Weighted quantile estimators

1 The method commonly used in CP papers

```r
weighted.quantile <- function(x, p, w = NULL, sorted = FALSE) {
  if (is.null(w)) w <- rep(1, length(x))
  if (!sorted) { o <- order(x); x <- x[o]; w <- w[o] }
  i <- which(cumsum(w/sum(w)) >= p)
  if (length(i) == 0) return(Inf) # Can happen with infinite weights
  else return(x[min(i)])
}
```

It corresponds to Type 1(Hyndman and Fan 1996). The mathematical inverse $\tilde{F}^{-1}(p)$ of the CDF is the smallest x such that $F(x) \geq p$.

2 Akinshin (2023)

2.1 Desired properties of the weighted quantile estimators

- **R1:** consistency with existing quantile estimators.

$$\mathrm{Q}^*(\mathbf{x}, \mathbf{w}, p) = \mathrm{Q}(\mathbf{x}, p), \text{ where } \mathbf{w} = \{1, 1, \dots, 1\}.$$

- **R2:** zero weight support.

$$\mathrm{Q}^*\left(\{x_1, x_2, \dots, x_{n-1}, x_n\}, \{w_1, w_2, \dots, w_{n-1}, 0\}, p\right) = \mathrm{Q}^*\left(\{x_1, x_2, \dots, x_{n-1}\}, \{w_1, w_2, \dots, w_{n-1}\}, p\right).$$

- **R3:** stability (continuity of the quantile estimations with respect to the weight coefficient).

$$\lim_{\varepsilon_i \to 0} Q^* \left(\{x_1, x_2, \dots, x_n\}, \{w_1 + \varepsilon_1, w_2 + \varepsilon_2, \dots, w_n + \varepsilon_n\}, p\right) \to \mathrm{Q}^* \left(\{x_1, x_2, \dots, x_n\}, \{w_1, w_2, \dots, w_n\}, p\right).$$

2.2 Some issues

1. Some simple R package implementations of weighted quantiles show **violations of R3**.

```
x <- c(0, 1, 100)
wA <- c(1, 0.00000, 1)
wB <- c(1, 0.00001, 1)
message(quantile(c(0, 100), 0.5))
## 50
message(modi::weighted.quantile(x, wA, 0.5), " | ",
        modi::weighted.quantile(x, wB, 0.5))
## 100 | 1
message(laeken::weightedQuantile(x, wA, 0.5), " | ",
        laeken::weightedQuantile(x, wB, 0.5))
## 100 | 1
message(MetricsWeighted::weighted_quantile(x, wA, 0.5), " | ",
        MetricsWeighted::weighted_quantile(x, wB, 0.5))
## 100 | 1
message(spatstat.geom::weighted.quantile(x, wA, 0.5), " | ",
        spatstat.geom::weighted.quantile(x, wB, 0.5))
## 1 | 0.499999999994449
message(matrixStats::weightedMedian(x, wA), " | ",
        matrixStats::weightedMedian(x, wB))
## 50 | 1.00000000000003
```

2. In order to satisfy R2, the sample size needs adjustments.

```
non_zero <- wA != 0
message(modi::weighted.quantile(x, wA, 0.5), " | ",
        modi::weighted.quantile(x[non_zero], wA[non_zero], 0.5))
## 100 | 50
message(laeken::weightedQuantile(x, wA, 0.5), " | ",
        laeken::weightedQuantile(x[non_zero], wA[non_zero], 0.5))
## 100 | 100
message(MetricsWeighted::weighted_quantile(x, wA, 0.5), " | ",
        MetricsWeighted::weighted_quantile(x[non_zero], wA[non_zero], 0.5))
## 100 | 100
message(spatstat.geom::weighted.quantile(x, wA, 0.5), " | ",
        spatstat.geom::weighted.quantile(x[non_zero], wA[non_zero], 0.5))
## 1 | 0
message(matrixStats::weightedMedian(x, wA), " | ",
        matrixStats::weightedMedian(x[non_zero], wA[non_zero]))
## 50 | 50
```

Consider using **Kish's effective sample size** given by

$$n^*(\mathbf{w}) = \frac{\left(\sum_{i=1}^n w_i\right)^2}{\sum_{i=1}^n w_i^2} = \frac{1}{\sum_{i=1}^n \bar{w}_i^2},$$

where $\mathbf{w}$ is the vector of non-negative weight coefficients, $\bar{\mathbf{w}}$ is the vector of normalized weights where $\bar{w}_i = \frac{w_i}{\sum_{i=1}^n w_i}$. So $n^*(\{1,1,1\}) = 3$ and $n^*(\{1,1,1,0,0\}) = 3$.

2.3 Method 1: Weighted Harrell–Davis quantile estimator

- **A linear combination of all order statistics.**
- **Pros:** provide higher statistical efficiency in the cases of light-tailed distributions.
- **Cons:** not robust (its breakdown point is zero).

The weighted Harrell–Davis quantile estimator is defined as

$$\mathrm{Q}^*_{\mathrm{HD}}(\mathbf{x}, \mathbf{w}, p) = \sum_{i=1}^n W^*_{\mathrm{HD},i} \cdot x_{(i)}, \quad W^*_{\mathrm{HD},i} = I_{t_i^*}\left(\alpha^*, \beta^*\right) - I_{t_{i-1}^*}\left(\alpha^*, \beta^*\right),$$

where $I_{t_i^*}\left(\alpha^*, \beta^*\right)$ is the CDF of the beta distribution $\mathrm{Beta}(\alpha^*, \beta^*)$, $\alpha^* = (n^*+1)\,p$, $\beta^* = (n^*+1)\,(1-p)$, $t_i^* = s_i(\bar{\mathbf{w}})$ is the partial sum of normalized weight coefficients.

$W^*_{\mathrm{HD},i}$ can be considered as probabilities of observing the target quantile at the given position. See details from Examples 7 and 8 in the paper.

For all $p \in (0;1)$, $\mathrm{Q}^*_{\mathrm{HD}}(\mathbf{x}, \mathbf{w}, p)$ satisfies R1, R2, and R3.

2.4 Method 2: Weighted trimmed Harrell–Davis quantile estimator

- **A trimmed modification** of the HD method.
- **Idea:** since most of the linear coefficients $W^*_{\mathrm{HD},i}$ are pretty small, they do not have a noticeable impact on efficiency, but they significantly reduce the breakdown point.
- **Pros:** allow customizing trade-off between robustness and efficiency.
- **Cons:** use the rule of thumb to decide $[L^*; R^*]$.

  For $p \in (0;1)$, the weighted trimmed Harrell–Davis quantile estimator based on the beta distribution highest density interval $[L^*; R^*]$ of the given size D^* (the rule of thumb: $D^* = 1/\sqrt{n^*}$) is defined as

$$\mathrm{Q}^*_{\mathrm{THD}}(\mathbf{x}, \mathbf{w}, p) = \sum_{i=1}^{n} W^*_{\mathrm{THD},i} \cdot x_{(i)}, \quad W^*_{\mathrm{THD},i} = F^*_{\mathrm{THD}}\left(t^*_i\right) - F^*_{\mathrm{THD}}\left(t^*_{i-1}\right), \quad t^*_i = s_i(\overline{\mathbf{w}}),$$

  where

$$F^*_{\mathrm{THD}}(t) = \begin{cases} 0 & \text{for } t < L^*, \\ \left(I_t\left(\alpha^*, \beta^*\right) - I_{L^*}\left(\alpha^*, \beta^*\right)\right) / \left(I_{R^*}\left(\alpha^*, \beta^*\right) - I_{L^*}\left(\alpha^*, \beta^*\right)\right) & \text{for } L^* \le t \le R^*, \\ 1 & \text{for } R^* < t. \end{cases}$$

2.5 Method 3: Weighted traditional quantile estimators

- Implement the framework of the Harrell–Davis quantile estimator to Hyndman and Fan (1996) in which Types 1-9 are considered.
  - Types 1–3 have discontinuities, so the corresponding estimators fail to satisfy R3.
  - Here only Types 4-9 (using a linear combination of **two order statistics**) are considered.
- **Pros:** robust, not so efficient.

Table: The Hyndman & Fan (1996) taxonomy of quantile estimators.

Type	h	Equation
1	np	$x_{(\lceil h \rceil)}$
2	$np+1/2$	$\left(x_{(\lceil h-1/2 \rceil)} + x_{(\lceil h+1/2 \rceil)}\right)/2$
3	np	$x_{(\lfloor h \rceil)}$
4	np	$x_{(\lfloor h \rfloor)} + (h - \lfloor h \rfloor)\left(x_{(\lceil h \rceil)} - x_{(\lfloor h \rfloor)}\right)$
5	$np+1/2$	$x_{(\lfloor h \rfloor)} + (h - \lfloor h \rfloor)\left(x_{(\lceil h \rceil)} - x_{(\lfloor h \rfloor)}\right)$
6	$(n+1)p$	$x_{(\lfloor h \rfloor)} + (h - \lfloor h \rfloor)\left(x_{(\lceil h \rceil)} - x_{(\lfloor h \rfloor)}\right)$
7	$(n-1)p+1$	$x_{(\lfloor h \rfloor)} + (h - \lfloor h \rfloor)\left(x_{(\lceil h \rceil)} - x_{(\lfloor h \rfloor)}\right)$
8	$(n+1/3)p+1/3$	$x_{(\lfloor h \rfloor)} + (h - \lfloor h \rfloor)\left(x_{(\lceil h \rceil)} - x_{(\lfloor h \rfloor)}\right)$
9	$(n+1/4)p+3/8$	$x_{(\lfloor h \rfloor)} + (h - \lfloor h \rfloor)\left(x_{(\lceil h \rceil)} - x_{(\lfloor h \rfloor)}\right)$

The weighted case of these quantile estimators can be rewritten in a form that matches the definition of $\mathrm{Q}^*_{\mathrm{THD}}$ given by

$$Q^*_k(\mathbf{x}, \mathbf{w}, p) = \sum_{i=1}^{n} W^*_{F^*_k, i} \cdot x_{(i)}, \quad W^*_{F^*_k, i} = F^*_k\left(t^*_i\right) - F^*_k\left(t^*_{i-1}\right), \quad t^*_i = s_i(\bar{\mathbf{w}}),$$

where

$$F^*_k(t) = \begin{cases} 0 & \text{for} & t < (h^*-1)/n^*, \\ tn^* - h^* + 1 & \text{for} & (h^*-1)/n^* \le t \le h^*/n^*, \\ 1 & \text{for} & t > h^*/n^*, \end{cases}$$

where h^* is calculated based on p^* rather than p. So, instead of Beta distribution, uniform distribution is suggested in the paper to define the required PDF and CDF.

2.6 Summary

- The methods tend to be too smooth as they tend to use more order statistics (more than two) for linear combination to get the weighted quantile.
- Regarding conformal prediction, we normally calculate $\mathrm{Q}_{1-\alpha}\left(\sum_{i=1}^{n} \tilde{w}_i \cdot \delta_{R_i} + \tilde{w}_{n+1} \cdot \delta_{+\infty}\right)$ in which the maximum is Inf, then the resulting weighted quantile tends to return Inf as well.
- We also tried to remove Inf from the calculation of the required quantile, the average coverage is still biased compared to nominal coverage, which is the same case when employing the weighted quantile method commonly used in conformal prediction papers.

3 The survey package

Quantile rules in the survey R package by Thomas Lumley.

In the `svyquantile` function of the **survey** R package, there is an argument named qrule in which we can choose the rule/function used to define the quantiles.

3.1 Discontinuous quantiles

$$\hat{F}(x) = \frac{\sum_i \{x_i \leq x\} w_{(i)}}{\sum_i w_{(i)}}$$

```
last <- function(x) {
  if (any(x)) max(which(x)) else 1
}

qs <- function(x, w, p){
  n <- length(x)
  ii <- order(x)
  x <- x[ii]
  cumw <- cumsum(w[ii])

  pos <- last(cumw <= p*sum(w))
  posnext <- if (pos == n) pos else pos+1

  list(qlow = x[pos], qup = x[posnext],
       ilow = pos, iup = posnext,
       wlow = p - cumw[pos]/sum(w), wup = cumw[posnext]/sum(w) - p)
}
```

Type1: the smallest x such that $F(x) \geq p$.

```
qdata <- qs(x, w, p)
if (qdata$wlow == 0) qdata$qlow else qdata$qup
```

Type 2: $q_{\text{low}} = \hat{F}^{-1}(p)$ if $\hat{F}(q_{\text{low}}) = p$ and otherwise is the the average of $\hat{F}^{-1}(p)$ and the next higher observation.

```
qdata <- qs(x, w, p)
if (qdata$wlow == 0) (qdata$qlow + qdata$qup)/2 else qdata$qup
```

Type 3: whichever of $\hat{F}^{-1}(p)$ and the next higher observation is at an even-numbered position when the distinct data values are sorted.

```
w <- rowsum(w, x)
x <- sort(unique(x))
qdata <- qs(x, w, p)
if ((qdata$wlow == 0) && (qdata$ilow %% 2 == 0)) qdata$qlow else qdata$qup
```

Some errors?

```
# Example data
source("../R/qrule.R") # error?

x <- c(1, 3, 4, 7, 9, 10.5)
w <- c(0.1, 0.1, 0.2, 0.3, 0.1, 0.2)
ps <- seq(0.05, 0.99, 0.05)

cbind(p = ps,
      Type1 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 1)),
      Type2 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 2)),
      Type3 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 3)))
```

```
         p Type1 Type2 Type3
 [1,] 0.05   3.0   3.0   3.0
 [2,] 0.10   1.0   2.0   3.0
 [3,] 0.15   3.0   3.0   3.0
 [4,] 0.20   3.0   3.5   3.0
 [5,] 0.25   4.0   4.0   4.0
 [6,] 0.30   4.0   4.0   4.0
 [7,] 0.35   4.0   4.0   4.0
 [8,] 0.40   4.0   5.5   7.0
 [9,] 0.45   7.0   7.0   7.0
[10,] 0.50   7.0   7.0   7.0
[11,] 0.55   7.0   7.0   7.0
[12,] 0.60   7.0   7.0   7.0
[13,] 0.65   7.0   7.0   7.0
[14,] 0.70   9.0   9.0   9.0
[15,] 0.75   9.0   9.0   9.0
[16,] 0.80  10.5  10.5  10.5
[17,] 0.85  10.5  10.5  10.5
[18,] 0.90  10.5  10.5  10.5
[19,] 0.95  10.5  10.5  10.5
```

3.2 Continuous quantiles

For the plotting position, it is redefined in terms of the cumulative weights $W_{(k)} = \sum_{i \le k} w_{(i)}$, the total weight $W_{(n)}$, and the weight $w_{(k)}$ on the kth observation, as shown in the table below (**typos fixed**).

The following weighted quantiles reduce to unweighted cases for equal weights settings.

Method	Hyndman & Fan	Weighted quantiles
Type 4	$p_k = k/n$	$p_k = W_{(k)}/W_{(n)}$
Type 5	$p_k = (k-1/2)/n$	$p_k = \left(W_{(k)} - w_{(k)}/2\right)/W_{(n)}$
Type 6	$p_k = k/(n+1)$	$p_k = W_{(k)}/\left(W_{(n)} + w_{(n)}\right)$
Type 7	$p_k = (k-1)/(n-1)$	$p_k = W_{(k-1)}/W_{(n-1)}$
Type 8	$p_k = (k-1/3)/(n+1/3)$	$p_k = \left(W_{(k)} - w_{(k)}/3\right)/\left(W_{(n)} + w_{(n)}/3\right)$
Type 9	$p_k = (k-3/8)/(n+1/4)$	$p_k = \left(W_{(k)} - 3w_{(k)}/8\right)/\left(W_{(n)} + w_{(n)}/4\right)$

```r
cbind(p = ps,
      Type4 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 4)),
      Type5 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 5)),
      Type6 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 6)),
      Type7 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 7)),
      Type8 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 8)),
      Type9 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 9))) |>
  round(2)
```

```
         p Type4 Type5 Type6 Type7 Type8 Type9
 [1,] 0.05  0.00  1.00  1.00  1.80  1.00  1.00
 [2,] 0.10  1.00  2.00  1.40  2.60  1.80  1.85
 [3,] 0.15  2.00  3.00  2.60  3.20  2.87  2.90
 [4,] 0.20  3.00  3.33  3.20  3.60  3.28  3.29
 [5,] 0.25  3.25  3.67  3.50  4.00  3.60  3.62
 [6,] 0.30  3.50  4.00  3.80  4.60  3.92  3.94
 [7,] 0.35  3.75  4.60  4.20  5.20  4.45  4.49
 [8,] 0.40  4.00  5.20  4.80  5.80  5.05  5.09
 [9,] 0.45  4.50  5.80  5.40  6.40  5.65  5.69
[10,] 0.50  5.00  6.40  6.00  7.00  6.25  6.29
[11,] 0.55  5.50  7.00  6.60  7.27  6.85  6.89
[12,] 0.60  6.00  7.50  7.40  7.53  7.48  7.49
[13,] 0.65  6.50  8.00  8.60  7.80  8.12  8.09
[14,] 0.70  7.00  8.50  9.30  8.07  8.76  8.69
[15,] 0.75  8.00  9.00  9.75  8.33  9.30  9.23
```

```
[16,] 0.80  9.00  9.50 10.20  8.60  9.78  9.72
[17,] 0.85  9.38 10.00 10.50  8.87 10.26 10.20
[18,] 0.90  9.75 10.50 10.50  9.30 10.50 10.50
[19,] 0.95 10.12 10.50 10.50  9.90 10.50 10.50
```

3.3 Shah & Vaish (Shah and Vaish 2006)

Define $w^*_{(i)} = w_{(i)}n/W_{(n)}$, so that $w^*_{(i)}$ sum to the sample size, and C^*k as partial sums of $w^*_{(k)}$.

The estimated CDF is defined by

$$\hat{F}\left(x_k\right) = \frac{1}{n+1}\left(W^*_{(k)} + 1/2 - w_{(k)}/2\right).$$

It is related to **Type 6**.

4 Weighted quantiles - Matthew Kay

The `weighted_quantile` function used in the **ggdist** R package and described in the blog.

4.1 Source code

```
# In the weighted_quantile function
if (1 <= type && type <= 3) {
  # discontinuous quantiles
  switch(type,
         # type 1
         stepfun(F_x, c(x, x[length(x)]), right = TRUE),
         # type 2
         {
           x_over_2 = c(x, x[length(x)])/2
           inverse_cdf_type2_left = stepfun(F_x, x_over_2, right = FALSE)
           inverse_cdf_type2_right = stepfun(F_x, x_over_2, right = TRUE)
           function(x) inverse_cdf_type2_left(x) + inverse_cdf_type2_right(x)
         },
         # type 3
         stepfun(F_x - f_x/2, c(x[[1]], x), right = TRUE)
  )
} else {
  # Continuous quantiles
  p_k = switch(type - 3,
               # type 4
               F_x,
               # type 5
               F_x - f_x/2,
               # type 6
               F_x / (1 + f_x),
               # type 7
               (F_x - f_x) / (1 - f_x),
               # type 8
               (F_x - f_x/3) / (1 + f_x/3),
               # type 9
               (F_x - f_x*3/8) / (1 + f_x/4)
  )
  approxfun(p_k, x, rule = 2, ties = "ordered")
}
```

4.2 Continuous quantiles

Per Hyndman and Fan (1996), the unweighted quantile function is that of a piecewise linear interpolation between the points $\left[p_k, X_{(k)}\right]$, with specific definitions of p_k for each quantile type depending on its definition of m:

$$p_k = \frac{k-\alpha}{n-\alpha-\beta+1} m = \alpha + p(1-\alpha-\beta)$$

- The formulas for p_k in the unweighted case are written in terms of k and n.
- The **survey** package obtains p_k in terms of the cumulative weights $W_{(k)} = \sum_{i \le k} w_{(i)}$, the total weight $W_{(n)}$, and the weight $w_{(k)}$ on the kth observation.
- The **ggdist** package here obtains p_k in terms of the $W_{(k)}$ and $w_{(k)}$.

The following two identities hold in the unweighted case:

$$n = \frac{1}{w_{(k)}} \text{ and } k = nW_{(k)} = \frac{W_{(k)}}{w_{(k)}}.$$

So, we have

$$p_k = \frac{k-\alpha}{n-\alpha-\beta+1} = \frac{\frac{W_{(k)}}{w_{(k)}} - \alpha}{\frac{1}{w_{(k)}} - \alpha - \beta + 1} = \frac{W_{(k)} - w_{(k)}\alpha}{1 + w_{(k)}(1-\alpha-\beta)}.$$

It is slightly different from the weighted quantile method in the survey package.

```
cbind(p = ps,
      Type4 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 4)),
      Type4_ = ggdist::weighted_quantile(x, ps, w, type = 4),
      Type5 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 5)),
      Type5_ = ggdist::weighted_quantile(x, ps, w, type = 5),
      Type6 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 6)),
      Type6_ = ggdist::weighted_quantile(x, ps, w, type = 6),
      Type7 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 7)),
      Type7_ = ggdist::weighted_quantile(x, ps, w, type = 7),
      Type8 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 8)),
      Type8_ = ggdist::weighted_quantile(x, ps, w, type = 8),
      Type9 = sapply(ps, function(p) weighted.quantile.qrule(x, w, p, type = 9)),
      Type9_ = ggdist::weighted_quantile(x, ps, w, type = 9)) |>
  round(2)
```

```
         p Type4 Type4_ Type5 Type5_ Type6 Type6_ Type7 Type7_ Type8 Type8_
 [1,] 0.05  0.00   1.00  1.00   1.00  1.00   1.00  1.80   1.90  1.00   1.00
 [2,] 0.10  1.00   1.00  2.00   2.00  1.40   1.20  2.60   2.80  1.80   1.73
 [3,] 0.15  2.00   2.00  3.00   3.00  2.60   2.30  3.20   3.28  2.87   2.77
 [4,] 0.20  3.00   3.00  3.33   3.33  3.20   3.12  3.60   3.64  3.28   3.26
 [5,] 0.25  3.25   3.25  3.67   3.67  3.50   3.45  4.00   4.00  3.60   3.59
 [6,] 0.30  3.50   3.50  4.00   4.00  3.80   3.78  4.60   4.47  3.92   3.92
 [7,] 0.35  3.75   3.75  4.60   4.60  4.20   4.24  5.20   4.93  4.45   4.48
 [8,] 0.40  4.00   4.00  5.20   5.20  4.80   4.97  5.80   5.40  5.05   5.13
 [9,] 0.45  4.50   4.50  5.80   5.80  5.40   5.71  6.40   5.87  5.65   5.77
[10,] 0.50  5.00   5.00  6.40   6.40  6.00   6.44  7.00   6.33  6.25   6.41
[11,] 0.55  5.50   5.50  7.00   7.00  6.60   7.12  7.27   6.80  6.85   7.05
[12,] 0.60  6.00   6.00  7.50   7.50  7.40   7.65  7.53   7.28  7.48   7.56
[13,] 0.65  6.50   6.50  8.00   8.00  8.60   8.18  7.80   7.76  8.12   8.06
[14,] 0.70  7.00   7.00  8.50   8.50  9.30   8.71  8.07   8.25  8.76   8.57
[15,] 0.75  8.00   8.00  9.00   9.00  9.75   9.32  8.33   8.73  9.30   9.09
[16,] 0.80  9.00   9.00  9.50   9.50 10.20  10.03  8.60   9.15  9.78   9.65
[17,] 0.85  9.38   9.38 10.00  10.00 10.50  10.50  8.87   9.49 10.26  10.22
[18,] 0.90  9.75   9.75 10.50  10.50 10.50  10.50  9.30   9.83 10.50  10.50
[19,] 0.95 10.12  10.12 10.50  10.50 10.50  10.50  9.90  10.16 10.50  10.50
      Type9 Type9_
 [1,]  1.00   1.00
 [2,]  1.85   1.80
 [3,]  2.90   2.83
 [4,]  3.29   3.27
 [5,]  3.62   3.61
 [6,]  3.94   3.94
 [7,]  4.49   4.51
 [8,]  5.09   5.15
 [9,]  5.69   5.78
[10,]  6.29   6.41
[11,]  6.89   7.04
[12,]  7.49   7.54
[13,]  8.09   8.05
[14,]  8.69   8.56
[15,]  9.23   9.07
[16,]  9.72   9.61
[17,] 10.20  10.16
[18,] 10.50  10.50
[19,] 10.50  10.50
```

4.3 Create flat regions

To satisfy the property that $\hat{Q}_{X,i}(0.5)$ should be equal to the sample median.

1. Estimate the sample size using Kish's effective sample size.
2. Construct a modified dataset with replications proportional to weights --- Introduce flat regions.

The weighted quantiles now match the unweighted quantiles much more closely. I think this strategy helps a lot.

5 Comparison

5.1 Three requirements proposed in Akinshin (2023)

```r
x <- c(0, 1, 100)
wE <- c(1, 1, 1)
wA <- c(1, 0, 1); nz <- wA != 0
wB <- c(1, 0.00001, 1)

kess <- function(w) sum(w)^2 / sum(w^2)
p <- 0.9

# R1: consistency with existing quantile estimators
```

```r
sapply(1:9, function(type){
  c(quantile = quantile(x, probs = p, type),
    survey = weighted.quantile.qrule(x, w = wE, p = p, type),
    ggdist = ggdist::weighted_quantile(x, weights = wE,
                                       probs = p, type),
    ggdist_kess = ggdist::weighted_quantile(x, weights = wE,
                                            probs = p, type, n = kess))
}, simplify = TRUE, USE.NAMES = TRUE) |> t()
```

```
      quantile.90% survey ggdist ggdist_kess
 [1,]         80.2  100.0   80.2        80.2
 [2,]         80.2  100.0   80.2        80.2
 [3,]         80.2  100.0   80.2        80.2
 [4,]         80.2   70.3  100.0        80.2
 [5,]         80.2  100.0  100.0        80.2
 [6,]         80.2  100.0  100.0        80.2
 [7,]         80.2   80.2  100.0        80.2
 [8,]         80.2  100.0  100.0        80.2
 [9,]         80.2  100.0  100.0        80.2
```

```r
## If Kish's effective sample size is not used in ggdist, R1 is violated.

# R2: zero weight support
sapply(1:9, function(type){
  c(quantile = quantile(x[nz], probs = p, type),
    survey = weighted.quantile.qrule(x, w = wA, p = p, type),
    survey_nz = weighted.quantile.qrule(x[nz], w = wA[nz], p = p, type),
    ggdist = ggdist::weighted_quantile(x, weights = wA, probs = p, type),
    ggdist_nz = ggdist::weighted_quantile(x[nz], weights = wA[nz],
                                          probs = p, type),
    ggdist_kess = ggdist::weighted_quantile(x, weights = wA, probs = p, type, n = kess),
    ggdist_nz_kess = ggdist::weighted_quantile(x[nz], weights = wA[nz],
                                               probs = p, type, n = kess))
}, simplify = TRUE, USE.NAMES = TRUE) |> t()
```

```
      quantile.90% survey survey_nz ggdist ggdist_nz ggdist_kess ggdist_nz_kess
 [1,]           90    100       100     90        90          90             90
 [2,]           90    100       100     90        90          90             90
 [3,]           90    100       100    100       100          90             90
 [4,]           90     80        80    100       100          90             90
```

```
[5,]           90    100       100    100       100          90             90
[6,]           90    100       100    100       100          90             90
[7,]           90     90        90    100       100          90             90
[8,]           90    100       100    100       100          90             90
[9,]           90    100       100    100       100          90             90
```

```r
# R3: stability
p <- 0.5
sapply(1:9, function(type){
  c(quantile = quantile(x[nz], probs = p, type),
    survey_wA = weighted.quantile.qrule(x, w = wA, p = p, type),
    survey_wB = weighted.quantile.qrule(x, w = wB, p = p, type),
    ggdist_wA = ggdist::weighted_quantile(x, weights = wA, probs = p, type),
    ggdist_wB = ggdist::weighted_quantile(x, weights = wB, probs = p, type),
    ggdist_wA_kess = ggdist::weighted_quantile(x, weights = wA, probs = p, type, n = kess)
    ggdist_wB_kess = ggdist::weighted_quantile(x, weights = wB, probs = p, type, n = kess)
}, simplify = TRUE, USE.NAMES = TRUE) |> t() |> round(2)
```

```
     quantile.50% survey_wA survey_wB ggdist_wA ggdist_wB ggdist_wA_kess
[1,]           50         0      1.00        50         1             50
[2,]           50        50      1.00        50         1             50
[3,]           50       100      1.00        50         1             50
[4,]           50         0      0.50        50         1             50
[5,]           50        50      1.00        50         1             50
[6,]           50        50     50.50        50         1             50
[7,]           50        50      0.50        50         1             50
[8,]           50        50     25.75        50         1             50
[9,]           50        50     20.80        50         1             50
     ggdist_wB_kess
[1,]              1
[2,]              1
[3,]              1
[4,]              1
[5,]              1
[6,]              1
[7,]              1
[8,]              1
[9,]              1
```

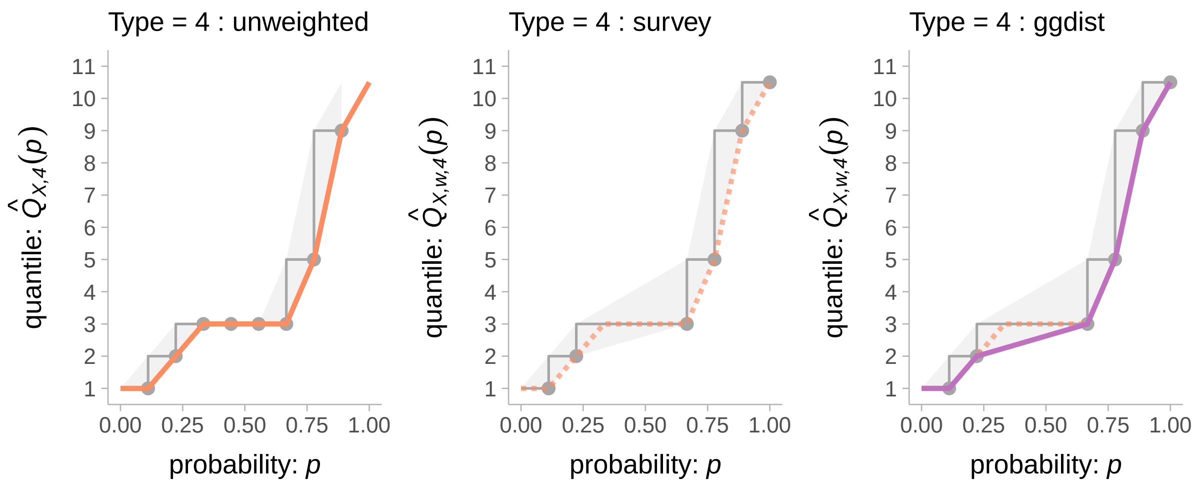
The stability property is not satisfied in either the survey or ggdist packages, but it is not common for neighboring ordered statistics (conformity scores) to differ significantly in CP problems.

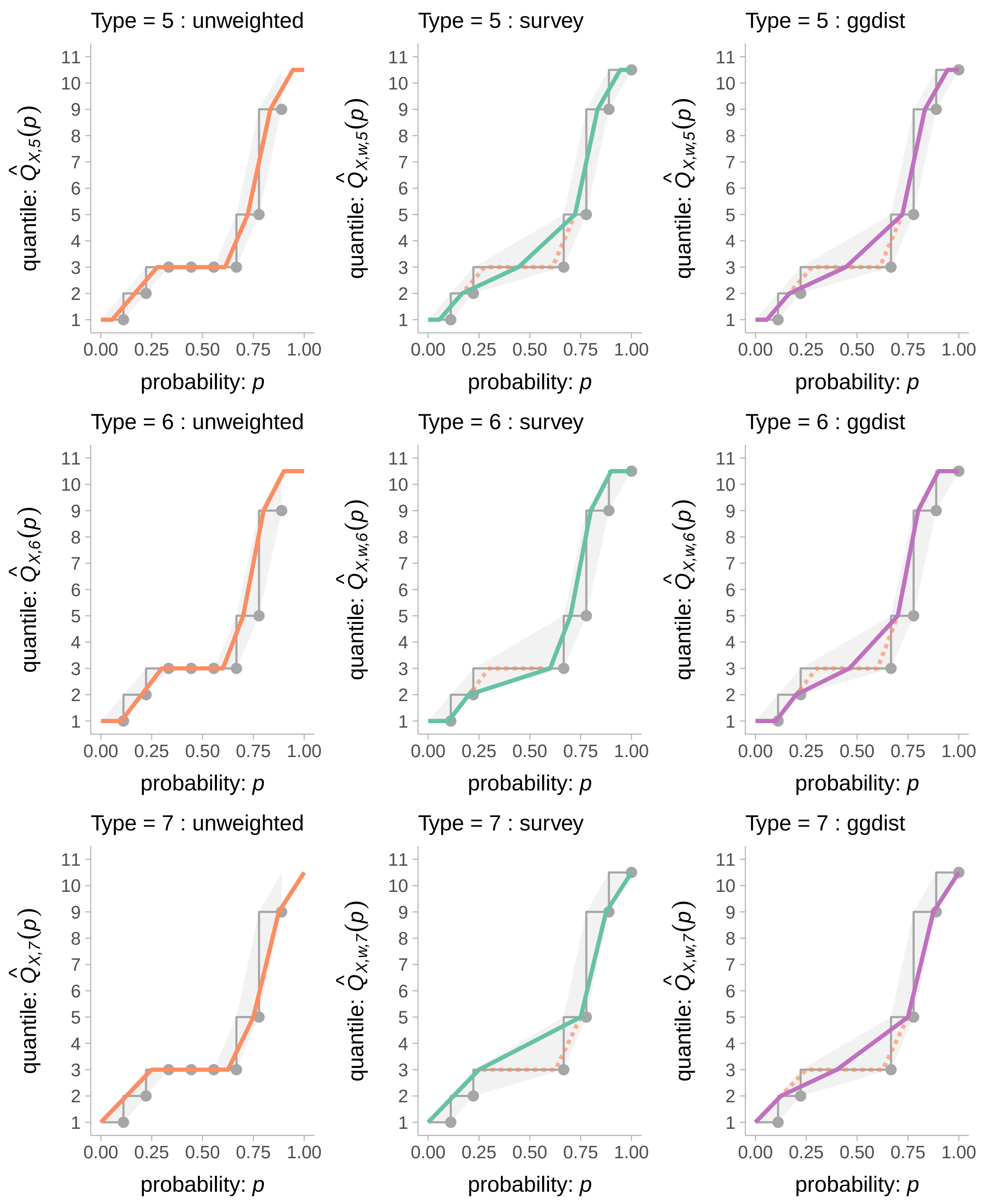
Maybe we can try to propose some locally smooth method to address it.

5.2 Non-weighted VS weighted quantiles

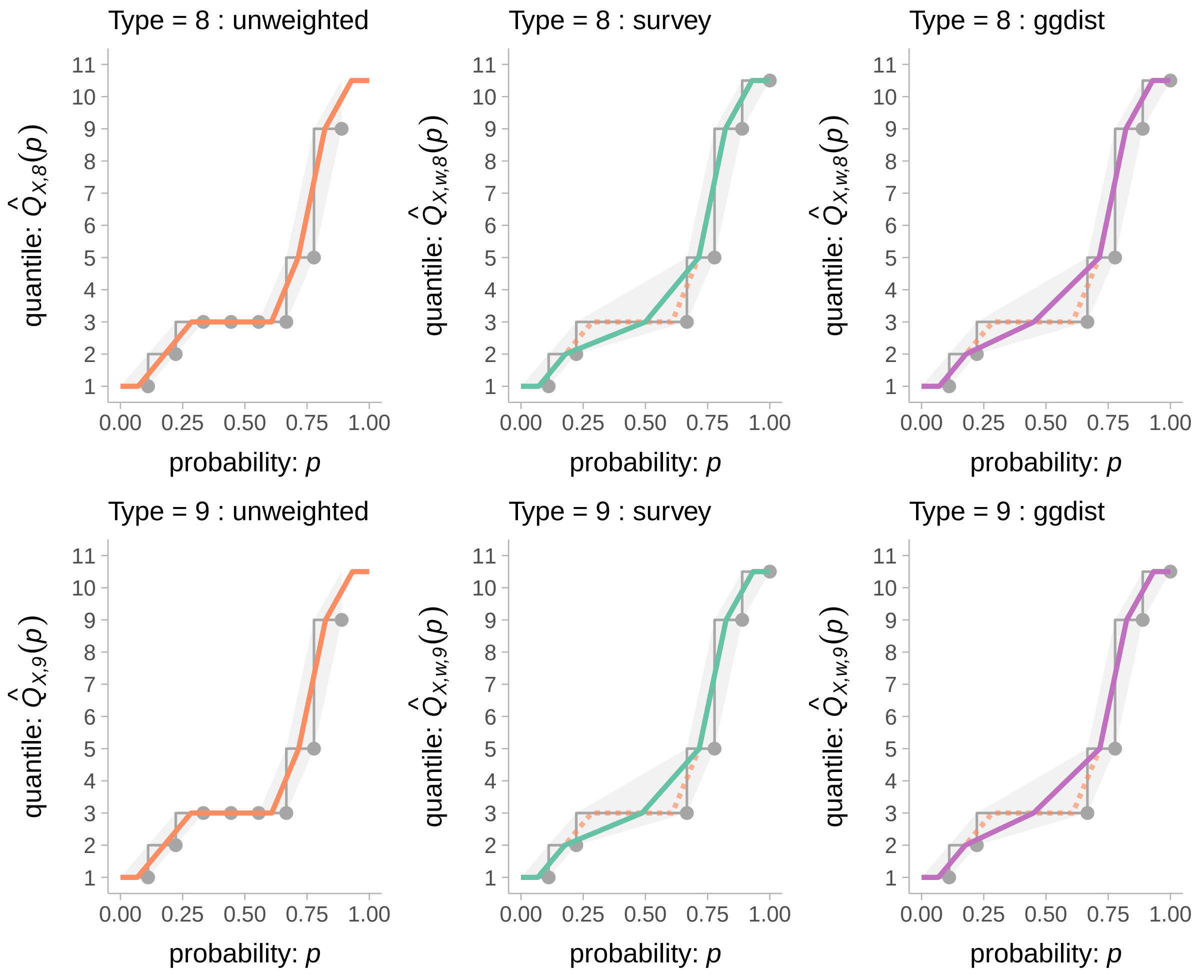
```
X_star <- c(1,2,3,3,3,3,5,9,10.5)
w_star <- rep(1, length(X_star)) / length(X_star)
W_star <- cumsum(w_star)

X <- unique(X_star)
w <- as.vector(prop.table(table(X_star)))
W <- cumsum(w)
tibble(X, w, W)
```

```
# A tibble: 6 x 3
      X     w     W
  <dbl> <dbl> <dbl>
1   1   0.111 0.111
2   2   0.111 0.222
3   3   0.444 0.667
4   5   0.111 0.778
5   9   0.111 0.889
6  10.5 0.111 1
```





Type = 5 : unweighted
Type = 5 : survey
Type = 5 : ggdist
Type = 6 : unweighted
Type = 6 : survey
Type = 6 : ggdist
Type = 7 : unweighted
Type = 7 : survey
Type = 7 : ggdist
quantile: $\hat{Q}_{X,5}(p)$
quantile: $\hat{Q}_{X,w,5}(p)$
quantile: $\hat{Q}_{X,6}(p)$
quantile: $\hat{Q}_{X,w,6}(p)$
quantile: $\hat{Q}_{X,7}(p)$
quantile: $\hat{Q}_{X,w,7}(p)$
probability: p
0.00 0.25 0.50 0.75 1.00
1 2 3 4 5 6 7 8 9 10 11



The **survey** and **ggdist** packages use different strategies for defining p_k for continuous quantiles (Types 4-9), the quantile outputs are slightly different, but still always within the reasonable bounds.

Which strategy makes more theoretical sense? We should explore this further.

5.3 Impact of Inf

```
n <- 500
x <- c(rnorm(n, 0, 1), Inf)
x_order <- order(x)
x <- x[x_order]

w <- c(0.99^(n+1-((1:n))), 1)
```

```r
w <- w/sum(w)
W <- cumsum(w)

# Data example
tibble(x, w, W) |> tail(10)

# 90% weighted quantile for a set including Inf
sapply(1:9, function(type){
  c(survey = weighted.quantile.qrule(x, w = w, p = 0.9, type),
    ggdist = ggdist::weighted_quantile(x, weights = w, probs = 0.9, type),
    ggdist_kess = ggdist::weighted_quantile(x, weights = w, probs = 0.9, type, n = kess))
}, simplify = TRUE, USE.NAMES = TRUE) |> t()

# Harrell-Davis quantile estimator & Type7 in Akinshin (2023)
paste(cNORM::weighted.quantile.harrell.davis(x, 0.9, w), "|",
      cNORM::weighted.quantile.type7(x, 0.9, w))
```

```
# A tibble: 10 x 3
        x       w     W
    <dbl>   <dbl> <dbl>
 1   1.80 0.00919 0.913
 2   1.83 0.00929 0.922
 3   2.01 0.00938 0.932
 4   2.03 0.00948 0.941
 5   2.10 0.00957 0.951
 6   2.20 0.00967 0.960
 7   2.39 0.00977 0.970
 8   2.71 0.00987 0.980
 9   3.13 0.00996 0.990
10 Inf    0.0101  1.00
        survey   ggdist ggdist_kess
 [1,] 1.797526 1.790365    1.797526
 [2,] 1.797526 1.790365    1.797526
 [3,] 1.797526 1.790365    1.797526
 [4,] 1.788085 1.790365    1.797526
 [5,] 1.797759 1.790365    1.797526
 [6,] 1.799074 1.790365    1.797526
 [7,] 1.788101 1.790365    1.797526
 [8,] 1.798199 1.790365    1.797526
 [9,] 1.798089 1.790365    1.797526
[1] "Inf | NaN"
```

6 Summary

- The `weighted_quantile` function in the **ggdist** R package seems to be a good choice in the context of conformal prediction.
- To satisfy the property that $\hat{Q}_{X,i}(0.5)$ is equal to the sample median, we suggest using **Kish's effective sample size** to create flat regions and make the output quantiles match the unweighted quantiles more closely. Using Kish's effective sample size is more likely to return quantile estimator satisfying the R1 consistency property.

```
# Function to calculate Kish's effective sample size
kess <- function(w) sum(w)^2 / sum(w^2)
```

- **Type 8** may be a good choice as it gives median-unbiased estimates in the equal weight settings, even if all types may give the same result for large sample size.

7 References

Akinshin, Andrey. 2023. "Weighted Quantile Estimators." *arXiv Preprint arXiv:2304.07265*.

Hyndman, Rob J, and Yanan Fan. 1996. "Sample Quantiles in Statistical Packages." *The American Statistician* 50 (4): 361–65.

Shah, Babubhai V, and Akhil K Vaish. 2006. "Confidence Intervals for Quantile Estimation from Complex Survey Data." In *Proceedings of the Section on Survey Research Methods*.