CvM Calculation for COVID-19 Forecasting Models - Demo

Estee Cramer, Nick Reich, Nutcha Wattanachit

02/16/2021

Cramer von-Mises Criterion

Definition

The Cramer von-Mises criterion is defined as

$$\omega^2 = \int_{-\infty}^{\infty} [F_n(x) - F^*(x)]^2 dF^*(x),$$

where $F_n(x)$ and $F^*(x)$ are empirical cumulative distribution functions. The CvM is symmetric.

CvM calculation

The two-sample formulation of CvM criterion can be written in many forms. The equation below is currently used in the first part of this demo (from `cramer::cramer.test()`).

$$T = \frac{mn}{(m+n)}\Big(\frac{2}{mn}\sum_{i=1..m,j=1..n}^{m,n}\phi(||X_i - Y_j||^2) - \frac{1}{m^2}\sum_{i=1..m,j=1..m}^{m,n}\phi(||X_i - X_j||^2) - \frac{1}{n^2}\sum_{i=1..n,j=1..n}^{m,n}\phi(||Y_i - Y_j||^2)\Big)$$

with $\phi_{\text{Cramer}}(z) = \sqrt{z}/2$. Since we have sample quantiles from COVID-19 forecasting models, we do not have the whole empirical distributions for the CvM calculation. In the current implementation, we do the following:

- The monotonic spline function to interpolate is used to interpolate points between available sample quantiles from the forecasting models. Now we have samples (with points interpolated between sample quantiles and extrapolated at the tails).
- We can apply the `ecdf()` function to create and plot the ecdfs from these samples. For the calculation of CvM, samples created in the first step are used.

In this demo, the models are from the week of 02/08/2021.

```r
source("./cvm_script.R")
# create sample data for this demo
sample_frame <- load_latest_forecasts(models = c("CU-select", "UMass-MechBayes", "Covid19Sim-Simulator"
                                                 "COVIDhub-ensemble", "COVIDhub-baseline"),
                              last_forecast_date = "2021-02-08",
                              forecast_date_window_size = 6,
                              locations = "US",
                              types = "quantile",
                              targets = "1 wk ahead inc death",
                              source = "zoltar")
```

```
FALSE polling for status change. job_url=https://zoltardata.com/api/job/43401/
FALSE QUEUED
FALSE QUEUED
FALSE QUEUED
FALSE QUEUED
FALSE QUEUED
FALSE QUEUED
FALSE QUEUED
FALSE SUCCESS
```

```
## make a single target data for demo run
small <- frame_format(sample_frame) %>%
   dplyr::filter(type=="quantile") %>%
   data.frame(.)
names <- colnames(small)[6:ncol(small)]
# interpolate points for these quantiles
point_to_interpolate <- seq(0, 1, by=0.001)
for(i in 1:5){
   assign(paste0("emp",i),
          ecdf(spline(x=small$quantile,
                      y=small[,names[i]],
                      method = "hyman",
                      xout=point_to_interpolate)$y))
}
```

Plot from the ecdf created from the samples:

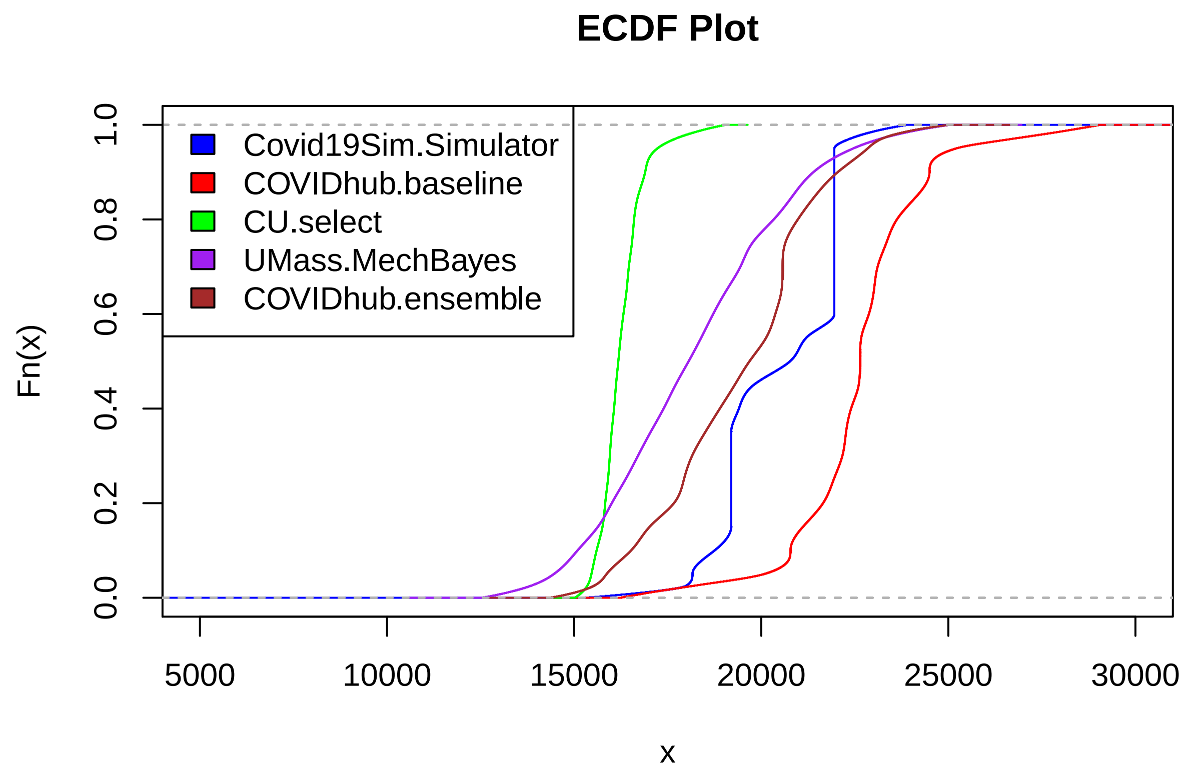
```
plot(emp1,verticals=TRUE, do.points=FALSE, col='blue',xlim=c(5000,30000), main="ECDF Plot")
plot(emp2,verticals=TRUE, do.points=FALSE, add=TRUE, col='red')
plot(emp3,verticals=TRUE, do.points=FALSE, add=TRUE, col='green')
plot(emp4,verticals=TRUE, do.points=FALSE, add=TRUE, col='purple')
plot(emp5,verticals=TRUE, do.points=FALSE, add=TRUE, col='brown')
legend("topleft",names,fill=c("blue","red","green","purple","brown"))
```

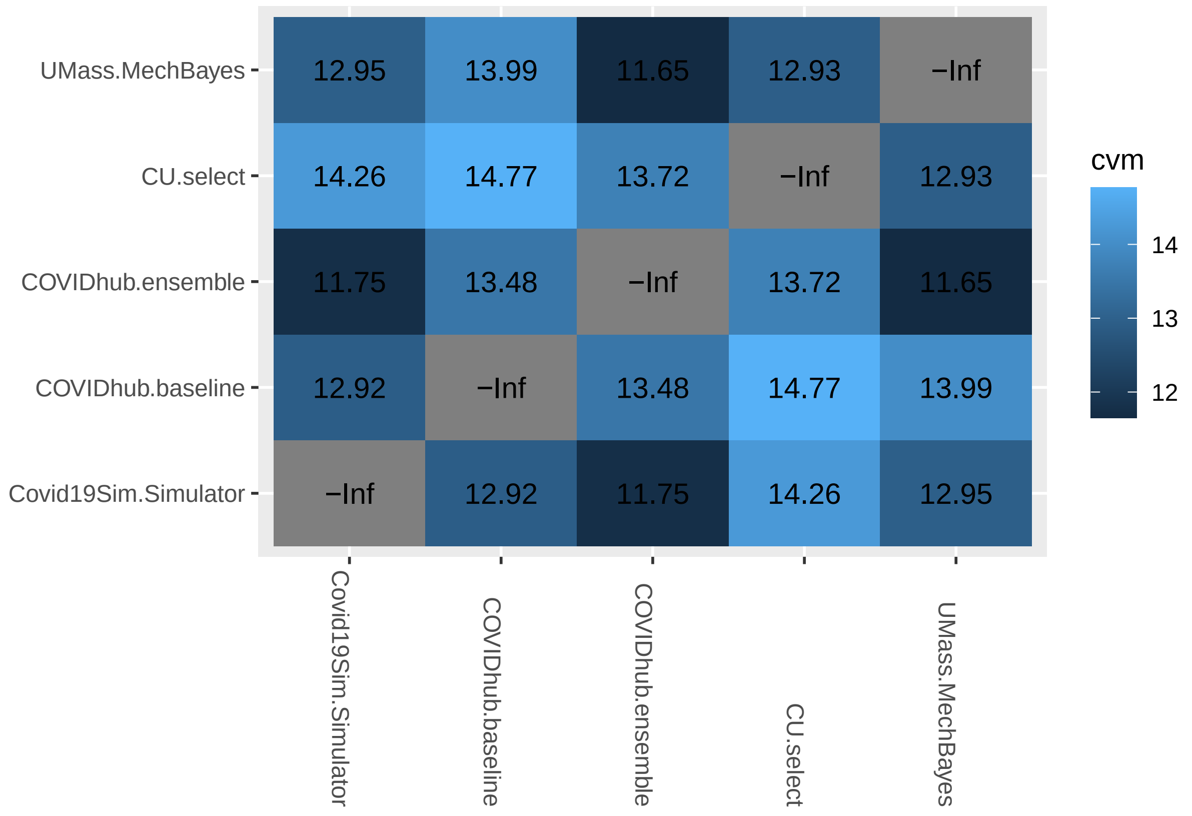
Now we can create a matrix of CvM for the models as shown in the table. We do a log transformation on these values for the heatmap because they are huge:

```
cvm_mat <- build_CvM_frame(small,
                           spline_method="hyman",
                           target_list=unique(small$target_variable),
                           point_to_interpolate)
kable(cvm_mat[[1]])
```

	Covid19Sim.Simulator	COVIDhub.baseline	CU.select	UMass.MechBayes	COVIDhub.ensemble
Covid19Sim.Simulator	0.0	406507.1	1553381.9	419701.4	127087.3
COVIDhub.baseline	406507.1	0.0	2588370.7	1195018.8	717939.8
CU.select	1553381.9	2588370.7	0.0	410456.0	911061.4
UMass.MechBayes	419701.4	1195018.8	410456.0	0.0	115126.9
COVIDhub.ensemble	127087.3	717939.8	911061.4	115126.9	0.0

```
transf_mat <-log(cvm_mat[[1]])
CvM_heatmap(transf_mat)
```





	Covid19Sim.Simulator	COVIDhub.baseline	COVIDhub.ensemble	CU.select	UMass.MechBayes
UMass.MechBayes	12.95	13.99	11.65	12.93	−Inf
CU.select	14.26	14.77	13.72	−Inf	12.93
COVIDhub.ensemble	11.75	13.48	−Inf	13.72	11.65
COVIDhub.baseline	12.92	−Inf	13.48	14.77	13.99
Covid19Sim.Simulator	−Inf	12.92	11.75	14.26	12.95

Comparison of CvM values from various sources

```r
for(i in 1:4){
   assign(paste0("cvm_vals",i),build_CvM_frame_compare(small,
                                   spline_method="hyman",
                                   target_list=unique(small$target_variable),
                                   point_to_interpolate,
                                   est_method=i))
}
kable(cvm_vals1[[1]],caption="From cramer package")
```

Table 2: From cramer package

	Covid19Sim.Simulator	COVIDhub.baseline	CU.select	UMass.MechBayes	COVIDhub.ensemble
Covid19Sim.Simulator	0.0	406507.1	1553381.9	419701.4	127087.3
COVIDhub.baseline	406507.1	0.0	2588370.7	1195018.8	717939.8
CU.select	1553381.9	2588370.7	0.0	410456.0	911061.4
UMass.MechBayes	419701.4	1195018.8	410456.0	0.0	115126.9
COVIDhub.ensemble	127087.3	717939.8	911061.4	115126.9	0.0

```r
kable(cvm_vals2[[1]],caption="From twosamples package")
```

Table 3: From twosamples package

	Covid19Sim.Simulator	COVIDhub.baseline	CU.select	UMass.MechBayes	COVIDhub.ensemble
Covid19Sim.Simulator	0.00000	209.0899	510.9355	165.28760	55.61059
COVIDhub.baseline	209.08991	0.0000	656.6394	471.68663	397.67584
CU.select	510.93550	656.6394	0.0000	233.43204	467.07893
UMass.MechBayes	165.28760	471.6866	233.4320	0.00000	60.88175
COVIDhub.ensemble	55.61059	397.6758	467.0789	60.88175	0.00000

```r
kable(round(cvm_vals3[[1]],2),caption="From  package")
```

Table 4: From package

	Covid19Sim.Simulator	COVIDhub.baseline	CU.select	UMass.MechBayes	COVIDhub.ensemble
Covid19Sim.Simulator	8.54	79.35	161.34	53.15	17.96
COVIDhub.baseline	79.35	0.00	164.16	117.92	99.42
CU.select	161.34	164.16	0.00	58.36	116.77
UMass.MechBayes	53.15	117.92	58.36	0.00	15.22
COVIDhub.ensemble	17.96	99.42	116.77	15.22	0.00

```r
kable(cvm_vals4[[1]],caption="From http://estatcomp.github.io/henrique/exer_chap8.html")
```

Table 5: From http://estatcomp.github.io/henrique/exer_chap8.html

	Covid19Sim.Simulator	COVIDhub.baseline	CU.select	UMass.MechBayes	COVIDhub.ensemble
Covid19Sim.Simulator	400440.4	446005.2	324111.5	358494.4	380704.7

	Covid19Sim.Simulator	COVIDhub.baseline	CU.select	UMass.MechBayes	COVIDhub.ensemble
COVIDhub.baseline	446005.2	491569.9	369676.3	404059.2	426269.5
CU.select	324111.5	369676.3	247782.7	282165.5	304375.8
UMass.MechBayes	358494.4	404059.2	282165.5	316548.4	338758.7
COVIDhub.ensemble	380704.7	426269.5	304375.8	338758.7	360969.0

Simpler toy example of known distributions

Due to large discrepancies in what we see above. We want to see how those functions compare in this toy example. We simulated 3 discrete uniform distributions:

$$f_1 \sim \text{U}(1,2) f_2 \sim \text{U}(1,3) f_3 \sim \text{U}(1,10)$$

```r
# create toy data
n <- 1000
b <- c(2,3,10)
a <- c(1,1,1)
q <- seq(0, 1, by=0.01)
#  fill in bin probs
toy_forecasts <- data.frame(q=q) %>%
   dplyr::mutate(forecast1=sample_quantile(rdunif(n, b[1] , a[1]),q),
                 forecast2=sample_quantile(rdunif(n, b[2] , a[2]),q),
                 forecast3=sample_quantile(rdunif(n, b[3] , a[3]),q))

tnames <- c("forecast1","forecast2","forecast3")
```

We can see the clear step functions here since the distributions are discrete and and the ranges of values are relatively narrow :

```r
# cut the tails (-Inf, Inf) out
plot(ecdf(toy_forecasts$forecast1),verticals=TRUE, do.points=FALSE,xlim=c(0,12),
     col='blue', main="ECDF Plot")
plot(ecdf(toy_forecasts$forecast2),verticals=TRUE, do.points=FALSE, add=TRUE, col='red')
plot(ecdf(toy_forecasts$forecast3),verticals=TRUE, do.points=FALSE, add=TRUE, col='green')
legend("topleft",tnames,fill=c("blue","red","green"))
```

The last two functions still seem off. At this point I'm not sure how correct the first two are since I have not calculated the actual integral. Since the squared difference of the ecdfs is integrated with respect to a function (one of the two distributions), it is not equal to the area between the two curves. The paper "A Geometric Interpretation of the Riemann-Stieltjes Integral" by Gregory L. Bullock is a good read for trying to visualize this.

The formula that `twosample` use is a simplified calculation of the actual CvM, $\sum |F_1(x) - F_2(x)|^p$ with $p = 2$. The CvM values differ greatly in scale. Depending on the distributions of CvM statistic from `cramer` and `twosample`, they might give the same two-test results. However, the two-sample test is not our objective and scale might matter.

```r
p <- seq(0, 1, by=0.001)
for(i in 1:4){
   assign(paste0("tcvm_vals",i),toy_compare(toy_forecasts,
                                            spline_method="hyman",
                                            p,
                                            est_method=i))}
kable(tcvm_vals1,caption="From cramer package")
```

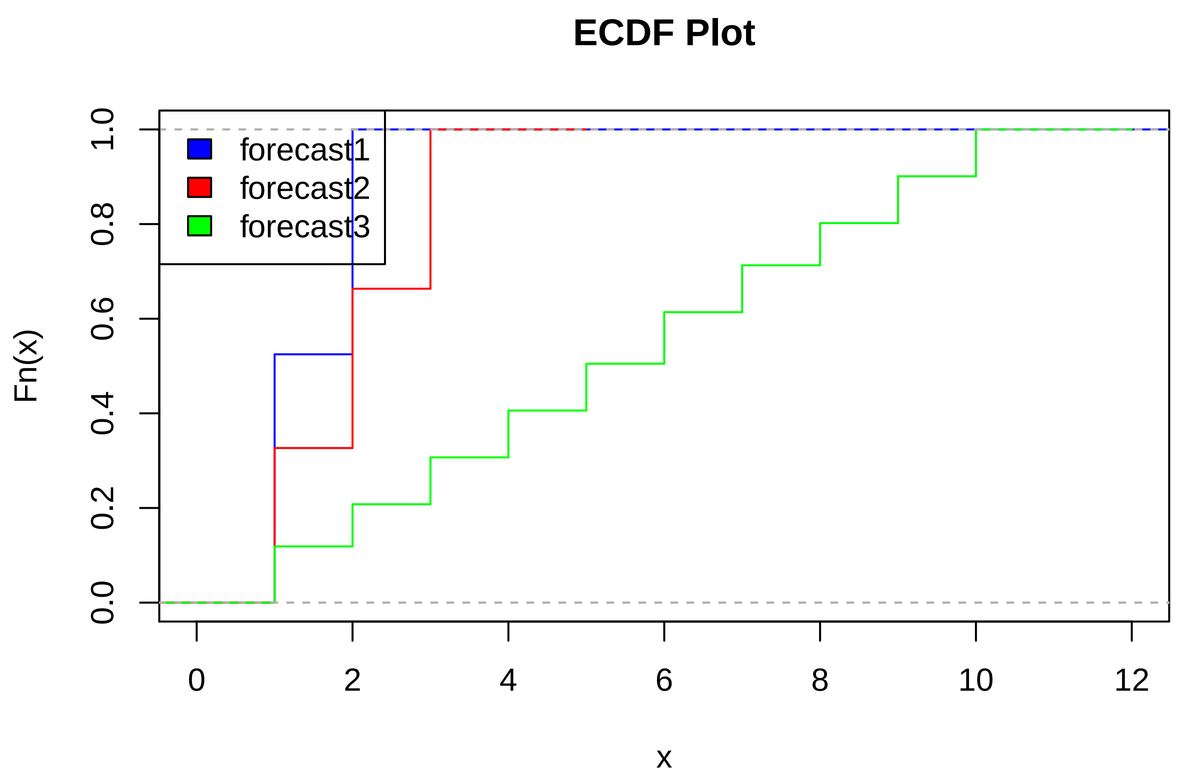


Table 6: From cramer package

	forecast1	forecast2	forecast3
forecast1	0.00000	76.20646	1080.0613
forecast2	76.20646	0.00000	807.7209
forecast3	1080.06127	807.72094	0.0000

```r
kable(tcvm_vals2,caption="From twosamples package")
```

Table 7: From twosamples package

	forecast1	forecast2	forecast3
forecast1	0.000000	1.522638	22.92857
forecast2	1.522638	0.000000	17.97863
forecast3	22.928568	17.978635	0.00000

```r
kable(round(tcvm_vals3,2),caption="From  package")
```

Table 8: From package

	forecast1	forecast2	forecast3
forecast1	40.90	13.92	91.69
forecast2	72.62	17.57	67.45
forecast3	126.55	97.58	1.33

```r
kable(tcvm_vals4,caption="From http://estatcomp.github.io/henrique/exer_chap8.html")
```

Table 9: From http://estatcomp.github.io/henrique/exer_chap8.h
tml