In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
pd.set_option('display.max_columns', None)
```

# Self-assessed wealth

From Alatas et al (2012)

In [3]:

```
df_SA =  pd.read_stata("../data/112522-V1/Targeting_Indonesia/codeddata/intermediate_data/selfassessment.dta"
```

```
// Subjective Rank questions

use "$data\baseline\hh_sw.dta", clear
keep hhid sw02 sw03

recode sw02 (8=.)
recode sw03 (7=.)

gen RANK_sw02= sw02/6
gen RANK_sw03= sw03/5
```

In [4]:

```
df_SA.RANK_sw02.value_counts()
```

Out[4]:

```
0.500000    2271
0.333333    1386
0.666667     929
0.166667     894
0.833333     207
1.000000      47
Name: RANK_sw02, dtype: int64
```

In [5]:

```
df_SA["self_assessed_wealth"] = df_SA.RANK_sw02*6
```

In [6]:

```
df_SA["self_assessed_wealth"].value_counts()
```

Out[6]:

```
3.0    2271
2.0    1386
4.0     929
1.0     894
5.0     207
6.0      47
Name: self_assessed_wealth, dtype: int64
```

Stata code used to define `hhid` :

```
rename hhid idrt
tostring idrt, replace
replace idrt = "00" + idrt if length(idrt)==1
replace idrt = "0" + idrt if length(idrt)==2

gen hhid = hhea + idrt
```

In [7]:

```
df_SA["hamlet"] = df_SA.hhid.astype('str').str[:-3]
```

In [8]:

```
df_SA["hamlet"] = df_SA["hamlet"].astype(int)
```

In [9]:

```
df_SA["id"] = df_SA.hhid.astype('str').str[-3:]
```

In [10]:

```
df_SA["id"] = df_SA["id"].astype(int)
```

In [11]:

```
df_SA[['hamlet','id','self_assessed_wealth']].to_csv("../data/self_assessed_wealth.csv", index=False)
```

# CBT Ranking

## Normal

In [12]:
```
df_CBT =  pd.read_stata("../data/112522-V1/Targeting_Indonesia/codeddata/intermediate_data/RTS_community.dta"
```

In [13]:
```
df_CBT.rename(columns={'hhea':'hamlet','idrt':'id'}, inplace=True)
```

In [14]:
```
df_CBT['id'] = df_CBT.id.astype('int')
```

## Hybrid

In [15]:
```
df_hybrid =  pd.read_stata("../data/112522-V1/Targeting_Indonesia/codeddata/intermediate_data/rts_hybrid.dta"
```

In [16]:
```
df_hybrid.rename(columns={'hhea':'hamlet','idrt':'id','nhhrank_2':'nhhrank'}, inplace=True)
```

In [17]:
```
df_hybrid['id'] = df_hybrid.id.astype('int')
```

In [18]:
```
chosen_vars = ['hamlet','id','ranking_meeting','nhhrank','quota_final','maintreatment']
```

In [19]:
```
df_CBT_combined = pd.concat([df_CBT[chosen_vars],df_hybrid[chosen_vars]])
```

## Hamlet level variables

In [20]:
```
df_targeting =  pd.read_stata("../data/119802-V1/Final Data/TargetingTables.dta")
```

In [21]:
```
df_targeting[df_targeting.ELITE==1].maintreatment.value_counts()
```

Out[21]:
```
Hybrid            108
Full community    108
PMT                 0
Name: maintreatment, dtype: int64
```

In [22]:
```
df_targeting[df_targeting.ELITE==0].maintreatment.value_counts()
```

Out[22]:
```
PMT               209
Hybrid            108
Full community    106
Name: maintreatment, dtype: int64
```

In [23]:
```
itr = pd.read_stata("../data/119802-V1/Final Data/TargetingTables.dta", iterator=True)
itr.variable_labels()
```

Out[23]:
```
{'village': 'village',
 'N': 'number of households',
 'avg_degree': 'average degree in village',
 'var_degree': 'mean of degree distribution',
 'avg_clustering': 'average clustering coefficient',
 'size_giant': 'size of giant component',
 'fraction_giant': 'fraction of nodes in giant component',
 'apl': 'average path length',
 'l': 'average path length - rule of thumb',
 'l_giant': 'average path length - rule of thumb for giant component',
 'first_eig': 'first eigenvalue of adjacency matrix',
 'errorRate_New': '(mean) errorRate_New',
 'errorRate_New_SA': '(mean) errorRate_New_SA',
 'connectivity': 'average degree over village size',
 'avg_CONSUMPTION': '(mean) avg_CONSUMPTION',
 'avg_EDUCATIONHHHEAD': '(mean) avg_EDUCATIONHHHEAD',
 'avg_PMTSCORE': '(mean) avg_PMTSCORE',
 'SHAREAGRICULTURALHH': '(mean) SHAREAGRICULTURALHH',
 'avg_YREDUCATIONHHHEAD': '(mean) avg_YREDUCATIONHHHEAD',
 'EDUCATIONRTHEAD': '(mean) EDUCATIONRTHEAD',
 'YREDUCATIONRTHEAD': '(mean) YREDUCATIONRTHEAD',
 'LOGNUMBERHH': '(mean) LOGNUMBERHH',
 'LOGVILLAGESIZE': '(mean) LOGVILLAGESIZE',
 'klas_desa': '(mean) klas_desa',
 'inequality': '(mean) inequality',
 'kecagroup': 'Keca Group',
 'degree_Random': 'average degree of the 8 random households',
 'eig_Random': 'average eig centrality of the 8 random households',
 'pc1': 'Scores for component 1',
 'MISTARGETDUMMY': '(mean) MISTARGETDUMMY',
 'ELITE': '(mean) ELITE',
 'maintreatment': '',
 'PMT': 'PMT treatment',
 'COMMUNITY': 'Community treatment',
 'HYBRID': 'Hybrid treatment',
 'DISTANCEKEC': 'Distance to kecamatan in km',
 'PRIMARYSCHOOLPERHH': 'Primary school per household',
 'RELIGIOUSBUILDINGPERHH': 'Religious building per household',
 'yCONS': '',
 'yCOM': '',
 'ySA': '',
 'correct_ijk': '(mean) correct_ijk',
 'sim_correct_ijk': '(mean) sim_correct_ijk',
 'sim_correct_ijkT': '(mean) sim_correct_ijkT',
 'sim_correct_ijk_w': '(mean) sim_correct_ijk_w',
 'sim_correct_ijk_wT': '(mean) sim_correct_ijk_wT',
 'percent_DK': '(mean) percent_DK',
 'DKT': '(mean) DKT',
 'dk_data': '(mean) dk_data',
 'dist_ij_new': '(mean) dist_ij_new',
 'dist_ik_new': '(mean) dist_ik_new',
 'degree_avg': 'average degree',
 'clustering_avg': 'average clustering coefficient',
 'max_eigenvalue': 'maximal eigenvalue of adjacency matrix',
 'spectral_gap': 'spectral gap of adjacency amtrix',
 'fraction_in_giant': 'fraction of nodes in giant component',
 'errorRateSim1': '',
 'comType': '',
 'cumul': 'field rank of (pc1)    ',
 'n1': '',
 'negerrorRateSim': '',
 'cumulE': 'field rank of (negerrorRate)    ',
 'n2': '',
 'yRank': 'field rank of (ySA)    ',
 'n3': '',
 'pc1_90': '',
 'inequality_33': ''}
```

In [24]:

```
df_targeting.rename(columns={'village':'hamlet','ELITE':'elite_meeting'}, inplace=True)
```

In [25]:

```
df_targeting['hamlet'] = df_targeting['hamlet'].astype('int')
```

In [26]:

```
df_CBT_combined = df_CBT_combined.merge(df_targeting[['hamlet','elite_meeting']],on='hamlet',how='left')
```

In [27]:

```
df_CBT_combined.head()
```

Out[27]:

| | hamlet | id | ranking_meeting | nhhrank | quota_final | maintreatment | elite_meeting |
|---|---|---|---|---|---|---|---|
| **0** | 2 | 1 | 20 | 61 | 29 | Full community | 0.0 |
| **1** | 2 | 4 | 25 | 61 | 29 | Full community | 0.0 |
| **2** | 2 | 11 | 45 | 61 | 29 | Full community | 0.0 |
| **3** | 2 | 13 | 49 | 61 | 29 | Full community | 0.0 |
| **4** | 2 | 20 | 52 | 61 | 29 | Full community | 0.0 |

## Export

In [28]:

```
df_CBT_combined.to_csv("../data/community_meeting.csv", index=False)
```