The results below are generated from an R script.

```
library(haven)
library(tidyverse)

## - Attaching packages ------------------------- tidyverse 1.3.0 -
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
## - Conflicts -------------------------- tidyverse_conflicts() -
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(randomForest)

## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package:  ’randomForest’
## The following object is masked from ’package:dplyr’:
##
##     combine
## The following object is masked from ’package:ggplot2’:
##
##     margin

library(ggthemes)

# dataset based on:
# Generating Skilled Self-Employment in Developing Countries: Experimental Evidence
# from Uganda (Blattman et al, 2014)
# outcome of interest is profit in last four weeks capped at 99th percentile;
# treatment is randomised assignment to the Program (intent to treat)
# method is based on:
# Generic Machine Learning Inference on Heterogenous Treatment Effects in Randomized
# Experiments (Chernozhukov et al, 2017)
# Note: This is merely preliminary EDA and will be refined once more of the data is
# understood
# Final code will be run with neural nets and other forest based methods to derive
# Best Linear Predictor

#some data cleaning

x<-cbind(read_dta(’https://www.dropbox.com/s/yxgigmtcrut9fii/yop_analysis.dta?dl=1’) %>%
  filter(e1==1) %>%  #filter results only from endline survey1
  select(assigned,loan_100k,loan_1mil,grantsize_pp_US_est3,
         risk_aversion,female,urban,age,live_together,
                       literate,voc_training,inschool,aggression_n,
                       violence_others,education,wealthindex,
                       risk_aversion,profits4w_real_p99_e),
  read_dta(’https://www.dropbox.com/s/fa7lh4qe8nszxdb/yop_fulldata.dta?dl=1’) %>%
  filter(e1==1) %>%
  select(hhsize,acres,credit_access))
```

```
#compute proportion of all NA values
sum(x%>%is.na())/(ncol(x)*nrow(x))
```

```
## [1] 0.0213672
```

```
#compute proportion of missing outcome values
sum(x$profits4w_real_p99_e%>%is.na())/nrow(x)
```

```
## [1] 0.2510273
```

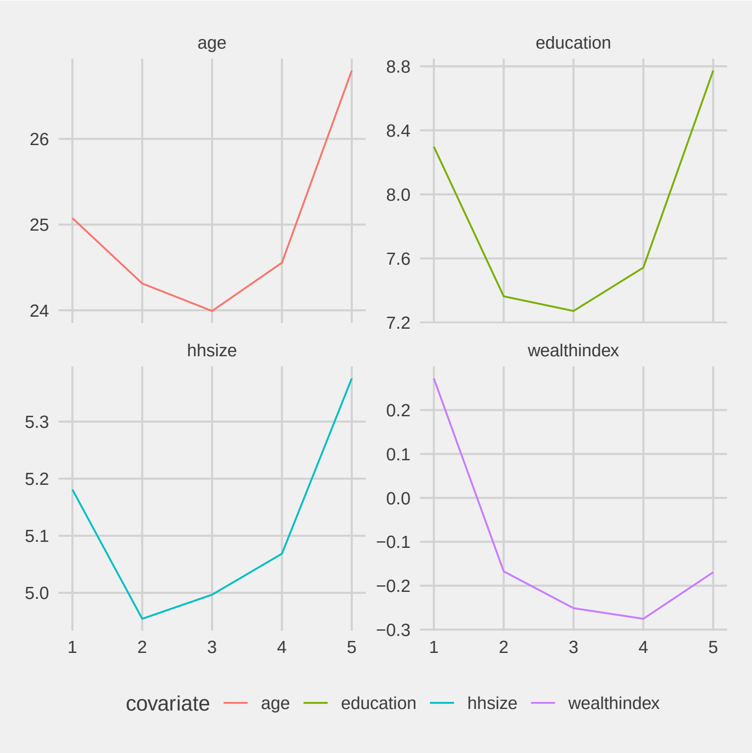
```
#eliminate NA values (assume missingness is random)
#df<-x[which(complete.cases(x)),]

#impute outcome values using median (assume missingness is non random)
df<-x %>% mutate(profits4w_real_p99_e=ifelse(is.na(profits4w_real_p99_e),
                                            median(x$profits4w_real_p99_e,na.rm = T),
                                            profits4w_real_p99_e))

#impute NA values for other vars
df<-rfImpute(profits4w_real_p99_e~.,df,ntree=500,iter=6)
```

```
##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##  500 |     5289    97.51 |
##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##  500 |     5300    97.70 |
##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##  500 |     5268    97.11 |
##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##  500 |     5270    97.15 |
##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##  500 |     5294    97.59 |
##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##  500 |     5283    97.39 |
```

```
# create empty dataframes to store values
n_split<-100
n_group<-5
blp_coef<-data.frame(B0=1:n_split,SE_B0=1:n_split,
                     P_value_B0=1:n_split,
                     B1=1:n_split,SE_B1=1:n_split,
                     P_value_B1=1:n_split)

gate_coef<-matrix(ncol = n_group*2,nrow = n_split) %>% as.data.frame()
colnames(gate_coef)<-paste(c("G",’SE_G’), rep(1:n_group, each=2), sep = "")

#f<-which(sapply(df, class) == "factor")
#mcol<-ncol(df[,-f])
mcol<-ncol(df)
gate_mean<-matrix(ncol = mcol*n_group,nrow = n_split)
```

```
colnames(gate_mean)<-rep(colnames(df[,]),n_group)

#split data
set.seed(55,sample.kind = ’Rounding’)

## Warning in set.seed(55, sample.kind = "Rounding"):  non-uniform ’Rounding’ sampler used

for(i in 1:n_split){

  #randomly split data into main and auxiliary
  random<-runif(nrow(df))
  main_ind<-which(random>0.5)
  aux_ind<-which(random<0.5)
  aux_df<-df[aux_ind,]

  # train data on auxiliary sample
  rftreat<-randomForest(profits4w_real_p99_e~., data = (aux_df%>%filter(assigned==1)),
                         ntree=500,nodesize=5)
  rfbase<-randomForest(profits4w_real_p99_e~., data = (aux_df%>%filter(assigned==0)),
                        ntree=500,nodesize=5)

  # predict baseline and treatment outcomes
  B<-predict(rfbase,df)
  treat<-predict(rftreat,df)

  # specifying regression variables
  S<-treat-B #CATE
  ES<-mean(S)
  p<-mean(df$assigned) #take mean as propensity score
  x<-S-ES #excess CATE
  w<-df$assigned-p #weighted treatment var

  #derive Best Linear Predictor from main sample
  blp<-lm(profits4w_real_p99_e~B+w+I((w*x)),data=cbind(df,B,S,x,w)[main_ind,])
  blp_coef[i,]<-c(blp$coefficients[3],summary(blp)$coefficients[3:4,c(2,4)][1,],
                  blp$coefficients[4],summary(blp)$coefficients[3:4,c(2,4)][2,])


  #Group Average Treatment Effect
  qt<-quantile(S,seq(0,1,length.out = n_group+1))

  for(k in 1:n_group){
    G<-ifelse(S>qt[k] & S<qt[k+1],1,0)
    gate<-lm(profits4w_real_p99_e~B+I(w*G),data = cbind(df,B,S,x,w,G)[main_ind,])
    gate_coef[i,c((2*k)-1,2*k)]<-summary(gate)$coefficients[3,c(1,2)]

    # data preparation for later
    gate_mean[i,((k*mcol)-(mcol-1)):(k*mcol)]<-apply(df[which(G==1),],2,mean)
  }
}

# obtain median values
# data for each column does not correspond to the same split
apply(blp_coef,2,median) #median for Best Linear Predictor
```

```
##           B0        SE_B0   P_value_B0           B1        SE_B1   P_value_B1
##  7.844052727  3.973069134  0.046898626 -0.001291021  0.179207081  0.430404031

apply(gate_coef,2,median) #median for Grouped Average Treatment Effect

##        G1     SE_G1        G2     SE_G2        G3     SE_G3        G4     SE_G4        G5
##  4.438908 10.038764  7.626188  8.641414  9.552804  8.214033  8.437232  8.532416  8.500762
##     SE_G5
##  9.423633

# examining heterogeneity

for(k in 1:n_group) {
  nam <- paste("gate_mean", k, sep = "")
  assign(nam, gate_mean[,((k*mcol)-(mcol-1)):(k*mcol)])
}

het<-matrix(ncol = mcol,nrow = n_group) %>% as.data.frame() %>%
  mutate(gate=1:n_group)
colnames(het)<-c(colnames(df[,]),'gate')

for(k in 1:n_group) {
  het[k,(1:mcol)]<-apply(gate_mean[,((k*mcol)-(mcol-1)):(k*mcol)],2,median)
}

#choose covariates to plot
sub<-c('education','hhsize','age','wealthindex')
het_sub<-het[,c('gate',sub)] %>%
  gather("covariate", "median", -gate)

ggplot(het_sub, aes(gate, median, color = covariate)) +
  geom_line() +
  facet_wrap(~covariate,scales = "free_y") +
  theme_fivethirtyeight()
```



The R session information (including the OS info, R version and all packages used):

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
## 
## Matrix products: default
## 
## Random number generation:
##  RNG:     Mersenne-Twister 
##  Normal:  Inversion 
##  Sample:  Rounding 
##  
## locale:
## [1] LC_COLLATE=English_United States.1252  LC_CTYPE=English_United States.1252   
## [3] LC_MONETARY=English_United States.1252 LC_NUMERIC=C                          
## [5] LC_TIME=English_United States.1252    
## 
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base     
## 
## other attached packages:
##  [1] ggthemes_4.2.0      randomForest_4.6-14 forcats_0.5.0       stringr_1.4.0      
##  [5] dplyr_1.0.2         purrr_0.3.4         readr_1.4.0         tidyr_1.1.2        
##  [9] tibble_3.0.4        ggplot2_3.3.2       tidyverse_1.3.0     haven_2.3.1        
## [13] knitr_1.30         
## 
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.5        cellranger_1.1.0  pillar_1.4.7      compiler_4.0.3   
##  [5] dbplyr_2.0.0      highr_0.8         tools_4.0.3       digest_0.6.27    
##  [9] jsonlite_1.7.2    lubridate_1.7.9.2 evaluate_0.14     lifecycle_0.2.0  
## [13] gtable_0.3.0      pkgconfig_2.0.3   rlang_0.4.9       reprex_0.3.0     
## [17] cli_2.2.0         rstudioapi_0.13   DBI_1.1.0         curl_4.3         
## [21] xfun_0.19         withr_2.3.0       xml2_1.3.2        httr_1.4.2       
## [25] fs_1.5.0          generics_0.1.0    vctrs_0.3.5       hms_0.5.3        
## [29] grid_4.0.3        tidyselect_1.1.0  glue_1.4.2        R6_2.5.0         
## [33] fansi_0.4.1       readxl_1.3.1      farver_2.0.3      modelr_0.1.8     
## [37] magrittr_2.0.1    backports_1.2.0   scales_1.1.1      ellipsis_0.3.1   
## [41] rvest_0.3.6       assertthat_0.2.1  colorspace_2.0-0  labeling_0.4.2   
## [45] stringi_1.5.3     munsell_0.5.0     broom_0.7.3       crayon_1.3.4
```

```
Sys.time()
```

```
## [1] "2021-01-13 02:55:30 GMT"
```