Manuel de Révision: Cas Spéciaux

Guide Express avec Mnémotechniques

Durée de révision: 30-45 minutes Objectif: Maîtriser les 4 sujets critiques

---

FICHE 1: VALEURS MANQUANTES

Mnémotechnique: "MCAR-MAR-MNAR = Mes Maux d'Analyse Rêvent"

Type	Signification	Exemple	Solution
MCAR	Manquant Complètement Aléatoire	Erreur de saisie	Imputation simple
MAR	Manquant Aléatoire	Jeunes ne déclarent pas patrimoine	Imputation par groupe
MNAR	Manquant Non Aléatoire	Riches cachent leur revenu	Modèle spécialisé

Règle des Seuils

```
< 5% manquants → SUPPRESSION OK
5-20% manquants → IMPUTATION NÉCESSAIRE
> 20% manquants → SUPPRIMER LA VARIABLE?
```

Formules Clés

Vérifier le type:

```python
# Test MCAR simplifié
for col in df.select_dtypes(include=[np.number]):
    mask = df[col].isnull()
    for other in df.columns:
        if other != col:
            stat, p = stats.mannwhitneyu(df[mask][other], df[~mask][other])
            if p < 0.05:
                print(f"{col} est MAR (lié à {other})")
```

Code Express

```python
# 1. Diagnostic
df.isnull().sum()  # Combien par colonne
df.isnull().mean() * 100  # Pourcentage

# 2. Imputation médiane (robuste)
df['col'].fillna(df['col'].median(), inplace=True)

# 3. Imputation par groupe
df['revenu'] = df.groupby('type_emploi')['revenu'].transform(
    lambda x: x.fillna(x.median())
)
```

```python
# 4. Indicateur de manquant (pour ML)
df['col_manquant'] = df['col'].isnull().astype(int)

# 5. KNN Imputation
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=5)
df_imputed = imputer.fit_transform(df)
```

Piège à Éviter

JAMAIS fit sur le test set!

```python
# BON
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# MAUVAIS (data leakage!)
scaler.fit(X)  # Tout le dataset
```

FICHE 2: PRÉVISIONS UNIVARIÉE/MULTIVARIÉE

Mnémotechnique: "UBM = Un, Bi, Multi"

Type	Variables	Objectif	Outils
Univariée	1	Distribution	Histogramme, stats
Bivariée	2	Relation	Corrélation, scatter
Multivariée	3+	Modèle	Régression, ACP

Tests par Situation (Mnémo: "2NC")

```
2 Numériques → Corrélation (Pearson/Spearman)
Numérique + Catégorielle → t-test/ANOVA
2 Catégorielles → Chi-carré
```

Corrélation de Pearson

```python
# Calcul
r, p_value = stats.pearsonr(x, y)

# Interprétation
# |r| < 0.3 → Faible
# 0.3 ≤ |r| < 0.7 → Modérée
# |r| ≥ 0.7 → Forte
```

Code Express Prévision

```python
# 1. Analyse univariée
df['col'].describe()
df['col'].hist()
```

```python
df['col'].skew()  # > 0 = queue droite

# 2. Analyse bivariée
# Corrélation
df[['col1', 'col2']].corr()

# t-test (2 groupes)
from scipy.stats import ttest_ind
stat, p = ttest_ind(groupe1, groupe2)

# ANOVA (3+ groupes)
from scipy.stats import f_oneway
stat, p = f_oneway(groupe1, groupe2, groupe3)

# 3. Régression multiple
import statsmodels.api as sm
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
print(model.summary())

# 4. Séries temporelles (Holt-Winters)
from statsmodels.tsa.holtwinters import ExponentialSmoothing
model = ExponentialSmoothing(serie, seasonal_periods=12,
                             trend='add', seasonal='add')
forecast = model.fit().forecast(12)
```

Diagnostics Régression

```
1. R² → Variance expliquée (0-1)
2. p-values coefficients → Significativité (< 0.05)
3. Résidus → Normalité (QQ-plot)
4. VIF → Multicolinéarité (< 5)
```

FICHE 3: ACP et ANOVA

ANOVA - Mnémo: “FEP”

- **F** = F-statistic (plus grand = plus de différence)
- **E** = Eta² (taille d’effet)
- **P** = Post-hoc (si significatif)

Conditions ANOVA: “NIH”

- **N**ormalité des distributions
- **I**ndépendance des observations
- **H**omogénéité des variances (Levene)

Code ANOVA Express

```python
from scipy import stats

# 1. Vérifier homogénéité (Levene)
```

```python
stat, p = stats.levene(groupe1, groupe2, groupe3)
# p > 0.05 → variances homogènes

# 2. ANOVA à un facteur
stat, p = stats.f_oneway(groupe1, groupe2, groupe3)
# p < 0.05 → différence significative

# 3. Eta-squared (taille d'effet)
ss_between = sum(len(g) * (g.mean() - grand_mean)**2 for g in groups)
ss_total = sum((df['var'] - df['var'].mean())**2)
eta_sq = ss_between / ss_total
# < 0.01 négligeable, < 0.06 petit, < 0.14 moyen, ≥ 0.14 grand

# 4. Post-hoc (si p < 0.05)
from scipy.stats import tukey_hsd
result = tukey_hsd(groupe1, groupe2, groupe3)
```

ACP - Mnémo: “SCALE”

- **S**tandardiser (OBLIGATOIRE)
- **C**orrélation à vérifier
- **A**nalyser les loadings
- **L**ooking at scree plot
- **E**xpliquer les composantes

Critères de Sélection des Composantes

```
KAISER: λ > 1 (valeurs propres)
COUDE: Point d'inflexion du scree plot
80%: Variance cumulée ≥ 80%
```

Code ACP Express

```python
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# 1. TOUJOURS standardiser d'abord!
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 2. ACP
pca = PCA()
X_pca = pca.fit_transform(X_scaled)

# 3. Variance expliquée
print(pca.explained_variance_ratio_)
print(np.cumsum(pca.explained_variance_ratio_))

# 4. Nombre de composantes (Kaiser)
n_kaiser = (pca.explained_variance_ > 1).sum()

# 5. Loadings
loadings = pd.DataFrame(pca.components_.T,
```

```
                    columns=[f'PC{i+1}' for i in range(len(pca.components_))],
                    index=colonnes)
```

Interprétation des Loadings

```
Loading > 0.4 → Contribution POSITIVE forte
Loading < -0.4 → Contribution NÉGATIVE forte
|Loading| < 0.3 → Contribution faible
```

FICHE 4: OUTLIERS

Mnémo: "IQR × 1.5"

```
Borne inférieure = Q1 - 1.5 × IQR
Borne supérieure = Q3 + 1.5 × IQR

où IQR = Q3 - Q1
```

Méthodes de Détection

Méthode	Formule	Quand utiliser
IQR	Q1/Q3 ± 1.5×IQR	Données asymétriques
Z-Score	|z| > 3	Données normales
MAD	|z_mod| > 3.5	Petits échantillons
Isolation Forest	score < seuil	Multivarié, fraude

Code Détection Express

```python
# 1. Méthode IQR
Q1 = df['col'].quantile(0.25)
Q3 = df['col'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR
outliers = df[(df['col'] < lower) | (df['col'] > upper)]

# 2. Méthode Z-Score
from scipy import stats
z_scores = np.abs(stats.zscore(df['col']))
outliers = df[z_scores > 3]

# 3. Isolation Forest (multivarié)
from sklearn.ensemble import IsolationForest
iso = IsolationForest(contamination=0.05)
df['outlier'] = iso.fit_predict(X)  # -1 = outlier
```

Traitement - Mnémo: "WISE"

- **W**insorize (capper aux percentiles)
- **I**mputer (remplacer par médiane)

- **S**upprimer (si erreur confirmée)
- **E**xclure (garder pour analyse séparée)

Code Traitement Express

```
# Winsorize (capping)
p5 = df['col'].quantile(0.05)
p95 = df['col'].quantile(0.95)
df['col_capped'] = df['col'].clip(lower=p5, upper=p95)

# Imputation par médiane
median = df.loc[~mask_outliers, 'col'].median()
df.loc[mask_outliers, 'col'] = median

# Flag (garder info)
df['col_outlier'] = ((df['col'] < lower) | (df['col'] > upper)).astype(int)
```

TABLEAU RÉCAPITULATIF: QUOI UTILISER QUAND?

Par Situation

Problème	Solution	Code Clé
Manquants < 5%	Supprimer lignes	`df.dropna()`
Manquants 5-20%	Imputer médiane	`fillna(median())`
Manquants MAR	Imputer par groupe	`groupby().transform()`
Comparer 2 groupes	t-test	`ttest_ind()`
Comparer 3+ groupes	ANOVA	`f_oneway()`
Réduire dimensions	ACP	`PCA()`
Détecter anomalies	Isolation Forest	`IsolationForest()`
Outliers univariés	IQR	`quantile()`

Par Type de Variable

Variable	Outliers	Manquants	Test
Continue normale	Z-score	Moyenne	t-test
Continue asymétrique	IQR	Médiane	Mann-Whitney
Catégorielle	N/A	Mode	Chi-carré
Ordinale	IQR	Médiane	Kruskal-Wallis

CHECKLIST JOUR DE L'EXAMEN

Valeurs Manquantes

```
□ MCAR/MAR/MNAR → Définir le type
□ < 5% → Suppression OK
□ fit sur TRAIN seulement → Éviter leakage
□ Créer indicateur → Si info importante
```

Prévisions

□ Univariée → 1 variable, stats descriptives
□ Bivariée → 2 variables, corrélation ou test
□ 2 groupes → t-test
□ 3+ groupes → ANOVA
□ 2 catégorielles → Chi-carré

ACP/ANOVA

□ ANOVA → p < 0.05 = différence significative
□ Eta² → taille d'effet (0.14 = grand)
□ ACP → TOUJOURS standardiser
□ Kaiser → λ > 1
□ 80% variance → nombre de composantes

Outliers

□ IQR → Q1/Q3 ± 1.5 × IQR
□ Z-score → |z| > 3
□ JAMAIS supprimer aveuglément
□ Vérifier si fraude possible
□ Documenter les choix

FORMULES À RETENIR

Statistiques

Moyenne: $\bar{x} = \Sigma x_i / n$
Variance: $s^2 = \Sigma(x_i - \bar{x})^2 / (n-1)$
Écart-type: $s = \sqrt{s^2}$
Z-score: $z = (x - \mu) / \sigma$
IQR: Q3 - Q1

Tests

t-test: $t = (\bar{x}_1 - \bar{x}_2) / \sqrt{(s_1^2/n_1 + s_2^2/n_2)}$
Chi²: $\Sigma(O-E)^2 / E$
Corrélation: $r = \Sigma(x_i-\bar{x})(y_i-\bar{y}) / \sqrt{[\Sigma(x_i-\bar{x})^2\Sigma(y_i-\bar{y})^2]}$

ACP

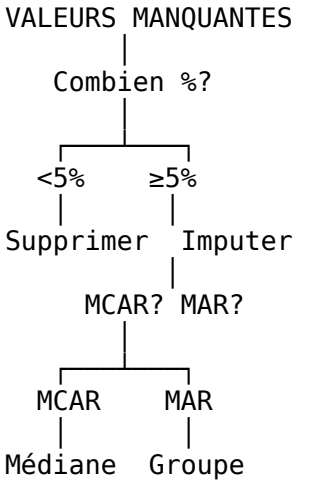
Variance expliquée = $\lambda_i / \Sigma\lambda$
Kaiser: garder si $\lambda > 1$
Gini (scoring): 2 × AUC - 1

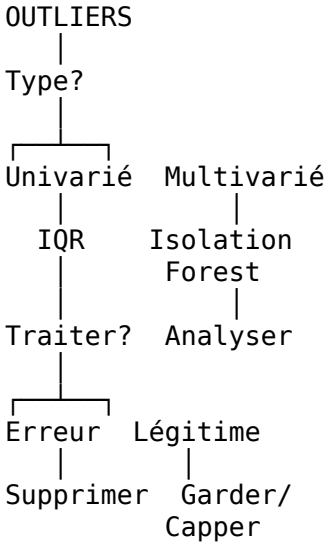
ERREURS FRÉQUENTES À ÉVITER

Erreur	Conséquence	Solution
Fit sur tout le dataset	Data leakage	Fit sur train seulement
Supprimer tous les outliers	Perte d'info	Analyser d'abord
ACP sans standardiser	Résultats biaisés	StandardScaler()
Ignorer le type de manquant	Biais non détecté	Tester MCAR/MAR
ANOVA sans vérifier conditions	Résultats invalides	Test Levene, normalité
Confondre corrélation/causalité	Conclusions fausses	Rester prudent

---

AIDE-MÉMOIRE VISUEL





CONFIANCE! VOUS MAÎTRISEZ LES CAS SPÉCIAUX!