Regression to the Mean's Impact on the Synthetic Control Method: Bias and Sensitivity Analysis

Nicholas Illenberger, Dylan S. Small, Pamela A. Shaw

Getting Started

This pdf was written with the goal of helping users replicate the results of "Regression to the Mean's Impact on the Synthetic Control Method: Bias and Sensitivity Analysis". Functions required to run the analysis can be installed using the `install_github()` function from the `devtools` package.

```
install_github("nillen0/SynthRTM",
               auth_token = "6d89bd6db7f64a1fec957f5a793f11b952851488")
```

Once installed, load the functions as normal using the `library()` function.

We provide a quick run-through of how the simulations work. A sample dataset can be simulated using the `SimulateData()` function. This function allows us to easily set the mean and standard deviation of measurements in the control and treatment group, the number of control units, the AR(1) coefficient, and the number of follow-up timepoints.

```
Data = SynthRTM::SimulateData(mu0 = 0, sd0 = 1, n0 = 40,
                    mu1 = 2, sd1 = 1,
                    rho = 0.9, t = 8, t0 = 5, theta = 0)
```

This function outputs a dataset in long format. That is, for each unit our dataframe will have t rows, corresponding to the first, to t^{th} observation for that unit.

Analyse Data

Once we have data, we need to fit weights for the synthetic control. We provide the `GetWeights()` function for this purpose. When we have no predictors and are making weights based off of the outcome, we put the name of the outcome variable in both the `Outcome` and `Predictors` arguments.

```
Weights = GetWeights(Data = Data,
           Outcome = "Outcome",
           Predictors = "Outcome",
           TrtID = 41,
           ContID = 1:40,
           t0 = 5, t = 8)
```

Using these weights we can obtain an estimate of the Average treatment effect on the treated (ATT).

```
GetSC(Data = Data, ContID = 1:40, t = 8, weights = Weights)
```

Replication of Tables/Figures

For all simulation examples, results for different settings can be obtained by varying the levels of `mu1` (the mean of the treated units), `rho` (the correlation between repeat observations), and `theta` (the true treatment effect)

Table 1

```
set.seed(807)
out1 = sapply(c(1.0, 2.0, 3.0, 4.0, 5.0), function(m) {
  resultsU = resultsSC = resultsNN1 = resultsNN2 = matrix(nrow = 1000, ncol = 2)
  for (i in 1:1000) {
  Data = SynthRTM::SimulateData(mu0 = 0, sd0 = 1, n0 = 40,
                    mu1 = m, sd1 = 1, rho = 0.5,
                    t = 8, t0 = 5, theta = 0)
  resultsU[i,] = DID(method = "Unmatched",
                     Data = Data,
                     ContID = 1:40,
                     TrtID = 41,
                     t0 = 5,
                     t = 8,
                     Permutation = T)
  resultsSC[i,] = DID(method = "SC",
                      Data = Data,
                      ContID = 1:40,
                      TrtID = 41,
                      t0 = 5,
                      t = 8,
                      Permutation = T)
  resultsNN1[i,] = DID(method = "NN",
                       Data = Data,
                       ContID = 1:40,
                       TrtID = 41,
                       t0 = 5,
                       t = 8,
                       ProcedureNN = 1,
                       Permutation = T)
  resultsNN2[i,] = DID(method = "NN",
                       Data = Data,
                       ContID = 1:40,
                       TrtID = 41,
                       t0 = 5,
                       t = 8,
                       ProcedureNN = 2,
                       Permutation = T)
}
c(mean(resultsU[,2] <= 0.05),
  mean(resultsSC[,2]<= 0.05),
  mean(resultsNN1[,2]<= 0.05),
  mean(resultsNN2[,2]<= 0.05))
})

t(out1)

set.seed(807)
out2 = sapply(c(0.0, 0.25, 0.5, 0.75, 0.9), function(m) {
  resultsU = resultsSC = resultsNN1 = resultsNN2 = matrix(nrow = 1000, ncol = 2)
  for (i in 1:1000) {
  Data = SynthRTM::SimulateData(mu0 = 0, sd0 = 1, n0 = 40,
```

```
                    mu1 = 5, sd1 = 1, rho = m,
                    t = 8, t0 = 5, theta = 0)
  resultsU[i,] = DID(method = "Unmatched",
                     Data = Data,
                     ContID = 1:40,
                     TrtID = 41,
                     t0 = 5,
                     t = 8,
                     Permutation = T)
  resultsSC[i,] = DID(method = "SC",
                      Data = Data,
                      ContID = 1:40,
                      TrtID = 41,
                      t0 = 5,
                      t = 8,
                      Permutation = T)
  resultsNN1[i,] = DID(method = "NN",
                       Data = Data,
                       ContID = 1:40,
                       TrtID = 41,
                       t0 = 5,
                       t = 8,
                       ProcedureNN = 1,
                       Permutation = T)
  resultsNN2[i,] = DID(method = "NN",
                       Data = Data,
                       ContID = 1:40,
                       TrtID = 41,
                       t0 = 5,
                       t = 8,
                       ProcedureNN = 2,
                       Permutation = T)
}
c(mean(resultsU[,2] <= 0.05),
  mean(resultsSC[,2]<= 0.05),
  mean(resultsNN1[,2]<= 0.05),
  mean(resultsNN2[,2]<= 0.05))
})

t(out2)
```

Figure 1

```
out = sapply(seq(0, -1.5, length.out = 8), function(th) {
  resultsU = resultsSC = resultsNN1 = resultsNN2 = matrix(nrow = 1000, ncol = 2)
  for (i in 1:1000) {
  Data = SynthRTM::SimulateData(mu0 = 0, sd0 = 1, n0 = 40,
                    mu1 = 2, sd1 = 1, rho = 0.5,
                    t = 8, t0 = 5, theta = th)
  resultsU[i,] = DID(method = "Unmatched",
                     Data = Data,
                     ContID = 1:40,
```

```
                      TrtID = 41,
                      t0 = 5,
                      t = 8,
                      Permutation = T)
  resultsSC[i,] = DID(method = "SC",
                       Data = Data,
                       ContID = 1:40,
                       TrtID = 41,
                       t0 = 5,
                       t = 8,
                       Permutation = T)
  resultsNN1[i,] = DID(method = "NN",
                        Data = Data,
                        ContID = 1:40,
                        TrtID = 41,
                        t0 = 5,
                        t = 8,
                        ProcedureNN = 1,
                        Permutation = T)
  resultsNN2[i,] = DID(method = "NN",
                        Data = Data,
                        ContID = 1:40,
                        TrtID = 41,
                        t0 = 5,
                        t = 8,
                        ProcedureNN = 2,
                        Permutation = T)
}

c(mean(resultsU[,2] <= 0.05),
  mean(resultsSC[,2]<= 0.05),
  mean(resultsNN1[,2]<= 0.05),
  mean(resultsNN2[,2]<= 0.05))
})


colnames(out) = paste("theta", round(seq(0, -1.5, length.out = 8), 2), sep = "_")
rownames(out) = c("Unmatched", "SC", "NN1", "NN2")
matplot(t(out), type = "l", col = "black", lwd = 2, frame.plot = F)
legend("topleft", inset=0.01, legend=rownames(out), lty =1:4,
bg= ("white"), horiz=F, lwd = 2)


# Figure 1
matplot(t(out), type = "l", col = "black", lwd = 2, frame.plot = F, ylim= c(0, 1),
        ylab = "Pr(Reject")
legend("topleft", inset=0.02, legend=rownames(out), lty =1:4,
bg= ("white"), horiz=F, lwd = 2)
```

Table 3

```
####-- Required packages --####
library(MASS)
library(Synth)
library(doParallel)
library(dplyr)

####-- Initialize --####
NSIM = 1000
N0 = 40
N1 = 1
N = N0 + N1
mu1 = 0.5
mu0 = 0
rho = 0.5
sd1 = sd0 = 1
t = 8
cl = makeCluster(4)

####-- Synthetic Control t=8 --####

t0 = t1 = 2
t = t0 + t1
sd1 = sd0 = 1
rho = 0.75  # change for different simulation settings
powers.matrix = abs(outer(1:t, 1:t, "-"))
sigma1 = sd1^2 * rho^powers.matrix
sigma0 = sd0^2 * rho^powers.matrix
cl = makeCluster(4)


results = matrix(nrow = NSIM, ncol = 2)
set.seed(12342)
for(s in 1:NSIM){
  id = rep(1:N, each = t)
  trt = rep(c(0,1), times = c(N0*t, N1*t))
  period = rep(c(0, 1), each = t0, times = N)
  time = rep(1:t, times = N)
  Y = NULL
  Y[trt == 0] = as.vector(t(mvrnorm(n = N0, mu = rep(mu0, times = t),
                                    Sigma = sigma0)))
  Y[trt == 1] = as.vector(t(mvrnorm(n = N1, mu = rep(mu1, times = t),
                                    Sigma = sigma1)))

  SimDat = data.frame(id, trt, period, time, Y)
  SimDat = SimDat %>%
    mutate(Y_new = Y, mu = ifelse(trt == 0, mu0, mu1),
           sdy = ifelse(trt == 0, sd0, sd1))
  SimDat[time %in% 3:4, "Y_new"] =
    as.vector(unlist(t(SimDat %>%
                         filter(time == 2) %>%
                         mutate(rtm_1 = rho*((Y - mu)/sdy) + mu,
```

```
                                rtm_2 = rho*((rtm_1 - mu)) + mu) %>%
                        dplyr::select(rtm_1,  rtm_2))))


  registerDoParallel(cl)
  SynDIDs = foreach(i=1:N, .packages = c("Synth", "dplyr")) %dopar% {
    prep = dataprep(
      foo = SimDat,
      predictors = c("Y") ,
      predictors.op = "mean" ,
      time.predictors.prior = 1:(t/2),
      dependent = "Y",
      unit.variable = "id",
      time.variable = "time",
      treatment.identifier = i,
      controls.identifier = (1:N)[-i],
      time.optimize.ssr = 1:(t/2),
      time.plot = 1:t
    )
    syn = synth(prep)
    dif_vector = prep$Y1plot - prep$Y0plot %*% syn$solution.w
    total_DID = mean(dif_vector[1:2]) - mean(dif_vector[3:4])
    trt_rtm = SimDat %>% filter(trt == 1) %>% dplyr::select(Y_new)
    cont_rtm = matrix(unlist(SimDat %>% filter(trt == 0) %>% dplyr::select(Y_new)),
                      nrow = t, ncol = N0)
    dif_rtm = trt_rtm - cont_rtm %*% syn$solution.w
    rtm_DID = mean(dif_rtm[1:2,1]) - mean(dif_rtm[3:4,1])
    total_DID - rtm_DID
  }
  adjusted_DID = unlist(SynDIDs)

  results[s,1] = adjusted_DID[N]
  results[s,2] = mean(abs(adjusted_DID[N]) <= abs(adjusted_DID))
}

mean(results[,2]<0.05)
# rho = 0.25, 0.049
# rho = 0.50, 0.051
# rho = 0.75, 0.055



####-- See how this works for t-distribution --####

rmvt = function(n, m, nu, mu, sigma) {
  vals = mvrnorm(n = n, mu = rep(0, times = m),
                 Sigma = sigma)
  prob.vals = pnorm(vals, mean = 0, sd = sigma[1,1])
  t.vals = qt(prob.vals, df = nu) + mu
  return(t.vals)
}
```

```r
# Change these for different simulation settings:
rho = 0.75
df = 5

t0 = t1 = 2
t = t0 + t1
mu0 = 0; mu1 = 5
sd0 = sd1 = 1
powers.matrix = abs(outer(1:t, 1:t, "-"))
sigma1 = sd1^2 * rho^powers.matrix
sigma0 = sd0^2 * rho^powers.matrix
set.seed(12342)
SynRes = matrix(nrow = 2, ncol = 2000)
for(s in 1:10){

  # Create Dataset:
  id = rep(1:N, each = t)
  trt = rep(c(0,1), times = c(N0*t, N1*t))
  period = rep(c(0, 1), each = t0, times = N)
  time = rep(1:t, times = N)
  Y = NULL
  Y[trt == 0] = as.vector(t(rmvt(n = N0, m = t, mu = mu0, nu = df, sigma = sigma0)))
  Y[trt == 1] = as.vector(t(rmvt(n = N1, m = t, mu = mu1, nu = df, sigma = sigma1)))
  SimDat = data.frame(id, trt, period, time, Y)


  # Augment to include expected values under RTM
  SimDat = SimDat %>%
    mutate(Y_new = Y,
           mu = ifelse(trt == 0, mu0, mu1),
           sdy = ifelse(trt == 0, df/(df-2), df/(df-2)))
  SimDat[time %in% 3:4, "Y_new"] =
    as.vector(unlist(t(SimDat %>%
                         filter(time == 2) %>%
                         mutate(rtm_1 = rho*((Y - mu)/sdy) + mu,
                                rtm_2 = rho*((rtm_1 - mu)/sdy) + mu) %>%
                         dplyr::select(rtm_1,  rtm_2))))


  registerDoParallel(cl)
  SynDIDs = foreach(i=1:N, .packages = c("Synth", "dplyr")) %dopar% {
    prep = dataprep(
      foo = SimDat,
      predictors = c("Y") ,
      predictors.op = "mean" ,
      time.predictors.prior = 1:(t/2),
      dependent = "Y",
      unit.variable = "id",
      time.variable = "time",
      treatment.identifier = i,
      controls.identifier = (1:N)[-i],
      time.optimize.ssr = 1:(t/2),
      time.plot = 1:t
```

```
    )
    syn = synth(prep)
    dif_vector = prep$Y1plot - prep$Y0plot %*% syn$solution.w
    total_DID = mean(dif_vector[1:2]) - mean(dif_vector[3:4])
    trt_rtm = SimDat %>% filter(trt == 1) %>% dplyr::select(Y_new)
    cont_rtm = matrix(unlist(SimDat %>% filter(trt == 0) %>% dplyr::select(Y_new)),
                      nrow = t, ncol = N0)
    dif_rtm = trt_rtm - cont_rtm %*% syn$solution.w
    rtm_DID = mean(dif_rtm[1:2,1]) - mean(dif_rtm[3:4,1])
    total_DID - rtm_DID
  }
  adjusted_DID = unlist(SynDIDs)

  SynRes[1,s] = adjusted_DID[N]
  SynRes[2,s] = mean(abs(adjusted_DID[N]) <= abs(adjusted_DID))
}


mean(SynRes[2,]<0.05)
# rho = 0.25, 0.049
# rho = 0.50, 0.051

# df = 50,  0.05;  0.052; 0.055
# df = 10,  0.059; 0.049; 0.078
# df = 5,   0.078; 0.052; 0.123
```

Smoking Cesseation Analysis

The data for this analysis can be downloaded from "http://fmwww.bc.edu/repec/bocode/s/synth_smoking.dta"

```
## Re-analysis of Smoking Cessation Data

## Re-analysis of Smoking Cessation Data

####-- Required functions / packages --####

library(dplyr)
library(foreign)
library(geepack)
library(Synth)
library(tidyverse)


####-- Read in and format data --####
smoking <- read.dta("~/Dropbox/Upenn/Research/Masters/synth_smoking.dta")


# Creates an id number and treatment indicator
smoke.work <- smoking %>%
  mutate( id = as.numeric(state),
          state = as.character(state),
          treatment = (state == "California"),
```

```
          year = year - min(year)) %>%
  arrange(id)

# Format variables to have the predictors used by Abadie et al.
vars.of.interest <- smoke.work %>%
  filter(year %in% 10:18) %>%
  group_by(state) %>%
  mutate(lnincome = mean(lnincome),
         age15to24 = mean(age15to24),
         retprice = mean(retprice)) %>%
  filter(year %in% 14:18) %>%
  mutate(beer = mean(beer)) %>%
  filter(year == 14)

# Append the reformatted variables:
smoke.work <- smoke.work %>%
  mutate(lnincome = rep(vars.of.interest$lnincome, each = 31),
         age15to24 = rep(vars.of.interest$age15to24, each = 31),
         beer = rep(vars.of.interest$beer, each = 31),
         retprice = rep(vars.of.interest$retprice, each = 31)) %>%
  group_by(id) %>%
  mutate(cigsale.1975 = cigsale[year == 5],
         cigsale.1980 = cigsale[year == 10],
         cigsale.1988 = cigsale[year == 18]) %>%
  ungroup(id)


####-- Show plots of smoking consumption over time for all states -- ####
smoke.plot <-  ggplot(data = smoke.work %>% filter(state != "California"), aes(x = year))+
  geom_line(aes(y = cigsale, group = state), colour = "gray80")

smoke.plot <-  smoke.plot +
  geom_line(data = smoke.work %>% filter(state == "California"),
            aes(y = cigsale, group = state), colour = "black")

smoke.plot +
  labs(x = 'year', y = 'tobacco consumption') +
  geom_vline(xintercept = 18, col = 'red', lty = 2)

####-- Find expected values under no treatment effect, just RTM --####

# Under the parallel trends assumption, mean trends in the outcome are the same in
# treated units and control units. Thus, we fit a gee without time:treatment interaction
# to determine the expected mean line

fit.gee.ar1 <-  geeglm(data = smoke.work,
                       formula = cigsale ~ year + lnincome + beer +
                         age15to24 + retprice + cigsale.1988 + cigsale.1980 + cigsale.1975 +
                         year:retprice + year:lnincome + year:beer + year:age15to24 +
                         year:cigsale.1988 + year:cigsale.1980 + year:cigsale.1975,
                       id= as.factor(state),
                       corstr = "ar1")
```

```r
# Look at residuals to determine if normality is plausible
# If so, then our correction may help!
resid <- smoke.work$cigsale  - fit.gee.ar1$fitted.values
resid.sd <- sqrt(fit.gee.ar1$geese$gamma)
resid.corr <- fit.gee.ar1$geese$alpha
standardized.resid.ar1 <- (resid - mean(resid))/resid.sd

qqnorm(y = standardized.resid.ar1)
qqline( y = standardized.resid.ar1)

# The distribution looks heavy-tailed, but roughly normal
# (at least, normal enough to check what the correction does)
smoke.work <- smoke.work %>%
  mutate(fitted = fit.gee.ar1$fitted.values,
         period = as.numeric(year > 18),
         time.from.trt = ifelse(year <= 18, 0, year - 18),
         resid = cigsale - fitted) %>%
  group_by(id) %>%
  mutate(diff.at.trt = resid[year == 18]) %>%
  ungroup(id)


smoke.work <- smoke.work %>%
  mutate(corr.from.trt = resid.corr^(time.from.trt),
         expect.diff = corr.from.trt*diff.at.trt,
         expect.val = ifelse(year > 18, expect.diff + fitted, cigsale))


smoke.work %>%
  filter(as.numeric(as.factor(state)) %in% c(3, sample((1:39)[-3], 1))) %>%
  ggplot(aes(x = year, y = cigsale, group = state)) +
  geom_line(col = "gray") +
  geom_line(aes(y = fitted)) +
  labs(y = 'Cigarette sales') +
  geom_vline(xintercept=18, col = 'red', lty = 2)

####-- Perform Synthetic Control --####

smoke.work <- smoke.work %>% as.data.frame
dids <- sapply(1:39, function(x) {
  prep <- dataprep(
    foo = smoke.work,
    special.predictors = list(
      list( "lnincome",           10:18, "mean"),
      list("age15to24",           10:18, "mean"),
      list( "retprice",           10:18, "mean"),
      list(     "beer",           14:18, "mean"),
      list(  "cigsale",                18, "mean"),
      list(  "cigsale",                10, "mean"),
      list(  "cigsale",                5, "mean")
    ),
    predictors.op = "mean" ,
    time.predictors.prior = c(0:18),
```

```
    dependent = "cigsale",
    unit.variable = "id",
    unit.names.variable = "state",
    time.variable = "year",
    treatment.identifier = x,
    controls.identifier = (1:39)[-x],
    time.optimize.ssr = c(0:18),
    time.plot = 0:30
  )

  syn <- synth(prep)

  diff.pre <- (prep$Y1plot - prep$Y0plot %*% syn$solution.w)[1:19]
  obs.diff.post <- (prep$Y1plot - prep$Y0plot %*% syn$solution.w)[20:31]

  exp.Y1 <- smoke.work$expect.val[smoke.work$id == x]

  exp.pre <- matrix(unlist(smoke.work %>% filter(period == 0 & id != x) %>% dplyr::select(cigsale)),
                    ncol = (39 - 1), nrow = 19)

  exp.post <- matrix(unlist(smoke.work %>% filter(period == 1 & id != x) %>% dplyr::select(expect.val)),
                     ncol = (39 - 1), nrow = 31-19)

  exp.Y0 <- rbind(exp.pre, exp.post)

  exp.diff.post <- (exp.Y1 - exp.Y0 %*% syn$solution.w)[20:31]

  adjusted.did <- mean(diff.pre) - (mean(obs.diff.post) - mean(exp.diff.post))
  unadjust.did <- mean(diff.pre) - mean(obs.diff.post)
  MSPE.pre <- mean(diff.pre^2)
  MSPE.post <- mean(obs.diff.post^2)
  MSPE.post.adj <- mean((obs.diff.post - exp.diff.post)^2)
  diff.in.2000 <- obs.diff.post[11]


  return(c(adjusted.did, MSPE.pre, MSPE.post, MSPE.post.adj, diff.in.2000, unadjust.did))
})



# unadjusted results
mean(abs(dids[6, 3]) <= abs(dids[6,])) # 2/39 = 0.103
dids[6,3] # 18.9

#p-value, removing poor fits:
cols <- which(2*dids[2,3] >= dids[2,])
mean(abs(dids[6, 3]) <= abs(dids[6,cols])) # 0.05

# Adjusted results
mean(abs(dids[1, 3]) <= abs(dids[1,])) # 2/39 = 0.154
dids[1,3] # 12.1
mean(abs(dids[1, 3]) <= abs(dids[1,cols])) # 2/39 = 0.154
```

```
####-- Find value for which if mean model for treated is shifted down, significant effect is erased -- #


# Recalculate fitted
delta <- 5
smoke.work <- smoke.work %>%
  mutate(new.fitted = ifelse(id == 3, fitted + delta, fitted),
         new.resid = cigsale - new.fitted) %>%
  group_by(id) %>%
  mutate(new.diff.at.trt = new.resid[year == 18]) %>%
  ungroup(id) %>%
  mutate(corr.from.trt = resid.corr^(time.from.trt),
         expect.diff = corr.from.trt*new.diff.at.trt,
         expect.val = expect.diff + new.fitted) %>%
  data.frame


dids = sapply(1:39, function(x) {
  prep = dataprep(
    foo = smoke.work,
    special.predictors = list(
      list( "lnincome",           10:18, "mean"),
      list("age15to24",           10:18, "mean"),
      list( "retprice",           10:18, "mean"),
      list(     "beer",           14:18, "mean"),
      list(  "cigsale",                18, "mean"),
      list(  "cigsale",                10, "mean"),
      list(  "cigsale",                5, "mean")
    ),
    predictors.op = "mean" ,
    time.predictors.prior = c(0:18),
    dependent = "cigsale",
    unit.variable = "id",
    unit.names.variable = "state",
    time.variable = "year",
    treatment.identifier = x,
    controls.identifier = (1:39)[-x],
    time.optimize.ssr = c(0:18),
    time.plot = 0:30
  )

  syn = synth(prep)

  diff.pre = (prep$Y1plot - prep$Y0plot %*% syn$solution.w)[1:19]
  obs.diff.post = (prep$Y1plot - prep$Y0plot %*% syn$solution.w)[20:31]

  exp.Y1 = c(smoke.work$cigsale[smoke.work$id == x][1:19], smoke.work$expect.val[smoke.work$id == x][20

  exp.pre = matrix(unlist(smoke.work %>% filter(period == 0 & id != x) %>% dplyr::select(cigsale)),
                   ncol = (39 - 1), nrow = 19)

  exp.post = matrix(unlist(smoke.work %>% filter(period == 1 & id != x) %>% dplyr::select(expect.val)),
                    ncol = (39 - 1), nrow = 31-19)
```

```
  exp.Y0 = rbind(exp.pre, exp.post)

  exp.diff.post = (exp.Y1 - exp.Y0 %*% syn$solution.w)[20:31]

  adjusted.did = mean(diff.pre) - (mean(obs.diff.post) - mean(exp.diff.post))

  MSPE = mean(diff.pre^2)

  return(c(adjusted.did, MSPE))
})

# delta = 0:
mean(abs(dids[1,]) >= abs(dids[1,3])) # 0.154
dids[1,3] # 12.1
mean(abs(dids[1,cols]) >= abs(dids[1,3])) # 0.1

# delta = -1; 0.154, 11.3, 0.1
# delta = -5; 0.256, 8.14, 0.3
# delta = -10; 0.667, 4.18, 0.65
# delta = 1; 0.128, 12.9, 0.05
# delta = 5; 0.128, 16.0, 0.05
# delta = 10; 0.103, 20.0, 0.05
```