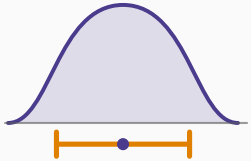
Statistique inférentielle

Chapitre 22 : Mise en œuvre sur R et Python



Jaouad Madkour

Faculté d'Économie et de Gestion de Tanger · 27 août 2026

Plan du chapitre

Le programme

Refaire les vingt et un chapitres

Ce que ce chapitre reprend

Ce qu'on refait	Chapitres
Distributions d'échantillonnage, par simulation	3 à 5
Intervalles de confiance	9 à 12
Tests de moyenne et de proportion	16
Comparaisons, ANOVA, khi-deux	17 à 19
Régression et test de coefficient	20, 21

Le but n'est pas d'apprendre un logiciel

C'est de **vérifier** que chaque nombre calculé à la main est bien celui qu'affiche la machine — et de voir qu'une fonction en remplace une page.

Comprendre la formule reste indispensable : c'est ce qui permet de choisir la bonne fonction et de reconnaître une sortie aberrante.

Sur R

Charger et tirer l'échantillon

```r
sal <- read.csv("salaires.csv", stringsAsFactors = TRUE)

# La population
mean(sal$salaire)                          # 7158.26
sd(sal$salaire) * sqrt(499/500)            # 2598.10  (ecart-type de population)

# L'echantillon fil rouge : un identifiant sur dix
ech <- subset(sal, id %% 10 == 0)
nrow(ech)                                  # 50
mean(ech$salaire)                          # 7005.80
sd(ech$salaire)                            # 2250.98
```

Deux écarts-types, deux touches

`sd()` divise par $n - 1$: c'est l'estimateur du chapitre 4.

Pour l'écart-type d'une population complète, il faut multiplier par $\sqrt{(n-1)/n}$. R ne propose pas la seconde formule, parce qu'elle ne sert qu'en statistique descriptive.

```r
t.test(ech$salaire, mu = 6500, alternative = "greater")

#   t = 1.5889, df = 49, p-value = 0.05933
#   95 percent confidence interval:  6470.05    Inf
#   sample estimates: mean of x  7005.80

t.test(ech$salaire)$conf.int                # 6365.94  7645.66
```

Une seule fonction fait tout

`t.test()` donne la statistique, les degrés de liberté, la p-value, l'intervalle et l'estimation — les cinq étapes du chapitre 13 en une ligne.

`alternative` vaut `"two.sided"` par défaut, `"greater"` ou `"less"` pour un test unilatéral.

```r
# Proportion : 13 Bac+5 sur 50, contre 15 % annonces
prop.test(13, 50, p = 0.15, correct = FALSE)      # X-squared = 4.745, p = 0.0294

# Deux groupes
t.test(salaire ~ sexe, data = sal)                # t = -0.0012, p = 0.9991

# Trois groupes
summary(aov(salaire ~ diplome, data = sal))       # F = 86.28, p < 2e-16

# Independance de deux caracteres
chisq.test(table(sal$diplome, sal$sexe))          # X-squared = 0.2292, df = 2, p = 0
```

Un détail qui change le résultat

`prop.test()` applique par défaut une correction de continuité. Avec `correct = FALSE`, on retrouve exactement le $z^2 = 2{,}178^2 = 4{,}745$ du chapitre 16.

Les deux sont défendables ; ce qui ne l'est pas, c'est de ne pas savoir laquelle on emploie.

```r
mod <- lm(salaire ~ anciennete, data = sal)
summary(mod)
confint(mod)

# Coefficients:
#              Estimate  Std. Error  t value  Pr(>|t|)
# (Intercept)  4347.23     177.18     24.54   < 2e-16
# anciennete    277.17      15.06     18.41   < 2e-16
#
# Residual standard error: 2008 on 498 degrees of freedom
# Multiple R-squared: 0.4049,  F-statistic: 338.8 on 1 and 498 DF
```

Chaque nombre a été calculé à la main

$277{,}17$ au chapitre 20, $15{,}06$ et $18{,}41$ au chapitre 21, $2\,008$ et $0{,}4049$ également.

Le logiciel n'ajoute rien : il calcule plus vite.

Sur Python

Les mêmes opérations

```
import pandas as pd, numpy as np
from scipy import stats

sal = pd.read_csv("salaires.csv")
ech = sal[sal.id % 10 == 0]

ech.salaire.mean()                    # 7005.80
ech.salaire.std()                     # 2250.98   (ddof=1 par defaut)
```

Les mêmes opérations (suite)

```
stats.ttest_1samp(ech.salaire, 6500, alternative="greater")
#   statistic=1.5889, pvalue=0.0593

stats.ttest_ind(sal[sal.sexe=="H"].salaire,
                sal[sal.sexe=="F"].salaire, equal_var=False)
#   statistic=-0.0012, pvalue=0.9991
```

Deux conventions opposées, à connaître

`pandas.std()` divise par $n-1$; `numpy.std()` divise par n.

Sur cinq cents observations l'écart est invisible ; sur dix il est de $5\,\%$. **C'est la source d'erreur la plus fréquente du passage de R à Python.**

```python
import statsmodels.formula.api as smf

# ANOVA
groupes = [g.salaire.values for _, g in sal.groupby("diplome")]
stats.f_oneway(*groupes)              # F=86.28, p=3.4e-33

# Independance
stats.chi2_contingency(pd.crosstab(sal.diplome, sal.sexe))
#   chi2=0.2292, dof=2, p=0.8917

# Regression
mod = smf.ols("salaire ~ anciennete", data=sal).fit()
print(mod.summary())
mod.conf_int()                        # 247.65   306.68
```

La correspondance des fonctions

Vérifier les théorèmes

La distribution d'échantillonnage, par simulation

```r
set.seed(2024)
moyennes <- replicate(20000, mean(sample(sal$salaire, 40)))

mean(moyennes)      # 7159.8   ->  m = 7158.26
sd(moyennes)        # 392.6    ->  sigma/sqrt(40) = 410.8
                    #              avec correction finie : 394.4

hist(moyennes, breaks = 50, freq = FALSE)
curve(dnorm(x, 7158.26, 394.42), add = TRUE, col = "red", lwd = 2)
```

Trois vérifications d'un coup

La moyenne des moyennes tombe sur m ; leur écart-type sur $\sigma/\sqrt{n}$ corrigé ; et l'histogramme épouse la cloche, alors que les salaires ne sont pas normaux.

Cinq lignes reproduisent tout le chapitre 3.

La couverture des intervalles

```r
set.seed(7)
couvre <- replicate(10000, {
  e  <- sample(sal$salaire, 40)
  ic <- t.test(e)$conf.int
  ic[1] <= 7158.26 & 7158.26 <= ic[2]
})
mean(couvre)        # 0.9503
```

La démonstration la plus convaincante du cours

Sur dix mille échantillons, $95{,}03\,\%$ des intervalles contiennent la vraie moyenne.

C'est très exactement ce que promettait le chapitre 9 — et c'est une chose de le lire, une autre de le vérifier sur sa propre machine. **Faites tourner ce bloc : il vaut trois heures d'explication.**

Ce qu’il faut retenir

Le résumé du chapitre

Six points

1. `t.test`, `prop.test`, `aov`, `chisq.test`, `lm` : cinq fonctions couvrent tout le cours sous R.
2. `t.test()` donne à la fois le test **et** l'intervalle — les deux faces du chapitre 16.
3. `prop.test(correct = FALSE)` retrouve exactement le z calculé à la main.
4. Attention aux conventions : `pandas.std()` divise par $n - 1$, `numpy.std()` par n.
5. Toute la sortie de `lm` a été calculée à la main aux chapitres 20 et 21.
6. Quelques lignes de simulation **vérifient** la distribution d'échantillonnage et la couverture des intervalles.

La fin du parcours

Nous sommes partis de cinq cents salaires que l'on se contentait de décrire. Nous arrivons à un modèle qui les explique, avec la mesure de sa fiabilité.

Entre les deux, une seule idée aura tout porté : **une statistique calculée sur un échantillon est une variable aléatoire, et l'on connaît sa loi**. Estimer, encadrer, tester, régresser n'en sont que quatre usages.

L'introduction à l'économétrie prend la suite immédiate : mêmes outils, plusieurs variables explicatives, et l'examen systématique des hypothèses du chapitre 20.