Measuring the similarity of forecasts provided in a quantile format

Johannes Bracher, johannes.bracher@kit.edu

(Please excuse that this document contains a lot of hand waving. I can write things up more formally if this becomes of interest.)

Short version

Consider two predictive distributions F and G. Their *Cramer distance* or *integrated quadratic distance* is defined as

$$\text{CD}(F, G) = \int_{-\infty}^{\infty} (F(x) - G(x))^2 dx$$

where $F(x)$ and $G(x)$ denote the cumulative distribution functions. The Cramer distance is the divergence associated with the continuous ranked probability score (Thorarinsdottir 2013, Gneiting and Raftery 2007)

$$\text{CRPS}(F, y) = \int_{-\infty}^{\infty} (F(x) - \mathbf{1}(x \geq y))^2 dx,$$

where y denotes the observed value. Indeed, it is a generalization of the CRPS in the sense that it simplifies to the CRPS if one out of F and G is a one-point distribution. The Cramer distance is commonly used to measure the similarity of forecast distributions (see Richardson et al 2020 for a recent application). To evaluate it using samples from the distributions F and G one can use the alternative formulation

$$\text{CD}(F, G) = \mathbb{E}_{F,G}|X - Y| - 0.5\left[\mathbb{E}_F|X - X'| + \mathbb{E}_G|Y - X'|\right], \tag{1}$$

where x, x' are independent random variables following F and y, y' are independent random variables following G. This formulation illustrates that the Cramer distance depends on the shift between F and G (first term) and the variability of both F and G (of which the two last expectations in above equation are a measure).

Now assume that for each of the distributions F and G we only know K quantiles at equally spaced levels $1/(K + 1), 2/(K + 1), \dots, K/(K + 1)$. Denote these quantiles by $q_1^F, \dots, q_K^F$ and $q_1^G, \dots, q_K^G$, respectively. It is well known that the CRPS can be approximated by an average of linear quantile scores (Laio and Tamea 2007, Gneiting and Raftery 2007):

$$\text{CRPS}(F, y) \approx \frac{1}{K} \times \sum_{k=1}^{K} 2\{\mathbf{1}(y \leq q_k^F)\} \times (q_k^F - y). \tag{2}$$

This approximation is equivalent to the weighted interval score (WIS) which is in use for evaluation of quantile forecasts at the Forecast Hub, see Section 2.2 of Bracher et al (2021). This approximation can be generalized to the Cramer distance as

$$\text{CD}(F, G) \approx \frac{1}{K(K + 1)} \times \sum_{i=1}^{2K-1} b_i(b_i + 1)(q_{i+1} - q_i), \tag{3}$$

where we use the following notation:

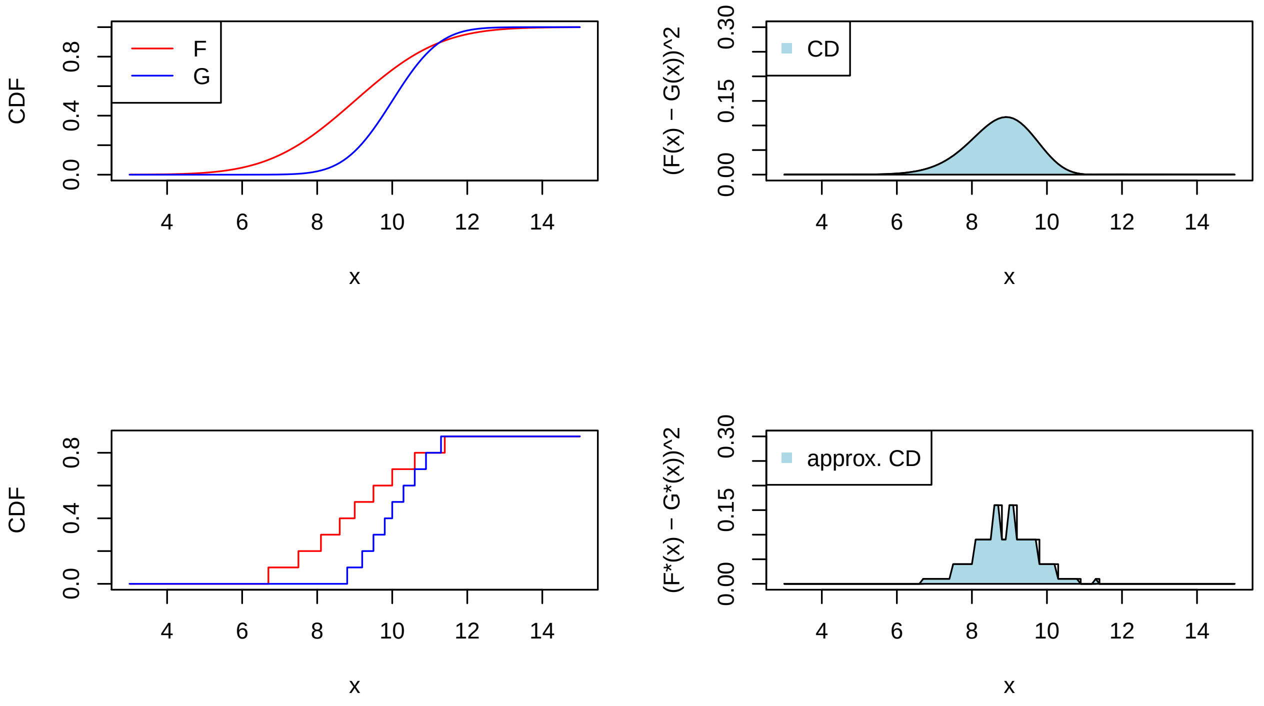
- $\mathbf{q}$ is a vector of length $2K$. It is obtained by pooling the $q_k^F, q_k^G, k = 1, \dots, K$ and ordering them in increasing order (ties can be ordered in an arbitrary manner).

- **a** is a vector of length $2K$ containing the value 1 wherever **q** contains a quantile of F and -1 wherever it contains a value of G.
- **b** is a vector of length $2K$ containing the absolute cumulative sums of **a**, i.e. $b_i = \left|\sum_{j=1}^{i} a_j\right|$.

For small K it seems that the slightly different approximation

$$\text{CD}(F, G) \approx \frac{1}{(K+1)^2} \times \sum_{i=1}^{2K-1} b_i^2 (q_{i+1} - q_i), \tag{4}$$

actually works better. This just corresponds to the integrated squared difference between two step functions F^* and G^* with $F^*(x) = 0$ for $x < q_1^F$, $F^*(x) = k/(K+1)$ for $q_k^F \leq x < q_k^F$, $F^*(x) = K/(K+1)$ for $x \geq x_K^F$ and G^* defined accordingly. We illustrate this in the figure below, with light blue areas representing the CD and approximated CD.



Examples

As an example we apply the approximation to the distributions F and G already used in the figures above. They are given by $N(9, 1.8)$ for F and $N(10, 1)$ for G.

```r
# define distributions:
mu_F <- 9
sigma_F <- 1.8

mu_G <- 10
sigma_G <- 1
```

We approximate the CD in four different ways:

- Using direct numerical integration based on a fine grid of values for x.

```r
# simple numerical integration using grid:
grid_x <- seq(from = 3, to = 15, by = 0.1)
p_F <- pnorm(grid_x, mu_F, sigma_F)
```

```
p_G <- pnorm(grid_x, mu_G, sigma_G)
(cd_grid <- 0.1*sum((p_F - p_G)^2))
```

```
## [1] 0.2532376
```

- Using sampling and the alternative expression (1) of the CD from above:

```
n <- 100000
set.seed(123)
x <- rnorm(n, mu_F, sigma_F)
x_dash <- rnorm(n, mu_F, sigma_F)

y <- rnorm(n, mu_G, sigma_G)
y_dash <- rnorm(n, mu_G, sigma_G)

(cd_sample <- (mean(abs(x - y)) - 0.5*(mean(abs(x - x_dash)) + mean(abs(y - y_dash)))))
```

```
## [1] 0.2457156
```

- Using the first quantile-based approximation (3) and various values of K:

```
# q_F: vector containing the (1:K)/(K + 1) quantiles of F
# q_G: vector containing the (1:K)/(K + 1) quantiles of G
approx_cd1 <- function(q_F, q_G){

  # compute quantile levels from length of provided quantile vectors:
  K <- length(q_F)
  if(length(q_G) != K) stop("q_F and q_G need to be of the same length")
  p <- (1:K)/(K + 1) # function assumes that the quantile levels are equally spaced

  # pool quantiles:
  q0 <- c(q_F, q_G)
  # vector of grouping variables, with 1 for values belonging to F, -1 for values
  # belonging to G
  a0 <- c(rep(1, length(q_F)), rep(-1, length(q_G)))

  # re-order both vectors:
  q <- q0[order(q0)]
  a <- a0[order(q0)]
  # and compute "how many quantiles ahead" F or G is at a given segment:
  b <- abs(cumsum(a))

  # compute the lengths of segments defined by sorted quantiles:
  diffs_q <- c(diff(q), 0) # zero necessary for indexing below, but we could put
  # anything (gets multiplied w zero)

  # and approximate CD
  cvm <- sum(diffs_q*b*(b + 1))/(K + 1)/(K)

  return(mean(cvm))
}

# values of K to check:
values_K <- c(10, 20, 50, 100, 200, 500, 1000, 2000)

# vector to store results:
```

```r
cd_approx1 <- numeric(length(values_K))

# apply approximation for each_
for(i in seq_along(values_K)){
  p_temp <- (1:(values_K[i]))/(values_K[i] + 1) # quantile levels
  q_F_temp <- qnorm(p_temp, mu_F, sigma_F) # quantiles of F
  q_G_temp <- qnorm(p_temp, mu_G, sigma_G) # quantiles of G
  cd_approx1[i] <- approx_cd1(q_F = q_F_temp, q_G = q_G_temp) # approximation
}

cd_approx1
```

```
## [1] 0.3550788 0.3078906 0.2764153 0.2652018 0.2593619 0.2557450 0.2545077
## [8] 0.2538792
```

- Using the second quantile-based approximation (4) and various values of K:

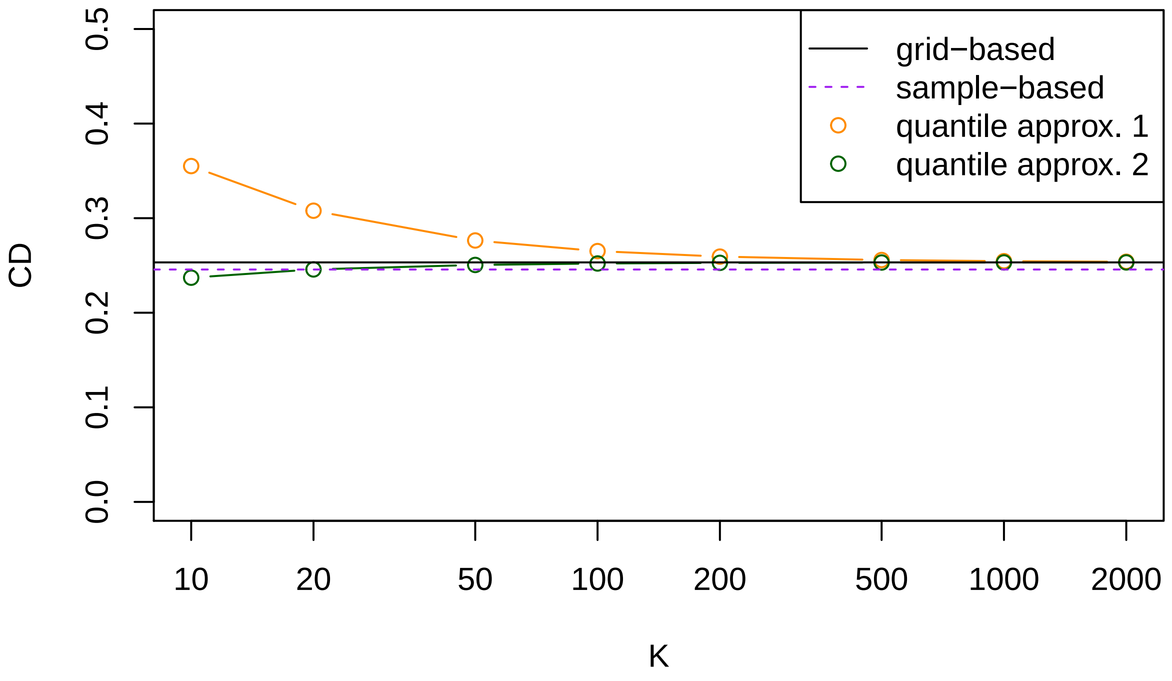
```r
# q_F: vector containing the (1:K)/(K + 1) quantiles of F
# q_G: vector containing the (1:K)/(K + 1) quantiles of G
approx_cd2 <- function(q_F, q_G){

  # compute quantile levels from length of provided quantile vectors:
  K <- length(q_F)
  if(length(q_G) != K) stop("q_F and q_G need to be of the same length")
  p <- (1:K)/(K + 1) # function assumes that the quantile levels are equally spaced

  # pool quantiles:
  q0 <- c(q_F, q_G)
  # vector of grouping variables, with 1 for values belonging to F, -1 for values
  # belonging to G
  a0 <- c(rep(1, length(q_F)), rep(-1, length(q_G)))

  # re-order both vectors:
  q <- q0[order(q0)]
  a <- a0[order(q0)]
  # and compute "how many quantiles ahead" F or G is at a given segment:
  b <- abs(cumsum(a))

  # compute the lengths of segments defined by sorted quantiles:
  diffs_q <- c(diff(q), 0) # zero necessary for indexing below, but we could put
  # anything (gets multiplied w zero)

  # and approximate CD
  cvm <- sum(diffs_q*b^2/(K + 1)^2)

  return(mean(cvm))
}

# values of K to check:
values_K <- c(10, 20, 50, 100, 200, 500, 1000, 2000)

# vector to store results:
cd_approx2 <- numeric(length(values_K))

# apply approximation for each_
```

```
for(i in seq_along(values_K)){
  p_temp <- (1:(values_K[i] - 1))/values_K[i] # quantile levels
  q_F_temp <- qnorm(p_temp, mu_F, sigma_F) # quantiles of F
  q_G_temp <- qnorm(p_temp, mu_G, sigma_G) # quantiles of G
  cd_approx2[i] <- approx_cd2(q_F = q_F_temp, q_G = q_G_temp) # approximation
}

cd_approx2
```

```
## [1] 0.2370715 0.2458022 0.2505461 0.2520862 0.2527531 0.2530874 0.2531764
## [8] 0.2532128
```

The below plot shows the results from the different computations.

```
# plot:
plot(values_K, cd_approx1, ylim = c(0, 0.5), xlab = "K", ylab = "CD",
     pch = 1, type = "b", log = "x", col = "darkorange")
lines(values_K, cd_approx2, type = "b", col = "darkgreen")
abline(h = cd_grid, col = "black")
abline(h = cd_sample, col = "purple", lty = 2)
legend("topright",
       c("grid-based", "sample-based", "quantile approx. 1", "quantile approx. 2"),
       pch = c(NA, NA, 1, 1), lty = c(1, 2, NA, NA),
       col = c("black", "purple", "darkorange", "darkgreen"))
```



While approximation (4) works better than approximation (3) in this example, this is not always the case:

```
# define another distribution:
mu_H <- 10
sigma_H <- 0.1

p_H <- pnorm(grid_x, mu_H, sigma_H)
(cd_grid <- 0.1*sum((p_F - p_H)^2))
```

```
## [1] 0.5822353
```

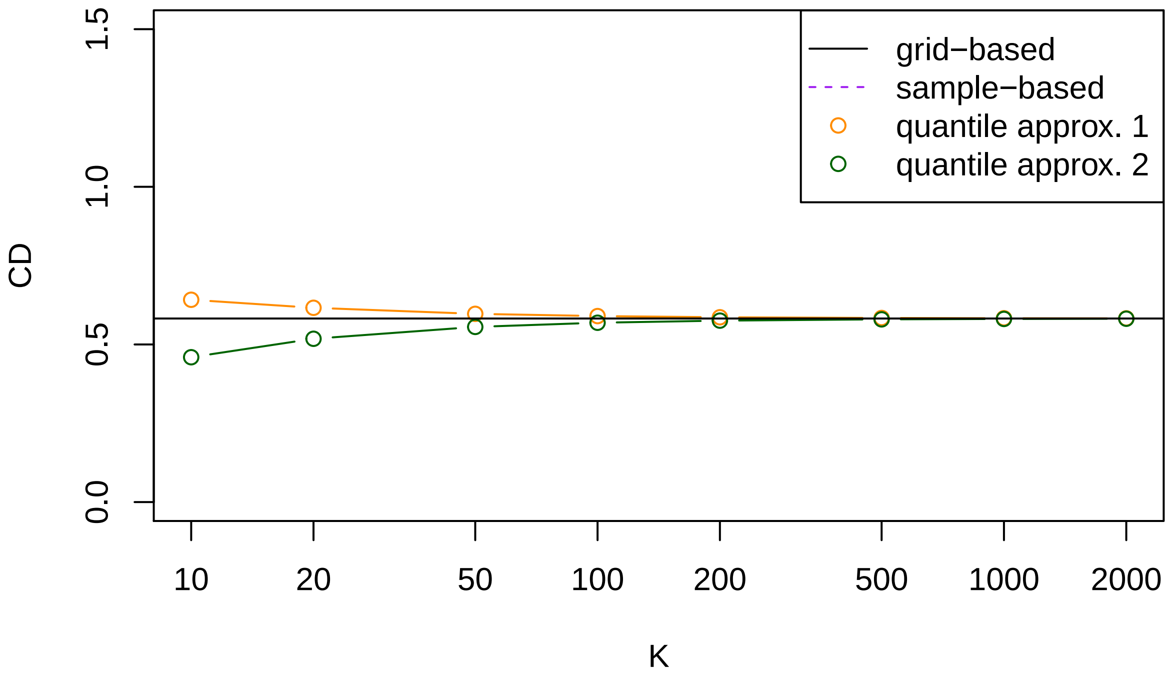
```r
# approximation 1
# vector to store results:
cd_approx1 <- numeric(length(values_K))

# apply approximation for each_
for(i in seq_along(values_K)){
  p_temp <- (1:(values_K[i]))/(values_K[i] + 1) # quantile levels
  q_F_temp <- qnorm(p_temp, mu_F, sigma_F) # quantiles of F
  q_H_temp <- qnorm(p_temp, mu_H, sigma_H) # quantiles of H
  cd_approx1[i] <- approx_cd1(q_F = q_F_temp, q_G = q_H_temp) # approximation
}

cd_approx1
```

```
## [1] 0.6417338 0.6162528 0.5971065 0.5900005 0.5862474 0.5838953 0.5830833
## [8] 0.5826676
```

```r
# approximation 2
# vector to store results:
cd_approx2 <- numeric(length(values_K))

# apply approximation for each_
for(i in seq_along(values_K)){
  p_temp <- (1:(values_K[i] - 1))/values_K[i] # quantile levels
  q_F_temp <- qnorm(p_temp, mu_F, sigma_F) # quantiles of F
  q_H_temp <- qnorm(p_temp, mu_H, sigma_H) # quantiles of G
  cd_approx2[i] <- approx_cd2(q_F = q_F_temp, q_G = q_H_temp) # approximation
}

cd_approx2
```

```
## [1] 0.4594666 0.5179726 0.5556011 0.5688302 0.5755465 0.5795848 0.5809226
## [8] 0.5815858
```

```r
# plot:
plot(values_K, cd_approx1, ylim = c(0, 1.5), xlab = "K", ylab = "CD",
     pch = 1, type = "b", log = "x", col = "darkorange")
lines(values_K, cd_approx2, type = "b", col = "darkgreen")
abline(h = cd_grid, col = "black")
legend("topright",
       c("grid-based", "sample-based", "quantile approx. 1", "quantile approx. 2"),
       pch = c(NA, NA, 1, 1), lty = c(1, 2, NA, NA),
       col = c("black", "purple", "darkorange", "darkgreen"))
```



So there is no approximation which is inherently better than the other. The advantage of approximation (3) is that it generalizes the WIS and reduces to it if G is just a point mass at y. We check via an example that approximation (3) then indeed coincides with (2):

```r
# define G_star
y <- 10
q_G_star <- rep(y, 9)

approx_cd1(q_F_10, q_G_star)# approximation 1
```

```
## [1] 0.6885672
```

```r
mean(2*((y <= q_F_10) - p_10)*(q_F_10 - y)) # WIS defined via quantile scores
```

```
## [1] 0.6885672
```

Preliminary suggestion for a decomposition (there is a separate document on this now, kept only to be able to check back)

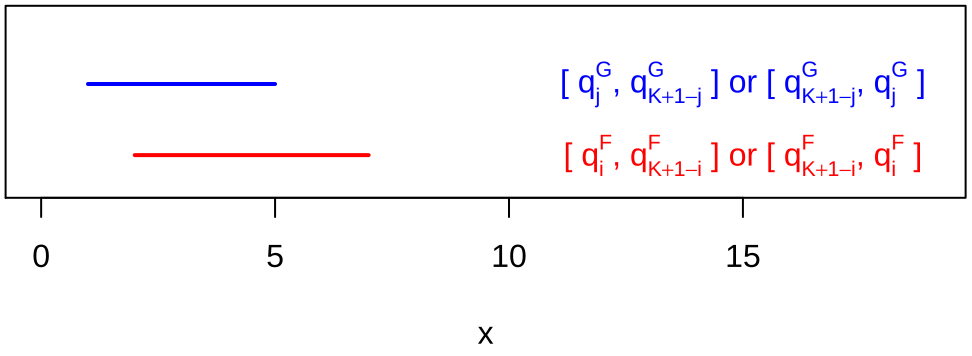
Can probably be generalized to "all pairs of intervals" not just symmetric ones. Should have nicer properties.

The first quantile-based approximation (4) can also be written as

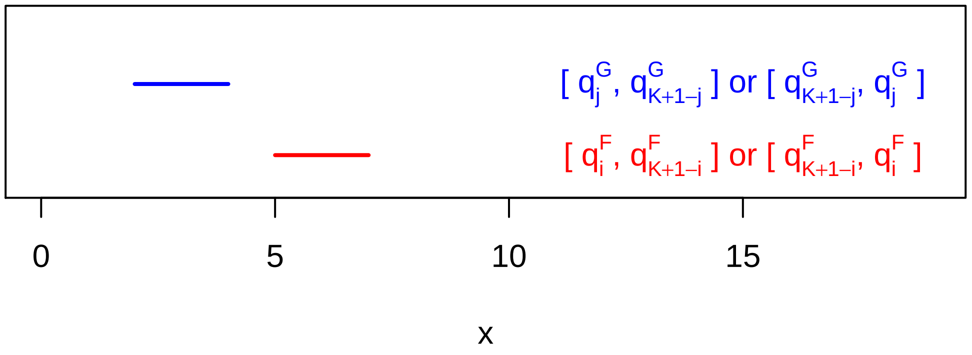
$$\text{CD}(F, G) \approx \frac{1}{K(K+1)} \sum_{i=1}^{K} \sum_{j=1}^{K} \mathbf{1}\{(i - j) \times (q_i^F - q_j^G) \leq 0\} \times \left|q_i^F - q_j^G\right|,$$

which sums up penalties for $q_i^F \geq q_j^G$ despite $i \leq j$ (or the other way around). For instance, if the 70% quantile of F is 100 and the 80% quantile of G is 60 this would result in a contribution of $40/K/(K+1)$ to the approximated Cramer distance. I haven't written this up properly, but this allows for a nice decomposition into terms representing shifts (F has larger/smaller quantiles than G) and differences in dispersion (F is more/less dispersed than G). The decomposition is based on comparisons of cental prediction intervals $[q_i^F, q_{K+1-i}^F]$ (or $[q_{K+1-i}^F, q_i^F]$, if $i > (K+1)/2$) and $[q_j^G, q_{K+1-j}^G]$ (or $[q_{K+1-j}^G, q_j^G]$), which potentially have different nominal coverage levels. I have not yet found a concise notation to describe this, but the intuition is the following:

- We count a term $\left|q_i^F - q_j^G\right|$ from the above equation as a contribution to the component "F larger than G" if in addition to $q_i^F \geq q_j^G$ despite $i \leq j$ the prediction intervals look like one of the below:

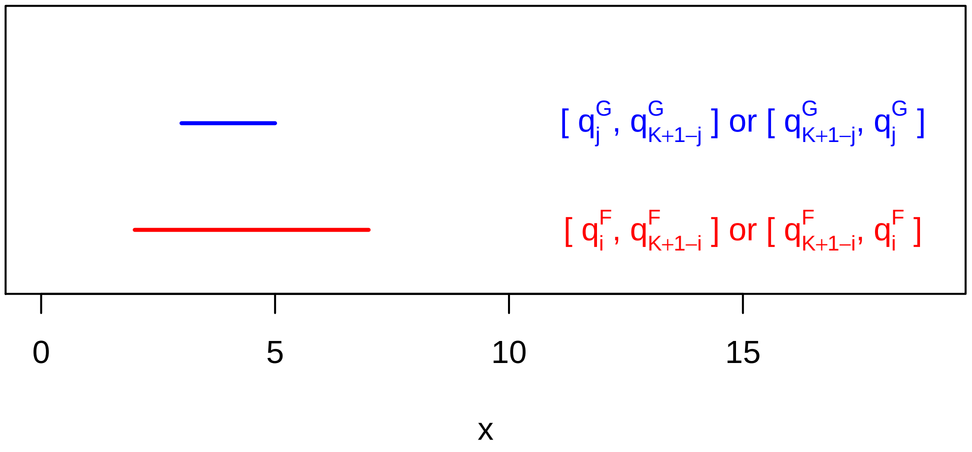


[q_j^G, q_{K+1−j}^G] or [q_{K+1−j}^G, q_j^G]
[q_i^F, q_{K+1−i}^F] or [q_{K+1−i}^F, q_i^F]
0
5
10
15
x



[q_j^G, q_{K+1−j}^G] or [q_{K+1−j}^G, q_j^G]
[q_i^F, q_{K+1−i}^F] or [q_{K+1−i}^F, q_i^F]
0
5
10
15
x

- A component "G larger than F" is defined accordingly.
- We count a term $\left|q_i^F - q_j^G\right|$ from the above equation as a contribution to the component "F more dispersed than G" if the respective interval for F has lower nominal coverage than that of G, but nonetheless completely includes the latter:



[q_j^G, q_{K+1−j}^G] or [q_{K+1−j}^G, q_j^G]
[q_i^F, q_{K+1−i}^F] or [q_{K+1−i}^F, q_i^F]
0
5
10
15
x

- A component "G more dispersed than F" is defined accordingly.

Note that in a sense this prioritizes shifts over differences in dispersion, and differences in dispersion are only counted if the CDFs cross. E.g., if $q_i^F \geq q_i^G$ for all i, all differences will be assigned to "F larger than G". This is the case even if one of the two distributions shows much larger dispersion than the other.

Interval comparison: Comparing $[l_F, u_F]$, $[l_G, u_G]$, where the former has lower nominal coverage.

$$d(F,G) = \underbrace{I(l_F < l_G < u_G < u_F) \times (u_F - l_F - u_G + l_G)}_{F \text{ more dispersed}} \tag{5}$$

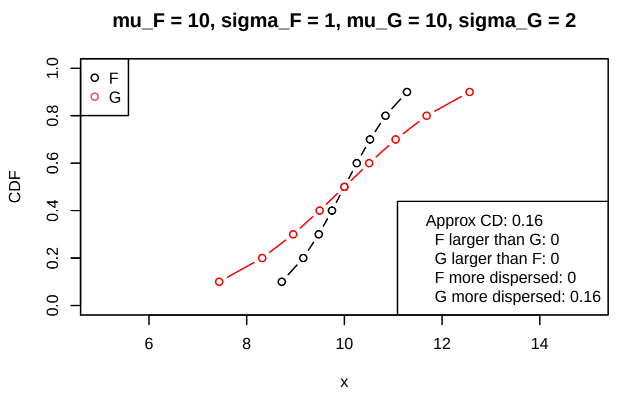
$$+ \underbrace{I(u_F < l_G) \times (2l_G - u_F - l_F) + I(l_F < l_G < u_F < u_G) \times (l_G - l_F)}_{G \text{ larger}} \tag{6}$$

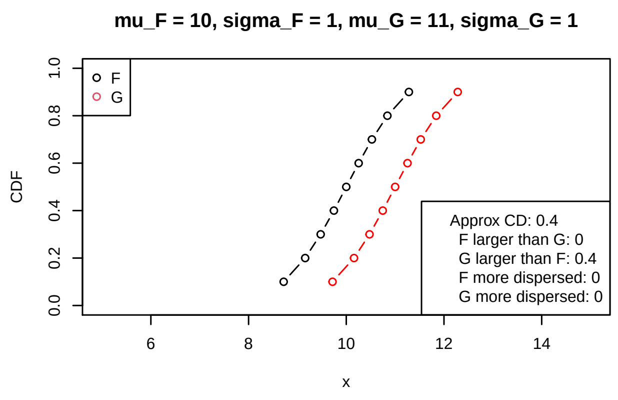
$$+ \underbrace{I(u_G < l_F) \times (u_F + l_F - 2u_G) + I(l_G < l_F < u_G < u_F) \times (u_F - u_G)}_{F \text{ larger}} \tag{7}$$

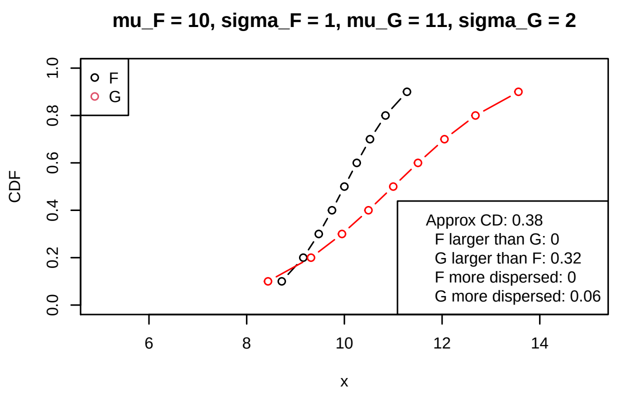
I wrote an R function which does the decomposition and a few examples.

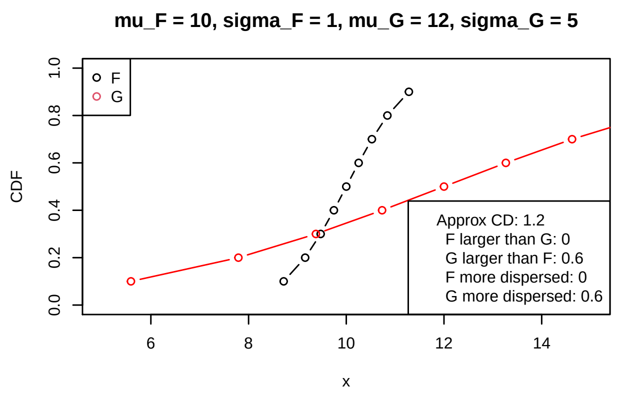
```r
# adapted function:
approx_cd1_decomposed <- function(q_F, q_G){
  # compute quantile levels from length of provided quantile vectors:
  K <- length(q_F)
  if(length(q_G) != K) stop("q_F and q_G need to be of the same length")
  p <- (1:K)/(K + 1) # function assumes that the quantile levels are equally spaced

  # matrices to store differences and booleans indicating whether there is a discrepancy and
  # in which component it should be counted
  matrix_differences <- matrix_disagreement <-
    matrix_F_larger <- matrix_G_larger <-
    matrix_F_dispersed <- matrix_G_dispersed <-
    matrix_unordered <- matrix(NA, ncol = K, nrow = K,
                               dimnames = list(paste("F", round(p, 2)), paste("G", round(p, 2))))

  # fill these matrices:
  for(i in 1:K){
    for(j in 1:K){
      # store absolute differences:
      matrix_differences[i, j] <- abs(q_F[i] - q_G[j])

      # detect disagreements, i.e. cases where q^F_i > q^G_j despite i < j
      matrix_disagreement[i, j] <- ((q_F[i] >= q_G[j]) & (i <= j)) | ((q_G[j] >= q_F[i]) & (j <= i))

      # component "F larger than G"
      matrix_F_larger[i, j] <-  q_F[min(i, K + 1 - i)] > q_G[min(j, K + 1 - j)] &
        q_F[max(i, K + 1 - i)] >= q_G[max(j, K + 1 - j)] &
        q_F[i] > q_G[j] & i <= j
      # component "G larger than F"
      matrix_G_larger[i, j] <-  q_F[min(i, K + 1 - i)] < q_G[min(j, K + 1 - j)] &
        q_F[max(i, K + 1 - i)] <= q_G[max(j, K + 1 - j)] &
        q_F[i] < q_G[j] & i >= j

      # component "F more dispersed than G"
      matrix_F_dispersed[i, j] <- q_F[max(i, K + 1 - i)] > q_G[max(j, K + 1 - j)] &
        q_F[min(i, K + 1 - i)] < q_G[min(j, K + 1- j)] &
        ((i <= j & i >= (K + 1)/2) | (i >= j & i <= (K + 1)/2))
      # component "G more dispersed than F"
      matrix_G_dispersed[i, j] <- q_F[min(i, K + 1 - i)] > q_G[min(j, K + 1 - j)] &
        q_F[max(i, K + 1 - i)] < q_G[max(j, K + 1- j)] &
        ((i >= j & j >= (K + 1)/2) | (i <= j & j <= (K + 1)/2))
    }
  }

  # compute averages:
  cramer_distance <- 2*sum(matrix_differences*matrix_disagreement)/K/(K + 1)
  F_larger <- 2*sum(matrix_differences*matrix_F_larger)/K/(K + 1)
  G_larger <- 2*sum(matrix_differences*matrix_G_larger)/K/(K + 1)
  F_dispersed <- 2*sum(matrix_differences*matrix_F_dispersed)/K/(K + 1)
  G_dispersed <- 2*sum(matrix_differences*matrix_G_dispersed)/K/(K + 1)

  # check whether components add up to CD:
  if(abs(F_larger + G_larger + F_dispersed + G_dispersed - cramer_distance) >= 0.001){
    warning("Decomposition does not seem to work")
  }

  # structure return object:
  return(list(
    cramer_distance = cramer_distance,
    F_larger = F_larger,
    G_larger = G_larger,
    F_dispersed = F_dispersed,
    G_dispersed = G_dispersed
  ))
}

# Examples: Define six pairs of normal distributions
# define distributions:
K <- 9
p <- (1:K)/(K + 1) # quantile levels

vector_mu_F <- c(10, 10, 10, 10, 10, 10)
vector_sigma_F <- c(1, 1, 1, 1, 1, 1)

vector_mu_G <- c(10, 11, 11, 12, 15, 5)
vector_sigma_G <- c(2, 1, 2, 5, 2, 0.5)

par(mfrow = c(3, 2))

for(i in 1:6){
  # compute quantiles
  q_F <- qnorm(p, vector_mu_F[i], vector_sigma_F[i]) # quantiles of F
  q_G <- qnorm(p, vector_mu_G[i], vector_sigma_G[i]) # quantiles of G
  # evaluate CD incl decomposition:
  cd_decomp <- approx_cd1_decomposed(q_F, q_G)

  # plot CCDFs/step functions:
  plot(q_F, p, type = "b", xlim = c(5, 15), ylim = 0:1,
       xlab = "x", ylab = "CDF",
       main = paste0("mu_F = ", vector_mu_F[i],
                     ", sigma_F = ", vector_sigma_F[i],
                     ", mu_G = ", vector_mu_G[i],
                     ", sigma_G = ", vector_sigma_G[i]))
  lines(q_G, p, col = "red", type = "b")
  # print CD and decomposition:
```

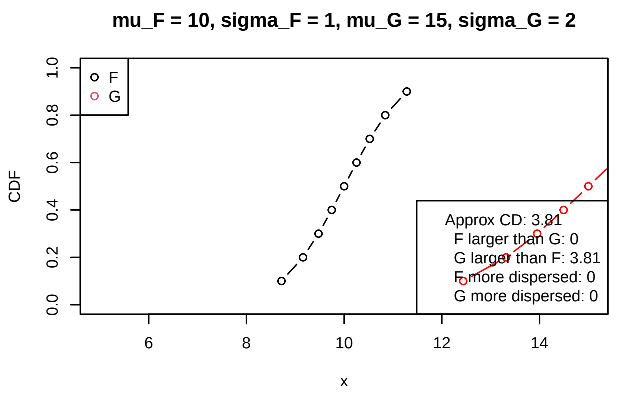
```
  legend("bottomright",
         legend = c(paste0("Approx CD: ", round(cd_decomp$cramer_distance, 2)),
                    paste0("  F larger than G: ", round(cd_decomp$F_larger, 2)),
                    paste0("  G larger than F: ", round(cd_decomp$G_larger, 2)),
                    paste0("  F more dispersed: ", round(cd_decomp$F_dispersed, 2)),
                    paste0("  G more dispersed: ", round(cd_decomp$G_dispersed, 2))))
  legend("topleft", legend = c("F", "G"), col = 1:2, pch = 1)
}
```

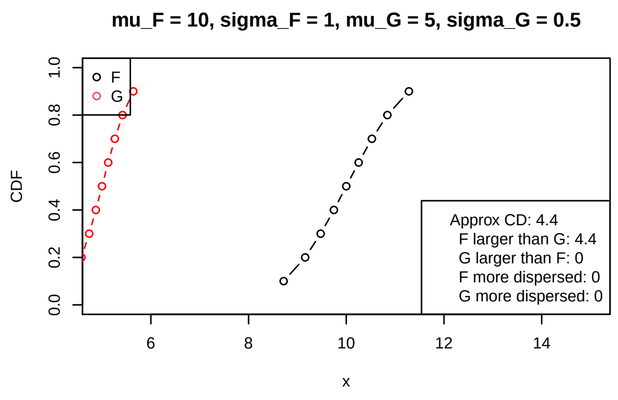












Background thoughts (to be tidied up)

Relationship between Cramer von Mises distance, CRPS and WIS

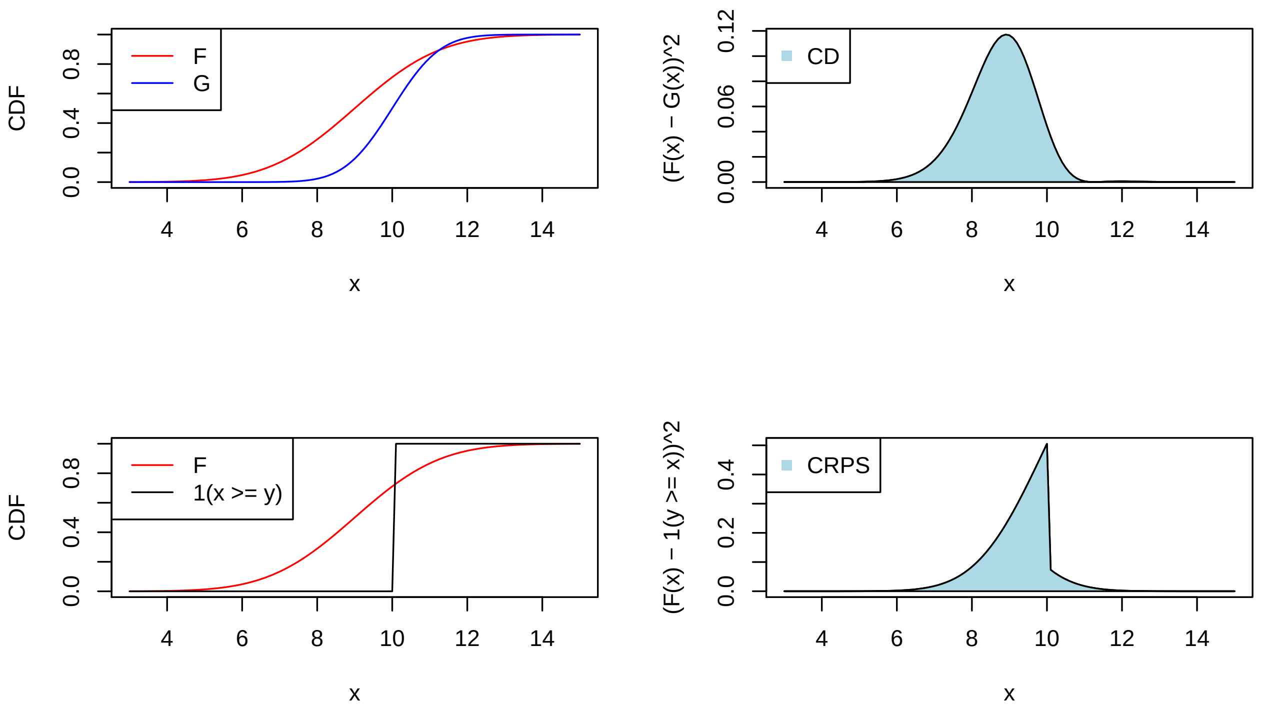
Consider two predictive distributions F and G and an observed value y. In the following we use F and G also to denote the respective cumulative distribution functions (CDFs). The similarity of the definitions of the Cramer von Mises distance (CD)

$$\mathrm{CD}(F, G) = \int_{-\infty}^{\infty} (F(x) - G(x))^2 dx$$

and the CRPS

$$\mathrm{CRPS}(F, y) = \int_{-\infty}^{\infty} (F(x) - \mathbf{1}(x \geq y))^2 dx$$

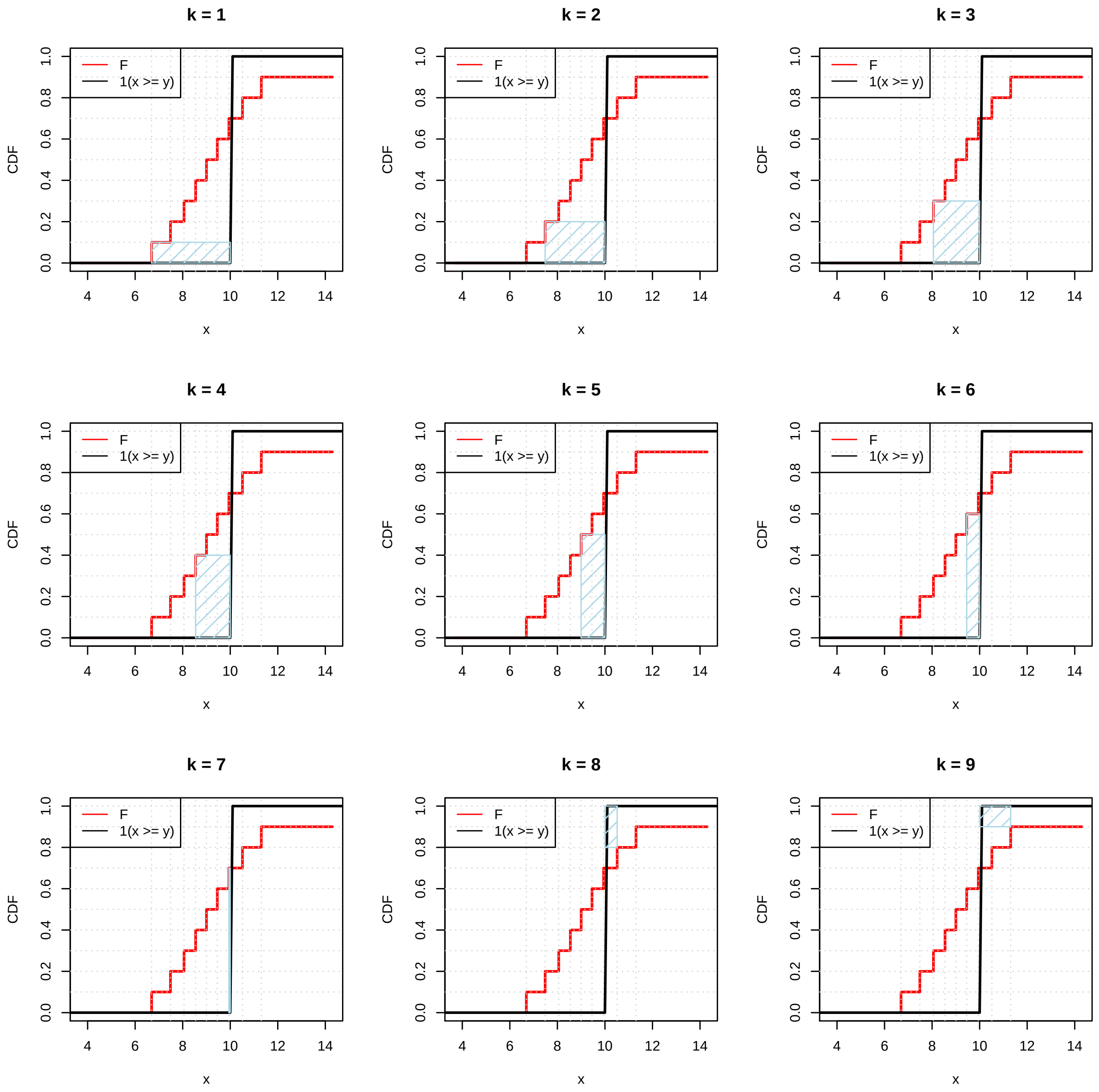
is obvious. Indeed, we can interpret $\mathbf{1}(x \geq y)$ as the CDF of a random variable which always takes the value y. In a sense the CRPS is thus a special case of the CD. The two integrals are illustrated in the figure below.



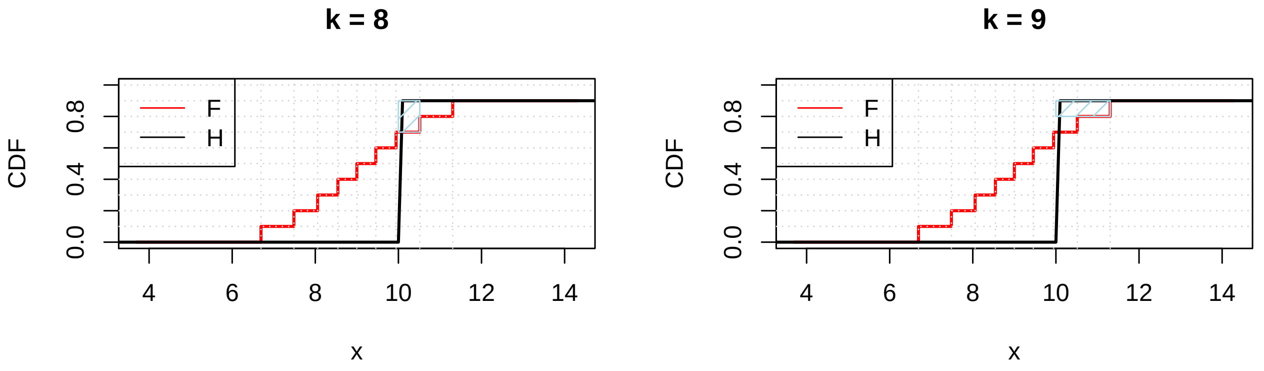
Now assume F is provided in a quantile format, with K quantiles at equally spaced levels $1/(K+1), 2/(K+1), \dots, K/(K+1)$. Denote these quantiles by $q_k^F, k = 1, ..., K$. The weighted interval score, or mean linear quantile score

$$\mathrm{WIS}(F, y) = \frac{2}{K} \times \sum_{k=1}^{K} \{\mathbf{1}(y \leq q_k^F)\} \times (q_k^F - y) \approx \mathrm{CRPS}(F, y)$$

is a commonly used quantile-based approximation of the CRPS. We start by getting an intuition of what this approximation does and why it works. For the following we use $K = 9$ for illustration.



The above figure shows rectangles representing the $K = 9$ terms which are averaged to obtain the WIS. Note that in the above figure the black line for $\mathbf{1}(x \geq y)$ is a "full" CDF in the sense that it reaches the value one. This is a problem if we want to translate things to a more general setting with a step function with jumps at K quantiles of a distribution H. In this case, the right horizontal part of the black line would need to be at $K/(K+1) = 0.9$ instead of 1, as is the case for the red line. Fortunately we can just shift down everything that happens right of y by $1/(K+1) = 0.1$ and maintain a nice geometrical intuition (maybe even a nicer one):



The WIS is given by the average of the size of the K light blue boxes. To get a better understanding of what these represent we slice up the boxes vertically at the K quantiles $q_1^F, \dots, q_K^F$ of F and horizontally at the quantile levels $1/(K+1), \dots K/(K+1)$ (dotted lines in plots). Then obviously all of the resulting small boxes are the same height $1/(K+1)$ and we have the following (specific to the example shown in the plot):

- One box of width $(q_2^F - q_1^F)$ (the one from $k = 1$)
- Three boxes of width $(q_3^F - q_2^F)$ (one from $k = 1$ and two from $k = 2$)
- Six boxes of width $(q_4^F - q_3^F)$ (one from $k = 1$, two from $k = 2$ and three from $k = 3$)

More generally, for segments where $|F(x) - H(x)| = k/(K+1)$ we have $\sum_{i=1}^k i = k(k+1)/2$ boxes. This means that we can re-write the WIS as

$$\text{WIS}(F, y) = \frac{2}{K} \times \sum_{k=1}^K \underbrace{\mu(\{x : |F(X) - H(x)| = k/(K+1)\})}_{\text{width of boxes}} \times \underbrace{\frac{1}{K+1}}_{\text{height of boxes}} \times \underbrace{\frac{k(k+1)}{2}}_{\text{number of boxes with respective width}}, \tag{1}$$

$$= \sum_{k=1}^K \mu(\{x : |F(x) - H(x)| = k/(K+1)\}) \times \underbrace{\frac{k}{K} \times \frac{k+1}{K+1}}_{\approx (F(x)-H(x))^2}$$

where $\mu(\{x : |F(x) - H(x)| = 1/k\})$ is the total length of the segments where $|F(x) - H(x)| = 1/k$. This makes it clear that for large K the approximation

$$\text{WIS}(F, y) \approx \int_{-\infty}^{\infty} (F(x) - H(x))^2 dx$$

indeed holds with

$$H(X) = \begin{cases} 0 \text{ if } x < y \\ K/(K+1) \text{ if } x \geq y. \end{cases}$$

Extension to comparison of two predictive distributions

Formulation (1) and its motivation are sufficiently general that they can also apply it to two "actual" CDFs which both have more than just one jump. Indeed, we can (1) it to approximate the Cramer von Mises distance of any pair of distributions F and G:

$$\text{CD}(F, G) \approx \sum_{k=1}^K \mu(\{x : |F(x) - H(x)| = k/(K+1)\}) \times \frac{k}{K} \times \frac{k+1}{K+1}$$

This expression could be evaluated directly (or at least approximated very closely) using a grid for the x values. However, a simpler way to compute this exists (writing down why it works is tedious and I need to clarify some details, but I've checked in numerous examples and it does work). Denote by $\mathbf{q}$ a vector of length $2K$ containing all quantiles $q_1^F, \dots, q_F^K$ of F and $q_1^G, \dots, q_G^K$ in increasing order; and denote by $\mathbf{a}$ a vector of length $2K$ with 1 wherever the corresponding entry of $\mathbf{q}$ comes from the quantiles of F and -1 if it comes from the quantiles of G. Then the above approximation can also be written as:

$$\mathrm{CD}(F,G) \approx \frac{1}{K(K+1)} \times \sum_{k=1}^{2K-1} b_k(b_k+1)(q_{i+1}-q_i),$$

where

$$b_k = \left| \sum_{i=1}^{k} a_k \right|.$$

Some advantages of this measure:

- Can be computed quickly for a large number of pairs of forecasts.
- Follows the same philosophy as the WIS.
- Similarly to the WIS it can be decomposed into various components (e.g. according to whether $F(X)$ or $G(x)$ predicts larger values or whether one of them is more dispersed).
- If G is just an additively shifted version of F the CD corresponds to the (absolute value of the) shift. This means that the measure can again be interpreted on the natural scale of the data as a sort of generalized absolute difference.

–>