scipy

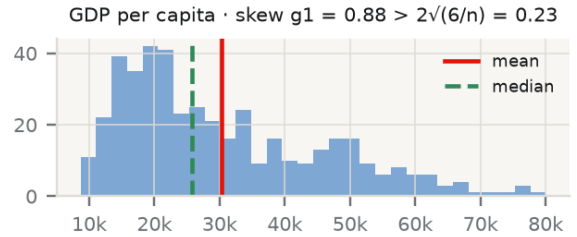
scipy.stats: describe a distribution, read a test, turn a statistic into a p-value

Python cheatsheet · MATH60033A · Quantitative Methods in IB · used in sessions 2, 4, 5, 6, 9 and 10

```
from scipy import stats
```

every distribution has .pdf, .cdf, .ppf, .sf

1 · Describe a distribution (session 2)



`stats.trim_mean(x, 0.05)`

the mean after cutting 5% off each tail: between the mean and the median.

```
stats.trim_mean(core["gdp_pc_eur"].dropna(), 0.05)
```

`stats.skew(x)`

g1: 0 = symmetric, > 0 = a long right tail. Substantially skewed if $|g1| > 2\sqrt{(6/n)}$.

`stats.kurtosis(x)`

g2, **excess** kurtosis: 0 for a normal; > 0 = more extreme values. Substantial if $|g2| > 4\sqrt{(6/n)}$.

`stats.shapiro(resid)`

a test of normality; p < 0.05 says the residuals are not normal.

> Always report a skewness or a kurtosis **with** its threshold — the number alone says little.

2 · Four methods, every distribution

`.pdf(x)`

density: the height of the curve.

`.cdf(x)`

P(X ≤ x): the area to the left.

`.sf(x)`

P(X > x) = 1 − cdf: the area to the right — a one-sided p-value.

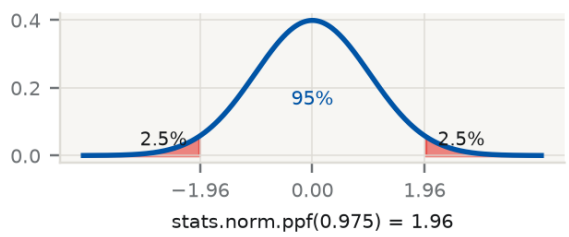
`.ppf(q)`

the inverse of the cdf: the critical value for a level q.

> `sf` is more precise than `1 - cdf` for tiny p-values.

3 · The normal distribution

`stats.norm` is the bell curve. The same four methods work for every distribution.



`stats.norm.pdf(x)`

the height of the curve at x — for drawing it.

`stats.norm.cdf(1.96)`

the area to the left of x: 0.975.

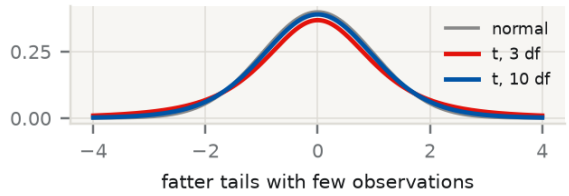
`stats.norm.ppf(0.975)`

the inverse: the x that leaves 97.5% to its left — 1.96, the 95% critical value.

`stats.norm.sf(z) * 2`

the area to the right, doubled: a two-sided p-value for a z-statistic.

4 · The t distribution



`stats.t.ppf(0.975, df=n - k - 1)`

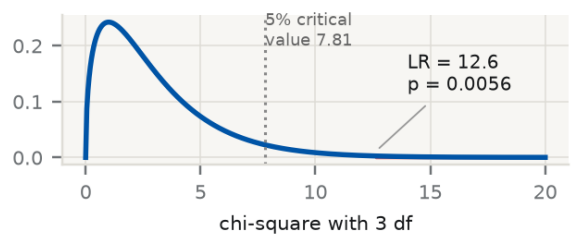
the critical value with few observations: above 1.96, closer to it as n grows.

`stats.t.sf(abs(t), df) * 2`

the two-sided p-value of a t-statistic — what a regression table prints.

5 · Chi-square: the LR test (session 4)

Several coefficients at once: compare the log-likelihoods of two nested models.



`lr = 2 * (full.llf - restricted.llf)`

the LR statistic: how much the dropped coefficients improved the fit.

`stats.chi2.sf(lr, df=q)`

its p-value, with q = the number of coefficients dropped.

```
stats.chi2.sf(12.6, 3)    # 0.0056: keep the three dummies
```

`stats.chi2.ppf(0.95, df=3)`

the 5% critical value: 7.81. Reject when LR is above it.

6 · Correlation

`r, p = stats.pearsonr(x, y)`

the correlation and the p-value of "no correlation".

`rho, p = stats.spearmanr(x, y)`

the same on ranks: robust to outliers and to curves that only go up.

> A p-value after you picked the variable with the strongest correlation is not valid: the same data chose and tested it.

7 · A regression line, by hand

What the summary table computes for each coefficient.

`t = coef / se`

how many standard errors the estimate is from zero.

`p = 2 * stats.t.sf(abs(t), df)`

the two-sided p-value, df = n − k − 1.

`coef ± stats.t.ppf(0.975, df) * se`

the 95% confidence interval.