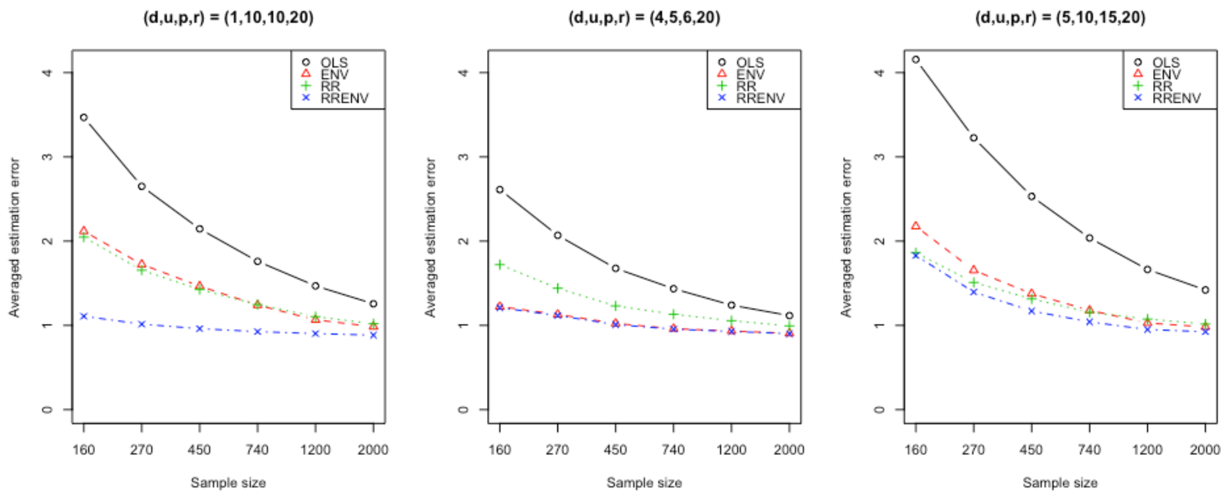
Data Simulations

Now we compare the performance of our Reduced-rank envelope regression model with those of other methods (OLS, Reduced-rank regression and Envelope regression). To compare the model performance, we can consider not only assessing the fit of models, but also calculating the running time of the code. Here we just compare the fit between the models since we are going to examine the running time at the benchmark test part.

First, we generated virtual datasets. Data was generated by three different combinations of parameters: (d, u, p, r) = (1, 10, 10, 20), (4, 5, 6, 20) and (5, 10, 15, 20). In addition, we also generate data by different sample size (N=160, 270, 450, 740, 1200, 2000).

After generating the virtual datasets, we did the modeling with our new Reduced-rank envelope package and existing R packages (for OLS, Reduced-rank regression and Envelope regression). We calculated $\|\hat{\beta} - \beta\|_F$ for assessing the averaged estimation error of models, and Figure X.1 is the result plot.

As we can see from the plot, our Reduced-rank envelope regression model always showed the best performance in all the three cases. In addition, it is also impressive that Reduced-rank envelope was stable between different sample sizes from 160 to 2000. In particular, it is noteworthy that in a small sample size, its performance was superior to other methods.



<Figure X> Estimation error $\|\hat{\beta} - \beta\|_F$ for each of the estimators: OLS (black), Envelope regression (red), Reduced-rank regression (green) and **Reduced-rank envelope regression (blue)**

Selection of d and u

Suppose we want to model a Reduced-rank envelope regression with our new R package with a dataset. In determining the parameters required for modeling, p is automatically determined according to the dimension of the X matrix, and r is determined by the dimension of the Y. matrix. However, d and u should be set by the user. Thus, it would be useful if we could recommend the combination of parameter (d, u) so that the user can use it for an effective modeling. The function *rrenv.choose_du()* in our package provides this recommendation for users.

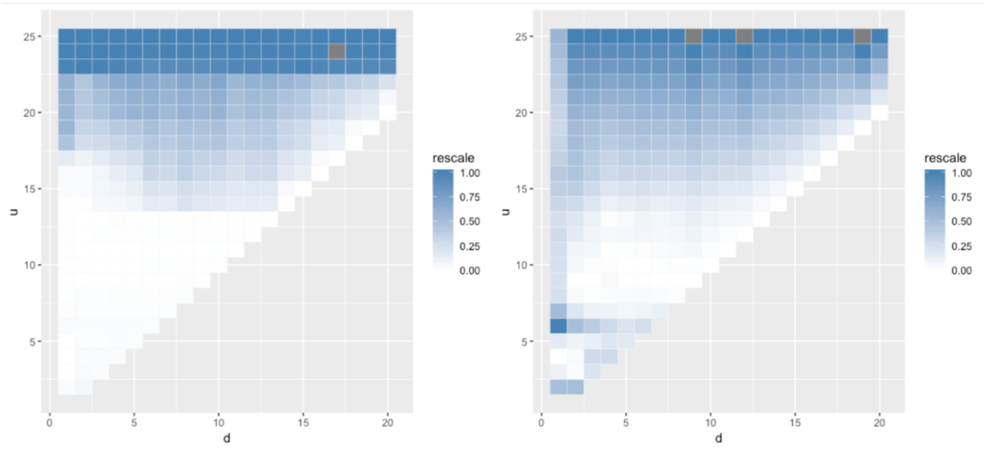
```
Algorithm: rrenv.choose_du(X, Y, Beta=NULL)

p = number of columns of X matrix
r = number of columns of Y matrix
max_u = r
max_d = min(p,r)
Error_table = An empty matrix
if Beta = NULL:
    for u = 2 to max_u:
        d=0
        while d < max_d & d<u:
            d = d+1
            est = R_reduced_rank_envelope_given_u_d(X,Y,u,d)
            NR = (p+u-d)*d + ((r*(r+1))/2)
            Lud = $\hat{L}_{u,d}$ (The conditional loglikelihood of Y)
            Error = 2NR – 2Lud
            Append a vector (Error, d and u) to Error_table

else if dimension of Beta = (r,p):
    for u = 2 to max_u:
        d=0
        while d < max_d & d<u:
            d = d+1
            est = R_reduced_rank_envelope_given_u_d(X,Y,u,d)
            Error = $\|\hat{\beta} - \beta\|_F$
            Append a vector (Error, d and u) to Error_table
else :
    print("Error: Invalid Beta Matrix!")

return (min(Error) with the combination of (d,u) from Error_table)
```

<Figure X.2> Algorithm of the function *rrenv.choose_du(X, Y, Beta)*

Let us look at an example of using *rrenv.choose_du().* The first case is when we know the true beta matrix. When modeling with data, it is highly unlikely that you already have had the true beta matrix. But we check this case to see how efficiently our function recommends the combination of parameters (d, u). We first set (d, u, p, r) = (10, 15, 25, 30) to generate the virtual data (X, Y, Beta matrix) and ran the function *rrenv.choose_du()* with this data as inputs of the function*.* After the implementation, our program calculated $\|\hat{\beta} - \beta\|_F$ and recommended (d, u) = (10, 13), which is very close to the actual d and u (10, 15), as the best combination and provided a heatmap of estimated errors.

The second case is when we do not know the true beta matrix. As mentioned above, in most of cases there would be no prior knowledge of the actual beta matrix. For this case, we input the same X and Y matrix as the first case, with the argument Beta set to NULL in the function *rrenv.choose_du()*.

In this case, the function calculates the AIC to recommend the best combination of (d,u) since there is no information about the true betas. With the heatmap of estimated errors (Figure X.3 – Right), our function recommended (7, 13) as the best combination of (d, u). Now we will be able to do an effective Reduced-rank envelope regression modeling with reference to (d, u) obtained from function *rrenv.choose_du()*.



<Figure X.3> Heatmaps of estimated error by (d,u)
– Left: the case with true beta matrix / Right: the case without true beta matrix