Test de conformité d'une moyenne.

Une entreprise usine des pièces dont le **diamètre** doit être de **0.5mm**. Pour **vérifier la qualité de la production et le bon réglage de la fraiseuse**, le contrôleur **soustrait aléatoirement 25 pièces après usinage**. Il relève le diamètre pour chacune d'entre elle. Le **diamètre moyen** est de **0.53mm**. L'écart étant suffisamment important pour envisager la défaillance de la fraiseuse, il décide de **vérifier qu'elle ne soit pas simplement une variation liée à l'échantillon sélectionné**. Le contrôleur doit donc mener une analyse inferentielle pour répondre à la question suivante : la moyenne observée est-elle imputable à l'ensemble des pièces usinées ? Autrement dit, la différence de moyenne relevée est-elle observable dans la population générale ? Le contrôleur tolère un **risque de répondre à tord** de **5%**.

Voici les donénes recueillis par le contrôleur qualité.

```
import pandas as pd
import numpy as np
import random
random.seed(10)

moy_cible = 0.50
n = 25
data = np.random.normal(loc=0.53, scale=0.01, size=n)
serie = pd.Series(data)
moy_obs = serie.mean()
display(moy_obs)

    0.532018746761403

alpha = 0.05
```

Pour accéder au notebook de l'étude de cas, c'est par ici .

Statistique de test.

Le contrôleur qualité va tester la différence entre la moyenne observée et la moyenne de référence. Toutefois, il s'intéresse à cette différence en unité d'écart-type à la moyenne. La statistique de test est donc égale à :

$t = \frac{moy_{obs} - moy_{cible}}{\frac{\sigma}{\sqrt{n}}},$

où σ est l'**l'écart-type de la populaton**.

Cette statistique de test suit une **loi de student à n-1 degrés de liberté** (ddl) : $t_{n-1ddl;0.05}$.

Cependant, la variance de la population n'est pas connue. Il faut donc donner une estimation sans biais de ce paramètre. Ce dernier s'exprime comme suit :

$s' = \frac{1}{n-1} s,$

où s représente la variance de l'échantillon.

Jeu des hypothèses.

La contrôleur souhaite vérifier que la moyenne ne s'écarte en aucun cas de celle attendue. Autrement dit, il souhaite réaliser un test bilatéral.

Formellement, le contrôleur qualité pose le jeu des hypothèses de son test statistique :

$H_0 : moy_{obs} = moy_{cible},$

$H_1 : moy_{obs} \neq moy_{cible}.$

La **moyenne cible** est **0.5mm**, la **moyenne observée** vaut **0.53mm**.

Test statistique.

On peut maintenant calculer la statistique de test et, soit la **comparer au quantile de la loi de student**, soit **calculer la probabilité d'observer une statistique au moins aussi grande** - aka pvalue.

Néanmoins, la validité du teste repose sur la normalité des donnés. Dans la mesure où, notre échantillon est de petite taille, le théorème central limit ne peut pas être utilisé. Nous allons donc réaliser un test d'adéquation à la loi normale. On peut utiliser le test de Shapiro-Wilk.

```
from scipy.stats import shapiro
statistic, pvalue = shapiro(serie)

display(statistic, pvalue)
```

```
    0.9695425629615784
    0.6334888339042664
```

Le résultat du test de Shapiro-Wilk ne nous permet pas de rejeter l'hypothèse nulle d'adéquation de l'échantillon à une loi normale. Les résultats de notre test de conformité à la moyenne seront donc valides. On calcul la statistique de test.

```
std = np.std(serie, ddof = 1) # estimation sans biais de la variance

t = (moy_obs - moy_cible) / (std / np.sqrt(n))
display(t)
```

```
    14.72308821188204
```

On peut maitenant comparer cette statstique de test au quantile de la loi de student à n-1 ddl pour un risque d'erreur de 5%.

```
# quantile loi de student.
from scipy import stats
quantile = stats.t.ppf(1-alpha/2, n-1)
display(quantile)
```

```
    2.0638985616280205
```

Puisque le quantile est inférieur à notre statistique de test, on peut conclure à un écart à la moyenne cible statistiquement significatif. La fraiseuse présente bien un défaut de réglage. Pour nous en assurer, calculons la probabilité d'observer une statistique de test au moins aussi extrême que celle observée.

```
# pvalue
from scipy import stats
pvalue = 2*(1-stats.t.cdf(t, n-1))
display(pvalue)
```

```
    1.6275869541004795e-13
```

La probabiltié d'observer une statistique de test au moins aussi extrême - **pvalue** - est faible. On peut donc rejeter l'hypothèse nulle au profit de l'hypothèse alternative et conclure à la défaillance machine.

Pour finir, calculons l'intervalle de confiance de notre moyenne observée.

```
# IC_95
ic_sup = moy_obs + quantile * std/np.sqrt(n)
ic_inf = moy_obs - quantile * std/np.sqrt(n)

display(ic_inf, moy_cible, ic_sup)
```

```
    0.5275303239227311
    0.5
    0.5365071696000749
```

La moyenne cible n'appartient pas à l'intervalle de confiance de notre moyenne observée. On a donc 5 chances sur 100 de conclure à tord à la défaillance machine.

Test de student pour un échantillon.

Nous allons vérifier nos conclusions avec le test t de Student pour un échantillon. Il permet de comparer la moyenne observée d'un échantillon à la moyenne de la population. Vous pouvez réaliser ce test grâce à la fonction *ttest_1samp* du module *scipy.stats*[^1].

```
# test de student à un échantillon
from scipy.stats import ttest_1samp as test_conformite
```

```
results = test_conformite(serie, moy_cible, alternative='two-sided')
display(results.statistic, results.pvalue,
results.confidence_interval(confidence_level=0.95))
```

```
    14.72308821188204
    1.6278066928222218e-13
    ConfidenceInterval(low=0.5275303239227312, high=0.5365071696000748)
```

Nos résultats sont concordants avec le test réalisé. Il existe une différence statistiquement significative entre la moyenne observée et la moyenne de la population cible. On peut donc affirmer, avec un risque d'erreur de 5%, que la fraiseuse est mal réglée.

[^1]: Depuis la *version 1.10.0* de scipy vous disposez d'une méthode pour obtenir l'intervalle de confiance (avec un risque d'erreur donné) ainsi que le nombre de degrés de liberté.