OPA for Wealth Tax

07 March, 2019

```
# - inputs: none
# - outputs: all the original source values
call_params_f <- function(){



    #################
    ####### Research:
    #################
    research_so <- read_csv("rawdata/edits/research.csv")      #load data set that contains parameters
    # Elasticities
    ela1_so <- as.numeric(research_so[1,"param"])           # 0.5 - David. 2017
    ela2_so <- as.numeric(research_so[2,"param"])           # 0.5 - Jakobsen et al. 2018
    ela3_1_so <- as.numeric(research_so[3,"param"])         # 2   - Londono-Velez 2018
    ela3_2_so <-   as.numeric(research_so[4,"param"])       # 3   - Londono-Velez 2018
    ela4_1_so <- as.numeric(research_so[5,"param"])         # 23  - Brülhart et al. 2016
    ela4_2_so <- as.numeric(research_so[6,"param"])         # 34  - Brülhart et al. 2016

    #################
    ###### Data:
    #################
    # Tax base called from data (SCF and DINA) already accounts for the adjustments
    # due to population growth, tax avoidance,
    # Number of tax payers SCF & DINA
    cum_numberTaxpayers_scf_so <- c(1100144, 298649, 72143, 34552, 7119,  2555,  671)
    cum_numberTaxpayers_dina_so <- c(673449, 194548, 78434, 32751, 10236, 4491, 1597)
    # Tax base for brackets_po below
    cum_tax_base_scf_so <-  c(203.97, 121.63,  81.79, 59.18, 36.67, 27.46, 20.78)
    cum_tax_base_dina_so <- c(187.35, 135.38, 105.02, 80.11, 53.43, 36.94, 23.39)
    # Total wealth and number if households [SOURCE NEEDED]
    total_wealth_so <- 94e12   # [SOURCE]
    total_hhlds_so <- 129.4e6  # [SOURCE]
    # Forbes: average across 400 and value for 400th billionare
    average_wealth_top_400_so <- 7.2e9
    wealth_last_400_so <- 2.1e9
    # Macro-economy/demographics
    inflation_so <- 0.025      # CBO/JCT
    population_gr_so <- 0.01   # CBO/JCT
    real_growth_so <- 0.02     # CBO/JCT
    is_dina_public_so <- TRUE

    #################
    #####  Guesswork:
    #################
    hhld_gr_so <- 0.009
    growth_wealth_so <- 0.055

    return( sapply( ls(pattern= "_so\\b"), function(x) get(x) ) )
}
```

```r
invisible( list2env(call_params_f(),.GlobalEnv) )


################
#####  Notes:
################
### Source ----------->  Input ---------->  Model ---------->  Policy Estimates (output)
###  (_so)               (_in)               (_mo)                (_pe)
### values            functions          functions              values
###                   & values           & values
# - call_params_f - tax_elasticity_in_f  - tax_revenue_mo_f     - ten_year_revenue_pe
# - policy_f      - est_billionares_in_f - total_rev_mo_f       - ten_year_top_tax_pe
#                                        - ten_years_mo_f       - total_rev_pe
### arguments in functions should used "_var" and functions should "_f"
```

1 - Policy choices

The wealth tax applies to net worth (sum of all assets net of debts) above $50 million dollars, and follows the following structure:

```r
# - inputs: none
# - outputs: all the original source values
#### Policy:
policy_f <- function(){



    # brackets_po
    brackets_po <- c(10, 25, 50, 100, 250, 500,  1000) * 1e6
    tax_rates_po <- c(  0,    0, 0.02,  0.02,  0.02,  0.02, 0.03)
    starting_brack_po <- brackets_po[min(which(tax_rates_po>0))]
    next_increase_po <- brackets_po[min(which(tax_rates_po>0.02))]
    main_tax_po <- median(tax_rates_po)
    max_tax_po <- max(tax_rates_po)



    return( sapply( ls(pattern= "_po\\b"), function(x) get(x)) )
}
invisible( list2env(policy_f(),.GlobalEnv) )
knitr::kable(cbind("Bracket (millions of $)" = brackets_po/1e6,"Marginal Tax Rate (%)" = 100*tax_rates_
  kable_styling()
```

Bracket (millions of $)	Marginal Tax Rate (%)
10	0
25	0
50	2
100	2
250	2
500	2
1000	3

Household net worth above $50 million would be taxed at 2%. Any wealth over $1 billion would be taxed an additional 1% (a billionaire surtax).

2 - Compute tax avoidance elasticity

To calculate the tax revenue from this wealth tax, the extent of wealth tax evasion/avoidance is estimated based on recent research. Recent research shows that the extent of wealth tax evasion/avoidance depends crucially on loopholes and enforcement. The tax-avoidance elasticity is computed as the average elasticity from four studies. The table lists the four studies and the avoidance/evasion response to a 1% wealth tax.

Authors	Year	paper	Publisher	Avoidance/evasion response
Seim, David	2017	"Behavioral Responses to an Annual Wealth Tax: Evidence from Sweden"	American Economic Journal: Economic Policy, 9(4), 395-421	0.5
Jakobsen, Kristian, Katrine Jakobsen, Henrik Kleven and Gabriel Zucman.	2018	"Wealth Accumulation and Wealth Taxation: Theory and Evidence from Denmark"	NBER working paper No. 24371	0.5
Londono-Velez, Juliana and Javier Avila.	2018	"Can Wealth Taxation Work in Developing Countries? Quasi-Experimental Evidence from Colombia"	UC Berkeley working paper	2-3
Brülhart, Marius, Jonathan Gruber, Matthias Krapf, and Kurt Schmidheiny.	2016	"Taxing Wealth: Evidence from Switzerland"	NBER working paper No. 22376, 2016	23-34

Seim (2017) and Jakobsen et al. (2018) obtain small avoidance/evasion responses in the case of Sweden and Denmark, two countries with systematic third party reporting of wealth: a 1% wealth tax reduces reported wealth by less than 1%. Londono-Velez and Avila (2018) show medium avoidance/evasion responses in the case of Colombia where enforcement is not as strong: a 1% wealth tax reduces reported wealth by about 2-3%. The study for Switzerland, Brülhart et al. (2016) is an outlier that finds very large responses to wealth taxation in Switzerland: a 1% wealth tax lowers reported wealth by 23-34%. This extremely large estimate is extrapolated from very small variations in wealth tax rates over time and across Swiss cantons and hence is not as compellingly identified as the other estimates based on large variations in the wealth tax rate. Switzerland has no systematic third party reporting of assets which can also make tax evasion responses larger than in Scandinavia.

```
# input: elasticity parameters from research, main tax, adjutment factor
# ouptut: final elasticity (final_ela_in), evasion parameter (evasion_param_in)
tax_elasticity_in_f <- function(ela1_var = ela1_so, ela2_var = ela2_so, ela3_1_var = ela3_1_so,
                                ela3_2_var = ela3_2_so, ela4_1_var = ela4_1_so, ela4_2_var = ela4_2_so,
                                main_tax_var = main_tax_po){
```

```
    final_ela_in <- mean(c(ela1_var, ela2_var, (ela3_1_var + ela3_2_var)/2, (ela4_1_var + ela4_2_var)/2
    evasion_param_in <- main_tax_var * final_ela_in



    return(list("final_ela_in" = final_ela_in,
                "evasion_param_in" = evasion_param_in))
}
invisible( list2env(tax_elasticity_in_f(),.GlobalEnv) )

# ls(pattern = "_in\\b|_in_")
```

The final 16% tax avoidance/evasion response to a 2% wealth tax was computed as and average across these four studies (2%*(0.5+0.5+2.5+28.5)/4)

3 - Data Sources

Three data sources were used in this analysis:

- Survey of Consumer Finances (SCF) from the Federal Reserve Board. Latest year available: 2016.
- Distributional National Accounts (DINA): estimates wealth by capitalizing investment income from income tax returns. Latest year available: 2019.
- Forbes 400: provides the best estimate of billionares in the US. Last year available: 2018.

3.1 Data cleanning

From each data set three variable were extracted: `networth` that contain information on wealth, `weigths` represents the number of households that each observation represents and `data` which tracks the data of origin.

The following transformations were applied to the data:

- Each observation in DINA is aggregated into broups of 5 observations to anonymize the data.
- SCF was aged by inflating the number of households and wealth uniformly to match the aggregate projections for population and total household wealth from the Federal Reserve Board. After that, SCF wealth was scaled to match the total of DINA minus the wealth of Forbes (to prevent double counting of wealth).
- After combinind (appending) all three data sources, the population weights of SCF and DINA where combined by the taking the average.

```
##### Reproducing do file ('wealthtax.do')
### Forbes data
df_forbes <- read_dta("rawdata/forbes_20112018_bdays.dta")
df_forbes1 <- df_forbes  %>% filter(forbes_yr == 2018) %>%
  mutate("networth" = net_worthmillions * 1e6,
           "weight" = 1,
           "data" = "FB400") %>%
  select(data, networth, weight) %>%
```

```r
  filter( !is.na(networth) )
forbesmin <- min(df_forbes1$networth)
f400tot <- sum(df_forbes1$networth * df_forbes1$weight) / 1e12
#cat("TOTAL FORBES NETWORTH 2018 (Tr) = ", f400tot,  "FORBES MIN WEALTH 2018 = ", forbesmin)


### DINA data
####### This section uses data that cannot be shared for confidentiality reasons
####### Below is the code used to aggregate the data.
####### If you have access to the orginal data set, set is_dina_public_so = TRUE.
####### To obtain this data please contact Gabriel Zucman at zucman@gmail.com
####### The file that you will obtain should have the following signature:
####### in R: digest("usdina2019.dta", file = TRUE) produces: "2f5d529b1e89e39171927dc28bfebbe4"
####### in Stata: datasignature produces: 282866:4(49628):4083279708:1806586907

if(is_dina_public_so){
  # paste below the path to where usdina2019.dta is in your computer
  df_dina_first <- read_dta("/Users/fhoces/Desktop/opa-wealthtax_test/rawdata/materials/usdina2019.dta")
  df_dina <- df_dina_first %>%
  group_by(id) %>%
  summarise("networth" = round(sum(hweal)),  # rounding of networth is to make it compatible with Stata
            "weight" = mean(dweght)/1e5)

  totw_dina <- sum(df_dina$networth * df_dina$weight) / 1e12
  #cat("TOTAL DINA NETWORTH 2019  (Tr) ", totw_dina)

  totn_dina <- sum(df_dina$weight)

  df_dina$data <- "DINA"

  # Aggregate into bins of 5 househodls info to protect confidentiality
  df_dina1 <- df_dina %>%
    mutate("aux_id" = 1:dim(df_dina)[1]) %>%
    arrange(desc(networth),aux_id) %>%
    # Still not clear what the role of the "+3" is.
    mutate("group" = floor((1:dim(df_dina)[1] + 3) / 5)) %>%
    group_by(group) %>%
    summarise("weight" = sum(weight),
              "networth" = mean(networth)) %>%
    mutate("data" = "DINA") %>%
    select("data", "networth", "weight")
  write_dta("analysis_data/dina.dta", data = df_dina1)
  df_dina1 <- read_dta("analysis_data/dina.dta")
  #### End of confidential section
} else {
  df_dina1 <- read_dta("analysis_data/dina.dta")
  totw_dina <- sum(df_dina1$networth * df_dina1$weight) / 1e12
}

### SCF data
df_scf <- read_dta("rawdata/rscfp2016.dta")
totw_scf <- sum(df_scf$networth * df_scf$wgt) / 1e12
```

```r
#cat("TOTAL SCF NETWORTH 2016  (Tr)", totw_scf)
# Increase the population weigths to reflect population growth from 2016 to 1019
df_scf <- df_scf %>% mutate("wgt2019" = round( wgt * (1 + hhld_gr_so)^( 2019 - 2016 )))
totw <- sum(df_scf$networth * df_scf$wgt2019) / 1e12
totn <- sum(df_scf$wgt2019)


# Rescaling SCF to match total total wealth reported in DINA excluding the f400
df_scf1 <- df_scf %>% mutate("networth" = networth * ( totw_dina - f400tot ) / totw,
                             "weight" = wgt2019,
                             "data" = "SCF") %>%
  select(data, networth, weight)

# Combine three data sources
df <- rbind(df_forbes1, df_scf1, df_dina1)

# If observation is in SCF or DINA, then divide their weights in 2
df$weight <- with(df, ifelse(data=="SCF" | data=="DINA",
                             round(weight/2), weight) )
# All obs from SCF and DINA that have wealth above the min of forbes are droped to avoid duplications
df <- df %>% filter( !(networth > forbesmin & ( data == "SCF" | data == "DINA" ) ) )

# df %>%
#   summarise( mean(networth), sd(networth) )

total_wealth <- df %>%
  summarise(sum(networth * weight) / 1e12) %>%
  as.numeric()
billio_wealth <- df %>%
  filter(networth >= 50e6) %>%
  summarise(sum(networth * weight) / 1e12) %>%
  as.numeric()

#cat("Total wealth (in trillions) is ", total_wealth, ". Wealth for billionares total wealth is ", bill

write_dta("analysis_data/wealth.dta", data = df, version = 11)

# Very small differences with stata output (but it should be zero differences)
# wealth <- read_dta("~/Downloads/wealthtaxsim/data/wealth.dta")
# diff_aux <- abs( df$networth -  wealth$networth )
# summary(abs(diff_aux))
#     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#    0.000    0.000    0.008    0.928    0.174 3328.000

total_wealth_scf_in <- df_scf %>% summarise(sum(networth * wgt)/1e12) %>%  as.numeric()
total_wealth_in <- df %>%  summarise(sum(networth * weight)/1e12)
```

The total household net worth projection is $94 trillion for 2019 (the SCF records a total household net worth of $87 trillion in 2016).

3.2 Generating Percentiles and micro-percentiles

In this section, the microdata generated before (`wealth.dta`) is aggregated in to percentiles and fractions of a

percentile. The final analytic file contains: percentile or fraction of percentile (`gperc`), number of households in that group (`nb`), lowest level of wealth in that group (`thres`) and average level of wealth in that group (`avg`)

```
*# The code below does not run as part of the dynamic document and is presented only
*# for reproducibility purposes.
*# To run the code below you will require a license of Stata 11 or higher. You will also need to
*# place a copy of the file 'gperc.ado', located in the 'rawdata' folder of this repository
*# into your ado folder.
global ado_dir "INCLUDE YOUR FILE PATH FOR DATA HERE"
global  datawork   "INCLUDE YOUR FILE PATH FOR ADOs HERE"

sysdir set PERSONAL "$ado_dir"

* creating an excel table for the simulation
use $datawork/wealth.dta, clear
gperc networth [w=weight], matname(wealthperc)
mat list wealthperc
clear
svmat wealthperc, names(col)
qui compress
export excel using "$datawork/wealthperc.xlsx", first(var) replace
*REPLACE AT THE END
*export delimited using "/Users/fhoces/Desktop/sandbox/opa-wealthtax/analysis_data/tax_grid.csv", repla
```

4 - Number of affected households and their total tax base

```
# TO DELETE

# - inputs: evasion_param_in, wealth_last_400_so, average_wealth_top_400_so,
# cum_numberTaxpayers_scf_so, cum_numberTaxpayers_dina_so, cum_tax_base_scf_so,
# cum_tax_base_dina_so
# - outputs: non_evaded_tax_in, totax_last_400_in, totax_average_400_in, totax_total_400_in,
# pareto_scale_in, ratio1_in, missing_fraction_in, missing_taxbase_in, num_billionares_in,
# final_taxbase_b_in, top_tax_rev_in, cum_numberTaxpayers_scf_in, cum_numberTaxpayers_dina_in,
# cum_tax_base_scf_in, cum_tax_base_dina_in
est_billionares_in_f <- function(evasion_param_var = evasion_param_in,
                                 wealth_last_400_var = wealth_last_400_so,
                                 average_wealth_top_400_var = average_wealth_top_400_so,
                                 cum_numberTaxpayers_scf_var = cum_numberTaxpayers_scf_so,
                                 cum_numberTaxpayers_dina_var = cum_numberTaxpayers_dina_so,
                                 cum_tax_base_scf_var = cum_tax_base_scf_so,
                                 cum_tax_base_dina_var = cum_tax_base_dina_so){
    non_evaded_tax_in <- (1 - evasion_param_var)

    # Post evasion taxable wealth
    totax_last_400_in <- non_evaded_tax_in * wealth_last_400_var/1e9
    totax_average_400_in <- non_evaded_tax_in * average_wealth_top_400_var/1e9
    totax_total_400_in <- 400 * (totax_average_400_in - 1)                    # above one billion

    # Compute pareto scale of wealth distribution
    aux1 <- average_wealth_top_400_var/wealth_last_400_var
```

```
    pareto_scale_in <- aux1/(aux1-1)
    ratio1_in <- round(aux1, 1)

    # Compute missing fraction of taxable wealth above 1 billion
    one_billion <- 1
    missing_fraction_in <- ( totax_last_400_in / one_billion )^( pareto_scale_in - 1 ) - 1

    # Compute missing wealth (billions of dollars) to tax above one billion dollars
    missing_taxbase_in <- missing_fraction_in * totax_total_400_in

    # Compute total number of billionares. Extrapolation from forbes 400.
    #Diff between original document (911) and below (906) is due to rounding in inputs
    num_billionares_in <- 400 * ( totax_last_400_in / 1 ) ^ pareto_scale_in
    final_taxbase_b_in <- totax_total_400_in + missing_taxbase_in

    #This is the taxed money (the aboves is the totale taxable) in billions  (minus one is just to matc
    top_tax_rev_in <- final_taxbase_b_in * 0.01                                    # ALERT: Hard coded

    #edit the number of billionares
    cum_numberTaxpayers_scf_in <- c(cum_numberTaxpayers_scf_var[1:6], num_billionares_in)
    cum_numberTaxpayers_dina_in <- c(cum_numberTaxpayers_dina_var[1:6], num_billionares_in)
    #edit the tax revenue for billionares
    cum_tax_base_scf_in <- c(cum_tax_base_scf_var[1:6], top_tax_rev_in)
    cum_tax_base_dina_in <- c(cum_tax_base_dina_var[1:6], top_tax_rev_in)

    return( list( "non_evaded_tax_in" = non_evaded_tax_in, "totax_last_400_in" = totax_last_400_in,
                  "totax_average_400_in" = totax_average_400_in, "totax_total_400_in" =
                    totax_total_400_in, "pareto_scale_in" = pareto_scale_in, "ratio1_in" = ratio1_in,
                  "missing_fraction_in" = missing_fraction_in, "missing_taxbase_in" =
                    missing_taxbase_in, "num_billionares_in" = num_billionares_in,
                  "final_taxbase_b_in" = final_taxbase_b_in, "top_tax_rev_in" = top_tax_rev_in,
                  "cum_numberTaxpayers_scf_in" = cum_numberTaxpayers_scf_in,
                  "cum_numberTaxpayers_dina_in" = cum_numberTaxpayers_dina_in,
                  "cum_tax_base_scf_in" = cum_tax_base_scf_in,
                  "cum_tax_base_dina_in" = cum_tax_base_dina_in ) )
}

invisible( list2env(est_billionares_in_f(),.GlobalEnv) )
#rm(list = c("aux1", "one_billion"))
```

To compute the relevant universe the evasion parameter of 16% is applied to both the threshold and the average wealth of each percentile (and fraction of a percentile)

```
# tax_base_grid --->
# Change the following line at the end.
#grid <- read.csv("analysis_data/tax_grid.csv")  %>%
grid <- read.csv("analysis_data/taxBaseGridUpdated.csv")  %>%
  filter(!is.na(gperc))
# print(head(grid)) ## check that app has access to this file

# Wealth per bin (percentile) after evasion
grid$thresNew <- (1 - evasion_param_in) * grid$thres
grid$avgNew <- (1 - evasion_param_in) * grid$avg
```

```r
# TO DELETE?
###### The following section depends on the cleaning data chunk
if (FALSE){
#QUESTION: Why counts from original source differ?
#    df_dina1 %>% filter(networth>50e6) %>% summarise(sum(weight))
#    grid %>% filter(thresNew>50e6) %>% summarise(sum(nb))
#    df_scf %>% filter(networth>50e6) %>% summarise(sum(weight))
target_hhlds_mo <- df %>%
  filter(networth>starting_brack_po) %>%
  summarise(sum(weight)) %>%
  as.numeric()
target_hhlds_round <- round(target_hhlds_mo/1000) * 1000

target_hhlds_scf_mo <- df_scf1 %>%
  filter(networth>starting_brack_po) %>%
  summarise(sum(weight)) %>%
  as.numeric()
target_hhlds_round_scf <- round(target_hhlds_scf_mo/1000) * 1000

target_hhlds_dina_mo <- df_dina1 %>%
  filter(networth>starting_brack_po) %>%
  summarise(sum(weight)) %>%
  as.numeric()
target_hhlds_round_dina <- round(target_hhlds_dina_mo/1000) * 1000

}

##### Num hhlds:
#no avoidance
target_hhlds_noav_mo <- grid %>%
  filter(thres > starting_brack_po) %>%
  summarise(sum(nb))
target_hhlds_noav_round <- format(round(target_hhlds_noav_mo / 1000) * 1000 , scientific = FALSE)

#with avoidance
target_hhlds_mo <- grid %>%
  filter(thresNew > starting_brack_po) %>%
  summarise(sum(nb))
target_hhlds_round <- format(round(target_hhlds_mo / 1000) * 1000, scientific = FALSE)

### Billionares:
#no avoidance
target_hhlds_bn_noav_mo <- grid %>%
  filter(thres > next_increase_po) %>%
  summarise(sum(nb))
target_hhlds_bn_noav_round <- round(target_hhlds_bn_noav_mo / 1000) * 1000

#with avoidance
target_hhlds_bn_mo <- grid %>%
  filter(thresNew > next_increase_po) %>%
  summarise(sum(nb))
target_hhlds_bn_round <- round(target_hhlds_bn_mo / 1000) * 1000
```

```
##### Total Taxable Wealth:
tax_base_total_noav_mo <- grid %>%
  filter(thres > starting_brack_po) %>%
  summarise(sum((avg - starting_brack_po) * nb)/1e12)

#with avoidance
#2% above 50m
tax_base_total_mo <- grid %>%
  filter(thresNew > starting_brack_po) %>%
  summarise(sum((avgNew - starting_brack_po) * nb)/1e12)

#billionares additional 1%
tax_base_total_surtax_noav_mo <- grid %>%
  filter(thres > next_increase_po) %>%
  summarise(sum((avg - next_increase_po) * nb)/1e12)
tax_base_total_surtax_mo <- grid %>%
  filter(thresNew > next_increase_po) %>%
  summarise(sum((avgNew - next_increase_po) * nb)/1e12)
```

In 2019, there would be around 63000 households liable to the wealth tax (78000 before accounting for avoidance). This would be less than 0.05% of the 130 million US households in 2019.

4.1 - 2% tax to all wealth above $50 millions

The 62589 households with assests above $50 million dollars would have a total taxable wealth (above the $50m each) of $8.9 trillion, i.e. approximately 10% of the $94 trillion population-wide total household net worth.

4.2 - 1% additional tax to all wealth above $1 billion

The 963 households with assests above $1 billion dollars would have a total taxable wealth (above the $1b each) of $2.2 trillion, i.e. approximately 2% of the $94 trillion population-wide total household net worth.

5 - Total tax revenue in one year

```
# Total tax collected in a year
# amount from 2%
# amount from extra 1%

#This function computes the total tax collected for a tax unit with wealth "wealth_var", applying "taxr
# - inputs: wealth, tax rates, brackets to tax
# - ouputs: total tax collected
get_tax_rev <- function(wealth_var = wealth_aux, taxrates_var = tax_rates_po,
                        brackets_var = brackets_po) {
    ## expecting taxLevels in percentage
    # taxLevels <- taxLevels / 100
    if (length(brackets_var) != length(taxrates_var)){
      stop("Tax brackets and tax rates do not match")
    }
   # Compute max taxable wealth per bracket
    max_tax_per_brack <- c(diff(c(0, brackets_var)), 1e100)
```

```
   # Substract wealth minus tax bracket. If wealth above a given bracket (difference is larger than max
   # then assign max taxable wealth to that given bracket
    to_tax <- ifelse( wealth_var - c(0, brackets_var) > max_tax_per_brack,
                      max_tax_per_brack,
                      ( wealth_var - c(0,brackets_var) ) )
   # If wealth if lower than a given bracket (difference between wealth and bracket is negative), then
    to_tax <- ifelse( to_tax<0, 0, to_tax )
    # Apply trax rates to each corresponding bracket and all together
    total_tax <- sum( to_tax * c(0, taxrates_var) )
    return(total_tax)
}

# IMPORTANT: this (similar to getTaxBasePerBracket) was differing from simple
# calculation below because this was not subseting to wealth above 50m.
# computes tax payed by each average wealth per percintile (up to 2%)

## gets taxes paid per group (percentile and micropercentile)
get_tax_rev_per_group <- function(grid_var = grid, taxLevels_var = tax_rates_po, brackets_var1 = bracke
  grid_var <- grid_var %>% filter(thresNew > starting_brack_po)
  aux_var <- sapply(grid_var$avgNew,
                    function(x) get_tax_rev(wealth_var = x,
                                            taxrates_var = taxLevels_var,
                                            brackets_var = brackets_var1))
  return(sum(grid_var$nb * aux_var) / 1e9)
}

tax_rev_init_mo <- get_tax_rev_per_group(taxLevels = c(tax_rates_po[-7], 0.02))
top_tax_rev_in <- get_tax_rev_per_group(taxLevels = c(rep(0,6), 0.01))
total_tax_rev <- get_tax_rev_per_group(taxLevels = tax_rates_po)
#199.7889

# The following replicates stata code more closely and seems more straightforward.
# Howevere it differs more from the code in the app. Consider this in both (app and DD)
# in the future
if (FALSE){
    #with avoidance
    #2% above 50m
    tax_rev_init_mo <- grid %>%
      filter(thresNew > starting_brack_po) %>%
      summarise(sum((avgNew - starting_brack_po) * nb * 0.02)/1e9) %>% as.numeric()

    #billionares additional 1%
    top_tax_rev_in <- grid %>%
      filter(thresNew > next_increase_po) %>%
      summarise(sum((avgNew - next_increase_po) * nb * 0.01)/1e9) %>% as.numeric()

    total_tax_rev <- total_tax_base + total_tax_sur_bill
}

# TO DELETE ALL BELOW?
getPeoplePerBracket=function(grid, brackets){
  brackets = c(brackets, 1e12) ## get last bracket
  grid$group=cut(grid$thresNew, brackets)
```

```
  toReturn = grid %>%
    group_by(group) %>%
    summarise(totalPeople=sum(nb)) %>%
    drop_na()
  return(toReturn)
}

numberTaxpayers <- getPeoplePerBracket(brackets = brackets_po, grid = grid)

#Revenue
#From here on: keep
target_hhlds_mo <- sum( numberTaxpayers$totalPeople[brackets_po>=starting_brack_po] )

# tax_base_total_mo
# tax_rev_init_mo
# final_taxbase_b_in tax_base_total_noav_mo
# num_billionares_in
# top_tax_rev_in
# total_rev_pe
#go with wealthperc and highlight discrepe =
#tax_base_total_mo <- sum(taxBase$taxBase[brackets_po>=starting_brack_po])/1e12* main_tax_po * 1000
#tax_rev_init_mo <-   tax_base_total_mo * main_tax_po * 1000
```

Starting with the $8.9 trillion tax base of wealth above $50 million ($11.4 with no avoidance), a two percent tax would raise $178 billion in 2019. The billionaire surtax is estimated to apply to a base of $2.2 trillion ($2.8 with no avoidance) from about 1000 billionaire families (1300 with no avoidance). Thus the billionaire surtax would raise $22 billion in 2019. The combination of the 2% tax above $50 million and the billionaire surtax would raise 178 + 22 = 200 billion in 2019.

5.1 - Visualization

The figure below illustrates the distribution of wealth tax across the population:

```
# Clean up the code (but do not make changes).
taxRate <- c(0, 2, 2, 3)
brackets <- c(10, 50, 500, 1000)
#this section sorts the tax brackets. Not needed outside the app
if (FALSE){
    ## reshuffle to make sure brackets are increasing
    ## tax rates not forced to be monotonic
    reorderIdx <- order(as.numeric(brackets))
    brackets <- brackets[reorderIdx]
    taxRate <- taxRate[reorderIdx]
}

### KATIE: change the 1e5 to whatever you want to be the minimum
xval <- 10^seq(log10(1e5), log10(45e9), by = 0.001) ## get uniform on log scale

if(FALSE){
    idx0 <- xval <= as.numeric(brackets[1]) * 1e6
    idx1 <- xval <= as.numeric(brackets[2]) * 1e6 & xval > as.numeric(brackets[1]) * 1e6
    idx2 <- xval > as.numeric(brackets[2]) * 1e6 & xval <= as.numeric(brackets[3]) * 1e6
```

```
    idx3 <- xval > as.numeric(brackets[3]) * 1e6 & xval <= as.numeric(brackets[4]) * 1e6
    idx4 <- xval > as.numeric(brackets[4]) * 1e6
    idx <- cbind.data.frame(idx0, idx1, idx2, idx3, idx4)
   # Indicator across income on tax bracket position
    getGroup <- unlist(apply(idx, 1, function(x) {
      which(x)[1]
    }))

    toPlot <- cbind.data.frame(xval, getGroup)
}
    #brackets_po <- c(0, 25, 50, 100, 250, 500, 1000) * 1e6
getGroup <- as.numeric(cut(xval, c(0, brackets * 1e6, 1e12), include.lowest = TRUE))

toPlot <- cbind.data.frame(xval, getGroup)
#summary(toPlot)
#toMatch <- cbind.data.frame(group = 1:7, tax = tax_rates_po)
toMatch <- cbind.data.frame(group = 1:(length(taxRate) + 1), tax = c(0, taxRate))

toPlot2 <- merge(toPlot, toMatch, by.x = "getGroup", by.y = "group")

#lapply(x = 1:5, f(x,y), y = 6:10) = (1, 6:10); (2, 6:10);... ;(5, 6:10)
toPlot2$averageInt <- sapply( toPlot2$xval,
                              function(x) get_tax_rev(wealth_var = x,
                                                      taxrates_var = taxRate/100,
                                                      brackets_var = brackets * 1e6) )

# Here is where the total tax payed by each individuals is transform into average tax rates
toPlot2$averageRate <- (toPlot2$averageInt / toPlot2$xval) * 100

toPlot2$id <- 1:nrow(toPlot2)
#browser()
if(FALSE) {
    # unaffected by new grouping
    toPlot2$marginalInt <- unlist(lapply(toPlot2$xval, getAverageTax, taxRate, brackets))

    toPlot2$marginalRate <- (toPlot2$marginalInt / toPlot2$xval) * 100

    toPlot2$id <- 1:nrow(toPlot2)
}
#summary(toPlot2)
#end of dataInputT

 #
 #      getGroup          xval                tax         marginalInt         marginalRate          id
 # Min.   :1.000   Min.   :1.000e+05   Min.   :0.000   Min.   :0.000e+00   Min.   :0.0000   Min.   :
 # 1st Qu.:1.000   1st Qu.:2.590e+06   1st Qu.:0.000   1st Qu.:0.000e+00   1st Qu.:0.0000   1st Qu.:141
 # Median :3.000   Median :6.707e+07   Median :2.000   Median :3.413e+05   Median :0.5089   Median :282
 # Mean   :2.807   Mean   :3.459e+09   Mean   :1.338   Mean   :9.947e+07   Mean   :1.1090   Mean   :282
 # 3rd Qu.:5.000   3rd Qu.:1.737e+09   3rd Qu.:3.000   3rd Qu.:4.110e+07   3rd Qu.:2.3667   3rd Qu.:424
 # Max.   :5.000   Max.   :4.498e+10   Max.   :3.000   Max.   :1.338e+09   Max.   :2.9755   Max.   :565
 #
```

```r
# These are mini data set that ggvis needs to create vertical lines
extra0 <- cbind.data.frame(x = rep(as.numeric(brackets[1]) * 1e6, 2), y = c(0, taxRate[1]))
extra1 <- cbind.data.frame(x = rep(as.numeric(brackets[2]) * 1e6, 2), y = c(0, taxRate[1]))
extra1b <- cbind.data.frame(x = rep(as.numeric(brackets[2]) * 1e6, 2), y = c(0, taxRate[2]))
extra2 <- cbind.data.frame(x = rep(as.numeric(brackets[3]) * 1e6, 2), y = c(0, taxRate[2]))
extra2b <- cbind.data.frame(x = rep(as.numeric(brackets[3]) * 1e6, 2), y = c(0, taxRate[3]))
extra3 <- cbind.data.frame(x = rep(as.numeric(brackets[4]) * 1e6, 2), y = c(0, taxRate[3]))
extra3b <- cbind.data.frame(x = rep(as.numeric(brackets[4]) * 1e6, 2), y = c(0, taxRate[4]))

showAvg <- function(x) {
  # https://stackoverflow.com/questions/28396900/r-ggvis-html-function-failing-to-add-tooltip/28399656#
  # https://stackoverflow.com/questions/31230124/exclude-line-points-from-showing-info-when-using-add-t
  if (sum(grepl("id", names(x))) == 0) return(NULL)
  if (is.null(x)) return(NULL)

  data <- toPlot2

  row <- data[data$id == x$id, ]

  #The following section does not work in the static plot
  if(FALSE){
  paste0("Average Tax Rate: ", round(row$averageRate, 2), "%",
         " <br> Wealth ($m): ", round(row$xval / 1e6, 0),
         "<br> Top ", getPercentile(updateGrid(), row$xval / 1e6),
         "%", "<br> Taxes Paid ($m): ", round(row$averageInt / 1e6, 2),
         sep = "") ## dividing by 1e6 may need to change if we do this for xval overall
  }
}


data <- toPlot2

rmIdx <- ncol(data)
plot <- data[, -rmIdx] %>%
  ggvis(x = ~ xval / 1e6, y = ~tax) %>%
  layer_points() %>%
  layer_points(data = data, x = ~ xval / 1e6, y = ~averageRate, stroke := "red", key := ~id) %>%
  add_tooltip(showAvg, "hover") %>%
  layer_lines(x = ~ xval / 1e6, y = ~averageRate, stroke := "red") %>%
  layer_paths(data = extra1, ~ x / 1e6, ~y) %>%
  layer_paths(data = extra2, ~ x / 1e6, ~y) %>%
  layer_paths(data = extra3, ~ x / 1e6, ~y) %>%
  layer_paths(data = extra0, ~ x / 1e6, ~y) %>%
  layer_paths(data = extra1b, ~ x / 1e6, ~y) %>%
  layer_paths(data = extra2b, ~ x / 1e6, ~y) %>%
  layer_paths(data = extra3b, ~ x / 1e6, ~y) %>%
  add_axis("x",
           title_offset = 80, title = "Wealth ($m)", grid = F, format = ",",
           values = brackets, properties = axis_props(labels = list(angle = 45, align = "left", baselin
  ) %>%
  add_axis("y", title = "Tax rate (%)") %>%
  scale_numeric("x", trans = "log", expand = 0) %>%
  set_options(width = 1000, height = 500)
```

```
plot
```

Renderer: SVG | Canvas

Download

Click here/ADD URL WHEN ALLOWED to explore different policy proposal and to see how the assumptions of the analysis affect the results.

There are two ways to edit the code behind this document:
1 - Download/Clone this repository into your computer. You will need to install R and RStudio.

2 - Go to this link and reproduce all the result in a computing enviroment (supported by project binder). You will **not** need to install anything in your computer.

6 - Ten year projections

```
# - inputs: inflation_so, population_gr_so, real_growth_so, total_rev_pe, top_tax_rev_in
# - ouputs: discount_rate_mo, ten_year_factor_mo, ten_year_revenue_pe, ten_year_top_tax_pe
ten_years_mo_f <- function(inflation_var = inflation_so, population_gr_var = population_gr_so,
                           real_growth_var = real_growth_so, total_rev_var = total_tax_rev,
                           top_tax_base_var = top_tax_rev_in){



    discount_rate_mo <- inflation_var + population_gr_so + real_growth_so
    ten_year_factor_mo <- sum( ( 1 + discount_rate_mo )^( 0:9 ) )

    ten_year_revenue_pe <- total_rev_var * ten_year_factor_mo
    #ten_year_revenue_pe <- total_rev_pe * ten_year_factor_mo                    #PE
    ten_year_top_tax_pe <- top_tax_base_var * ten_year_factor_mo
    #ten_year_top_tax_pe <- top_tax_base_var * ten_year_factor_mo                #PE



    return( list("discount_rate_mo" = discount_rate_mo, "ten_year_factor_mo" = ten_year_factor_mo,
           "ten_year_revenue_pe" = ten_year_revenue_pe, "ten_year_top_tax_pe" = ten_year_top_tax_pe) )
}

invisible( list2env(ten_years_mo_f(),.GlobalEnv) )

#TO DELETE
ten_year_factor_round <- round(ten_year_factor_mo)



# test to run from the beginning (only functions)
if (FALSE) {
    rm(list = ls()[!(ls() %in% ls(pattern = "_f\\b"))])
    invisible( list2env(call_params_f(), .GlobalEnv) )
    invisible( list2env(policy_f(), .GlobalEnv) )
    invisible( list2env(tax_elasticity_in_f(), .GlobalEnv) )
    invisible( list2env(est_billionares_in_f(), .GlobalEnv) )
```

```
    invisible( list2env(tax_revenue_mo_f(), .GlobalEnv) )
    invisible( list2env(total_rev_mo_f(), .GlobalEnv) )
    invisible( list2env(ten_years_mo_f(),.GlobalEnv) )
    sapply(ls(pattern = "_pe\\b"), get)
}
```

To project tax revenues over a 10-year horizon, we assume that nominal taxable wealth would grow at the same pace as the economy, at 5.5% per year as in standard projections of the Congressional Budget Office or the Joint Committee on Taxation. This growth is decomposed into 2.5% price, 1% population growth, and 2% of real growth per capita. This implies that tax revenue over the 10 years 2019-2028 is about 13 times the revenue raised in 2019[1]. This uniform growth assumption is conservative as the wealth of the rich has grown substantially faster than average in recent decades. The estimates by Saez and Zucman[2] show that, from 1980 to 2016, real wealth of the top 0.1% has grown at 5.3% per year on average, which is 2.8 points above the average real wealth growth of 2.5% per year. Average real wealth of the Forbes 400 has grown even faster at 7% per year, 4.5 points above the average. The historical gap in growth rates of top wealth vs. average wealth is larger than the proposed wealth tax. Therefore, even with the wealth tax, it is most likely that top wealth would continue to grow at least as fast as the average.

This 10-year projection implies that revenue raised by the progressive wealth tax would be 12.9 * 199.8 = \$2572 billion, rounded to \$2.6 trillion. Out of these \$2.6 trillion, the billionaire surtax would raise 21.7 * 12.9 = \$278.8 billion, rounded to \$0.3 trillion.

It is important to emphasize that our computations assume that the wealth tax base is comprehensive with no major asset classes exempt from wealth taxation. Introducing exemptions for specific asset classes would reduce the revenue estimates both mechanically and dynamically as wealthy individuals would shift their wealth into tax exempt assets. Because your proposal does not include any large exemptions, we do not believe our revenue estimate needs to be adjusted.

7 - Wealth inequality

One of the key motivations for introducing a progressive wealth tax is to curb the growing concentration of wealth. The top 0.1% wealth share has increased dramatically from about 7% in the late 1970s to around 20% in recent years. Conversely, the wealth share of the bottom 90% of families has declined from about 35% in the late 1970s to about 25% today. This fall has been primarily the consequence of increased debt for the bottom 90% (through mortgage refinance, consumer credit, and student loans). As a result, the top 0.1% today owns almost as much wealth as the bottom 90% of US families, which includes the vast majority of US families.

8 - Tax burden on the wealthiest 0.1%

The estimates of Piketty, Saez, and Zucman (2018) show that the total burden (including all taxes both at the federal, state, and local levels) of the wealthiest 0.1% families is projected to be 3.2% of their wealth in 2019 (they have on average \$116 million in wealth, and pay total taxes of \$3.68 million). The proposed progressive wealth tax would add an extra \$1.27 million (or 1.1% of wealth) to their tax burden for a total tax burden (relative to wealth) of 4.3%.

In contrast, the bottom 99% families have a total tax burden of 7.2% relative to their wealth. Their tax burden relative to wealth is much higher than for the top 0.1% because the bottom 99% relies primarily on labor income, which bears tax but is not part of net worth. In contrast, the majority of the income of the top 0.1% wealthiest comes from returns to their wealth.

[1] With r=5.5%, we have [1+(1+r)+..+(1+r)^9]=[(1+r)^10-1]/r=12.9, approximately 13.

[2] Saez, Emmanuel and Gabriel Zucman, "Wealth Inequality in the United States since 1913: Evidence from Capitalized Income Tax Data", Quarterly Journal of Economics 131(2), 2016, 519-578, updated series available at http://gabriel-zucman.eu/usdina/

Note: Our analysis complies with the highest levels of transparency and reproducibilty for open policy analysis proposed by the *Berkeley Initiative for Transparency in the Social Sciences*. We invite contributors and critics of this analysis to follow similar standards.