

Statistique descriptive

Chapitre 15 : Mise en œuvre sur Python

Jaouad Madkour

12 octobre 2026

Plan du chapitre

La même chaîne, un autre outil	2
Ce qui change, ce qui ne change pas	2
Le seul vrai piège du chapitre	3
Charger et explorer	3
L'import	3
La sortie de describe()	3
Les variables qualitatives	4
Qui divise par quoi	4
Le contrôle des types	4
Un salaire lu comme du texte	5
Le tableau statistique	5
Variable qualitative	5
Trier l'index n'est pas décoratif	6
Variable continue	6
Le tableau de référence, en Python	6
Sur données groupées	7
Les graphiques	7
Les quatre représentations	7
Ce que Python affiche	8
L'argument density=True	8
Les boîtes à moustaches	8
Les indicateurs univariés	9
Tendance centrale, dispersion, forme	9
Le résultat	9
Une même forme, plusieurs réponses	10
Concentration	10
La courbe et l'aire de concentration	11
Le bivarié	11
Contingence	11

Les profils-lignes	11
Les effectifs théoriques	12
Conditionnelles	12
Décomposition de la variance	12
Covariance, corrélation, ajustement	13
La version statsmodels	13
La sortie de summary()	13
Le graphique final	14
Le nuage et sa droite	14
Comparaison des deux langages	15
Importer et résumer	15
Les indicateurs	15
Le bivarié	15
S'entraîner	16
Exercice 1 : trois écarts-types pour une série	16
Exercice 1 : solution	16
Exercice 2 : les quantiles groupés à la main	16
Exercice 2 : solution	17
Exercice 3 : traduire un script R	17
Exercice 3 : solution	17
Ce qu'il faut retenir	18
Le résumé du chapitre	18
Bilan des seize chapitres	18
La phrase qui résume le semestre	18
Et maintenant	19

La même chaîne, un autre outil

Ce qui change, ce qui ne change pas

Les quatre bibliothèques

Bibliothèque	Rôle
pandas	tableaux de données, tableaux croisés, groupes
numpy	calcul numérique
matplotlib	graphiques
scipy.stats	coefficients de forme

```
import numpy as np, pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
```

Ce qui ne change pas

Les **nombre**s. Le chapitre 14 a retrouvé en R chaque résultat du cours ; Python retrouve les mêmes — à condition d'éviter les mêmes pièges de convention, plus un.

Le seul vrai piège du chapitre

Deux bibliothèques, deux conventions opposées

Là où R divise par $n - 1$ **partout**, pandas et numpy ne font **pas le même choix** : numpy divise par n par défaut, pandas par $n - 1$.

```
v = np.array([2, 3, 4, 5, 6, 8, 10, 12, 15, 20]) # dix anciennetés
np.std(v)                                     # 5.4818    numpy : divise par n
pd.Series(v).std()                           # 5.7783    pandas : divise par n - 1
pd.Series(v).std(ddof=0)                     # 5.4818    divise par n - 0
```

ddof : delta degrees of freedom

Le diviseur est $n - \text{ddof}$. ddof vaut 0 par défaut dans numpy et 1 dans pandas. **Le préciser explicitement est la seule façon de ne jamais se tromper.**

Charger et explorer

L'import

```
sal = pd.read_csv("salaires.csv")
sal.info()                                     # types, manquants
sal[["salaire", "anciennete"]].describe()     # quantitatives
sal[["diplome", "region", "sexe"]].describe() # qualitatives
```

Un DataFrame, des Series

sal est un **DataFrame** : un tableau dont chaque colonne est une **Series**. `sal.salaire` (ou `sal["salaire"]`) extrait une colonne, et toutes les méthodes statistiques — `mean`, `std`, `quantile` — s'appliquent à une colonne comme à un tableau entier.

La sortie de describe()

Les deux variables quantitatives

	salaire	anciennete
count	500.000000	500.000000
mean	7158.260000	10.142000
std	2600.701688	5.970354
min	2810.000000	0.000000
25%	5317.500000	6.000000
50%	6770.000000	9.000000
75%	8330.000000	13.000000
max	19050.000000	35.000000

std vaut 2 600,70, pas 2 598,10

pandas divise par n — 1. Pour l'écart-type **descriptif** du cours, il faut `sal.salaire.std(ddof=0)`. Les quartiles, eux, sont ceux du chapitre 6 : rang $1 + p(n - 1)$, interpolé, comme dans R.

Les variables qualitatives

La sortie de `describe()` sur les trois qualitatives

	diplome	region	sexe
count	500	500	500
unique	3	4	2
top	Bac	Tanger-Tetouan	H
freq	219	180	285

Ce que disent unique, top et freq

unique compte les modalités ; **top** donne la modalité la plus fréquente — le **mode** du chapitre 5 — et **freq** son effectif. Le salarié type est un bachelier (219 sur 500), un homme (285), à Tanger-Tétouan (180). Pour le tableau complet, il faut `value_counts()`.

Qui divise par quoi

Les fonctions de dispersion, et leur diviseur

Fonction	Divise par	Pour le descriptif
<code>np.std(v)</code> , <code>np.var(v)</code>	n	tel quel
<code>s.std()</code> , <code>s.var()</code> (pandas)	$n - 1$	<code>ddof=0</code>
<code>np.cov(a, b)</code>	$n - 1$	<code>ddof=0</code>
<code>np.corrcoef(a, b)</code> , <code>s.corr()</code>	—	rapport : rien à faire

Pourquoi deux conventions

numpy est une bibliothèque de **calcul** : elle renvoie la moyenne des carrés des écarts, la définition. pandas est une bibliothèque d'**analyse de données**, pensée pour des échantillons : elle renvoie l'estimateur de la statistique inférentielle. Les deux choix se défendent ; le cours, descriptif, suit celui de numpy.

Le contrôle des types

```
sal.isna().sum()      # 0 partout : aucune valeur manquante
sal.salaire.nunique()  # 381 valeurs distinctes : continue
sal.dtypes
```

La sortie de sal.dtypes

```
id          int64
salaire     int64
anciennete  int64
diplome      str
region      str
sexe        str
dtype: object
```

Un salaire lu comme du texte

Le premier réflexe, comme en R

Si `salaire` apparaît comme `str` (ou `object`, dans les versions anciennes de pandas), une ligne contient du texte. **Aucune moyenne n'est alors possible.**

```
# Fichier au format français, avec « n.d. » pour les manquants
sal = pd.read_csv("salaires.csv", sep=";", decimal=".",
                  na_values=["n.d."])
# Texte résiduel transformé en valeur manquante
sal["salaire"] = pd.to_numeric(sal.salaire, errors="coerce")
```

Forcer la conversion

Avec `errors="coerce"`, ce qui ne se lit pas comme un nombre devient NaN. Il faut ensuite **compter** ces manquants avec `isna().sum()` et dire sur combien d'individus portent les calculs.

Le tableau statistique

Variable qualitative

```
n = len(sal)
ni = sal.diplome.value_counts().sort_index()
fi = ni / n
tableau = pd.DataFrame({"ni": ni, "fi": fi,
                        "Ni": ni.cumsum(), "Fi": fi.cumsum()})
print(tableau.round(3))
```

Le résultat : le tableau du chapitre 2

	ni	fi	Ni	Fi
diplome				
Bac	219	0.438	219	0.438
Bac+2	174	0.348	393	0.786
Bac+5	107	0.214	500	1.000

Trier l'index n'est pas décoratif

Le tri par effectif décroissant

`value_counts()` trie par effectif décroissant. Sur une variable **ordinaire**, cela détruit l'ordre — et les cumuls deviennent absurdes. `sort_index()` rétablit l'ordre alphabétique, qui coïncide ici avec l'ordre des diplômes. Ce n'est pas toujours le cas : « Bac+10 » passerait avant « Bac+2 ».

```
sal.region.value_counts() # Tanger-Tetouan 180, Casablanca 137, ...
ordre = ["Bac", "Bac+2", "Bac+5"]
sal["diplome"] = pd.Categorical(sal.diplome, categories=ordre,
                                ordered=True)
```

Le type Categorical ordonné

Il déclare une fois pour toutes l'ordre des modalités : tris, tableaux et graphiques le respectent ensuite. C'est l'équivalent du `factor(..., levels = ordre)` de R.

Variable continue

```
bornes = [2000, 4000, 6000, 8000, 10000, 14000, 20000]
cl = pd.cut(sal.salaire, bins=bornes, right=False) # [a ; b[
nk = cl.value_counts().sort_index()                # effectifs
ai = np.diff(bornes)                               # amplitudes
pd.DataFrame({"ni": nk.values, "fi": nk.values/n,
              "ai": ai, "di": nk.values/ai}, index=nk.index).round(5)
```

right=False, encore

Comme dans R, la convention par défaut est $]a; b]$, et le salarié payé exactement 6 000 dirhams change de classe : 158 et 165 au lieu de 157 et 166. Deux logiciels, deux langages, le même piège au même endroit.

Le tableau de référence, en Python

La sortie

	ni	fi	ai	di
salaire				
[2000, 4000)	31	0.062	2000	0.01550
[4000, 6000)	157	0.314	2000	0.07850
[6000, 8000)	166	0.332	2000	0.08300
[8000, 10000)	79	0.158	2000	0.03950
[10000, 14000)	57	0.114	4000	0.01425
[14000, 20000)	10	0.020	6000	0.00167

La notation des intervalles

pandas écrit `[2000, 4000)` : la parenthèse finale signale, à l'anglo-saxonne, une borne **exclue** — c'est notre crochet tourné vers l'extérieur. On lit donc $[2\,000; 4\,000[$.

Sur données groupées

```
nk = nk.to_numpy()
ci = (np.array(bornes[:-1]) + np.array(bornes[1:])) / 2 # centres
mg = (nk * ci).sum() / n # 7210 moyenne
np.sqrt((nk * ci**2).sum() / n - mg**2) # 2772.35 écart-type
Fk = np.cumsum(nk) / n # F aux bornes droites
np.interp([.10, .25, .50, .75, .90], np.r_[0, Fk], bornes)
# D1 4242 Q1 5197.5 Me 6747 Q3 8531.6 D9 11193
```

np.interp, c'est le polygone des fréquences cumulées

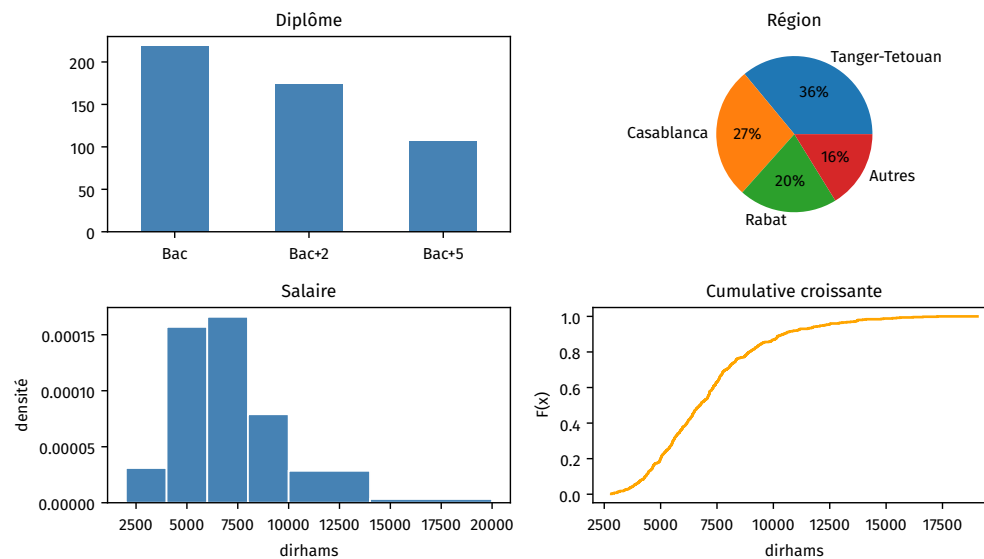
`np.interp(p, F, bornes)` lit sur le polygone qui relie les points $(F_k; b_k)$ la valeur où la fréquence cumulée atteint p : c'est **exactement** l'interpolation linéaire des chapitres 5 et 6. Une ligne pour les cinq quantiles groupés.

Les graphiques

Les quatre représentations

```
fig, ax = plt.subplots(2, 2, figsize=(11, 7))
sal.diplome.value_counts().sort_index().plot.bar(
    ax=ax[0,0], color="steelblue", rot=0, title="Diplôme")
sal.region.value_counts().plot.pie(
    ax=ax[0,1], autopct="%1.0f%%", ylabel="", title="Région")
ax[1,0].hist(sal.salaire, bins=bornes, density=True,
    color="steelblue")
ax[1,0].set(title="Salaire", xlabel="dirhams", ylabel="densité")
x = np.sort(sal.salaire)
ax[1,1].step(x, np.arange(1, n+1)/n, color="orange")
ax[1,1].set(title="Cumulative croissante", xlabel="dirhams",
    ylabel="F(x)")
plt.tight_layout(); plt.show()
```

Ce que Python affiche



L'argument `density=True`

L'équivalent exact de `freq = FALSE` en R

`density=True` trace la **densité**, seule hauteur correcte pour des classes d'amplitudes inégales. Sans lui, l'histogramme est faux dès la cinquième classe — et matplotlib, contrairement à R, **ne prévient pas**.

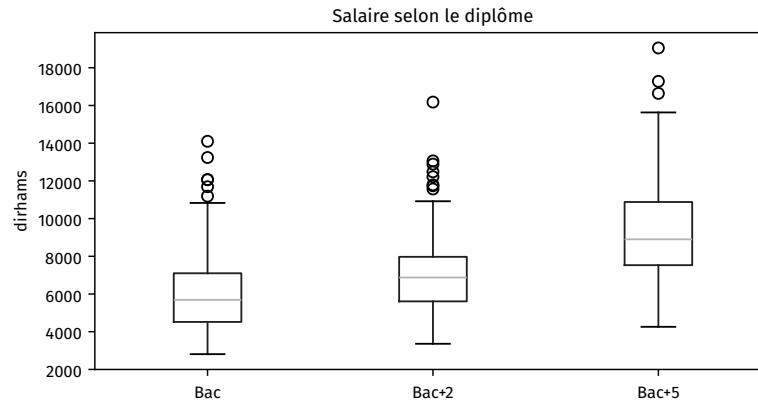
Les deux autres graphiques

La cumulative. `np.sort` range les 500 salaires ; `np.arange(1, n+1)/n` donne les fréquences cumulées $1/500, 2/500, \dots, 1$; `step` trace l'escalier de la fonction de répartition du chapitre 4 — une marche par salarié.

Le camembert. `autopct="%1.0f%%"` écrit la part de chaque région : 36 %, 27 %, 20 %, 16 %. Les angles ne se lisent pas ; les nombres, si.

Les boîtes à moustaches

```
sal.boxplot(column="salaire", by="diplome", grid=False)
plt.suptitle(""); plt.title("Salaire selon le diplôme")
plt.ylabel("dirhams"); plt.show()
```



Les indicateurs univariés

Tendance centrale, dispersion, forme

```
x = sal.salaire
resume = {
    "moyenne": x.mean(),          "mediane": x.median(),
    "ecart_type": x.std(ddof=0),  "CV": x.std(ddof=0) / x.mean(),
    "D1": x.quantile(.10),       "Q1": x.quantile(.25),
    "Q3": x.quantile(.75),       "D9": x.quantile(.90),
    "IQ": x.quantile(.75) - x.quantile(.25),
    "asymetrie": stats.skew(x),
    "aplatissement": stats.kurtosis(x, fisher=False),
}
pd.Series(resume).round(3)
```

Le résultat

Les indicateurs des chapitres 5 à 8, à la troisième décimale

moyenne	7158.260
mediane	6770.000
ecart_type	2598.100
CV	0.363
D1	4348.000
Q1	5317.500
Q3	8330.000
D9	10473.000
IQ	3012.500
asymetrie	1.164
aplatissement	4.837

Une même forme, plusieurs réponses

Quatre appels sur les mêmes 500 salaires

Appel	Asymétrie	Aplatissement
<code>stats.skew, kurtosis(fisher=False)</code>	1,164	4,837
<code>stats.kurtosis</code> , par défaut	—	1,837
les mêmes, avec <code>bias=False</code>	1,168	1,868
<code>x.skew()</code> , <code>x.kurt()</code> (pandas)	1,168	1,868

`fisher=False`, l'argument à ne jamais oublier

Par défaut, `scipy.stats.kurtosis` renvoie l'excès γ_2 — 3. `fisher=False` renvoie γ_2 , la convention du chapitre 8. `bias=False` et pandas appliquent en plus des **corrections d'échantillon**, comme $n - 1$ pour la variance. Lue sans sa convention, une valeur d'aplatissement est **ininterprétable** : 4,84 ou 1,84 ?

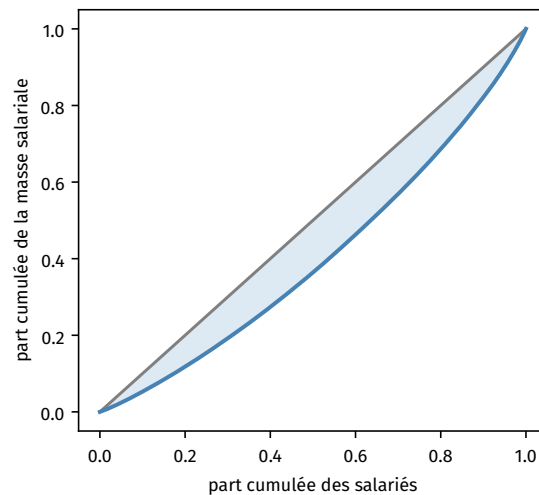
Concentration

```
def gini(v):
    v = np.sort(np.asarray(v, dtype=float)); m = len(v)
    return 2*np.sum(np.arange(1, m+1)*v)/(m*v.sum()) - (m+1)/m

gini(x) # 0.1954
sal.groupby("diplome").salaire.apply(gini) # 0.181 0.155 0.165

v = np.sort(x); F = np.arange(1, n+1)/n; Q = np.cumsum(v)/v.sum()
plt.plot([0,1], [0,1], color="grey")
plt.plot(np.r_[0,F], np.r_[0,Q], color="steelblue", lw=2)
plt.fill_between(np.r_[0,F], np.r_[0,Q], np.r_[0,F], alpha=.15)
plt.xlabel("part cumulée des salariés")
plt.ylabel("part cumulée de la masse salariale"); plt.show()
```

La courbe et l'aire de concentration



Colorier l'aire de concentration

`fill_between` colore la surface entre la diagonale et la courbe : G en est le **double**, soit 0,195. `np.r_[0, F]` ajoute l'origine en tête, comme $c(0, F)$ en R.

Le bivarié

Contingence

```
T = pd.crosstab(sal.diplome, sal.region)
pd.crosstab(sal.diplome, sal.region, margins=True)      # marges
pd.crosstab(sal.diplome, sal.region, normalize="index") # lignes
pd.crosstab(sal.diplome, sal.region, normalize="columns")
E = np.outer(T.sum(axis=1), T.sum(axis=0)) / n # théoriques
pd.DataFrame(E, index=T.index, columns=T.columns).round(1)
```

normalize, en trois valeurs

"index" divise par les totaux de ligne, "columns" par ceux de colonne, "all" par le total général. Les trois lectures du chapitre 10, en un seul argument.

Les profils-lignes

La sortie de `normalize = index`

region	Autres	Casablanca	Rabat	Tanger-Tetouan
diplome				
Bac	0.183	0.269	0.201	0.347
Bac+2	0.172	0.264	0.201	0.362
Bac+5	0.103	0.299	0.215	0.383

Ce qu'on y lit

Chaque **ligne** somme à 1 : on compare les diplômes. 10,3 % seulement des Bac+5 travaillent dans la région « Autres », contre 18,3 % des bacheliers ; à l'inverse, 29,9 % des Bac+5 sont à Casablanca. C'est le même constat que les profils-colonnes du chapitre 14, lu dans l'autre sens.

Les effectifs théoriques

np.outer, la formule du chapitre 10 sans boucle

`np.outer(a, b)` forme le tableau de tous les produits $a_i b_j$. Avec les deux marges, c'est $n_i \cdot n_j$; divisé par n , l'effectif attendu sous **indépendance**.

region	Autres	Casablanca	Rabat	Tanger-Tetouan
diplome				
Bac	35.5	60.0	44.7	78.8
Bac+2	28.2	47.7	35.5	62.6
Bac+5	17.3	29.3	21.8	38.5

L'écart à l'indépendance le plus net

Bac+5 dans la région « Autres » : 11 observés contre 17,3 attendus. Le tableau observé est au chapitre 14.

Conditionnelles

```
sal.groupby("diplome").salaire.agg(  
    n="size", moyenne="mean", ecart_type=lambda v: v.std(ddof=0),  
    mediane="median").round(1)
```

La sortie de agg()

	n	moyenne	ecart_type	mediane
diplome				
Bac	219	6058.9	2021.2	5690.0
Bac+2	174	7080.7	2045.3	6875.0
Bac+5	107	9534.4	2872.1	8900.0

groupby, l'équivalent de y ~ x

`sal.groupby("diplome").salaire` se lit « le salaire **selon** le diplôme » : c'est la formule de R, écrite autrement. Chaque méthode appliquée ensuite est calculée **par groupe**.

Décomposition de la variance

```
g = sal.groupby("diplome").salaire  
nj = g.size(); mj = g.mean(); vj = g.apply(lambda v: v.var(ddof=0))  
mg = sal.salaire.mean()  
intra = (nj*vj).sum()/n # 5010426  
inter = (nj*(mj-mg)**2).sum()/n # 1739696
```

```
eta2 = inter/(intra+inter) # 0.2577
```

Le ddof=0 dans le lambda

Sans lui, v_j serait calculé en $n_j - 1$: la variance intra vaudrait 5 043 703, et $\text{intra} + \text{inter}$ dépasserait la variance totale. **Le contrôle est toujours le même** : $\text{intra} + \text{inter}$ doit valoir `sal.salaire.var(ddof=0)`, soit 6 750 122.

Covariance, corrélation, ajustement

```
z = sal.anciennete

cov = np.cov(z, x, ddof=0)[0, 1] # 9859.91
r    = np.corrcoef(z, x)[0, 1]   # 0.6363
r2   = r**2                      # 0.4049

b = cov / z.var(ddof=0)          # 277.17
a = x.mean() - b*z.mean()       # 4347.2
```

Les formules du chapitre 13, telles quelles

$\hat{b} = \text{cov}(z, x) / V(z)$ puis $\hat{a} = \bar{x} - \hat{b}\bar{z}$. Attention au **même** ddof en haut et en bas : `np.cov` par défaut et `z.var()` divisent tous deux par $n - 1$ et donneraient la bonne pente ; mélanger `ddof=0` et le défaut la fausserait de 0,2%.

La version statsmodels

```
import statsmodels.formula.api as smf
mod = smf.ols("salaire ~ anciennete", data=sal).fit()
print(mod.summary())
```

La syntaxe formule, à nouveau

"salaire ~ anciennete" est **exactement** la syntaxe de R : statsmodels l'a reprise volontairement. `ols` signifie *ordinary least squares*, les moindres carrés ordinaires du chapitre 13.

À installer une fois

`statsmodels` ne fait pas partie des bibliothèques de base : `pip install statsmodels` dans un terminal, ou `%pip install statsmodels` dans un carnet Jupyter.

La sortie de summary()

Un extrait

Dep. Variable:	salaire	R-squared:	0.405
No. Observations:	500	Adj. R-squared:	0.404
=====			
	coef	std err	t
			P> t

Intercept	4347.2284	177.177	24.536	0.000
anciennete	277.1674	15.059	18.406	0.000

Skew:	0.985	Kurtosis:	4.494
-------	-------	-----------	-------

Ce qu'il faut lire, et un piège

Seuls coef ($\hat{a} = 4347$, $\hat{b} = 277$) et R-squared (0,405) relèvent de ce cours. Skew et Kurtosis, en bas, sont ceux des **résidus** (0,985), pas des salaires (1,164).

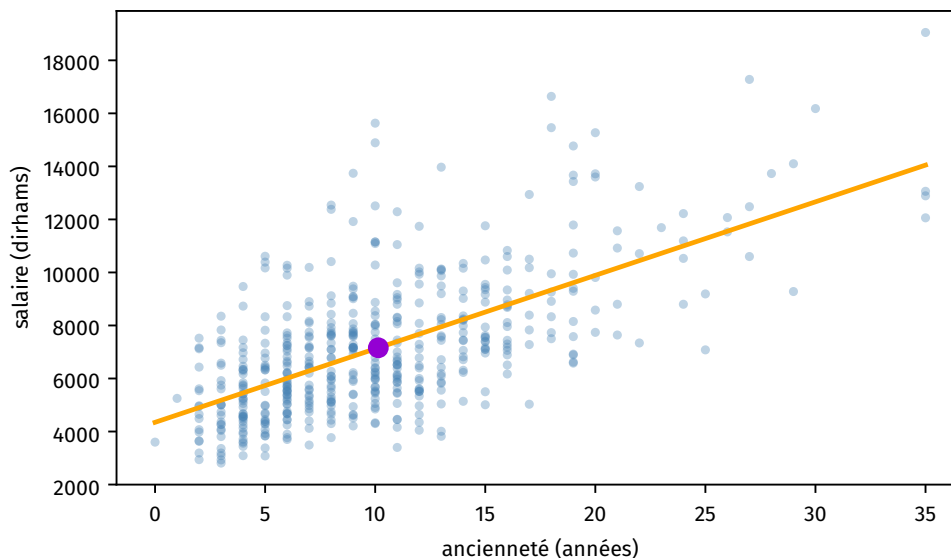
Le graphique final

```
plt.figure(figsize=(8, 5))
plt.scatter(z, x, s=8, alpha=.35, color="steelblue")
xx = np.linspace(0, 35, 2)
plt.plot(xx, a + b*xx, color="orange", lw=2)
plt.scatter(z.mean(), x.mean(), s=60, color="darkviolet", zorder=3)
plt.xlabel("ancienneté (années)"); plt.ylabel("salaire (dirhams)")
plt.show()
```

Deux points suffisent pour une droite

`np.linspace(0, 35, 2)` donne les deux abscisses 0 et 35 ; `zorder=3` dessine le point moyen **par-dessus** le nuage. Le contrôle est celui du chapitre 14 : le point moyen sur la droite.

Le nuage et sa droite



Comparaison des deux langages

Importer et résumer

R et Python, côte à côte

Opération	R	Python
Importer	<code>read.csv()</code>	<code>pd.read_csv()</code>
Structure	<code>str()</code>	<code>.info()</code>
Résumé	<code>summary()</code>	<code>.describe()</code>
Manquants	<code>colSums(is.na())</code>	<code>.isna().sum()</code>
Effectifs	<code>table()</code>	<code>.value_counts()</code>
Classes	<code>cut(right = FALSE)</code>	<code>pd.cut(right=False)</code>

Une différence de style

R appelle des **fonctions** sur les données : `summary(sal)`. Python appelle des **méthodes** des données : `sal.describe()`. Le résultat est le même ; seule l'écriture change.

Les indicateurs

R et Python, côte à côte

Opération	R	Python
Histogramme	<code>hist(freq = FALSE)</code>	<code>plt.hist(density=True)</code>
Quantiles	<code>quantile()</code>	<code>.quantile()</code>
Écart-type	<code>sd() * sqrt((n-1)/n)</code>	<code>.std(ddof=0)</code>
Asymétrie	<code>moments::skewness()</code>	<code>stats.skew()</code>
Aplatissement	<code>moments::kurtosis()</code>	<code>kurtosis(fisher=False)</code>
Gini	à écrire	à écrire

Les mêmes pièges, des deux côtés

Classes $]a; b]$ par défaut, histogramme en effectifs par défaut, $n - 1$ par défaut (R, pandas), excès d'aplatissement par défaut (scipy). **Aucun logiciel n'a les réglages par défaut du cours** : il faut les connaître.

Le bivarié

R et Python, côte à côte

Opération	R	Python
Croisement	<code>table(a, b)</code>	<code>pd.crosstab(a, b)</code>
Profils-lignes	<code>prop.table(T, 1)</code>	<code>normalize="index"</code>
Par groupe	<code>tapply(y, g, f)</code>	<code>.groupby(g).y.agg(f)</code>

Covariance	<code>cov() * (n-1)/n</code>	<code>np.cov(ddof=0)</code>
Corrélation	<code>cor()</code>	<code>np.corrcoef()</code>
Régression	<code>lm(y ~ x)</code>	<code>smf.ols("y ~ x")</code>

Un même nombre, deux écritures

`cor(sal$anciennete, x)` en R et `sal.anciennete.corr(x)` en Python renvoient tous deux 0,6363 : la corrélation du chapitre 13.

S'entraîner

Exercice 1 : trois écarts-types pour une série

Les dix salariés, en Python

```
v = np.array([2, 3, 4, 5, 6, 8, 10, 12, 15, 20]) # années
w = np.array([4.5, 5, 6.5, 5.5, 7.5, 6, 8.5, 7.5, 9, 11]) # milliers
np.std(v) # 5.4818
pd.Series(v).std() # 5.7783
np.cov(v, w)[0, 1] # 10.7778
pd.Series(v).quantile(.25) # 4.25
```

1. Lequel des deux écarts-types est descriptif ? Vérifier que leur rapport vaut $\sqrt{9/10}$.
2. Donner la covariance descriptive.
3. Retrouver $Q_1 = 4,25$ avec la règle du rang $1 + p(n - 1)$.

Exercice 1 : solution

ddof et rang des quantiles

1. `np.std` divise par n : 5,4818 est descriptif ($\sigma^2 = 30,05$). Rapport : $5,4818/5,7783 = 0,9487 = \sqrt{0,9}$.
2. `np.cov` divise par $n - 1$: $10,7778 \times 9/10 = 9,70$, la valeur du chapitre 12 — ou directement `np.cov(v, w, ddof=0)`.
3. Rang $1 + 0,25 \times 9 = 3,25$: entre $x_{(3)} = 4$ et $x_{(4)} = 5$, au quart du chemin, $Q_1 = 4 + 0,25 \times (5 - 4) = 4,25$.

Exercice 2 : les quantiles groupés à la main

Ce que fait `np.interp`

Fréquences cumulées aux bornes droites des six classes de salaire :

Borne (milliers de dh)	4	6	8	10	14	20
F	0,062	0,376	0,708	0,866	0,980	1

1. Retrouver par interpolation linéaire $Q_3 = 8\,531,6$ et $D_9 = 11\,193$.
2. Les valeurs exactes sont $Q_3 = 8\,330$ et $D_9 = 10\,473$. Pourquoi l'écart est-il plus grand pour D_9 ?

Exercice 2 : solution

Interpolation dans la classe

1. F franchit 0,75 dans $[8\,000; 10\,000[$ et 0,90 dans $[10\,000; 14\,000[$:

$$Q_3 = 8\,000 + 2\,000 \times \frac{0,75 - 0,708}{0,158} = 8\,531,6$$

$$D_9 = 10\,000 + 4\,000 \times \frac{0,90 - 0,866}{0,114} = 11\,193$$

2. L'interpolation suppose les salariés **répartis uniformément** dans la classe. Or la densité décroît : dans $[10\,000; 14\,000[$, les salaires se tassent vers le bas de la classe. Plus la classe est **large**, plus l'erreur est grande : +720 dh pour D_9 , contre +200 pour Q_3 .

Exercice 3 : traduire un script R

Cinq lignes de R, à écrire en Python

```
cl <- cut(x, breaks = bornes, right = FALSE)
prop.table(table(sal$diplome, sal$sexe), 2)
tapply(x, sal$sexe, median)
sd(x) * sqrt((n-1)/n)
cor(sal$anciennete, x)^2
```

Pour chaque ligne, écrire l'équivalent Python et donner le résultat attendu sur nos données lorsqu'il tient en un ou deux nombres.

Exercice 3 : solution

La traduction

```
cl = pd.cut(x, bins=bornes, right=False)
pd.crosstab(sal.diplome, sal.sexe, normalize="columns")
sal.groupby("sexe").salaire.median()      # F 6580   H 6820
x.std(ddof=0)                             # 2598.10
sal.anciennete.corr(x)**2                  # 0.405
```

Le profil-colonne obtenu

Les femmes comptent 22,3 % de Bac+5, les hommes 20,7 % : des structures de diplôme très proches, ce qui explique en partie l'égalité des salaires moyens par sexe.

Ce qu'il faut retenir

Le résumé du chapitre

Sept points

1. `pandas` divise par $n - 1$, `numpy` par n : **toujours préciser `ddof`**.
2. `pd.cut(..., right=False)` pour des classes $[a; b[$.
3. `density=True` dans `hist` : la densité, jamais l'effectif.
4. `stats.kurtosis(..., fisher=False)` pour γ_2 ; `x.kurt()` donne un excès corrigé.
5. `np.interp` sur les fréquences cumulées donne les quantiles groupés.
6. `pd.crosstab(..., normalize=)` : "index", "columns" ou "all".
7. `groupby` et `statsmodels` reprennent la logique « y selon x » de R.

Bilan des seize chapitres

Le fil complet

Étape	Chapitres	Résultat sur nos données
Définir	1	500 salariés, six variables typées
Résumer	2, 3	six classes de salaire
Représenter	4	quatre graphiques
Situer	5, 6	moyenne 7 158, médiane 6 770
Disperser	7	$\sigma = 2\,598$, $CV = 0,36$
Décrire la forme	8	$\gamma_1 = 1,16$, $\gamma_2 = 4,84$
Mesurer le partage	9	Gini = 0,195
Croiser	10, 11	$\eta^2 = 0,258$ pour le diplôme
Lier	12, 13	$r = 0,636$, $r^2 = 0,405$
Automatiser	14, 15	une quinzaine de lignes de code

La phrase qui résume le semestre

Résumer, et savoir ce qu'on a laissé de côté

Un tableau de mille lignes ne dit rien tant qu'on ne l'a pas résumé — et un résumé ne vaut rien tant qu'on ne sait pas ce qu'il a laissé de côté. Vous savez maintenant faire les deux.

Ce que chaque résumé laisse de côté

La moyenne ignore la dispersion ; l'écart-type ignore la forme ; r ignore tout ce qui n'est pas linéaire ; η^2 ignore la composition des groupes. **Chaque indicateur répond à une question, et à une seule.**

Et maintenant

Lequel choisir

Les deux donnent **les mêmes nombres**, ce qui est le seul point réellement important. **R** est plus direct pour la statistique ; **Python** est plus polyvalent : le même langage récupère les données, les nettoie, les analyse et publie le résultat. En Licence, apprenez celui que vos enseignants emploient.

La suite du parcours

Ce cours s'est arrêté au constat. Restent les **probabilités**, pour modéliser le hasard ; la **statistique inférentielle**, pour généraliser au-delà de l'échantillon ; l'**économétrie**, pour passer de la corrélation à l'explication.