debt_fedrate.r

Alexandros

2023-01-01

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.2
```

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

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## 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 4.1.3
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.1.2
```

```
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.1.3
```

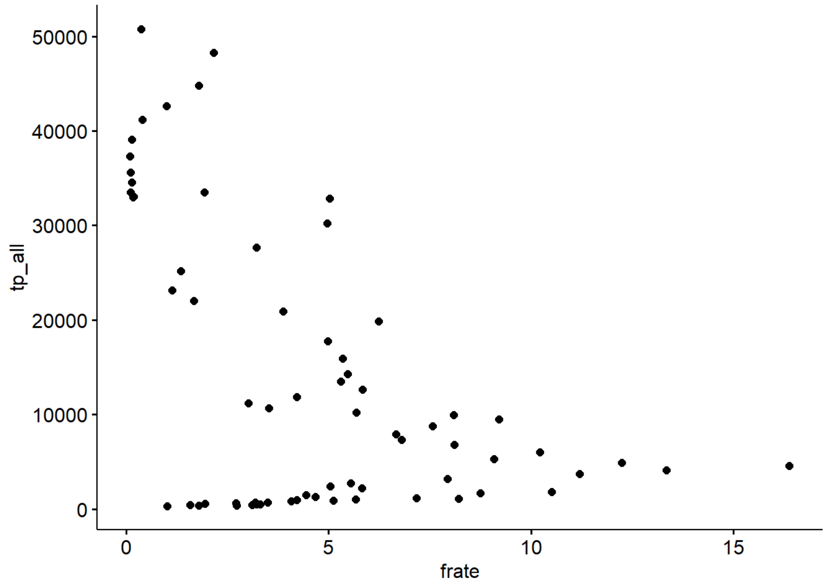
```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.1.2
```

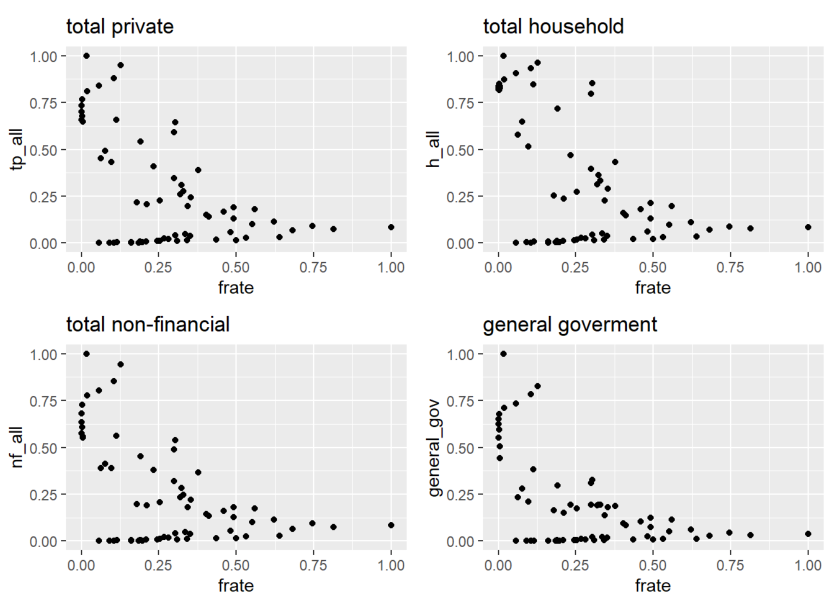
```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

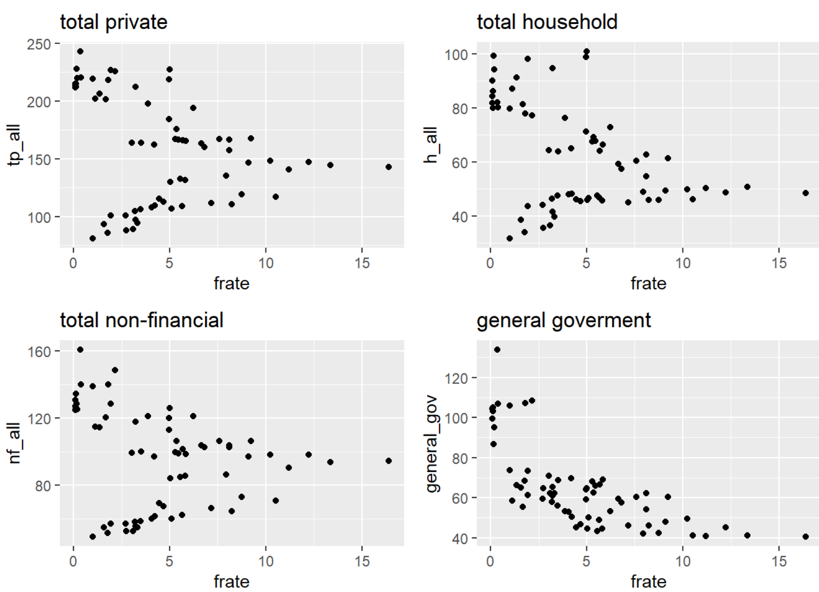
```
df=read_excel("fed_rate_debt.xlsx",sheet="nominal")



ggscatter(data=df,x="frate",y="tp_all")
```



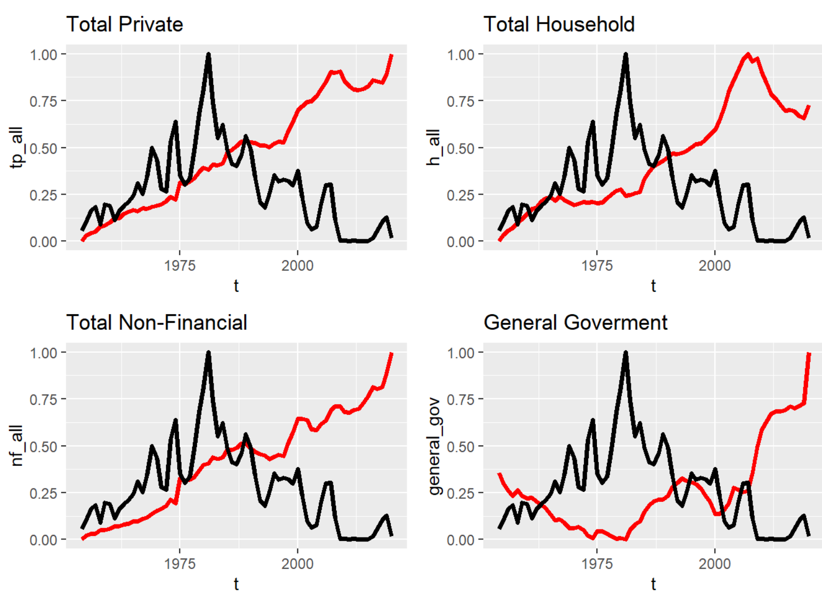

```
# min max normalization in order to better visualize the kind of relationship between different variables

normalize <- function(x, na.rm = TRUE) {
  return((x- min(x)) /(max(x)-min(x)))
}

#tibble with normalized variables

dfnorm=df%>% select_if(is.numeric) %>% transmute_all(normalize)


## Scatter plots between the federal rate and the various secotr's debt

ptp=ggplot(data=dfnorm) +
  geom_point(aes(x=frate,y=tp_all))+
  ggtitle("total private")

ph=ggplot(data=dfnorm) +
  geom_point(aes(x=frate,y=h_all))+
  ggtitle("total household")

pnf=ggplot(data=dfnorm) +
  geom_point(aes(x=frate,y=nf_all))+
  ggtitle("total non-financial")

pg=ggplot(data=dfnorm) +
  geom_point(aes(x=frate,y=general_gov))+
  ggtitle("general goverment")


grid.arrange(ptp,ph,pnf,pg)
```



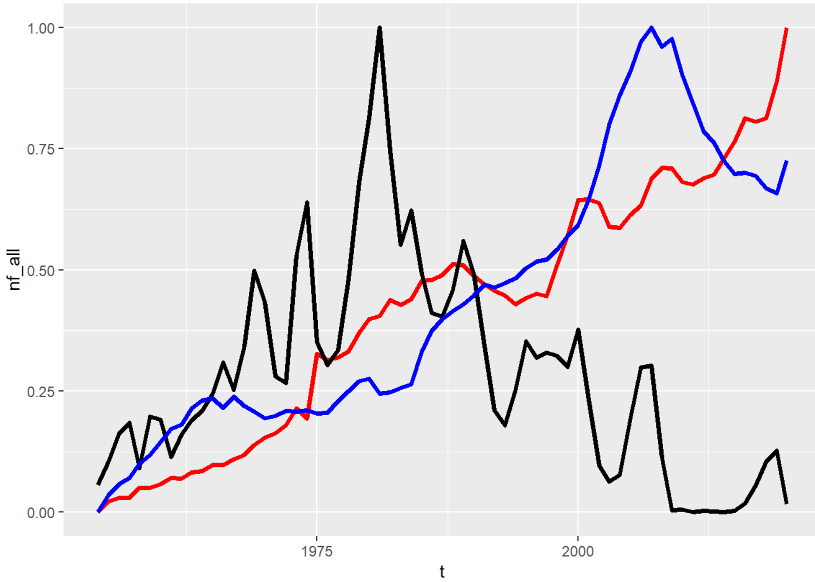
```
# from the above plots, the relationship between the federal rate and the level of debt of the various household
#sectors, seems to be negative and non-linear (convex)

# The level of debt is not a good indicator of sustainability, for this reason scatterplots between the various
#sectors shares of debt to gdp and the federal rate are produced.


dfper=read_excel("fed_rate_debt.xlsx",sheet="percent")



ptp=ggplot(data=dfper) +
  geom_point(aes(x=frate,y=tp_all))+
  ggtitle("total private")

ph=ggplot(data=dfper) +
  geom_point(aes(x=frate,y=h_all))+
  ggtitle("total household")

pnf=ggplot(data=dfper) +
  geom_point(aes(x=frate,y=nf_all))+
  ggtitle("total non-financial")

pg=ggplot(data=dfper) +
  geom_point(aes(x=frate,y=general_gov))+
  ggtitle("general goverment")


grid.arrange(ptp,ph,pnf,pg)
```



```
# The above plots show a highly non linear relationship, still negative and non-linear. However there is an
#indication of a possible positive relationship (except for the general goverment's debt). Thus it is very likely
#that there is a confounding variable that distorts the association between the federal rate and the percentage of
#debt to gdp.


## We will produce a time series plot the various sector debt percentage to gdp ( normalized) and the federal rate
#in order to visualize their intertemporal relationship/



dfnorm=dfper%>% select_if(is.numeric) %>% transmute_all(normalize)
t=c(1954:2020)

ptp <- dfnorm %>%
  ggplot()+
  geom_line(aes(x=t,y=tp_all),color="red",size=1.3)+
  geom_line(aes(x=t,y=frate),color="black",size=1.3)+
  ggtitle("Total Private")


ph <- dfnorm %>%
  ggplot()+
  geom_line(aes(x=t,y=h_all),color="red",size=1.3)+
  geom_line(aes(x=t,y=frate),color="black",size=1.3)+
  ggtitle("Total Household")



pnf <-  dfnorm %>%
  ggplot()+
  geom_line(aes(x=t,y=nf_all),color="red",size=1.3)+
  geom_line(aes(x=t,y=frate),color="black",size=1.3)+
  ggtitle("Total Non-Financial")


pgov <- dfnorm %>%
  ggplot()+
  geom_line(aes(x=t,y=general_gov),color="red",size=1.3)+
  geom_line(aes(x=t,y=frate),color="black",size=1.3)+
  ggtitle("General Goverment")

grid.arrange(ptp,ph,pnf,pgov)
```




```
dfnorm %>%
  ggplot()+
  geom_line(aes(x=t,y=nf_all),color="red",size=1.3)+
  geom_line(aes(x=t,y=frate),color="black",size=1.3)+
  geom_line(aes(x=t,y=h_all),color="blue",size=1.3)
```




```
#The federal rate reached its maximum value at 1981. Before that year, there was a slight increace in both household
#and non-financial corporations debt to gdp ratio, which accelerated in the period 1981-2020, a period where the
#federal rate shows a decreasing trend

which(dfnorm$frate==1)
```

```
## [1] 28
```

```
cbind(t,dfnorm)
```

```
##       t        frate     tp_all      tp_ld      h_all       h_ld     nf_all
## 1  1954 0.0564448702 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 2  1955 0.1043148778 0.03012383 0.02619475 0.03534233 0.03637158 0.02177145
## 3  1956 0.1620517372 0.04563044 0.05219200 0.05840542 0.05950403 0.02995464
## 4  1957 0.1851070482 0.05089428 0.06909844 0.07115775 0.07220810 0.02967553
## 5  1958 0.0910464772 0.07762389 0.10556406 0.10042978 0.10157498 0.05029037
## 6  1959 0.1976694146 0.08524891 0.11692890 0.11885440 0.11972931 0.04991585
## 7  1960 0.1916358900 0.10158717 0.14493066 0.14479196 0.14602392 0.05752116
## 8  1961 0.1141976761 0.12250230 0.17471014 0.17210842 0.17368241 0.07091225
## 9  1962 0.1608682204 0.12533148 0.18416009 0.18153011 0.18359047 0.06916798
## 10 1963 0.1896105381 0.14815413 0.21533198 0.21434147 0.21684617 0.08191545
## 11 1964 0.2090826106 0.15745219 0.23069188 0.23053867 0.23346179 0.08535186
## 12 1965 0.2447114673 0.16830811 0.24190714 0.23502155 0.23774185 0.09832253
## 13 1966 0.3083755833 0.16003119 0.23547457 0.21600714 0.21811867 0.09811721
## 14 1967 0.2532813782 0.17737792 0.26486563 0.23880928 0.24072242 0.10913281
## 15 1968 0.3419707290 0.17498940 0.25922815 0.21898641 0.22016448 0.11797471
## 16 1969 0.4985812074 0.18444418 0.26546579 0.20795296 0.20839579 0.13854620
## 17 1970 0.4345607122 0.18861416 0.27850279 0.19359544 0.19295056 0.15351234
## 18 1971 0.2809848169 0.19737840 0.28822641 0.19907816 0.19832537 0.16282674
## 19 1972 0.2668225885 0.21347721 0.30645163 0.20975148 0.20916922 0.17956220
## 20 1973 0.5309764117 0.23612459 0.31817574 0.20776885 0.20767988 0.21365909
## 21 1974 0.6395258140 0.22297778 0.33903457 0.20995368 0.20935601 0.19322388
## 22 1975 0.3517445770 0.31305566 0.31687528 0.20463080 0.20422396 0.32724982
## 23 1976 0.3041250753 0.30405755 0.30387224 0.20554318 0.20620494 0.31362527
## 24 1977 0.3346333329 0.31848725 0.32566667 0.22925621 0.23121848 0.31984210
## 25 1978 0.4815590581 0.33636639 0.33899469 0.25063419 0.25369897 0.33251460
## 26 1979 0.6819483855 0.37047057 0.35917042 0.27007266 0.27398470 0.36993712
## 27 1980 0.8136792912 0.39250732 0.36679132 0.27514274 0.27904177 0.39876879
## 28 1981 1.0000000000 0.38377921 0.34785710 0.24461232 0.24789553 0.40505906
## 29 1982 0.7454460792 0.40772426 0.37182082 0.24745608 0.25065667 0.43804238
## 30 1983 0.5523289044 0.40437243 0.38608507 0.25670539 0.26003404 0.42743526
## 31 1984 0.6219581554 0.41626051 0.41547060 0.26419499 0.26833595 0.44003692
## 32 1985 0.4915477615 0.47167970 0.47714985 0.33226024 0.33813630 0.47819914
## 33 1986 0.4117709356 0.49017463 0.53573292 0.37380055 0.38124770 0.47924638
## 34 1987 0.4032346454 0.50794991 0.56037983 0.39889807 0.39762901 0.48945867
## 35 1988 0.4591589782 0.53087648 0.58124487 0.41550245 0.41521275 0.51241994
## 36 1989 0.5599759276 0.53559126 0.58417116 0.42982256 0.42644835 0.51037060
## 37 1990 0.4913544426 0.52789082 0.58247851 0.44776366 0.44299148 0.48805600
## 38 1991 0.3433898385 0.52491403 0.56384825 0.46999228 0.46454289 0.46993423
## 39 1992 0.2106148776 0.51348582 0.52734857 0.46402897 0.45856378 0.45705227
## 40 1993 0.1799529983 0.51053681 0.52351014 0.47291122 0.46678918 0.44725765
## 41 1994 0.2526610767 0.50263287 0.52703705 0.48338247 0.47760212 0.42928579
## 42 1995 0.3525497922 0.52019773 0.54602148 0.50305646 0.49625019 0.44256023
## 43 1996 0.3197947685 0.53265178 0.55298205 0.51818842 0.51102113 0.45123803
## 44 1997 0.3296759639 0.52989616 0.56564482 0.52108180 0.51374942 0.44544256
## 45 1998 0.3228913111 0.58365731 0.61040147 0.54220371 0.53609763 0.51034599
## 46 1999 0.2996627846 0.63783398 0.65366794 0.57006808 0.56459691 0.57166601
## 47 2000 0.3772849568 0.69816662 0.68307768 0.59331150 0.58888058 0.64478872
## 48 2001 0.2325374198 0.72155030 0.72568095 0.64591178 0.64329502 0.64606332
## 49 2002 0.0968410011 0.74554490 0.76286474 0.71584958 0.71202737 0.63745954
## 50 2003 0.0636893315 0.74819757 0.79766668 0.79994053 0.79807287 0.58909660
## 51 2004 0.0774233234 0.77221704 0.83347260 0.86062221 0.85903102 0.58627606
## 52 2005 0.1919623045 0.81154328 0.87308644 0.90917396 0.90950444 0.61320024
## 53 2006 0.2992374830 0.85138522 0.93502306 0.97040862 0.97049469 0.63299793
## 54 2007 0.3024213612 0.90189401 1.00000000 1.00000000 1.00000000 0.68792280
## 55 2008 0.1126336119 0.90065517 0.99654167 0.96069370 0.95770745 0.71053048
## 56 2009 0.0043017658 0.90668794 0.98434958 0.97675770 0.96993986 0.70931102
## 57 2010 0.0053843516 0.85521855 0.90424548 0.90159240 0.89731859 0.68128920
## 58 2011 0.0007984911 0.82702215 0.85842270 0.84346140 0.83726640 0.67646442
## 59 2012 0.0031955490 0.81153800 0.82781034 0.78593216 0.77919206 0.68971404
## 60 2013 0.0011565513 0.80628864 0.81949477 0.76218579 0.75553768 0.69684042
## 61 2014 0.0000000000 0.81303056 0.81058182 0.72517426 0.71690757 0.72960493
## 62 2015 0.0027753870 0.82558209 0.81084060 0.69802106 0.68826095 0.76467925
## 63 2016 0.0186959217 0.86031568 0.82806207 0.70011986 0.69099687 0.81378147
## 64 2017 0.0560125127 0.85190964 0.84356249 0.69455152 0.68575474 0.80504004
## 65 2018 0.1043758195 0.84632867 0.83601311 0.66850328 0.66163535 0.81311442
## 66 2019 0.1272912023 0.89441983 0.83501430 0.65822184 0.65241237 0.88928807
## 67 2020 0.0168215348 1.00000000 0.96384965 0.72667985 0.72229208 1.00000000
##          nf_ld general_gov Central_gov         gdp
## 1  0.000000000 0.354730315  0.40923767 0.000000000
## 2  0.005861529 0.300528809  0.34983509 0.001648610
## 3  0.028185911 0.260192819  0.30447564 0.002769554
## 4  0.045682296 0.234569987  0.27036808 0.003880914
## 5  0.080918020 0.262345073  0.28578237 0.004188774
## 6  0.080438277 0.234637763  0.25563231 0.006157535
## 7  0.102728792 0.218568604  0.23313805 0.007130916
## 8  0.126823363 0.224560351  0.23169101 0.008021353
## 9  0.133029095 0.202944968  0.20953612 0.010016907
## 10 0.152770413 0.187797190  0.19224188 0.011585539
## 11 0.162206442 0.166776126  0.17021498 0.013815289
## 12 0.179093635 0.135466858  0.13959486 0.016534170
## 13 0.191266779 0.104290435  0.11023244 0.019910510
## 14 0.221023501 0.106976949  0.10856191 0.022095129
## 15 0.235970320 0.089239432  0.09070316 0.025870336
```

```
## 16 0.263377387 0.059332133  0.05953591 0.029475995
## 17 0.309007238 0.062421616  0.05548786 0.032126568
## 18 0.321529803 0.066810583  0.05392545 0.036465312
## 19 0.344021164 0.050067650  0.03888024 0.041921370
## 20 0.369267450 0.019960041  0.01159043 0.049002170
## 21 0.408676394 0.008236997  0.00000000 0.054635096
## 22 0.371077538 0.043079077  0.03904272 0.061181634
## 23 0.342658119 0.044994974  0.05141844 0.070390944
## 24 0.354217025 0.031120605  0.05042714 0.080513866
## 25 0.352139096 0.018137351  0.04035753 0.093417877
## 26 0.366493636 0.002968651  0.02493847 0.106615854
## 27 0.375232923 0.009388466  0.03367671 0.117577462
## 28 0.377178698 0.000000000  0.03076641 0.134243872
## 29 0.421389995 0.049421354  0.07614986 0.140762454
## 30 0.437859693 0.080509193  0.10798296 0.154594335
## 31 0.485814647 0.098135435  0.12779909 0.173830766
## 32 0.519786251 0.147709910  0.16815797 0.188192855
## 33 0.581572047 0.184686079  0.20619191 0.199660841
## 34 0.609802649 0.203954774  0.21916092 0.212796899
## 35 0.628970260 0.213279145  0.22344921 0.230963381
## 36 0.620493048 0.214537844  0.22388272 0.250273683
## 37 0.596059052 0.232786366  0.24664978 0.265597295
## 38 0.531515067 0.279647618  0.28462284 0.274890833
## 39 0.466433964 0.303850896  0.30914688 0.292152982
## 40 0.448316576 0.325100266  0.32654494 0.308272503
## 41 0.441573736 0.312084791  0.31898930 0.328703987
## 42 0.455640752 0.307068203  0.31522394 0.345503842
## 43 0.450697098 0.297704279  0.30832556 0.366158133
## 44 0.472442933 0.272098680  0.27961381 0.390198607
## 45 0.533127535 0.237503462  0.24651875 0.413326408
## 46 0.583012661 0.199305950  0.21137054 0.440413494
## 47 0.610667694 0.136629860  0.15161479 0.469951453
## 48 0.626236536 0.136379994  0.14373713 0.485725445
## 49 0.612782096 0.161948161  0.15970550 0.502271515
## 50 0.572540626 0.194954915  0.18760372 0.527404576
## 51 0.566240172 0.274922968  0.19633934 0.563660093
## 52 0.580872944 0.268012275  0.19783279 0.602837118
## 53 0.626574219 0.254498274  0.18924953 0.639839694
## 54 0.718417625 0.258503319  0.19297595 0.671230402
## 55 0.765377202 0.353226780  0.28301030 0.685318451
## 56 0.725520687 0.493864545  0.40940021 0.671411507
## 57 0.658445274 0.585394764  0.50423054 0.698621316
## 58 0.643642400 0.631833729  0.55757537 0.724869600
## 59 0.656614802 0.669453358  0.60416722 0.756049417
## 60 0.670170269 0.685382508  0.62981080 0.784133772
## 61 0.701603228 0.684940346  0.63838465 0.817850248
## 62 0.738591770 0.689474951  0.64918726 0.849083681
## 63 0.769407113 0.711401297  0.67592371 0.872392587
## 64 0.806952514 0.701312131  0.67180541 0.909782393
## 65 0.822626001 0.712806624  0.69376440 0.959706559
## 66 0.832379534 0.727822240  0.71708490 1.000000000
## 67 1.000000000 1.000000000  1.00000000 0.977178409
```

```
trend=1:67

## Simple Regressions



model_total=lm(data=df,log(tp_all)~log(frate))

model_total %>% summary
```

```
##
## Call:
## lm(formula = log(tp_all) ~ log(frate), data = df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.391 -1.112  0.454  1.103  2.047
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.1479     0.2306  39.661  < 2e-16 ***
## log(frate)   -0.4927     0.1386  -3.555 0.000711 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.473 on 65 degrees of freedom
## Multiple R-squared:  0.1628, Adjusted R-squared:  0.1499
## F-statistic: 12.64 on 1 and 65 DF,  p-value: 0.0007114
```

```
model_house=lm(data=df,log(h_all)~log(frate))

model_house %>% summary
```

```
##
## Call:
## lm(formula = log(h_all) ~ log(frate), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4122 -1.0781  0.4082  1.0544  2.1803
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.2294     0.2264  36.350  < 2e-16 ***
## log(frate)   -0.5084     0.1360  -3.737 0.000395 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.446 on 65 degrees of freedom
## Multiple R-squared:  0.1769, Adjusted R-squared:  0.1642
## F-statistic: 13.97 on 1 and 65 DF,  p-value: 0.0003947
```

```
model_nf=lm(data=df,log(nf_all)~log(frate))

model_nf %>% summary
```

```
##
## Call:
## lm(formula = log(nf_all) ~ log(frate), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3739 -1.1369  0.5097  1.1010  2.1035
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.6351     0.2337  36.950  < 2e-16 ***
## log(frate)   -0.4829     0.1404  -3.439  0.00102 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.493 on 65 degrees of freedom
## Multiple R-squared:  0.1539, Adjusted R-squared:  0.1409
## F-statistic: 11.83 on 1 and 65 DF,  p-value: 0.001024
```

```
model_gov=lm(data=df,log(general_gov)~log(frate))

model_gov %>% summary
```

```
##
## Call:
## lm(formula = log(general_gov) ~ log(frate), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6888 -1.0782  0.3392  1.0374  2.1440
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.3536     0.2005  41.663  < 2e-16 ***
## log(frate)   -0.5787     0.1205  -4.803 9.56e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.281 on 65 degrees of freedom
## Multiple R-squared:  0.262,  Adjusted R-squared:  0.2506
## F-statistic: 23.07 on 1 and 65 DF,  p-value: 9.558e-06
```

```
model_total$coefficients[2]
```

```
## log(frate)
##  -0.492665
```

```
model_house$coefficients[2]
```

```
## log(frate)
## -0.5084442
```

```
model_nf$coefficients[2]
```

```
## log(frate)
## -0.4829396
```

```
model_gov$coefficients[2]
```

```
## log(frate)
## -0.5787272
```

```
##correlation and cross correlation


df %>% select_if(is.numeric) %>% cor
```

```
##                  frate     tp_all      tp_ld      h_all       h_ld     nf_all
## frate        1.0000000 -0.5273538 -0.5269532 -0.5397819 -0.5388292 -0.5149127
## tp_all      -0.5273538  1.0000000  0.9990802  0.9915550  0.9913194  0.9966366
## tp_ld       -0.5269532  0.9990802  1.0000000  0.9957803  0.9956245  0.9924736
## h_all       -0.5397819  0.9915550  0.9957803  1.0000000  0.9999937  0.9775923
## h_ld        -0.5388292  0.9913194  0.9956245  0.9999937  1.0000000  0.9772140
## nf_all      -0.5149127  0.9966366  0.9924736  0.9775923  0.9772140  1.0000000
## nf_ld       -0.5072416  0.9970707  0.9940263  0.9798219  0.9794783  0.9992958
## general_gov -0.5463644  0.9690883  0.9603308  0.9376937  0.9368863  0.9804955
## Central_gov -0.5365957  0.9522045  0.9408392  0.9123998  0.9114742  0.9690728
## gdp         -0.4971859  0.9949478  0.9949730  0.9877537  0.9874689  0.9908379
##                  nf_ld general_gov Central_gov        gdp
## frate       -0.5072416  -0.5463644  -0.5365957 -0.4971859
## tp_all       0.9970707   0.9690883   0.9522045  0.9949478
## tp_ld        0.9940263   0.9603308   0.9408392  0.9949730
## h_all        0.9798219   0.9376937   0.9123998  0.9877537
## h_ld         0.9794783   0.9368863   0.9114742  0.9874689
## nf_all       0.9992958   0.9804955   0.9690728  0.9908379
## nf_ld        1.0000000   0.9770688   0.9647085  0.9927093
## general_gov  0.9770688   1.0000000   0.9974560  0.9580405
## Central_gov  0.9647085   0.9974560   1.0000000  0.9399979
## gdp          0.9927093   0.9580405   0.9399979  1.0000000
```

```
### - means x causes y, lead/+ means y causes x

ccf(x=df$frate, df$tp_all, lag.max = 5,plot=F)
```

```
##
## Autocorrelations of series 'X', by lag
##
##     -5     -4     -3     -2     -1      0      1      2      3      4      5
## -0.410 -0.447 -0.472 -0.486 -0.496 -0.527 -0.530 -0.529 -0.525 -0.516 -0.511
```

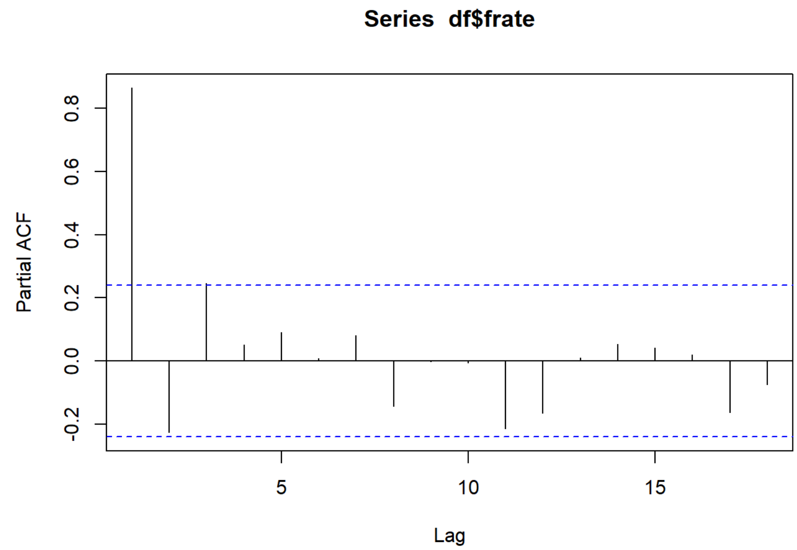
```
#in differences
ccf(y=diff(df$tp_all), x=diff(df$frate), lag.max = 5,plot=F)
```

```
##
## Autocorrelations of series 'X', by lag
##
##     -5     -4     -3     -2     -1      0      1      2      3      4      5
## -0.132 -0.159 -0.145 -0.014  0.112  0.069 -0.086 -0.156 -0.173 -0.154 -0.057
```

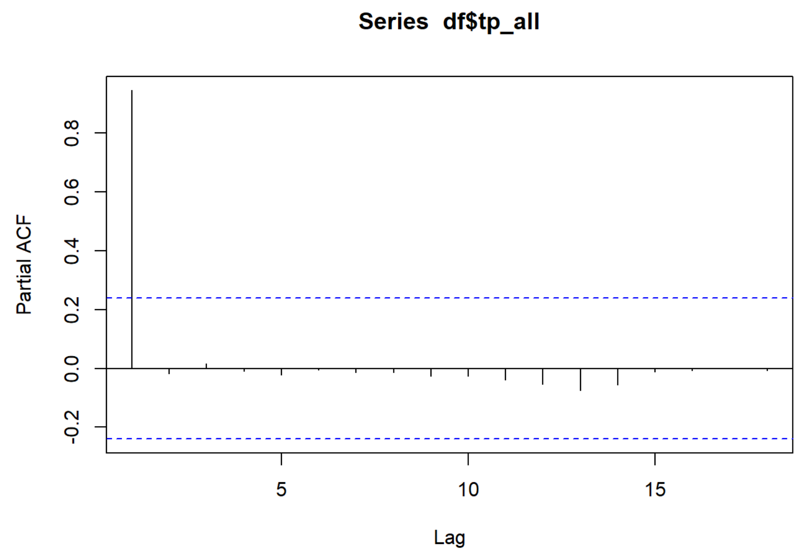
```
####

ccf(x=df$frate, df$tp_all, lag.max = 5,plot=F)
```

```
##
## Autocorrelations of series 'X', by lag
##
##     -5     -4     -3     -2     -1      0      1      2      3      4      5
## -0.410 -0.447 -0.472 -0.486 -0.496 -0.527 -0.530 -0.529 -0.525 -0.516 -0.511
```

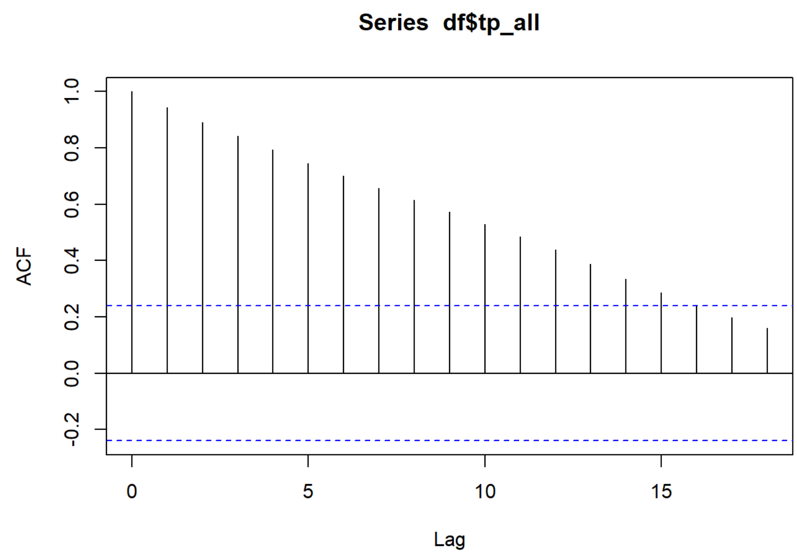
```
pacf(df$frate)
```




```
pacf(df$tp_all)
```




```
acf(df$tp_all)
```



```
# FORECASTING ---------------------------------------------------------------




df$Class="Train"



df$Class %>% length
```

```
## [1] 67
```

```
df$Class[63:67]="Test"


df %>% tail(10)
```

```
## # A tibble: 10 x 12
##    year   frate tp_all  tp_ld  h_all   h_ld nf_all  nf_ld gener~1 Centr~2    gdp
##    <chr>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>   <dbl>   <dbl>  <dbl>
##  1 2011  0.102  33524. 23905. 14054. 13633. 19470. 10272.  15519.  12250. 15600.
##  2 2012  0.141  34522. 24382. 13994. 13568. 20528. 10814.  16742.  13457. 16254.
##  3 2013  0.107  35630. 25118. 14224. 13791. 21406. 11327.  17600.  14340. 16843.
##  4 2014  0.0885 37319. 26008. 14371. 13912. 22948. 12096.  18332.  15080. 17551.
##  5 2015  0.134  39083. 26984. 14564. 14080. 24519. 12904.  19094.  15823. 18206.
##  6 2016  0.393  41186. 28049. 14983. 14492. 26204. 13557.  19990.  16705. 18695.
##  7 2017  1.00   42649. 29545. 15536. 15032. 27113. 14513.  20645.  17333. 19480.
##  8 2018  1.79   44757. 30970. 16000. 15506. 28756. 15464.  21976.  18677. 20527.
##  9 2019  2.16   48268. 32223. 16507. 16011. 31761. 16212.  23181.  19903. 21373.
## 10 2020  0.363  50765. 34344. 17130. 16638. 33635. 17706.  27981.  24865. 20894.
## # ... with 1 more variable: Class <chr>, and abbreviated variable names
## #   1: general_gov, 2: Central_gov
```

```
dat=df
dat %>% tail
```

```
## # A tibble: 6 x 12
##   year  frate tp_all  tp_ld  h_all   h_ld nf_all  nf_ld general~1 Centr~2    gdp
##   <chr> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>     <dbl>   <dbl>  <dbl>
## 1 2015  0.134 39083. 26984. 14564. 14080. 24519. 12904.    19094.  15823. 18206.
## 2 2016  0.393 41186. 28049. 14983. 14492. 26204. 13557.    19990.  16705. 18695.
## 3 2017  1.00  42649. 29545. 15536. 15032. 27113. 14513.    20645.  17333. 19480.
## 4 2018  1.79  44757. 30970. 16000. 15506. 28756. 15464.    21976.  18677. 20527.
## 5 2019  2.16  48268. 32223. 16507. 16011. 31761. 16212.    23181.  19903. 21373.
## 6 2020  0.363 50765. 34344. 17130. 16638. 33635. 17706.    27981.  24865. 20894.
## # ... with 1 more variable: Class <chr>, and abbreviated variable names
## #   1: general_gov, 2: Central_gov
```

```
dat_train = subset(dat, Class == 'Train')
dat_test = subset(dat, Class == 'Test')

nrow(dat_train); nrow(dat_test)
```

```
## [1] 62
```

```
## [1] 5
```

```
### CONVERT TO TS for forecast package
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.1.1
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:ggpubr':
##
##     gghistogram
```

```
dat_train %>% tail
```

```
## # A tibble: 6 x 12
##   year   frate tp_all  tp_ld  h_all   h_ld nf_all  nf_ld genera~1 Centr~2    gdp
##   <chr>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>    <dbl>   <dbl>  <dbl>
## 1 2010  0.176  33029. 23789. 14165. 13762. 18864. 10027.   14318.  11083. 15049.
## 2 2011  0.102  33524. 23905. 14054. 13633. 19470. 10272.   15519.  12250. 15600.
## 3 2012  0.141  34522. 24382. 13994. 13568. 20528. 10814.   16742.  13457. 16254.
## 4 2013  0.107  35630. 25118. 14224. 13791. 21406. 11327.   17600.  14340. 16843.
## 5 2014  0.0885 37319. 26008. 14371. 13912. 22948. 12096.   18332.  15080. 17551.
## 6 2015  0.134  39083. 26984. 14564. 14080. 24519. 12904.   19094.  15823. 18206.
## # ... with 1 more variable: Class <chr>, and abbreviated variable names
## #   1: general_gov, 2: Central_gov
```

```
dat_ts <- ts(dat_train[, 3], start = c(1954, 1), end = c(2015, 1), frequency = 1)

##Mean absolute error percentage

#lines 2 to 4
mape <- function(actual,pred){
  mape <- mean(abs((actual - pred)/actual))*100
  return (mape)
}


#NAIVE FORECASTING

naive_mod <- naive(dat_ts, h = 5)
summary(naive_mod)
```

```
##
## Forecast method: Naive method
##
## Model Information:
## Call: naive(y = dat_ts, h = 5)
##
## Residual sd: 947.7004
##
## Error measures:
##                    ME     RMSE      MAE      MPE     MAPE MASE      ACF1
## Training set 635.5375 947.7004 652.5945 7.531884 7.583594    1 0.8045716
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2016       39082.82 37868.29 40297.35 37225.36 40940.28
## 2017       39082.82 37365.22 40800.42 36455.98 41709.66
## 2018       39082.82 36979.20 41186.44 35865.61 42300.03
## 2019       39082.82 36653.77 41511.87 35367.90 42797.74
## 2020       39082.82 36367.06 41798.58 34929.42 43236.22
```

```
#add naive forecast to test

dat_test$naive=722.773

mape(dat_test$tp_all, dat_test$naive) #98&
```

```
## [1] 98.40287
```

```
####Simple Exponential Smoothing
se_model <- ses(dat_ts, h = 5)
summary(se_model)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = dat_ts, h = 5)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 314.6853
##
##   sigma:  955.6492
##
##      AIC     AICc      BIC
## 1110.786 1111.200 1117.167
##
## Error measures:
##                    ME     RMSE      MAE      MPE    MAPE      MASE     ACF1
## Training set 625.3522 940.1092 642.1319 7.412839 7.46371 0.9839678 0.805956
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2016       39082.64 37857.93 40307.36 37209.61 40955.68
## 2017       39082.64 37350.72 40814.56 36433.90 41731.39
## 2018       39082.64 36961.52 41203.77 35838.66 42326.62
## 2019       39082.64 36633.40 41531.89 35336.85 42828.44
## 2020       39082.64 36344.32 41820.97 34894.74 43270.55
```

```
df_fc = as.data.frame(se_model)  ##save output in dataframe

dat_test$simplexp = df_fc$`Point Forecast`

mape(dat_test$tp_all, dat_test$simplexp)  #13.6%
```

```
## [1] 13.63783
```

```
### holts trend method

holt_model <- holt(dat_ts, h = 5)
summary(holt_model)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = dat_ts, h = 5)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.9999
##
##   Initial states:
##     l = 278.0224
##     b = 58.8622
##
##   sigma:  418.6783
##
##      AIC     AICc      BIC
## 1010.348 1011.420 1020.984
##
## Error measures:
##                    ME     RMSE     MAE       MPE     MAPE      MASE      ACF1
## Training set 27.50987 404.9474 214.069 0.4112807 2.415873 0.3280276 0.2266918
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2016       40847.12 40310.56 41383.67 40026.52 41667.71
## 2017       42611.42 41411.74 43811.10 40776.66 44446.18
## 2018       44375.72 42368.29 46383.15 41305.63 47445.82
## 2019       46140.03 43201.46 49078.59 41645.88 50634.17
## 2020       47904.33 43925.50 51883.16 41819.24 53989.42
```

```
df_holt = as.data.frame(holt_model)   #save results on df_holt
dat_test$holt = df_holt$`Point Forecast`     #add holt forecast to test
mape(dat_test$tp_all, dat_test$holt)  #2.3%
```

```
## [1] 2.361173
```

```
###arima

arima_model <- auto.arima(dat_ts)
summary(arima_model)
```

```
## Series: dat_ts
## ARIMA(2,2,1)
##
## Coefficients:
##          ar1      ar2      ma1
##       0.9709  -0.4352  -0.8432
## s.e.  0.1276   0.1169   0.0830
##
## sigma^2 estimated as 136268:  log likelihood=-438.69
## AIC=885.39   AICc=886.11   BIC=893.76
##
## Training set error measures:
##                   ME     RMSE      MAE       MPE     MAPE      MASE        ACF1
## Training set 56.8127 353.9476 205.8245 0.9188444 2.406438 0.3153942 -0.09027856
```

```
fore_arima = forecast::forecast(arima_model, h=5)
df_arima = as.data.frame(fore_arima)
dat_test$arima = df_arima$`Point Forecast`
mape(dat_test$tp_all, dat_test$arima)  ## 5%
```

```
## [1] 5.094956
```

```
#Tbats

model_tbats <- tbats(dat_ts)
summary(model_tbats)
```

```
##                   Length Class  Mode
## lambda              1    -none- numeric
## alpha               1    -none- numeric
## beta                1    -none- numeric
## damping.parameter   1    -none- numeric
## gamma.values        0    -none- NULL
## ar.coefficients     0    -none- NULL
## ma.coefficients     0    -none- NULL
## likelihood          1    -none- numeric
## optim.return.code   1    -none- numeric
## variance            1    -none- numeric
## AIC                 1    -none- numeric
## parameters          2    -none- list
## seed.states         2    -none- numeric
## fitted.values      62    ts     numeric
## errors             62    ts     numeric
## x                 124    -none- numeric
## seasonal.periods    0    -none- NULL
## y                  62    ts     numeric
## call                2    -none- call
## series              1    -none- character
## method              1    -none- character
```

```
for_tbats <- forecast::forecast(model_tbats, h = 5)
df_tbats = as.data.frame(for_tbats)
dat_test$tbats = df_tbats$`Point Forecast`
mape(dat_test$tp_all, dat_test$tbats) #3.8%
```

```
## [1] 3.812087
```

```
#### percentage deviation of preds
pred_results=dat_test %>% select(-c(frate,tp_ld,h_ld,nf_all,h_all,nf_ld,general_gov,Central_gov,gdp,Class))

pred_results[,-1] %>% transmute_all(.funs=function(x) 1- x/pred_results$tp_all)
```

```
## # A tibble: 5 x 6
##   tp_all naive simplexp     holt  arima  tbats
##    <dbl> <dbl>    <dbl>    <dbl>  <dbl>  <dbl>
## 1      0 0.982   0.0511 0.00823  0.0134 0.0141
## 2      0 0.983   0.0836 0.000880 0.0165 0.0127
## 3      0 0.984   0.127  0.00851  0.0366 0.0247
## 4      0 0.985   0.190  0.0441   0.0831 0.0626
## 5      0 0.986   0.230  0.0564   0.105  0.0765
```

```
df1960=df %>% filter(year>1960)




rep(1:3,each=20)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```
df1960=df1960 %>% mutate(decade=rep(1:3,each=20))
df1960
```

```
## # A tibble: 60 x 13
##    year  frate tp_all tp_ld h_all  h_ld nf_all nf_ld gener~1 Centr~2   gdp Class
##    <chr> <dbl>  <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>   <dbl>   <dbl> <dbl> <chr>
##  1 1961   1.95   562.  453.  243.  238.   319.  215.    343.    272.  559. Train
##  2 1962   2.71   607.  493.  265.  260.   342.  233.    356.    280.  600. Train
##  3 1963   3.18   664.  540.  294.  288.   369.  252.    367.    286.  633. Train
##  4 1964   3.50   723.  591.  323.  317.   399.  274.    381.    293.  680. Train
##  5 1965   4.08   796.  650.  353.  346.   444.  304.    391.    297.  737. Train
##  6 1966   5.11   862.  707.  376.  368.   486.  338.    405.    304.  808. Train
##  7 1967   4.22   935.  773.  411.  402.   524.  371.    430.    320.  854. Train
##  8 1968   5.66  1018.  839.  436.  427.   582.  413.    455.    334.  933. Train
##  9 1969   8.21  1116.  914.  464.  453.   652.  461.    463.    332. 1009. Train
## 10 1970   7.17  1185.  979.  479.  467.   706.  512.    492.    347. 1064. Train
## # ... with 50 more rows, 1 more variable: decade <int>, and abbreviated
## #   variable names 1: general_gov, 2: Central_gov
```

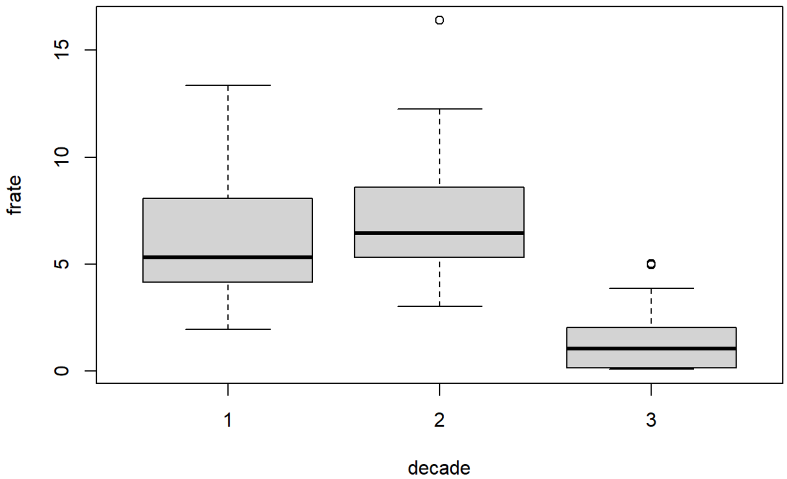
```
df1960 %>% group_by(decade) %>% summarize(sd=sd(frate)/mean(frate))
```

```
## # A tibble: 3 x 2
##   decade    sd
##    <int> <dbl>
## 1      1 0.491
## 2      2 0.438
## 3      3 1.09
```

```
df1960 %>% split(.$decade) %>%
  map_dfr(~sort(.x$frate,decreasing=TRUE))
```

```
## # A tibble: 20 x 3
##      `1`   `2`    `3`
##    <dbl> <dbl>  <dbl>
##  1 13.3  16.4  5.02
##  2 11.2  12.2  4.97
##  3 10.5  10.2  3.88
##  4  8.74  9.21 3.22
##  5  8.21  9.09 2.16
##  6  7.94  8.10 1.92
##  7  7.17  8.10 1.79
##  8  5.82  7.57 1.67
##  9  5.66  6.80 1.35
## 10  5.54  6.66 1.13
## 11  5.11  6.24 1.00
## 12  5.05  5.83 0.393
## 13  4.67  5.69 0.363
## 14  4.44  5.46 0.176
## 15  4.22  5.35 0.159
## 16  4.08  5.30 0.141
## 17  3.50  4.97 0.134
## 18  3.18  4.21 0.107
## 19  2.71  3.52 0.102
## 20  1.95  3.02 0.0885
```

```
boxplot(frate~decade,data=df1960)
```


frate
0
5
10
15
1
2
3
decade