Global Measures of Data Utility

MATH 301 Data Confidentiality

Henrik Olsson

February 18, 2020

After reading the Woo paper on Global Measures of Data Utility and discussing with Kevin Ros, I attempted to implement the measures. The methods include Propensity Score Measure, Cluster Analysis Measure, and Empirical CDF Measures.

Propensity Score Measure

I attempted to generate the code for propensity score measures. First, we merge the original and synthetic datasets from the CEsample in HW 1. We must add variable T equal to one for the synthetic dataset and T equal to zero for the original dataset.

```
## Merge original and masked dataset (Method from Kevin Ros)
## Add variable T equal to 1 for synthetic dataset
one_data = rep(1, nrow(synthetic_one))
syn_data = data.frame(synthetic_one$SynLogIncome)
syn_data$T = one_data
colnames(syn_data)[colnames(syn_data) == "synthetic_one.SynLogIncome"] <- "Income"

## Add variable T equal to 0 for original dataset
zero_data = rep(0, nrow(synthetic_one))
orig_data = data.frame(synthetic_one$OrigLogIncome)
orig_data$T = zero_data
colnames(orig_data)[colnames(orig_data) == "synthetic_one.OrigLogIncome"] <- "Income"

merged_data = rbind(orig_data, syn_data)
head(merged_data,10)
```

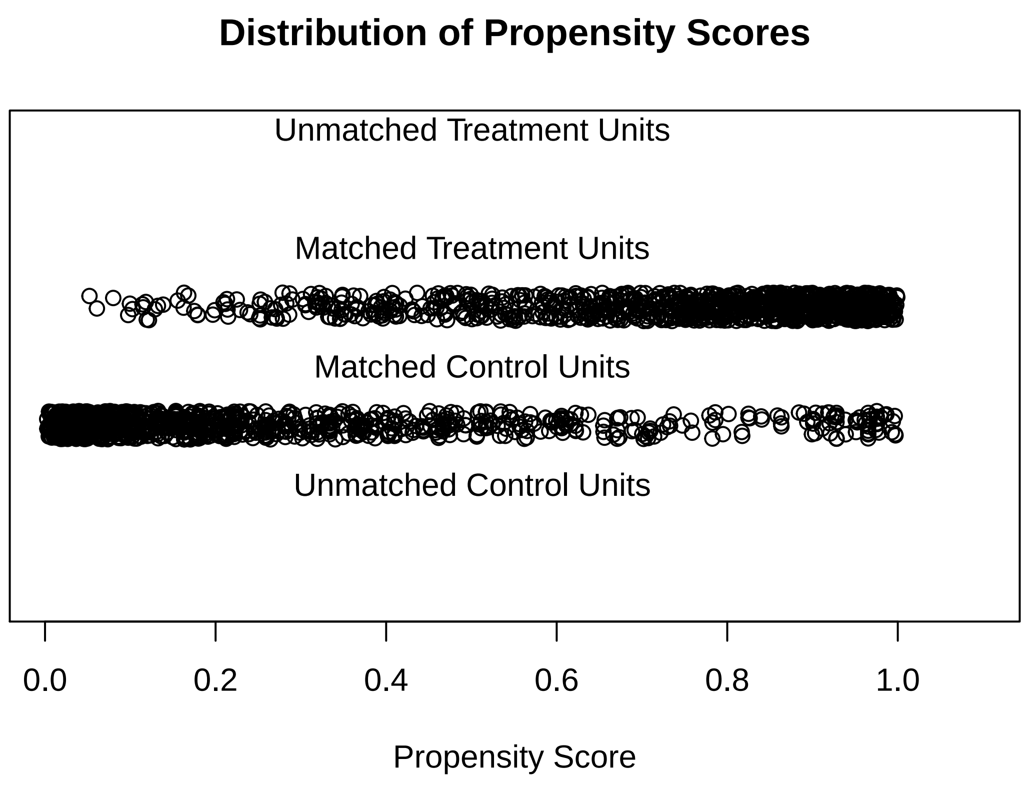
```
##       Income T
## 1  11.498827 0
## 2  10.100698 0
## 3  11.292279 0
## 4  11.921718 0
## 5  11.775290 0
## 6  10.399281 0
## 7   7.414573 0
## 8  11.624538 0
## 9   8.732950 0
## 10 11.571194 0
```

Second, for each record in the original and masked data, we compute the probability of being in the masked data set, or the propensity score. We use the function MatchIt, which I found only to find the propensity score.

```
## Using the matchit function for propensity score, nearest neighbor matching
## Grouping variable is T and the variables being matched in Income (original and synthetic)
m.out = matchit(T ~ Income, data = merged_data, method = "nearest")
## Results of matchit saved in a variable called m.out
summary(m.out)
```

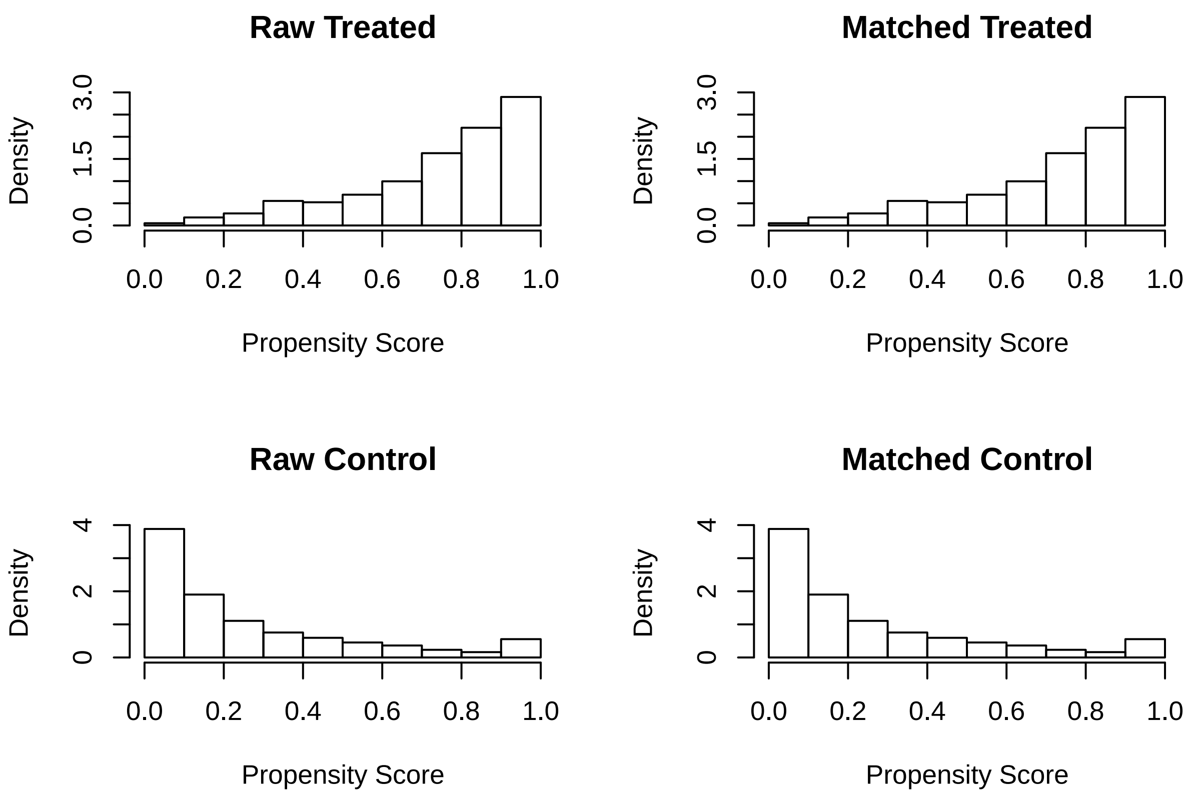
```
##
## Call:
## matchit(formula = T ~ Income, data = merged_data, method = "nearest")
##
## Summary of balance for all data:
##          Means Treated Means Control SD Control Mean Diff eQQ Med eQQ Mean
## distance        0.7429        0.2571     0.2643    0.4859  0.5547   0.4859
## Income          8.7614       10.5951     1.1535   -1.8337  1.9294   1.8337
##          eQQ Max
## distance  0.6552
## Income    2.1163
##
##
## Summary of balance for matched data:
##          Means Treated Means Control SD Control Mean Diff eQQ Med eQQ Mean
## distance        0.7429        0.2571     0.2643    0.4859  0.5547   0.4859
## Income          8.7614       10.5951     1.1535   -1.8337  1.9294   1.8337
##          eQQ Max
## distance  0.6552
## Income    2.1163
##
## Percent Balance Improvement:
##          Mean Diff. eQQ Med eQQ Mean eQQ Max
## distance          0       0        0       0
## Income            0       0        0       0
##
## Sample sizes:
##           Control Treated
## All           994     994
## Matched       994     994
## Unmatched       0       0
## Discarded       0       0
```

```
## propensity score plots
plot(m.out, type ="jitter")
```



```
## [1] "To identify the units, use first mouse button; to stop, use second."
## integer(0)
```

```
plot(m.out, type ="hist")
```



```
## Fit a propensity score model. Logistic regression
psmodel <- glm(T ~ Income, family = "binomial", data = merged_data)

## Calculate the propensity score
c = 1/2
N = nrow(merged_data)
p_i <- psmodel$fitted.values
pscore = (p_i-c)^2
pscore = sum(pscore)/N
pscore
```

```
## [1] 0.1174358
```

Nearest Neighbor method matches a treated unit to a control unit that is closest in terms of a distance measure such as logit. The each dot in the jitterplot represent's a case's propensity score. The absence of cases in the upper stratification indicates that there were no unmatched treatment units. The middle stratifications show the close match between the treatment and matched control units. The histogram are before and after matching. They are very close.

The propensity score is approximately 0.115, which close to 0.25 target to be completely distinguishable, but also close to 0, which completely matches the original data.

Cluster Analysis Measure

Cluster analysis places records into groups whose members have similar values of selected variables. We use O and M to denote the original and masked data, respectively.

We first merge the original and masked datasets. Then we prespecify a value for G to be 20

```
## Merge original and masked datasets (Taken from propensity score measure)

## Perform a cluster analysis on the merged data with a fixed number of groups G
## K-Means Cluster Analysis
fit <- kmeans(merged_data, 20)
## get cluster means
aggregate(merged_data,by=list(fit$cluster),FUN=mean)
```

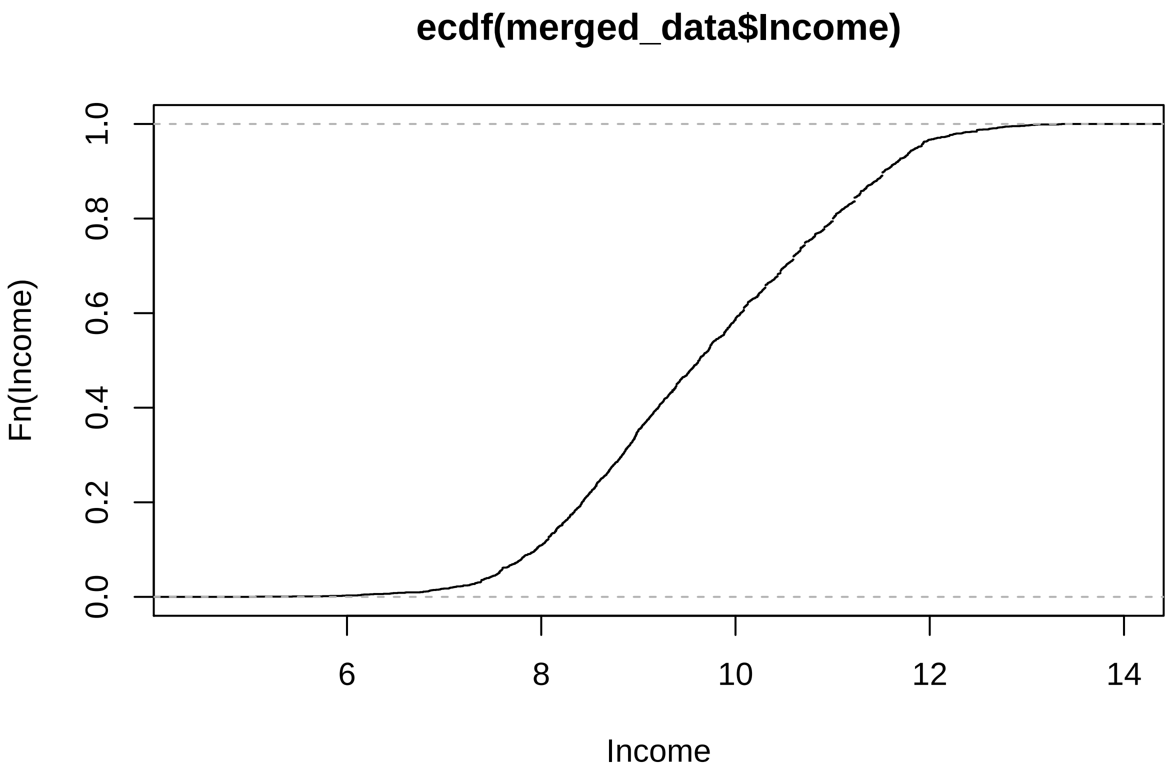
```
##    Group.1    Income         T
## 1        1  8.473198 1.0000000
## 2        2 10.955353 0.0000000
## 3        3 10.485276 0.0000000
## 4        4  7.876420 0.0000000
## 5        5  8.852642 1.0000000
## 6        6 11.361270 0.0000000
## 7        7  9.849473 1.0000000
## 8        8  8.089603 1.0000000
## 9        9  9.889053 0.0000000
## 10      10  9.186907 1.0000000
## 11      11 10.216025 1.0000000
## 12      12 10.881369 1.0000000
## 13      13  7.644910 1.0000000
## 14      14  9.521214 1.0000000
## 15      15  7.062081 1.0000000
## 16      16  6.109069 0.7222222
## 17      17 11.756867 0.0000000
## 18      18 12.754886 0.0000000
## 19      19  9.218900 0.0000000
## 20      20 12.158916 0.0000000
```

```
## append cluster assignment
mydata <- data.frame(merged_data, fit$cluster)
## Then we calculate using the following measure
```

If I could discover the code, I would try several values of G on the original dataset to examine the sensitivity to the choice G.

Empirical CDF Measures

```
plot(ecdf(merged_data$Income), xlab = "Income", ylab = "Fn(Income)")
```



The ecdf function shows the proportion of scores that are less than or equal to each score. The Fun means, in effect, “cumulative function”.

```
S_x = ecdf(synthetic_one$OrigLogIncome)
S_y = ecdf(synthetic_one$SynLogIncome)
diff = c()
for(i in 1:length(synthetic_one$OrigLogIncome)) {
  diff = c(diff, (synthetic_one$OrigLogIncome[i] - synthetic_one$SynLogIncome[i]))
}

## Calculation for U_m maximum absolute difference
max(diff)
```

```
## [1] 4.74341
```

This is definitely incorrect

```r
## Calculation for U_s (average squared differences)
for(i in 1:length(synthetic_one$OrigLogIncome)) {
  diff = c(diff, (synthetic_one$OrigLogIncome[i] - synthetic_one$SynLogIncome[i]))
}
sum(diff)/N
```

```
## [1] 1.833714
```

The drawback to using the CDF Empirical measure is that they can have low power to detect differences in distributions. Unfortunately, I am not sure how to accurately use this method.