

Louvain School of Management

Annexes techniques

Auteur : Di Tullio Léa

Promoteur(s) : D'Hondt Catherine

Année académique : 2025-2026

Travail de fin d'étude (TFE) en vue d'obtenir le titre de
Master [120] Ingénieur de gestion à finalité spécialisée

Horaire de jour

Table des matières

Annexe 1 - Code Python	3
Initialisation.....	3
Suppression des colonnes.....	3
Attribution des noms aux colonnes.....	3
Suppression des 6 premières lignes (en-têtes Qualtrics).....	4
Encodage sémantique des variables catégorielles.....	4
Création des colonnes binaires « asset ».....	5
Encodage des questions financières (Q8, Q9, Q10).....	5
Encodage des échelles de Likert	5
Normalisation des colonnes Fonds B / Fonds C	6
Sauvegarde du fichier final	6
Annexe 2 - Code R.....	7
Chargement des packages	7
Ajout des variables dummies	7
Test de normalité.....	7
Fiabilité des échelles	8
Test de multicollinéarité.....	8
Matrice de corrélation de Spearman	8
Régressions OLS.....	9
Régressions Logit.....	9
Régressions Ordinales.....	10

Annexe 1 - Code Python

Initialisation

```
"""
Nettoyage des données issues de Qualtrics.
Produit un fichier CSV final : data_memoire_VF.csv
"""

import pandas as pd
import numpy as np

# Chemins des fichiers
CSV_PATH = "/content/memoire_lea2.csv"
OUTPUT_PATH = "/content/data_memoire_VF.csv"

# Fonctions utilitaires
def step(n, label):
    print(f"\n{'='*70}")
    print(f"STEP {n}: {label}")
    print(f"{'='*70}")

def show(df, name, extra=''):
    print(f" [{name}] shape: {df.shape}")
    if extra:
        print(f" {extra}")

# Chargement des données
df = pd.read_csv(CSV_PATH)
show(df, 'df', f'Total rows: {len(df)}, Total cols: {len(df.columns)}')
```

Suppression des colonnes

```
cols_to_drop = [
    "StartDate", "EndDate", "Status", "Progress", "Duration (in seconds)",
    "Finished", "RecordedDate", "ResponseId", "DistributionChannel",
    "UserLanguage", "Q_RecaptchaScore", "Q_DuplicateRespondent",
    "Q29_1", "Q29_2", "Q29_3",
    "SC0", "Q_RecaptchaStatus", "Q_RecaptchaError",
    "Q_DuplicateRespondentStatus", "Q_BallotBoxStuffing",
    " ", "12", " 12", "12 ",
]
dropped = [c for c in cols_to_drop if c in df.columns]
df_clean = df.drop(columns=dropped)
```

Attribution des noms aux colonnes

```
rename_map = {
    "Q1": "age", "Q2": "genre", "Q3": "etude", "Q4": "profession",
    "Q5": "salaire", "Q6": "investir", "Q7": "risque",
    "Q31_1": "Fond B", "Q31_2": "Fond C",
}
df_renamed = df_clean.rename(columns={k: v for k, v in rename_map.items()
                                     if k in df_clean.columns})

rename_map_incremental = {
    # Q11..Q14 -> SECURITY1..4
    **{f'Q{i}_1': f'SECURITY{i - 10}' for i in range(11, 15)},
    # Q15..Q18 -> FREEDOM1..4
```

```

    **{f'Q{i}_1': f'FREEDOM{i - 14}' for i in range(15, 19)},
    # Q19..Q21 -> POWER1..3
    **{f'Q{i}_1': f'POWER{i - 18}' for i in range(19, 22)},
    # Q22..Q23 -> LOVE1..2
    **{f'Q{i}_1': f'LOVE{i - 21}' for i in range(22, 24)},
    # Q24..Q26 -> POWER4..6
    **{f'Q{i}_1': f'POWER{i - 20}' for i in range(24, 27)},
}
df_renamed2 = df_renamed.rename(columns={k: v for k, v in
rename_map_incremental.items()
                                if k in df_renamed.columns})

```

Suppression des 6 premières lignes (en-têtes Qualtrics)

```
df_filtered = df_renamed2.iloc[6:].reset_index(drop=True)
```

Encodage sémantique des variables catégorielles

```

# Age
age_map = {
    "Moins de 18 ans": 1, "18-24 ans": 2, "25-34 ans": 3,
    "35-44 ans": 4, "45-54 ans": 5, "55 ans et plus": 6,
}
df_mapped = df_filtered.copy()
df_mapped["age"] = df_mapped["age"].map(age_map)

# Genre
genre_map = {
    "Homme": 1, "Femme": 2,
    "Ne souhaite pas répondre": 3, "Non binaire": 4,
}
df_mapped["genre"] = df_mapped["genre"].map(genre_map)

# Etude (avec fallback texte libre)
ETUDE_MAPPING = {
    "Secondaire": 1, "Bachelier": 2, "Master": 3, "Doctorat": 4, "Autre": 5,
}
TEXT_MAPPING = {
    "CEB": 1, "BTS": 2, "Bac+2": 2,
    "Licence (Bac+3 en France)": 2, "Promotion social": 5,
}
q3_text = df_mapped["Q3_5_TEXT"].astype(str).str.strip()
df_mapped["etude"] = np.where(
    df_mapped["etude"] == "Autre",
    q3_text.map(TEXT_MAPPING),
    df_mapped["etude"].map(ETUDE_MAPPING),
)
df_mapped = df_mapped.drop(columns=["Q3_5_TEXT"])

# Profession
profession_map = {
    "Étudiant(e)": 1, "Sans emploi": 2,
    "Indépendant(e)": 3, "Salarié(e)": 4, "Retraité(e)": 5,
}
df_mapped["profession"] = df_mapped["profession"].map(profession_map)

# Salaire
salaire_map = {
    "Moins de 1 000€": 1, "1 000 - 1 999€": 2,
    "2 000 - 2 999€": 3, "3 000 - 3 999€": 4, "4 000€ et plus": 5,
}

```

```

}
df_mapped["salaire"] = df_mapped["salaire"].map(salaire_map)

# Investir
df_mapped["investir"] = df_mapped["investir"].map({"Non": 0, "Oui": 1})

```

Création des colonnes binaires « asset »

```

CATEGORIES = [
    "Actions", "ETFs et fonds d'investissements",
    "Obligations", "Immobilier", "Crypto-actifs", "Autre",
]
for category in CATEGORIES:
    safe_name = category.replace("'", '').replace(' ', '_').lower()
    col_name = f'asset_{safe_name}'
    contains = df_mapped[SOURCE_COL].fillna('').str.split(',').apply(
        lambda lst: category in [x.strip() for x in lst]
    )
    df_mapped[col_name] = np.where(df_mapped[SOURCE_COL].isna(),
                                   np.nan, contains.astype(int))

```

Encodage des questions financières (Q8, Q9, Q10)

```

# Q8 – Taux d'intérêt
q8_map = {
    "Moins de 102€": 1, "Environ 102€": 2,
    "Environ 110€": 3, "Plus de 110€": 4,
}
df_mapped["Q8"] = df_mapped["Q8"].where(df_mapped["Q8"] != "Je ne sais
pas").map(q8_map)

# Q9 – Inflation
q9_map = {
    "Moins que aujourd'hui": 1,
    "Autant que aujourd'hui": 2,
    "Plus que aujourd'hui": 3,
}
df_mapped["Q9"] = df_mapped["Q9"].where(df_mapped["Q9"] != "Je ne sais
pas").map(q9_map)

# Q10 – Diversification
df_mapped["Q10"] = df_mapped["Q10"].map({"Faux": 0, "Vrai": 1})

```

Encodage des échelles de Likert

```

LIKERT_MAPPING = {
    "Pas du tout d'accord": 1, "Plutôt pas d'accord": 2,
    "Ni d'accord ni pas d'accord": 3,
    "Plutôt d'accord": 4, "Tout à fait d'accord": 5,
}
TYPO_FIXES = {
    "Plutot d'accord": "Plutôt d'accord",
    "Plutot pas d'accord": "Plutôt pas d'accord",
}
for c in df_mapped.columns:
    if df_mapped[c].dtype == object:
        df_mapped[c] = df_mapped[c].replace(TYPO_FIXES)

for c in df_mapped.columns:

```

```

if df_mapped[c].dtype == object:
    vals = set(df_mapped[c].dropna().unique())
    if vals and vals.issubset(set(LIKERT_MAPPING.keys())):
        df_mapped[c] = df_mapped[c].map(LIKERT_MAPPING)

```

Normalisation des colonnes Fonds B / Fonds C

```

for c in ['Fond B', 'Fond C']:
    flag_col = c.replace(' ', '_') + '_flag'
    clean_col = c.replace(' ', '_') + '_pct'
    tmp = (df_mapped[c].astype(str)
           .str.replace('%', '', regex=False)
           .str.replace(',', '.', regex=False)
           .str.strip())
    num = pd.to_numeric(tmp, errors='coerce')
    is_strict = tmp.str.match(r'^\d+(\.\d+)?$', na=False)
    flag = pd.Series('valid', index=df_mapped.index, dtype=object)
    flag[df_mapped[c].isna()] = 'missing'
    flag[~is_strict & ~df_mapped[c].isna()] = 'non_numeric'
    flag[(num > 10).fillna(False)] = 'above_10pct'
    df_mapped[flag_col] = flag
    df_mapped[clean_col] = num

df_final = df_mapped[
    (df_mapped['Fond_B_flag'] == 'valid') &
    (df_mapped['Fond_C_flag'] == 'valid')
].reset_index(drop=True)

```

Sauvegarde du fichier final

```

df_final.to_csv(OUTPUT_PATH, index=False)
print(f'Saved {df_final.shape[0]} rows x {df_final.shape[1]} cols ->
{OUTPUT_PATH}')

```

Annexe 2 - Code R

Chargement des packages

```
library(writexl)
library(openxlsx)
library(readxl)
library(psych)
library(car)
library(Hmisc)
library(MASS)
library(psc1)
library(lmtest)
```

Ajout des variables dummies

```
# Chargement des données originales
DATASET <- read_excel("Desktop/DATA_mémoire_version197.xlsx")

# Recodage de gender (0 = homme référence, 1 = femme)
DATASET$gender <- ifelse(DATASET$gender == 2, 1, 0)

# Dummies pour study (référence = secondaire)
DATASET$d_bachelier <- ifelse(DATASET$study == 2, 1, 0)
DATASET$d_master <- ifelse(DATASET$study == 3, 1, 0)
DATASET$d_doctorat <- ifelse(DATASET$study == 4, 1, 0)

# Dummies pour profession (référence = salarié)
DATASET$d_inactif <- ifelse(DATASET$profession %in% c(2, 5), 1, 0)
DATASET$d_independant <- ifelse(DATASET$profession == 3, 1, 0)
DATASET$d_etudiant <- ifelse(DATASET$profession == 4, 1, 0)

# Dummy pour risk (coupure à la médiane)
mediane_risk <- median(DATASET$risk, na.rm = TRUE)
DATASET$d_risk <- ifelse(DATASET$risk > mediane_risk, 1, 0)

# Sauvegarde
write_xlsx(DATASET, "Desktop/DATASET.xlsx")

# Chargement du DATASET final
DATASET <- read_excel("Desktop/DATASET.xlsx")
str(DATASET)
```

Test de normalité

```
variables_normalite <- c("prime_B", "prime_C", "intention",
                        "important_durability", "risk",
                        "security", "power", "freedom", "love")
for (var in variables_normalite) {
  test <- shapiro.test(DATASET[[var]])
  cat(var, ": W =", round(test$statistic, 4),
      ", p =", round(test$p.value, 4), "\n")
}
```

Fiabilité des échelles

```
# Security
print(alpha(DATASET[, c("SECURITY1", "SECURITY2", "SECURITY3",
"SECURITY4")])$total$raw_alpha)
# Freedom
print(alpha(DATASET[, c("FREEDOM1", "FREEDOM2", "FREEDOM3",
"FREEDOM4")])$total$raw_alpha)
# Power
print(alpha(DATASET[, c("POWER1", "POWER2", "POWER3", "POWER4", "POWER5",
"POWER6")])$total$raw_alpha)
# Love
print(alpha(DATASET[, c("LOVE1", "LOVE2")])$total$raw_alpha)
```

Test de multicolinéarité

```
model_vif <- lm(prime_B ~ gender + age + salary +
                d_bachelier + d_master + d_doctorat +
                d_inactif + d_independant + d_etudiant +
                Score_litteracy + invest +
                security + power + freedom + love,
                data = DATASET)
vif(model_vif)
```

Matrice de corrélation de Spearman

```
vars <- DATASET[, c("prime_B", "prime_C", "intention",
                    "important_durability", "d_risk",
                    "security", "power", "freedom", "love",
                    "gender", "age", "salary",
                    "d_bachelier", "d_master", "d_doctorat",
                    "d_inactif", "d_independant", "d_etudiant",
                    "Score_litteracy", "invest")]
matrice <- rcorr(as.matrix(vars), type = "spearman")

n <- ncol(vars)
var_names <- colnames(vars)
matrice_finale <- matrix("", nrow=n, ncol=n)
rownames(matrice_finale) <- var_names
colnames(matrice_finale) <- var_names

for(i in 1:n){
  for(j in 1:n){
    if(i == j){
      matrice_finale[i,j] <- "-"
    } else if(i > j){
      r <- round(matrice$r[i,j], 4)
      p <- matrice$p[i,j]
      if(p < 0.01){
        matrice_finale[i,j] <- paste0(r, "****")
      } else if(p < 0.05){
        matrice_finale[i,j] <- paste0(r, "***")
      } else if(p < 0.10){
        matrice_finale[i,j] <- paste0(r, "**")
      } else {
        matrice_finale[i,j] <- as.character(r)
      }
    }
  }
}
print(matrice_finale)
```

```
# Sauvegarde en Excel
matrice_df <- as.data.frame(matrice_finale)
matrice_df <- cbind(Variable = rownames(matrice_finale), matrice_df)
write_xlsx(matrice_df, "Desktop/matrice_correlation.xlsx")
```

Régressions OLS

```
controls <- "gender + age + salary + d_bachelier + d_master + d_doctorat +
            d_inactif + d_independant + d_etudiant + Score_litteracy + invest +
            risk"
controls_no_risk <- "gender + age + salary + d_bachelier + d_master + d_doctorat +
                    d_inactif + d_independant + d_etudiant + Score_litteracy +
                    invest"
emotions <- "security + power + freedom + love"

# VD1 : prime_B
modell1_PRIMEB <- lm(as.formula(paste("prime_B ~", controls)), data = DATASET)
modell2_PRIMEB <- lm(as.formula(paste("prime_B ~", controls, "+", emotions)), data =
DATASET)
summary(modell1_PRIMEB)
summary(modell2_PRIMEB)
cat("ΔR² PRIME_B:", summary(modell2_PRIMEB)$r.squared -
summary(modell1_PRIMEB)$r.squared, "\n")

# VD2 : prime_C
modell1_PRIMEC <- lm(as.formula(paste("prime_C ~", controls)), data = DATASET)
modell2_PRIMEC <- lm(as.formula(paste("prime_C ~", controls, "+", emotions)), data =
DATASET)
summary(modell1_PRIMEC)
summary(modell2_PRIMEC)
cat("ΔR² PRIME_C:", summary(modell2_PRIMEC)$r.squared -
summary(modell1_PRIMEC)$r.squared, "\n")

# VD3 : risk
modell1_RISK_OLS <- lm(as.formula(paste("risk ~", controls_no_risk)), data =
DATASET)
modell2_RISK_OLS <- lm(as.formula(paste("risk ~", controls_no_risk, "+", emotions)),
data = DATASET)
summary(modell1_RISK_OLS)
summary(modell2_RISK_OLS)
cat("ΔR² RISK_OLS:", summary(modell2_RISK_OLS)$r.squared -
summary(modell1_RISK_OLS)$r.squared, "\n")
```

Régressions Logit

```
# Création des variables binaires
DATASET$d_primeB <- ifelse(DATASET$prime_B > 0, 1, 0)
DATASET$d_primeC <- ifelse(DATASET$prime_C > 0, 1, 0)

# VD1 : d_primeB (annexe)
modell1_PRIMEB_logit <- glm(as.formula(paste("d_primeB ~", controls)),
                           data = DATASET, family = binomial)
modell2_PRIMEB_logit <- glm(as.formula(paste("d_primeB ~", controls, "+",
emotions)),
                           data = DATASET, family = binomial)
summary(modell1_PRIMEB_logit)
summary(modell2_PRIMEB_logit)
cat("Pseudo R² modell1_PRIMEB:", pR2(modell1_PRIMEB_logit)["McFadden"], "\n")
cat("Pseudo R² modell2_PRIMEB:", pR2(modell2_PRIMEB_logit)["McFadden"], "\n")
```

```

cat("LR test PRIMEB:\n")
print(lrtest(model1_PRIMEB_logit, model2_PRIMEB_logit))

# VD2 : d_primeC (annexe)
model1_PRIMEC_logit <- glm(as.formula(paste("d_primeC ~", controls)),
                           data = DATASET, family = binomial)
model2_PRIMEC_logit <- glm(as.formula(paste("d_primeC ~", controls, "+",
                                             emotions)),
                           data = DATASET, family = binomial)
summary(model1_PRIMEC_logit)
summary(model2_PRIMEC_logit)
cat("Pseudo R² model1_PRIMEC:", pR2(model1_PRIMEC_logit)["McFadden"], "\n")
cat("Pseudo R² model2_PRIMEC:", pR2(model2_PRIMEC_logit)["McFadden"], "\n")
cat("LR test PRIMEC:\n")
print(lrtest(model1_PRIMEC_logit, model2_PRIMEC_logit))

# VD3 : d_risk (modèle principal)
model1_RISK_logit <- glm(as.formula(paste("d_risk ~", controls_no_risk)),
                        data = DATASET, family = binomial)
model2_RISK_logit <- glm(as.formula(paste("d_risk ~", controls_no_risk, "+",
                                           emotions)),
                        data = DATASET, family = binomial)
summary(model1_RISK_logit)
summary(model2_RISK_logit)
cat("Pseudo R² model1_RISK:", pR2(model1_RISK_logit)["McFadden"], "\n")
cat("Pseudo R² model2_RISK:", pR2(model2_RISK_logit)["McFadden"], "\n")
print(lrtest(model1_RISK_logit, model2_RISK_logit))

```

Régressions Ordinales

```

# Convertir en facteurs ordonnés
DATASET$intention_ord <- factor(DATASET$intention, ordered = TRUE)
DATASET$simp_durability_ord <- factor(DATASET$important_durability, ordered = TRUE)

# VD4 : intention
model1_INTENTION <- polr(as.formula(paste("intention_ord ~", controls)),
                        data = DATASET, method = "logistic", Hess = TRUE)
model2_INTENTION <- polr(as.formula(paste("intention_ord ~", controls, "+",
                                           emotions)),
                        data = DATASET, method = "logistic", Hess = TRUE)
coeftest(model1_INTENTION)
coeftest(model2_INTENTION)
pR2(model1_INTENTION) ["McFadden"]
pR2(model2_INTENTION) ["McFadden"]
AIC(model1_INTENTION)
AIC(model2_INTENTION)
lrtest(model1_INTENTION, model2_INTENTION)

# VD5 : important_durability
model1_IMP <- polr(as.formula(paste("imp_durability_ord ~", controls)),
                  data = DATASET, method = "logistic", Hess = TRUE)
model2_IMP <- polr(as.formula(paste("imp_durability_ord ~", controls, "+",
                                     emotions)),
                  data = DATASET, method = "logistic", Hess = TRUE)
coeftest(model1_IMP)
coeftest(model2_IMP)
pR2(model1_IMP) ["McFadden"]
pR2(model2_IMP) ["McFadden"]
AIC(model1_IMP)
AIC(model2_IMP)
lrtest(model1_IMP, model2_IMP)

```

