances chimiques dans l'intestin de souris femelles et permettant aussi le rétablissement de la flore intestinale. Le plan d'expérience 1.2.3 comporte 3 facteurs catégoriels : les traitements, les souris et le temps.

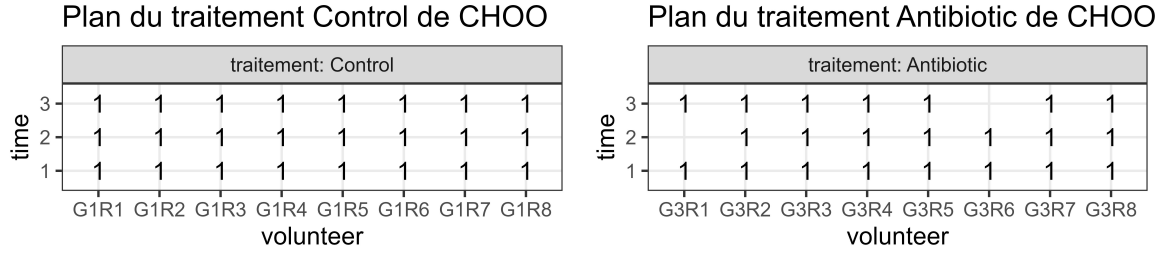


FIGURE 1.2.3 – Plan d'expérience de CHOO

Dans les données originales, il y a 3 groupes de 8 souris avec 3 collectes d'échantillons de matières fécales pour chaque souris à 3 moments différents : avant le traitement, après

14 jours et après 28 jours. Le premier groupe est le groupe de contrôle et donc, on n'injecte rien au souris de ce groupe. Les souris de l'un des deux autres groupes reçoivent comme antibiotique la vancomycine-imipénem et le dernier groupe de souris le ciprofloxacine. Le modèle se base sur les données CHOO de M.Tournay qui ne comportent plus que 2 groupes de souris dont le groupe contrôle et, donc, au total, il reste 16 souris. Les observations sont nommées en concaténant la lettre T, le numéro de la collecte d'échantillon, un tiret, la lettre G, le numéro du groupe, la lettre R, le numéro de la souris dans le groupe. Par exemple, l'observation T1-G2R7 est la collecte numéro 1 de la souris 7 dans le groupe 2. Si la matrice de réponse CHOO était complète, il y aurait 48 = (8 x 2 x 3) observations, mais il manque 2 observations T2-G3R1 et T3-G3R6. Le plan d'expérience n'est donc pas équilibré. Chaque échantillon de matière fécale a été analysé dans un spectromètre et nous donne un vecteur numérique de taille 1652 représentant le spectre. Cependant, les données utilisées ont été transformées par un pré-traitement de M.Tournay, ce qui a réduit la taille à 1452 réponses. Donc, la matrice des réponses aura 46 observations et 1452 variables. Le modèle mixte basé sur les données CHOO est un modèle longitudinal identique à celui de M.Tournay. Il comporte 3 effets fixes et 1 effet aléatoire. Les effets fixes dans le modèles sont l'effet traitement *treatment*, l'effet temps *time* et l'effet d'interaction entre le traitement et le temps *treatment* : *time*. La variable aléatoire est l'effet souris *volunteer*.La formule du modèle sera :

$$outcome \sim treatment + time + treatment : time + (1|volunteer) \tag{1.2.3}$$

L'*outcome* est l'une des 1452 colonnes de la matrice réponse. Le tableau récapitulatif des données CHOO est le suivant :

TABLE 1.2.3 – Description des données CHOO pour notre modèle

Nom	**Valeur**
Nom de l'ensemble	CHOO
Nombre d'observations	46
Nombre de réponses	1452
Facteurs utilisés dans le modèle	*treatment* (2 niveaux) *time* (3 niveaux) *volunteer* (16 niveaux)
Effets utilisés dans le modèle	effet fixe *treatment* (1 paramètres) effet fixe *time* (2 paramètres) effet fixe *treatment* : *time* (2 paramètres) effet aléatoire *volunteer* (16 paramètres)
Formule du modèle	$outcome \sim treatment + time + treatment : time + (1\|volunteer)$

L'ensemble de données UCH du package `limpca` sera aussi utilisé pour tester que le package modifié fonctionne toujours pour des modèles linéaires généraux à effets fixes. La

page web de l'exemple UCH du package limpca est disponible ici pour plus des détailles sur l'ensemble UCH. Le tableau récapitulatif des données UCH est le suivant :

TABLE 1.2.4 – Description des données UCH pour le modèle

Nom	**Valeur**
Nom de l'ensemble	UCH
Nombre d'observations	34
Nombre de réponses	600
Facteurs utilisés dans le modèle	$Hippurate$ (3 niveaux) $Citrate$ (3 niveaux) $Time$ (2 niveaux)
Effets utilisés dans le modèle	effet fixe $Hippurate$ (2 paramètres) effet fixe $Citrate$ (2 paramètres) effet fixe $Time$ (1 paramètres) effet fixe $Hippurate : Citrate$ (4 paramètres) effet fixe $Time : Hippurate$ (2 paramètres) effet fixe $Time : Citrate$ (2 paramètres) effet fixe $Hippurate : Citrate : Time$ (4 paramètres)
Formule du modèle	$outcomes \sim Hippurate + Citrate + Time + Hippurate :$ $Citrate + Time : Hippurate + Time :$ $Citrate + Hippurate : Citrate : Time$

Chapitre 2

Méthodologies et package existant

2.1 ASCA+ et APCA+

L'ASCA+ et l'APCA+, présentés par Thiel et al. (2017) [3], se basent sur l'ASCA (ANOVA-simultaneous component analysis) et l'APCA (ANOVA-PCA) mais ne combinent plus une ANOVA avec des analyses en composantes principales (ACP) mais un modèle linéaire général (LM) avec des ACP. Là où les estimateurs de l'ANOVA étaient biaisés pour des données dans un plan non balancé et où la décomposition de la variance totale ne produit plus de somme des carrés orthogonaux, la méthode LM profaire des solutions à ces problèmes avec des résultats équivalents dans le cas d'une ANOVA avec un plan balancé.

La formule du LM est écrite :

$$\mathbf{Y} = \mathbf{X}\mathbf{\Theta} + \mathbf{E} \tag{2.1.1}$$

où $\mathbf{Y}$ est la matrice des réponses ou de *outcomes* de dimensions $(n \times m)$, $\mathbf{X}$ est la matrice de modèle de dimensions $(n \times p)$, $\mathbf{\Theta}$ est la matrice des paramètres de dimensions $(p \times m)$ et $\mathbf{E} \sim N(\mathbf{0}, \sigma_\epsilon^2 \mathbf{I_n})$ est la matrice des résidus de dimensions $(n \times m)$. n est le nombre d'observations, p est le nombre de paramètres du modèle linéaire pour une réponse et m le nombre de variables de réponse.

La matrice de modèles $\mathbf{X}$ peut être décomposée en $F + 1$ blocs pour récupérer la matrice de modèle pour chaque effet et le vecteur du terme constant :

$$\mathbf{X} = (\mathbf{X}_0|\mathbf{X}_1|...|\mathbf{X}_F) \tag{2.1.2}$$

$\mathbf{X}_0$ est le vecteur du terme constant de dimensions $(n \times 1)$ et $\mathbf{X}_i$ est la matrice du modèle pour l'effet i de dimensions $(n \times p_f)$ où p_f est le nombre de colonnes liées à l'effet f. La matrice du modèle $\mathbf{X}$ est encodée avec la méthode *sum coding* de Thiel et al. (2017) [3] qui permet de transformer le tableau de données du plan expérimental en une matrice de modèle où, pour un facteur du plan avec l niveaux, il y aura (l - 1) colonnes avec des valeurs 0 et 1 pour les (l - 1) premiers niveaux et que des -1 pour le dernier niveau. Pour la

création d'une matrice d'interaction entre l'effet a et l'effet b, il suffira d'utiliser la multiplication par colonne sur les 2 matrices du modèle des effets intervenant dans l'interaction et donnera $(l_a - 1)$ x $(l_b - 1)$ colonnes.

La matrice des paramètres $\mathbf{\Theta}$ sera estimée avec la méthode des moindres carrés ordinaires (OLS).

$$\hat{\mathbf{\Theta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \tag{2.1.3}$$

Cet estimateur est non biaisé et peut être décomposé en $F + 1$ blocs pour retrouver les paramètres par effet et les termes constants :

$$\hat{\mathbf{\Theta}}' = (\hat{\mathbf{\Theta}}'_0|\hat{\mathbf{\Theta}}'_1|...|\hat{\mathbf{\Theta}}'_F) \tag{2.1.4}$$

Dans l'ASCA+ et l'APCA+, la matrice de réponse est décomposée en matrice d'effet :

$$\mathbf{Y} = \hat{\mathbf{M}}_0 + \sum_{f=1}^{F} \hat{\mathbf{M}}_f + \hat{\mathbf{E}} \tag{2.1.5}$$

avec l'estimation de la matrice pour l'effet f $\hat{\mathbf{M}}_f = \mathbf{X}_f\hat{\Theta}_f$ et la matrice des résidus $\hat{\mathbf{E}}_f = \mathbf{Y} - \mathbf{X}\hat{\Theta}$.

En ASCA et APCA, l'analyse de la quantification d'information expliquée pour chaque effet fixe f est basé sur les sommes des carrées SS_f de l'ANOVA et le pourcentage de variance expliqué par l'effet f est calculé comme suit :

$$\%\text{Var}_f = \frac{\|\hat{\mathbf{M}}_f\|^2}{\|\mathbf{Y}\|^2 - \|\hat{\mathbf{M}}_0\|^2} * 100 \tag{2.1.6}$$

mais cela induit un problème : si le plan n'est pas balancé, alors la somme des pourcentage de variance pour chaque effet ne sera pas égale à 100%. En ASCA+ et APCA+, on peut utiliser le *type III SS* du LM calculé comme la différence entre les résidus du modèle sans l'effet f et les résidus du modèle avec l'effet f :

$$\%\text{Var}_f = \frac{\|\hat{\mathbf{E}}_{/f}\|^2 - \|\hat{\mathbf{E}}_{full}\|^2}{\|\mathbf{Y} - \hat{\mathbf{M}}_0\|^2} * 100 \tag{2.1.7}$$

où $\hat{\mathbf{E}}_{/f}$ est la matrice des résidus du modèle sans l'effet f et $\hat{\mathbf{E}}_{full}$ est la matrice des résidus du modèles complet avec tous les effets. Le pourcentage de variance expliqué par les résidus peut être calculé par :

$$\%\text{Var}_e = \frac{\|\hat{\mathbf{E}}_{full}\|^2}{\|\mathbf{Y} - \hat{\mathbf{M}}_0\|^2} * 100. \tag{2.1.8}$$

Le *type III* SS_f a les mêmes résultats que le SS_f d'une ANOVA si le plan d'expérience est balancé mais le *type III* SS_f est plus informatif quand le plan n'est pas balancé.

En ASCA et APCA, traditionnellement des tests de permutations proposés pour quantifier la significativité d'un effet du modèle. Ils consistent à permuter aléatoirement N_u fois les lignes de la matrice de réponse $\mathbf{Y}$, de calculer la statistique $SS_f^u(\mathbf{T}_g)$ qui est le carré de la norme de Frobenius de la matrice des scores des g premières composantes principales $\mathbf{T}_g$ pour l'effet f et de calculer la p-valeur :

$$\text{p-valeur} = \frac{\#\{\text{SS}_f^u(\mathbf{T}_g) \geq \text{SS}_f(\mathbf{T}_g), u = 1, ..., N_u\}}{N^u}. \tag{2.1.9}$$

Ces tests de permutations fonctionnent sur des plans balancés et non-balancés mais doivent être adaptés pour chaque structure de plan d'expérience et de modèle. Pour cette raison, plus loin, il sera expliqué que dans le package `limpca` une approche par bootstrap est plutôt utilisées pour calculer la significativité des modèles.

Pour visualiser les résultats de la modélisation, une analyse en composantes principales est faite sur chaque matrice d'effets $\hat{\mathbf{M}}_f$. Il y a 3 méthodes pour appliquer cette ACP : l'ASCA, l'APCA et l'ASCA-E (une variante de l'APCA) :

— La méthode ASCA applique l'ACP sur la matrice d'effet f "pure" :

$$\hat{\mathbf{M}}_f = \mathbf{T}_f \mathbf{P}'_f \tag{2.1.10}$$

où $\mathbf{P}_f$ est la matrice des *loadings* et $\mathbf{T}_f$ est la matrice des scores pour l'effet f. La première colonne de la matrice des *loadings* $\mathbf{P}_f$ qui est la première composante principale permet de voir quelles réponses sont les plus affectées par l'effet f.

— La méthode APCA n'applique pas l'ACP sur la matrice d'effet pure $\hat{\mathbf{M}}_f$ mais sur la matrice augmentée $\tilde{\mathbf{M}}_f = \hat{\mathbf{M}}_f + \hat{\mathbf{E}}$ qui prend en compte les résidus. Le graphique des scores de l'APCA ne montre pas seulement les moyennes de chaque niveau de l'effet mais aussi les groupes d'observations liées aux différents niveaux.

— La méthode ASCA-E applique l'ACP sur la matrice d'effet "pure" $\hat{\mathbf{M}}_f$ pour estimer les *loadings* $\mathbf{P}_f$ mais la matrice des scores $\mathbf{T}_f$ est calculée autrement :

$$\mathbf{T}_f^E = (\hat{\mathbf{M}}_f + \hat{\mathbf{E}})\mathbf{P}_f = \hat{\mathbf{M}}_f \mathbf{P}_f + \hat{\mathbf{E}}\mathbf{P}_f \tag{2.1.11}$$

La projection $\mathbf{T}_f^E$ décrit la variation entre les niveaux dans l'espace des *loadings* $\mathbf{P}_f$ du modèle ACP pour la matrice d'effet $\mathbf{M}_f$. $\mathbf{T}_f^E$ n'a pas une variation maximale par rapport à l'APCA.

2.2 Package `limpca` existant

Le package `limpca` expliqué dans Thiel et al.(2023) [1] est un package en R qui implémente les méthodes ASCA+ et APCA+ pour pouvoir tester la significativité et l'importance ainsi que visualiser les différents effets d'un ensemble de données avec des graphiques et tables. Le package `limpca` cible des ensembles de données avec une structure bien précise à savoir peu d'observations, beaucoup de variables de réponse, ayant un plan d'expérience balancé ou non, et comportant uniquement des variables catégorielles (facteurs) à ce stade du développement. Cette structure de données se retrouve pour les données *-omiques* mais pas uniquement. `limpca` peut gérer toutes les formules de modèle linéaire général. Le package est disponible sur la page web par le lien `https://manonmartin.github.io/limpca/` et peut être téléchargé sur Rstudio par la commande suivante :

```
remotes::install_github("ManonMartin/limpca", dependencies = TRUE)
```

Dans ce package, en plus de toutes les fonctions, il est possible de retrouver 2 vignettes détaillées qui analysent les ensembles de données *UCH* et *Trout* mais aussi un manuel d'utilisateur.

Même si le package se base fortement sur l'ASCA+ et l'APCA+ de Thiel et al. (2017) [3], quelques améliorations ont été apportées à celui-ci. Comme l'ASCA+ et l'APCA+ ont été expliqués plus haut, la section suivante se limitera à expliquer les améliorations de l'ASCA+ et l'APCA+. Ensuite, l'organisation du package et l'ordre d'utilisation des objets et des fonctions seront présentés.

2.2.1 Méthodologie de `limpca`

La Figure 2.2.1 est reprise de Thiel et al.(2023) [1] et est basée sur Guisset et al.(2019) [14] et présente la structure générale de la méthodologie implémentée dans `limpca`. La première partie est la décomposition LM de la matrice de réponse $\mathbf{Y}$ en matrices d'effet $\hat{\mathbf{M}}$. Cette décomposition ce base sur la formule du modèle et le plan d'expériences. La deuxième partie permet de comprendre l'importance de chaque effet du modèle en calculant leur pourcentage de variance expliqué et en testant leur significativité via un test boostrap. La troisième partie permet de visualiser les différents effets du modèle grâce au graphique des scores et des *loadings*. Les scores et *loadings* ont été calculés par une ACP sur les matrices d'effets $\hat{\mathbf{M}}$ "pures" ou augmentées.

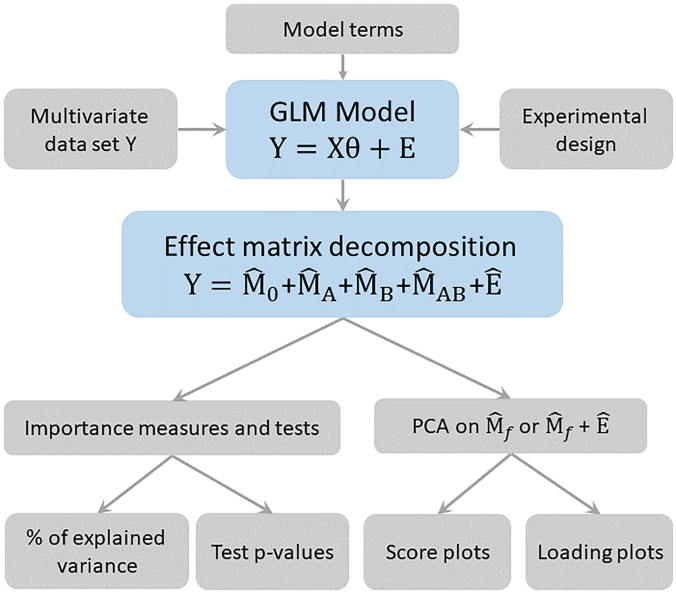


FIGURE 2.2.1 – Vue d'ensemble de l'ASCA+ et l'APCA+ dans `limpca`.

Il n'y a pas de changement majeur pour la première étape qui est la décomposition en matrices d'effets par LM déjà présentée dans la section 2.1. Il y a qu'une précision en plus qui permet de retrouver facilement la matrice du modèle $\mathbf{X}_f$ et la matrice de coefficient $\hat{\mathbf{\Theta}}_f$ pour l'effet f. La matrice de contraste $\mathbf{L}_f$ de dimensions $(p_f \times p)$ est multipliée avec les matrices complètes $\hat{\mathbf{\Theta}}$ et $\mathbf{X}$ pour obtenir $\mathbf{X}_f$ et $\hat{\mathbf{\Theta}}_f$:

$$\mathbf{X}_f = \mathbf{X}\mathbf{L}_f' \quad \text{et} \quad \hat{\mathbf{\Theta}}_f = \mathbf{L}_f\hat{\mathbf{\Theta}} \tag{2.2.1}$$

La matrice contraste $\mathbf{L}_f$ permet de sélectionner les p_f lignes de la matrice $\hat{\mathbf{\Theta}}_f$ dans la matrice complète $\hat{\mathbf{\Theta}}$. Elle permet aussi de sélectionner les p_f colonnes de la matrice $\mathbf{X}_f$ dans la matrice complète $\mathbf{X}$. La formule pour estimer la matrice d'effet $\hat{\mathbf{M}}_f$ devient alors :

$$\hat{\mathbf{M}}_f = \mathbf{X}\mathbf{L}_f'\mathbf{L}_f\hat{\mathbf{\Theta}} \tag{2.2.2}$$

Dans la deuxième étape, La formule pour calculer le pourcentage de variance expliqué $\%Var_f$ pour un effet f, se base toujours sur le principe du *type III SS* mais n'est plus la même que dans l'ASCA+ et l'APCA+ de Thiel et al. (2017) [3]. Cette nouvelle formule permet de ne pas devoir estimer tous les sous-modèles ne contenant pas un effet. Le pourcentage de variance expliqué $\%Var_f$ pour un effet f est défini par :

$$\%\text{Var}_f = \frac{\text{tr}\left(\hat{\mathbf{\Theta}}'\mathbf{L}_f'(\mathbf{L}_f(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}_f')^{-1}\mathbf{L}_f\hat{\mathbf{\Theta}}\right)}{\|\mathbf{Y} - \hat{\mathbf{M}}_0\|^2} * 100 \tag{2.2.3}$$

où $\mathbf{L}_f$ est la matrice de contraste permettant de sélectionner dans $\mathbf{X}$ et $\hat{\mathbf{\Theta}}$, les colonnes/-lignes en lien avec l'effet f.

L'ASCA+ et l'APCA+ de Thiel et al. (2017) [3] proposent les tests de permutation pour calculer la significativité d'un effet. Dans le package `limpca`, un test bootstrap est implémenté au lieu des tests de permutation car ces derniers sont dépendants par rapport à la structure du modèle alors que c'est bien le cas pour le test bootstrap. Dans le package `limpca`, les étapes de la procédure du test bootstrap pour tester la significativité de l'effet f sont :

1. Le modèle sous l'hypothèse H_0 est défini en enlevant l'effet f de la matrice $\mathbf{X}$. Ce modèle est défini par :
$$\mathbf{Y} = \mathbf{X}_{/f}\hat{\mathbf{\Theta}}_{/f} + \mathbf{E}_{/f} \tag{2.2.4}$$
2. Ajuster ce modèle sous H_0 et le modèle sous H_1 (modèle complet avec tous les effets) sur la matrice de réponse $\mathbf{Y}$
3. Calculer le test statistique observé :
$$F_{obs} = \frac{\left(\|\hat{\mathbf{E}}_{/f}\|^2 - \|\hat{\mathbf{E}}_{full}\|^2\right)/(p - p_{/f})}{\|\hat{\mathbf{E}}_{full}\|^2/(n-p)} = \frac{\text{tr}\left(\hat{\mathbf{\Theta}}'\mathbf{L}_f'(\mathbf{L}_f(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}_f')^{-1}\mathbf{L}_f\hat{\mathbf{\Theta}}\right)/p_f}{\|\hat{\mathbf{E}}_{full}\|^2/(n-p)} \tag{2.2.5}$$
où $p_{/f}$ est le nombre de paramètres dans le modèle sans l'effet f. $(n - p_{/f})$ peut être simplifié par p_f. Comme pour le test bootstrap, $\left(\|\hat{\mathbf{E}}_{/f}\|^2 - \|\hat{\mathbf{E}}_{full}\|^2\right)$ est remplacé par $\text{tr}\left(\hat{\mathbf{\Theta}}'\mathbf{L}_f'(\mathbf{L}_f(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}_f')^{-1}\mathbf{L}_f\hat{\mathbf{\Theta}}\right)$ appelé *type III* SS_f. Ce qui permet de ne pas devoir à nouveau calculer cette valeur *type III* SS_f puisqu'elle a déjà été calculée pour le pourcentage de variance expliqué de l'effet f. Le calcul de F_{obs} a été optimisé dans `limpca` en faisant des calculs parallèles en R.
4. Calculer les valeurs prédites $\hat{\mathbf{Y}}_{/f}$ et les résidus $\hat{\mathbf{E}}_{/f}$ du modèle sous H_0 :
$$\hat{\mathbf{Y}}_{/f} = \mathbf{X}_{/f}\hat{\mathbf{\Theta}}_{/f} \quad \text{et} \quad \hat{\mathbf{E}}_{/f} = \mathbf{Y} - \hat{\mathbf{Y}}_{/f}. \tag{2.2.6}$$
Ces valeurs seront utilisées dans la boucle bootstrap.
5. Répéter les étapes suivante pour un grand nombre B d'itérations ($b = 1, ..., B$ est l'indice de l'itération) :
   (a) Construire une matrice $\hat{\mathbf{E}}_b$ sur base d'un échantillon aléatoire avec remise tiré dans les n lignes de $\hat{\mathbf{E}}_{/f}$.
   (b) Générer la matrice de réponses bootstrap $\mathbf{Y}_b$:
   $$\mathbf{Y}_b = \hat{\mathbf{Y}}_{/f} + \hat{\mathbf{E}}_b \tag{2.2.7}$$
   (c) Ajuster le modèle sous H_0 et sous H_1 sur la matrice des nouvelles réponses $\mathbf{Y}_b$
   (d) Calculer la statistique de test :
   $$F_b = \frac{\text{tr}\left(\hat{\mathbf{\Theta}}_b'\mathbf{L}_f'(\mathbf{L}_f(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}_f')^{-1}\mathbf{L}_f\hat{\mathbf{\Theta}}_b\right)/p_f}{\|\hat{\mathbf{E}}_{/full}\|^2/(n-p)} \tag{2.2.8}$$

où $\hat{\mathbf{\Theta}}_b$ est la matrice des coefficients bootstrap.

6. Calculer la p-valeur de l'effet f :

$$\text{p-valeur}_f^{boot} = \frac{\sum_{b=1}^{B} I(F_b \geq F_{obs}) + 1}{B + 1} \tag{2.2.9}$$

Pour la troisième partie, il n'y a pas de changement par rapport à l'ASCA+ et l'APCA+, il est possible de faire une ACP sur les matrices d'effet $\hat{\mathbf{M}}_f$ du modèle avec les méthodes ASCA,APCA et ASCA-E.

Dans `limpca`, il est possible de combiner un ensemble d'effets g du modèle. Pour combiner des effets et calculer la matrice d'effets $\hat{\mathbf{M}}_g$, il faut garder toutes les colonnes/lignes dans $\mathbf{X}/\hat{\Theta}$ qui ont un lien avec un effet contenu dans l'ensemble d'effets g. Le modèle sans g est obtenu en enlevant toutes les colonnes/lignes de $\mathbf{X}/\hat{\mathbf{\Theta}}$ qui ont un lien avec un effet contenu dans l'ensemble d'effets g. Avec cette logique, il est possible de calculer le pourcentage de variance expliqué de g, de tester la significativité de la combinaison d'effets g avec le test bootstrap et de faire une décomposition avec l'ACP pour visualiser les scores et les *loadings* de g.

2.2.2 Objets et fonctions dans `limpca`

Cette section présente, dans l'ordre d'utilisation, les objets et les fonctions contenus dans `limpca`. Sur la Figure 2.2.2 venant de Thiel et al.(2023) [1], on peut voir que le package `limpca` est organisé en 2 ensembles de fonctions : l'un pour l'exploration des données et l'autre pour l'ajustement d'un modèle et la visualisation. Sur la Figure 2.2.2, les objets sont présentés dans des ovales bleus alors que les fonctions sont reprises dans des rectangles verts.

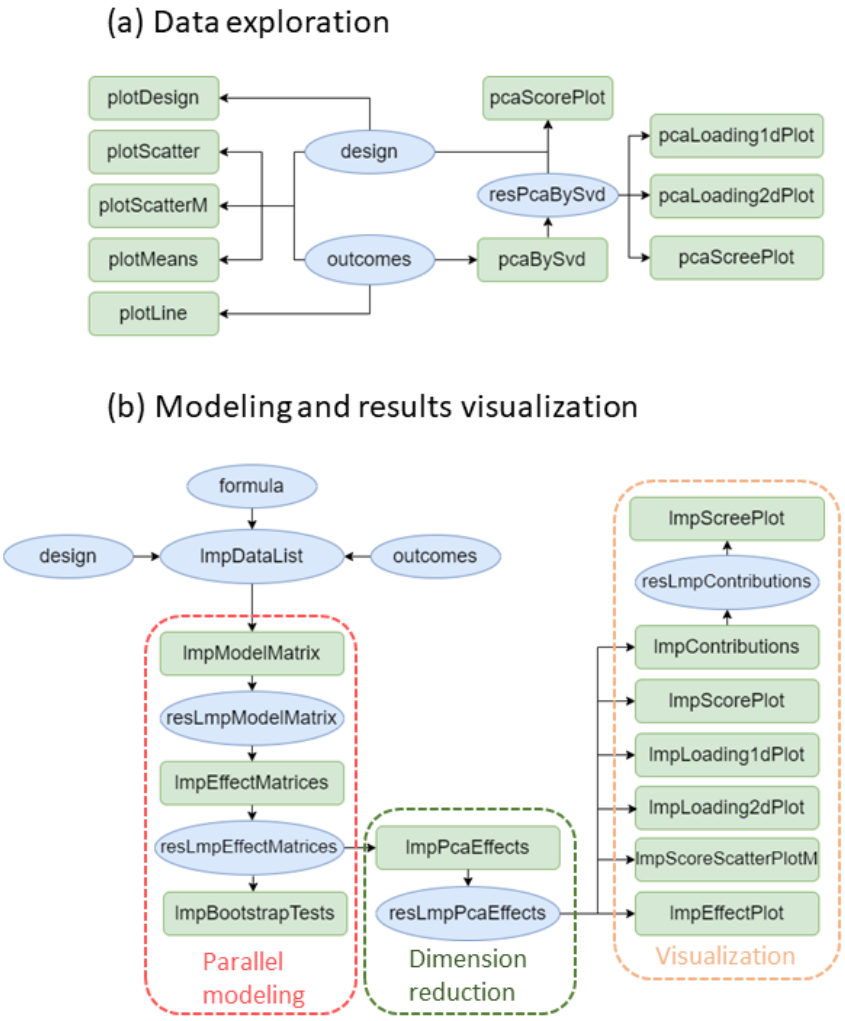


FIGURE 2.2.2 – Vue d'ensemble des fonctions et objets principaux de `limpca`.

La première partie du package `limpca` Figure(2.2.2.(a)) est dédiée à l'exploration des données. Les objets utiles pour cette partie sont les données du plan d'expérience `design` de dimension $(n \times k)$ et la matrice des réponses `outcomes` de dimension $(n \times m)$. k est le nombre de facteurs dans le plan et m est le nombre de réponses.

La liste des fonctions de la partie (a) est :

— **plotDesign** permet de vérifier que le plan est bien balancé ou non et de visualiser les différents niveaux en affichant un graphique du plan expérimental.

— **plotScatter** permet d'afficher la relation entre deux colonnes réponses de la matrice **Y**. Il est possible de choisir des couleurs et/ou des formes pour les différents niveaux d'un facteur. Le facteur en relation avec la couleur n'est pas forcément le même que la forme. Il y a aussi une possibilité de regrouper les observations d'un même niveau de facteur avec des segments, des ellipses ou encore des polygones.

— **plotScatterM** permet d'afficher une matrice de graphiques. Chaque graphique montre la relation entre deux colonnes de la matrice de réponses Y sélectionnée. Il est possible d'y identifier les niveaux de quatre facteurs maximums : deux pour la couleur et la forme au dessus de la diagonale et deux pour la couleur et la forme en dessous de la diagonale.
— **plotMeans** permet de représenter les différentes moyennes des niveaux de facteurs pour une réponse de $\mathbf{Y}$. Attention à l'interprétation, surtout pour un plan non balancé.
— **plotLine** permet de visualiser les réponses de $\mathbf{Y}$ pour une ou plusieurs observations. Le graphique aura des lignes, segments ou points suivant le type de réponse.
— **pcaBySvd** permet de faire une réduction de dimension ACP sur $\mathbf{Y}$ par la décomposition en valeurs singulières(SVD). Cette fonction crée un objet `resPcaBySvd` contenant les résultats de l'ACP.
— **pcaScorePlot** permet de visualiser les scores de l'objet `resPcaBySvd` avec les mêmes options graphiques que le `plotScatter`.
— **pcaScreePlot** permet de visualiser sous forme d'un graphique en barres le pourcentage de variance expliqué par chaque composante principale (PC) se trouvant dans l'objet `resPcaBySvd`. Les variances sont affichées dans l'ordre décroissant.
— **pcaLoading1dPlot** permet de visualiser les *loadings* de la PCA, pour une ou plusieurs PCs, avec les mêmes options graphiques que `plotLine`.
— **pcaLoading2dPlot** permet d'afficher la relation entre les *loadings* de deux PCs avec les mêmes options graphiques que dans `plotScatter`.

La deuxième partie du package `limpca` Figure(2.2.2.(b)) est dédiée à la construction du modèle et à la visualisation des résultats. Elle est constituée de 3 parties :

1. La construction du modèle avec les matrices d'effets $\hat{\mathbf{M}}_f$ et les tests de significativité sur les effets f. Cette partie est entourée en rouge pointillé sur la Figure 2.2.2.(b).
2. La réduction de matrice avec, comme méthode possible, ASCA, APCA ou ASCA-E sur chaque matrice d'effet $\hat{\mathbf{M}}_f$ et une ASCA sur la matrice des résidus $\hat{\mathbf{E}}$. Cette partie est entourée en vert pointillé sur la Figure 2.2.2.(b).
3. La visualisation des résultats de la réduction de dimension est disponible via les fonctions dans la partie entourée en jaune pointillé sur la Figure 2.2.2.(b).

Dans la partie (b) (La modélisation), les fonctions doivent être exécutées dans l'ordre montré sur la figure. Cela permet d'appliquer, étape par étape, la méthodologie discutée dans la section précédente. L'utilisateur doit, en premier lieu, créer une liste `lmpDataList` qui sera mise en paramètre de la fonction `lmpModelMatrix`. La liste `lmpDataList` comporte le plan d'expérience dans un tableau de données `design`, la matrice de réponses $\mathbf{Y}$ ou `outcomes` et la formule `formula` de modèle à construire. L'ordre des fonctions de la partie (b) est le suivant :

— **lmpModelMatrix** permet de construire la matrice de modèle complète $\mathbf{X}$ et une liste de matrices pour chaque effet $\mathbf{X}_f$. `lmpModelMatrix` renvoie un objet

`resLmpModelMatrix`. L'objet `resLmpModelMatrix` sera mis en paramètre de la fonction `lmpEffectMatrices`.

— **lmpEffectMatrices** permet d'estimer les matrices pour chaque effet $\hat{\mathbf{M}}_f$ et la matrice des résidus $\hat{\mathbf{E}}$. Elle permet aussi de calculer le pourcentage de variance expliqué de type III pour chaque effet. Les résultats de cette fonction sont mis dans l'objet `resLmpEffectMatrices`.

— **lmpBootstrapTests** permet, via l'objet `resLmpEffectMatrices`, de calculer une p-valeur pour chaque effet afin de déterminer quels effets sont significatifs dans le modèle. Il est aussi possible de tester des combinaisons d'effets.

— **lmpPcaEffects** permet d'exécuter une ACP sur chaque matrice d'effets de l'objet `resLmpEffectMatrices` par la méthode ASCA, APCA ou ASCA-E. Il est aussi possible d'exécuter une ACP sur des effets combinés qui seront ajoutés à la liste de résultats nommée `resLmpPcaEffects`.

— **lmpContributions** permet, via des tableaux et des graphiques mis dans une liste `resLmpContributions`, de visualiser la contribution de chaque effet par rapport à la variance totale. Cette fonction permet aussi de calculer la contribution de chaque PC par rapport au total de variances par effet et par rapport à la variance totale. Cette fonction se base sur l'objet `resLmpPcaEffects`.

— **lmpScreePlot** permet de visualiser sur différents graphiques en barres (un graphique par effet pour les effets demandés par l'utilisateur) le pourcentage de variance de chaque PC via les résultats dans l'objet `resLmpContributions`.

— **lmpScorePlot** permet d'afficher les scores de 2 composantes principales pour chaque effet du modèle et chaque combinaison d'effets qui se trouvent dans l'objet `resLmpPcaEffects`. Cette fonction utilise les mêmes options que la fonction `plotScatter`.

— **plotScoreScatterPlotM** permet d'afficher une matrice de graphiques X-Y qui associent les scores des effets deux à deux. Chaque graphique affiche les scores entre une PC d'un effet et une PC d'un autre effet du modèle. Pour chaque effet, l'utilisateur décide quelle PC afficher. Cette fonction a les mêmes options que `plotScatterM` et peut différencier au maximum les niveaux de quatre facteurs différents.

— **lmpLoading1dPlot** permet d'afficher les *loadings* d'une ou plusieurs PCs pour chaque effet et chaque combinaison d'effets se trouvant dans l'objet `resLmpPcaEffects`. Le graphique utilise les mêmes options que `plotLine`.

— **lmpLoading2dPlot** permet d'afficher un graphique des *loadings* en deux dimensions pour chaque effet et combinaison d'effets dans l'objet `resLmpPcaEffects`. Cette fonction comporte les mêmes options que `plotScatter`.

— **lmpEffectPlot** permet de visualiser les termes d'interactions ou les combinaisons d'effets avec un graphique en lignes. Cette fonction se base sur l'objet `resLmpPcaEffects` et ne fonctionne que pour les scores calculés par la méthode ASCA. Le graphique se base sur une seule PC et les niveaux des facteurs de l'interaction ou de la combinaison d'effets. `lmpEffectPlot` utilise les mêmes options que `plotMeans`.

2.3 LMM

Un petit rappel sur les modèles mixtes venant du chapitre 16 du livre *Comprehensive Chemometrics* par Govaerts et al. [4]. Les modèles linéaires mixtes (LMM) sont plus généraux qu'un modèle ANOVA. Ils ont aussi une plus grande capacité à traiter des plans non équilibrés ainsi que des équations de modèle et des structures de covariance plus avancées. Le LMM différencie les effets fixes $\mathbf{X}\boldsymbol{\beta}$, les effets aléatoires $\mathbf{Z}\boldsymbol{\alpha}$ et les erreurs $\boldsymbol{\epsilon}$. La formule pour un LMM peut être écrite :

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\alpha} + \boldsymbol{\epsilon} \tag{2.3.1}$$

avec $\mathbf{Z}$ étant la matrice du modèle pour les effets aléatoires de dimensions $(n \times q)$, $\boldsymbol{\alpha}$ le vecteur des coefficients aléatoires de dimensions $(q \times 1)$, le vecteur réponse $\boldsymbol{y}$ de dimensions $(n \times 1)$ et le vecteur des résidus $\boldsymbol{\epsilon}$ de dimension $(n \times 1)$. La matrice de modèle des effets fixes $\mathbf{X}$ est construite de la même façon que dans ASCA+ et elle peut être donc décomposée en $F + 1$ blocs pour chaque effet fixe. Pour la matrice $\mathbf{Z}$, le nombre de colonnes est la somme des niveaux de tous les effets aléatoires et peut aussi être décomposée en R blocs pour chaque effet aléatoire :

$$\mathbf{Z} = (\mathbf{Z}_1|\mathbf{Z}_2|...|\mathbf{Z}_R). \tag{2.3.2}$$

La formule (2.3.1) prend comme hypothèse que $\boldsymbol{\alpha} \sim N(\mathbf{0}, \mathbf{G})$ et $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$. La matrice $\mathbf{G}$ est la matrice des variances-covariances des effets aléatoires de dimensions $(q \times q)$ et la matrice $\boldsymbol{\Sigma}$ est la matrice des variances-covariances des résidus de dimensions $(n \times n)$. Il est supposé ici que le modèle ne prend en compte que des facteurs fixes ou aléatoires et qu'il n'y a pas de covariance entre deux effets aléatoires différents, ce qui donne comme variance pour les effets aléatoires :

$$\text{var}(\boldsymbol{\alpha}) = \mathbf{G} = \left\{\sigma_r^2 \mathbf{I}_{q_r}\right\}_{r=1}^{R} \tag{2.3.3}$$

où R est le nombre d'effets aléatoires et q_r est le nombre de niveaux pour l'effet aléatoire r. La variance des résidus est décrite aussi :

$$\text{var}(\boldsymbol{\epsilon}) = \boldsymbol{\Sigma} = \sigma_\epsilon^2 \mathbf{I}_n \tag{2.3.4}$$

$\mathbf{I}_n$ est la matrice identité de dimension n. Les composantes de la variance pour les effets aléatoires sont dans le vecteur $\hat{\boldsymbol{\gamma}} = (\hat{\sigma_1}^2, ..., \hat{\sigma_R}^2)$ et sont estimées par la méthode du maximum de vraisemblance restreinte (REML).

La matrice de variance-covariance de $\mathbf{y}$ notée $\mathbf{V}$ est :

$$\mathbf{V} = \text{var}(\mathbf{y}) = \text{var}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\alpha} + \boldsymbol{\epsilon}) = \mathbf{Z}\mathbf{G}\mathbf{Z}' + \boldsymbol{\Sigma} = \sum_{r=1}^{R} \sigma_r^2 \mathbf{Z}_r \mathbf{Z}_r' + \sigma_\epsilon^2 \mathbf{I}_n \tag{2.3.5}$$

L'espérance de $\mathbf{y}$ est $\mathbb{E}[\boldsymbol{y}] = \mathbf{X}\boldsymbol{\beta}$, ce qui permet de dire que $\boldsymbol{y} \sim N(\mathbf{X}\boldsymbol{\beta}, \mathbf{ZGZ'} + \boldsymbol{\Sigma})$.

L'ajustement du modèle est séparé en 3 parties :

1. L'estimation des effets fixes $\hat{\boldsymbol{\beta}}$ par la méthode du maximum de vraisemblance (ML) construit par la formule :

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{y}. \tag{2.3.6}$$

2. L'estimation des paramètres de variance $(\hat{\boldsymbol{\gamma}}, \hat{\sigma_\epsilon}^2)$ qui constituent $\hat{\mathbf{V}}$ se fait par la méthode du maximum de vraisemblance restreinte(REML).
3. La prédiction des coefficients des effets aléatoires $\hat{\boldsymbol{\alpha}}$ nommé (EBLUPs : Empirical Best Linear Unbiased Predictors) par la méthode ML :

$$\hat{\boldsymbol{\alpha}} = \hat{\mathbf{G}}\mathbf{Z}'\hat{\mathbf{V}}^{-1}(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) \tag{2.3.7}$$

2.4 LiMM-PCA

Dans cette section, la méthode de LiMM-PCA est expliquée de façon synthétique en se basant sur l'article de Martin & Govaerts (2020) [2]. LiMM-PCA est l'extension de l'ASCA+ au modèle linéaire mixte (LMM). Le but de ce mémoire est donc de l'implémenter dans le package `limpca`.

La méthode LiMM-PCA se base sur l'ASCA+ au lieu de l'ASCA car ce dernier présente quelques lacunes notamment pour l'estimation de la significativité des effets de modèles linéaires mixtes.

La décomposition ASCA se base sur l'estimation des variances. L'estimation de la variance d'un effet g est calculée avec l'équation :

$$\text{var}(g) = \text{MS}_g = \frac{\text{SS}_g}{\text{df}_g} \tag{2.4.1}$$

où MS_g représente les carrés moyens de l'effet, SS_g est la somme des carrés et df_g est le degré de liberté de l'effet. Toutefois, l'estimation de la variance des effets aléatoires ignore cette formule. Comme la décomposition ASCA en matrice d'effets utilise l'estimation des variances, les matrices d'effets aléatoires $\mathbf{M}_r$ ne peuvent être utilisées pour interpréter les effets aléatoires r. Pour résoudre ce problème, il faut corriger les variances estimées en utilisant, non plus les carrés moyens, mais leur espérance $\mathbb{E}(MS)$, ce qui est beaucoup plus compliqué. La méthode ASCA peut utiliser le test-F comme test de significativité d'un effet en calculant normalement le F-ratio = MS_g/MS_E. Mais le test F doit être également modifié car la correction des MS en $\mathbb{E}(MS)$ modifie le calcul pour s'assurer que F-ratio est égale à 1 lorsqu'il n'y a pas d'effet. Pour avoir un test F valide, il faut trouver la bonne matrice à mettre au dénominateur du F-ratio pour chaque effet. Cela implique que le dénominateur ne sera plus forcément la matrice des résidus. Le F-ratio est aussi utilisé dans d'autres formules, notamment la matrice d'effet augmentée ASCA-E. Avec toutes ces modifications, il devient encore plus compliqué d'estimer un modèle quand le plan d'expérience n'est pas balancé. C'est pour cela que l'ASCA+ qui utilise les LM est utilisé au lieu

de l'ASCA qui utilise l'ANOVA et permet d'expliquer des modèles linéaires mixtes avec un grand nombre de réponses et un plan d'expérience non balancé.

La figure 2.4.1 est la figure 1 de l'article de Martin & Govaerts (2020) [2]. Il s'agit d'un résumé des étapes de la méthode LiMM-PCA. Il montre aussi les modifications et ajouts apportés par rapport à la méthode ASCA+ en jaune.

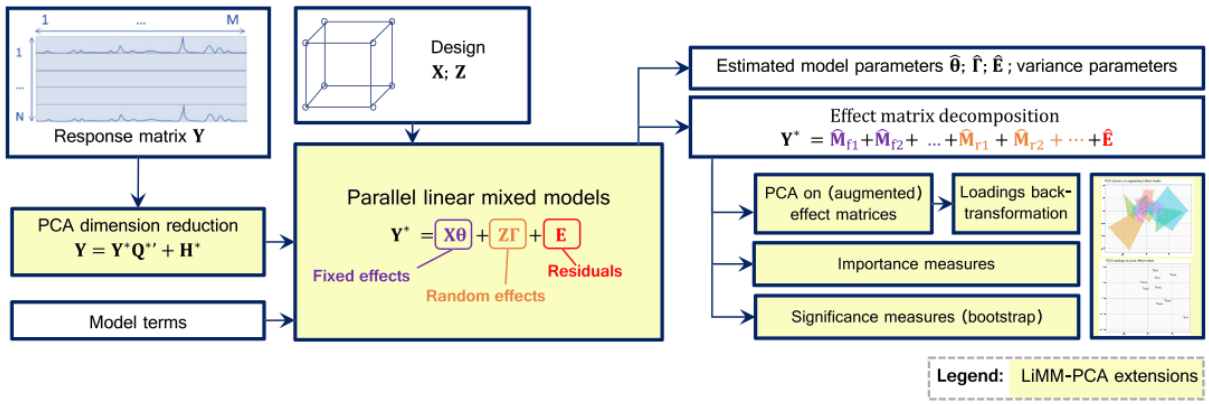


FIGURE 2.4.1 – Etapes de la méthode LiMM-PCA.

2.4.1 Etape 1 : Dimension réduction de la matrice de réponse par ACP

Pour que la méthode LiMM-PCA fonctionne, il faut que la matrice de réponse $\mathbf{Y}$ respecte les hypothèses d'un modèle linéaire mixte, c'est-à-dire que les données de chaque colonne de la matrice de réponse doivent suivre plus ou moins une distribution normale. Si cela n'est pas le cas, il faut alors faire une transformation mathématique sur $\mathbf{Y}$ (exemple : `ln`). Il est recommandé d'appliquer une ACP sur les $\mathbf{Y}$, ce qui permet, d'une part, de réduire le nombre de variables surtout quand elles sont plus nombreuses que les observations et, d'autre part, rendre les variables non-corrélées entre elles et se rapprocher de l'indépendance. La réduction de dimension ACP sur la matrice $\mathbf{Y}$ s'écrit :

$$\mathbf{Y} = \mathbf{Y}^*\mathbf{Q}^{*'} + \mathbf{H}^* \tag{2.4.2}$$

où $\mathbf{Y}^*$ est la nouvelle matrice de réponses et la matrice de scores de la ACP de dimensions $(n \times m*)$, $\mathbf{Q}^*$ est la matrice des *loadings* de dimensions $(m \times m*)$ et enfin $\mathbf{H}^*$ est la matrice des résidus contenant l'information qui n'a pas été captée par les $m*$ premières PC de dimension $(n \times m)$. La transformation fait passer le nombre de variables de m à $m*$.

Après ces deux transformations,il est considéré dans LiMM-PCA que la matrice $\mathbf{Y}$ a comme hypothèse que les m réponses sont indépendantes entre elles et normales. Si une transformation ACP est appliquée, lors de la visualisation des matrices d'effets (point 5), il faudra

projeter les données dans l'espace de réponse de départ en multipliant les *loadings* $\mathbf{Q}^*$ afin d'exprimer les résultats par rapport aux variables de départ.

2.4.2 Etape 2 : Modélisation parallèle mixte

Pour chaque variable de réponse $\mathbf{y}_j^*$ de la nouvelle matrice de réponse $\mathbf{Y}^*$, un modèle linéaire mixte sera ajusté par la méthode de maximum de vraisemblance restreinte (REML) :

$$\mathbf{y}_j^* = \mathbf{X}\boldsymbol{\beta}_j + \mathbf{Z}\boldsymbol{\alpha}_j + \boldsymbol{\epsilon}_j \text{ pour } j = 1, ..., m^* \tag{2.4.3}$$

Les matrices de modèles $\mathbf{X}$ pour les effets fixes et $\mathbf{Z}$ pour les effets aléatoires sont construites en se basant, comme la méthode ASCA+, sur le plan d'expérience. Les matrices de modèles $\mathbf{X}$ et $\mathbf{Z}$ sont organisées en F blocs d'effets et R blocs d'effets respectivement :

$$\mathbf{X} = (\mathbf{X}_0|\mathbf{X}_1|...|\mathbf{X}_F) \text{ et } \mathbf{Z} = (\mathbf{Z}_1|\mathbf{Z}_2|...|\mathbf{Z}_R). \tag{2.4.4}$$

$\mathbf{X}_0$ est le terme constant qui n'est pas toujours nécessaire si le plan est balancé dans le cas où les réponses résultant de l'ACP sont centrées. Pour chaque réponse, les paramètres fixes sont estimés comme $\hat{\boldsymbol{\beta}}_j = (\mathbf{X'}\hat{\mathbf{V}}_j^{-1}\mathbf{X})^{-1}\mathbf{X'}\hat{\mathbf{V}}_j^{-1}\mathbf{y}_j^*$ et les prédictions aléatoires sont estimées comme $\hat{\boldsymbol{\alpha}}_j = \hat{\mathbf{G}}_j\mathbf{Z'}\hat{\mathbf{V}}_j^{-1}(\mathbf{y}_j^* - \mathbf{X}\hat{\boldsymbol{\beta}}_j)$, où la matrice $\hat{\mathbf{G}}_j$ est la matrice de covariance estimée des effets aléatoires pour la réponse j et $\hat{\mathbf{V}}_j$ est la matrice de covariance totale estimée pour la réponse j. Le modèle estime aussi, avec la méthode REML, les paramètres de variances des effets aléatoires $\hat{\alpha}_{rj}^2$ associés à chaque effet r et le paramètre de variance des résidus $\hat{\alpha}_{\epsilon j}^2$ pour la réponse j.

Le résultat de ces m^* estimations de modèles, peut être modélisé comme un seul modèle multivarié sous la forme :

$$\mathbf{Y}^* = \mathbf{X}\hat{\boldsymbol{\Theta}} + \mathbf{Z}\hat{\boldsymbol{\Gamma}} + \hat{\mathbf{E}} \tag{2.4.5}$$

où $\hat{\boldsymbol{\Theta}}$ est la matrice des paramètres fixes de dimensions $(p \times m^*)$ et $\hat{\boldsymbol{\Gamma}}$ est la matrice des prédictions aléatoires de dimensions $(q \times m^*)$. Comme les matrices $\mathbf{X}$ et $\mathbf{Z}$, les matrices $\hat{\boldsymbol{\Theta}}'$ et $\hat{\boldsymbol{\Gamma}}'$ sont décomposables en blocs : $\hat{\boldsymbol{\Theta}}' = (\hat{\boldsymbol{\Theta}}_0'|\hat{\boldsymbol{\Theta}}_1'|...|\hat{\boldsymbol{\Theta}}_F')$ et $\hat{\boldsymbol{\Gamma}}' = (\hat{\boldsymbol{\Gamma}}_1'|\hat{\boldsymbol{\Gamma}}_2'|...|\hat{\boldsymbol{\Gamma}}_R')$. La matrice des résidus $\hat{\mathbf{E}}$ de dimensions $(n \times m^*)$ est estimée par :

$$\hat{\mathbf{E}} = \mathbf{Y}^* - (\mathbf{X}\hat{\boldsymbol{\Theta}} + \mathbf{Z}\hat{\boldsymbol{\Gamma}}). \tag{2.4.6}$$

2.4.3 Etape 3 : Décomposition en matrices d'effets

Comme dans la méthode ASCA+, la matrice de réponse peut être décomposée en une somme de matrices d'effets fixes et aléatoires :

$$\mathbf{Y} = \hat{\mathbf{M}}_0 + \sum_{f=1}^{F} \hat{\mathbf{M}}_f + \sum_{r=1}^{R} \hat{\mathbf{M}}_r + \hat{\mathbf{E}} \tag{2.4.7}$$

La matrice d'effet $\hat{\mathbf{M}}_f = \mathbf{X}_f\hat{\mathbf{\Theta}}_f$ pour un effet fixe f et la matrice d'effet $\hat{\mathbf{M}}_r = \mathbf{X}_r\hat{\mathbf{\Theta}}_r$ pour un effet aléatoire r.

2.4.4 Etape 4 : Quantification de l'importance et test de significativité d'un effet

L'objectif de cette étape est de savoir quels sont les effets les plus importants dans notre modèle. Pour cela, deux approches sont réalisées. La première approche permet de calculer le pourcentage de variance expliqué par chaque effet du modèle. La deuxième approche permet via un test bootstrap de voir, pour chaque effet du modèle, si l'effet est significatif du point de vue statistique sur la matrice de réponse. Ces 2 approches permettent de mieux comprendre quels graphiques d'effets seront les plus importants à regarder et comment interpréter ceux-ci.

Cela n'est pas aussi direct de calculer la variance expliquée par effet dans un modèle linéaire mixte que dans une ANOVA ou un LM car il faut pouvoir comparer les variances aléatoires et les variances fixes sur la même échelle. Pour pallier ce problème, la méthode Nakagawa and Schielzeth(2013) [10] est utilisée. Cette méthode permet de trouver une solution pour quantifier les effets fixes et aléatoires dans le cas d'un plan balancé. L'estimation de la variance d'une réponse $\mathbf{y}_j^*$ sachant $\mathbf{X}$ est la somme des variances des effets aléatoires estimés par le modèle mixte avec la variance des résidus inclus :

$$\widehat{\text{var}(\mathbf{y}_j^*|\mathbf{X})} = \sum_{r=1}^{R} \hat{\sigma}_{rj}^2 + \hat{\sigma}_{\epsilon j}^2 \tag{2.4.8}$$

Le calcul de la variance d'un effet fixe f pour une réponse j est juste la variance de la j-ème colonne de la matrice d'effet $\hat{\mathbf{M}}_f$ notée $\hat{\mathbf{M}}_{fj}$:

$$\hat{\sigma}_{fj}^2 = \text{var}(\hat{\mathbf{M}}_{fj}) \tag{2.4.9}$$

La variance globale de la matrice de réponse Y^* est estimée en additionnant les variances des effets fixes et des effets aléatoires de toutes les réponses considérées comme indépendantes :

$$\widehat{\text{var}(\mathbf{Y}^*)} = \sum_{j=1}^{m^*} \widehat{\text{var}(\mathbf{y}_j^*)} = \sum_{j=1}^{m^*}(\sum_{f=1}^{F} \hat{\sigma}_{fj}^2 + \sum_{r=1}^{R} \hat{\sigma}_{rj}^2 + \hat{\sigma}_{\epsilon j}^2) \tag{2.4.10}$$

Le pourcentage de variance expliqué pour un effet g qu'il soit aléatoire ou fixe :

$$\%\widehat{\text{Var}(g)} = \frac{\widehat{\text{var}(g)} * 100}{\text{var}(\mathbf{Y}^*)} \text{ avec } \widehat{\text{var}(g)} = \sum_{j=1}^{m^*} \hat{\sigma}_{gj}^2 \tag{2.4.11}$$

Cette méthode fonctionne si le plan d'expérience est balancé. Par contre, cela pose problème lorsque le plan n'est pas balancé pour l'estimation des variances des effets fixes. L'estimation

de la variance des effets fixes $\widehat{\text{var}(f)}$ peut être adaptée sur base du type III SS comme suggéré dans Thiel et al.(2017) [11] :

$$\widehat{\text{var}(f)} = \frac{\text{tr}\left(\hat{\mathbf{\Theta}}'\mathbf{L}_f'(\mathbf{L}_f(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}_f')^{-1}\mathbf{L}_f\hat{\mathbf{\Theta}}\right)}{n} \tag{2.4.12}$$

où $\mathbf{L}_f$ est la matrice de contraste pour sélectionner les colonnes dans $\mathbf{X}$ et les lignes dans $\hat{\mathbf{\Theta}}$ en lien avec l'effet f.

Un test bootstrap peut être utilisé pour tester statistiquement si un effet ou une combinaison d'effets, qu'ils soient fixes ou aléatoires, est vraiment significatif. Le test bootstrap se base sur la méthode énoncée notamment dans Davison et al.(1997) [12]. Il n'est pas le même que celui expliqué dans ASCA+, celui-ci se concentre sur le rapport de vraisemblance entre le modèle complet avec l'effet à tester et le modèle sans l'effet à tester. L'utilisation du rapport de vraisemblance dans ces 2 tests se justifie par le fait que, pour des modèles linéaires mixtes univariés, c'est la procédure la plus générale. Il est étendu au modèle multivarié en effectuant, pour chaque réponse et chaque effet du modèle, un rapport de vraisemblance entre le modèle avec et sans l'effet. Pour que le test statistique donne des résultats corrects, il faut que les réponses $\mathbf{y}_j^*$ soient indépendantes entre elles, ce qui est approximativement le cas si la réduction en composantes principales de matrice $\mathbf{Y}$ est effectuée dans l'étape 1. La statistique du rapport de vraisemblance globale observé pour un effet ou un groupe d'effets g (GLLR_g^{obs}) est calculée simplement par la somme des rapports de vraisemblance de g pour chaque réponse $\mathbf{y}_j^*$:

$$\text{GLLR}_g^{obs} = \Lambda_g^{obs} = 2\left[\sum_{j=1}^{m^*}\left(\ln(L_{(H_1,jg)}) - \ln(L_{(H_0,jg)}\right)\right], \tag{2.4.13}$$

où $L_{(H_1,jg)}$ est la vraisemblance du modèle estimé pour la réponse j sous l'hypothèse H_1, ce qui veut dire que le modèle est complet et comprend l'effet ou le groupe d'effets g et $L_{(H_0,jg)}$ est la vraisemblance du modèle estimé pour l'effet j sous l'hypothèse H_0. Cela signifie que le modèle ne comporte pas g. La procédure de l'évaluation de la significativité de(s) effets(s) g dans le modèle estimé par LiMM-PCA via le test bootstrap est découpé en différentes étapes :

1. Définir le modèle sous l'hypothèse nulle en excluant du modèle complet le ou les effet(s) g :

$$\mathbf{Y}^* = \tilde{\mathbf{X}}\tilde{\mathbf{\Theta}} + \tilde{\mathbf{Z}}\tilde{\mathbf{\Gamma}} + \tilde{\mathbf{E}} \tag{2.4.14}$$

   avec des suppressions de colonnes dans les matrices de modèles $\tilde{\mathbf{X}}, \tilde{\mathbf{Z}}$ et des suppressions de lignes dans les matrices $\tilde{\mathbf{\Theta}}, \tilde{\mathbf{\Gamma}}$ par rapport au modèle global permettant de supprimer l'effet ou les effets g pour toutes les matrices avec la notation du tilde.

2. Ajuster le modèle sous l'hypothèse H_1 (le modèle complet) et le modèle sous l'hypothèse H_0 (le modèle sans g).

3. Calculer la statistique du GLLR observée nommée Λ_g^{obs}.
4. Garder les paramètres estimés du modèle sous H_0, ce qui veut dire la matrice des coefficients fixes estimés $\hat{\tilde{\mathbf{\Theta}}}$, les variances des effets aléatoires $\hat{\tilde{\sigma}}^2_{rj}$ et les variances des résidus $\hat{\tilde{\sigma}}^2_{\epsilon j}$ pour toutes les réponses j.
5. Répéter B fois les étapes suivantes :
   — Générer une nouvelle matrice réponse Y_b^*,

$$\mathbf{Y}_b^* = \tilde{\mathbf{X}}\tilde{\mathbf{\Theta}} + \tilde{\mathbf{Z}}\tilde{\mathbf{\Gamma}}_b + \tilde{\mathbf{E}}_b \tag{2.4.15}$$

   avec, pour chaque boucle $b = 1, ..., B$, une nouvelle matrice $\tilde{\mathbf{\Gamma}}_b$ générée aléatoirement suivant une distribution normale de moyenne nul et des variances issues des paramètres $\hat{\tilde{\sigma}}^2_{rj}$. Une nouvelle matrice $\tilde{\mathbf{E}}_b$ est aussi générée aléatoirement suivant une moyenne de 0 et de variances issues des paramètres $\hat{\tilde{\sigma}}^2_{\epsilon j}$. Cette méthode est sans biais puisque la matrice de réponses $\mathbf{Y}^*$ a été orthogonalisée et cette opération permet de supposer que les réponses sont indépendantes deux à deux. La méthode pour construire la matrice $\tilde{\mathbf{E}}_b$ n'est pas la même que dans le test bootstrap du package `limpca` qui prend un échantillon aléatoire de taille n tiré avec remise dans les lignes de $\hat{\mathbf{E}}_{/f}$.
   — Estimer les 2 modèles sous l'hypothèse H_0 et l'hypothèse H_1 avec, comme réponse, la nouvelle matrice générée $\mathbf{Y}_b^*$ et calculer la statistique du rapport de vraisemblance générale entre ces 2 modèles pour la boucle b nommé Λ_g^b.
6. Calculer la p-valeur pour le test bootstrap défini par Davison et al.(1997) [12] dans le chapitre 4 :

$$\text{p-valeur}_g^{boot} = \frac{\sum_{b=1}^{B} I\left(\Lambda_g^b \geq \Lambda_g^{obs}\right) + 1}{B + 1} \tag{2.4.16}$$

Cette procédure peut être utilisée pour chaque effet du modèle, qu'il soit aléatoire ou fixe, et permet avec la p-valeur du test de confirmer ou d'infirmer que l'effet est significatif d'un point de vue statistique. Théoriquement il est possible de faire des tests bootstrap sur des effets combinés mais cela n'a pas été traité ou testé dans ce mémoire.

2.4.5 Etape 5 : Visualisation des matrices d'effet

Cette étape permet d'illustrer les résultats du modèle via un ensemble de graphiques. Pour visualiser les résultats, LiMM-PCA utilise une méthodologie similaire à ASCA. Celle-ci effectue une décomposition d'analyse en composante principale sur chaque matrice d'effet $\hat{\mathbf{M}}_g$ pour l'effet g :

$$\hat{\mathbf{M}}_g = \mathbf{T}_g\mathbf{P}_g^* \tag{2.4.17}$$

où $\mathbf{T}_g$ est la matrice des scores "pure", de dimensions $(n \times c_g)$ et $\mathbf{P}_g^*$ est la matrice des *loadings* de dimensions $(c_g \times m^*)$. c_g est le rang de la matrice d'effet $\hat{\mathbf{M}}_g$.

Comme la matrice de réponse $\mathbf{Y}$ a subi une transformation par ACP, ce qui a permis de

construire une matrice de réponse orthogonale et réduite $\mathbf{Y}^*$, il faut la "rétro-transformer" (*back-transformed*) pour l'interpréter dans l'espace des réponses initiales. La nouvelle matrice des *loadings* $\mathbf{P}_g$ de dimensions $(c_g \times m)$ est obtenue en multipliant les *loadings* de cette étape $\mathbf{P}_g^*$ par les *loadings* de l'étape 1 $\mathbf{Q}^{*'}$:

$$\mathbf{P}_g = \mathbf{P}_g^* \mathbf{Q}^{*'}. \tag{2.4.18}$$

En ASCA+, il est possible de calculer la matrice d'effet augmentée par les résidus. Cependant, en LiMM-PCA, les scores augmentés avec, au minimum, un effet aléatoire peuvent poser problème car il n'y a plus une seule source de variabilité venant des résidus mais il y en a aussi venant des effets aléatoires. Par conséquent, les scores ne seront plus toujours augmentés par les résidus mais peuvent être augmentés par une matrice d'effet aléatoire. Tout dépend si la variance d'un effet fixe ou aléatoire dépend de la variance d'un effet aléatoire. Pour trouver la matrice d'effet aléatoire $\mathbf{M}_{\text{add}}$ avec laquelle augmenter la matrice d'un effet $\mathbf{M}_g$, il faut regarder quelle matrice est mise au dénominateur du F-ratio du test F du modèle d'ANOVA correspondant tout en restant cohérent avec la valeur du F-ratio. Donc, s'il n'y a pas d'effet g, le F-ratio se retrouve à 1. Pour plus de détails sur le choix de la matrice d'effet à augmenter $\tilde{\mathbf{M}}_g$ et de la matrice d'effet à ajouter $\mathbf{M}_{\text{add}}$, voir la section 4.2.8. En plus de ne pas forcément augmenter la matrice d'effet avec la matrice de résidus, il faut prendre en compte les degrés de liberté (df) et le quantile de la distribution F utilisée dans la formule du test F. La formule proposée pour augmenter la matrice d'effet $\mathbf{M}_g$ est construite :

$$\tilde{\mathbf{M}}_g = \mathbf{M}_g + \sqrt{\frac{\text{df}_g}{\text{df}_{\text{add}}} F_{(\text{df}_g, \text{df}_{\text{add}}, 1-\alpha)}} * \mathbf{M}_{\text{add}} \tag{2.4.19}$$

$F_{\text{df}_g, \text{df}_{\text{add}}, 1-\alpha}$ représente le (1 - α)ième percentile de la distribution $F_{(\text{df}_g, \text{df}_{\text{add}})}$. Comme les effets aléatoires ne sont pas estimés à partir de la simple décomposition de l'ANOVA, mais avec les prédictions des effets aléatoires, cela veut dire que les degrés de liberté ne sont pas à la bonne échelle. Les degrés de liberté, aussi appelé les dimensions effectives (ED), peuvent être calculés en suivant la méthodologie de Eilers (2018) [13]. Une dimension effective prend la même valeur qu'un degré de liberté dans un modèle ANOVA seulement pour un effet fixe f qui est en réalité le nombre de coefficients de l'effet f moins 1. La dimension effective pour un effet aléatoire r, par contre, peut être n'importe quelles valeurs entre 0 et le nombre de coefficients pour l'effet r. Tout dépend de la valeur de la variance associée σ_r^2 à l'effet r.

Chapitre 3

Comparaison et visualisation des sorties Candies avec `limpca`, LiMM-PCA et l'extension de `limpca`

Ce chapitre compare :

- les différentes sorties de l'extension de package `limpca` aux modèles linéaires mixtes (LMM) se trouvant dans le fichier `test_limpca.html` du GitHub et dans l'annexe du mémoire
- les sorties du package de `limpca` se trouvant dans le fichier `candies_limpca.html` du GitHub
- les sorties se basant sur le code de M.Martin pour sa thèse [15] utilisant la méthodologie LiMM-PCA se trouvant dans le fichier `Sensory_Data_Naes_interaction.html` dans la partie LiMM-PCA venant du GitHub de M.Martin.

Toutes les sorties du chapitre se basent sur l'ensemble de données Candies. Cependant, les sorties des trois ensembles de données pour la partie de l'extension du package `limpca` peuvent être visualisées dans le fichier `test_limpca.html` du Github et dans le fichier en annexe du mémoire. Les sorties seront comparées pour les étapes où il y en a. Le premier objectif de cette section est de montrer que les graphiques et tables du package de base `limpca` et de l'extension du package `limpca` ont le même affichage (càd le même type de graphiques et de tables). Le second objectif est de montrer les différences et les similarités des sorties entre l'implémentation de M.Martin et l'extension du package `limpca`.

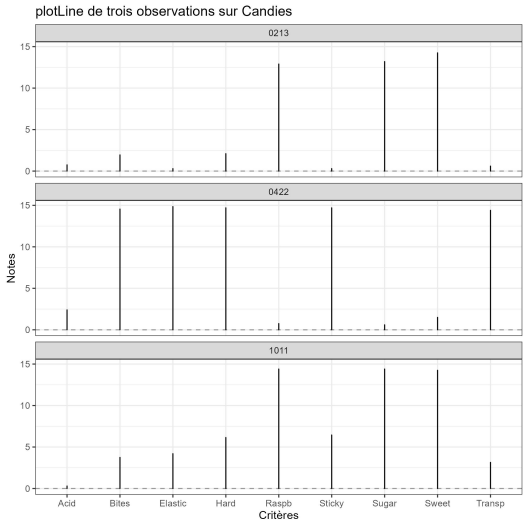
3.1 Sorties par étapes

3.1.1 Exploration des données

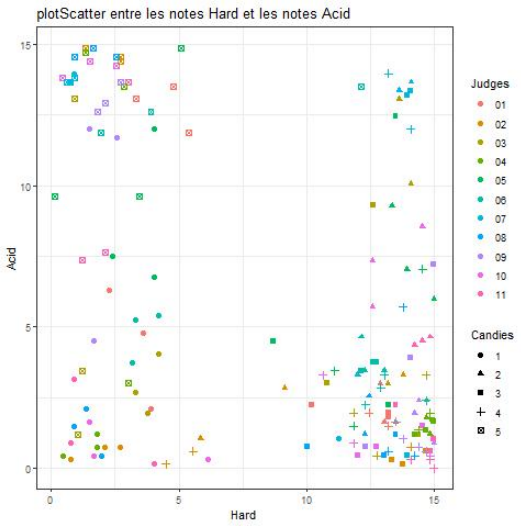
Pour les trois implémentations différentes dans l'exploration des données, les sorties seront les mêmes puisqu'il n'y a aucune transformation sur les données en dehors d'une

ACP sur la matrice de réponse $\mathbf{Y}$. Il est possible de visualiser le plan d'expérience de Candies sur la Figure 1.2.1 en utilisant la fonction `plotDesign`. Ci-dessous, la Figure 3.1.1 permet de visualiser les différentes réponses et observations de l'ensemble des données Candies en utilisant les fonctions du package `limpca` :

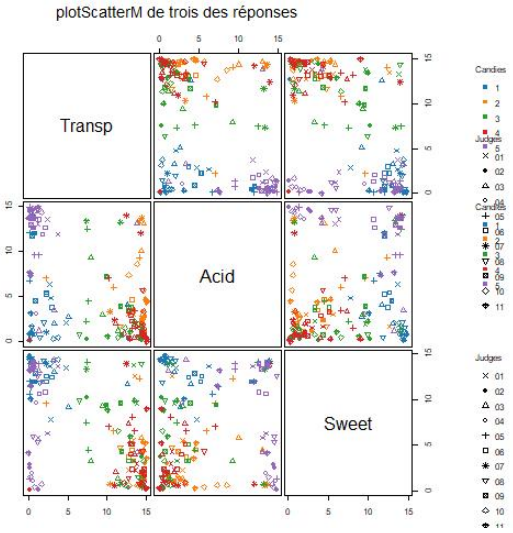
- la Figure 3.1.1.(a) utilise la fonction `plotLine` pour visualiser les valeurs des différentes réponses possibles pour les observations 0213, 0422 et 1011 de Candies. Pour rappel 0213 correspond à la $3^{\text{ième}}$ répétition de l'évaluation par le juge 2 du bonbon 1.
- la Figure 3.1.1.(b) utilise la fonction `plotScatter` pour visualiser toutes les observations pour deux réponses simultanées de Candies : "Acid" et "Hard". La figure permet aussi d'identifier, pour chaque observation, le juge qui note le bonbon par la couleur et le bonbon qui est noté par la forme.
- la Figure 3.1.1.(c) utilise la fonction `plotScatterM` pour visualiser toutes les observations pour trois réponses simultanées de Candies : "Transp", "Acid" et "Sweet". La figure permet aussi d'identifier, pour chaque observation, le juge qui note le bonbon par la forme et le bonbon qui est noté par la couleur.
- la Figure 3.1.1.(d) utilise la fonction `plotMeans` pour visualiser la moyenne des trois notes du critère "Hard" pour les observations faites par le même juge et avec le même bonbon. Les juges semblent globalement d'accord que les bonbons 1 et 5 sont mous et les bonbons 2,3 et 4 sont durs.



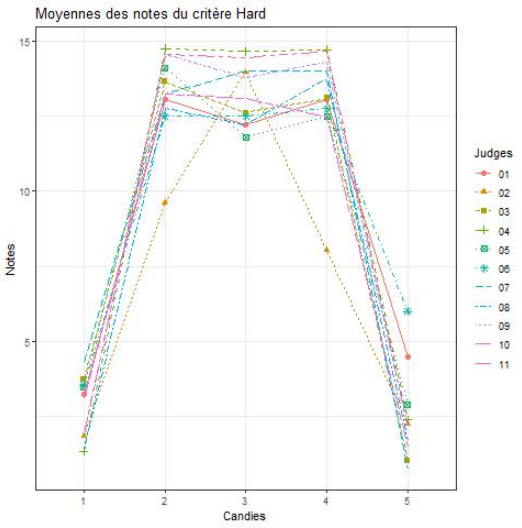
(a) `plotLine` de Candies



(b) `plotScatter` de Candies



(c) `plotScatterM` de Candies



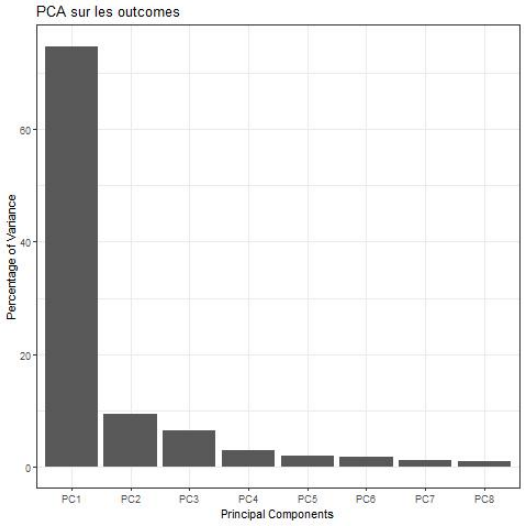
(d) `plotMeans` de Candies

FIGURE 3.1.1 – Visualisation des réponses et observations de Candies

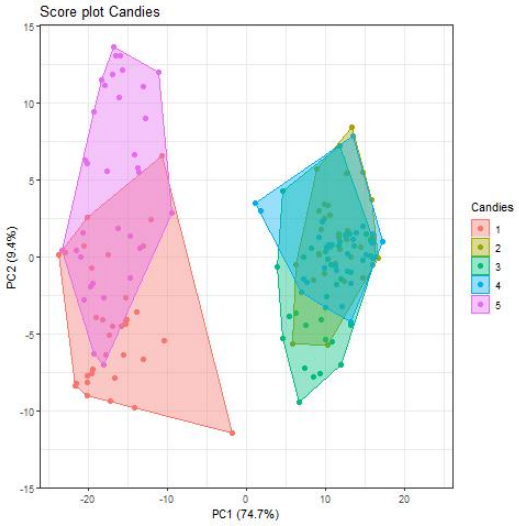
3.1.2 Réduction de dimensions par ACP

La réduction de dimensions ACP sur les réponses $\mathbf{Y}$ est la même pour les trois méthodologies s'il est décidé de prendre 8 PC, ce qui fait un pourcentage de variance expliqué cumulée de 99%. Les résultats de l'ACP `resPCA` pour l'ensemble de données Candies sont affichés sur les graphiques de la Figure 3.1.2 avec les fonctions du package `limpca` :

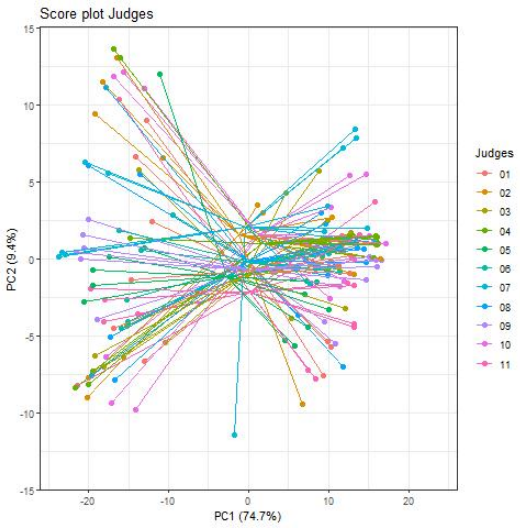
- la Figure 3.1.2.(a) est un graphique en barres du pourcentage de variance expliqué par chaque PC fournie par la fonction `pcaScreePlot`.
- la Figure 3.1.2.(b) utilise la fonction `pcaScorePlot` pour visualiser les scores des observations des deux premières composantes principales et en mettant en évidence le facteur Candies avec des polygones de couleurs. Sur ce graphique, les bonbons 1 et 5 sont similaires ainsi que les bonbons 2, 3 et 4.
- la Figure 3.1.2.(c) utilise la fonction `pcaScorePlot` pour visualiser les scores des observations des deux premières composantes principales tout en mettant en évidence le facteur Judges avec des segments de couleurs. Sur ce graphique, aucun juge ne se différencie des autres.
- la Figure 3.1.2.(d) permet de visualiser les réponses de Candies pour les deux premières composantes principales avec la fonction `pcaLoading2dPlot`. Cela permet, par exemple, de voir avec le graphique 3.1.2.(b) que les bonbons 2, 3 et 4 ont tendance à avoir comme caractéristiques : une grande dureté, transparence et élasticité. Il faut mordre fort et le bonbon colle aux dents.



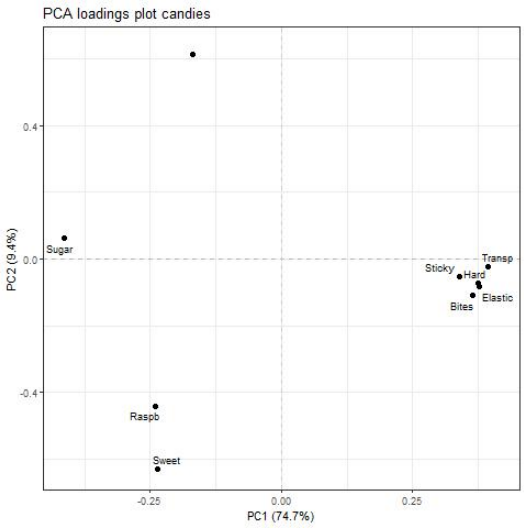
(a) `pcaScreePlot` sur `resPCA` de Candies



(b) `pcaScorePlot` sur `resPCA` de Candies avec le facteur Candies



(c) `pcaScorePlot` sur `resPCA` de Candies avec le facteur Judges



(d) `pcaLoading2dPlot` sur `resPCA` de Candies

FIGURE 3.1.2 – Visualisation des résultats `resPCA` sur l'ensemble Candies

3.1.3 Quantification et significativité de chaque effet

Les résultats de cette étape sont différents pour chaque méthodologie car le modèle a été estimé, à chaque fois, d'une autre manière. L'estimation du modèle via le package

`limpca` de base utilise un modèle linéaire général avec, comme formule :

$$outcomes \sim Candies + Judges + Candies : Judges \tag{3.1.1}$$

L'estimation du modèle via l'extension de `limpca` et le code M.Martin utilisent tous 2 la formule d'un modèle linéaire mixte :

$$\sim Candies + (1|Judges) + (1|Candies : Judges) \tag{3.1.2}$$

Le modèle utilisant la formule de l'Equation 3.1.2 est plus rigoureux que le modèle utilisant la formule de l'Equation 3.1.1 qui suppose le facteur Judges fixe plutôt que aléatoire.

La différence de l'estimation entre le modèle estimé par l'extension de `limpca` et le modèle estimé par le code de M.Martin est la fonction utilisée pour cette estimation. Le code de M.Martin utilise une fonction créée et nommée `parlmer_interaction` tandis que l'extension de `limpca` utilise la fonction `lmer` du package `lme4`. La différence entre `lmer` et `parlmer_interaction` vient de la construction de la matrice du modèle $\mathbf{Z}$ pour les effets aléatoires d'interaction. La fonction `parlmer_interaction` construit la matrice du modèle d'interaction aléatoire en faisant le produit matriciel de Kronecker en colonnes (nommé aussi Khatri-Rao) des matrices du modèle des effets principaux $\mathbf{X}_{\text{Candies}}$ et $\mathbf{Z}_{\text{Judges}}$, sachant que la matrice de l'effet fixe est codée via un contraste somme. La matrice $\mathbf{Z}_{\text{CJ}}$ est alors de taille $(n \times 44)$. La fonction `lmer` construit la matrice de modèle complète de taille $(n \times 55)$ avec comme codage 0/1 de base. La structure des deux matrices de modèle pour l'effet d'interaction aléatoire est expliquée plus en détails dans la section 4.2.5. Afin de s'assurer que les modèles estimés par `lmer` dans l'extension de `limpca` soient corrects, une validation a été effectuée en comparant les résultats avec ceux obtenus par la procédure `MIXED` en SAS.

Le test bootstrap est également différent entre le package `limpca` de base et les deux autres implémentations. Vous trouverez ci-dessous, pour l'ensemble Candies, les tables avec les pourcentages de variances expliqués et la p-valeur du test bootstrap pour 2000 simulations pour chaque approche et chaque effet du modèle.

La Table 3.1.1 est la table dans la sortie de la fonction `lmpBootstrapTests` du package `limpca`.

TABLE 3.1.1 – Pourcentage de variance expliqué et p-valeur du test bootstrap par effet pour l'ensemble de données Candies de `limpca` de base

Effet	**% de variance**	**P-valeur**
Candies	74.48	$<$ 5e-04
Judges	4.37	$<$ 5e-04
Candies :Judges	7.68	$<$ 5e-04
Residuals	13.47	-

La Table 3.1.2 est la table dans la sortie de la fonction `UPDATE_lmpBootstrapTests` de l'extension du package `limpca`.

TABLE 3.1.2 – Pourcentage de variance expliqué et p-valeur du test bootstrap par effet pour l'ensemble de données Candies de l'extension de `limpca`

Effet	**% de variance**	**P-valeur**
Candies	73.2	$<$ 5e-04
Judges	2.68	$<$ 5e-04
Candies :Judges	3.85	$<$ 5e-04
Residuals	19.63	-

La Table 3.1.3 est la table 5.3 dans la thèse de M.Martin (2020) [15].

TABLE 3.1.3 – Pourcentage de variance expliqué et p-valeur du test bootstrap par effet pour l'ensemble de données Candies de M.Martin

Effet	**% de variance**	**P-valeur**
Candies	74.77	$<$ 5e-04
Judges	3.49	$<$ 0.0355
Candies :Judges	2.08	$<$ 5e-04
Residuals	19.63	-

L'affichage des tableaux 3.1.1 et 3.1.2 est identique mais les résultats et surtout la manière de calculer le pourcentage de variance et la p-valeur de test bootstrap sont différents. Les explications de ces calculs sont dans la section 2.4.4 de la théorie. Les résultats des tableaux 3.1.2 et 3.1.3 ne sont pas totalement identiques puisque les estimations de modèles ne sont pas les mêmes mais, par contre, le pourcentage de variance expliqué et la p-valeur de chaque effet sont calculés de la même manière dans les deux implémentations. Notons néanmoins que les pourcentages de variance expliqués par les résidus sont identiques.

3.1.4 Les dimensions effectives (ED) et les facteurs de correction pour chaque effet

Cette section ne montrera pas les résultats de l'implémentation du package `limpca` puisque la matrice augmentée n'utilise pas le facteur de correction et que, dans cette section, l'objectif est de comparer les facteurs de correction.

Dans le cas de la méthode APCA et ASCA-E, la matrice augmentée peut être corrigée en tenant compte d'un facteur de correction de la formule 2.4.19 repris ici pour plus de lisibilité :

$$\tilde{\mathbf{M}}_g = \mathbf{M}_g + \sqrt{\frac{\mathrm{df}_g}{\mathrm{df}_{\mathrm{add}}} F_{(\mathrm{df}_g, \mathrm{df}_{\mathrm{add}}, 1-\alpha)}} * \mathbf{M}_{\mathrm{add}} \tag{3.1.3}$$

Le facteur de correction est le facteur multiplié par la matrice $\mathbf{M}_{\text{add}}$ se trouvant dans la formule 3.1.3. Pour obtenir ce facteur de correction, il faut calculer les degrés de liberté par l'approche des dimensions effectives (ED). Les résultats des ED pour les effets de Candies de l'extension du package `limpca` sont repris dans la table 3.1.4. Les résultats des ED pour les effets de Candies dans la Table 5.8 de la thèse de M.Martin (2020) [15] sont repris dans la Table 3.1.5. Le calcul des ED par effet sera expliqué dans la section 4.2.8 (première sous-section).

TABLE 3.1.4 – Les ED par effet pour l'ensemble de données Candies de l'extension du package `limpca`

Effet	**PC1**	**PC2**	**PC3**	**PC4**	**PC5**
Intercept	1	1	1	1	1
Candies	4	4	4	4	4
Judges	2.1	0.5	8.8	1.7	0.0
Candies :Judges	20.6	27.2	7.0	8.8	17.3
Residuals	138.3	133.3	145.2	150.5	143.7
Total = n	165	165	165	165	165

Effet	**PC6**	**PC7**	**PC8**	**ED max.**	**Max. dim.**
Intercept	1	1	1	1	1
Candies	4	4	4	4	4
Judges	6.5	4.9	0.0	8.8	11
Candies :Judges	0.0	25.0	7.5	27.2	55
Residuals	154.5	131.1	153.5	154.5	160
Total = n	165	165	165		

TABLE 3.1.5 – Les ED par effet pour l'ensemble de données Candies de M.Martin

Effet	PC1	PC2	PC3	PC4	PC5
Intercept	1	1	1	1	1
Candies	4	4	4	4	4
Judges	5.3	5.8,	9.1	3.4	2.5
Candies :Judges	13.3	18.9	9.8	8.6	12.4
Residuals	141.3	135.3	141.1	148	145.1
Total = n	165	165	165	165	165

Effet	PC6	PC7	PC8	ED max.	Max. dim.
Intercept	1	1	1	1	1
Candies	4	4	4	4	4
Judges	6.6	7.7	1.7	9.1	11
Candies :Judges	2.6	19.1	9.7	19.1	44
Residuals	150.8	133.2	148.6	150.8	160
Total = n	165	165	165		

Les Tables 3.1.5 et 3.1.4 ont des résultats différents pour le ED maximum pour l'effet Candies :Judges. Dans l'implémentation de M.Martin, la matrice du modèle pour cet effet comporte 44 colonnes, ce qui implique un degré de liberté de maximum 44. Dans l'implémentation de l'extension du package *limpca*, la matrice du modèle pour l'effet Candies :Judges comporte 55 colonnes, ce qui implique un degré de liberté de maximum 55.

Dans la Table 3.1.4, les ED des effets aléatoires peuvent être proches de 0. Elles ne sont pas égales à 0 car il a fallu ajouter une valeur par défaut très petite. Pour plus de détails, voir Section 4.2.8 (première sous-section) pour les modèles avec, en réalité, une variance estimée de l'effet aléatoire égale à 0 dans le modèle d'une réponse PC. Par exemple, pour le modèle de la PC5, la variance estimée pour l'effet aléatoire Judges est 0 et donc l'ED aura une valeur par défaut très petite. Ces valeurs très petites posent problème pour calculer les facteurs de corrections car, si le deuxième degré de liberté dans la distribution de ficher est très petit, alors la valeur F_{ED_1,ED_2} sera très grande. Donc, une solution apportée (qui n'est vraiment pas idéale) est de mettre une valeur minimum de ED. La valeur minimum des ED pour le deuxième degré de liberté de Fisher est fixée à 4 car, en dessous de ce seuil, les ED petits prennent trop de poids. Les ED plus petits que 4 sont donc remplacés par 4.

Maintenant que les ED sont calculés, il est aussi possible de calculer les facteurs de corrections pour chaque effet. Pour calculer le facteur de corrections, il faut connaître les degrés de liberté (ED) df_g et $\mathrm{df}_{\mathrm{add}}$ pour chaque effet Candies. La Table 3.1.6 indique dans sa dernière colonne comment chaque facteur de corrections doit être calculé pour chaque effet de Candies.

TABLE 3.1.6 – Description matrice augmentée pour le modèle Candies

Effet	**Type**	**ED associés**	**Matrice ajoutée**	**Facteur de correction**
Candies	F	$ED_C = a - 1$	$\hat{\mathbf{M}}_{CJ}$	$\sqrt{F_{ED_C,ED_{CJ}} \times \frac{ED_C}{ED_{CJ}}}$
Judges	A	ED_J	$\hat{\mathbf{E}}$	$\sqrt{F_{ED_J,ED_E} \times \frac{ED_J}{ED_E}}$
Candies :Judges	A	ED_{CJ}	$\hat{\mathbf{E}}$	$\sqrt{F_{ED_{CJ},ED_E} \times \frac{ED_{CJ}}{ED_E}}$
Résidus	A	ED_E	-	-

La Table 3.1.6 a, comme type, soit F (fixe), soit A (aléatoire), $a = 5$ est le nombre de niveaux pour le facteur Candies, $ED_E = n - (ED_C + ED_J + ED_{CJ})$ et F_{ED_1,ED_2} est le $95^{\text{ième}}$ quantile de la distribution de Fisher avec ED_1 et ED_2, les ED partiels.

La Table 3.1.6 permet de montrer que, dans le calcul de la correction des facteurs, les ED pour l'effet Candies seront divisés par les ED de l'effet Candies :Judges et les ED des autres effets seront divisés par les ED des résidus. Ce qui donne, comme estimation, des facteurs de correction pour les deux implémentations :

TABLE 3.1.7 – Les facteurs de corrections par effet pour l'ensemble de données Candies de l'extension du package `limpca`

Effet	**PC1**	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**	**PC7**	**PC8**
Candies	0.7	0.6	1.5	1.3	0.8	2.5	0.7	1.4
Judges	0.2	0.1	0.3	0.2	0.0	0.3	0.3	0.0
Candies :Judges	0.5	0.6	0.3	0.3	0.5	0.0	0.6	0.3

TABLE 3.1.8 – Les facteurs de corrections par effet pour l'ensemble de données Candies de M.Martin

Effet	**PC1**	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**	**PC7**	**PC8**
Candies	1.0	0.8	1.2	1.3	1.0	4.2	0.8	1.2
Judges	0.3	0.3	0.4	0.2	0.2	0.3	0.3	0.2
Candies :Judges	0.4	0.5	0.4	0.3	0.4	0.2	0.5	0.4

Dans la Table 3.1.7, les facteurs de corrections égaux à 0 sont les modèles qui ont estimé une variance pour l'effet aléatoire égale à 0. Par contre, la Table 3.1.8 n'a pas de 0 car il n'y a aucun modèle avec des estimations de variances pour les effets aléatoires égales à 0.

3.1.5 Contribution de chaque PC par effet

Il est possible de visualiser, sous forme de tableaux, les contributions PC par effets pour un modèle basé sur un ensemble de données via le paramètre `contribTable` de l'objet en

sortie de la fonction `UPDATE_lmpContributions` de l'extension du package `limpca`. Les contributions des PC pour les effets des Candies calculés sont les suivantes :

TABLE 3.1.9 – Les contributions des PC pour les effets des Candies

Effet	PC1	PC2	PC3	PC4	PC5	Contrib
Candies	69.58	3.76	0.43	0.07	0.00	73.84
Judges	2.40	0.18	0.08	0.01	0.01	2.68
Candies :Judges	2.22	0.87	0.38	0.21	0.08	3.85
Residuals	5.41	3.54	3.13	2.54	1.84	19.63

Dans l'objet en sortie de la fonction `UPDATE_Contribution`, il y a le paramètre `plotContrib` qui permet de visualiser sur un graphique en barres les contributions les plus importantes. La Figure 3.1.3 montre les contributions des PC les plus importantes sur la globalité des effets de Candies :

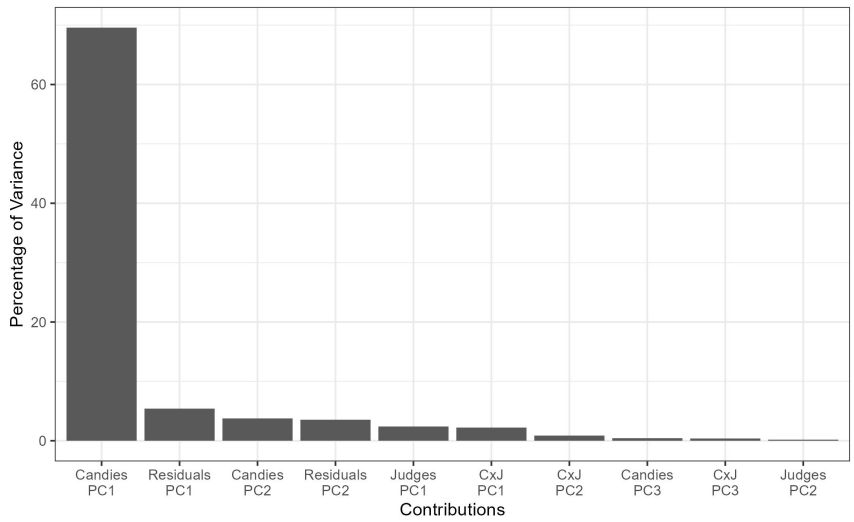


FIGURE 3.1.3 – Graphique des contributions des PC les plus importantes sur les effets de Candies

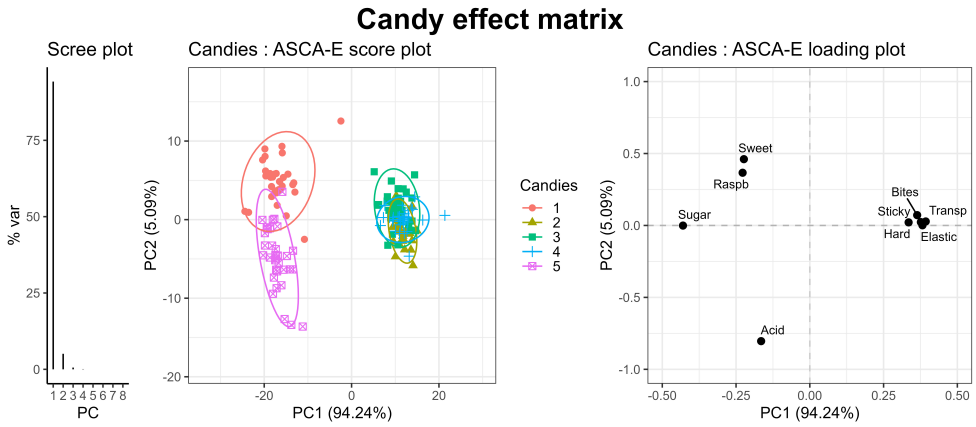
La Table 3.1.9 et la Figure 3.1.3 sont utiles pour visualiser quel PC de quel effet permet d'expliquer le mieux la variance des données. Dans notre cas, la majorité de la variance est expliquée par la PC1 de l'effet Candies et peu de variance est expliqué par l'effet Judge et Candies :Judges.

3.1.6 Visualisation des matrices d'effets

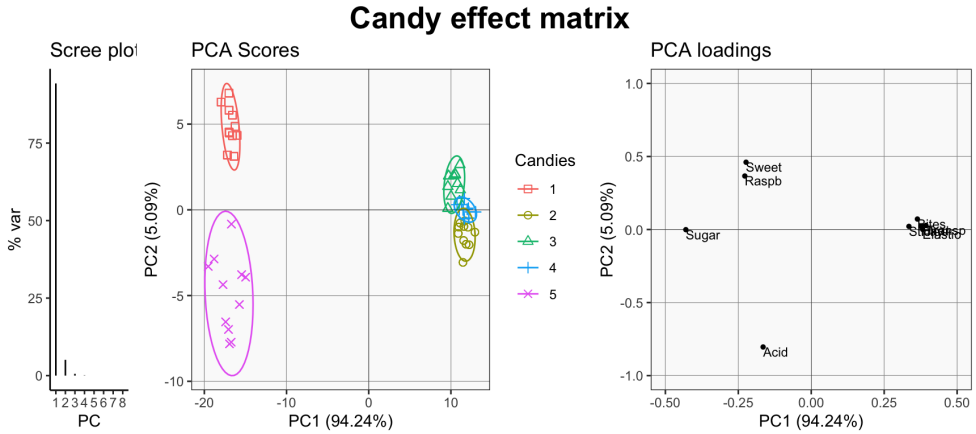
Dans cette étape, il est possible de visualiser les matrices d'effets avec l'ASCA, APCA et l'ASCA-E. Comme, le nombre de graphiques est important pour présenter chaque ACP multiplié par trois, les sorties pour l'ASCA-E seront affichées dans ce mémoire mais il est

possible de visualiser l'ASCA, l'APCA et l'ASCA-E sur l'ensemble de données Candies et sur les trois autres ensembles de données dans les sorties R en annexe. Pour les sorties ci-dessous, de l'extension du package `limpca` et de LiMM-PCA de M.Martin, les scores ASCA-E sont calculés en multipliant les *loadings* de l'ACP de la matrice d'effet par la matrice augmentée corrigée sur base de la formule 3.1.3. Par contre, les sorties de la fonction de base de `limpca` sont juste calculées en multipliant les *loadings* de l'ACP de la matrice d'effet par la matrice d'effet additionné avec la matrice des résidus.

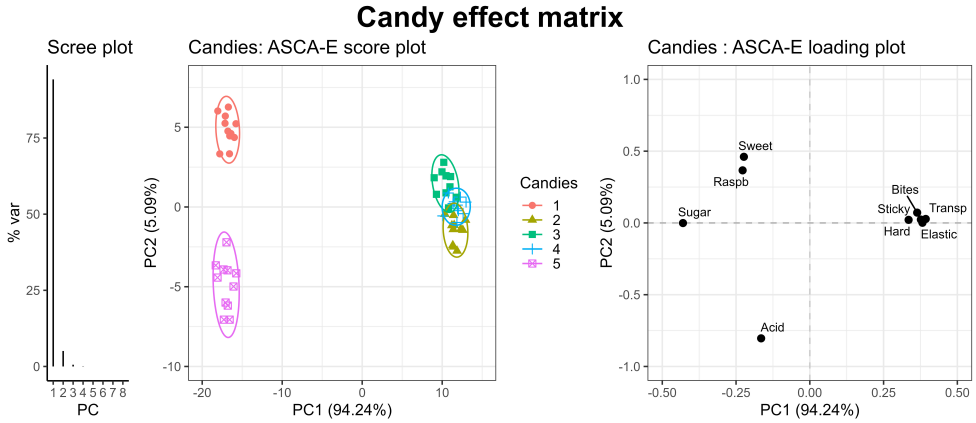
Pour chaque effet des données Candies, il est possible de visualiser les sorties. Ces sorties utilisent la méthode ASCA-E et sont : a) les sorties du package de base `limpca` calculées par la fonction `lmpPcaEffects`, b) les sorties de M.Martin Figure 4 dans Martin & Govaerts (2020) [2] et c) les sorties de l'extension du package `limpca` calculées par la fonction `UPDATE_lmpPcaEffects_3`. Il faut faire attention que les graphiques des scores des sorties a) n'ont pas la même échelle que les scores des sorties b) et c). Les sorties sont données ci-dessous :



(a) Package `limpca` initial



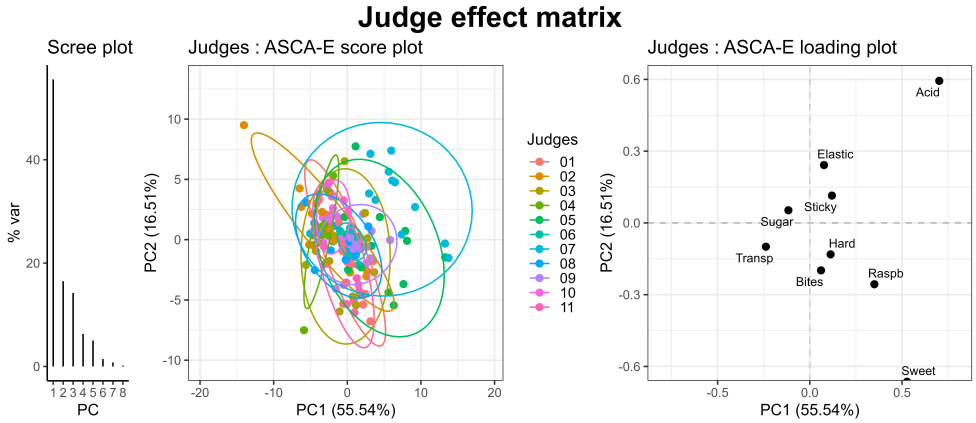
(b) LiMM-PCA de M.Martin



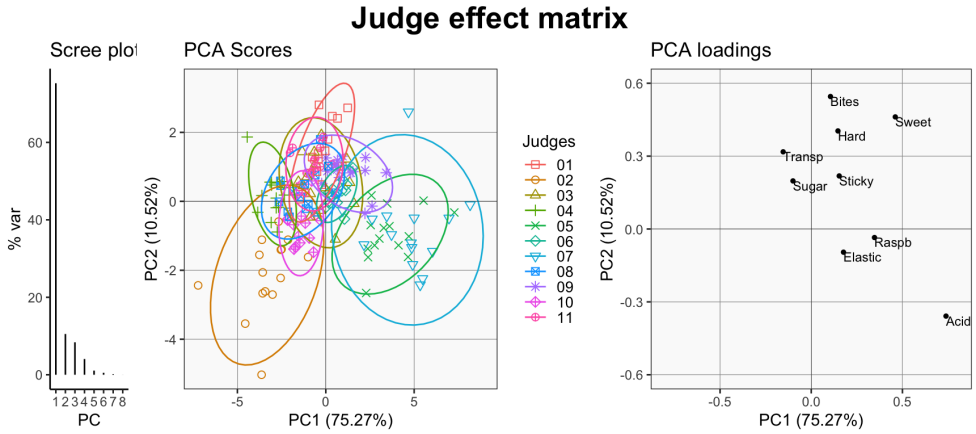
(c) Extension du package `limpca`

FIGURE 3.1.4 – Comparaison des sorties pour l'effet Candies

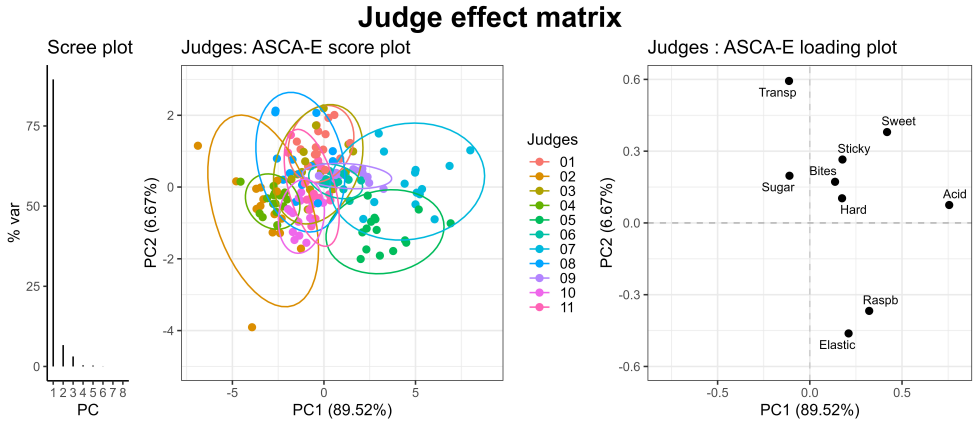
La Figure 3.1.4 permet de montrer que, dans le package `limpca` initial, il y a une grande différence de résultats pour les scores par rapport aux deux autres implémentations. La différence se justifie puisque la matrice d'effet Candies est augmentée avec la matrice des résidus alors que l'implémentation de M.Martin et de l'extension du package `limpca` utilisent la matrice d'effet d'interaction Candies :Judges pour l'augmentation de la matrice d'effet Candies. L'explication de cette augmentation sera discutée dans la section 4.2.8. Il faut aussi prendre en compte que le modèle est différent. Les scores de la matrice d'effet Candies de l'extension du package `limpca` et du code de M.Martin sont différents car le pourcentage de variance expliquée par l'effet Candies :Judges est plus grand chez M.Martin que dans l'extension de `limpca`. A l'inverse, le pourcentage de variance expliqué par l'effet Candies est moins grand chez M.Martin que dans l'extension de `limpca`. Cela a pour conséquence que les scores des observations par Candies sont moins dispersés dans l'extension du package `limpca`. Les graphiques des *loadings* sont assez semblables pour les trois implémentations.



(a) Package `limpca` initial



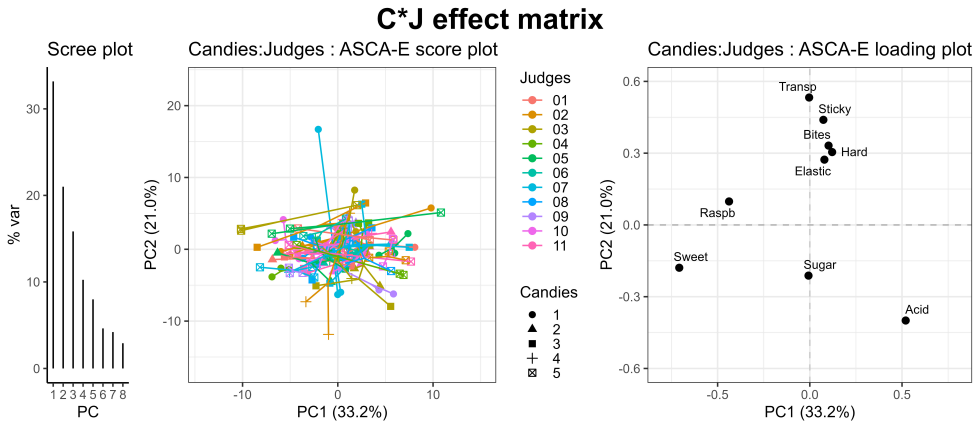
(b) LiMM-PCA de M.Martin



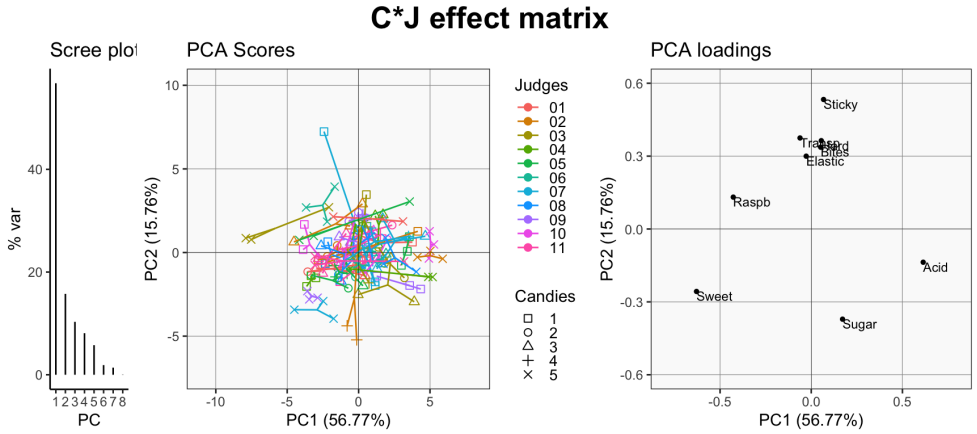
(c) Extension du package `limpca`

FIGURE 3.1.5 – Comparaison des sorties pour l'effet Judges

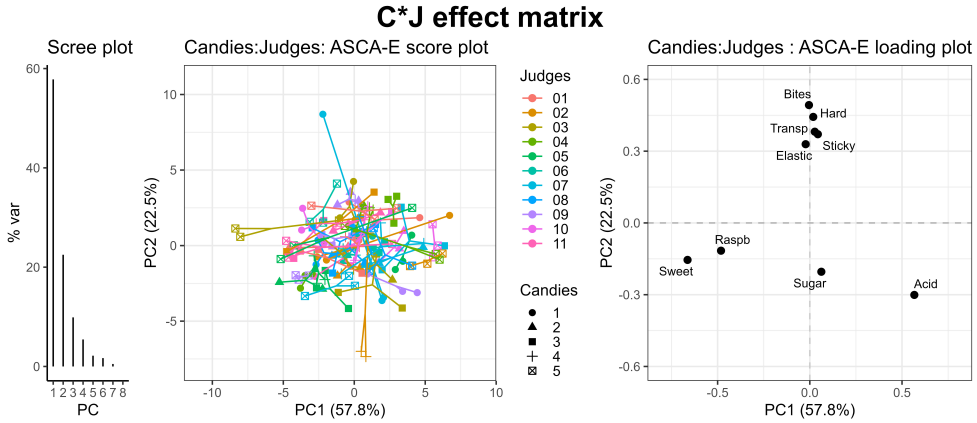
Pour l'effet Judges, La Figure 3.1.5 montre que les scores et les *loadings* pour les trois implémentations sont différentes même si la matrice ajoutée à la matrice d'effet Judges est la matrice des résidus.



(a) Package `limpca` initial



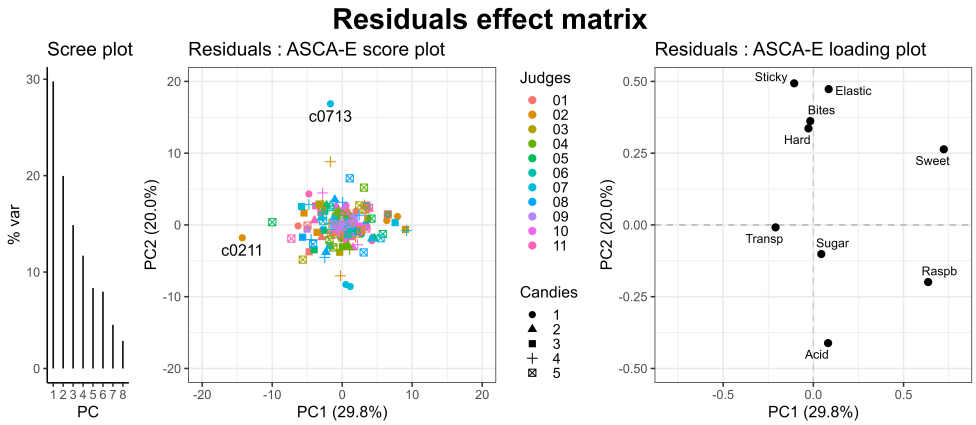
(b) LiMM-PCA de M.Martin



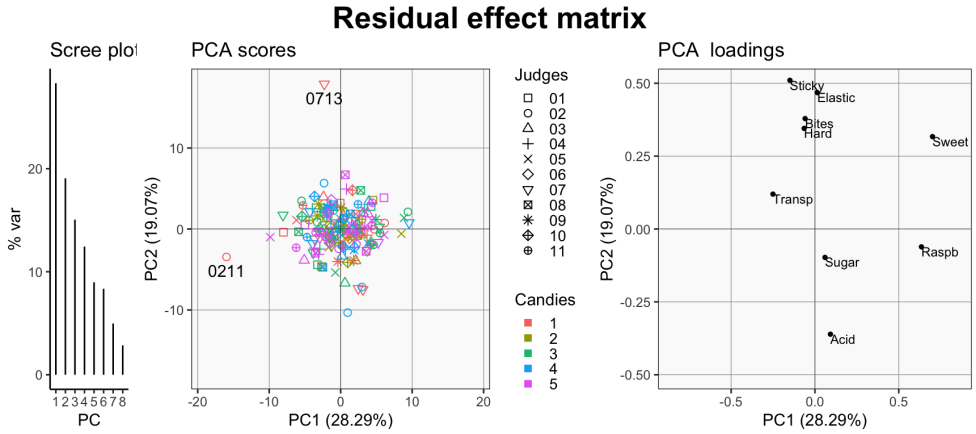
(c) Extension du package `limpca`

FIGURE 3.1.6 – Comparaison des sorties pour l'effet Candies :Judges

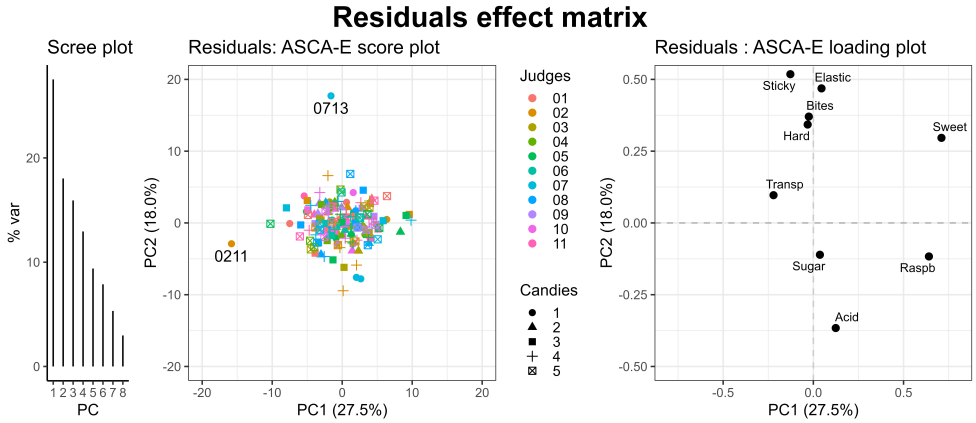
La Figure 3.1.6 montre que les scores et les *loadings* pour les trois implémentations sont différentes. Pour les trois implémentations, la matrice ajoutée à la matrice d'effet Candies :Judges est la matrice des résidus.



(a) Package `limpca` initial



(b) LiMM-PCA de M.Martin



(c) Extension du package `limpca`

FIGURE 3.1.7 – Comparaison des sorties pour l'effet des résidus

La Figure 3.1.7 montre que les scores et les *loadings* pour les trois implémentations ne sont pas totalement identiques.

Il est possible de visualiser des combinaisons d'effets dans l'extension du package `limpca`. Que les effets soient aléatoires ou fixes ne change rien mais, par contre, la matrice augmentée doit être calculée sans correction, c'est-à-dire en ajoutant, à la matrice d'effet des effets combinés, la matrice des résidus. Un exemple est montré par la Figure 3.1.8 en utilisant la méthode ASCA sur les effets combinés Candies, Judges et (Candies :Judges).

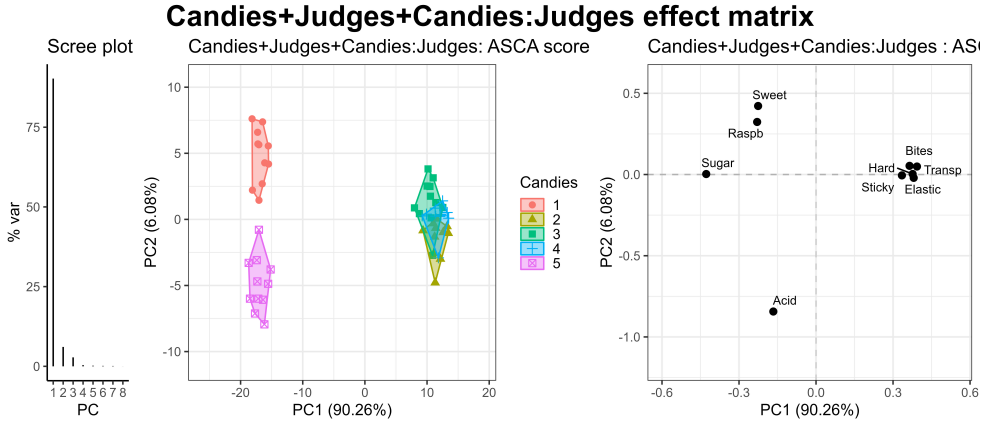
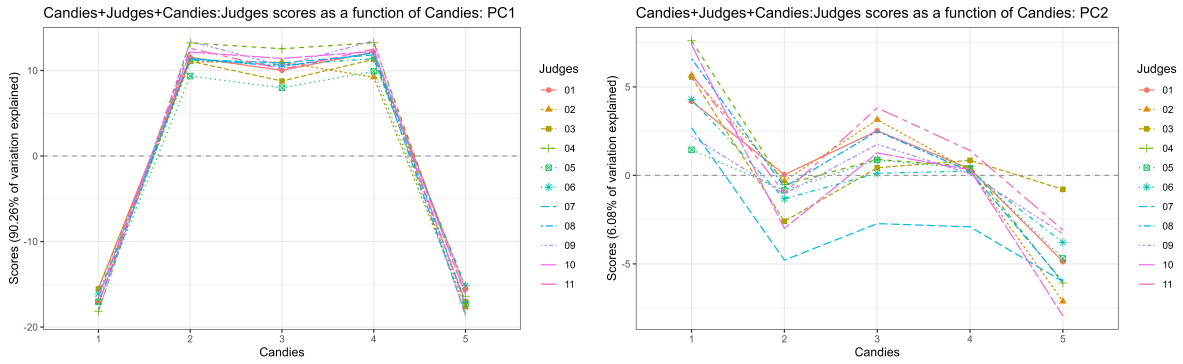


FIGURE 3.1.8 – Sorties avec la méthode ASCA pour la combinaison des effets Candies, Judges et (Candies :Judges) dans l'extension du package `limpca`

Le graphique des scores de la Figure 3.1.8 représente les scores des effets combinées Candies, Judges et (Candies :Judges). Il peut être comparé à la Figure 3.1.2.(b) représentant les scores totaux. On constate une différence entre les deux graphiques car toute la variation des données n'a pas été expliquée par ces trois effets et les résidus ne sont pas nuls.

Il est aussi possible de visualiser les scores d'une combinaison d'effets pour une ou plusieurs PC calculée par la méthode ASCA dans un graphique en lignes. Pour cela, il faut utiliser la fonction `UPDATE_lmpEffectPlot`. Un exemple est donné. Celui-ci représente les scores de la combinaison d'effets Candies, Judges et (Candies :Judges) pour la PC1 et la PC2 en mettant en évidence les facteurs Candies et Judges :



(a) Les scores pour la PC1

(b) Les scores pour la PC2

FIGURE 3.1.9 – Scores de la combinaison d'effets Candies, Judges et (Candies :Judges) en fonction des Judges et des Candies

La Figure 3.1.9 montre que, pour la PC1, tous les juges trouvent que les bonbons 2, 3 et 4 sont similaires et que les bonbons 1 et 5 sont similaires. Le graphique pour la PC2 montre que, malgré leur similarité, les bonbons ont quand même des différences pour les juges.

Chapitre 4

Implémentation de LiMM-PCA dans `limpca`

4.1 Introduction

Le Chapitre 2 a permis d'expliquer la structure du package `limpca` existant, la méthodologie sur laquelle l'implémentation de `limpca` est basée (ASCA+ et APCA+) et la méthodologie utilisée pour implémenter l'extension du package à l'analyse des modèles linéaires mixtes (LiMM-PCA). Le Chapitre 3 présentait les différentes sorties du package. Ce chapitre est consacré à la réalisation par étapes de l'implémentation de LiMM-PCA dans le package `limpca`. L'implémentation de l'extension du package a pour objectif final de fonctionner pour tous les modèles mixtes. Mais dans ce projet, l'extension permettra juste de gérer des modèles linéaires mixtes avec la même structure qu'un des trois ensembles de données expliqué dans la Section 1.2 tout en vérifiant que les modèles linéaires globaux fonctionnent encore. Les trois types de modèles linéaires mixtes gérés par le package `limpca` modifié sont :

- Un modèle linéaire mixte défini comme une ANOVA2 mixte avec interaction. Ce type de modèle comporte un effet principal fixe, un effet principal aléatoire et un effet aléatoire d'interaction entre le facteur de l'effet fixe et le facteur de l'effet aléatoire. Ce type de modèle sera testé avec l'ensemble des données Candies.
- Un modèle linéaire mixte défini comme une ANOVA2 mixte hiérarchique. Ce type de modèle comporte un effet aléatoire et un autre effet aléatoire hiérarchique entre le facteur du premier effet aléatoire de niveau supérieur et un autre facteur aléatoire de niveau inférieur. Ce type de modèle sera testé avec l'ensemble de données Serum.
- Un modèle linéaire mixte défini comme un modèle longitudinal mixte à deux facteurs. Ce type de modèle comporte deux effets fixes dont un effet *Temps*, un effet d'interaction fixe entre les deux facteurs principaux fixes et un effet aléatoire. Ce type de modèle sera testé avec l'ensemble de données CHOO.

La variation aléatoire se fait au niveau de l'intercept pour tous les effets aléatoires des trois types de modèles. L'utilisation de l'ensemble de données UCH permet de tester que les mo-

dèles linéaires globaux fonctionnent toujours. Cela ne veut pas dire que d'autres modèles linéaires mixtes ne pourraient pas fonctionner mais il faudrait alors tester et vérifier les résultats. Par exemple, la fonction `UPDATE_lmpModelMatrix` autorise, pour les trois types de modèle, autant d'effets fixes que l'utilisateur le souhaite. Dans ce cas, l'étape *Construction des matrices d'effet augmentées* 4.2.8 pose problème : les matrices $\mathbf{M}_g$ sont augmentées d'une matrice différente car dépendante de la structure du modèle mixte. Pour solutionné ce problème, lorsque la structure du modèle diffère à celui de l'un des trois ensembles, alors toutes les matrices sont augmentées de la matrice des résidus. A défaut, chacun des trois types de modèle gère l'augmentation des matrices de manière différente.

L'implémentation de l'extension se base fortement sur le code de M.Martin utilisé dans le Chapitre 5 de sa thèse [15] et l'article Martin & Govaerts (2020) [2]. Ce code applique les données LiMM-PCA sur les deux ensembles de données décrits dans l'introduction 1.2 : l'ensemble Candies et l'ensemble Serum. L'objectif est de modifier ce code pour en faire des fonctions générales qui ne dépendent ni d'un seul modèle, ni d'un ensemble de données. Les fonctions modifiées et les nouvelles fonctions sont dans le répertoire *R_new_update* du GitHub et les fonctions initiales sont dans le répertoire *R* du GitHub. Les fonctions modifiées ont comme nom "UPDATE_" suivi du nom des fonctions dans `limpca`. Il n'y a qu'une seule nouvelle fonction que les utilisateurs peuvent utiliser `lmpOutcomesReduct`. Quelques fonctions cachées qui sont inclus dans le package pour être utilisées dans une autre fonction sont aussi ajoutées. Le test de ces nouvelles fonctions sur les quatre ensemble de données, Candies, Serum, CHOO et UCH, est effectué dans le code RMarkDown *test_limpca.Rmd*.

Pour l'implémentation de l'extension, le choix est fait de séparer, dans les objets en sortie des fonctions, les paramètres en relation avec les effets aléatoires et les paramètres en relation avec les effets fixes. Les noms des paramètres des effets fixes garderont les mêmes noms que dans le package initial `limpca`. Par contre, les noms des paramètres aléatoires seront complétés d'un "R" final pour pouvoir les distinguer. Ce qui permet de mieux visualiser les effets qui sont fixes et les effets qui sont aléatoires.

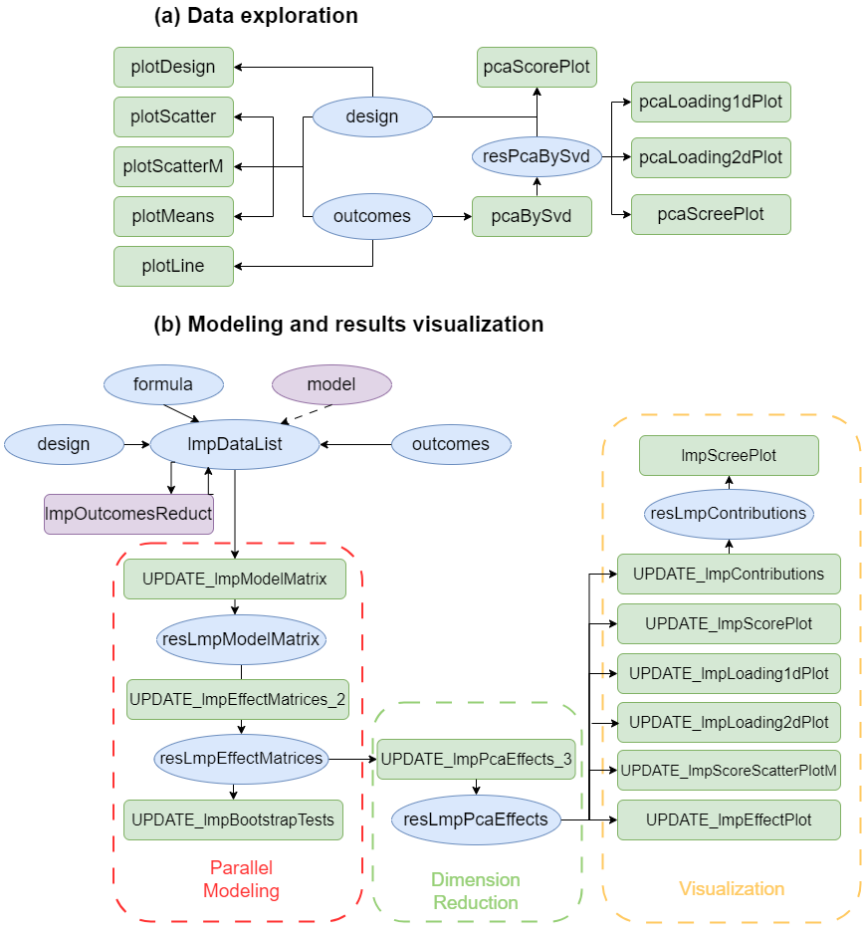


FIGURE 4.1.1 – Vue d'ensemble des fonctions et objets principaux de l'extension de `limpca`.

La Figure 4.1.1 présente la nouvelle structure du package et montre, en mauve un nouveau paramètre optionnel `model` dans l'objet `lmpDataList` et une nouvelle fonction `lmpOutcomesReduct` qui modifie l'objet `lmpDataList` expliqué plus en détails dans les étapes de l'implémentation ci-dessous. Les fonctions avec le préfixe "UPDATE" ont été modifiées durant l'extension.

4.2 Modification des étapes du package `limpca`

Cette section décrit, pour chaque étape du package `limpca`, les modifications et ajouts apportés à l'ensemble des fonctions et objets de l'étape. Les problèmes ou difficultés en rapport avec ces modifications et ajouts sont aussi expliqués si nécessaire.

4.2.1 Importation et exploration des données

Trois fichiers RData avec les ensembles de données ont été créés : `Candies.RData`, `Serum.RData` et `CHOO.RData`. Lorsque la fonction `data` est utilisée (exemple : `data("Candies")`), une liste avec le nom de l'ensemble est créée avec le même format que l'objet `lmpDataList`. Il contient donc le plan d'expérience `design`, la matrice de réponse `outcomes` et la formule `formula`.

L'objet `lmpDataList` peut contenir un quatrième paramètre dans l'extension du package `limpca`. Ce paramètre optionnel a pour nom `model` et comme valeur exclusivement "lmm" ou "lm". Le paramètre `model` détermine le type de modèle qui sera estimé dans `limpca`, soit un modèle linéaire mixte ("lmm"), soit un modèle linéaire à effets fixes ("lm"). Si ce paramètre n'existe pas dans le `lmpDataList` alors il sera déterminé automatiquement dans la fonction
`UPDATE_lmpModelMatrix` via une analyse de la structure de la formule.

4.2.2 Exploration des données

Il n'y a eu aucune modification ou ajout pour cette partie du package. Cette étape fonctionne de la même manière que dans le package `limpca` de base.

Les plans d'expérience peuvent toujours être affichés via la fonction `plotDesign` utilisée pour les sorties dans la Section 1.2 du RMarkDown : le plan de Candies 1.2.1, le plan de Serum 1.2.2 et le plan de CHOO 1.2.3.

Pour visualiser les réponses, il est toujours possible d'utiliser les fonctions `plotLine`, `plotScatter`, `plotScatterM` et `plotMeans`. Par exemple, les graphiques de la Figure 3.1.1 sont les sorties des quatre fonctions pour les données Candies.

Concernant la visualisation de la matrice de réponse réduite par l'ACP, il est encore possible d'utiliser la fonction `pcaBySvd` pour réduire $\mathbf{Y}$ et les fonctions `pcaScreePlot`, `pcaScorePlot`, `pcaLoading1dPlot` et `pcaLoading2dPlot`. Par contre, si le but est de réduire la matrice de réponse et d'utiliser les réponses réduites pour la modélisation, il faut utiliser la nouvelle fonction `lmpOutcomesReduct` expliquée dans la prochaine section.

4.2.3 Réduction de dimension par ACP

Cette étape est obligatoire pour les LMM et optionnelle pour les LM. Elle est néanmoins fortement recommandée, surtout s'il y a beaucoup de colonnes dans l'ensemble des données. La fonction `lmpOutcomesReduct` permet de réduire la matrice des réponses `outcomes` en utilisant la fonction `pcaBySvd` et de modifier l'objet `lmpDataList` pour garder toutes les informations essentielles pour la suite de l'analyse. La réduction permet aussi d'obtenir des réponses de la matrice réduite $\mathbf{Y}^*$ qui sont orthogonales entre elles et donc se rapproche

de l'indépendance. Elle permet également d'utiliser les tests de bootstrap de LiMM-PCA.

La fonction `lmpOutcomesReduct` prend en entrée l'objet `lmpDataList` qui ne comprend que les paramètres suivants : le `design`, les `outcomes` $\mathbf{Y}$, la `formula` et optionnellement le `model`. Le deuxième paramètre en entrée est le nombre de composantes principales, nommé `nPC`, que l'utilisateur voudrait garder pour sa nouvelle matrice de réponse $\mathbf{Y}^*$. Une liste de trois objets est en sortie de la fonction : `lmpDataList`(modifié), `resPCA` et `nPC`, où l'objet `resPCA` est la liste reprenant les résultats de ACP, `lmpDataList` est l'objet en entrée `lmpDataList` mais dont les *outcomes* sont modifiés par la fonction.

`lmpOutcomesReduct` vérifie si les `outcomes` sont bien une matrice et que le paramètre `nPC` est bien un seul nombre positif qui ne peut dépasser le nombre de colonnes de la matrice des `outcomes`. Si `nPC` est défini par l'utilisateur, une ACP est effectuée via la fonction `pcaBySvd` en précisant le paramètre `nPC`. Par contre, un `nPC` non défini implique que la fonction `pcaBySvd` calcule toutes les CPs puis l'objet `resPCA` est modifié en supprimant toutes les CPs où la variance expliquée dépasse 99%. L'objet `lmpDataList` est modifié avec, comme outcomes, les scores de l'ACP et l'ajout d'un booléen `isReduct` égal à TRUE, l'ancienne matrice `outcomes` appelée `outcomesRaw` et, enfin, les *loadings* de l'ACP nommés `loadingsPCA` qui permettront la rétro-transformation dans la fonction `UPDATE_lmpPcaEffects`.

Les résultats contenus dans l'objet `resPCA` ont la même structure que si l'utilisateur utilise directement la fonction `pcaBySvd` dans l'étape de l'exploration de données et donc permet d'utiliser les mêmes fonctions pour visualiser les résultats : `pcaScreePlot`, `pcaScorePlot`, `pcaLoading1dPlot` et `pcaLoading2dPlot`. Comme les sorties sont identiques pour l'ensemble Candies de la fonction `pcaBySvd(Candies, nPC = 8)` et de l'objet `resPCA` de `lmpOutcomesReduct`, la visualisation des résultats n'est montrée qu'une seule fois pour l'ensemble Candies sur la Figure 3.1.2.

4.2.4 Construction des matrices de modèle

Cette étape se base sur la fonction `UPDATE_lmpModelMatrix` qui permet de construire la matrice des effets fixes complète $\mathbf{X}$, la liste des matrices du modèle par effet fixe $\mathbf{X}_f$ pour $f = 0, ..., F$, la matrice des effets aléatoires complète $\mathbf{Z}$ et la liste des matrices du modèle par effet aléatoire $\mathbf{Z}_r$ pour $r = 1, ..., R$. L'objet `lmpDataList` modifié ou non par la fonction `lmpOutcomesReduct` est mis comme paramètre de la fonction `UPDATE_lmpModelMatrix`. Elle permet aussi d'identifier les noms des effets aléatoires et fixes. Si la formule du modèle `formula` ne contient pas d'effet aléatoire de type (1|...) et donc l'élément `model` de l'objet `lmpDataList` définis dans la Section 4.2.1 est égale à "lm", alors la fonction `UPDATE_lmpModelMatrix` a les mêmes résultats que dans `limpca` non modifié. Par contre, si `formula` comporte au moins un effet aléatoire et est une formule avec une structure valide, en plus des paramètres des effets fixes, les matrices de modèle des effets aléatoires sont alors ajoutés à la liste `resLmpModelMatrix` sortie par la fonction.

Différence entre les fonctions `lmer` et `parlmer_interaction`

Le code de M.Martin n'utilise pas directement la fonction `lmer` du package `lme4` (documentation de `lme4` dans Bates et al. (2022) [16]), mais une fonction `parlmer_interaction` créée par Mme Martin, laquelle utilise aussi les quatre modules présents dans `lmer` [1] :

1. le module de la formule `lFormula` : permet de "parser" et convertir la formule en une liste d'objets qui sera utile pour l'ajustement d'un modèle linéaire mixte et donnée en paramètre pour la deuxième fonction.
2. le module de la fonction objective `mkLmerDevfun` : retourne une fonction qui calcule la fonction de vraisemblance restreints avec, comme paramètre de covariances, θ.
3. le module d'optimisation `optimizeLmer` : utilise la fonction retournée par `mkLmerDevfun` pour optimiser la fonction de vraisemblance restreinte.
4. le module output `mkMerMod` : permet de mettre dans un objet utilisable les résultats du maximum de vraisemblance restreint optimisé.

Une des différences entre la fonction `parlmer_interaction` et la fonction `lmer` est la modification de la matrice du modèle pour les effets d'interaction aléatoires. Les deux fonctions ont les mêmes matrices de modèle pour les effets aléatoires principaux et les effets fixes mais pas pour les effets d'interactions aléatoires. Pour la fonction `lmer`, le nombre de colonnes de $\mathbf{Z}_r$ avec l'effet d'interaction aléatoire r est égale à $(l_{var1} \times l_{var2})$ où l_{var1} est le nombre de niveaux pour le premier facteur de l'interaction $var1$ fixe et l_{var2} est le nombre de niveaux pour le deuxième facteur de l'interaction $var2$ aléatoire. Elle permet d'avoir une valeur 1 lorsque l'observation est dans la colonne où l'interaction comporte le bon juge et le bon candies, 0 sinon. Alors qu'avec `parlmer_interaction`, le nombre de colonnes de $\mathbf{Z}_r$ est égale à $((l_{var1} - 1) \times l_{var2})$, laquelle permet d'avoir une matrice avec des 1 pour les observations dans la colonne avec le bon juge et le bon candies, -1 si l'observation est dans la colonne qui contient le dernier niveau de l'effet fixe et le bon juge, 0 sinon. Dans ce mémoire, l'extension du package repose sur la fonction `lmer` car celle-ci utilise la même manière de calculer que dans SAS puisqu'en exécutant la procédure `MIXED` en SAS sur les données Candies pour la réponse par exemple `PC1` (mais toutes les réponses auront la même matrice de modèle $\mathbf{Z}$), la structure de la matrice du modèle $\mathbf{Z}$ est la même :

1. Pour plus de détails sur les quatre fonctions et la fonction lmer, voir l'article Bates et al.(2015) [17].

```
Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
  ..@ i       : int [1:330] 0 55 0 55 0 55 9 64 9 64 ...
  ..@ p       : int [1:166] 0 2 4 6 8 10 12 14 16 18 ...
  ..@ Dim     : int [1:2] 66 165
  ..@ Dimnames:List of 2
  .. ..$ : chr [1:66] "1:01" "1:02" "1:03" "1:04" ...
  .. ..$ : chr [1:165] "0111" "0112" "0113" "1011" ...
  ..@ x       : num [1:330] 1 1 1 1 1 1 1 1 1 1 ...
  ..@ factors : list()
```

(a) La structure de la matrice **Z** dans le résultat de `lmer`

Dimensions	
Covariance Parameters	3
Columns in X	6
Columns in Z	66
Subjects	1
Max Obs per Subject	165

(b) La structure de la matrice **Z** dans le résultat de la procédure `MIXED` en SAS

FIGURE 4.2.1 – Comparaison des structures des matrices **Z** en R (lmer) et SAS pour la réponse PC1 de Candies

La matrice du modèle aléatoire complet **Z** a bien 66 colonnes dans SAS comme dans les résultats `lmer`. Il y a 11 colonnes pour les 11 juges et 55 colonnes pour l'effet d'interaction entre les juges et les bonbons. Comme il y a 5 bonbons et 11 juges, il y a bien 55 colonnes pour la matrice de modèle d'interaction et non 44 comme dans les résultats de la fonction `parlmer_interaction` car (5 - 1) x 11 = 44 :

```
$ RanModMatlist  :List of 2
 ..$ 1 | CandiesJudges: num [1:165, 1:44] 1 1 1 0 0 0 0 0 0 0 ...
 .. ..- attr(*, "dimnames")=List of 2
 ..$ 1 | Judges       : num [1:165, 1:11] 1 1 1 0 0 0 0 0 0 0 ...
 .. ..- attr(*, "dimnames")=List of 2
```

FIGURE 4.2.2 – La structure de la matrice **Z** pour la réponse PC1 de Candies dans l'objet `res.parlmer` étant le résultat de la fonction `parlmer_interaction`.

Explication de l'implémentation de la fonction `UPDATE_lmpModelMatrix`

Normalement, les matrices de modèles sont directement calculées par la fonction `lmer` et il n'est pas possible de créer **Z** sans estimer le modèle comme dans `lm`. Mais, comme pour cette étape, il est inutile de calculer déjà les estimations du modèle. Le choix est de

construire la matrice du modèle aléatoire manuellement **Z**. Un message d'avertissement est affiché pour informer l'utilisateur de ceci :

```
Avis : The random model matrix is provided for indicative purposes only.
```

La fonction `lmpModelMatrix` est modifiée pour vérifier via la formule si le modèle est un LMM ou un LM. Si l'utilisateur a défini le type de modèle avec le paramètre `model` dans l'objet `lmpDataList` par "lmm" (LMM) ou "lm" (LM), il n'y a pas besoin de déterminer le type de modèle. Cependant, si l'utilisateur encode le modèle en tant que LMM et qu'il n'y a pas d'effet aléatoire (il n'y a pas de caractère "|" dans la formule), un message d'erreur apparaît. Si l'utilisateur indique que le modèle est un LM et qu'il y a un effet aléatoire (il y a un caractère "|" dans la formule) alors un message d'erreur apparaît également. Si le paramètre `model` est de valeur NULL, il est décidé que, s'il y a au moins un caractère "|", il s'agit d'un modèle LMM et `model` = "lmm". Sinon, c'est un modèle LM et `model` = "lm". La fonction `checkArg` est modifiée en ajoutant dans la liste `check.list` le type de modèle `model` qui ne peut être que "lm" ou "lmm". Dans le cas d'un LMM, le booléen `isReduct` de l'objet `lmpDataList` doit être TRUE sinon un message d'erreur apparaît.

Dans l'extension de `limpca`, la fonction `UPDATE_lmpModelMatrix` gère la validité de la formule. La formule d'un LMM est valide si elle définit un des trois types de modèles linéaires mixtes suivant : une ANOVA2 mixte avec interaction, une ANOVA2 mixte hiérarchique, un modèle longitudinal mixte à deux facteurs. La fonction ne vérifie pas les effets fixes dans la formule mais bien les effets aléatoires. Tous les effets aléatoires doivent avoir un préfixe "1 |" dans la formule qui montre que la variation aléatoire ne se fait qu'au niveau de l'intercept. Il ne peut y avoir qu'un ou deux effets aléatoires dans le modèle. S'il n'y a qu'un seul effet aléatoire alors ce n'est pas un effet d'interaction. S'il y a deux effets aléatoires, un des effets aléatoires est un effet principal, l'autre est un effet d'interaction où celui d'interaction doit contenir l'effet principal.

La fonction `lmpModelMatrix` a été modifiée pour pouvoir construire le paramètre `modelMatrixR` contenant la matrice du modèle aléatoire **Z** et le paramètre `modelMatrixByEffectR` contenant la liste des matrices de modèle par effet aléatoire $\mathbf{Z}_r$. La matrice **Z** est formée en calculant la matrice pour chaque effet aléatoire $\mathbf{Z}_r$ puis ces matrices $\mathbf{Z}_r$ sont fusionnées. Pour calculer chaque matrice $\mathbf{Z}_r$, une formule avec seulement l'effet r et le plan des données est utilisée dans la fonction `model.matrix`. Pour ne pas prendre en compte la colonne d'intercept, il faut ajouter "0 +" à la formule.

Pour construire les noms complets des effets aléatoires de `effectsNamesAllR`, il faut prendre les noms des colonnes de **Z** et, pour chaque nom, supprimer tous les chiffres si ce n'est un exposant, et tous les niveaux des facteurs possibles du plan d'expérience.
A titre d'exemple : les niveaux du facteur volunteer dans CHOO ont le format "G.R." où les points sont des chiffres. Les noms des colonnes dans `modelMatrixR` de CHOO sont du format "volunteerG.R.". Pour avoir seulement "volunteer", il faut supprimer "G.R." du nom de la colonne. Les noms des effets aléatoires uniques `effectsNamesUniqueR` sont

déterminés en prenant les noms uniques dans le vecteur effectsNamesAllR.

Pour la création des objets liés à la partie fixe du modèle, le code de base est réutilisé. Ils s'agit de la matrice modelMatrix, de la liste modelMatrixByEffect, du vecteur effectsNamesAll et du vecteur effectsNamesUnique. Pour les créer la formule est simplement modifiée en ne prenant que les effets fixes.

L'objet resLmpModelMatrix est une liste contenant cinq éléments si le modèle est un LM : lmpDataList, modelMatrix, modelMatrixByEffect, effectsNamesAll et effectsNamesUnique. Par contre, l'objet resLmpModelMatrix est de taille neuf pour un LMM avec, comme paramètres supplémentaires, modelMatrixR, modelMatrixByEffectR, effectsNamesAllR et effectsNamesUniqueR

4.2.5 Construction des matrices d'effet

Nous avons eu le soucis d'améliorer les performance de UPDATE_lmpEffectMatrices en créant UPDATE_lmpEffectMatrices_2, ce qui permet de calculer correctement la variance des effets fixes même dans un plan non-balancé et sera expliqué plus en détail dans la Section 4.2.6. La fonction UPDATE_lmpEffectMatrices_2 permet de calculer les matrices des effets aléatoires et fixes et prend comme paramètre d'entrée, l'objet resLmpModelMatrix. La fonction sort, comme résultat, l'objet resLmpEffectMatrices qui comporte notamment, dans le cas d'un LMM :

— MM_full, le résultat de lmer ;
— EffectMatrices,les matrices d'effet fixe $\mathbf{M}_f$;
— EffectMatricesR, les matrices d'effet aléatoire $\mathbf{M}_r$.

Explication de l'implémentation de la fonction UPDATE_lmpEffectMatrices_2

La fonction UPDATE_lmpEffectMatrices_2 vérifie d'abord si le modèle est un modèle linéaire mixte. Si ce n'est pas le cas, elle réagit de la même façon que lmpEffectMatrices du package limpca initial. Si le modèle est un LMM, UPDATE_lmpEffectMatrices_2 vérifie que resLmpModelMatrix a le bon nombre et les bons noms des paramètres. Une liste fmla avec la formule pour chaque réponse est créée. Chaque élément de fmla est créé en concaténant le nom de la réponse avec la formule de base. Afin de créer la liste des contrastes pour définir le codage des facteurs utilisée dans lmer, une fonction appelée createContrastsSum a été développé avec, comme paramètre d'entrée, la formule du modèle. Celle-ci permet de dresser une liste avec, comme nom pour chaque élément, ceux des effets fixes de la formule qui ne sont pas des interactions et, comme valeur, "contr.sum". La fonction lmer estime un modèle pour chaque réponse de la matrice $\mathbf{Y}^*$ en parallèle. Si lmer envoie des messages ou des avertissements, UPDATE_lmpEffectMatrices_2 les capte et les transforme de sorte que, dans le message affiché, il y a la liste des réponses pour laquelle le message a été envoyé. Un exemple du message est donné début de la page 54 pour une erreur de singularité sur l'ensemble de données Candies. Si c'est un message d'avertissement, par exemple, le

modèle qui ne converge pas, alors un message supplémentaire informe que les estimations peuvent être biaisés :

```
Avis : The model estimates for these response(s) can be biased.
```

La fonction `UPDATE_lmpEffectMatrices_2` ignore les valeurs des paramètres de l'objet `resLmpModelMatrix` ; elle prend les résultats calculés par `lmer`. La matrice de modèle par effet fixe `modelMatrixByEffect` est élaborée grâce à `createModelMatrixFixByEffect`, laquelle est construite sur base d'un bout du code de la fonction `parlmer_interaction` de M.Martin. La généralisation de celle-ci est l'un des objectifs du présent mémoire. `createModelMatrixFixByEffect` crée un vecteur de signe avec un signe identique pour tous les niveaux qui appartiennent au même effet fixe f puis sépare la matrice complète du modèle fixe $\mathbf{X}$ en sélectionnant les colonnes ayant le même signe. L'intercept est aussi pris en compte. Les résultats des valeurs prédites `predictedvalues`, des coefficients des effets aléatoires `parametersR` ($\hat{\Gamma}_r$), des coefficients des effets fixes `parameters` ($\hat{\Theta}_f$), de l'écart type des résidus `resStdError` ($\hat{\sigma}_\epsilon$) et des variances estimées pour les effets aléatoires `variancesEstimatedR` ($\hat{\sigma}^2_r$) sont extraits de l'objet `MM_full` via des fonctions du package `lme4` et ajoutés à l'objet `resLmpEffectMatrices`. Les matrices d'effet fixe `effectMatrices` et d'effet aléatoire `effectMatricesR` sont calculées en multipliant, pour chaque effet fixe f / aléatoire r, la matrice du modèle $\mathbf{X}_f$ / $\mathbf{Z}_r$ avec la matrice des coefficients $\hat{\Theta}_f$ / $\hat{\Gamma}_r$.

Explication des messages d'avertissement affichés par `lmer`

L'utilisation de `lmer` dans `UPDATE_lmpEffectMatrices_2` permet d'estimer un très grand nombre de modèles linéaires mixtes. Mais, lors de son utilisation, deux messages d'avertissement ont été identifiés. En voici l'explication (et la résolution de l'un des deux) ci-dessous.

Dans la fonction `lmer`, il est possible d'utiliser l'optimisation BOBYQA (Bound Optimization BY Quadratic Approximation). Celle-ci a besoin du package `minqa`. L'optimisation BOBYQA est plus robuste et rapide mais prend plus de ressource que l'optimisation Nelder-Mead (méthode de simplexe) [2]. Le fait d'ajouter aux paramètres d'entrée de la fonction `lmer`, le paramètre `control = lmerControl(optimizer = "bobyqa")` permet d'optimiser avec la méthode BOBYQA plutôt que la méthode par défaut Nelder-Mead. Pour la PC8 dans l'ensemble des données Serum, le modèle n'arrivait pas à converger car il y avait une valeur propre négative. En conséquence, l'estimation des variances des effets aléatoires était biaisée et un message d'avertissement apparaît. Mais, en mettant le paramètre permettant d'utiliser la méthode BOBYQA, il n'y a plus de message d'avertissement ni de biais sur les estimations :

2. Voir l'article de Soritz et al. (2022) [18] et l'article de Bates et al.(2015) [17].

Covariance Parameter Estimates				
Cov Parm	Estimate	Alpha	Lower	Upper
Volunteer	0.04539	0.05	0.009277	3.21E153
Volunteer*Sampling	2.4279	0.05	1.4551	4.8465
Residual	0.3680	0.05	0.2854	0.4929

(a) Estimation avec SAS

```
Random effects:
 Groups             Name        Variance Std.Dev.
 Volunteer:Sampling (Intercept) 2.42793  1.5582
 Volunteer          (Intercept) 0.04539  0.2131
 Residual                       0.36804  0.6067
```

(b) Estimation avec l'optimisation BOBYQA en R

```
Random effects:
 Groups             Name        Variance  Std.Dev.
 Volunteer:Sampling (Intercept) 2.469e+00 1.571401
 Volunteer          (Intercept) 3.953e-05 0.006288
 Residual                       3.681e-01 0.606718
```

(c) Estimation avec l'optimisation Nelder-Mead en R

FIGURE 4.2.3 – Comparaison des variances estimées pour les effets aléatoires sur la PC8 des données Serum

Sur la Figure 4.2.3, la différence entre, d'une part la variance estimée pour l'effet *Volunteer* par SAS, d'autre part la même variance estimé par Nelder-Mead en R, est de 0,045 ; par contre, entre SAS et BOBYQA en R, il n'y a pas de différence. Concernant la variance estimée pour l'effet *Volunteer* : *Sampling*, il y a une différence de 0,041 entre SAS et Nelder-Mead en R ; il n'y a toujours pas de différence entre SAS et BOBYQA en R . De plus, le message d'avertissement suivant est affiché pour le résultat de la PC8 de Serum avec Nelder-Mead mais non avec BOBYQA en R :

```
Avis : unable to evaluate scaled gradient.
Avis : Model failed to converge: degenerate Hessian with 1 negative
eigenvalues
```

Il faut noter que tous les autres résultats d'estimations des modèles sont pratiquement les mêmes entre la procédure `MIXED` de SAS et la fonction `lmer` de R. On enregistre une différence maximale de 0,0002 entre les estimations.

Un autre message apparaît lorsque `lmer` est utilisé :

```
Avis : boundary (singular) fit: see help('isSingular')
```

Ce message explique que le modèle est singulier, cela veut dire que des valeurs estimées de certains paramètres de variances des effets aléatoires sont nulles. Par exemple, pour les données Candies, lorsque la fonction `UPDATE_lmpEffectMatrices_2` est exécutée, le message ci-dessous apparaît :

```
Avis : boundary (singular) fit: see help('isSingular')
 For the response(s) :  PC5, PC6, PC8
```

Ceci indique que, pour la PC5, PC6 et PC8, une des variances estimées pour les effets aléatoires est égale à zéro. Les résultats pour les trois réponses sont les suivants :

```
Random effects:
 Groups         Name        Std.Dev.
 Candies:Judges (Intercept) 0.9107
 Judges         (Intercept) 0.0000
 Residual                   2.1644
```

(a) Résultat de la réponse PC5

```
Random effects:
 Groups         Name        Std.Dev.
 Candies:Judges (Intercept) 0.0000
 Judges         (Intercept) 0.7585
 Residual                   2.1631
```

(b) Résultat de la réponse PC6

```
Random effects:
 Groups         Name        Std.Dev.
 Candies:Judges (Intercept) 0.4113
 Judges         (Intercept) 0.0000
 Residual                   1.6914
```

(c) Résultat de la réponse PC8

FIGURE 4.2.4 – Résultats des estimations des variances pour les effets aléatoires des trois modèles singuliers dans les données Candies.

La Figure 4.2.4 permet de constater que la variance estimée de l'effet *Judges* est zéro pour les réponses PC5 et PC8. La variance estimée pour l'effet *Candies* : *Judges* est zéro pour la réponse PC6. Pour les données Serum, la réponse PC13 estime un modèle singulier et, dans les données CHOO, les réponses PC8, PC10, PC11, PC12 et PC14 estiment des modèles singuliers. Ceux-ci poseront problème dans le calcul des dimensions effectives (ED) et dans l'augmentation des matrices d'effet $\mathbf{M}_g$. Voir l'étape 4.2.8.

4.2.6 Calcul du pourcentage de variance expliqué par effet

Cette étape est aussi inclus dans la fonction `UPDATE_lmpEffectMatrices_2`. Lorsque le modèle est un LM et que le paramètre `SS = TRUE` alors la fonction s'exécute de la même manière qu'avant l'extension et permet d'ajouter les trois paramètres à l'objet `resLmpEffectMatrices` : `type3SS`, `variationPercentages` et `varPercentagesPlot`. Par contre, si le modèle est un LMM, un élément sera ajouté dans l'objet (liste) `resLmpEffectMatrices` précité : `varComponentsAbs` qui représente toutes les variances

des effets du modèle (intercept, fixes, aléatoires et résidus).

La grande difficulté dans cette étape est de mettre sur la même échelle les variances des effets aléatoires et fixes. Les variances des effets aléatoires sont déjà estimées par `lmer`, par contre, il faut se poser la question pour les effets fixes. L'implémentation d'une première version de la fonction `UPDATE_lmpEffectMatrices` a été réalisée en se basant sur le code Candies de l'article Martin & Govaerts(2020) [2] avec, comme solution, la méthode venant de Nakagawa and Schielzeth (2012) [10]. Dans cette version, pour calculer la variance d'un effet fixe f, il faut seulement calculer la variance de la matrice d'effet $\mathbf{M}_f$. Mais cette variance était biaisée pour les données avec un plan non équilibré. En relisant plus attentivement le même article [2], on y trouve la formule pour calculer les variances des effets fixes. Il s'agit de la Formule 2.4.12. Ainsi une seconde version `UPDATE_lmpEffectMatrices_2` qui sera celle dans l'extension du package `limpca`, est implémentée en utilisant la Formule 2.4.12 pour calculer les variances des effets fixes.

Il est possible de voir une différence entre les résultats des deux fonctions dans les tables des pourcentages de variances expliqués par effet. `UPDATE_lmpEffectMatrices` a un seul paramètre en moins dans l'objet en sortie que `UPDATE_lmpEffectMatrices_2`. Il s'agit du paramètre `type3SS`. Deux tables sont affichées pour les données Candies et permettent de voir que les deux résultats sont identiques car le plan est balancé (voir Figure 4.2.5). Deux autres tables sont affichées pour les données CHOO et permettent de constater que les pourcentages de variances sont différents lorsque le plan n'est pas balancé (voir Figure 4.2.6) :

Candies	Candies:Judges	Judges	Residuals
73.837513	3.845329	2.683879	19.633279

(a) Résultat avec les variances des effets fixes estimées sur base des variances des matrices d'effet fixe

Candies	Candies:Judges	Judges	Residuals
73.837513	3.845329	2.683879	19.633279

(b) Résultat avec les variances des effets fixes estimées sur base des types III SS des effets fixes

FIGURE 4.2.5 – Résultats des pourcentages de variances expliqués pour les données Candies.

treatment	time	treatment:time	volunteer	Residuals
10.113500	0.749969	2.857907	29.334948	56.943676

(a) Résultat avec les variances des effets fixes estimées sur base des variances des matrices d'effet fixe

treatment	time	treatment:time	volunteer	Residuals
10.0958591	0.7481884	2.8495072	29.3444075	56.9620377

(b) Résultat avec les variances des effets fixes estimées sur base des types III SS des effets fixes

FIGURE 4.2.6 – Résultats des pourcentages de variances expliqués pour les données CHOO.

Si le modèle est un LMM, la fonction `UPDATE_lmpEffectMatrices_2` utilise la fonction `UPDATE_lmpSS` qui est pratiquement la même fonction que `lmpSS`. Elles calculent toutes les deux le type III SS pour chaque effet fixe `type3SS`. Il y a cependant une différence car, pour un effet LMM, `UPDATE_lmpSS` ne calculera pas le type III SS des résidus et le code n'utilisera pas le résultat des pourcentages de variances `var_percentage` car ils ne sont pas calculés correctement. Il faut bien noter que l'objet `type3SS` ne comporte que les effets fixes. Pour un modèle LMM, les résultats du paramètre `type3SS` de la fonction `UPDATE_lmpSS` vont être divisés par le nombre d'observations afin d'avoir les variances expliquées par les effets fixes. Les variances expliquées par les effets aléatoires sont la somme des variances estimées pour toutes les réponses de chaque effet aléatoire dans `variancesEstimatedR` ($\hat{\sigma}_r^2$) et le carré de l'écart type des résidus `resStdError` ($\hat{\sigma}_\epsilon^2$). Les variances des effets aléatoires et fixes sont stockées dans l'objet `varComponentsAbs`. La Formule 2.4.11 est utilisée pour calculer le pourcentage de variance estimé pour chaque effet sauf l'intercept et mis dans le paramètre `variationPercentages`. Dans la Formule 2.4.11, var($\mathbf{Y}^*$) est la somme de toutes les variances des effets sans l'intercept. Le graphique de la contribution des effets contenus dans le paramètre `varPercentagesPlot` est créé via la fonction interne `createVarPercentagesPlot`. `createVarPercentagesPlot` est une partie du code de la fonction `lmpEffectMatrices` qui est isolée dans une fonction pour pouvoir l'utiliser à plusieurs endroits dans la fonction `UPDATE_lmpEffectMatrices_2` mais, au final, elle n'a été utilisée qu'une seule fois pour le moment. Cette fonction permet de générer un graphique en barres avec, sur l'axe des x, les noms des effets et, sur l'axe des y, la valeur du pourcentage de variance expliqué pour chaque effet.

4.2.7 Tests Bootstrap

Cette étape permet de tester la significativité de chaque effet. Elle est implémentée dans la fonction `UPDATE_lmpBootstrapTests`. Celle-ci utilise, comme paramètre d'entrée, l'objet `resLmpEffectMatrices` et, comme objet de sortie, `resLmpBootstrapTests`. Ce dernier contient :

- `f.obs`, le test statistique observé pour chaque effet ;
- `f.boot`, le test statistique de chaque simulation pour chaque effet ;
- `p-values`, la p-valeur pour chaque effet ;
- `resultsTable`, un tableau avec les variances expliquées et les p-valeurs .

Cette fonction n'utilise pas le même test Bootstrap selon qu'il s'agit d'un LMM ou d'un LM. Pour un LM, le test Bootstrap utilise la méthodologie implémentée dans la fonction `lmpBootstrapTests` du package `limpca` de base comme expliqué dans la section 2.2.1. Pour un LMM, le test Bootstrap utilise la méthodologie expliqué dans LiMM-PCA à la section 2.4.4. Le test Bootstrap d'un LMM se base sur les rapports de vraisemblance entre le modèle avec et sans l'effet, alors que le test Bootstrap d'un LM se base sur un test statistique qui prend en compte les types III SS. Le test Bootstrap du LM offre l'avantage de ne pas devoir estimer à nouveau des modèles sans les effets car l'estimation de modèles prend plus de temps que des calculs matriciels. Dans le cas d'un modèle LMM, les effets aléatoires entrent en compte et il n'existait pas de formule pour estimer un modèle réduit à partir d'un modèle complet. Le test Bootstrap d'un LMM prend beaucoup plus de temps que le test Bootstrap d'un LM. En utilisant la fonction `UPDATE_lmpBootstrapTests` avec 200 simulations par effet sur l'ensemble de données CHOO et UCH, le temps d'exécution est de plus de 3 minutes pour CHOO et moins d'une minute pour UCH, alors que le modèle UCH a sept effets et que le modèles CHOO n'en a que quatre.

Parallélisation des processus

La parallélisation des processus autorise le calcule simultanée des éléments d'une liste sur une fonction, ce qui permet une économie du temps d'exécution.
`UPDATE_lmpBootstrapTests` n'utilise pas les mêmes packages pour paralléliser les calculs que dans la fonction `lmpBootstrapTests` lorsque le modèle est un LMM.
`lmpBootstrapTests` utilise les packages `parallel` et `doParallel` qui permettent, via plusieurs processus, de faire des calculs en parallèle. Ces deux packages sont aussi maintenus dans la fonction `UPDATE_lmpBootstrapTests` pour les LM. Par contre, pour les LMM, les packages utilisés sont le package `future` et le package `doFuture`. Ils peuvent faire les mêmes actions que `parallel` et `doParallel`. Les packages ont été modifiés car, avec les packages existants, les messages et les avertissements n'étaient pas envoyés par les processus en parallèle et aucune solution n'avait été trouvée pour régler ce problème. Donc, il était impossible de récupérer les messages pour les afficher aux utilisateurs alors qu'avec `future` et `doFuture`, les différents messages s'affichent correctement. Pour les LMM, la fonction `lmer` affiche des messages durant l'exécution des processus parallèles, il fallait donc que les processus puissent aussi afficher les messages générés durant leur exécution. Si des messages d'avertissement sont générés, ils sont maintenant affichés à l'utilisateur avec le nombre total de fois où ils ont été générés durant l'exécution. Pour une amélioration future, il faudrait utiliser les packages `future` et `doFuture` également pour les LM.

Dans le code de `UPDATE_lmpBootstrapTests`, il existe des fonctions `source` avec un paramètre `local = TRUE`. Sans cela, les processus parallèles construits par les packages `parallel` et `doParallel` ne trouvent pas certaines fonctions. Ces lignes de code avec les fonctions `source` pourront être enlevées lorsque la fonction `UPDATE_lmpBootstrapTests` sera dans le package `limpca`.

Les packages énoncés ci-dessus sont des packages permettant d'instancier les processus parallèles et de les arrêter. Le package `plyr` permet de faire des calculs en parallèle et contient deux fonctions nommées `laply` et `llply`. Celles-ci sont utilisées dans le code de `UPDATE_lmpBootstrapTests`. Les fonctions `laply` et `llply` ont, comme paramètres d'entrée, un vecteur d'éléments `items` et une fonction `FUN(item)` où `item` est un élément de la liste `items`. Les deux fonctions permettent aux processus de calculer en parallèle les éléments de `items` avec la fonction `FUN(item)`. La différence entre les deux fonctions est le type d'objet renvoyé par la fonction. `laply` renvoie les résultats sous forme de tableau alors que `llply` les renvoie sous forme de liste.

Problèmes rencontrés durant l'implémentation

Deux des problèmes rencontrés durant l'implémentation de la fonction `UPDATE_lmpBootstrapTests` surviennent dans les données CHOO lorsqu'un des effets est enlevé du modèle.

Le premier problème survient lorsque l'effet *time* ou *treatment* est enlevé du modèle complet : pour les effets fixes, il ne reste que l'interaction *treatment* : *time* et l'effet conservé. Lorsque la fonction `lmer` est utilisée pour estimer le modèle sans l'effet *time* ou *treatment*, `lmer` considère que le modèle est hiérarchique car l'effet d'interaction englobe l'effet conservé. La matrice du modèle fixe $\mathbf{X}$ n'a donc plus la bonne structure. Pour régler ce problème, on crée une nouvelle formule en additionnant tous les noms des effets aléatoires avec, comme format, "(1| nom)" et tous les noms des colonnes des matrices `modelMatrixByEffect` sans les noms des colonnes de la matrice de l'effet retirée. Dans la nouvelle formule amendée, le nom des colonnes fixes sera comme format "nomEff.nomNiv" où "nomEff" est le nom de l'effet et "nomNiv" est le nom du niveau de l'effet. Il faut aussi comme valeur du paramètre `data` de la fonction `lmer`, la fusion :

- des colonnes des matrices fixes `modelMatrixByEffect` qui sont dans la nouvelle formule et avec transformation des noms dans le format "nomEff.nomNiv" ;
- de la matrice du plan `design` ;
- de la matrice des réponses $\mathbf{Y}^*$.

Pour implémenter cela, deux listes sont créées : une liste `namesAll` reprenant tous les noms des effets aléatoires et fixes sauf l'intercept et une liste `namesAllTransformed` avec tous les noms fixes et les noms pour les effets aléatoires transformés dans le format "(1 | nom)". Avec ces deux listes, une liste de formules `null_formulas` est créée reprenant, pour chaque nom des effets `namesAll`, la nouvelle formule sans l'effet créée comme expliqué ci-dessus. La liste `namesAllTransformed` est utilisée pour ajouter les noms des effets aléatoires dans les nouvelles formules. Une liste `null_form_data` est aussi créée pour chaque effet avec, comme valeur, la matrice qui sera donnée au paramètre `data` de la fonction `lmer`.

Le deuxième problème rencontré dans les données CHOO survient lorsque l'effet aléatoire *volunteer* est retiré. Il ne reste alors que des effets fixes et la fonction `lmer` ne gère

pas les modèles sans effet aléatoire. Il n'est pas possible d'estimer le modèle sans l'effet *volunteer* avec la fonction `lmer`. Le modèle sans l'effet est estimé dans le but de calculer sa vraisemblance pour ensuite évaluer le rapport de vraisemblance entre le modèle complet et le modèle sans l'effet. Mais ce rapport peut être trouvé d'une autre manière. Le problème peut être résolu en utilisant la fonction `rand` du package `lmerTest` sur l'objet `MM_full` qui est le résultat estimé par `lmer` sur le modèle complet dans la fonction `UPDATE_lmpEffectMatrices_2`. La fonction permet de calculer la vraisemblance du modèle et la vraisemblance du modèle sans un effet aléatoire. Comme il n'y a qu'un seul effet aléatoire (dans le modèle de CHOO, c'est *volunteer*), la fonction `rand` calcule bien les deux vraisemblances qui interviennent dans le rapport de vraisemblance.

Explication de l'implémentation de la fonction `UPDATE_lmpBootstrapTests` pour les LMM

Pour gérer les LMM donc `model = "lmm"`, la fonction `UPDATE_lmpBootstrapTests` vérifie :

- que l'objet `resLmpEffectMatrices` est valide en vérifiant qu'il a bien 22 arguments ;
- qu'elle a les mêmes noms et dans le même ordre que l'objet en sortie de la fonction `UPDATE_lmpEffectMatrices_2` ;
- qu'elle a le même nombre d'effets que dans la liste des matrices d'effets `effectMatrices`/`effectMatricesR` et des noms des effets `effectsNamesUnique`/`effectsNamesUniqueR`.

La fonction crée des processus parallèles au nombre de `nCores`. Elle crée aussi les objets `namesAll`, `namesAllTransformed`, `null_form_data` et `null_formulas`. Si la fonction n'a qu'un seul effet aléatoire, la formule dans `null_formulas` et la matrice dans `null_form_data` liée à l'effet aléatoire sont mises dans des objets temporaires `tmp_formula` et `tmp_form_data` afin de les conserver et sont supprimés des deux objets. Le nom de l'effet aléatoire est aussi retiré dans `namesAll`. Cela permet d'estimer tous les modèles avec, à chaque fois, un des effets en moins sauf l'effet aléatoire retiré des objets car le rapport de vraisemblance est calculé en utilisant la fonction `rand`.

Un vecteur `REML` est créé avec, comme noms des éléments, les noms des effets. Pour les effets fixes, il aura une valeur FALSE et, pour les effets aléatoires, une valeur TRUE. Pour chaque effet, les vraisemblances des modèles complets pour toutes les réponses `loglik_full` sont calculées en prenant les résultats du modèle `MM_full` et la valeur de `REML` liée à l'effet et en les mettant en paramètre de la fonction `logLik` de `lme4`. `REML` permet de calculer le maximum de vraisemblance restreinte (REML) pour les effets aléatoires et le maximum de vraisemblance (ML) pour les effets fixes. Pour chaque effet, les maximums de vraisemblance avec le modèle sans l'effet vont être calculés. La fonction `lmer` est exécutée pour chaque effet avec, comme paramètre, la formule dans `null_formulas` et la matrice dans `null_form_data` en lien avec l'effet. Une fois que tous les résultats `MM_null` de `lmer` pour chaque modèle ont été calculés, alors, pour chaque effet, le maximum de vraisemblance sans l'effet `loglik_null` est calculé avec la fonction `logLik` qui a comme paramètre : les

résultats MM_null et la valeur liée à l'effet retirée dans le vecteur REML. Enfin, pour chaque effet, le rapport de vraisemblance totale entre le modèle complet et le modèle sans l'effet peut être calculé en additionnant les rapports de vraisemblance pour chaque réponse et peut être mis dans l'objet f.obs. Pour l'effet où le rapport de vraisemblance est calculé par la fonction rand, tous ses résultats sont ajoutés dans les différents objets de la fonction.

Afin de calculer f.boot pour chaque effet, la fonction laply de plyr est utilisée avec le paramètre .parallel = TRUE pour que les processus travaillent en parallèle. laply permet d'appeler nboot fois la fonction bootstrapLT, nboot étant le nombre de simulations et de rapports de vraisemblance calculés pour chaque effet déterminé par l'utilisateur (par défaut 100).

La fonction bootstrapLT calcule un rapport de vraisemblance totale sumlog et les rapports de vraisemblance de chaque réponse ratio pour un effet g. La fonction prend en paramètres le booléen useREML pour l'effet g, le résultat de l'estimation du modèle sans l'effet g MM_null, la formule du modèle sans l'effet g null_formula, les données utiles pour l'estimation du modèle sans l'effet g null_form_data, la formule du modèle complet form_full et la matrice de réponses outcomes.

Si la formule sans l'effet g est un modèle linéaire général, alors bootstrapLT estime le modèle sans l'effet g en utilisant la fonction lm. Puis, avec cette estimation, cette dernière simule une matrice de réponses y et estime un nouveau modèle lmerand avec les nouvelles réponses y et la formule du modèle complet form_full. Elle calcule les maximums de vraisemblance du modèle complet et du modèle sans l'effet g pour chaque réponse avec la fonction rand sur base des résultats lmerand.

Si la formule sans l'effet g est un LMM, le résultat MM_null permet de simuler directement une matrice de réponses y. Avec la matrice de réponses y et la fonction lmer, le modèle complet MM_f_full et le modèle sans l'effet g MM_f_null sont estimés. Le maximum de vraisemblance du modèle complet est estimé grâce à MM_f_full, le booléen useREML et la fonction logLik. Même chose pour le maximum de vraisemblance du modèle sans l'effet g mais avec les résultats MM_f_null. Les rapports de vraisemblance par réponse ratio sont calculés ainsi que le rapport de vraisemblance total sumlog en faisant la somme des rapports de vraisemblance par réponse. La fonction bootstrapLT renvoie les résultats sumlog et ratio qui sont stockés dans les objets sumlog_boot et ratio_boot.

Lorsque toutes les simulations pour chaque effet ont été effectuées, la p-valeur p-value de la formule 2.4.16 peut être calculée avec les objets f.obs et sumlog_boot pour chaque effet. sumlog_boot est nommée f.boot. Le tableau resultsTable est construit de la même manière que dans lmpBootstrapTests. A la fin de l'exécution, les processus parallèles sont arrêtés.

4.2.8 ACP sur les matrices d'effet et construction des matrices augmentées

Dans cette étape, c'est la fonction `lmpPcaEffect` du package `limpca` qui va être modifiée pour gérer les modèles LMM.

La fonction `lmpPcaEffect` permet de faire une ACP sur les matrices d'effets pour la méthode ASCA. Pour les méthodes APCA et ASCA-E, cette fonction augmente les matrices d'effets $\hat{\mathbf{M}}_f$ par la matrice des résidus $\hat{\mathbf{E}}$. Mais, comme vu dans la section 2.4.5, les LMM n'augmentent pas obligatoirement toutes leurs matrices d'effets par la matrice des résidus ; elle peuvent par contre les augmenter par une matrice d'effets aléatoires. L'augmentation de la matrice d'effets corrigée utilise les dimensions effectives (ED) dans la formule 2.4.19 et, donc, il faut les calculer.

Trois versions de la fonction `lmpPcaEffect` ont été créées.
La première version `UPDATE_lmpPcaEffects` augmente juste tous les effets fixes et aléatoires par la matrice des résidus.
La deuxième version `UPDATE_lmpPcaEffects_2` calcule les ED et utilise la formule 2.4.19. Cette fonction permet seulement d'augmenter certaine matrices d'effets pour des effets précisés dans le code avec une matrice d'effets aléatoires autre que celle des résidus, elle aussi figurant dans le code. Ces effets précisés dans le code n'appartiennent exclusivement qu'aux trois modèles créés par les données Candies, Serum et CHOO.
La troisième version `UPDATE_lmpPcaEffects_3` est la plus développée et permet d'augmenter la matrice d'effet d'un effet g précis avec la matrice d'effet d'un effet aléatoire r précis. L'utilisateur peut décider, avec le paramètre `correctedMatrixAdd` mis à TRUE, d'augmenter les matrices d'effets avec la matrice d'effet ajoutée et multipliée par le coefficient illustré dans la formule 2.4.19 et non seulement par la matrice d'effet ajoutée. Les effets g et r doivent appartenir à un modèle avec une structure totalement identique (le même nombre d'effets aléatoires, d'effets fixes, d'interactions aléatoires et d'interactions fixes) qu'une ANOVA2 mixte avec interactions, qu'une ANOVA2 mixte hiérarchique ou qu'un modèle longitudinal mixte à deux facteurs (avec effet "Temps" ou "Time"). Si le modèle des effets g et r n'ont pas une des trois structures attendues, toutes les matrices d'effets de ce modèle sont augmentées par la matrice des résidus.

Avant d'expliquer la fonction `UPDATE_lmpPcaEffects_3`, il faut expliquer le calcul des ED et déterminer quelles matrices d'effets sont augmentées par celle des résidus et par quelle matrice d'effets aléatoires, ces matrices d'effets sont augmentées.

Calcul des dimensions effectives (ED)

La méthode de Eilers (2018) [13] permet de calculer les dimensions effectives (ED) pour chaque effet fixe et aléatoire du modèle. M.Martin l'utilise dans son code cette méthode. La fonction `computeED` est la fonction qui calcule tous les ED des effets pour chaque réponse

d'un modèle et sort un objet `ED` avec tous les ED. Cette fonction se base sur le code de M.Martin mais est modifiée pour être généralisée à n'importe quelle structure de LMM et prend comme paramètres `resLmpEffectMatrices`.

La fonction calcule pour chaque réponse les ED et les met dans l'objet `ED` qui est une matrice avec, comme, lignes tout les effets du modèle sans l'intercept et , comme, colonnes les réponses. Pour chaqu'une de celles-ci réponse, il faut calculer une matrice $\mathbf{K}$ puis extraire ses blocs diagonaux $\mathbf{K}_{ii}$ avec $i = 2, ..., I$ où I est le nombre d'effets aléatoires plus un. Puis calculer la trace de chaque matrice $\mathbf{K}_{ii}$ qui est égale à l'ED de l'effet aléatoire numéro $i - 1$. La trace de la matrice $\mathbf{K}_{11}$ correspond au ED des effets fixes mais n'est pas pris en compte dont le calcule est aisée. Si le modèle est un LM, la fonction `computeED` ne calcule pas la matrice $\mathbf{K}$ expliquée dans le prochain paragraphe car il n'y a pas d'ED aléatoire. Dans ce cas, elle calcule seulement les ED des effets fixes et les ED des résidus. Les ED d'un effet fixe f sont les mêmes pour toutes les réponses et sont calculés en prenant le nombre de colonnes de la matrice du modèle $\mathbf{X}_f$. Les ED des résidus pour chaque réponse est le nombre d'observations de la matrice de réponses $\mathbf{Y}^*$ moins la somme des ED des autres effets pour la même réponse.

Pour calculer la matrice $\mathbf{K}$ pour chaque réponse, les équations de l'article de Eilers (2018) [13] sont utilisées. Une première matrice à calculer dans la fonction `computeED` est données par l'équation (4) de l'article :

$$\mathbf{Q} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix}^{-1} \tag{4.2.1}$$

où la matrice diagonale $\mathbf{G}$ est construite avec les variances estimées des effets aléatoires σ_r^2 de la réponse (voir la formule 2.3.3), et la matrice diagonale $\mathbf{R}$ (notée $\mathbf{\Sigma}$ dans la section 2.3) construite avec les variances estimées des résidus σ_ϵ^2 de la réponse (voir la formule 2.3.4). L'équation (11) de l'article permet ensuite de calculer la matrice $\mathbf{K}$:

$$\begin{aligned} \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} &= \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} \end{bmatrix} \\ \mathbf{K} = \underbrace{\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix}^{-1}}_{\mathbf{Q}} \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} \end{bmatrix} \\ \mathbf{K} = \mathbf{Q} \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} \end{bmatrix} \end{aligned} \tag{4.2.2}$$

Les modèles singuliers posent problème pour le calcul des ED : il s'agit d'un calcul matriciel qui utilise les variances des effets aléatoires σ_r^2 sur la diagonale de la matrice $\mathbf{G}$. Si ne fut-ce qu'un seul (ou plusieur) σ_r^2 est nul, $\mathbf{G}$ n'est plus inversible puisque son déterminant est nul. Cependant, le calcul de $\mathbf{Q}$ requière celui de $\mathbf{G}^{-1}$. Dans ce mémoire, un modèle singulier est défini précisément comme un modèle où il y a au moins une variance σ_r^2 nul. Pour

régler ce problème, une solution, mais pas idéale car elle biaise les résultats des ED, est de remplacer les variances σ_r^2 nul par une petite valeur. Celle-ci est "10*e-7 * `max_var`", valeur qui, comme son nom l'indique, ne peut être dépassée où `max_var`. Si σ_r^2 est zéro et remplacé dans une telle configuration, un message d'avertissement s'affiche pour l'utilisateur. Par exemple, l'exécution de la fonction `computeED` sur l'objet `resLmpEffectMatrices` du modèle Candies, affiche les messages :

```
Avis : The variance of the random Judges variable for response PC5 is 0.
Avis : The variance of the random Candies:Judges variable for response
PC6 is 0.
Avis : The variance of the random Judges variable for response PC8 is 0.
```

Les réponses PC5, PC6 et PC8 étaient bien les modèles singuliers pour l'ensemble Candies.

Construction des matrices d'effet augmentées pour l'APCA et l'ASCA-E

Pour construire les matrices d'effet augmentées, nous avons opter pour deux nouvelles fonctions plutôt qu'une fonction avec un paramètre. Les deux fonctions sont utilisées dans `UPDATE_lmpPcaEffects_3`.

La première `computeAugmentedScoresAPCA`, permet de calculer la matrice augmentée corrigée ou non avec la méthode APCA. Cette fonction a comme paramètre d'entrée :

- la matrice d'effet à augmenter `Meffect` pour l'effet g ;
- la matrice d'effet à additionner `Madd` pour l'effet r ;
- es ED pour l'effet g `df1` et les ED pour l'effet r `df2` ;
- la valeur minimum que les ED `df2` doivent avoir `min_df2`

La fonction construit la matrice d'effet augmentée corrigée avec la formule 2.4.19. Pour cela, la fonction `qf` du package `stats` est utilisée pour trouver le $95^{\text{ième}}$ percentile de la distribution de Fisher `Fstat` avec `df1` et `df2` pour degrés de liberté. La matrice à ajouter `mat`, est calculée :

$$\texttt{mat} = \sqrt{\frac{\texttt{df1}}{\texttt{df2}}\texttt{Fstat}} * \texttt{Madd} \tag{4.2.3}$$

La matrice d'effet augmentée est la matrice de l'effet g `Meffect` additionnée à la matrice `mat`. Si la fonction doit calculer la matrice augmentée non corrigée, `mat` est égale à `Madd`. Une ACP est exécuté sur la matrice d'effet augmentée avec la fonction `pcaBySvd` et les résultats sont renvoyés à la fonction `UPDATE_lmpPcaEffects_3`.

La deuxième fonction `computeAugmentedScoresASCAE` permet de calculer la matrice augmentée, corrigée ou non avec la méthode ASCA-E. Elle est similaire à `computeAugmentedScoresAPCA`. Elles ont toutes deux les mêmes paramètres d'entrées. La matrice d'effet augmentée est calculée exactement de la même façon en additionnant la matrice de l'effet g `Meffect` et la matrice `mat`. Ce qui diffère est l'ACP effectuée sur la matrice d'effet g et ses scores de cette ACP sont modifiés en multipliant les *loadings* de

l'ACP par la matrice d'effet augmentée.

Dans les deux fonctions `computeAugmentedScoresAPCA` et `computeAugmentedScoresASCAE`, les modèles singuliers posent soucis. Toutes les ED petites et problématiques sont les ED des modèles singuliers. Pour l'augmentation de la matrice d'effet $\tilde{\mathbf{M}}_g$, la formule relative à un effet 2.4.19 exploite la distribution de Fisher qui utilise les ED de l'effet à augmenter ED_g `df1` et les ED de l'effet à additionner ED_{add} `df2`. Lorsqu'une des valeurs `df2` est petite, la fonction du percentile de la distribution de Fisher `qf` renvoie un résultat très élevé, voir infini, qui biaise totalement les scores. Le même problème se pose, si une des valeurs `df1` est petite car le résultat de `qf` est alors très petit et biaisé. Lorsque `df1` est trop petit un message d'avertissement apparaît :

```
qbeta(a, *) =: x0 with |pbeta(x0,*) - alpha| = 0.047908 is not accurate
```

La valeur `0.047908` change pour chaque message. Le problème d'un petit `df1` n'influence pas beaucoup les scores de l'ACP. Par contre, un petit `df2` les influence beaucoup. Une solution est de mettre un `df2` minimum nommé `min_df2` qui sera la valeur accordé si celui-ci est plus petit que `min_df2` ; le message d'avertissement suivant est alors affiché :

```
Avis : The degree of freedom df2 is too small. Using df2 = 4
```

Cette solution n'est vraiment pas idéale et devra être améliorée ou changée car celle-ci permet de donner des scores qui n'ont pas l'air mauvais mais la solution ne se base pas sur une vraie logique statistique et `min_df2` avec comme valeur par défaut 4 est déterminée de manière purement intuitif. Le minimum 4 est déjà beaucoup mais un minimum plus petit augmentait beaucoup trop le facteur de correction.

Si le modèle a la même structure que l'un des trois modèles : une ANOVA2 mixte avec interactions, une ANOVA2 mixte hiérarchique ou un modèle longitudinal mixte à deux facteurs (avec effet "Temps" ou "Time"), la fonction `UPDATE_lmpPcaEffects_3` augmente la matrice d'effet de l'effet g par la matrice d'effet de l'effet aléatoire r qui n'est pas la matrice des résidus. Les effets g et r sont déterminés par la structure du modèle.

Pour le modèle ANOVA2 mixte avec interactions, l'effet g de la matrice d'effet à augmenter est l'effet fixe et l'effet r de la matrice d'effet ajoutée est l'effet aléatoire d'interaction. La Table 4.2.1 montre la matrice d'effet et le facteur de correction ajouté pour augmenter chaque matrice d'effet d'un exemple d'un modèle ANOVA2 mixte avec interactions (modèle Candies) :

TABLE 4.2.1 – Description matrice augmentée pour le modèle Candies

Effet	**Type**	**ED associés**	**Matrice ajoutée**	**Facteur de correction**
Candies	F	$ED_C = a - 1$	$\hat{\mathbf{M}}_{CJ}$	$\sqrt{F_{ED_C,ED_{CJ}} \times \frac{ED_C}{ED_{CJ}}}$
Judges	A	ED_J	$\hat{\mathbf{E}}$	$\sqrt{F_{ED_J,ED_E} \times \frac{ED_J}{ED_E}}$
Candies :Judges	A	ED_{CJ}	$\hat{\mathbf{E}}$	$\sqrt{F_{ED_{CJ},ED_E} \times \frac{ED_{CJ}}{ED_E}}$
Résidus	A	ED_E	-	-

où le type, est soit F (fixe), soit A (aléatoire), $a = 5$ est le nombre de niveaux pour le facteur Candies, $ED_E = n - (ED_C + ED_J + ED_{CJ})$ et F_{ED_1,ED_2} est le $95^{\text{ième}}$ quantile de la distribution de Fisher avec ED_1 et ED_2 les ED partiels.

Pour le modèle ANOVA2 mixte hiérarchique, l'effet g de la matrice d'effet à augmenter est l'effet aléatoire principale et l'effet r de la matrice d'effet ajoutée est l'effet aléatoire d'interaction. La Table 4.2.2 montre la matrice d'effet et le facteur de correction ajouté pour augmenter chaque matrice d'effet d'un exemple d'un modèle ANOVA2 mixte hiérarchique(modèle Serum) :

TABLE 4.2.2 – Description matrice augmentée pour le modèle Serum

Effet	**Type**	**ED associés**	**Matrice ajoutée**	**Facteur de correction**
Volunteer	A	ED_V	$\hat{\mathbf{M}}_{VS}$	$\sqrt{F_{ED_V,ED_{VS}} \times \frac{ED_V}{ED_{VS}}}$
Volunteer :Sampling	A	ED_{VS}	$\hat{\mathbf{E}}$	$\sqrt{F_{ED_{VS},ED_E} \times \frac{ED_{VS}}{ED_E}}$
Résidus	A	ED_E	-	-

où le type est, soit F (fixe), soit A (aléatoire), $ED_E = n - (ED_V + ED_{VS})$ et F_{ED_1,ED_2} est le $95^{\text{ième}}$ quantile de la distribution de Fisher avec ED_1 et ED_2 les ED partiels.

Pour le modèle longitudinal mixte à deux facteurs, l'effet g de la matrice d'effet à augmenter est l'effet fixe qui n'est pas l'effet temps et l'effet r de la matrice d'effet ajoutée est l'effet aléatoire. La Table 4.2.3 montre la matrice d'effet et le facteur de correction ajouté pour augmenter chaque matrice d'effet d'un exemple d'un modèle longitudinal mixte à deux facteurs (modèle CHOO) :

TABLE 4.2.3 – Description matrice augmentée pour le modèle CHOO

Effet	**Type**	**ED associés**	**Matrice ajoutée**	**Facteur de correction**
treatment	F	$ED_{Tr} = a - 1$	$\hat{\mathbf{M}}_V$	$\sqrt{F_{ED_{Tr},ED_V} \times \frac{ED_{Tr}}{ED_V}}$
time	F	$ED_{Ti} = b - 1$	$\hat{\mathbf{E}}$	$\sqrt{F_{ED_{Ti},ED_E} \times \frac{ED_{Ti}}{ED_E}}$
treatment :time	F	$ED_{TrTi} = (a-1)(b-1)$	$\hat{\mathbf{E}}$	$\sqrt{F_{ED_{TrTi},ED_E} \times \frac{ED_{TrTi}}{ED_E}}$
volunteer	A	ED_V	$\hat{\mathbf{E}}$	$\sqrt{F_{ED_V,ED_E} \times \frac{ED_V}{ED_E}}$
Résidus	A	ED_E	-	-

où le type est, soit F (fixe), soit A (aléatoire), $a = 2$ est le nombre de niveaux pour le facteur treatment, $b = 3$ est le nombre de niveaux pour le facteur treatment, $ED_E = n - (ED_{Tr} + ED_{Ti} + ED_{TrTi} + ED_V)$ et F_{ED_1,ED_2} est le $95^{\text{ième}}$ quantile de la distribution de Fisher avec ED_1 et ED_2 les ED partiels.

ACP sur les matrices d'effet

La fonction `UPDATE_lmpPCAEffect_3` comporte six paramètres d'entrée :
- `resLmpEffectMatrices` est l'objet en sortie de la fonction `UPDATE_lmpEffectMatrices_2` ;
- `method` qui peut prendre, soit la méthode ASCA, soit la méthode APCA, soit la méthode ASCA-E. Les trois méthodes sont les mêmes que décrites dans la section 2.1 ;
- `backtransform` est un booléen qui, mis à TRUE, permet de "rétro-transformer" (*back-transformed*) la matrice des réponses réduite $\mathbf{Y}^*$ à la section 4.2.3 avec la fonction `lmpOutcomesReduct` pour pouvoir visualiser les vraies réponses se $\mathbf{Y}$ dans le graphe des *loadings* ;
- `combineEffects` est une liste de vecteurs contenant les effets qui seront combinés ;
- `verbose`, si mis à TRUE, permet d'afficher des messages informant de la méthode utilisée, du modèle utilisée et du temps d'exécution de la fonction ;
- `correctedMatrixAdd` est un booléen qui permet, lorsque il est à TRUE, d'augmenter les matrices d'effets avec la formule 2.4.19. Sinon, juste additionner la matrice d'effet aléatoire à ajouter ou la matrice des résidus.

En sortie de la fonction `UPDATE_lmpPCAEffect_3`, l'objet `resLmpPcaEffects` comporte :
- `lmpDataList`, la liste comportant les données de départ ;
- les noms des effets fixes `effectsNamesUnique` et les noms des effets aléatoires `effectsNamesUniqueR` dans le cas LMM ;
- les noms des effets avec les effets combinées fixes `effectsNamesUniqueCombined` et les noms des effets avec les effets combinées aléatoires `effectsNamesUniqueCombinedR` dans le cas LMM ;

— un objet pour chaque effet et effets combinées avec les résultats de l'ACP de la méthode choisie par l'utilisateur ;
— method, le nom de la méthode utilisée pour l'ACP ;
— varComponentsAbs la variance expliquée pour chaque effet dans le cas d'un LMM et type3SS dans le cas d'un LM.
— variationPercentages, le pourcentage de variance expliqué pour chaque effet ;
— combineEffects,la liste de vecteur contenant les effets qui sont combinés.

Le code de UPDATE_lmpPCAEffect_3 a fortement changé même pour la partie LM par rapport à la fonction lmpPCAEffect. La fonction commence par vérifier si resLmpEffectMatrices a le bon nombre de paramètres et a les noms dans le bonne ordre. Cette fonction vérifie aussi s'il y a le même nombre d'effets fixes dans effectMatrices et dans effectsNamesUnique et, ce modèle est un LM et s'assure de plus s'il y a le même nombre d'effets aléatoires dans effectMatricesR et dans effectsNamesUniqueR. Si l'utilisateur n'a pas mis de méthode (method = NULL), la méthode par défaut sera la première donc l'ASCA.

Le paramètre combineEffect doit être une liste. Si des effets doivent être combinés via la liste de vecteurs combineEffect, une liste de matrices d'effets combinées combinedMatrices est construite avec la fonction combineMatrices. combineMatrices permet d'aller chercher pour chaque combinaisons d'effets, les matrices des effets dans effectMatrices pour les effets fixes et dans effectMatricesR pour les effets aléatoires. Un message d'erreur est affiché si la matrice d'effet n'est pas trouvée. combineMatrices combine en additionnant les matrices d'effets avec la fonction Reduce. Le nom des effets combinés est le nom des effets séparés par "+". Le booléen isRandom permet pour chaque combinaison d'effets de dire si au moins un des effets est aléatoire (= TRUE). Si isRandom = TRUE, la combinaison d'effets est considérée comme aléatoire et la matrice d'effet de la combinaison est ajoutée à la liste des matrices d'effets aléatoires effectMatricesR et le nom est ajouté au vecteur effectsNamesUniqueCombinedR. Sinon, la combinaison d'effet est ajouté à la liste des matrices d'effets fixes effectMatrices et le nom est ajoutée au vecteur effectsNamesUniqueCombined. Malheureusement, la matrice augmentée corrigée (correctedMatrixAdd = TRUE) n'est pas supportée dans la version actuelle avec des combinaisons d'effets (combineEffects n'est pas une liste vide). Si l'utilisateur est dans ce cas, un message d'erreur est affiché. Ce cas précis pose problème puisqu'il faut calculer les ED pour les combinaisons d'effets.

Lorsque les effets combinée sont ajoutés aux matrices d'effets, il faut calculer l'ACP en fonction de la méthode demandée. Pour cela, un objet EffectMatLM pour les effets fixes sans l'intercept et un objet EffectMatLMR sont créés, contenant tous les matrices d'effet. Si le modèle est un LM, les résidus sont dans la liste EffectMatLM et il n'y a pas d'objet EffectMatLMR. Par contre, dans le cas d'un LMM, la matrice des résidus est dans l' objet EffectMatLMR.

Si le modèle est un LMM et la méthode est APCA ou ASCA-E, il faut déterminer le type du modèle. Si celui-ci à une structure identique à un des trois modèles, le nom de l'effet de la matrice à ajouter `nameMadd` et le nom de l'effet de la matrice a augmenter `nameMaug` doivent être déterminés comme expliqué dans la sous-section précédente 4.2.8 :

- Pour un modèle ANOVA2 mixte avec interactions, `nameMaug` est l'effet fixe et `nameMadd` est l'effet d'interaction aléatoire ;
- Pour un modèle ANOVA2 mixte hiérarchique, `nameMaug` est l'effet principal aléatoire et `nameMadd` est l'effet d'interaction aléatoire ;
- Pour un modèle longitudinal mixte à deux facteurs, `nameMaug` est l'effet principal fixe qui n'est pas l'effet temps et `nameMadd` est l'effet aléatoire.

Si le modèle n'est pas un des trois types de modèle, toutes les matrices d'effet seront augmentées avec la matrice de résidus. Toutes les autres matrices qui ne sont pas l'effet `nameMaug` seront augmentées par la matrice de résidus.

Les résultats des ACP pour chaque effet contenus dans `EffectMatLM` et `EffectMatLMR` sont mis dans l'objet `resLmpPcaEffects` et `resLmpPcaEffectsR`. Si la méthode est ASCA, l'ACP est exécutée pour chaque matrice d'effet. Si la méthode est APCA, les `ED` sont calculées par la fonction `computeED` si `correctedMatrixAdd = TRUE` ; sinon, une matrice `ED` vide est construite. La fonction `computeAugmentedScoresAPCA` est exécutée sur chaque matrice d'effet sauf les résidus en n'oubliant pas pour l'effet `nameMaug` de l'augmenter avec `nameMadd` et en utilisant la matrice augmentée, corrigée ou non décidé par le paramètre `correctedMatrixAdd`. Si la méthode est ASCA-E, les `ED` sont aussi calculées par la fonction `computeED` ; si `correctedMatrixAdd = TRUE` sinon une matrice `ED` vide est construite. La fonction `computeAugmentedScoresASCAE` est exécutée sur chaque matrice d'effet sauf les résidus en n'oubliant pas pour l'effet `nameMaug` de l'augmenter avec `nameMadd` et en utilisant la matrice augmenté corrigée ou non décidé par le paramètre `correctedMatrixAdd`. Pour l'ASCA-E et l'APCA, une simple ACP est exécutée sur la matrice des résidus et mise dans l'objet `resLmpPcaEffects` pour un LM et dans l'objet `resLmpPcaEffectsR` pour un LMM.

Lorsque le paramètre `backtransform = TRUE` et le paramètre `isReduct = TRUE`, pour chaque effet dans les objets `resLmpPcaEffects` et `resLmpPcaEffectsR`, les `loadings` des résultat de l'ACP de cette étape sont multipliés par les `loadingsPCA` de l'objet `lmpDataList`. Après cela, tous les paramètres pour visualiser les matrices d'effets sont ajoutés à l'objet `resLmpPcaEffects` sorti de la fonction `UPDATE_lmpPCAEffect_3`.

4.2.9 Visualisation des matrices d'effets

Les fonctions dans cette étape sont : `UPDATE_lmpContributions`, `UPDATE_lmpScorePlot`, `UPDATE_lmpLoading1dPlot`, `UPDATE_lmpLoading2dPlot`, `UPDATE_lmpScoreScatterPlotM` et `lmpEffectPlot`. La fonction `lmpScreePlot` n'a quant à elle aucune modification apportée. Les fonctions de cette étape sont pratiquement les mêmes que les fonctions initiales si ce n'est deux changements identiques apportés à toutes les fonctions modifiées sauf `UPDATE_lmpContributions`. Plus précisément :

1. L'objet `resLmpPcaEffects` en entrée de toutes ces fonctions n'a pas le même nombre de paramètres et les mêmes noms dans le cas LM et LMM. Donc, dans le cas où l'objet `resLmpPcaEffects` est utilisé, un contrôle a été ajouté, lorsque le modèle est un LMM, pour vérifier si l'objet a bien le bon nombre de paramètres et si les noms des paramètres sont dans le bon ordre.
2. Il faut ajouter dans les noms des effets `effectNames` ou `effectsNamesUnique`, les effets aléatoires si le modèle est un LMM.

Pour la fonction `UPDATE_lmpContributions`, il faut ajouter au nombre d'effets `neffect`, le nombre d'effets aléatoires si le modèle est un LMM. `neffectTot` est calculé par le nombre de paramètres dans `resLmpPcaEffects` diminué de huit pour un LMM et de six pour un LM.

4.3 Problèmes rencontrés et proposition d'améliorations

Dans cette section, les problèmes rencontrés durant l'élaboration du mémoire et les améliorations effectuées sont développées.

Une des difficultés a été de comprendre la matière et la structure de `limpca`, de LiMM-PCA et du code de M.Martin. Cela m'a pris beaucoup de temps pour assimiler tous les nouveaux concepts comme les tests bootstraps, les méthodes ASCA, APCA et ASCA-E, etc. Cependant, lorsque la matière et les nouveaux concepts ont été compris dans leur globalité, le travail était moins fastidieux. Les principaux problèmes rencontrés ont été expliqués à chaque étape de ce chapitre.

Les améliorations pouvant être apportées à ce projet d'extension sont expliquées ci-dessous.

Une idée d'amélioration serait de créer une fonction qui regroupe toutes les fonctions modifiées, ce qui permet à l'utilisateur de rentrer seulement un objet `lmpDataList` et qui recevrait en sortie :

— la table contenant le pourcentage de variance expliqué et la p-valeur du test de significativité pour chaque effet ;
— les visuels des scores et `loadings` des matrices d'effets pour les trois méthodes ASCA,APCA,ASCA-E ;
— les tables des contributions pour chaque effet.

Il faudrait trouver une meilleure solution pour les problèmes des modèles singuliers dans le calcul des ED et le calcul des matrices augmentées car, pour le moment, mettre des valeurs minimales par défaut n'est pas une solution idéale du point-de-vus statistique.

L'objectif final est la gestion de tous les modèles mixtes. Une étape pour se rapprocher de cette objectif est d'étendre le package à tous les modèles que peut gérer la fonction `lmer`. Cela ne semble pas compliqué sauf pour l'étape de la matrice augmentée, qui est bloquante

pour cette extension car les matrices à augmenter et les matrices à ajouter sont déterminées par la structure du modèle, et généralisation sera difficile. Si le package `limpca` veut vraiment gérer tous les modèles mixtes, une étape supplémentaire consistera à la gestion des modèles de covariance (Covariance pattern models). Les modèles de covariance ne sont pas supportés par la fonction `lmer`, mais il est possible d'utiliser la fonction `lme` du package `nlme` pour estimer ce type de modèle. Toutefois, il faut faire attention à ce que `lme` se base sur la vraisemblance maximale (ML), et non la vraisemblance maximale restreinte (REML) comme `lmer`.

Le test bootstrap pour estimer la significativité d'un effet d'un LMM est trop lent. Il faudrait trouver une solution pour rendre le test bootstrap plus rapide, en cherchant un autre moyen qui permettrait de ne pas estimer un nombre conséquent de modèles.

Le code des fonctions de l'extension du packages `limpca` aux modèles linéaires mixtes (LMM) est fonctionnel et a été fait avec sérieux mais il peut être amélioré et revu par une autre personne pour :

- mieux structurer le code : en ajoutant d'autres extensions de modèles, le code ne devient pas illisible ;
- modifier le code, par exemple les noms des paramètres et des fonctions internes, afin respecter les conventions des packages en R ;
- regarder en profondeur la performance du nouveau code. il y a peut-être des méthode plus facils, par exemple, dans la fonction `UPDATE_lmpBootstrapTests`, le code pour la construction des objets `null_formulas` et `null_form_data` est une opération un peu complexe pour seulement enlever un effet d'une formule ;
- vérifier s'il n'y a pas de redondance de code, surtout entre la gestion des effets fixes et des effets aléatoires ;
- utiliser un seul type de package pour les processus parallèles dans la fonction `UPDATE_lmpBootstrapTests` ;
- permettre de calculer le test bootstrap pour des effets combinés et modifier le code pour permettre que chaque étape soit possible pour des effets combinés. L'extension de `limpca` gèrent actuellement les effets combinées seulement dans la fonction `UPDATE_lmpPCAEffect_3` pour les matrices augmentées non corrigées.

Chapitre 5

Conclusion

Ce mémoire a eu pour objectif d'étendre le package `limpca` aux modèles linéaires mixtes (LMM). Pour réaliser cette extension, quatre ensembles de données ont été utilisés : Candies, Serum, CHOO et UCH. Les trois premiers ensembles ont servis à vérifier que l'extension fonctionnait pour trois structures de LMM différentes et l'ensemble UCH a permis de contrôler que l'extension du package `limpca` fonctionnait toujours correctement. Finalement, l'extension s'est focalisée sur trois modèles linéaires mixtes : une ANOVA2 mixte avec interactions, une ANOVA2 mixte hiérarchique et un modèle longitudinal mixte à deux facteurs (avec effet "Temps" ou "Time"). Le package `limpca` initial se base sur la méthodologie ASCA+ et APCA+ pour gérer les modèles linéaires généraux (LM) mais la méthodologie ne supporte pas les LMM. Pour pouvoir estimer des modèles linéaires mixtes dans l'extension du package `limpca`, une nouvelle méthodologie LiMM-PCA développée par Martin & Govaerts (2020) [2] est introduite.

LiMM-PCA est une extension de ASCA+ et APCA+ au LMM. Les étapes du package `limpca` reste les mêmes mais la réalisation de chacune de ces étapes est adaptée pour pouvoir gérer la partie fixe mais aussi la partie aléatoire. Les principaux changements de l'extension sont :

- la gestion et la vérification de la formule afin de détecter le type de modèle estimé (LM ou LMM) et, s'il s'agit d'un LMM, quelle est sa structure,
- la réduction de dimensions de la matrice de réponses par ACP obligatoire pour les LMM,
- l'estimation du modèle qui est une modélisation parallèle mixte. Pour chaque réponse en parallèle, un modèle linéaire mixte est estimé par la méthode du maximum de vraisemblance restreinte (REML),
- l'utilisation d'une nouvelle formule pour pouvoir calculer le pourcentage de variance expliqué pour les effets aléatoires et fixes sur la même échelle,
- un test bootstrap comme test de significativité d'un effet différent de celui dans le package `limpca` initial,
- une augmentation de matrice d'effet corrigé pour les méthodes ASCA-E et APCA tenant compte :

— des dimensions effectifs (ED) variables pour les effets aléatoires
— de la matrice à additionner qui n'est plus obligatoirement la matrice des résidus.

Pour parvenir à cette extension de package, une fonction a été ajoutée pour réduire les dimensions de la matrice de réponse, la majorité des fonctions (pas les fonctions dans l'exploration des données) du package `limpca` ont été modifiées et des éléments pour la partie aléatoire ont été ajoutés aux objets en sortie des fonctions. Ces modifications ont permis de ressortir, pour des effets aléatoires et fixes, les mêmes types de tables et de graphiques que ceux du package `limpca` de base.

Néanmoins, un problème n'a pas été résolu de manière optimale. Lorsqu'un modèle estime au moins une variance aléatoire nulle, un problème se pose dans le calcul des ED puisqu'une des matrices ne peut pas être inversée et donc, pour pouvoir inverser le calcul des ED, la solution est de remplacer la valeur nulle par une valeur par défaut minimale. Ce qui n'est pas une solution idéale. Ce type de modèle engendre également un autre problème car les ED d'un effet avec une variance estimée nulle par le modèle donnera, si l'effet est ajouté à une matrice d'effet, un facteur de correction très grand qui peut aller jusqu'à l'infini. Pour régler ce deuxième problème, une valeur minimum des ED des effets à ajouter à une matrice d'effet est fixée à 4. Cette solution biaise fortement les résultats des scores. Pour les modèles sans effet aléatoire avec une variance nulle, ces deux problèmes ne se posent pas.

Des améliorations peuvent être apportées à l'extension de package `limpca`. L'extension du package ne gère que trois types de modèles linéaires mixtes. Il pourra dans le futur être étendu à d'autres types de modèles mixtes. Toutes les fonctions de l'extension fonctionnent mais ne sont pas toutes optimales du point de vue performance et lisibilité du code. L'exécution du test bootstrap pour un LMM prend beaucoup trop de temps et devra être améliorée. Une amélioration serait de créer une fonction qui exécute toutes les étapes de LiMM-PCA et renvoie à l'utilisateur toutes les sorties de chaque étape.
Cette extension permet de gérer, en plus des LM, les LMM. Par la suite, d'autres extensions pourront être fusionnées pour pouvoir gérer d'autres types de modèles ou types de données.

Bibliographie

[1] Thiel M, Benaiche N, Martin M, Franceschini S, Van Oirbeek R, Govaerts B. (2023). limpca : An R package for the linear modeling of high-dimensional designed data based on ASCA/APCA family of methods. *Journal of Chemometrics*, 37(7). https://doi.org/10.1002/cem.3482

[2] Martin M, Govaerts B. (2020). LiMM-PCA : Combining ASCA+ and linear mixed models to analyse high-dimensional designed data. *Journal of Chemometrics*, 34(6). https://doi.org/10.1002/cem.3232

[3] Thiel M, Féraud B, Govaerts B. (2017). ASCA+ and APCA+ : extensions of ASCA and APCA in the analysis of unbalanced multifactorial designs.*Journal of Chemometrics*, 31(6). https://doi.org/10.1002/cem.2895

[4] Govaerts B, Francq B, Marion R, Martin M, Thiel M. (2020). 1.16 - The Essentials on Linear Regression, ANOVA, General Linear and Linear Mixed Models for the Chemist. *Comprehensive Chemometrics (Second Edition)*, pages 431-463. https://doi.org/10.1016/B978-0-12-409547-2.14579-2

[5] Liland K H, Smilde A, Marini F, & Næs T. (2018). Confidence ellipsoids for ASCA models based on multivariate regression theory. *Journal of Chemometrics*, 32(5). https://doi.org/10.1002/cem.2990

[6] Luciano G, & Næs T. (2009). Interpreting sensory data by combining principal component analysis and analysis of variance. *Food Quality and Preference*, 20(3), pages 167 – 175. https://doi.org/10.1016/j.foodqual.2008.08.003

[7] Rousseau R. (2011). Statistical contribution to the analysis of metabonomics data in ^{1}H-NMR spectroscopy. *Ph.D. thesis, Institute of Statistics, Biostatistics and Actuarial Sciences, Universit´e catholique de Louvain, Belgium.*

[8] Choo J.M, Kanno T, Zain N.M.M, Leong L.E, Abell G.C, Keeble J.E, Bruce K.D, Mason A.J, Rogers, G.B. (2017). Divergent relationships between fecal microbiota and metabolome following distinct antibioticinduced disruptions. *mSphere*, 2(1) :e00005–17.

[9] Tournay M. (2020). Analyse de données longitudinales à haute dimension : application aux données -omiques. *Faculté des sciences, Université catholique de Louvain.* http://hdl.handle.net/2078.1/thesis:26836

[10] Nakagawa S, Schielzeth H. (2013). A general and simple method for obtaining R2 from generalized linear mixed-effects models. *Methods Ecol Evol*, 4(2), pages 133-142. https://doi.org/10.1111/j.2041-210x.2012.00261.x

[11] Thiel M, Féraud B, Govaerts B. (2017). ASCA+ and APCA+ : extensions of ASCA and APCA in the analysis of unbalanced multifactorial designs. *Journal of Chemometrics*, 31(6). https://doi.org/10.1002/cem.2895

[12] Davison AC, Hinkley DV. (1997). Bootstrap methods and their application (CambridgeSeries in Statistical and Probabilistic Mathematics). *Cambridge : Cambridge university press*. https://doi.org/10.1017/CBO9780511802843

[13] Eilers PHC. (2018). The truth about the effective dimension. *Statistica Neerlandica*, 72(3) :pages 201-209. https://onlinelibrary.wiley.com/doi/abs/10.1111/stan.12131

[14] Guisset S, Martin M, Govaerts B. (2019). Comparison of PARAFASCA, AComDim, and AMOPLS approaches in the multivariate LM modelling of multi-factorial designs. *Chemom Intell Lab Syst*, 184 : pages 44-63.

[15] Martin M. (2020). Uncovering informative content in metabolomics data : from preprocessing of ^{1}H NMR spectra to biomarkers discovery in multifactorial designs. *UCL - SSH/LIDAM/ISBA - Institut de Statistique, Biostatistique et Sciences Actuarielles*. http://hdl.handle.net/2078.1/227671

[16] Bates D., Mächler M., Bolker B., Walker S. (2022). Linear Mixed-Effects Models using 'Eigen' and S4. *Comprehensive R Archive Network (CRAN)*, 1.1-31.

[17] Bates D., Mächler M., Bolker B., Walker S. (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), pages 1-48. https://doi.org/10.18637/jss.v067.i01

[18] Soritz S., Moser D., Gruber-Wölfler H. (2022). Comparison of Derivative-Free Algorithms for their Applicability in Self-Optimization of Chemical Processes. *Chemistry–Methods*, 2(5). https://doi.org/10.1002/cmtd.202100091