

# RAPPORT DE STAGE D'EXCELLENCE

UNIVERSITÉ GRENOBLE ALPES (UGA)

LABORATOIRE JEAN KUNTZMANN (LJK)

---

## Chute de blocs rocheux : organisation et analyse statistique de données acquises sur modèle réduit

---

*Auteur :*

Thomas PIRAS

*Maitre de stage :*  
Dr. Frédérique LEBLANC (LJK)

*Commanditaire de l'étude :*  
Dr. Dominique DAUDON (3SR)

Durée du stage : 03 Juin 2019 - 26 Juillet 2019



## Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Présentation générale . . . . .	2
1.2	Présentation du laboratoire . . . . .	2
1.3	Appréciation personnelle du dispositif de stage d'excellence . . . . .	2
1.4	Langue des documents annexes . . . . .	2
1.5	Problématique et objectif . . . . .	2
<b>2</b>	<b>Présentation du modèle réduit et de l'expérience</b>	<b>2</b>
2.1	Le modèle réduit . . . . .	2
2.2	L'expérience et la notion de setup . . . . .	3
<b>3</b>	<b>Organisation des données fournies</b>	<b>4</b>
3.1	Données fournies . . . . .	4
3.1.1	Convention de nommage des fichiers . . . . .	4
3.1.2	Nombre de mesures effectuées . . . . .	4
3.1.3	Éjecta - Données censurées . . . . .	4
3.2	Choix et organisation du support de stockage . . . . .	4
3.2.1	Base de donnée <b>SQLLite</b> . . . . .	4
3.2.2	Table <b>data</b> . . . . .	5
3.2.3	Table <b>stat</b> . . . . .	6
3.3	Création de la base de données SQLite . . . . .	6
3.3.1	<b>CsvLEANER.py</b> . . . . .	6
3.3.2	<b>RocheDB.py</b> . . . . .	6
3.3.3	Exemple d'exécution des scripts . . . . .	7
3.4	Interaction entre la base de données et R . . . . .	7
3.4.1	Requête <b>SQLLite</b> . . . . .	7
<b>4</b>	<b>Analyse statistique</b>	<b>8</b>
4.1	Analyse individuelle des lancés . . . . .	8
4.1.1	Recherche de modèles adaptés à nos données . . . . .	8
4.1.2	Ajustement d'un modèle Gamma . . . . .	9
4.1.2.1	La loi Gamma $\Gamma(k, \theta)$ . . . . .	9
4.1.2.2	Estimation des paramètres . . . . .	9
4.1.2.3	Validation par un test d'adéquation et calcul d'une p-valeur . . . . .	10
4.1.3	Transformée de Box-Cox . . . . .	10
4.1.4	Application des modèles ajustés . . . . .	11
4.1.4.1	Problème causé par les éjectas . . . . .	11
4.2	Analyse et comparaison de setup . . . . .	11
4.2.1	Présentation de la démarche . . . . .	11
4.2.1.1	Exemple 1 : impact du rangement des objets avant leurs lancés . . . . .	11
4.2.1.2	Exemple 2 : croisement du type d'objet lancé et des revêtements . . . . .	12
4.2.1.3	Résumé des preuves et observations . . . . .	13
<b>5</b>	<b>Conclusion</b>	<b>13</b>
5.1	Contact . . . . .	13
<b>6</b>	<b>Bibliographie</b>	<b>14</b>
<b>A</b>	<b>README.md pour CsvLEANER.py et RocheDB.py</b>	<b>15</b>
<b>B</b>	<b>CsvLEANER.py</b>	<b>16</b>
<b>C</b>	<b>RocheDB.py</b>	<b>17</b>

# 1 Introduction

## 1.1 Présentation générale

Je m'appelle Thomas PIRAS et je suis un étudiant en 2ème année de licence Informatique au Département de la Licence Sciences et Technologies (DLST) de l'Université Grenoble Alpes (UGA). Cet été j'ai eu l'occasion de faire un stage d'une durée d'environ deux mois au Laboratoire Jean Kuntzmann (LJK), sous la tutelle de Frédérique LEBLANC. Ce stage a été réalisé en collaboration avec Dominique DAUDON du laboratoire 3SR (Sols, Solides, Structures, Risque), c'est elle qui a créé le modèle réduit et l'expérience sur laquelle j'ai travaillé durant ce stage.

## 1.2 Présentation du laboratoire

Le Laboratoire Jean Kuntzmann, nommé d'après le célèbre mathématicien et informaticien Grenoblois est un laboratoire d'informatique et de mathématiques appliquées situé dans le bâtiment IMAG du campus de Saint-Martin d'Hères.

## 1.3 Appréciation personnelle du dispositif de stage d'excellence

Je suis content d'avoir eu l'occasion de faire un stage d'excellence, ce fut une bonne expérience qui m'a permis, l'espace de quelques mois, de m'immiscer dans la vie des chercheurs et de découvrir un tout nouveau sujet de travail.

Ce stage m'a aussi permis d'approfondir mes connaissances dans plusieurs domaines tels que les statistiques, le logiciel R ainsi que la librairie ggplot2 [WG17] mais aussi les bases de données et le logiciel Python.

## 1.4 Langue des documents annexes

Tous les documents annexes ainsi que les commentaires dans les scripts et programmes ont été rédigé en anglais pour permettre aux personnes ne parlant pas français de pouvoir utiliser les outils créés et de travailler sur le sujet.

## 1.5 Problématique et objectif

Les chutes de bloc rocheux constituent un vrai problème dans les régions montagneuses, leurs études s'avèrent très importantes pour qu'une gestion du territoire bonne et sûre soit mise en place. Par exemple leurs études peuvent permettre de cartographier les zones où les risques de chutes de roches sont les plus importantes et ainsi définir si un terrain est constructible ou non.

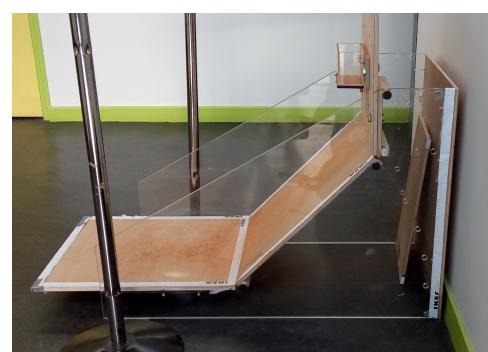
Identifier et quantifier les multiples facteurs environnementaux qui ont un impact sur la distance parcouru par les blocs, est le problème qui nous intéresse ici. Le but principal étant de pouvoir estimer la distance à laquelle un objet va se stopper pour un niveau des facteurs donné, ce qui pourrait nous aider à définir les délimitations de zones à risques et d'estimer plus précisément lesdits risques.

# 2 Présentation du modèle réduit et de l'expérience

## 2.1 Le modèle réduit

Le modèle réduit a été construit par Dominique DAUDON du laboratoire 3SR, il très modulaire pour permettre de simuler au mieux divers facteurs environnementaux tels que :

- Le nombre d'objets qui chutent en même temps.
- La nature des objets qui vont tomber (6 types d'objets lancés).
- La disposition des objets avant la chute (mis en vrac ou rangé dans la zone de lancement)
- Les matériaux qui vont recouvrir la pente et la zone d'arrivée (impact de la rugosité et de la dureté du sol).
- L'angle de la pente.



Tous les facteurs et leurs différents niveaux sont résumés ci dessous :

- **Nombre d'objet lancés** : 14, 28, 40, 42
- **Nature de l'objet** : brickFull, brickHalf, dice4, dice6, dice8, smallDice4
- **Organisation des objets** : loose, neat
- **Matériaux pente** : wood, grey, black, teflon
- **Matériaux arrivée** : wood, grey, black, teflon, sand
- **Angle de la pente** : 45



Matériaux pouvant recouvrir la pente ou l'arrivée



Les différents objets lancés

## 2.2 L'expérience et la notion de setup

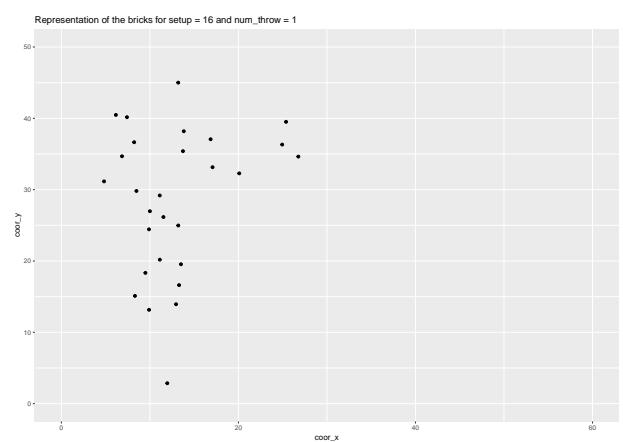
Un *setup* est une combinaison unique des différents niveaux des facteurs.

L'expérience menée est la suivante : pour chaque *setup* on a  $N$  lancés de  $n$  objets, avec  $N$  de l'ordre de 50 et  $n$  de l'ordre de 28. À la fin de chaque lancé, une photo est prise de la zone d'arrivée, ce qui nous permet après un traitement informatique de récupérer les coordonnées des  $n$  objets lancés.

Par la suite on notera  $X$  et  $Y$  les variables aléatoires décrivant l'abscisse et l'ordonnée de la position d'un bloc.  $X$  représente la distance parcourue dans le sens de la pente. Les résultats obtenu sur un lancé peuvent être représenté par un nuage de points  $(x_i, y_i)_i$ .



Photo de la zone d'arrivée après un lancé.



Nuage de points du lancé.

### 3 Organisation des données fournies

#### 3.1 Données fournies

Au début du stage, nous avons reçu 20 fichiers au format .CSV (Comma Separated Values), les données présentes dans ces fichiers ont été recueillis par un précédent stagiaire d'excellence (du laboratoire 3SR). Dans chaque fichier nous pouvons trouver les coordonnées des objets pour chaque lancé relatif à un setup, ainsi que d'autres informations tel que le nombre d'objets dans une partie de la planche d'arrivée.

##### 3.1.1 Convention de nommage des fichiers

Nous avons choisi de nommer ces fichiers en suivant le format définit ci-dessous pour pouvoir facilement identifier les données :

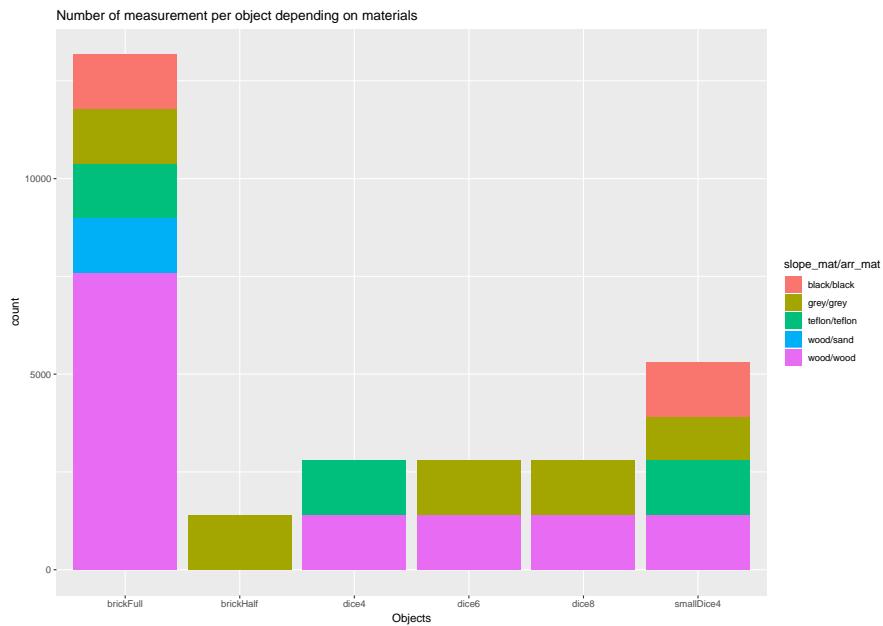
nombreObjetsLancés\_natureObjet\_rangéOuNon\_matériauxPente\_matériauxArrivée\_angleDeLaPente

Exemple de nom de fichier : 28\_brickFull\_loose\_wood\_sand\_45.csv

##### 3.1.2 Nombre de mesures effectuées

Il est important de noter que la quantité de mesures effectuées n'est pas la même selon les setups, de plus tous les croisements entre objets lancés et matériaux de la pente/arrivée n'ont pas été réalisé, comme nous pouvons le voir sur le graphique à droite.

Le nombre de mesures pour l'objet `brickFull` sur les matériaux `wood/wood` est très important, car c'est avec cet objet et ces matériaux qu'on a fait plusieurs fois la même expérience mais avec un nombre d'objets lancés variable, dans le but de pouvoir étudier l'effet de ce facteur sur la distance parcourue par les objets.



##### 3.1.3 Éjecta - Données censurées

**Définition :** Un objet est appelé un *éjecta* s'il sort de la zone d'arrivée du modèle réduit. Comme celle-ci fait 60 cm de longueur, on sait que l'objet est allé plus loin que 60 cm mais on ne sait pas la distance réelle qu'il a parcourue. En statistiques on appelle ce type de donnée des données censurées à droite.

Dans les données qui nous ont été fournies un éjecta est codé par des coordonnées *x* et *y* valant 0.

### 3.2 Choix et organisation du support de stockage

#### 3.2.1 Base de donnée SQLite

Nous avons tout d'abord commencé par dresser une liste d'objectifs/constraints pour nous aider à choisir un système de stockage adapté à nos besoins :

##### Objectifs :

- Regrouper l'ensemble des données fournies dans un même support de stockage.
- Organiser les données de manière structurée et facilement lisible.

##### Contraintes :

- Traitement des données commode avec R.

- Ajout facile de nouvelles données au support de stockage choisi.
- Le support doit avoir une bonne gestion mémoire de la redondance.

À partir de cette liste nous avons choisi d'utiliser une base de données **SQLite**. Nous avons fait ce choix car en plus de respecter les contraintes établies dans la liste précédente, ces bases de données sont "serverless", c'est à dire qu'elles tournent en local et donc cela nous évite de devoir avoir un serveur pour faire tourner la base de données ; ici c'est plus simple tout le monde peut créer, modifier et partager la sienne aisément , puisque une base de données **SQLite** est un fichier local.

Par ailleurs utiliser une base de données avec **R** peut s'avérer très pratique, car avec ces dernières il n'est pas nécessaire de charger l'intégralité des données dans un dataframme. Il est très facile de mettre dans un dataframme un sous-ensemble bien précis de nos données, cela peut être pratique pour les personnes disposant d'ordinateurs peu puissants et peut généralement économiser du temps de calcul.

Pour plus de détails consultez la section 3.4.

### 3.2.2 Table data

La première table de la base de données est nommée **data**, c'est elle qui va contenir les coordonnées des objets pour chaque lancé de chaque **setup**.

Nous avons choisi d'organiser cette table de la manière suivante :

SETUP	NUM_THROW	BR_ID	COOR_X	COOR_Y	NUM_OBJ	NUM_OOB	ANGLE	SORTED	OBJ	SLOPE_MAT	ARR_MAT
Entier	Entier	Entier	Réel	Réel	Entier	Entier	Entier	Booléen	Entier	Entier	Entier

Détails des colonnes :

- **SETUP** : Entier allant de 1 à 20 correspondants au numéro du fichier initial d'où sont extraites les données, peut-être utile pour comparer *directement* deux environnements différents, mais aussi pour savoir dans quel environnement on se trouve sans avoir à regarder les matériaux de la pente, le type d'objet lancé etc...
- **NUM\_THROW** : Entier allant de 1 à  $N$  correspondants au numéro de lancé pour un environnement donné.
- **BR\_ID** : Pour "*brick identifier*", est un entier allant de 1 à **NUM\_OBJ** ( $n$ ), correspondant à l'identifiant d'un objet lancé. Cet identifiant est défini par l'ordre des données initiale et a pour seul but de simplifier l'utilisation de la base de données, car pour un même type d'objet il est très peu probable qu'un même objet est le même identifiant entre deux lancés différents.
- Nous avons fait ce choix de colonne pour avoir des données plus verticales et donc plus simples à lire que le format de données initial qui est très horizontal.
- **COOR\_X** : Flottant correspondant à la coordonnées en  $x$  d'un objet d'identifiant **BR\_ID**
- **COOR\_Y** : Flottant correspondant à la coordonnées en  $y$  d'un objet d'identifiant **BR\_ID**
- **NUM\_OBJ** : Entier valant 14, 28, 40 ou 42 avec les données actuelles. Correspond au nombre d'objets étant lâchés ( $n$ ).
- **NUM\_OOB** : Pour "*number out of bounds*", entier correspondant au nombre d'éjecta pour un lancé donné, c'est à dire au nombre d'objets qui sort de la zone d'arrivée à l'issue d'un lancé.
- **ANGLE** : Entier indiquant à quel degré d'inclinaison la pente a été réglée. Toutes nos données ont été récoltées avec un même degré d'inclinaison mais nous rajoutons cette colonne dans le but de faciliter l'ajout de nouvelles données au dataset.
- **SORTED** : Entier qui vaut 1 si les objets ont été rangés dans le boîtier de lancement et 0 s'ils ont été disposés en vrac.
- **OBJ** : Entier indiquant de quel objet il s'agit. Comme pour **SLOPE\_MAT** et **ARR\_MAT** il est possible de voir quel objet (respectivement quelle matière) correspond à quel numéro dans le fichier **DB\_Creator/filenameSyntax.json**.
- **SLOPE\_MAT** : Pour "*slope material*", est un entier indiquant quel matériaux ou revêtement est présent sur la pente du modèle réduit.
- **ARR\_MAT** : Pour "*arrival material*", est un entier indiquant quel matériaux ou revêtement est présent sur la partie où les objets arrivent et se stoppent ("l'arrivée").

**Remarque** : Il est possible d'organiser la base de données pour éviter la redondance importante de données

qui existe actuellement, en effet toutes les colonnes après `C00R_Y` sont propres au setup (sauf `NUM_OOB`) et seront répétés  $N \times n$  fois. Il suffirait de mettre ces colonnes ainsi que le numéro de setup dans une autre table de la base de données et de les enlever de la table `data`, ensuite il faudrait joindre la table `data` avec cette autre table (en utilisant l'opérande `JOIN`) dans les requêtes SQL pour obtenir les mêmes résultats que précédemment, la redondance en moins. Nous avons choisi de ne pas adopter cette solution car elle aurait pour but de complexifier les requêtes SQL que nous allons utiliser par la suite, or notre objectif est qu'une personne sans aucune expérience avec les bases de données et les langages de programmation puisse se servir aisément de la base de données avec R. Néanmoins cette solution pourrait s'avérer utile si le jeu de données venait à s'accroître considérablement.

### 3.2.3 Table stat

La seconde table de la base de données est nommée `stat`, elle regroupe les valeurs de plusieurs outils statistiques pour chaque lancé de chaque setup.

Toutes les valeurs dans la table ont été calculées uniquement pour  $X$  et *sans tenir compte des éjectas*, à moins qu'il ne soit précisé autrement. Les éléments de la table seront expliqués en détail plus tard.  
Elle est organisée comme suit :

<code>min</code>	<code>max</code>	<code>mean</code>	<code>25%</code>	<code>50%</code>	<code>75%</code>	<code>88%</code>	<code>91%</code>	<code>25%_oob</code>	<code>50%_oob</code>	<code>75%_oob</code>	<code>88%_oob</code>
Réel	Réel	Réel	Réel	Réel	Réel	Réel	Réel	Réel	Réel	Réel	Réel
<code>91%_oob</code>	<code>cor_coef</code>	<code>pval_shap_norm</code>	<code>pval_lillie_norm</code>	<code>pval_ks_gamma</code>	<code>pval_shap_boxcox</code>	<code>prop_oob</code>	<code>prob_oob_ks_gamma</code>	<code>prob_oob_boxcox</code>			
Réel	Réel	Réel	Réel	Réel	Réel	Réel	Réel	Réel			

Détails des colonnes :

- `min`, `max`, `mean` : Réels correspondant à la valeur de la coordonnée minimale, maximale et moyenne de l'échantillon.
- `25%`, `50%`, `75%`, `88%`, `91%` : Réels correspondant à la valeur de différent quantile.
- `25_oob%`, `50_oob%`, `75_oob%`, `88_oob%`, `91_oob%` : Réels correspondant à la valeur de différent quantile, calculé avec les éjectas.
- `cor_coef` : Coefficient de corrélation entre  $X$  et  $Y$ .
- `pval_shap_norm`, `pval_lillie_norm`, `pval_ks_gamma`, `pval_shap_boxcox` : Respectivement, Réels correspondant à la p-valeur obtenue en effectuant le test de normalité de Shapiro-Wilk, le test de normalité de Lilliefors, un test d'adéquation à une loi Gamma de Kolmogorov-Smirnov, le test de normalité de Shapiro-Wilk sur des données auxquelles on a appliqué une transformée de Box-Cox.
- `prop_oob`, `prob_oob_ks_gamma`, `prob_oob_boxcox` : Respectivement, Réels correspondant à la proportion d'éjecta dans l'échantillon, la probabilité estimée que  $P(X > 60)$  obtenue grâce à une loi Gamma, similaire à l'élément précédent mais pour une loi normale obtenue en transformant nos données avec la transformée de Box-Cox.

## 3.3 Création de la base de données SQLite

Pour que tout le monde puisse créer et mettre à jour la base de données en partant simplement de données brutes, nous avons créé deux scripts Python pour automatiser et faciliter ce processus.

Il est **impératif** que les fichiers de données brutes (ceux utilisés pour créer la table `data`) aient été renommés comme défini dans la sous-section 3.1.1, car les scripts utilisent les informations contenues dans les noms des fichiers.

### 3.3.1 CsvLEANER.py

Le but de `CsvLEANER.py` est de 'nettoyer' les fichiers de données brutes en enlevant toutes les colonnes qui sont inutiles à la création de la table `data`. Vous pouvez consulter le code source ici : Annexe B.

### 3.3.2 RocheDB.py

Pour créer ou ajouter de nouvelles données à la table `data` ce script utilise les fichiers .CSV créé par le script précédent.

De manière analogue, il utilise les fichiers .CSV créé par le script `4_stat_table_data.R` pour créer ou ajouter de nouvelles données à la table `stat`. Vous pouvez consulter le code source ici : Annexe C.

### 3.3.3 Exemple d'exécution des scripts

```
[bot@X200 ~]$ cd Documents/Cours/Stage-LJK/DB_Creator/
[bot@X200 DB_Creator]$ python CsvLEANER.py
# --- CsvLEANER --- #
```

Please write the path of the directory containing the CSV files: `../DATA/sorted_raw_data`

```
- 28_smallDice4_loose_teflon_teflon_45.csv -- CLEANED
- 28_brickfull_loose_wood_wood_45.csv -- CLEANED
- 45_brickfull_loose_wood_wood_45.csv -- CLEANED
- 14_brickfull_loose_wood_wood_45.csv -- CLEANED
- 28_smallDice4_loose_black_black_45.csv -- CLEANED
- 28_dice4_loose_wood_wood_45.csv -- CLEANED
- 28_dice8_loose_grey_grey_45.csv -- CLEANED
- 28_dice4_loose_teflon_teflon_45.csv -- CLEANED
- 28_dice6_loose_grey_grey_45.csv -- CLEANED
- 28_brickfull_loose_grey_grey_45.csv -- CLEANED
- 28_brickfull_loose_black_black_45.csv -- CLEANED
- 28_smallDice4_loose_wood_wood_45.csv -- CLEANED
- 28_smallDice4_loose_grey_grey_45.csv -- CLEANED
- 28_brickfull_loose_wood_sand_45.csv -- CLEANED
- 28_brickfull_loose_teflon_teflon_45.csv -- CLEANED
- 40_brickfull_loose_wood_wood_45.csv -- CLEANED
- 28_brickHalf_loose_grey_grey_45.csv -- CLEANED
- 28_dice6_loose_wood_wood_45.csv -- CLEANED
- 28_dice8_loose_wood_wood_45.csv -- CLEANED
```

All the CSV files in the given directory have been cleaned.

```
[bot@X200 DB_Creator]$
```

Exécution de `CsvLEANER.py`

```
[bot@X200 ~]$ cd Documents/Cours/Stage-LJK/DB_Creator/
[bot@X200 DB_Creator]$ python RocheDB.py
# --- RocheDB --- #
[0] Create the table data on the DB from CSV files
[1] Add new data to an existing data table on the DB from CSV files
[2] Create the stat table in an existing DB from CSV files
[3] Add new data to an existing stat table on the DB from CSV files
[4] Exit

Please select an option [0-4]: 0
--- Data Table Creation ---
Please input a name for the DB: exemple
[WARNING]: Make sure to feed CSV files cleaned with CsvLEANER
Please write the path of your directory containing the clean CSV files: ../DATA/clean_data/
- 14_brickfull_loose_wood_wood_45_clean.csv -- ADDED TO DB
- 28_brickfull_loose_black_black_45_clean.csv -- ADDED TO DB
- 28_brickfull_loose_grey_grey_45_clean.csv -- ADDED TO DB
- 28_brickfull_loose_wood_sand_45_clean.csv -- ADDED TO DB
- 28_brickfull_loose_wood_wood_45_clean.csv -- ADDED TO DB
- 28_brickfull_loose_neat_wood_wood_45_clean.csv -- ADDED TO DB
- 28_brickHalf_loose_grey_grey_45_clean.csv -- ADDED TO DB
- 28_dice4_loose_teflon_teflon_45_clean.csv -- ADDED TO DB
- 28_dice6_loose_wood_wood_45_clean.csv -- ADDED TO DB
- 28_dice8_loose_grey_grey_45_clean.csv -- ADDED TO DB
- 28_dice8_loose_black_black_45_clean.csv -- ADDED TO DB
- 28_smallDice4_loose_grey_grey_45_clean.csv -- ADDED TO DB
- 28_smallDice4_loose_black_black_45_clean.csv -- ADDED TO DB
- 28_smallDice4_loose_wood_wood_45_clean.csv -- ADDED TO DB
- 40_brickfull_loose_wood_wood_45_clean.csv -- ADDED TO DB
- 42_brickfull_loose_wood_wood_45_clean.csv -- ADDED TO DB
```

Exécution de `RocheDB.py`

Pour plus de détails sur l'exécution et le fonctionnement des scripts se référer à l'annexe A.

### 3.4 Interaction entre la base de données et R

Pour pouvoir utiliser la base de données avec R il faut tout d'abord installer trois librairies : `RSQlite`, `DBI` et `odbc`.

Pour les installer il suffit d'ouvrir un script R et d'exécuter les commandes suivantes :

```
1 install.packages("RSQlite")
2 install.packages("DBI")
3 install.packages("odbc")
```

Pour plus de détails, ou si vous rencontrez des problèmes d'installation vous pouvez consulter ce site : [db.rstudio.com](http://db.rstudio.com)

Le script R suivant montre comment récupérer un sous ensemble des données dans un data frame :

```
1 # Charger les librairies
2 library(odbc)
3 library(DBI)
4 library(RSQlite)
5
6 # Définition du répertoire de travail
7 setwd("~/Documents/Cours/Stage-LJK/DATA/data_base")
8
9 # Connexion a la base de données
10 conn = dbConnect(SQLite(), dbname="chute_de_bloc.sqlite")
11
12 # Exécution d'une requête
13 req = dbSendQuery(conn,
14                     "SELECT num_oob, num_obj, coor_x, coor_y FROM data WHERE (obj = 0 AND slope_mat = 3)"
15                     )
16
17 # Récupération des résultats dans un data frame
18 df <- dbFetch(req, n = -1)
```

Nous allons parler plus en détail de la ligne 14 dans la sous-section suivante.

#### 3.4.1 Requête SQLite

La ligne numéro 14 correspond à ce que l'on appelle une *requête SQLite*, c'est grâce à celle-ci que l'on communique avec la base de données et qu'on lui indique quelles informations on souhaite qu'elle nous

donne. Notre base de données a été organisée (intentionnellement) de façon simple, il n'y a donc que trois mots-clés à connaître :

- **SELECT** : indique les colonnes à sélectionner dans la table.
  - Si le nom d'une colonne commence par un nombre il faut le mettre entre guillemets ("").
  - Pour sélectionner toutes les colonnes il suffit d'écrire \* après **SELECT**.
  - **FROM** : indique dans quelle table de la base de données il faut récupérer les colonnes sélectionnées précédemment, donc dans notre cas il faut écrire soit **data** soit **stat** après **FROM**.
  - **WHERE** : Ce mot clé est facultatif, il sert à mettre une ou des conditions sur les données que l'on souhaite récupérer.
- Dans l'exemple ci dessus on l'utilise pour récupérer quatre colonnes pour un type d'objet et un revêtement de la pente donnés.

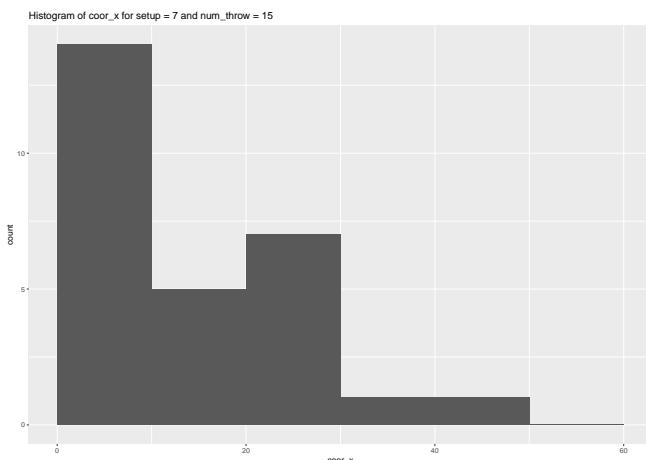
Si vous voulez en apprendre plus sur les requêtes SQLite ou que vous avez besoin d'aide vous pouvez consulter les tutoriels de ce très bon site web : [sqlitetutorial.net](http://sqlitetutorial.net)

## 4 Analyse statistique

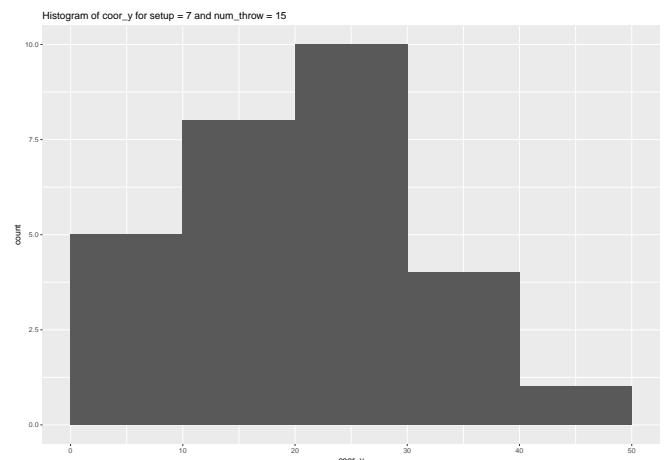
### 4.1 Analyse individuelle des lancés

#### 4.1.1 Recherche de modèles adaptés à nos données

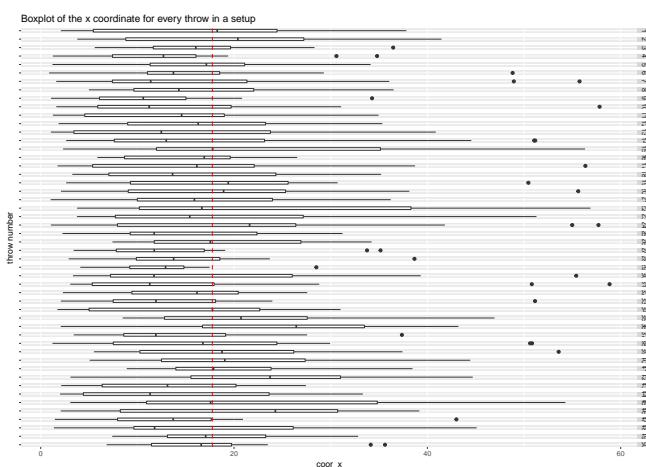
Nous avons tout d'abord commencé par produire des Box-plots et des histogrammes, ainsi que, des descriptions numériques simples (minimum, maximum, moyenne, quartile...) pour des lancés individuels. Ceci a servi à nous donner de l'intuition pour choisir des modèles statistiques correspondant bien à nos données.



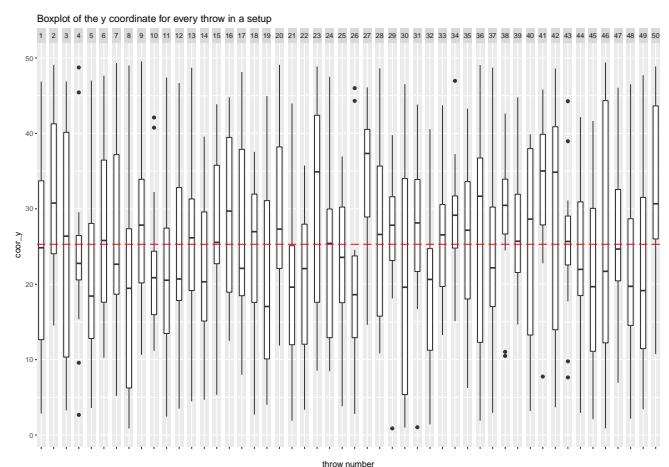
Histogramme de **coor\_x** pour un lancé,  $n = 28$



Histogramme de **coor\_y** pour un lancé,  $n = 28$



Box-plots de **coor\_x** pour tous les lancés d'un setup



Box-plots de **coor\_y** pour tous les lancés d'un setup

On peut voir sur les histogrammes que la distribution horizontale des objets ( $X$ ) semble asymétrique, au contraire de la distribution verticale ( $Y$ ) qui est relativement symétrique.  
Si l'on regarde les Box-plots des 50 lancés d'un setup, malgré les variations, on observe globalement des répartitions similaires aux histogrammes plus haut.

Suite aux observations précédentes nous proposons donc les modèles suivants :

- Pour  $X$  : une loi Gamma  $\Gamma(k, \theta)$ .
- Pour  $Y$  : Loi Normale  $\mathcal{N}(\mu, \sigma^2)$ .

Par ailleurs afin de pouvoir utiliser les méthodes statistiques usuelles supposant un modèle normal, nous allons aussi transformer les données en  $X$  avec la transformée de *Box-Cox* afin de les rendre gaussiennes.

Par la suite nous nous sommes majoritairement concentré sur l'étude des données en  $X$ , puisque l'objectif principal est d'estimer la distance maximale parcourue par les objets. Par ailleurs, les données en  $Y$  ne montrent pas d'effets significatifs des facteurs, nous ne discuterons pas ici plus longuement de ces données.

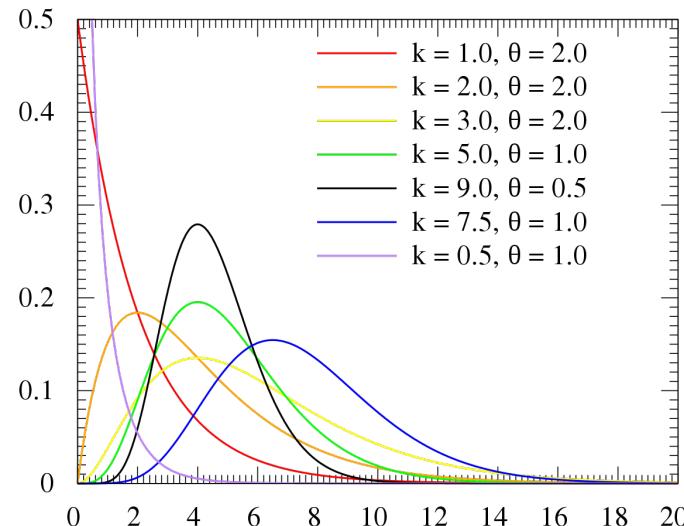
#### 4.1.2 Ajustement d'un modèle Gamma

##### 4.1.2.1 La loi Gamma $\Gamma(k, \theta)$

La loi Gamma possède 2 paramètres toujours strictement positifs :

- $k$  : Le paramètre de forme.  
C'est lui qui impacte la forme de la représentation graphique. Pour  $k \leq 1$  elle ressemble à une distribution exponentielle et pour  $k \geq 1$  à une loi du  $\chi^2$ .
- $\theta$  : Le paramètre d'échelle.  
Ce paramètre régis "l'aplatissement" de la courbe.  
Plus  $\theta$  est petit et plus la densité est concentrée vers 0.
- $E(X) = k\theta$
- $Var(X) = k\theta^2$

Sur l'image adjacente on peut voir la densité de probabilité de la loi Gamma en fonction de différentes valeurs de  $k$  et  $\theta$ .



##### 4.1.2.2 Estimation des paramètres

Pour ajuster ce modèle en fonction de nos données, il nous faut estimer les paramètres de la loi Gamma grâce à nos données. Pour faire cela nous avons utilisé la *méthode des moments*.

Cette méthode consiste à résoudre le système d'équations obtenu en égalisant certains moments *théoriques* avec leurs contreparties *empiriques* (ici nous utiliserons l'espérance et la variance).

On connaît le meilleur estimateur de l'espérance et de la variance, respectivement  $\widehat{E(X)} = \bar{X}$  (la moyenne) et  $\widehat{Var(X)} = S'^2$  (variance empirique corrigée). On a donc le système :

$$\bar{X} = k\theta$$

$$S'^2 = k\theta^2$$

Il nous suffit de le résoudre pour trouver l'estimation de nos deux paramètres. On obtient donc :

$$\hat{k} = \frac{\bar{X}^2}{S'^2}$$

$$\hat{\theta} = \frac{\bar{X}}{S'^2}$$

#### 4.1.2.3 Validation par un test d'adéquation et calcul d'une p-valeur

Nous avons estimé les paramètres de la loi Gamma en fonction de nos données, nous voulons maintenant vérifier que  $X$  suit bien cette loi. Pour s'assurer que c'est bien le cas nous allons utiliser un test d'adéquation : le test de Kolmogorov-Smirnov.

Nous commençons par poser les hypothèses du test :

$$\begin{aligned}\mathcal{H}_0 &: X \text{ suit la loi } \Gamma(\hat{k}, \hat{\theta}) \\ \mathcal{H}_1 &: X \text{ ne suit pas la loi } \Gamma(\hat{k}, \hat{\theta})\end{aligned}$$

Ce test est basé sur une quantification de la distance entre la fonction de répartition empirique de nos données et la fonction de répartition testée (ici  $\Gamma(\hat{k}, \hat{\theta})$ ). En effet, la statistique du test [Con99] est :

$$T = \sup_X |F_n(X) - F(X)|$$

Avec  $F_n(X)$  la fonction de répartition empirique et  $F(X)$  la fonction de répartition testée.

On note  $T_{calc}$  la valeur de  $T$  obtenue pour les données observées. Grâce à cette statistique nous allons pouvoir calculer une *p-valeur*. La p-valeur est un nombre entre 0 et 1 et c'est elle qui va nous aider à déterminer à quel point nos résultats sont significatifs. Dans notre cas la formule de la p-valeur est la suivante :

$$pval = P(T > T_{calc} | \mathcal{H}_0)$$

Interprétation de la p-valeur :

- Une *petite p-valeur* (typiquement  $\leq 0.05$ ) indique une validation statistiquement significative de  $\mathcal{H}_1$ . En effet dans notre cas si la p-valeur est petite, cela veut dire que  $T_{calc}$  est grand et donc qu'il y a une grande distance entre la fonction de répartition empirique et la fonction de répartition testée, ce qui implique que  $X$  ne suit pas la fonction de répartition testée. On choisira donc l'hypothèse  $\mathcal{H}_1$  (avec un faible risque de se tromper).
- Une *grande p-valeur* (typiquement  $> 0.1$ ), de manière analogue, va nous indiquer que  $T_{calc}$  est petit et donc que nos deux fonctions sont relativement proches. Par conséquent on ne dispose d'aucun élément significatif permettant de rejeter  $\mathcal{H}_0$ .

Pour les données fournies, pour chaque lancé individuel, on ne rejette jamais le modèle Gamma pour  $X$  (la plupart des p-valeurs dépassent 50%). Nous pouvons en conclure que  $X \rightsquigarrow \Gamma(\hat{k}, \hat{\theta})$ .

**Remarques :**

1. Il est possible de consulter les différentes p-valeurs que le test a retourné dans la table `stat` de la base de données, à la colonne `pval_ks_gamma` ou dans les fichiers .CSV du dossier `DATA/basic_statistics`.
2. Les p-valeurs obtenues avec le test de Kolmogorov-Smirnov (proposé dans la routine R: `ks.test`) ne sont pas valides si les paramètres de la distribution testée ont été estimés à partir des données. Nous avons donc calculé la loi de la statistique de test sous l'hypothèse nulle ( $\mathcal{H}_0$ ) par simulations. Pour plus de détails se référer au script `3_estimations_probability_oob.R`.

#### 4.1.3 Transformée de Box-Cox

Les statisticiens George Box et David Cox ont développé une procédure pour identifier un exposant (que nous appellerons  $\lambda$ ) qui va être utilisé pour "normaliser" des données.

La fonction `boxcox` (du paquet `fitdistr` de R) fournit  $\hat{\lambda}$  entre -6 et +6, qui maximise la log-vraisemblance. La définition formelle de la transformée [BC64] est :

$$B(X, \lambda) = \begin{cases} \frac{X^\lambda - 1}{\lambda} & \text{si } \lambda \neq 0 \\ \log(X) & \text{si } \lambda = 0 \end{cases}$$

En notant les données transformées :

$$\tilde{X} = B(X, \hat{\lambda})$$

On a :

$$\tilde{X} \rightsquigarrow \mathcal{N}(\widehat{\mu}_{\tilde{X}}, \widehat{\sigma}_{\tilde{X}}^2)$$

Où  $\widehat{\mu}_{\tilde{X}}$  et  $\widehat{\sigma}_{\tilde{X}}^2$  sont les paramètres estimés de la loi Normale suivie par  $\tilde{X}$

#### 4.1.4 Application des modèles ajustés

À l'aide des deux modèles que nous avons ajustés sur nos données, il nous est désormais possible de calculer des estimations de la probabilité que  $X$  dépasse un certain seuil fixé  $s$ , on note  $\hat{p}_s = P(X > s)$ .

Si on prend  $s = 60$  (la longueur de la zone d'arrivée), que  $X \rightsquigarrow \Gamma(\hat{k}, \hat{\theta})$ ,  $\tilde{X} = B(X, \hat{\lambda}) \rightsquigarrow \mathcal{N}(\widehat{\mu}_{\tilde{X}}, \widehat{\sigma}_{\tilde{X}}^2)$  nous sommes à même de proposer trois estimations de probabilités de sortie :

- $\widehat{p}_{s,0}$  = la proportion d'éjecta observée
- $\widehat{p}_{s,1} = 1 - F(s, \hat{k}, \hat{\theta})$  avec  $F$  : FdR de la  $\Gamma(\hat{k}, \hat{\theta})$
- $\widehat{p}_{s,2} = 1 - F(s, \widehat{\mu}_{\tilde{X}}, \widehat{\sigma}_{\tilde{X}}^2)$  avec  $F$  : FdR de la  $\mathcal{N}(\widehat{\mu}_{\tilde{X}}, \widehat{\sigma}_{\tilde{X}}^2)$

On peut appliquer cette démarche pour tout autre seuil  $s$  différent de 60.

##### 4.1.4.1 Problème causé par les éjectas

On observe que, pour des setups ayant un pourcentage d'éjectas assez faible (environ  $\leq 5\%$ )  $\widehat{p}_{60,1}$  et  $\widehat{p}_{60,2}$  sont très proches de la proportion d'éjecta observée. En revanche, quand le pourcentage d'éjecta augmente on peut voir que  $\widehat{p}_{60,1}$  et  $\widehat{p}_{60,2}$  sont beaucoup plus petit que la proportion d'éjecta observée.

Ceci peut s'expliquer par le fait que nous ignorons les éjectas dans nos calculs pour estimer les paramètres de nos lois (comme ceux-ci sont codés par des 0).

Pour la loi Gamma ceci à pour effet de sous-estimer la valeur des paramètres  $k$  et  $\theta$ , et donc de venir "décaler" la courbe à gauche et de réduire de manière importante la queue de distribution. De manière similaire pour la loi Normale, les paramètres sont aussi sous-estimés mais les impacts sont légèrement moins importants que pour la loi Gamma.

## 4.2 Analyse et comparaison de setup

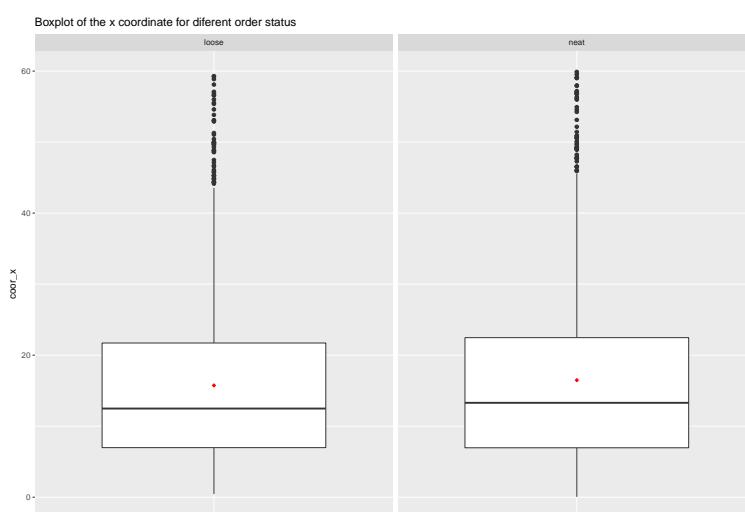
À la suite de cette première étude de comparaison de lancé individuel, nous nous sommes intéressés à l'analyse et à la comparaison de setup, dans le but de trouver quels facteurs sont les plus importants et de voir les différences entre les niveaux de facteurs.

### 4.2.1 Présentation de la démarche

En se servant de la librairie `ggplot2` de R [WG17] nous avons fait de nombreux graphiques croisant les différents facteurs. Après avoir analysé ces graphiques nous utilisons des tests statistiques pour prouver de manière formelle les observations visuelles faites sur les graphiques.

Ci-dessous vous pourrez trouver deux exemples pour illustrer cette démarche.

#### 4.2.1.1 Exemple 1 : impact du rangement des objets avant leurs lancés



Répartition observée de  $X$  selon le fait que les objets soient rangés ou non dans la boîte avant le lancé.

On peut voir à droite le Box-plot de  $X$  pour les objets (ici ce sont des briques) mis en vrac dans la boîte et à gauche celui des objets rangés, le point rouge représente la moyenne et la barre noire la médiane. Il est possible d'observer que ces deux Box-plots semblent très similaires. On pose comme hypothèse nulle que le fait de ranger ou non les objets dans la boîte n'impacte pas le centre de gravité en  $X$ . Et nous prouvons que cette hypothèse ne peut être rejetée de façon statistiquement significative. Le test de comparaison des moyennes utilisé pour donner cette conclusion est le test de *Mann-Whitney-Wilcoxon* qui est un test non-paramétrique, ce qui nous permet de ne poser aucune hypothèse sur la loi de  $X$ . Par ailleurs comme les tailles d'échantillons sont suffisamment grande le test de comparaison de moyenne de *Student* s'applique, sans qu'il soit indispensable de supposer les variables de la loi normale.

```

1 # Après récupération des données dans df
2 x_loose = df$coor_x[df$setup == 6 & !is.na(df$coor_x)]
3 x_neat = df$coor_x[df$setup == 7 & !is.na(df$coor_x)]
4
5 # Mann-Whitney-Wilcoxon's Test
6 wilcox.test(x_loose, x_neat, paired = F) # pval = 0.2119
7
8 # -- Sortie du test --
9 # Wilcoxon rank sum test with continuity correction
10 # data: x_loose and x_neat
11 # W = 874690, p-value = 0.2119
12 # alternative hypothesis: true location shift is not equal to 0
13
14 # Student's test
15 t.test(x_loose, x_neat, paired = F) # pval = 0.1156
16
17 # -- Sortie du test --
18 # Welch Two Sample t-test
19 # data: x_loose and x_neat
20 # t = -1.574, df = 2667.1, p-value = 0.1156
21 # alternative hypothesis: true difference in means is not equal to 0
22 # 95 percent confidence interval:
23 # -1.6634526 0.1820404
24 # sample estimates:
25 # mean of x mean of y
26 # 15.73891 16.47961

```

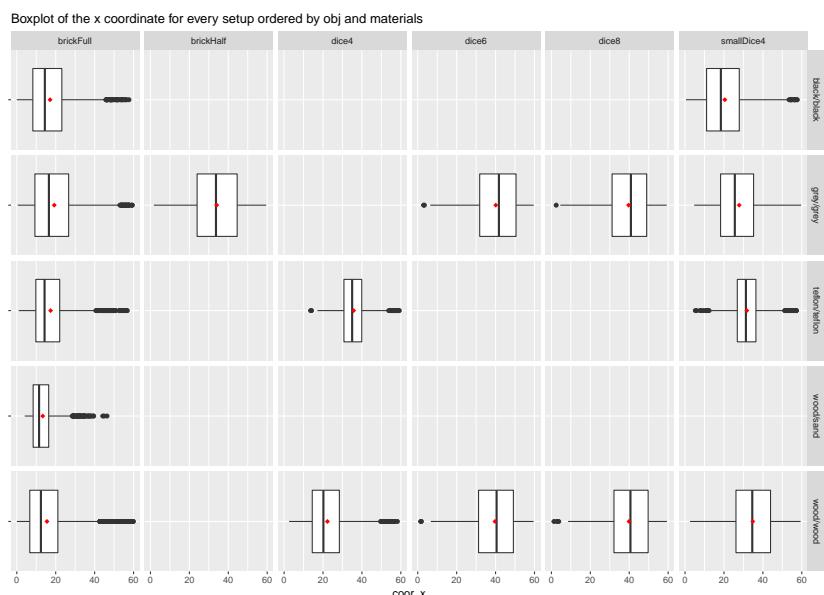
Les deux tests retournent une p-valeur supérieure à 10%, on ne peut donc pas conclure que la portion du centre de gravité des objets en  $X$  est impacté par le fait de ranger ou pas les objets dans la boîte.

#### 4.2.1.2 Exemple 2 : croisement du type d'objet lancé et des revêtements

On peut voir sur le graphique ci-contre les Box-plots de  $X$  pour tous les croisements d'objets et de revêtement. Ce graphique est très intéressant car il met en valeur de nombreuses informations, il permet aussi de se rendre compte que tous les croisements d'objets et de revêtements n'ont pas été fait, cela limite l'étude que nous allons conduire ici.

Pour commencer regardons la première colonne (celle de l'objet brickFull), on peut voir que le type de revêtement à un effet certain sur la variation de la distribution et du centre de gravité des objets.

Par exemple, sur le sable (avant-dernière ligne), les objets vont bien moins loin et sont bien plus concentrés que sur les



autres revêtements. En utilisant des procédés similaires nous avons aussi démontré que les lancés avec le revêtement black/black et wood/wood produisent des échantillons pour lesquels on ne rejette pas l'égalité de variance, nous n'avons pas trouvé d'autre similarité de ce genre entre les autres revêtements.

En revanche, nous pouvons observer que le centre de gravité et la variation de la distribution des objets sont la plus importante sur le revêtement gris (grey/grey). Ce résultat nous semblait de prime abord surprenant, car nous nous attendions à voir ce résultat sur un des revêtements lisse comme le bois ou le téflon, mais D. Daudon a corroboré ce résultat, l'attribuant au fait que le revêtement gris (de même que le revêtement noir) est un peu molletonné et donc plus rebondissant. Cela pousse les objets à rebondir et donc à aller plus loin et ce, malgré la rugosité du revêtement.

Ce graphique nous permet aussi de constater que le type d'objet joue un grand rôle sur la distance que les objets vont parcourir. Si l'on regarde la ligne du revêtement gris, on peut voir que les objets qui ont une forme de type "ronde" (`brickHalf`, `dice6`, `dice8`) vont plus loin que les autres (`brickFull`, `dice4`). Ce résultat semble normal, et l'on peut constater le même phénomène sur le revêtement bois, car plus un objet est rond et moins il lui faudra d'énergie pour passer d'une face à une autre et par conséquent qu'il puisse rouler, ce qui va faire qu'il atteindra une distance plus élevée.

Nous pensons donc que le facteur qui va impacter le plus la distance qui va être parcourue par les objets est leurs formes.

#### 4.2.1.3 Résumé des preuves et observations

- Le nombre d'objets lancés en même temps à un impact sur la distribution des objets : plus il y a d'objets lancés en même temps et moins la distance parcourue par les objets est grande.
- Le fait que les objets soient rangés ou non dans la boîte avant le lancé n'a pas d'influence sur leurs distributions.
- Le revêtement à un effet sur la distribution des objets.
- La forme de l'objet lancé semble être le facteur qui a le plus d'effet sur la distance que l'objet va parcourir.

Pour voir l'intégralité des graphiques réalisé se référer au script `2_db_exploration_graph.R`, pour tout les tests statistiques conduit sur nos données voir le script `1_descriptive_statistics.R`.

## 5 Conclusion

Pour conclure cette étude, nous avons montré que le problème des données censurées à droite s'avère crucial pour être en mesure de donner des estimations précises de dépasser un certain seuil.

Nous proposons deux solutions à ce problème : soit modifier le modèle réduit pour faire en sorte de ne plus avoir de données manquantes, soit utiliser des outils statistiques qui prennent en compte les données censurées. La première méthode requiert l'acquisition de nouvelles données, la deuxième en revanche nécessite de faire une fouille des données plus en profondeur et d'utiliser des modèles statistiques plus complexes qui prennent en compte les données censurées.

Indépendamment de ce problème il est nécessaire d'acquérir plus de données pour pouvoir mieux étudier l'impact des différents facteurs sur la distance parcourue par les blocs. Cela pourrait faire l'objet d'un futur stage.

### 5.1 Contact

Pour toutes questions vous pouvez me contacter a l'adresse mail suivante :  
`thomas.piras@etu.univ-grenoble-alpes.fr`

## 6 Bibliographie

### Références

- [BC64] G. E. P. Box et D. R. Cox. “An Analysis of Transformations”. In : *Journal of the Royal Statistical Society. Series B (Methodological)* 26.2 (1964). ISSN : 00359246. URL : <http://www.jstor.org/stable/2984418>.
- [Con99] W.J CONOVER. *Practical nonparametric statistics*. John Wiley et Sons, inc., 1999. Chap. Statistics of the Kolmogorov-Smirnov Type.
- [WG17] Hadley WICKHAM et Garrett GROLEMUND. *R for Data Science : Import, Tidy, Transform, Visualize, and Model Data*. 1st. O'Reilly Media, Inc., 2017. ISBN : 1491910399, 9781491910399.

## A README.md pour CsvLEANER.py et RocheDB.py

```

1 # Execution
2 **[WARNING]**: These scripts have been created and tested on GNU/LINUX. I cannot guarantee that they will
3 → work flawlessly on Windows or Mac.
4
5 To run these two scripts you need to have Python 3 installed on your system.
6 Then you just open a terminal, go where these script are located and write ```python RocheDB.py``` or
7 → ```python3 CsvLEANER.py``` depending on your system.
8
9 # CsvLEANER.py
10
11 CsvLEANER.py ask for the path of a directory containing CSV files and then proceed to remove all unneeded
12 → data from these files.
13 (Columns XcoinHG1, YcoinHG1, XcoinHD2... As well as all the columns after the objects' coordinates).
14
15 ## Requirement and comments
16 Your CSV files' names must follow this pattern (the name of the file is used to identify the
17 → modifications to do):
18 *numberObjectThrown_objectName_orderedOrNot_slopeMaterial_arrivalMaterial_angleOfTheSlope*
19
20 Your CSV files must have the same organisation as the ones in ```DATA/sorted_raw_data```.
21
22 If the number of objects thrown in your CSV is not in the following list: 14, 28, 40, 42; You will need to
23 → add a condition in CsvLEANER.py and define an array of columns to ignore and the number of the last
24 → column to accept in your CSV files.
25 Even if you have no experience with Python this should be fairly easy, you can try to understand better by
26 → looking at the already existing conditions and their related CSV files.
27
28 The clean CSV files will be saved in the directory in which you execute the script.
29
30 # RocheDB.py
31 RocheDB.py creates or updates the two tables in the database.
32 * The table **data**: This is the table that contains all the coordinates of the different objects as well
33 → as all the information relative to the setups, you can create this table by choosing the option *0*.
34 The .CSV files you will feed to the script must have been cleaned by ```CsvLEANER.py``` beforehand.
35 Same thing if you want to update the table of an existing database, except you will need to choose the
36 → option *1*.
37 * The table **stat**: This table contains a lot of already computed statistical information, the .CSV
38 → files required to create this table can be created by launching the following script
39 → ```statistical_analysis/4_stat_table_data.R```.
40 To be able to launch this script you need to have already created a database with the data table.
41 To create the table you will need to choose the option *2* and to update it the option *3*.
42
43 ## Requirement and comments
44 Your CSV files' names must follow this pattern (the name of the file is used to identify the
45 → modifications to do):
46 *numberObjectThrown_objectName_orderedOrNot_slopeMaterial_arrivalMaterial_angleOfTheSlope*
47
48 If your data has been collected using new objects, new materials for the slope, new materials for the
49 → arrival, a number of objects or an angle that has not been used before you must input these new
50 → information in ```filenameSyntax.json```.
51 This JSON file is used to make sure that there are no mistakes in the CSV files' names, but also to
52 → associate a number to each kind of objects and material in the DB.
53
54 It is important to understand that you can always give path to the script, for example, if you execute the
55 → script from ```DB_Creator``` and your database is located at ```DATA/data_base```, when the script
56 → asks you for the *name* of the database you are expected to give:
57 → ```../DATA/data_base/chute_de_bloc.sqlite```.
58
59 The script will also create (or update if you have already created a DB) a file called ```context.txt```,
60 → where you can see which filename is linked to which setup number.
61
62 The SQLite DB and the context.txt file will be saved in the directory in which you execute the script.
63 For the most part the script should guide you through its use.

```

## B CsvLEANER.py

```
1 # Import zone
2 import os
3 import csv
4
5 def CsvLEANER(directory, fileToClean):
6     if (fileToClean.startswith("28")):
7         unwanted = [3, 4, 5, 6, 7, 8, 9, 10]
8         limit = 66
9     elif (fileToClean.startswith("14")):
10        unwanted = [3, 4, 5, 6, 7, 8, 9, 10, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38]
11        limit = 52
12    elif (fileToClean.startswith("40")):
13        unwanted = [3, 4, 5, 6, 7, 8, 9, 10]
14        limit = 90
15    elif (fileToClean.startswith("42")):
16        unwanted = [3, 4, 5, 6, 7, 8, 9, 10]
17        limit = 94
18    else:
19        print("Error: the number of objects thrown in a file as not been configured, or your files are not
20             ↪ named correctly.")
21        exit()
22
23 cleanFile = fileToClean.replace('.csv', '_clean.csv')
24 fileToClean = directory + fileToClean
25 j = 0
26
27 with open(fileToClean, 'r') as f_in, open(cleanFile, 'w') as f_out:
28     csv_reader = csv.reader(f_in, delimiter=' ')
29     csv_writer = csv.writer(f_out, delimiter=' ')
30
31     for row in csv_reader:
32         if (j > 0):
33             colValues = []
34             for i in range(0, len(row)):
35                 if (i > limit):
36                     csv_writer.writerow(colValues)
37                     break
38                 elif (i not in unwanted):
39                     colValues.append(row[i])
40             j+=1
41
42 # --- MAIN ---
43 print("# --- CsvLEANER --- #")
44 print()
45
46 path = input("Please write the path of the directory containing the CSV files: ")
47 print()
48
49 # Makes sure that the given path is a directory
50 if (not path.endswith('/')):
51     path = path + '/'
52
53 if (not os.path.isdir(path)):
54     print("Error: This directory does not exist")
55     exit()
56
57 directory = os.fspath(path)
58 directory = directory.decode()
59
60 for filename in os.listdir(directory):
61     if (filename.endswith(".csv")):
62         CsvLEANER(directory, filename)
63         print("- {} -- CLEANED".format(filename))
64     else:
65         print("- {} -- IGNORED".format(filename))
```

```

66
67     print()
68     print("All the CSV files in the given directory have been cleaned.")

```

## C RocheDB.py

```

1  # Import zone
2  import os
3  import re
4  import csv
5  import json
6  import sqlite3
7
8
9  def checkSyntaxFilename(filename):
10     """
11         Check if the syntax of the file is correct using filenameSyntax.json.
12     """
13     with open('filenameSyntax.json', 'r') as json_file:
14         syntax = json.load(json_file)
15         lst = filename.split('_')
16         if (len(lst) != 7):
17             print('Error: The following filename is not formated properly {}'.format(filename))
18             return False
19         for i in range(len(lst)):
20             if (i == 0 and int(lst[i]) not in syntax["j_num_obj"]):
21                 print("Error: Filename syntax invalid, --num_obj-- not recognized or approved for file
22                     {}".format(filename))
23                 return False
24             if (i == 1 and lst[i] not in syntax["j_obj"]):
25                 print("Error: Filename syntax invalid, --obj-- not recognized or approved for file
26                     {}".format(filename))
27                 return False
28             if (i == 2 and lst[i] not in syntax["j_sorted"]):
29                 print("Error: Filename syntax invalid, --sorted-- not recognized or approved for file
30                     {}".format(filename))
31                 return False
32             if (i == 3 and lst[i] not in syntax["j_slope_mat"]):
33                 print("Error: Filename syntax invalid, --slope_mat-- not recognized or approved for file
34                     {}".format(filename))
35                 return False
36             if (i == 4 and lst[i] not in syntax["j_arr_mat"]):
37                 print("Error: Filename syntax invalid, --arr_mat-- not recognized or approved for file
38                     {}".format(filename))
39                 return False
40
41     def filenameToInfo(filename):
42         """
43             Return a list of the informations in the filename.
44             The list will be as follow: [num_obj, obj, sorted, slope_mat, arr_mat, angle, clean.csv]
45         """
46         filename.replace('.csv', '')
47         lst = filename.split('_')
48         return lst
49
50
51     def maxSetup(data=True):
52         """
53             Return the biggest setup number in the table data.
54             Exit if the table data is empty as there's no max setup number to return.
55             [WARNING]: A connexion must have been established with the DB before using this function.

```

```
56
57     """
58     if data:
59         req_max_number_setup = 'SELECT MAX(setup) FROM data'
60     else:
61         req_max_number_setup = 'SELECT MAX(setup) FROM stat'
62     cursor.execute(req_max_number_setup)
63     setup = cursor.fetchone()[0] # fetchone returns a tuple
64
65     if (setup is None):
66         print('Error: This database is empty')
67         exit()
68
69     setup += 1
70     return setup
71
72 def checkEndDB(dbName):
73     """
74     Check if the file extension is missing from the DB name given by the user.
75     """
76     if (not dbName.endswith('.sqlite')):
77         dbName = dbName + '.sqlite'
78     return dbName
79
80
81 def atoi(text):
82     """
83     Convert the string text to an integer if the string is a number.
84     """
85     return int(text) if text.isdigit() else text
86
87
88 def natural_keys(text):
89     """
90     Generate the natural keys to sort in human order.
91     alist.sort(key=natural_keys) sorts in human order.
92     """
93     return [ atoi(c) for c in re.split(r'(\d+)', text) ]
94
95
96 def addNewElementsToData(setup, directory):
97     """
98     Add the content of every clean CSV file that follows the syntax to the table 'data' of the DB.
99     """
100    req_insertion = 'INSERT INTO data VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)'
101    json_file = open('filenameSyntax.json', 'r')
102    syntax = json.load(json_file)
103
104    # Create or open context.txt
105    if (setup == 1):
106        context_file = open("context.txt", 'w')
107        context_file.write('---- Setup number to full filename ----\n\n')
108    elif (setup > 1):
109        context_file = open("context.txt", 'a')
110
111    # Sort the list of files in the directory
112    dir = os.listdir(directory)
113    dir.sort(key = natural_keys)
114
115    for filename in dir:
116        if (filename.endswith("_clean.csv")):
117            if (not checkSyntaxFilename(filename)):
118                print("- {} -- IGNORED".format(filename))
119                continue
120            else:
121                print("- {} -- ADDED TO DB".format(filename))
122                context_file.write('{} -- {}'.format(setup, filename))
123                context_file.write('\n')
124
```



```
190 # Push the changes to the DB
191 conn.commit()
192 conn.close()
193 exit()

195
196 # --- MAIN ---
197 print("# --- RocheDB --- #")
198 print()
199 print("[0] Create the table data on the DB from CSV files")
200 print("[1] Add new data to an existing data table on the DB from CSV files")
201 print("[2] Create the stat table in an existing DB from CSV files")
202 print("[3] Add new data to an existing stat table on the DB from CSV files")
203 print("[4] Exit")
204 print()
205 choice = input("Please select an option [0-4]: ")

207
208 if(choice == '0'):
209     print()
210     print("-- Data Table Creation --")

212 dbName = input("Please input a name for the DB: ")
213
214 dbName = checkEndDB(dbName)
215
216 # If a DB with this name exist delete it
217 # otherwise there will be an error during the creation of the DB
218 if (os.path.isfile(dbName)):
219     os.remove(dbName)
220
221 # Create the table
222 req_table_creation = '''CREATE TABLE data
223             (setup INTEGER NOT NULL,
224              num_throw INTEGER NOT NULL,
225              br_id INTEGER NOT NULL,
226              coor_x REAL NOT NULL,
227              coor_y REAL NOT NULL,
228              num_obj INTEGER NOT NULL,
229              num_oob INTEGER NOT NULL,
230              angle INTEGER NOT NULL,
231              sorted INTEGER NOT NULL CHECK (sorted=0 OR sorted=1),
232              obj INTEGER NOT NULL,
233              slope_mat INTEGER NOT NULL,
234              arr_mat INTEGER NOT NULL,
235              PRIMARY KEY (setup, num_throw, br_id)
236          )''''
237 conn = sqlite3.connect(dbName)
238 cursor = conn.cursor()
239 cursor.execute(req_table_creation)
240
241 print("[WARNING]: Make sure to feed CSV files cleaned with CsvLEANER")
242 path = input("Please write the path of your directory containing the clean CSV files: ")
243 if (not path.endswith('/')):
244     path = path + '/'
245
246 if (not os.path.isdir(path)):
247     print("Error: This directory does not exist")
248     exit()
249 directory = os.fsencode(path)
250
251 addNewElementsToData(1, directory.decode())
252
253
254 elif (choice == '1'):
255     print()
256     print("-- Data Table Update --")
257
258 # Connect to the DB
```

```
259     dbName = input("Please input the name of the database to open: ")
260
261     dbName = checkEndDB(dbName)
262
263     if (not os.path.isfile(dbName)):
264         print("Error: the database file doesn't exist")
265         exit()
266
267     conn = sqlite3.connect(dbName)
268     cursor = conn.cursor()
269
270     print("[WARNING]: Make sure to feed CSV files cleaned with CsvLEANER")
271     print("[WARNING]: Make sure that all of the CSV files in your directory are not already in the DB")
272     path = input("Please write the path of your directory containing the clean CSV files to add to the DB:
273     ↪ ")
273     if (not path.endswith('/')):
274         path = path + '/'
275
276     if (not os.path.isdir(path)):
277         print("Error: This directory does not exist")
278         exit()
279     directory = os.fsencode(path)
280
281
282     addNewElementsToData(maxSetup(), directory.decode())
283
284
285 elif (choice == '2'):
286     print()
287     print("-- Stat Table Creation --")
288
289     # Connect to the DB
290     dbName = input("Please input the name of the database to open: ")
291
292     dbName = checkEndDB(dbName)
293
294     if (not os.path.isfile(dbName)):
295         print("Error: the database file doesn't exist")
296         exit()
297
298     conn = sqlite3.connect(dbName)
299     cursor = conn.cursor()
300
301     # Create the table
302     req_table_creation = '''CREATE TABLE stat
303             (setup INTEGER NOT NULL,
304              num_throw INTEGER NOT NULL,
305              min REAL NOT NULL,
306              max REAL NOT NULL,
307              mean REAL NOT NULL,
308              "25q" REAL,
309              "50q" REAL,
310              "75q" REAL,
311              "88q" REAL,
312              "91q" REAL,
313              "25q_oob" REAL,
314              "50q_oob" REAL,
315              "75q_oob" REAL,
316              "88q_oob" REAL,
317              "91q_oob" REAL,
318              cor_coef REAL,
319              pval_shap_norm REAL,
320              pval_lillie_norm REAL,
321              pval_ks_gamma REAL,
322              pval_shap_boxcox REAL,
323              prop_oob REAL,
324              prob_oob_ks_gamma REAL,
325              prob_oob_boxcox REAL,
326              PRIMARY KEY (setup, num_throw)
```

```
327             )'''  
328     cursor.execute(req_table_creation)  
329  
330     print("[WARNING]: Make sure that your CSV files are designed for this table")  
331     path = input("Please write the path of your directory containing CSV files to add to the stat table: ")  
332     if (not path.endswith('/')):  
333         path = path + '/'  
334  
335     if (not os.path.isdir(path)):  
336         print("Error: This directory does not exist")  
337         exit()  
338     directory = os.fsencode(path)  
339  
340     addNewElementsToStat(1, directory.decode())  
341  
342  
343 elif (choice == '3'):  
344     print()  
345     print("-- Stat Table Update --")  
346  
347 # Connect to the DB  
348 dbName = input("Please input the name of the database to open: ")  
349  
350 dbName = checkEndDB(dbName)  
351  
352 if (not os.path.isfile(dbName)):  
353     print("Error: the database file doesn't exist")  
354     exit()  
355  
356 conn = sqlite3.connect(dbName)  
357 cursor = conn.cursor()  
358  
359 print("[WARNING]: Make sure that your CSV files are designed for this table")  
360 path = input("Please write the path of your directory containing CSV files to add to the stat table: ")  
361 if (not path.endswith('/')):  
362     path = path + '/'  
363  
364     if (not os.path.isdir(path)):  
365         print("Error: This directory does not exist")  
366         exit()  
367     directory = os.fsencode(path)  
368  
369     addNewElementsToStat(maxSetup(data=False), directory.decode())  
370  
371  
372 elif (choice == '4' or int(choice) not in range(4+1)):  
373     exit()
```