rsa_ml.R

jonah

2021-04-06

```
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, 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,ingr
         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
```

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

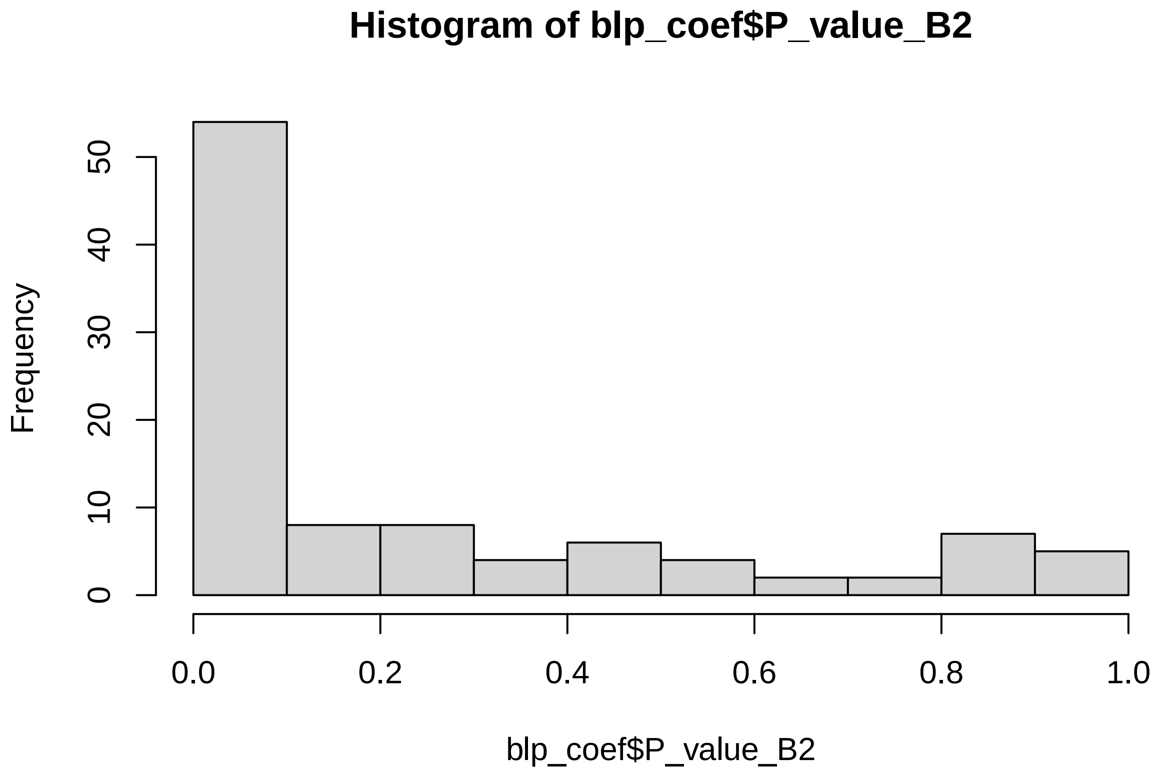
```
## [1] 0.302204
```

```
#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
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")
```

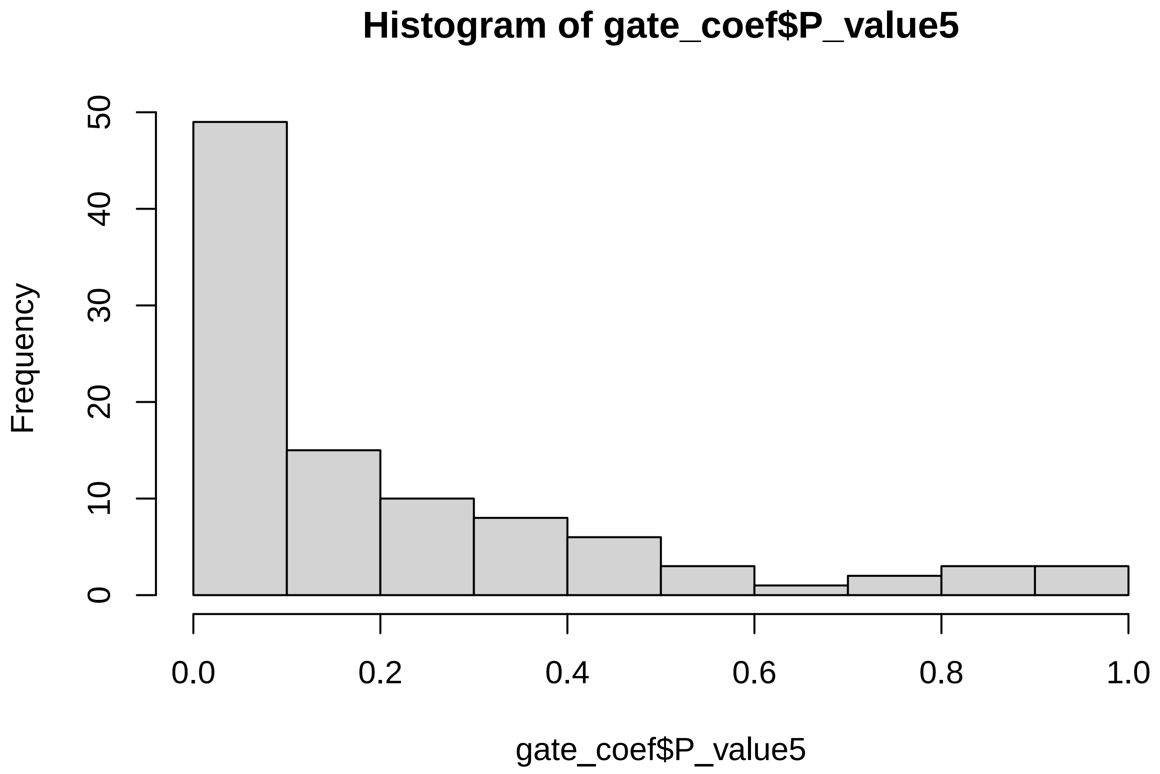
```r
#mcol<-ncol(df[,-f])
mcol<-ncol(df)
gate_mean<-matrix(ncol = mcol*n_group,nrow = n_split)
colnames(gate_mean)<-rep(colnames(df[,]),n_group)

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

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

```r
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=500,nodesize=5)
  rfbase<-randomForest(profits4w_real_p99_e~., data = (aux_df%>%filter(assigned==0)),
                       ntree=500,nodesize=5)

  # 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$assigned) #take mean as propensity score
  x<-S-ES #excess CATE: how far one's predicted treatment effect is from the mean
  w<-main_df$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,
                       -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)
  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]
  }
}

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



```
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
## 11.71240001  6.17144994  0.06023404  0.30983913  0.18788793  0.08355674
```

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

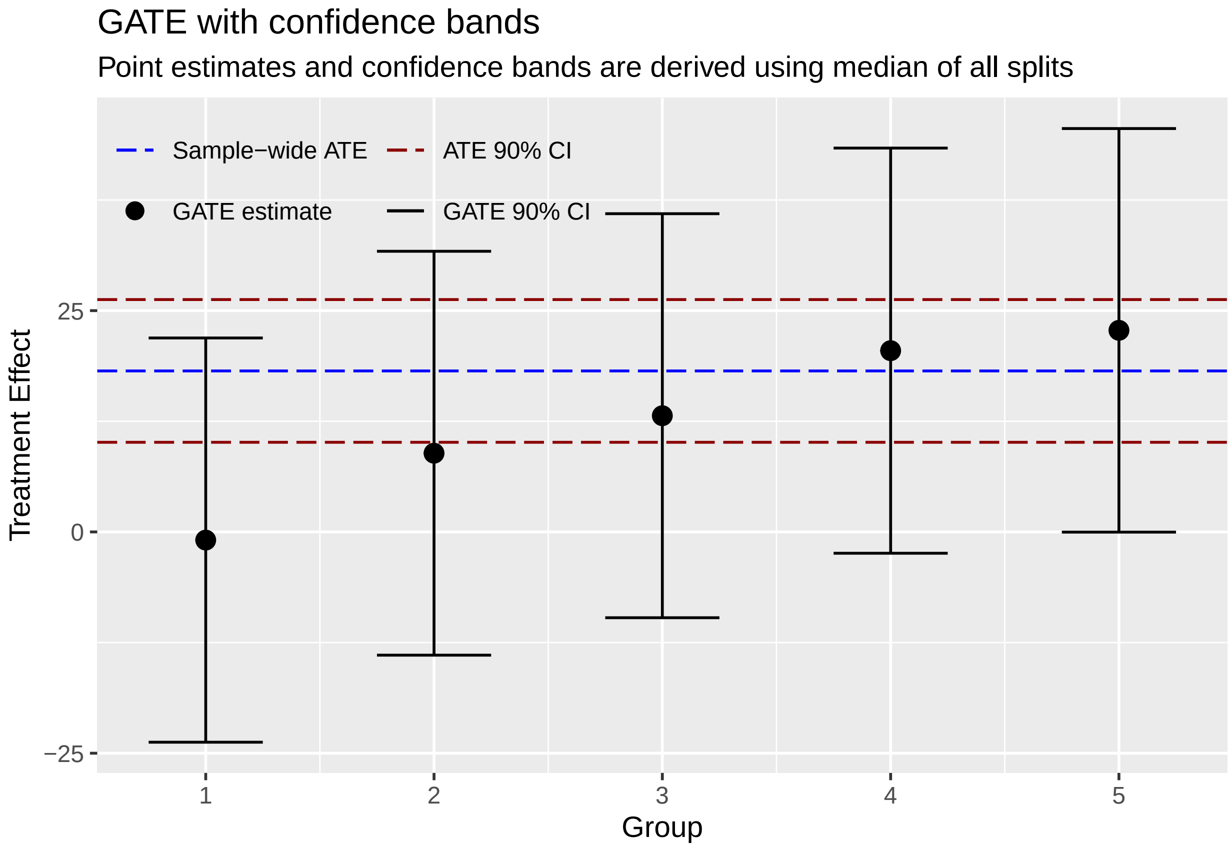
```
##         G1       SE_G1    P_value1          G2       SE_G2    P_value2          G3
## -0.9219174 13.8653376  0.4793581  8.8925038 13.8523339  0.4551271 13.1287592
##      SE_G3   P_value3          G4       SE_G4   P_value4          G5       SE_G5
## 13.8570655  0.3489968 20.4791444 13.8976400  0.1341243 22.7751362 13.8423085
##   P_value5
##  0.1041146
```

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

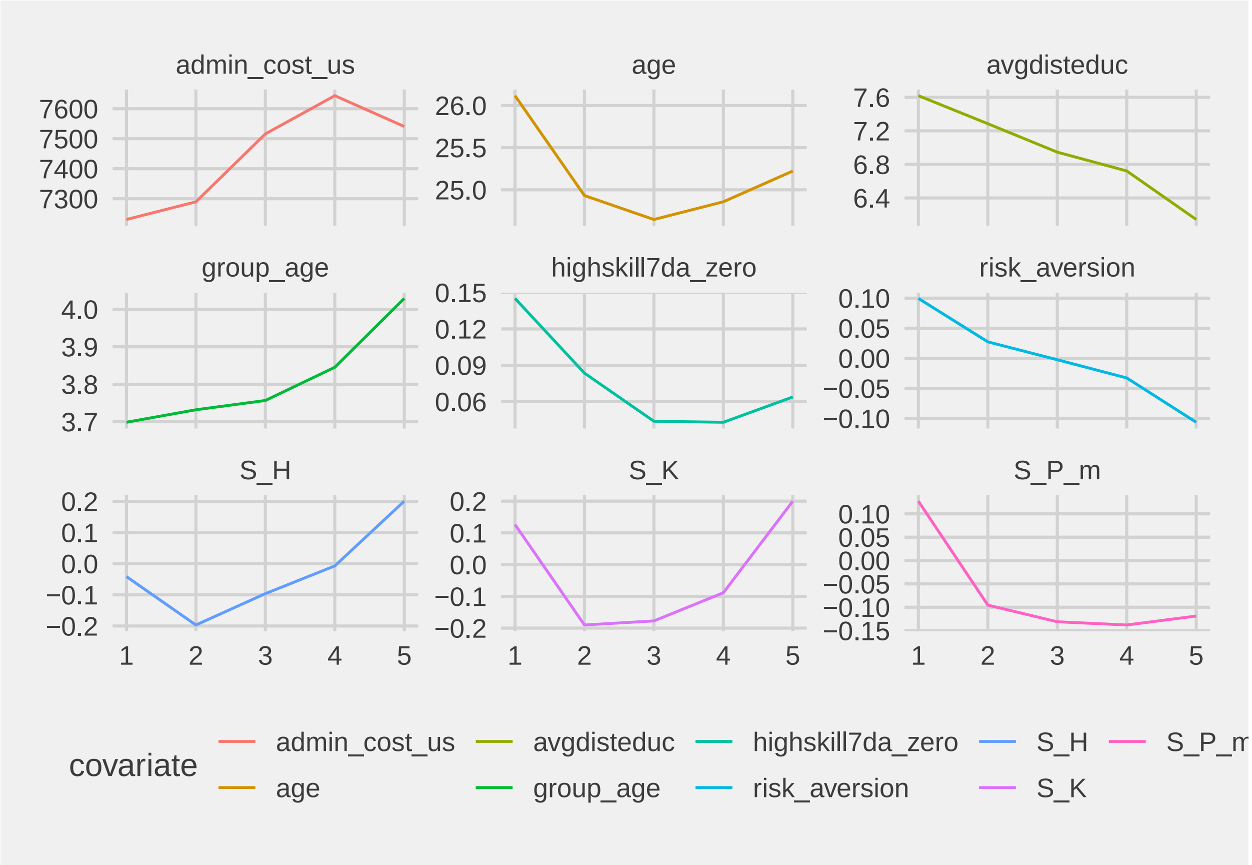
```
##      diff        SE   P_value
## 22.376135 14.907757  0.122659
```

```
# 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",
    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 = 18.19+c(-1,1)*(1.646*4.898)),
    aes(yintercept = yintercept, color = "darkred"), linetype = "longdash"
  ) + # ATE and CI from original paper
  geom_hline(
    data = data.frame(yintercept = 18.19),
    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<=0.05) %>%
  arrange(-desc(P_value))
```

```
##                  var   estimate          SE   lower_conf     upper_conf      P_value
## 1            S_P_m -0.2630031 0.10246666 -0.457633983 -0.0620133243 0.003258999
## 2            urban  0.1129148 0.04285560  0.032098859  0.1963831716 0.005223420
## 3    risk_aversion -0.2123593 0.10699286 -0.418020250  0.0005123978 0.006294560
## 4  ingroup_dynamic  0.2967003 0.10454470  0.080664569  0.5091663230 0.007139508
## 5              S_H  0.2178325 0.10896574  0.001175123  0.4324383948 0.011417867
## 6      avgdisteduc -1.4349099 0.65722910 -2.723171482 -0.2028861262 0.013389239
## 7     acto7da_zero  1.4120640 0.58653225  0.258238219  2.6566699792 0.013924192
## 8       zero_hours  0.1239901 0.05033799  0.023329934  0.2243181488 0.014748294
## 9   ingroup_hetero  0.2446042 0.10843910  0.032249766  0.4633295549 0.015551000
## 10             age -0.7354318 0.55885226 -1.815550785  0.3621369451 0.036670426
## 11 aghours7da_zero -1.7647849 1.14683252 -3.927053655  0.5721026982 0.047526597
## 12  chores7da_zero -3.6623076 1.82046628 -7.140949831  0.1275594687 0.048649724
## 13  lowbus7da_zero -0.6532051 0.78865306 -2.156656801  0.8516187287 0.048990597
```