Short benchmark of Fleiss' kappa and Brennan-Prediger

Benchmark for the aggregated functions

Jonas Moss

4/25/23

The most feature complete `R` package for agreement coefficients is `irrCAC`. It implements Fleiss' kappa and the Brennan-Prediger coefficient for both aggregated and long form data. Despite the fact that their method of inference, based on U-statistics, does not look the same as ours (based on moments) they are equivalent in the case of aggregated data. Compare the confidence intervals below to be convinced of this.

```
library("quadagree")
irrCAC::bp.coeff.dist(dat.fleiss1971, weights = "quadratic")
```

```
       coeff.name     coeff    stderr      conf.int     p.value        pa   pe
1 Brennan-Prediger 0.3338889 0.1036175 (0.122,0.546) 0.003134299 0.8334722 0.75
```

```
bpci_aggr(dat.fleiss1971)
```

```
Call: call

95% confidence interval (n = 30).
    0.025     0.975
0.1219674 0.5458104

Sample estimates.
    kappa        sd
0.3338889 0.5579971
```

For aggregated data, `quadagree` supports user-supplied `values` vectors, transforms, and studentized bootstrapping, which `irrCAC` does not. But is it faster? It turns out it is roughly twice as fast for reasonable numbers of raters. This suggests there is no benefit in using the moment formulation (as done in `quadagree`) when calculating these coefficients, as `quadagree` is likely to be more optimized than `irrCAC`.

Benchmarks

We will run three benchmarks of various sizes using the `microbenchmark` package. We start off with `dat.fleiss1971`, which contains $n = 30$ rows.

```
x = dat.fleiss1971
irr_bp <- \(x) irrCAC::bp.coeff.dist(x, weights = "quadratic")
irr_fleiss <- \(x) irrCAC::fleiss.kappa.dist(x, weights = "quadratic")
microbenchmark::microbenchmark(
  irr_bp(x),
  bpci_aggr(x),
  irr_fleiss(x),
  fleissci_aggr(x),
  times = 1000
)
```

```
Unit: microseconds
             expr   min     lq     mean median     uq    max neval
        irr_bp(x) 280.1 311.20 379.1637 327.60 369.35 3863.9  1000
     bpci_aggr(x) 104.5 131.10 163.1652 145.50 163.70 1950.9  1000
    irr_fleiss(x) 284.5 323.50 386.6799 340.40 387.80 4610.3  1000
 fleissci_aggr(x) 116.5 142.05 179.5976 158.25 179.85 3079.4  1000
```

So `quadagree` is roughly twice as fast. Let's see what happens when $n = 300$.

```
x = dat.fleiss1971
x = rbind(x, x, x, x, x, x, x, x, x, x)
microbenchmark::microbenchmark(
  irr_bp(x),
  bpci_aggr(x),
  irr_fleiss(x),
  fleissci_aggr(x),
  times = 1000
)
```

```
Unit: microseconds
             expr   min     lq     mean median     uq    max neval
        irr_bp(x) 304.6 344.15 411.8070 367.80 410.70 3977.3  1000
     bpci_aggr(x) 110.8 138.90 170.4819 154.60 176.60 1890.5  1000
    irr_fleiss(x) 320.3 356.45 433.3315 380.25 433.90 4988.6  1000
 fleissci_aggr(x) 129.2 161.05 209.1645 178.20 204.35 4142.2  1000
```

The run time is almost the same for all methods as it was for $n = 30$, suggesting that there is substantial overhead to both methods. Let's check $n = 3000$.

```
# recall that x has 300 elements.
x = rbind(x, x, x, x, x, x, x, x, x, x)
microbenchmark::microbenchmark(
  irr_bp(x),
  bpci_aggr(x),
  irr_fleiss(x),
  fleissci_aggr(x),
  times = 1000
)
```

```
Unit: microseconds
             expr   min     lq      mean median     uq     max neval
        irr_bp(x) 537.1 751.15 1020.6125 796.65 964.60 57192.8  1000
     bpci_aggr(x) 193.8 273.35  344.1337 296.30 344.25  6094.7  1000
    irr_fleiss(x) 559.0 799.80 1037.9914 844.05 998.35  9290.1  1000
 fleissci_aggr(x) 260.6 379.90  472.3803 406.55 469.00  6021.9  1000
```

It appears that `bpci_aggr` is pulling ahead of `irrCAC::bp.coeff.dist`.

Let's finish off with a larger number of categories.

```
# recall that x has 3000 elements.
x = cbind(x, x, x, x, x, x, x, x, x, x)
microbenchmark::microbenchmark(
  irr_bp(x),
  bpci_aggr(x),
  irr_fleiss(x),
  fleissci_aggr(x),
  times = 1000
)
```

```
Unit: microseconds
             expr    min      lq     mean  median      uq     max neval
        irr_bp(x) 4493.7 4859.95 6186.148 5318.00 6854.55 54246.6  1000
     bpci_aggr(x)  724.8  851.85 1136.090  911.90 1073.80  6765.2  1000
    irr_fleiss(x) 4658.3 5047.55 6339.300 5429.15 7318.15 33798.4  1000
 fleissci_aggr(x)  786.1  925.90 1193.103  981.90 1139.40  8777.2  1000
```

So `quadagree` could substantially faster on data with very many categories and items rated.