rsa_ml.R

jonah

2021-06-08

```
library(haven)
```

```
## Warning: package 'haven' was built under R version 4.0.5
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## Warning: package 'forcats' was built under R version 4.0.5

## -- 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
```

```r
library(ggthemes)
```

```
## Warning: package 'ggthemes' was built under R version 4.0.5
```

```r
# 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, LASSO, and boosted trees

#some data cleaning

x<-read_dta('https://www.dropbox.com/s/yxgigmtcrut9fii/yop_analysis.dta?dl=1') %>%
  filter(e2==1) %>%  #filter results only from endline survey1
  select(assigned,S_K,S_H,S_P_m,
         admin_cost_us,groupsize_est_e,grantsize_pp_US_est3,group_existed,group_age,ingroup_hetero,ingro
         lowskill7da_zero,lowbus7da_zero,skilledtrade7da_zero,highskill7da_zero,acto7da_zero,aghours7da
         age,urban,ind_found_b,risk_aversion,inschool,
         D_1,D_2,D_3,D_4,D_5,D_6,D_7,D_8,D_9,D_10,D_11,D_12,D_13,
         profits4w_real_p99_e)

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

```
## [1] 0.007195332
```

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

```
## [1] 0.302204
```

```r
#eliminate NA values (robustness checks with lee and manski bounds to be added)
df<-x[which(complete.cases(x)),]

# create empty dataframes to store values
n_split<-100
n_group<-5
```

```r
blp_coef<-data.frame(B1=1:n_split,SE_B1=1:n_split,
                     P_value_B1=1:n_split,
                     B2=1:n_split,SE_B2=1:n_split,
                     P_value_B2=1:n_split)

gate_coef<-matrix(ncol = n_group*3,nrow = n_split) %>% as.data.frame()
colnames(gate_coef)<-paste(c("G",'SE_G','P_value'), rep(1:n_group, each=3), sep = "")

gate_diff<-data.frame(diff=1:n_split,SE=1:n_split,P_value=1:n_split)

clan<-matrix(ncol = 6,nrow = n_split*(ncol(df)-15)) %>% as.data.frame()
colnames(clan)<-c('estimate','SE','lower_conf','upper_conf','P_value','var')

clan_os<-matrix(ncol = 7,nrow = n_split*(ncol(df)-15)) %>% as.data.frame()
colnames(clan_os)<-c('g1','g1_lower_conf','g1_upper_conf','gk','gk_lower_conf','gk_upper_conf','var')


#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')
```

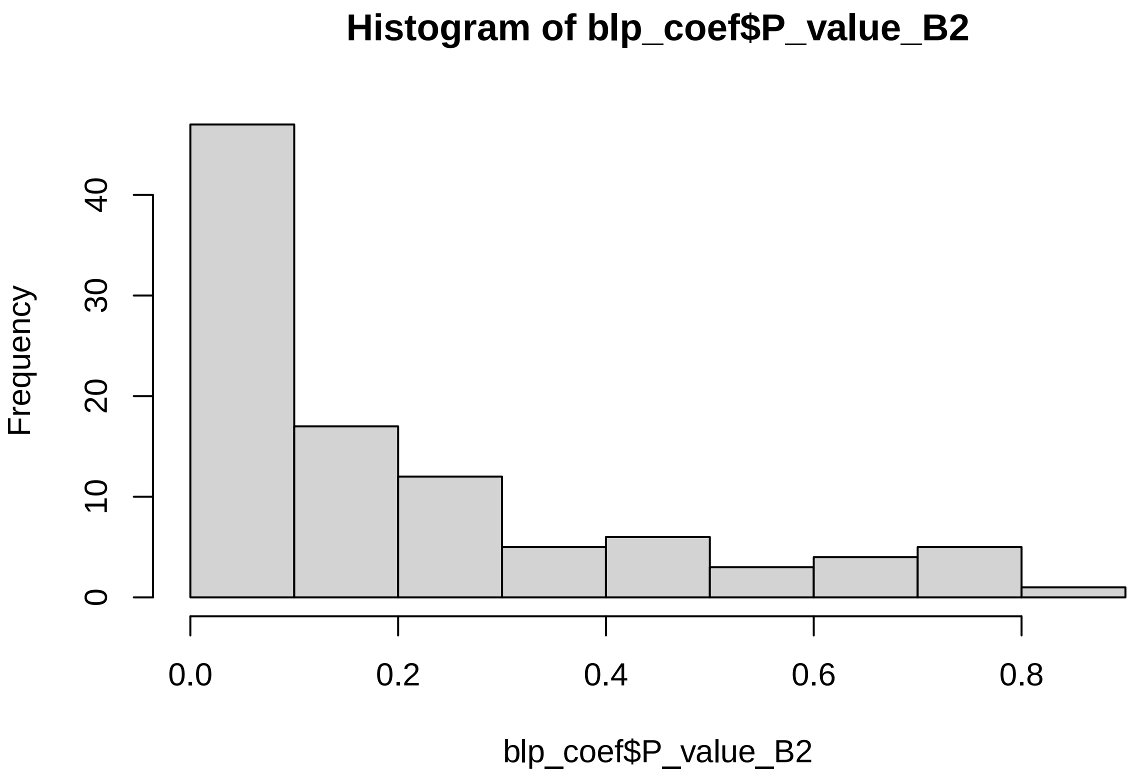
```
## Warning in set.seed(55, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
start<-Sys.time()
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,]
  main_df<-df[main_ind,]

  # train data on auxiliary sample
  rftreat<-randomForest(profits4w_real_p99_e~., data = (aux_df%>%filter(assigned==1)),
                      ntree=700,nodesize=7, mtry=2)
  rfbase<-randomForest(profits4w_real_p99_e~., data = (aux_df%>%filter(assigned==0)),
                     ntree=700,nodesize=7, mtry=2)

  # predict baseline and treatment outcomes on main sample
  B<-predict(rfbase,main_df)
  treat<-predict(rftreat,main_df)

  # specifying regression variables
  S<-treat-B #CATE: what the algorithm predicts is an individual's treatment effect
  ES<-mean(S) # the average predicted treatment effect
  p<-mean(main_df$assigned) #take mean as propensity score
```

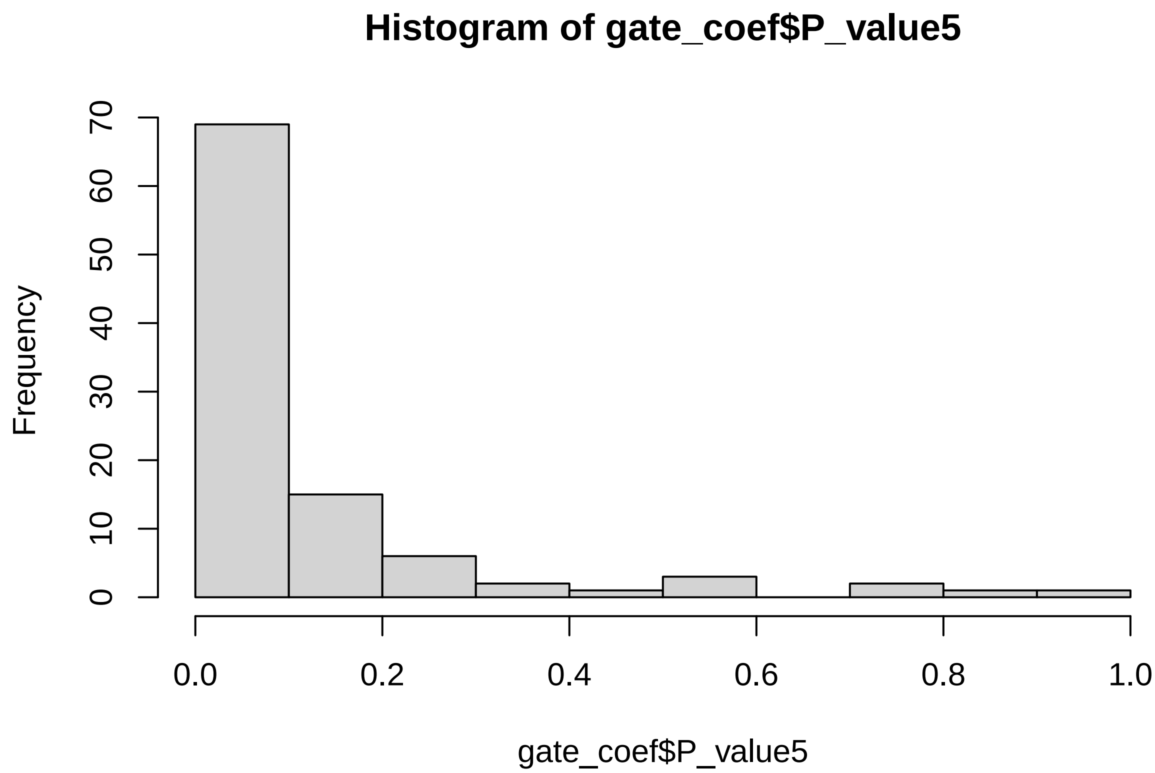
```r
  x<-S-ES #excess CATE: how far one's predicted treatment effect is from the mean
  w<-main_df$assigned-p #weighted treatment var

  #derive Best Linear Predictor from main sample
  blp<-lm(profits4w_real_p99_e~B+w+I((w*x)),data=cbind(main_df,B,S,x,w))
  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))
  diff<-cbind(main_df,B,S,w) %>%
    filter(S<=qt[2]|S>qt[n_group]) %>%
    mutate(G=ifelse((S>qt[n_group]),1,0))
  gate_diff[i,]<-summary(lm(profits4w_real_p99_e~B+I(w*G),data = diff))$coefficients[3,c(1,2,4)]

  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(main_df,B,S,x,w,G))
    gate_coef[i,((3*k)-2):(3*k)]<-summary(gate)$coefficients[3,c(1,2,4)]

    # data preparation for later
    gate_mean[i,((k*mcol)-(mcol-1)):(k*mcol)]<-apply(main_df[which(G==1),],2,mean)
  }
  #Classificaton Analysis (CLAN)
  diff2<-diff%>%select(-profits4w_real_p99_e,-assigned,-B,-w,-G,-ind_found_b
                      )
  n<-ncol(diff2)

  for (j in (1:n)){
    #two sample t test
    b<-t.test(diff2%>%filter(S>qt[n_group])%>%.[,j],diff2%>%filter(S<qt[2])%>%.[,j],
              alternative="two.sided",var.equal=F)
    clan[((i*n)-n+j),1]<-b$estimate[1]-b$estimate[2]
    clan[((i*n)-n+j),2]<-b$stderr
    clan[((i*n)-n+j),3:4]<-b$conf.int
    clan[((i*n)-n+j),5]<-b$p.value
    clan[((i*n)-n+j),6]<-colnames(diff2)[j]

    #one sample t test
    d<-t.test(diff2%>%filter(S<=qt[2])%>%.[,j])
    clan_os[((i*n)-n+j),1]<-d$estimate
    clan_os[((i*n)-n+j),2:3]<-d$conf.int
    d<-t.test(diff2%>%filter(S>qt[n_group])%>%.[,j])
    clan_os[((i*n)-n+j),4]<-d$estimate
    clan_os[((i*n)-n+j),5:6]<-d$conf.int
    clan_os[((i*n)-n+j),7]<-colnames(diff2)[j]
  }
}
end<-Sys.time()
end-start
```

```
## Time difference of 2.130664 mins
```

```r
# check distribution of p value
hist(blp_coef$P_value_B2)
```




```r
hist(gate_coef$P_value5)
```




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

```
##          B1       SE_B1  P_value_B1          B2       SE_B2  P_value_B2
## 13.32079920  6.22860608  0.03429353  0.56471456  0.33779305  0.10351390
```

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

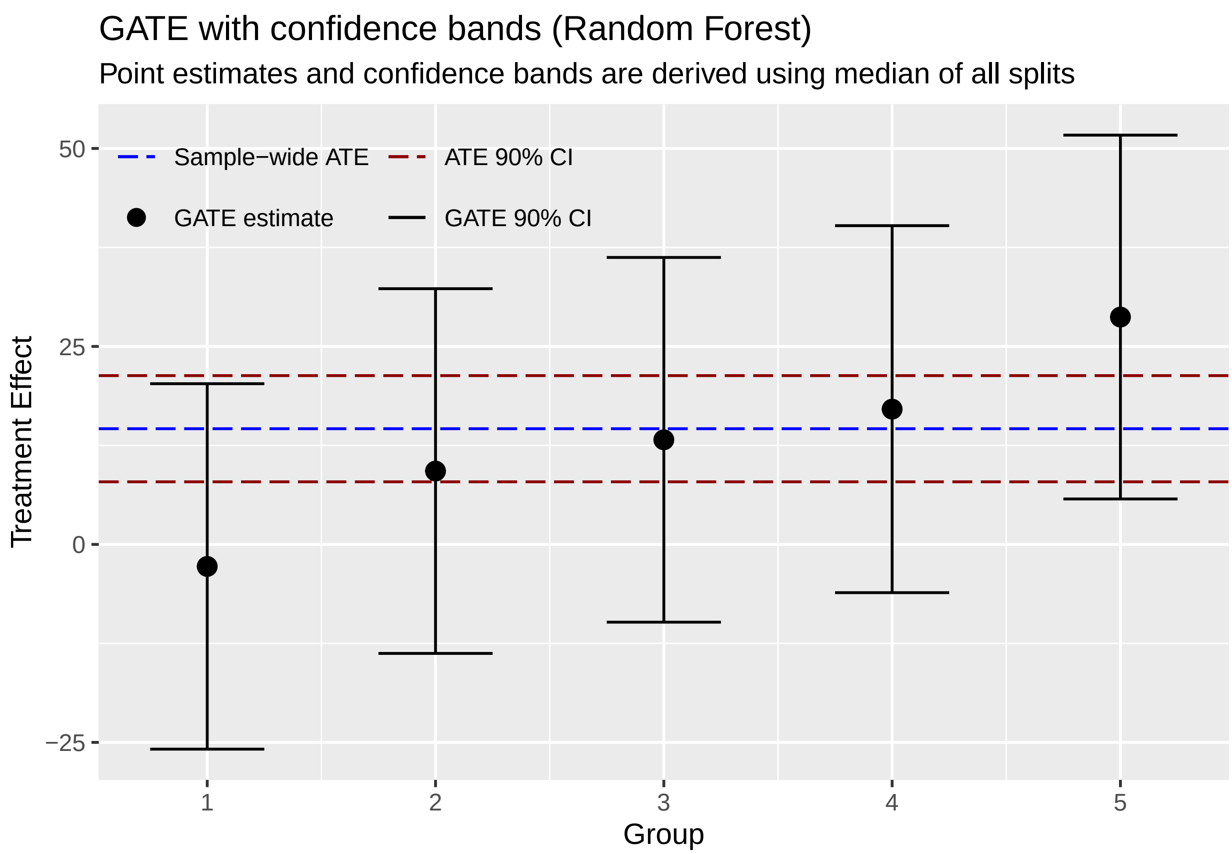
```
##         G1      SE_G1   P_value1         G2      SE_G2   P_value2         G3
## -2.7805900 14.0094663  0.5300773  9.2638580 13.9814957  0.4847700 13.2091327
##      SE_G3   P_value3         G4      SE_G4   P_value4         G5      SE_G5
## 13.9815718  0.3385163 17.0797064 14.0686392  0.2388240 28.7100929 13.9488239
##   P_value5
##  0.0407023
```

```
apply(gate_diff,2,median) #median for difference in GATE between G1 and Gk
```

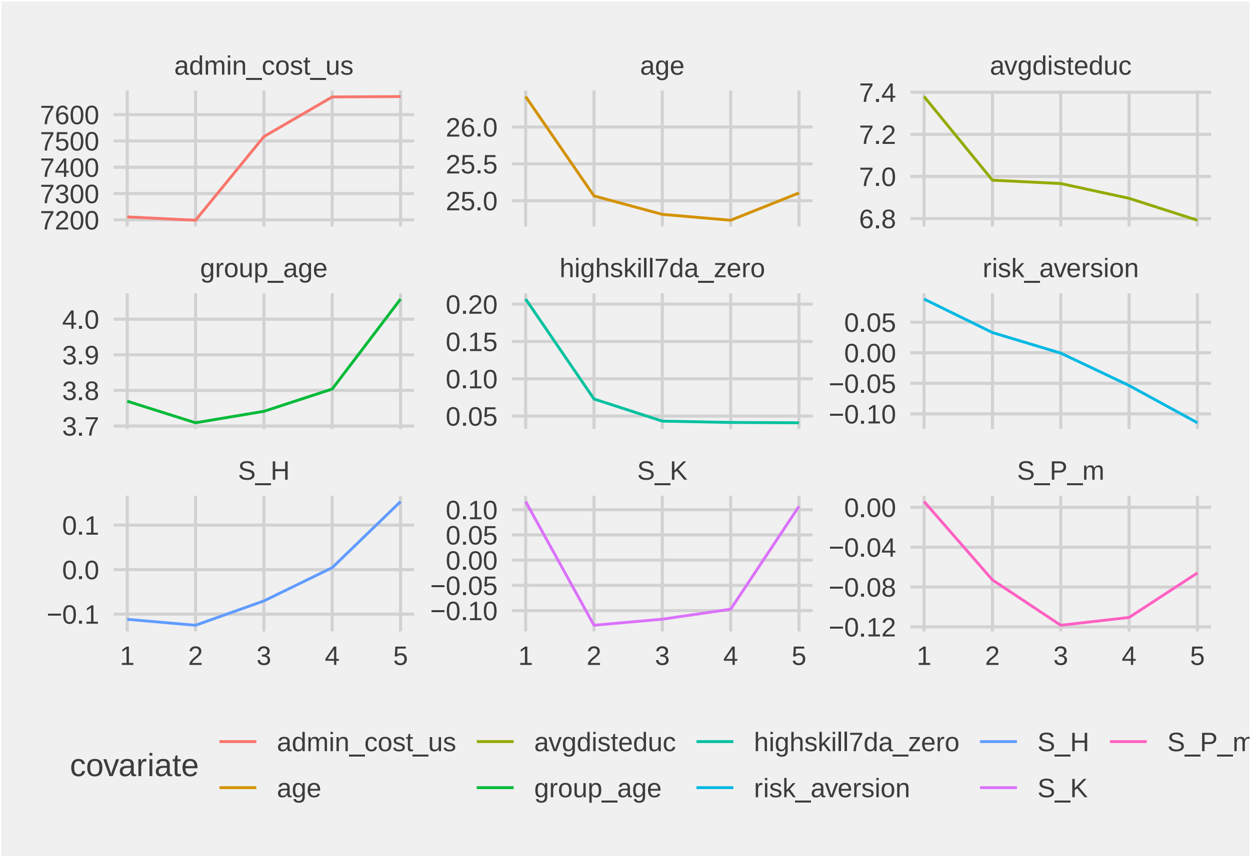
```
##        diff          SE     P_value
## 29.71258005 14.74344683  0.05459385
```

```
# plot GATE with confidence bands
ci<-data.frame(group = 1:5,estimate = NA,SE = NA,P_value=NA)
```

```r
for (k in 1:5){
ci[k,2:4]<-apply(gate_coef,2,median)[((k*3)-2):(k*3)]
}

labels <- c(point = "GATE estimate", error = "GATE 90% CI",
            blue = "Sample-wide ATE", darkred = "ATE 90% CI")
breaks <- c("blue", "darkred", "point", "error")

ggplot(ci, aes(x = group, y = estimate)) +
  geom_point(aes(color = "point"), size = 3) +
  geom_errorbar(width = .5, aes(
    ymin = estimate - (1.647 * SE),
    ymax = estimate + (1.647 * SE),
    color = "error"
  )) +
  scale_color_manual(values = c(
    point = "black",
    error = "black",
    blue = "blue",
    darkred = "darkred"
  ), labels = labels, breaks = breaks) +
  labs(
    title = "GATE with confidence bands (Random Forest)",
    subtitle = "Point estimates and confidence bands are derived using median of all splits",
    x = "Group",
    y = "Treatment Effect",
    color = NULL, linetype = NULL, shape = NULL
  ) +
  geom_hline(
    data = data.frame(yintercept = 14.61+c(-1,1)*(1.646*4.073)), # for endline 2 it's 18.19 (4.898)
    aes(yintercept = yintercept, color = "darkred"), linetype = "longdash"
  ) + # ATE and CI from original paper
  geom_hline(
    data = data.frame(yintercept = 14.61),
    aes(yintercept = yintercept, color = "blue"), linetype = "longdash"
  ) +
  guides(color = guide_legend(override.aes = list(
    shape = c(NA, NA, 16, NA),
    linetype = c("longdash", "longdash", "blank", "solid")
  ), nrow = 2, byrow = TRUE)) +
  theme(legend.position = c(0, 1),
        legend.justification = c(0, 1),
        legend.background = element_rect(fill = NA),
        legend.key = element_rect(fill = NA)) +


# 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('S_K','S_H','S_P_m','age','group_age','risk_aversion',
       'highskill7da_zero','admin_cost_us','avgdisteduc')
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()
```




```
#classification analysis (CLAN)
col<-clan$var %>% unique

clan_os_med<-matrix(ncol = 7,nrow = length(col)) %>% as.data.frame()
colnames(clan_os_med)<-c('g1','g1_lower_conf','g1_upper_conf','gk','gk_lower_conf','gk_upper_conf','var

clan_med<-matrix(ncol = 6,nrow = length(col)) %>% as.data.frame()
colnames(clan_med)<-c('estimate','SE','lower_conf','upper_conf','P_value','var')

#obtain medians of means and confidence interval for G1 and GK (1 sample t test)
for (i in col){
  clan_os_med[which(col==i),]<-clan_os %>% filter(var==i) %>% select(-var) %>%
    apply(2,median,na.rm=T) %>%
    as.list() %>% as.data.frame() %>% mutate(var=i)
}

# obtain medians for difference in means for G1 and GK (2 sample t test)
for (i in col){
  clan_med[which(col==i),]<-clan %>% filter(var==i) %>% select(-var) %>% apply(2,median) %>%
  as.list() %>% as.data.frame() %>% mutate(var=i)
}

clan_med[,c(6,1,2,3,4,5)] %>% filter(var!='S',P_value<=1) %>%
  arrange(-desc(P_value))
```

```
##                       var      estimate            SE    lower_conf      upper_conf
```

```
## 1           zero_hours   0.349594571   0.04795003   0.25471924  4.444699e-01
## 2                  D_1   0.091145833   0.02220585   0.04896873  1.335009e-01
## 3                 D_13   0.087284694   0.02235671   0.04402182  1.309002e-01
## 4      ingroup_dynamic   0.352413153   0.10316404   0.15121648  5.588421e-01
## 5       chores7da_zero  -6.150138073   1.76845665  -9.59419883 -2.754844e+00
## 6                  D_7   0.115271786   0.03328100   0.05229104  1.810940e-01
## 7      groupsize_est_e   2.361369100   0.72060147   0.87517628  3.768606e+00
## 8      aghours7da_zero  -3.115260404   1.08685720  -5.23023003 -1.065117e+00
## 9                  D_9   0.061148840   0.02214717   0.02264403  1.029690e-01
## 10       admin_cost_us 459.826451281 205.22924070  49.07525443  8.705776e+02
## 11               urban   0.117793658   0.04165581   0.03720363  1.989332e-01
## 12                D_11  -0.059784375   0.02007001  -0.09570322 -2.044628e-02
## 13      lowbus7da_zero  -2.171503939   0.77974328  -3.69992369 -6.793970e-01
## 14                 D_6  -0.065942029   0.02616227  -0.11361119 -1.614054e-02
## 15           grp_chair  -0.074470192   0.03470961  -0.14270041 -3.437261e-03
## 16 grantsize_pp_US_est3  -2.648882066  16.09705180 -35.09023178  2.996960e+01
## 17                 D_2  -0.090186311   0.04195046  -0.17169235 -5.748842e-03
## 18                 age  -1.433544862   0.56158312  -2.51885621 -3.358128e-01
## 19         nonag_dummy  -0.085917786   0.04671760  -0.17823138  6.325192e-03
## 20                 S_H   0.264760779   0.10802267   0.04706464  4.758188e-01
## 21                 D_3  -0.040023895   0.03055616  -0.10075323  2.363463e-02
## 22       risk_aversion  -0.204527659   0.10549150  -0.41584134  1.735648e-03
## 23                D_10  -0.037942887   0.02076429  -0.07612912  1.996287e-04
## 24                 D_4  -0.037533782   0.01687243  -0.07136260 -2.722847e-03
## 25         acto7da_zero   1.012783587   0.54088089  -0.09671513  2.067291e+00
## 26                 D_5  -0.002717391   0.02417082  -0.04782306  4.197519e-02
## 27       group_existed   0.086142475   0.05138409  -0.01418070  1.864656e-01
## 28   highskill7da_zero  -0.171583003   0.09726784  -0.37027753  3.013539e-03
## 29                 D_8  -0.032876959   0.01847899  -0.07044780  7.550675e-04
## 30                D_12  -0.021563499   0.02758376  -0.07938268  3.021144e-02
## 31      ingroup_hetero   0.089139717   0.11011052  -0.12931756  3.053158e-01
## 32           group_age   0.325223350   0.20722146  -0.08169331  7.343271e-01
## 33          avgdisteduc  -0.476146351   0.67265016  -1.77334082  7.513167e-01
## 34          grp_leader  -0.013229747   0.04782052  -0.10750772  7.906756e-02
## 35 skilledtrade7da_zero   0.659945039   0.94737765  -1.02907559  2.399280e+00
## 36    lowskill7da_zero  -0.534652691   0.50924664  -1.54778522  4.387693e-01
## 37               S_P_m  -0.090060429   0.10214612  -0.29733944  1.145042e-01
## 38                 S_K  -0.035199437   0.12741610  -0.28480003  2.041313e-01
## 39            inschool  -0.002631579   0.02116623  -0.04200406  4.003959e-02
##         P_value
## 1  3.501825e-12
## 2  4.116641e-05
## 3  8.182723e-05
## 4  3.645730e-04
## 5  4.831974e-04
## 6  4.866918e-04
## 7  6.947140e-04
## 8  1.426362e-03
## 9  1.740634e-03
## 10 3.064036e-03
## 11 3.166397e-03
## 12 3.390011e-03
## 13 3.866978e-03
## 14 5.092161e-03
```

```
## 15 7.857397e-03
## 16 9.285242e-03
## 17 9.388507e-03
## 18 1.093265e-02
## 19 1.127286e-02
## 20 1.316184e-02
## 21 1.571703e-02
## 22 2.573213e-02
## 23 2.764428e-02
## 24 3.244093e-02
## 25 3.838190e-02
## 26 4.343774e-02
## 27 4.986347e-02
## 28 5.449953e-02
## 29 5.595577e-02
## 30 5.689995e-02
## 31 5.799915e-02
## 32 6.385055e-02
## 33 6.459974e-02
## 34 7.486315e-02
## 35 9.525947e-02
## 36 1.102456e-01
## 37 1.516765e-01
## 38 1.539286e-01
## 39 1.589259e-01
```