# Web Appendix of "Fast and Reliable Computation of Generalized Synthetic Controls"

Martin Becker
Statistics and Econometrics
Saarland University

Stefan Klößner
Statistics and Econometrics
Saarland University

January 30, 2017

**Abstract**

This is the web appendix of "Fast and Reliable Computation of Generalized Synthetic Controls".

## A Data Preparation

To exemplify various issues discussed in the main paper, we consider the estimation of synthetic control units for the Basque country and for Catalonia (placebo study), as done in Abadie and Gardeazabal (2003). We start with loading and attaching the R packages **Synth** and **MSCMT** and the preparation of the data (as contained in the example on the manual page of `synth` in package **Synth**) for Basque country as treated unit:

```
library(Synth)

## ##
## ## Synth Package:  Implements Synthetic Control Methods.
## ## See http://www.mit.edu/~jhainm/software.htm for additional information.

library(MSCMT)
data(basque)

# dataprep: prepare data for synth
dataprep.out <-
  dataprep(
  foo = basque
  ,predictors= c("school.illit",
                 "school.prim",
                 "school.med",
                 "school.high",
                 "school.post.high"
                 ,"invest"
                 )
   ,predictors.op = c("mean")
   ,dependent     = c("gdpcap")
   ,unit.variable = c("regionno")
   ,time.variable = c("year")
   ,special.predictors = list(
```

```
    list("gdpcap",1960:1969,c("mean")),
    list("sec.agriculture",seq(1961,1969,2),c("mean")),
    list("sec.energy",seq(1961,1969,2),c("mean")),
    list("sec.industry",seq(1961,1969,2),c("mean")),
    list("sec.construction",seq(1961,1969,2),c("mean")),
    list("sec.services.venta",seq(1961,1969,2),c("mean")),
    list("sec.services.nonventa",seq(1961,1969,2),c("mean")),
    list("popdens",1969,c("mean")))
    ,treatment.identifier  = 17
    ,controls.identifier   = c(2:16,18)
    ,time.predictors.prior = c(1964:1969)
    ,time.optimize.ssr     = c(1960:1969)
    ,unit.names.variable   = c("regionname")
    ,time.plot            = c(1955:1997)
    )

# 1. combine highest and second highest
# schooling category and eliminate highest category
dataprep.out$X1["school.high",] <-
 dataprep.out$X1["school.high",] +
 dataprep.out$X1["school.post.high",]
dataprep.out$X1                  <-
 as.matrix(dataprep.out$X1[
  -which(rownames(dataprep.out$X1)=="school.post.high"),])
dataprep.out$X0["school.high",] <-
 dataprep.out$X0["school.high",] +
 dataprep.out$X0["school.post.high",]
dataprep.out$X0                  <-
dataprep.out$X0[
 -which(rownames(dataprep.out$X0)=="school.post.high"),]

# 2. make total and compute shares for the schooling catgeories
lowest  <- which(rownames(dataprep.out$X0)=="school.illit")
highest <- which(rownames(dataprep.out$X0)=="school.high")

dataprep.out$X1[lowest:highest,] <-
 (100 * dataprep.out$X1[lowest:highest,]) /
 sum(dataprep.out$X1[lowest:highest,])
dataprep.out$X0[lowest:highest,] <-
 100 * scale(dataprep.out$X0[lowest:highest,],
             center=FALSE,
             scale=colSums(dataprep.out$X0[lowest:highest,])
                                                )
# the following step has been added to provide a consistent dataprep object
# 3. fix dataprep object (remove school.post.high from list of predictors)
dataprep.out$tag$predictors <- dataprep.out$tag$predictors[-5]
dataprep.out.Basque <- dataprep.out
```

Next, we modify the example on the manual page of `synth` in package **Synth** to prepare the data for Catalonia as treated unit:

```
# dataprep: prepare data for synth (treated unit: Catalonia)
dataprep.out <-
  dataprep(
  foo = basque
  ,predictors= c("school.illit",
                 "school.prim",
```

```
                 "school.med",
                 "school.high",
                 "school.post.high"
                 ,"invest"
                 )
   ,predictors.op = c("mean")
   ,dependent     = c("gdpcap")
   ,unit.variable = c("regionno")
   ,time.variable = c("year")
   ,special.predictors = list(
    list("gdpcap",1960:1969,c("mean")),
    list("sec.agriculture",seq(1961,1969,2),c("mean")),
    list("sec.energy",seq(1961,1969,2),c("mean")),
    list("sec.industry",seq(1961,1969,2),c("mean")),
    list("sec.construction",seq(1961,1969,2),c("mean")),
    list("sec.services.venta",seq(1961,1969,2),c("mean")),
    list("sec.services.nonventa",seq(1961,1969,2),c("mean")),
    list("popdens",1969,c("mean")))
    ,treatment.identifier  = 10
    ,controls.identifier   = c(2:9,11:16,18)
    ,time.predictors.prior = c(1964:1969)
    ,time.optimize.ssr     = c(1960:1969)
    ,unit.names.variable   = c("regionname")
    ,time.plot            = c(1955:1997)
    )

# 1. combine highest and second highest
# schooling category and eliminate highest category
dataprep.out$X1["school.high",] <-
 dataprep.out$X1["school.high",] +
 dataprep.out$X1["school.post.high",]
dataprep.out$X1                 <-
 as.matrix(dataprep.out$X1[
  -which(rownames(dataprep.out$X1)=="school.post.high"),])
dataprep.out$X0["school.high",] <-
 dataprep.out$X0["school.high",] +
 dataprep.out$X0["school.post.high",]
dataprep.out$X0                 <-
dataprep.out$X0[
 -which(rownames(dataprep.out$X0)=="school.post.high"),]

# 2. make total and compute shares for the schooling catgeories
lowest  <- which(rownames(dataprep.out$X0)=="school.illit")
highest <- which(rownames(dataprep.out$X0)=="school.high")

dataprep.out$X1[lowest:highest,] <-
 (100 * dataprep.out$X1[lowest:highest,]) /
 sum(dataprep.out$X1[lowest:highest,])
dataprep.out$X0[lowest:highest,] <-
 100 * scale(dataprep.out$X0[lowest:highest,],
             center=FALSE,
             scale=colSums(dataprep.out$X0[lowest:highest,])
                                                 )
# the following step has been added to provide a consistent dataprep object
# 3. fix dataprep object (remove school.post.high from list of predictors)
dataprep.out$tag$predictors <- dataprep.out$tag$predictors[-5]
dataprep.out.Catalonia <- dataprep.out
```

# B Why Predictor Weights Must Be Bounded Away From Zero

In the paper, we state that $W^*(v_1, \ldots, v_K)$ and, as a consequence, the outer objective function

$$(v_1, \ldots, v_K) \mapsto W^*(v_1, \ldots, v_K)' \widetilde{Z}' \widetilde{Z} W^*(v_1, \ldots, v_K)$$

are often discontinuous around vectors of predictor weights $v = (v_1, \ldots, v_K)$ that are not entirely positive, i.e., that have one or more components $k$ with $v_k = 0$. To exemplify this, we consider the sequences $v_1(\varepsilon)$ and $v_2(\varepsilon)$ of predictor weights as defined below.

```
v_1 <- sapply(10^(-(6:10)),function(x) c(1,1,1,1,2*x,2*x,rep(x,6),2*x))
v_2 <- sapply(10^(-(6:10)),function(x) c(1,1,1,1,x,x,rep(2*x,6),x))
print(v_1)

##        [,1]  [,2]  [,3]  [,4]  [,5]
##  [1,] 1e+00 1e+00 1e+00 1e+00 1e+00
##  [2,] 1e+00 1e+00 1e+00 1e+00 1e+00
##  [3,] 1e+00 1e+00 1e+00 1e+00 1e+00
##  [4,] 1e+00 1e+00 1e+00 1e+00 1e+00
##  [5,] 2e-06 2e-07 2e-08 2e-09 2e-10
##  [6,] 2e-06 2e-07 2e-08 2e-09 2e-10
##  [7,] 1e-06 1e-07 1e-08 1e-09 1e-10
##  [8,] 1e-06 1e-07 1e-08 1e-09 1e-10
##  [9,] 1e-06 1e-07 1e-08 1e-09 1e-10
## [10,] 1e-06 1e-07 1e-08 1e-09 1e-10
## [11,] 1e-06 1e-07 1e-08 1e-09 1e-10
## [12,] 1e-06 1e-07 1e-08 1e-09 1e-10
## [13,] 2e-06 2e-07 2e-08 2e-09 2e-10

print(v_2)

##        [,1]  [,2]  [,3]  [,4]  [,5]
##  [1,] 1e+00 1e+00 1e+00 1e+00 1e+00
##  [2,] 1e+00 1e+00 1e+00 1e+00 1e+00
##  [3,] 1e+00 1e+00 1e+00 1e+00 1e+00
##  [4,] 1e+00 1e+00 1e+00 1e+00 1e+00
##  [5,] 1e-06 1e-07 1e-08 1e-09 1e-10
##  [6,] 1e-06 1e-07 1e-08 1e-09 1e-10
##  [7,] 2e-06 2e-07 2e-08 2e-09 2e-10
##  [8,] 2e-06 2e-07 2e-08 2e-09 2e-10
##  [9,] 2e-06 2e-07 2e-08 2e-09 2e-10
## [10,] 2e-06 2e-07 2e-08 2e-09 2e-10
## [11,] 2e-06 2e-07 2e-08 2e-09 2e-10
## [12,] 2e-06 2e-07 2e-08 2e-09 2e-10
## [13,] 1e-06 1e-07 1e-08 1e-09 1e-10
```

The columns of $v_1$ and $v_2$ correspond to predictor weights, which from the left to the right move more and more towards predictor weights $(1, 1, 1, 1, 0, 0, 0, 0, 0, 0)'$.

For each of the columns of $v_1$ and $v_2$, the corresponding vector $W^*(V)$ of donor weights is calculated using `mscmt`, which invokes the `WNNLS` algorithm (see Haskell and Hanson (1981) and Hanson and Haskell (1982)). `WNNLS` has turned out to be the method of choice for solving the inner optimization, see the corresponding benchmark in the main paper.

```r
W_1 <- apply(v_1,2,function(x) mscmt(dataprep.out.Basque,outer.optim="fixed",
        outer.opar=list(v=x),std.v="max",verbose=FALSE)$w)
W_2 <- apply(v_2,2,function(x) mscmt(dataprep.out.Basque,outer.optim="fixed",
        outer.opar=list(v=x),std.v="max",verbose=FALSE)$w)
print(100*W_1)
```

```
##                                  [,1]       [,2]       [,3]       [,4]       [,5]
## Andalucia                   0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## Aragon                      0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## Principado De Asturias      8.1674172  8.1674032  8.1674018  8.1674017  8.1674017
## Baleares (Islas)            0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## Canarias                    0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## Cantabria                  74.5889848 74.5889900 74.5889905 74.5889906 74.5889906
## Castilla Y Leon             0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## Castilla-La Mancha          0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## Cataluna                    0.3046507  0.3040748  0.3040172  0.3040114  0.3040108
## Comunidad Valenciana        0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## Extremadura                 0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## Galicia                     0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## Madrid (Comunidad De)      10.8775233 10.8774247 10.8774148 10.8774138 10.8774137
## Murcia (Region de)          0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## Navarra (Comunidad Foral De) 6.0614241  6.0621073  6.0621757  6.0621825  6.0621832
## Rioja (La)                  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
```

```r
print(100*W_2)
```

```
##                                 [,1]      [,2]      [,3]      [,4]      [,5]
## Andalucia                   0.000000  0.000000  0.000000  0.000000  0.000000
## Aragon                      0.000000  0.000000  0.000000  0.000000  0.000000
## Principado De Asturias      0.000000  0.000000  0.000000  0.000000  0.000000
## Baleares (Islas)            0.000000  0.000000  0.000000  0.000000  0.000000
## Canarias                    0.000000  0.000000  0.000000  0.000000  0.000000
## Cantabria                  79.336606 79.335777 79.335694 79.335686 79.335685
## Castilla Y Leon             0.000000  0.000000  0.000000  0.000000  0.000000
## Castilla-La Mancha          0.000000  0.000000  0.000000  0.000000  0.000000
## Cataluna                    2.303488  2.302800  2.302732  2.302725  2.302724
## Comunidad Valenciana        0.000000  0.000000  0.000000  0.000000  0.000000
## Extremadura                 0.000000  0.000000  0.000000  0.000000  0.000000
## Galicia                     0.000000  0.000000  0.000000  0.000000  0.000000
## Madrid (Comunidad De)       9.596451  9.596410  9.596406  9.596406  9.596406
## Murcia (Region de)          0.000000  0.000000  0.000000  0.000000  0.000000
## Navarra (Comunidad Foral De) 8.763455  8.765012  8.765167  8.765183  8.765185
## Rioja (La)                  0.000000  0.000000  0.000000  0.000000  0.000000
```

Although $v_1(\varepsilon)$ and $v_2(\varepsilon)$ have the same limit for $\varepsilon \downarrow 0$, the corresponding limits of $W^*(v_1(\varepsilon))$ and $W^*(v_2(\varepsilon))$ are clearly distinct.

Thus, for $W^*$ and the outer objective function to be continuous on (the closure of) their domain, $v_1, \ldots, v_K$ must be bounded away from zero.

# C Why Using ipop as Inner Optimizer May Be Hazardous

We now illustrate that issues with the inner optimization arise in common applications when package **Synth**[1] is used with its default inner optimizer, `ipop` from package **kernlab**[2]. In particular, the example below shows that a not properly working numerical inner optimizier might significantly compromise the outer optimization: the example considers a case where for predictor weights $V$ close to the true optimum, the inner optimizer solves the inner optimization so badly that the wrong $\widetilde{W}^*(V)$ delivered as an approximation for $W^*(V)$ results in a much too large value for the outer objective function. In such a case, the outer optimizer may stay away from the true optimal predictor weights, because it is fooled by the results delivered by the inner optimizer.

As a first step, we calculate the synthetic control unit with function `synth` with an initialization of the random seed to `1` using `ipop` for the inner optimization (per default).

```
# run synth
set.seed(1);
synth.out <- synth(data.prep.obj = dataprep.out.Basque)

##
## X1, X0, Z1, Z0 all come directly from dataprep object.
##
##
## ****************
##  searching for synthetic control unit
##
##
## ****************
## ****************
## ****************
##
## MSPE (LOSS V): 0.008864629
##
## solution.v:
##  0.01556808 0.001791073 0.04417159 0.03409436 8.45034e-05 0.2009837 0.09484593 0.007689228 0.1339
##
## solution.w:
##  4.92e-08 5.17e-08 1.352e-07 2.85e-08 5.32e-08 5.177e-07 5.24e-08 7.29e-08 0.8507986 2.274e-07 4.
```

The final solution of the outer optimization obtained with package **Synth** corresponds to the results reported by Abadie and Gardeazabal (2003), the outer objective function for the "optimal" donor weights has a value of 0.008865.

If we let function `synth`/`ipop` solve the inner optimization task for the particular (fixed) vector of predictor weights given by

$$v := \left(10^{-8}, 10^{-8}, 10^{-8}, 10^{-8}, 8.5 \times 10^{-5}, 1, 10^{-8}, 10^{-8}, 10^{-8}, 10^{-8}, 10^{-8}, 5.5 \times 10^{-5}, 10^{-8}\right)' ,$$

we obtain:

[1] See Abadie et al. (2011).
[2] See Karatzoglou et al. (2004).

```r
# define custom predictor weights v
v <- c(1e-8,1e-8,1e-8,1e-8,8.5e-5,1,1e-8,1e-8,1e-8,1e-8,1e-8,5.5e-5,1e-8)
# run synth's inner optimization
synth2.out <- synth(data.prep.obj = dataprep.out.Basque, custom.v=v)

##
## X1, X0, Z1, Z0 all come directly from dataprep object.
##
##
## ****************
##  optimization over w weights: computing synthtic control unit
##
##
##
## ****************
##  v weights supplied manually: computing synthtic control unit
##
##
##
## ****************
## ****************
## ****************
##
## MSPE (LOSS V): 0.01085428
##
## solution.v:
##  1e-08 1e-08 1e-08 1e-08 8.49881e-05 0.9998599 1e-08 1e-08 1e-08 1e-08 1e-08 5.49923e-05 1e-08
##
## solution.w:
##  0.001585319 0.002379284 0.002474546 0.0001053374 0.002041706 0.003252474 0.001724444 0.001406568

print(as.numeric(synth2.out$loss.w))

## [1] 0.0001663998
```

The solution of the inner optimization produces donor weights $\widetilde{W}^*(v)$ which correspond to a value of $\widetilde{W}^*(v)'\widetilde{X}'V(v)\widetilde{X}\widetilde{W}^*(v) = 1.6639979 \times 10^{-4}$ for the inner objective function and a value of $\widetilde{W}^*(v)'\widetilde{Z}'\widetilde{Z}\widetilde{W}^*(v) = 0.0108543$ for the outer objective function.

However, solving the inner optimization with function `synth/ipop` failed for this particular vector of predictor weights. Using `MSCMT/WNNLS`, one obtains:

```r
# perform (only) inner optimization with WNNLS by using "fixed" v
sol <- mscmt(dataprep.out.Basque,outer.optim="fixed",outer.opar=list(v=v),
             std.v="max",verbose=FALSE)
print(sol$loss.v)

## [1] 0.004340111

print(sol$loss.w)

##        fixed
## 0.0001119264
```

`MSCMT/WNNLS`'s solution $W^*(v)$ of the inner optimization leads to a considerably reduced value of only $W^*(v)'\widetilde{X}'V(v)\widetilde{X}W^*(v) = 1.1192636 \times 10^{-4}$ for the inner objective function. This correct solution $W^*(v)$ of the inner optimization produces a value of

$W^*(v)'\widetilde{Z}'\widetilde{Z}W^*(v) = 0.0043401$ for the outer objective function: this is not only better compared to the value of $\widetilde{W}^*(v)'\widetilde{Z}'\widetilde{Z}\widetilde{W}^*(v) = 0.0108543$ corresponding to the (wrong) solution $\widetilde{W}^*(v)$, but also considerably better than the outer objective function value of 0.008865 reported by `synth/ipop`. So, the 'optimum' of the outer objective function delivered by `synth/ipop` is clearly suboptimal, leading to wrong donor weights of the synthetic control unit.

The vector $v$ of predictor weights was deliberately chosen near optimal predictor weights obtained with package **MSCMT**[3]. Whether suboptimal solutions of `ipop` for predictor weight vectors $v$ near the optimal solution have prevented the outer optimizer to move towards optimal predictor weights remains unclear, but the example indicates that suboptimal solutions of the inner optimization may – at least in parts – be responsible for suboptimal solutions of the outer optimization problem.

Put differently, for properly solving the outer optimization, it is mandatory that the inner optimization is solved with very small approximation errors only.

# D Reproducible Calculation of Empirical Results

## D.1 Synthetic Control Unit for the Basque Country

The following code shows that `synth` produces suboptimal results for the synthetic control unit of the Basque country irrespective of the choice of the inner (`ipop` or `LowRankQP`) and outer (default or `genoud`) optimizer (leaving all other function arguments at their defaults) by calling function `synth` with all four possible combinations of inner and outer optimizers:

```
# for the preparation of object dataprep.out.Basque see above!
# run synth with standard settings
set.seed(1)
synth.out <- synth(data.prep.obj = dataprep.out.Basque)

##
## X1, X0, Z1, Z0 all come directly from dataprep object.
##
##
## ****************
##  searching for synthetic control unit
##
##
## ****************
## ****************
## ****************
##
## MSPE (LOSS V): 0.008864629
##
## solution.v:
##  0.01556808 0.001791073 0.04417159 0.03409436 8.45034e-05 0.2009837 0.09484593 0.007689228 0.1339
##
## solution.w:
##  4.92e-08 5.17e-08 1.352e-07 2.85e-08 5.32e-08 5.177e-07 5.24e-08 7.29e-08 0.8507986 2.274e-07 4.

print(synth.out$loss.v)
```

[3]See Becker and Klößner (2017).

```
##             17
## 17 0.008864629

print(synth.out$solution.w)

##        w.weight
## 2  4.921274e-08
## 3  5.169827e-08
## 4  1.352267e-07
## 5  2.853779e-08
## 6  5.323209e-08
## 7  5.177271e-07
## 8  5.240707e-08
## 9  7.287595e-08
## 10 8.507986e-01
## 11 2.274436e-07
## 12 4.030868e-08
## 13 9.512086e-08
## 14 1.491998e-01
## 15 5.608373e-08
## 16 9.021814e-08
## 18 1.061039e-07

# run synth again with genoud=TRUE
# establish reproducibility
if (is.loaded("rgenoud")) devtools::unload(devtools::inst("rgenoud"))
set.seed(1)
# genoud is VERY verbose, so capture the output
tmp <- capture.output(suppressWarnings(
  synth2.out <- synth(data.prep.obj = dataprep.out.Basque, genoud=TRUE)))

## Loading required package:  rgenoud
## ##  rgenoud (Version 5.7-12.4, Build Date:  2015-07-19)
## ##  See http://sekhon.berkeley.edu/rgenoud for additional documentation.
## ##  Please cite software as:
## ##  Walter Mebane, Jr.  and Jasjeet S. Sekhon.  2011.
## ##  ``Genetic Optimization Using Derivatives:  The rgenoud package for R.''
## ##  Journal of Statistical Software, 42(11):  1-26.
## ##

print(synth2.out$loss.v)

##             17
## 17 0.008864553

# run synth again with quadopt="LowRankQP"
# establish reproducibility
set.seed(1)
# LowRankQP is VERY verbose, so capture the output
library(LowRankQP)

## LowRankQP 1.0 loaded
## Copyright J.T. Ormerod & M. P. Wand 2005

tmp <- capture.output(suppressWarnings(
  synth3.out <- synth(data.prep.obj = dataprep.out.Basque, quadopt="LowRankQP")))
print(synth3.out$loss.v)
```

```
##             17
## 17 0.008864545

# run synth again with genoud=TRUE and quadopt="LowRankQP"
# establish reproducibility
if (is.loaded("rgenoud")) devtools::unload(devtools::inst("rgenoud"))
set.seed(1)
# genoud and LowRankQP are VERY verbose, so capture the output
tmp <- capture.output(suppressWarnings(
  synth4.out <- synth(data.prep.obj = dataprep.out.Basque, genoud=TRUE,
                      quadopt="LowRankQP")))

## Loading required package:  rgenoud
## ##  rgenoud (Version 5.7-12.4, Build Date:  2015-07-19)
## ##  See http://sekhon.berkeley.edu/rgenoud for additional documentation.
## ##  Please cite software as:
## ##  Walter Mebane, Jr.  and Jasjeet S. Sekhon.  2011.
## ##  ``Genetic Optimization Using Derivatives:  The rgenoud package for R.''
## ##  Journal of Statistical Software, 42(11):  1-26.
## ##

print(synth4.out$loss.v)

##             17
## 17 0.008864545
```

Now, we repeat the estimation with **mscmt** to obtain a considerably improved result of the outer optimization:

```
# repeat estimation with mscmt
sol <- mscmt(dataprep.out.Basque,outer.optim="DEoptC",seed=1)

## 10:50:12: Number of 'sunny' donors: 16 out of 16
## 10:50:12: Unrestricted outer optimum (obtained by ignoring all predictors) with
## 10:50:12: RMSPE 0.064236669716524 and MSPE (loss v) 0.0041263497362698 is INFEASIBLE
## 10:50:12: when respecting the predictors.
## 10:50:12: Starting optimization via DEoptC, random seed 1.
## 10:50:21: Optimization finished (1130881 calls to inner optimizer), rmspe:
## 10:50:21: 0.0654680949292754, mspe: 0.00428607145366861.
## Final rmspe: 0.06546809, mspe (loss v): 0.004286071
## Optimal weights:
##      Baleares (Islas)             Cataluna Madrid (Comunidad De)
##             0.2192728            0.6327857             0.1479414
```

## D.2 Synthetic Control Unit for Catalonia

The following code shows that **synth** produces suboptimal results for the synthetic control unit of Catalonia irrespective of the choice of the inner (**ipop** or **LowRankQP**) and outer (default or **genoud**) optimizer (leaving all other function arguments at their defaults) by calling function **synth** with all four possible combinations of inner and outer optimizers:

```
# for the preparation of object dataprep.out.Catalonia see above!
# run synth with standard settings
set.seed(1)
synth.out <- synth(data.prep.obj = dataprep.out.Catalonia)
```

```
##
## X1, X0, Z1, Z0 all come directly from dataprep object.
##
##
## ****************
##  searching for synthetic control unit
##
##
## ****************
## ****************
## ****************
##
## MSPE (LOSS V): 0.0003093978
##
## solution.v:
##  0.01405653 0.01342696 0.005037431 0.001792396 0.07939944 0.5166658 0.281674 0.08745683 9.4099e-06
##
## solution.w:
##  1.09272e-05 7.4858e-06 0.03591013 0.2715622 1.49049e-05 0.2574583 3.0584e-06 2.667e-06 2.53151e-0

print(synth.out$loss.v)

##              10
## 10 0.0003093978

# run synth again with genoud=TRUE
# establish reproducibility
if (is.loaded("rgenoud")) devtools::unload(devtools::inst("rgenoud"))
set.seed(1)
# genoud is VERY verbose, so capture the output
tmp <- capture.output(suppressWarnings(
  synth2.out <- synth(data.prep.obj = dataprep.out.Catalonia, genoud=TRUE)))

## Loading required package:  rgenoud
## ##  rgenoud (Version 5.7-12.4, Build Date:  2015-07-19)
## ##  See http://sekhon.berkeley.edu/rgenoud for additional documentation.
## ##  Please cite software as:
## ##  Walter Mebane, Jr.   and Jasjeet S. Sekhon.  2011.
## ##  ``Genetic Optimization Using Derivatives:  The rgenoud package for R.''
## ##  Journal of Statistical Software, 42(11):  1-26.
## ##

print(synth2.out$loss.v)

##              10
## 10 0.0003029811

print(synth2.out$solution.w)

##        w.weight
## 2  5.082267e-07
## 3  3.318236e-07
## 4  1.942877e-02
## 5  2.741967e-01
## 6  4.250162e-06
## 7  2.404876e-01
## 8  1.675238e-07
## 9  1.816361e-07
```

```
## 11 1.581297e-06
## 12 1.346784e-07
## 13 2.447532e-07
## 14 4.427370e-01
## 15 2.314220e-02
## 16 1.552468e-07
## 18 1.367770e-07

# run synth again with quadopt="LowRankQP"
# establish reproducibility
set.seed(1)
# LowRankQP is VERY verbose, so capture the output
library(LowRankQP)
tmp <- capture.output(suppressWarnings(
  synth3.out <- synth(data.prep.obj = dataprep.out.Catalonia, quadopt="LowRankQP")))
print(synth3.out$loss.v)

##              10
## 10 0.0003097263

# run synth again with genoud=TRUE and quadopt="LowRankQP"
# establish reproducibility
if (is.loaded("rgenoud")) devtools::unload(devtools::inst("rgenoud"))
set.seed(1)
# genoud and LowRankQP are VERY verbose, so capture the output
tmp <- capture.output(suppressWarnings(
  synth4.out <- synth(data.prep.obj = dataprep.out.Catalonia, genoud=TRUE,
                      quadopt="LowRankQP")))

## Loading required package:  rgenoud
## ##  rgenoud (Version 5.7-12.4, Build Date:  2015-07-19)
## ##  See http://sekhon.berkeley.edu/rgenoud for additional documentation.
## ##  Please cite software as:
## ##  Walter Mebane, Jr.  and Jasjeet S. Sekhon.  2011.
## ##  ``Genetic Optimization Using Derivatives:  The rgenoud package for R.''
## ##  Journal of Statistical Software, 42(11):  1-26.
## ##

print(synth4.out$loss.v)

##              10
## 10 0.0003097735
```

Now, we repeat the estimation with `mscmt` to obtain a considerably improved result of the outer optimization:

```
# repeat estimation with mscmt
sol <- mscmt(dataprep.out.Catalonia,outer.optim="DEoptC",seed=1)

## 10:54:02: Number of 'sunny' donors: 15 out of 15
## 10:54:02: Unrestricted outer optimum (obtained by ignoring all predictors) with
## 10:54:02: RMSPE 0.00894429277703889 and MSPE (loss v) 8.00003732813901e-05 is
## 10:54:02: INFEASIBLE when respecting the predictors.
## 10:54:02: Starting optimization via DEoptC, random seed 1.
## 10:54:14: Optimization finished (1274881 calls to inner optimizer), rmspe:
## 10:54:14: 0.00897426922820976, mspe: 8.05375081803927e-05.
## Final rmspe: 0.008974269, mspe (loss v): 8.053751e-05
```

```
## Optimal weights:
##            Baleares (Islas)        Madrid (Comunidad De)
##                   0.2324732                    0.4378377
## Navarra (Comunidad Foral De)
##                   0.3296891
```

# E Settings for Outer Optimizers

The default parameters for the outer optimizers are documented in function `mscmt` of `R` package **MSCMT**. Table 1 lists the deviations from these defaults which were necessary to adjust the mean computing time to about 20 seconds in the benchmark study for the Basque Country (as treated unit).

| Optimizer | Non-default Parameters |
|---|---|
| `cmaes` | `maxit=4000`, `lambda=20`, `mu=10` |
| `crs` | `maxeval=26666` |
| `DEoptC` | `nG=2000` |
| `DEoptim` | `itermax=114`, `reltol=1e-14`, `steptol=22` |
| `genoud` | `max.generations=20`, `wait.generations=6`, `pop.size=1000` |
| `GenSA` | `max.time=1.5385` |
| `hydroPSO` | `maxit=256`, `npart=40` |
| `isres` | `maxeval=22000` |
| `malschains` | `maxEvals=26000` |
| `optim` | `nrandom=23` |
| `psoptim` | `maxit=660`, `reltol=1e-14` |
| `soma` | `nMigrations=90`, `minRelativeSep=1e-14` |

Table 1: Non-default parameters in benchmark study of outer optimizers for the Basque Country as treated unit.

Table 2 contains the non-default settings which were necessary to adjust the mean computing time to about 20 seconds in the benchmark study for Catalonia (as treated unit).

| Optimizer | Non-default Parameters |
|---|---|
| cmaes | maxit=1600, lambda=10 |
| crs | maxeval=26666 |
| DEopt | nG=106 |
| DEoptC | nG=1000, waitgen=90 |
| DEoptim | itermax=114, reltol=1e-14, steptol=22 |
| genoud | max.generations=19, wait.generations=6, pop.size=1000 |
| GenSA | max.time=1.5385 |
| hydroPSO | maxit=240, npart=40 |
| isres | maxeval=22000 |
| malschains | maxEvals=26000 |
| nlminb | nrandom=20 |
| optim | nrandom=16 |
| psoptim | maxit=660, reltol=1e-14 |
| soma | nMigrations=90, minRelativeSep=1e-14 |

Table 2: Non-default parameters in benchmark study of outer optimizers for Catalonia as treated unit.

# References


**Abadie, Alberto, Alexis Diamond, and Jens Hainmueller**, "Synth: An R Package for Synthetic Control Methods in Comparative Case Studies," *Journal of Statistical Software*, 6 2011, *42* (13), 1–17.

_ **and Javier Gardeazabal**, "The Economic Costs of Conflict: A Case Study of the Basque Country," *The American Economic Review*, 2003, *93* (1), 113–132.

**Becker, Martin and Stefan Klößner**, *MSCMT: Multivariate Synthetic Control Method Using Time Series* 2017. R package version 1.2.0.

**Hanson, Richard J. and Karen H. Haskell**, "Algorithm 587: Two Algorithms for the Linearly Constrained Least Squares Problem," *ACM Trans. Math. Softw.*, September 1982, *8* (3), 323–333.

**Haskell, Karen H. and Richard J. Hanson**, "An algorithm for linear least squares problems with equality and nonnegativity constraints," *Mathematical Programming*, 1981, *21* (1), 98–118.

**Karatzoglou, Alexandros, Alex Smola, Kurt Hornik, and Achim Zeileis**, "kernlab – An S4 Package for Kernel Methods in R," *Journal of Statistical Software*, 2004, *11* (9), 1–20.