Raport

Marcin Łukaszyk

March 31, 2020

First Article

Associative Classification in R: arc, arulesCBA, and rCBA.

Link to the article. The peper is from 2018-05-29 so it should be repricleble. Article contains intorduction to the CBA algorithm. Our problem is a classification problem. We want to predict species using the four measurements. To do it we will use algorithms named before.

First we dived iris data set to a trainig and testting set in 80:20 proportion.

```
data("iris")
iris <- iris[sample(seq(nrow(iris))), ]
iris_train <- iris[1:(nrow(iris)*.8), ]
iris_test <- iris[-(1:(nrow(iris)*.8)), ]

kable(head(iris_train))
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
28	5.2	3.5	1.5	0.2	setosa
80	5.7	2.6	3.5	1.0	versicolor
101	6.3	3.3	6.0	2.5	virginica
111	6.5	3.2	5.1	2.0	virginica
137	6.3	3.4	5.6	2.4	virginica
133	6.4	2.8	5.6	2.2	virginica

```
kable(head(iris_test))
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
124	6.3	2.7	4.9	1.8	virginica
77	6.8	2.8	4.8	1.4	versicolor
23	4.6	3.6	1.0	0.2	setosa
15	5.8	4.0	1.2	0.2	setosa
112	6.4	2.7	5.3	1.9	virginica
37	5.5	3.5	1.3	0.2	setosa

As we can se the table differs due to nature of a sample function. It samples randomly thus making results diffrent each time without setting a seed. Unfortunately authors didnt set one in the paper or if they did

they set diffrent one that is in the soruce code. We eill use set from source code

arulesCBA

```
iris_train_disc <- discretizeDF.supervised(Species ~ ., data = iris_train,
method = "mdlp")
kable(head(iris_train_disc))
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
28	[-Inf,5.45)	[2.95, Inf]	[-Inf,2.45)	[-Inf,0.75)	setosa
80	[5.45,6.25)	[-Inf,2.95)	[2.45,4.75)	[0.75,1.65)	versicolor
101	[6.25, Inf]	[2.95, Inf]	[4.75, Inf]	[1.65, Inf]	virginica
111	[6.25, Inf]	[2.95, Inf]	[4.75, Inf]	[1.65, Inf]	virginica
137	[6.25, Inf]	[2.95, Inf]	[4.75, Inf]	[1.65, Inf]	virginica
133	[6.25, Inf]	[-Inf,2.95)	[4.75, Inf]	[1.65, Inf]	virginica

We got similar resluts. Thats sign that the algorithm works.

```
trans_train <- as(iris_train_disc, "transactions")
inspect(head(trans_train, n = 3))
```

```
##     items                       transactionID
## [1] {Sepal.Length=[-Inf,5.45),
##      Sepal.Width=[2.95, Inf],
##      Petal.Length=[-Inf,2.45),
##      Petal.Width=[-Inf,0.75),
##      Species=setosa}                       28
## [2] {Sepal.Length=[5.45,6.25),
##      Sepal.Width=[-Inf,2.95),
##      Petal.Length=[2.45,4.75),
##      Petal.Width=[0.75,1.65),
##      Species=versicolor}                   80
## [3] {Sepal.Length=[6.25, Inf],
##      Sepal.Width=[2.95, Inf],
##      Petal.Length=[4.75, Inf],
##      Petal.Width=[1.65, Inf],
##      Species=virginica}                   101
```

```
rules <- apriori(trans_train, parameter = list(support = 0.01, confidence = 0.8),
  appearance = list(rhs = grep("Species=", itemLabels(trans_train), value = TRUE),
  default = "lhs"))
```

```
rules <- mineCARs(Species ~ ., data = trans_train, support = 0.01, confidence = 0.8)
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
```

```
##         0.8    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 1
##
## set item appearances ...[14 item(s)] done [0.00s].
## set transactions ...[14 item(s), 120 transaction(s)] done [0.00s].
## sorting and recoding items ... [14 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [83 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(head(rules,n=3))
```

```
##     lhs                           rhs              support   confidence
## [1] {Petal.Length=[-Inf,2.45)} => {Species=setosa} 0.3083333 1.0000000
## [2] {Petal.Width=[-Inf,0.75)}  => {Species=setosa} 0.3083333 1.0000000
## [3] {Sepal.Length=[-Inf,5.45)} => {Species=setosa} 0.2916667 0.8536585
##     lift     count
## [1] 3.243243 37
## [2] 3.243243 37
## [3] 2.768622 35
```

We come to simmilar yet diffrent result.

arc

Next we got a "arc" R package. This is package that implements CBA algorithm.

We can learn CBA model from iris data set in method shown below.

```
classifier <- arc::cba(iris_train, "Species")
```

```
## Using automatic threshold detection

## Running apriori with setting: confidence =  0.5 , support =  0 , minlen =  2 , maxlen =  3 , MAX_RULE

## Rule count:  99  Iteration:  1

## Increasing maxlen to:  4

## Running apriori with setting: confidence =  0.5 , support =  0 , minlen =  2 , maxlen =  4 , MAX_RULE

## Rule count:  508  Iteration:  2
```

```
## Increasing maxlen to:   5

## Running apriori with setting: confidence =  0.5 , support =  0 , minlen =  2 , maxlen =  5 , MAX_RULE

## Rule count:  1468  Iteration:  3

## Target rule count satisfied:   1000

## Removing excess discovered rules

## Rule learning took: 0.04  seconds

## Original rules:  1000

## Rules after data coverage pruning: 8

## Performing default rule pruning.

## Final rule list size:  3

## Pruning took: 0.18  seconds
```

```
inspect(classifier@rules)
```

```
##     lhs                           rhs                    support confidence     lift count lhs_length
## [1] {Petal.Length=(2.45;4.75],
##      Petal.Width=(0.75;1.65]}  => {Species=versicolor} 0.3250000  1.0000000 2.857143    39  2.000000
## [2] {Petal.Length=[-Inf;2.45]} => {Species=setosa}     0.3083333  1.0000000 3.243243    37  1.000000
## [3] {}                         => {Species=virginica}  0.3416667  0.3416667 1.000000     0  0.341666
```

arulesCBA

Now we will look at the arulesCBA package.

```
classifier <- arulesCBA::CBA(Species ~ ., data = iris_train,
supp = 0.05, confidence = 0.9)
```

```
classifier
```

```
## CBA Classifier Object
## Class: Species=setosa, Species=versicolor, Species=virginica
## Default Class: Species=virginica
## Number of rules: 2
## Classification method: first
## Description: CBA algorithm by Liu, et al. 1998 with support=0.05 and
##      confidence=0.9
```

We got te same results.

rCBA

Lastly we will look at the rCBA package.

```
library("rCBA")
```

```
## Loading required package: rJava

##
## Attaching package: 'rJava'

## The following object is masked from 'package:R.oo':
##
##     clone
```

```
classifier <- rCBA::build(iris_train)
```

```
## 2020-03-31 23:12:32 rCBA: initialized

## 2020-03-31 23:12:32 rCBA: data 120x5

##   took: 0.01  s

## Iteration: 0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-3

## 2020-03-31 23:12:43 rCBA: best solution 0.6872, 0.079, 2

##   took: 11.71  s

## 2020-03-31 23:12:43 rCBA: rules 3

##   took: 0  s

## 2020-03-31 23:12:43 rCBA: pruned rules 4

##   took: 0.02  s
```

```
inspect(classifier$model)
```

```
##     lhs                   rhs                   support   confidence lift
## [1] {Petal.Width=0.2}  => {Species=setosa}     0.1750000 1.0000000  3.243243
## [2] {Petal.Width=1.3}  => {Species=versicolor} 0.1000000 1.0000000  2.857143
## [3] {Petal.Length=1.5} => {Species=setosa}     0.1000000 1.0000000  3.243243
## [4] {}                 => {Species=virginica}  0.3416667 0.3416667  1.000000
```

Conclusios

This article is repriceable.

Second Article

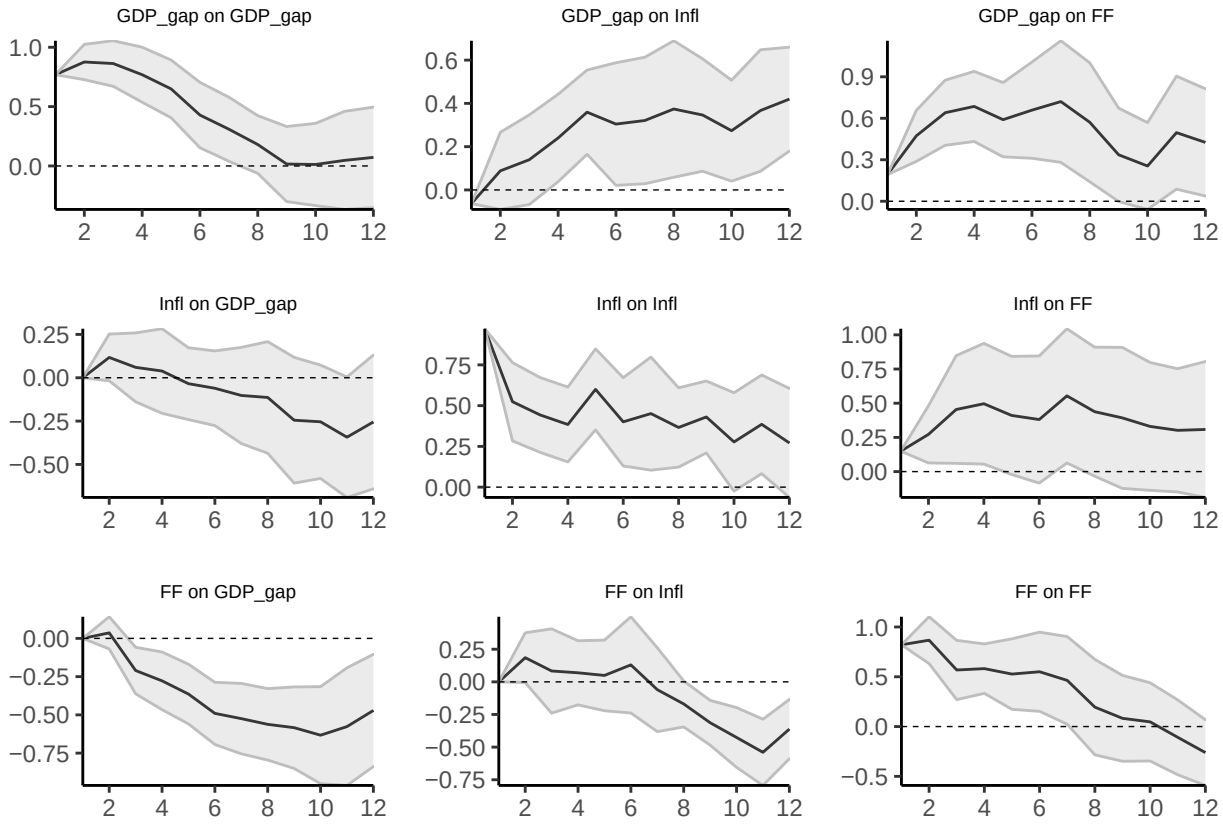
In this paper we use “lpirfs” package which is used to analysis of impulse in econometrics. Link to the peper

```
library(lpirfs)
library(ggpubr)
library(gridExtra)
library(tidyverse)
```

```
endog_data <- interest_rules_var_data
```

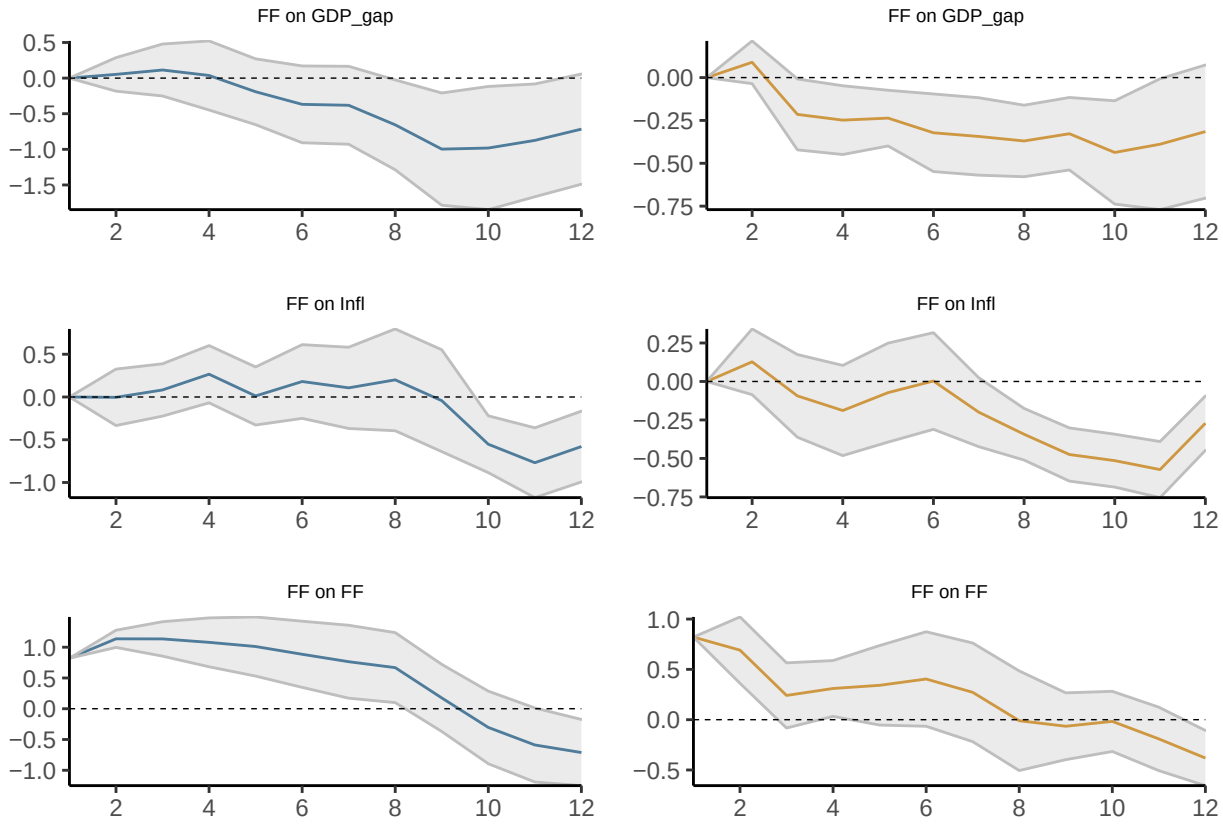
```
results_lin <- lp_lin(endog_data,
                      lags_endog_lin = 4,
                      trend          = 0,
                      shock_type     = 0,
                      confint        = 1.96,
                      hor            = 12)

linear_plots <- plot_lin(results_lin)
```

```
lin_plots_all <- sapply(linear_plots, ggplotGrob)
lp_lin_plots  <- marrangeGrob(lin_plots_all, nrow = ncol(endog_data),
                                             ncol = ncol(endog_data), top = NULL)

lp_lin_plots
```



We got simmilar resulsts.

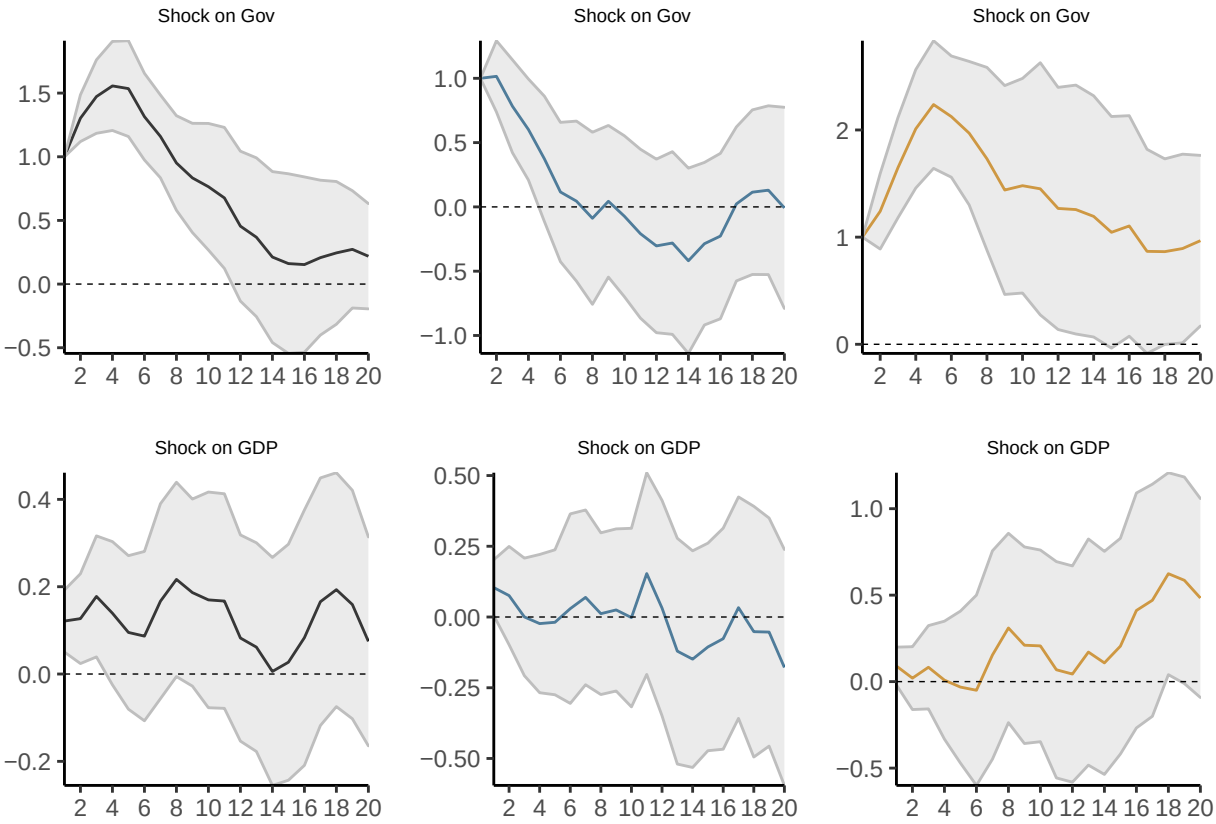
```
switching_data <-  if_else(dplyr::lag(endog_data$Infl, 3) > 4.75, 1, 0)
results_nl    <- lp_nl(endog_data,
                       lags_endog_lin  = 4,
                       lags_endog_nl   = 4,
                       trend           = 0,
                       shock_type      = 0,
                       confint         = 1.96,
                       hor             = 12,
                       switching       = switching_data,
                       lag_switching   = F,
                       use_logistic    = F)

 nl_plots <- plot_nl(results_nl)

 single_plots <- nl_plots$gg_s1[c(3, 6, 9)]
single_plots[4:6] <- nl_plots$gg_s2[c(3, 6, 9)]
all_plots <- sapply(single_plots, ggplotGrob)
marrangeGrob(all_plots, nrow = 3, ncol = 2, top = NULL)
```



Everything works as intended.

```
ag_data       <- ag_data
sample_start  <- 7
sample_end    <- dim(ag_data)[1]

endog_data    <- ag_data[sample_start:sample_end,3:5]
```

```
shock         <- ag_data[sample_start:sample_end, 3]

results_lin_iv <- lp_lin_iv(endog_data,
                            lags_endog_lin = 4,
                            shock          = shock,
                            trend          = 0,
                            confint        = 1.96,
                            hor            = 20)


iv_lin_plots    <- plot_lin(results_lin_iv)

ag_data           <- ag_data
endog_data        <- ag_data[sample_start:sample_end, 3:5]

shock             <- ag_data[sample_start:sample_end, 7]

exog_data         <- ag_data[sample_start:sample_end, 6]

switching_variable <- ag_data$GDP_MA[sample_start:sample_end] - 0.8

results_nl_iv <- lp_nl_iv(endog_data,
                          lags_endog_nl     = 3,
                          shock             = shock,
                          exog_data         = exog_data,
                          lags_exog         = 4,
                          trend             = 0,
                          confint           = 1.96,
                          hor               = 20,
                          switching         = switching_variable,
                          use_hp            = FALSE,
                          gamma             = 3)

plots_nl_iv <- plot_nl(results_nl_iv)

combine_plots <- list()

combine_plots[[1]] <- iv_lin_plots[[1]]
combine_plots[[2]] <- iv_lin_plots[[3]]

combine_plots[[3]] <- plots_nl_iv$gg_s1[[1]]
combine_plots[[4]] <- plots_nl_iv$gg_s1[[3]]

combine_plots[[5]] <- plots_nl_iv$gg_s2[[1]]
combine_plots[[6]] <- plots_nl_iv$gg_s2[[3]]

lin_plots_all     <- sapply(combine_plots, ggplotGrob)
combine_plots_all <- marrangeGrob(lin_plots_all, nrow = 2, ncol = 3, top = NULL)
combine_plots_all
```



We recreated plots from article.

```
  library(httr)
  library(readxl)
```

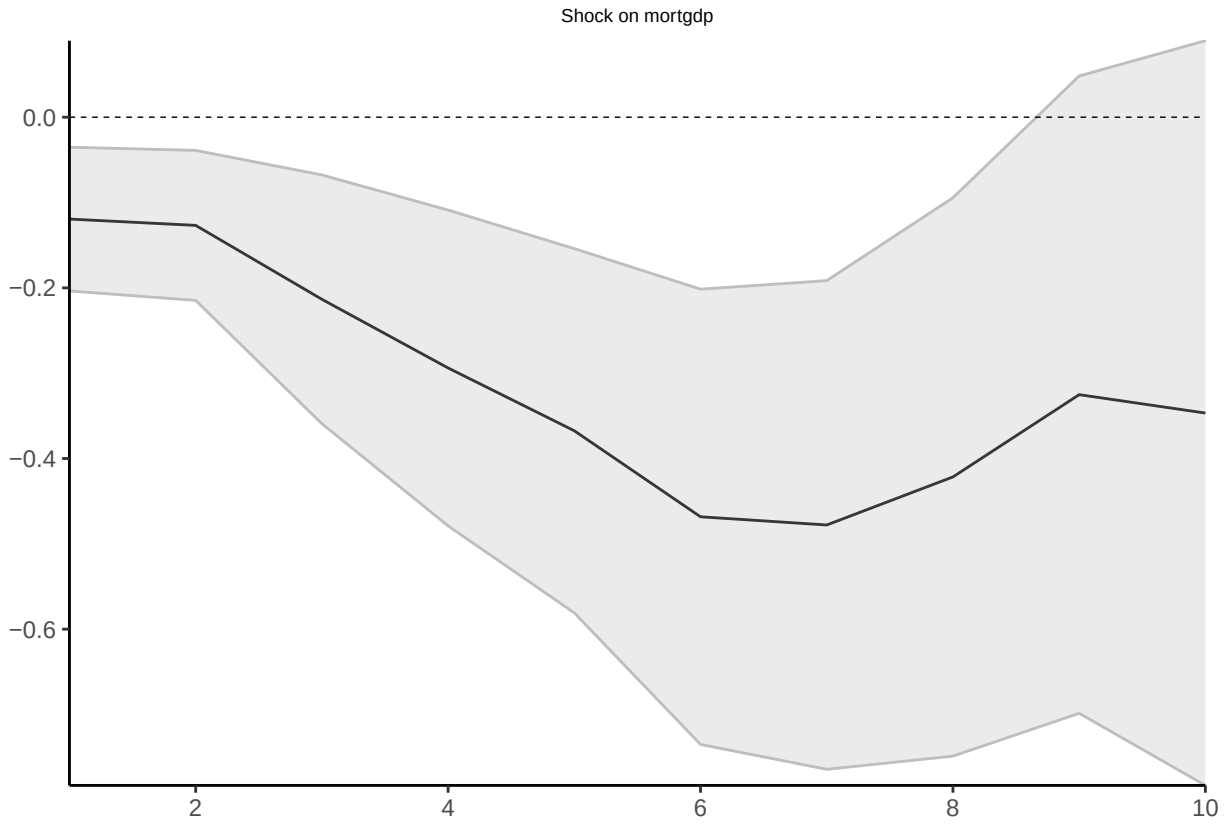
```
## Warning: package 'readxl' was built under R version 3.6.2
```

```
  library(dplyr)

  url_jst <-"http://www.macrohistory.net/JST/JSTdatasetR3.xlsx"
  GET(url_jst, write_disk(jst_link <- tempfile(fileext = ".xlsx")))
```

```
## Response [http://www.macrohistory.net/JST/JSTdatasetR3.xlsx]
##   Date: 2020-03-31 21:15
##   Status: 200
##   Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
##   Size: 635 kB
## <ON DISK>  C:\Users\lukas\AppData\Local\Temp\Rtmpau9UxC\file6cc483730de.xlsx
```

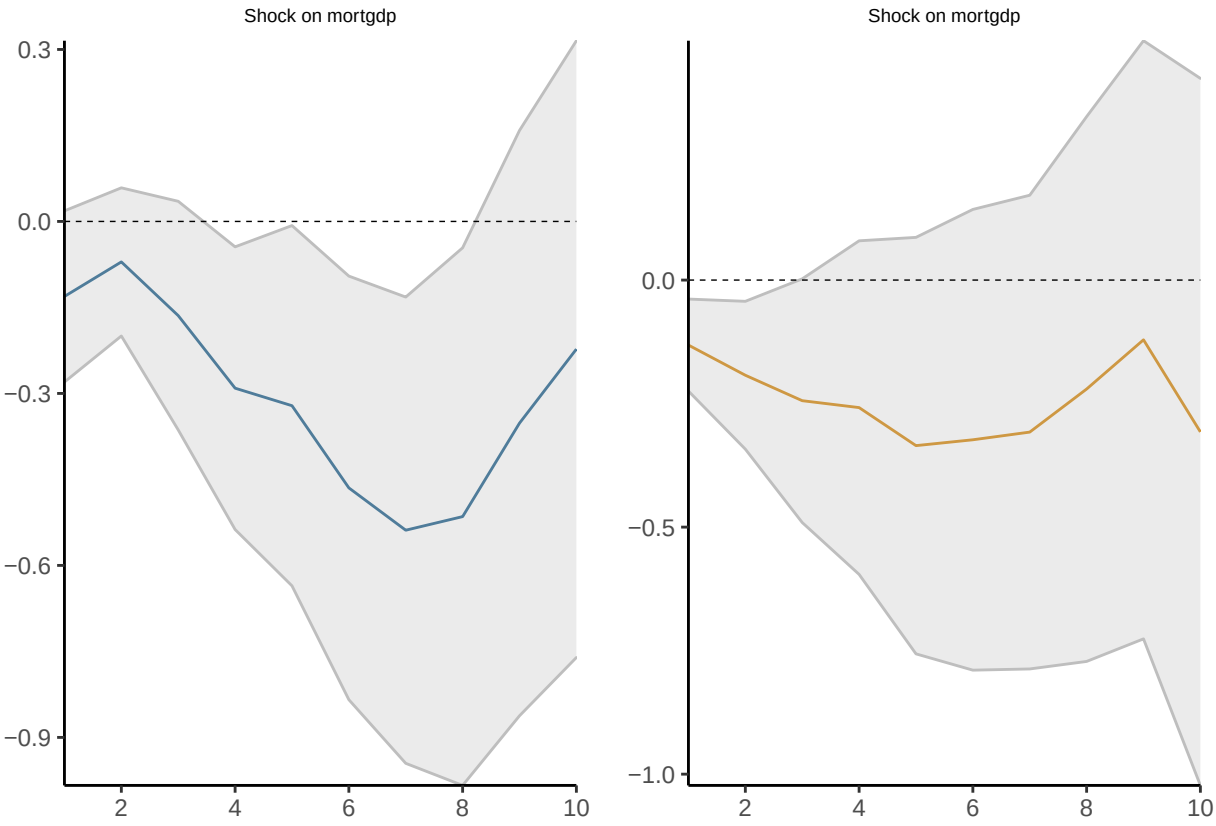
```
  jst_data <- read_excel(jst_link, 2L)

  jst_data <- jst_data                      %>%
              dplyr::filter(year <= 2013) %>%
              dplyr::select(country, year, everything())
```

```r
data_set <- jst_data %>%
            mutate(stir    = stir)                          %>%
            mutate(mortgdp = 100*(tmort/gdp))               %>%
            mutate(hpreal  = hpnom/cpi)                     %>%
            group_by(country)                               %>%
            mutate(hpreal  = hpreal/hpreal[year==1990][1])  %>%
            mutate(lhpreal = log(hpreal))                   %>%

            mutate(lhpy    = lhpreal - log(rgdppc))         %>%
            mutate(lhpy    = lhpy - lhpy[year == 1990][1])  %>%
            mutate(lhpreal = 100*lhpreal)                   %>%
            mutate(lhpy    = 100*lhpy)                      %>%
            ungroup()                                       %>%

            mutate(lrgdp   = 100*log(rgdppc))               %>%
            mutate(lcpi    = 100*log(cpi))                  %>%
            mutate(lriy    = 100*log(iy*rgdppc))            %>%
            mutate(cay     = 100*(ca/gdp))                  %>%
            mutate(tnmort  = tloans - tmort)                %>%
            mutate(nmortgdp = 100*(tnmort/gdp))             %>%
            dplyr::select(country, year, mortgdp, stir, ltrate,
                                                  lhpy, lrgdp, lcpi, lriy, cay, nmortgdp)


data_sample <- seq(1870, 2016)[which(!(seq(1870, 2016) %in%
                                       c(seq(1914, 1918),
                                       seq(1939, 1947))))]

results_panel <- lp_lin_panel(data_set  = data_set,  data_sample  = data_sample,
                              endog_data        = "mortgdp", cumul_mult   = TRUE,
                              shock             = "stir",    diff_shock   = TRUE,
                              panel_model       = "within",  panel_effect = "individual",
                              robust_cov        = "vcovSCC", c_exog_data  = "cay",
                              c_fd_exog_data    = colnames(data_set)[c(seq(4,9),11)],
                              l_fd_exog_data    = colnames(data_set)[c(seq(3,9),11)],
                              lags_fd_exog_data = 2,      confint      = 1.67,
                              hor               = 10)

plot_lin_panel <- plot_lin(results_panel)
plot(plot_lin_panel[[1]])
```



The last plot

```
results_panel <- lp_nl_panel(data_set          = data_set,
                             data_sample       = data_sample,
                             endog_data        = "mortgdp", cumul_mult     = TRUE,
                             shock             = "stir",    diff_shock     = TRUE,
                             panel_model       = "within",  panel_effect   = "individual",
                             robust_cov        = "vcovSCC", switching      = "lrgdp",
                             lag_switching     = TRUE,      use_hp         = TRUE,
                             lambda            = 6.25,      gamma          = 10,
                             c_exog_data       = "cay",
                             c_fd_exog_data    = colnames(data_set)[c(seq(4,9),11)],
                             l_fd_exog_data    = colnames(data_set)[c(seq(3,9),11)],
                             lags_fd_exog_data = 2,
                             confint           = 1.67,
                             hor               = 10)

 nl_plots      <- plot_nl(results_panel)
 combine_plots <- list(nl_plots$gg_s1[[1]], nl_plots$gg_s2[[1]])
 nonlinear_panel_plots <- marrangeGrob(combine_plots, nrow = 1, ncol = 2, top = NULL)
 nonlinear_panel_plots
```



Whole article is replicable

Third Article

Last arlicle will be about glmperm packege. It is used for inferencing with small and moderate sized data sets. You can access paper here. This packege is not on CRAN so we need to install it manually from github.

```
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 3.6.3
```

```
## Loading required package: usethis
```

```
## Warning: package 'usethis' was built under R version 3.6.3
```

```
##
## Attaching package: 'devtools'
```

```
## The following object is masked from 'package:rCBA':
##
##     build
```

```
## The following objects are masked from 'package:R.oo':
##
##     check, unload
```

```
install_github('cran/glmperm')
```

```
## Skipping install of 'glmperm' from a github remote, the SHA1 (9b5ed21c) has not changed since last in
##   Use `force = TRUE` to force installation
```

```
library(glmperm)
```

```
## Loading required package: survival
```

```
n <- 20
set.seed(4278)
x1 <- rnorm(n)
x0 <- rnorm(n)+x1
y1 <- ifelse(x0+x1+2*rnorm(n)>0,1,0)
test1 <- prr.test(y1~x0+x1,
var="x0", family=binomial())
x2 <- rbinom(n,1,0.6)
y2 <- ifelse(x1+x2+rnorm(n)>0,1,0)
test2 <- prr.test(y2~x1+x2, var="x1",
nrep=10000,family=binomial())
set.seed(4278)
x1 <- rnorm(n)
x0 <- rnorm(n) + x1
nu <- rgamma(n, shape = 2, scale = 1)
y <- rpois(n, lambda = exp(2) * nu)
test3 <- prr.test(y~x0+x1,
var="x0", family=poisson())
test4 <- prr.test(y~x0, var="x0",
nrep=1000,family=poisson())
```

```
summary(test2)
```

```
##
##
##     Permutation of Regressor Residuals Test:
##
##
## Call:
##  prr.test(formula = y2 ~ x1 + x2, var = "x1", family = binomial(),      nrep = 10000)
##
## number of observations used:  20
##
## null hypothesis: regression coefficient of covariate x1 = 0
## observed Likelihood Ratio Test Statistics:  20.6857
##
##     ----------------------------------------
##     Results based on chi-squared distribution
```

```
##     ----------------------------------------
##
## observed p-value: 0
##
##     ------------------------------------------------
##     Results based on permutation of regressor residuals
##     ------------------------------------------------
##
## permutation p-value for simulated p-values <= observed p-value: 1e-04 (Std.err: 1e-04)
##
## permutation p-value for simulated p-values <= 1.005 observed p-value: 1e-04 (Std.err: 1e-04)
##
## permutation p-value for simulated p-values <= 1.01 observed p-value: 1e-04 (Std.err: 1e-04)
##
## permutation p-value for simulated p-values <= 1.02 observed p-value: 1e-04 (Std.err: 1e-04)
##
## permutation p-value for simulated p-values <= 1.04 observed p-value: 1e-04 (Std.err: 1e-04)
##
## ********************************************************************************
## WARNING: estimated dispersion is < 0.5, rather use family = quasibinomial
## ********************************************************************************
```

```
library('coin')
```

```
## Warning: package 'coin' was built under R version 3.6.3
```

```
##
## Attaching package: 'coin'
```

```
## The following objects are masked from 'package:arules':
##
##     size, support
```

```
data(treepipit, package="coin")
test5<-prr.test(counts~cbpiles+coverstorey
+coniferous+coverregen,data=treepipit,
var="cbpiles",family=poisson())
```

```
## Warning in prr.test(counts ~ cbpiles + coverstorey + coniferous + coverregen, :
## estimated dispersion is > 1.5, rather use family = quasipoisson
```

```
summary(test5)
```

```
##
##
##     Permutation of Regressor Residuals Test:
##
##
## Call:
##  prr.test(formula = counts ~ cbpiles + coverstorey + coniferous +      coverregen, var = "cbpiles", f
##
```

```
## number of observations used:  86
##
## null hypothesis: regression coefficient of covariate cbpiles = 0
## observed Likelihood Ratio Test Statistics:  8.4202
##
##     ----------------------------------------
##     Results based on chi-squared distribution
##     ----------------------------------------
##
## observed p-value: 0.0037
##
##     ----------------------------------------------------
##     Results based on permutation of regressor residuals
##     ----------------------------------------------------
##
## permutation p-value for simulated p-values <= observed p-value: 0.069 (Std.err: 0.008)
##
## permutation p-value for simulated p-values <= 1.005 observed p-value: 0.069 (Std.err: 0.008)
##
## permutation p-value for simulated p-values <= 1.01 observed p-value: 0.069 (Std.err: 0.008)
##
## permutation p-value for simulated p-values <= 1.02 observed p-value: 0.069 (Std.err: 0.008)
##
## permutation p-value for simulated p-values <= 1.04 observed p-value: 0.069 (Std.err: 0.008)
##
## ************************************************************************
## WARNING: estimated dispersion is > 1.5, rather use family = quasipoisson
## ************************************************************************
```

```
test6<-prr.test(counts~cbpiles+coverstorey
+coniferous+coverregen,data=treepipit,
var="cbpiles",family=quasipoisson())
summary(test6)
```

```
##
##
##     Permutation of Regressor Residuals Test:
##
##
## Call:
##  prr.test(formula = counts ~ cbpiles + coverstorey + coniferous +      coverregen, var = "cbpiles", f
##
## number of observations used:  86
##
## null hypothesis: regression coefficient of covariate cbpiles = 0
## observed Likelihood Ratio Test Statistics:  3.4019
##
##     ----------------------------------------
##     Results based on chi-squared distribution
##     ----------------------------------------
##
## observed p-value: 0.0651
##
##     ----------------------------------------------------
```

```
##     Results based on permutation of regressor residuals
##     ---------------------------------------------------
##
## permutation p-value for simulated p-values <= observed p-value: 0.064 (Std.err: 0.0077)
##
## permutation p-value for simulated p-values <= 1.005 observed p-value: 0.064 (Std.err: 0.0077)
##
## permutation p-value for simulated p-values <= 1.01 observed p-value: 0.064 (Std.err: 0.0077)
##
## permutation p-value for simulated p-values <= 1.02 observed p-value: 0.064 (Std.err: 0.0077)
##
## permutation p-value for simulated p-values <= 1.04 observed p-value: 0.066 (Std.err: 0.0079)
```

Everything works.

Summary

All of given articles are repilicable. The only problem was were the packege wasn't on cran but it is easily fixed when it is on a git hub page.