CvM Calculation for COVID-19 Forecasting Models - Demo

Estee Cramer, Nick Reich, Nutcha Wattanachit

02/23/2021

Model Similarity Demo

Definitions

Cramer von-Mises criterion

The Cramer von-Mises criterion is defined as

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

where $F_n(x)$ is an empirical cumulative distribution function and $F^*(x)$ is a theoretical cumulative distribution function for a one-sample case.

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()`).

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

with $\phi_{\text{Cramer}}(z) = \sqrt{z}/2$. The formula that `twosample` use is $\sum |F_1(x) - F_2(x)|^p$ with $p = 2$. The CvM values differ greatly in scale.

Cramer's Distance

Cramer's Distance is defined as

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

For univariate distributions, the Cramer's distance is exactly twice the energy distance. The current implementatino in this demo uses the function `eqdist.e()` from the `energy` package to calculate the energy distance or statistic and then divide the number by 2 to get Cramer's distance.

Setup Process

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.

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:

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

```
source("./distance_func_script.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")
```

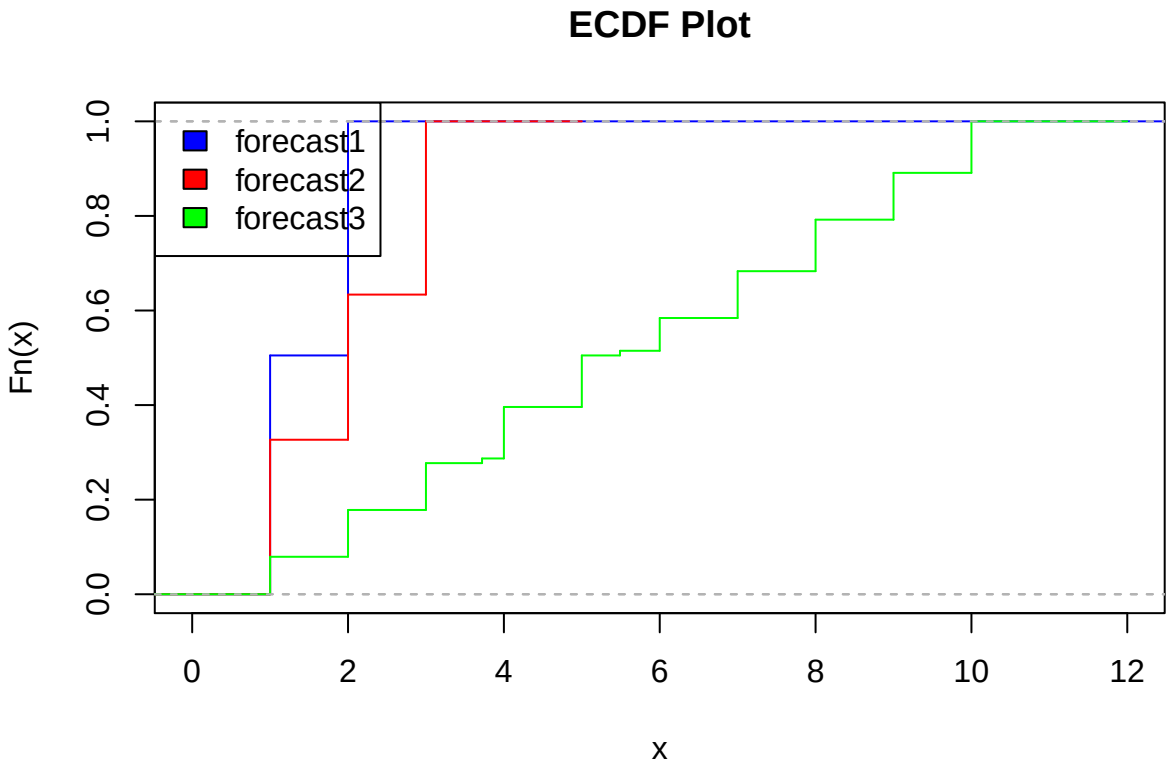
We can see the clear step functions here since the distributions are discrete and and the ranges of values are relatively narrow :

```
# 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"))
```

```
p <- seq(0, 1, by=0.001)
for(i in 1:6){
   assign(paste0("tcvm_vals",i),toy_compare(toy_forecasts,
                                            spline_method="hyman",
                                            p,
                                            est_method=i))}
kable(tcvm_vals1,caption="From cramer package")
```

Table 1: From cramer package

	forecast1	forecast2	forecast3
forecast1	0.00000	82.91475	1159.5294
forecast2	82.91475	0.00000	863.6976
forecast3	1159.52940	863.69756	0.0000




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

Table 2: From twosamples package

	forecast1	forecast2	forecast3
forecast1	0.00000	1.65667	31.82354
forecast2	1.65667	0.00000	26.66045
forecast3	31.82354	26.66045	0.00000

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

Table 3: From package

	forecast1	forecast2	forecast3
forecast1	40.61	15.34	102.74
forecast2	70.34	17.86	75.76
forecast3	138.47	107.82	1.28

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

Table 4: From http://estatcomp.github.io/henrique/exer_chap8.html

	forecast1	forecast2	forecast3
forecast1	-1.243737	-1.615940	-3.968336
forecast2	-1.615940	-1.988143	-4.340539
forecast3	-3.968336	-4.340539	-6.692934


```
kable(round(tcvm_vals5,3),caption="cramer's distance using energy distance from energy package divided b
```

Distance metrics of Some COVID-19 Forecasting Models

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

```
# 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/44225/
FALSE QUEUED
FALSE QUEUED
FALSE SUCCESS
```

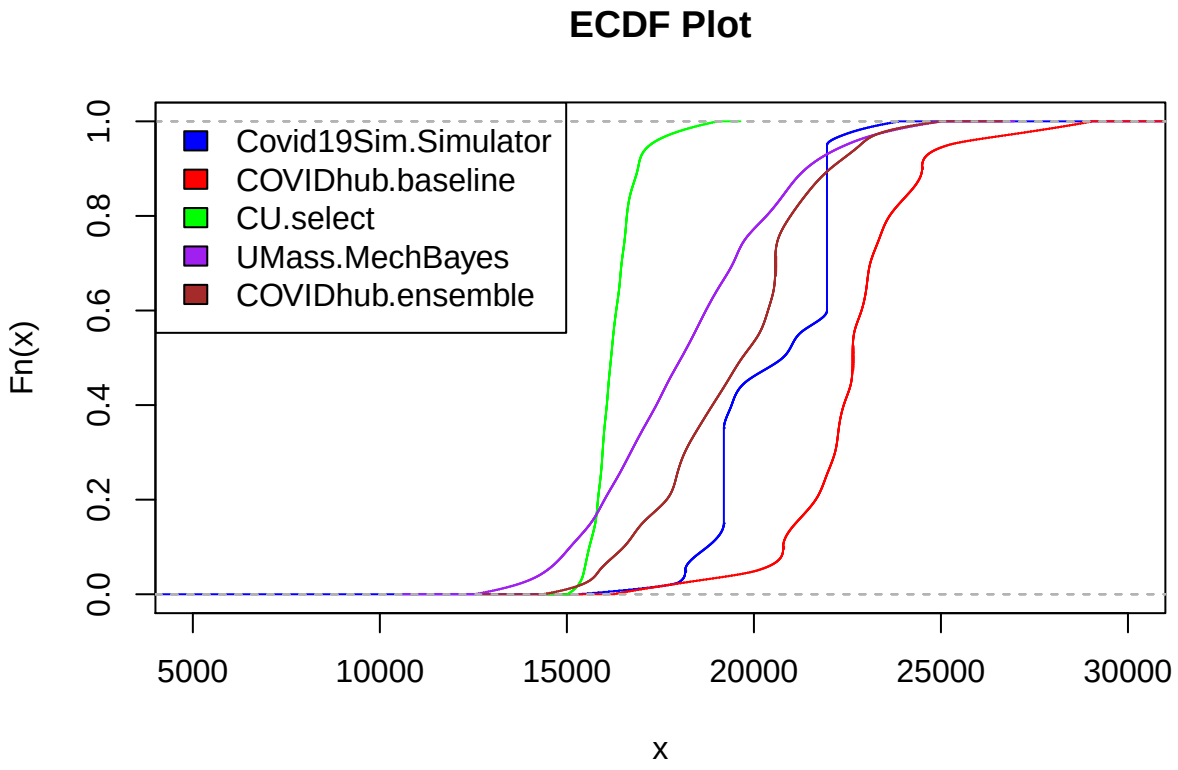
```
## make a single target data for demo run
small <- frame_format(sample_frame) %>%
   dplyr::filter(type=="quantile") %>%
   data.frame(.)
names <- colnames(small)[6:ncol(small)]
# interpolate points for these quantiles
point_to_interpolate <- seq(0, 1, by=0.001)
for(i in 1:5){
   assign(paste0("emp",i),
          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 distance matrices for the models.The CvM criterion and Cramer's distance results are the same. Is it possible the author of the `cramer` package confuse CvM with Cramer's distance just like we did? The authors of the `energy` package made it clear that those are different, so I assume they are aware. On the other hand, the approximated Cramer's distance based on quantile look different from the Cramer's distance we see. It is possible that I interpolated so many points and created a large sample (and sample size matters in the calculation of the the distances based on their formula).

```
cvm_mat <- build_distance_frame(small,
                            spline_method="hyman",
                            target_list=unique(small$target_variable),
                            point_to_interpolate,
                            distance="CvM")
cramer_mat <- build_distance_frame(small,
                            spline_method="hyman",
```




```
                          target_list=unique(small$target_variable),
                          point_to_interpolate,
                          distance="cramer")
approx_cd_mat <- build_distance_frame(small,
                          target_list=unique(small$target_variable),
                          distance="approx_cramer")
kable(cvm_mat[[1]])
```

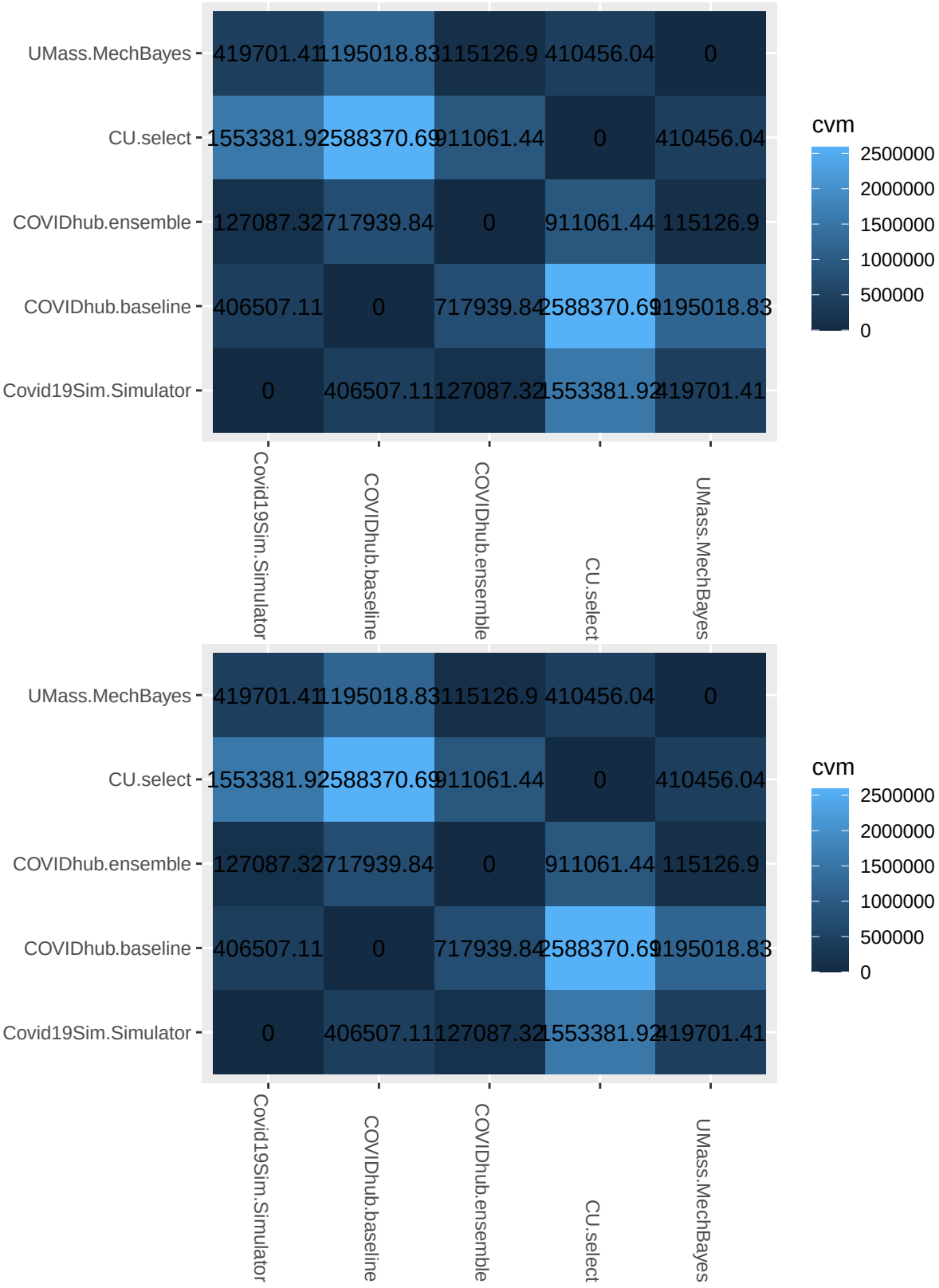
	Covid19Sim.Simulator	COVIDhub.baseline	CU.select	UMass.MechBayes	COVIDhub.ensemble
Covid19Sim.Simulator	0.0	406507.1	1553381.9	419701.4	127087.3
COVIDhub.baseline	406507.1	0.0	2588370.7	1195018.8	717939.8
CU.select	1553381.9	2588370.7	0.0	410456.0	911061.4
UMass.MechBayes	419701.4	1195018.8	410456.0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

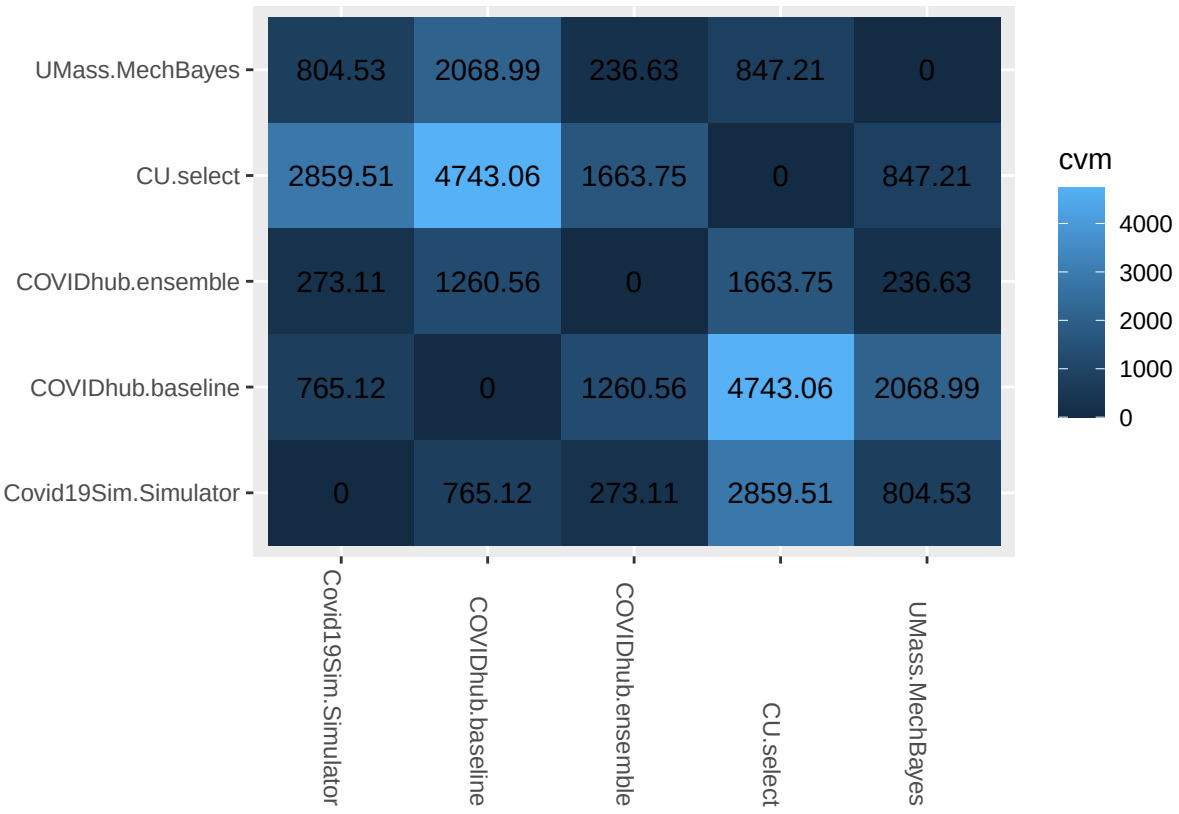
```
kable(round(cramer_mat[[1]],3))
```

	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

```
kable(round(approx_cd_mat[[1]],3))
```

	Covid19Sim.Simulator	COVIDhub.baseline	CU.select	UMass.MechBayes	COVIDhub.ensemble
Covid19Sim.Simulator	0.000	765.125	2859.507	804.531	273.110
COVIDhub.baseline	765.125	0.000	4743.063	2068.992	1260.557
CU.select	2859.507	4743.063	0.000	847.207	1663.754
UMass.MechBayes	804.531	2068.992	847.207	0.000	236.627


UMass.MechBayes
CU.select
COVIDhub.ensemble
COVIDhub.baseline
Covid19Sim.Simulator
419701.41
1195018.83
115126.9
410456.04
0
1553381.92
2588370.69
911061.44
0
410456.04
127087.32
717939.84
0
911061.44
115126.9
406507.11
0
717939.84
2588370.69
1195018.83
0
406507.11
127087.32
1553381.92
419701.41
Covid19Sim.Simulator
COVIDhub.baseline
COVIDhub.ensemble
CU.select
UMass.MechBayes
cvm
2500000
2000000
1500000
1000000
500000
0


UMass.MechBayes
CU.select
COVIDhub.ensemble
COVIDhub.baseline
Covid19Sim.Simulator
804.53
2068.99
236.63
847.21
0
2859.51
4743.06
1663.75
0
847.21
273.11
1260.56
0
1663.75
236.63
765.12
0
1260.56
4743.06
2068.99
0
765.12
273.11
2859.51
804.53
Covid19Sim.Simulator
COVIDhub.baseline
COVIDhub.ensemble
CU.select
UMass.MechBayes
cvm
4000
3000
2000
1000
0