# stats.py



```python
import random, bisect

"""
True if x, y are indistinguishable (via KS) and differ by a negligible effect (via Cliff's Delta).

References:
1. Cliff's Delta thresholds: Romano, J., Kromrey, J. D., Coraggio, J., & Skowronek, J. (2006).
   'Appropriate statistics for ordinal level data: Should we really be using t-test and cohen's d
   for evaluating group differences?' Journal of Modern Applied Statistical Methods, 5(2), 24.
2. KS Test Jitter: The addition of epsilon noise ensures the data is strictly continuous, satisfying
   the theoretical assumptions of the Kolmogorov distribution (Smirnov, 1948) regarding ties.
"""
def same(x: list[float], y: list[float], Ks=0.95, Delta="smed") -> bool:
  x = sorted(i + 1e-16 * random.random() for i in x)
  y = sorted(i + 1e-16 * random.random() for i in y)
  n, m = len(x), len(y)
  def _cliffs():
    gt = sum(m - bisect.bisect_right(y, a) for a in x)
    lt = sum(bisect.bisect_left(y, a) for a in x)
    return abs(gt - lt) / (n * m)
  def _ks():
    all_steps = sorted([(a, 1/n) for a in x] + [(b, -1/m) for b in y])
    cdf, max_d = 0, 0
    for _, step in all_steps:
      cdf += step
      max_d = max(max_d, abs(cdf))
    return max_d

  ks_crit = {0.1: 1.22, 0.05: 1.36, 0.01: 1.63}[round(1 - Ks, 2)]
  cliffs_thresh={ 'small':0.147, 'smed':0.238, 'medium':0.33, 'large':0.474 }[Delta]
  return _cliffs() <= cliffs_thresh and _ks() <= ks_crit * ((n + m)/(n * m))**0.5

from types import SimpleNamespace as Obj

from types import SimpleNamespace as Obj

def rx_sk(data, same):
  def div(cols):
    if len(cols) < 2: return [cols]
    st, nt = sum(c.sum for c in cols), sum(c.n for c in cols)
    best, sl, nl = None, 0, 0
    for i in range(len(cols) - 1):
      sl += cols[i].sum; nl += cols[i].n
      if (val := sl**2/nl + (st-sl)**2/(nt-nl)) > (best[0] if best else -1):
        best = (val, i)

    cut = best[1] + 1
    lhs = [x for c in cols[:cut] for x in c.has]
    rhs = [x for c in cols[cut:] for x in c.has]
    return [cols] if same(lhs, rhs) else div(cols[:cut]) + div(cols[cut:])

  cols = [Obj(k=k, n=len(v), sum=sum(v), has=v) for k,v in data.items()]
  cols.sort(key=lambda c: c.sum / c.n)
  return {c.k: i+1 for i, group in enumerate(div(cols)) for c in group}
```