```
#Script for RDD in time modelling for time period DST models, with specification checks on:
  #Autocorrelation of error term up to lag 10, Newey-West errors applied if present
  #Heteroskedasticity, HC3 errors applied if present
  #Specification checks of polynomial order of forcing variable (i.e. time); BIC is used to judge performance though
other indicators also output
  #Specification checks by varying bandwidth for linear models

#attach packages
library(data.table)
library(DescTools)
library(rdrobust)
library(sandwich)
library(lmtest)

#set working directory
setwd("~/Documents/DST")

#Define sequences of data sets
  #The STATS19 data have been segmented into separate files representing 7 easting and 13 northing bands and 1
aggregate data set (21 files in total)
  #Datasets are named with same text prefix "data_", different numerical suffix "i"
  #We define the sequence of datasets as seq_i, which refers to the 21 segmented datasets described above
  #We define seq_t as the sequence of time periods ranging from 1 to 5
  #The functions will iterate through all sequences and output results tables with all models' results
seq_i<-c(1:21)
seq_t<-c(1:5)

#Define functions:
  #inputparams function calculates optimal bandwidth from rdrobust package; outputs from this are used in subsequent
polynomial and bandwidth trials
  #rdd_calcs_poly function performs RDD with optimal bandwidth, and trials different polynomials for time
  #rdd_calcs_bw function trials RDD with different bandwidths for linear in time form

inputparams<-function(i,t){

  #compute bandwidth
  data_model<-data
```

```
  cols<-c("dow", "year")
  data_model<-data_model[, (cols):=lapply(.SD, as.factor), .SDcols=cols]
  tryCatch(
    expr = bw<-rdbwselect(y=data_model$tot_casualties, x=data_model$time_variable_tp,
                          covs=cbind(data_model$year,data_model$dow)),
    error = function(e) NULL
  )
  if(exists("bw")==TRUE){
    bw_exists=1
    bwl=ceiling(bw$bws[[1]])
    bwr=ceiling(bw$bws[[2]])
    #tabulate
    resultsline<-data.table(i,t,bwl,bwr,nrow(data_model),bw_exists)
    names(resultsline)<-c("model_no","tp","bw_mainl","bw_mainr","n","bw_exists")
  } else if (exists("bw")==FALSE){
    resultsline<-data.table(i,t,0,0,0,0)
    names(resultsline)<-c("model_no","tp","bw_mainl","bw_mainr","n","bw_exists")
  }
  return(resultsline)
}

rdd_calcs_poly<-function(i,t){
  #extract bandwidth info
  paras_i<-mod_paras[model_no==i & tp==t]
  bwl=paras_i$bw_mainl
  bwr=paras_i$bw_mainr
  bwexists=paras_i$bw_exists
  if (bwexists==1){
    #prepare data: need wt, kt and ktpost variables (cutoff is when time_variable_tp=0)
    data_model<-data[time_variable_tp>=-bwl & time_variable_tp<=bwr]
    data_model<-data_model[time_variable_tp>=0, wt:=1]
    data_model<-data_model[time_variable_tp<0, wt:=0]
    mintp=-1*min(data_model$time_variable_tp)
    interventiontp=mintp+1
    data_model<-data_model[time_variable_tp<0,kt:=(interventiontp)+time_variable_tp]
    data_model<-data_model[time_variable_tp>=0,kt:=mintp+time_variable_tp]
    data_model<-data_model[time_variable_tp<0, ktpost:=0]
```

```
    data_model<-data_model[time_variable_tp>=0, ktpost:=kt-interventiontp+1]

    cols<-c("dow", "year")
    data_model<-data_model[, (cols):=lapply(.SD, as.factor), .SDcols=cols]

    rdd_polytrial<-function(j){

      tryCatch(
        expr=lm_rdd<-lm(tot_casualties ~ wt + poly(kt, degree=j) + poly(ktpost, degree=j) + dow + year,
data=data_model),
        error=function(e) NULL
      )
      if(exists("lm_rdd")==TRUE){
        #Breusch Godfrey autocorrelation test up to lag 10
        for (l in c(1:10)){
          tryCatch(
            expr = assign(paste0("bgtest_",l), value=BreuschGodfreyTest(lm_rdd, order = l, order.by = data_model$kt,
type = "Chisq", data = data_model)),
            error = function(e) NULL
          )
          tryCatch(
            expr = assign(paste0("lag_",l), value=eval(parse(text=paste0("bgtest_",l,"$p.value")))),
            error = function(e) NULL
          )
          tryCatch(
            expr = assign(paste0("b",l),value=eval(parse(text=paste0("data.table(as.numeric(lag_",l,"),",l,")")))),
            error = function(e) NULL
          )
        }
        tryCatch(
          expr = bgtab<-rbind(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10),
          error = function(e) NULL
        )
        if (exists("bgtab")==FALSE){
          lag_val=0
        } else if (exists("bgtab")==TRUE){
          bgtab_select<-bgtab[V1<=0.1]
```

```
          if (nrow(bgtab_select)>=1){
            lag_val=max(bgtab_select$V2)
          } else if (lag_1>0.1){
            lag_val=0
          }
        }
        #Bruesch Pagan heteroskedasticity test
        bp_pval=bptest(lm_rdd)$p.value
        if (is.na(bp_pval)){
          bp_pval=100
        }
        #adjust errors if needed to account for autocorrelation or heteroskedasticity
        if (lag_val>0){
          nw_vcov <- NeweyWest(lm_rdd, order.by = data_model$kt, data=data_model, lag = lag_val, prewhite = F,
adjust = T)
          lmsum<-as.matrix(coeftest(lm_rdd, vcov = nw_vcov))
          lm_sum<-lmsum[2,]
        } else if (lag_val==0 & bp_pval<=0.1){
          hc_vcov <- vcovHC(lm_rdd)
          lmsum<-as.matrix(coeftest(lm_rdd, vcov = hc_vcov))
          lm_sum<-lmsum[2,]
        } else if (lag_val==0 & bp_pval>0.1){
          lmsum<-as.matrix(summary(lm_rdd)$coefficients)
          lm_sum<-lmsum[2,]
        }

        data_model_l<-data_model[time_variable_tp<0]
        data_model_r<-data_model[time_variable_tp>=0]
        totcas_l=sum(data_model_l$tot_casualties)
        totcas_r=sum(data_model_r$tot_casualties)
        n_year=length(unique(data_model$year))

        resultsline<-
data.table(i,t,j,lm_sum[[1]],lm_sum[[2]],lm_sum[[4]],nrow(data_model),nrow(data_model_l),nrow(data_model_r),
                                as.numeric(bwl), as.numeric(bwr),as.numeric(lag_val),as.numeric(bp_pval),
                                summary(lm_rdd)$r.squared, summary(lm_rdd)$adj.r.squared, BIC(lm_rdd),
AIC(lm_rdd),totcas_l,totcas_r,n_year)
```

```
        names(resultsline)<-c("model_no","tp","poly_deg","coef","se", "pval",
"n_tot","n_left","n_right","bw_l","bw_r","lag","bp_pval",
                              "rsq","adj_rsq","bic","aic","totcas_l","totcas_r","n_year")
        return(resultsline)
      }}
    #run for polynomials order 1 to 4
    results_table<-c()
    for (j in 1:4){
      calcs<-rdd_polytrial(j)
      results_table<-rbind(calcs,results_table)
    }
    return(results_table)
  }}

rdd_calcs_bw<-function(i,t){
  #extract bandwidth info
  paras_i<-mod_paras[model_no==i & tp==t]
  bwl=paras_i$bw_mainl
  bwr=paras_i$bw_mainr
  bwexists=paras_i$bw_exists

  if (bwexists==1){
    rdd_bwtrial<-function(j){

      #prepare data: need wt, kt and ktpost variables (cutoff is when time_variable_tp=0)
      data_model<-data[time_variable_tp>=-(bwl+j) & time_variable_tp<=(bwr+j)]
      data_model<-data_model[time_variable_tp>=0, wt:=1]
      data_model<-data_model[time_variable_tp<0, wt:=0]
      mintp=-1*min(data_model$time_variable_tp)
      interventiontp=mintp+1
      data_model<-data_model[time_variable_tp<0,kt:=(interventiontp)+time_variable_tp]
      data_model<-data_model[time_variable_tp>=0,kt:=mintp+time_variable_tp]
      data_model<-data_model[time_variable_tp<0, ktpost:=0]
      data_model<-data_model[time_variable_tp>=0, ktpost:=kt-interventiontp+1]

      cols<-c("dow", "year")
      data_model<-data_model[, (cols):=lapply(.SD, as.factor), .SDcols=cols]
```

```
      tryCatch(
        expr=lm_rdd<-lm_rdd<-lm(tot_casualties ~ wt + poly(kt, degree=1) + poly(ktpost, degree=1)  + year + dow,
data=data_model),
        error=function(e) NULL
      )
      if(exists("lm_rdd")==TRUE){
        #Breusch Godfrey autocorrelation test up to lag 10
        for (l in c(1:10)){
          tryCatch(
            expr = assign(paste0("bgtest_",l), value=BreuschGodfreyTest(lm_rdd, order = l, order.by = data_model$kt,
type = "Chisq", data = data_model)),
            error = function(e) NULL
          )
          tryCatch(
            expr = assign(paste0("lag_",l), value=eval(parse(text=paste0("bgtest_",l,"$p.value")))),
            error = function(e) NULL
          )
          tryCatch(
            expr = assign(paste0("b",l),value=eval(parse(text=paste0("data.table(as.numeric(lag_",l,"),",l,")")))),
            error = function(e) NULL
          )
        }
        tryCatch(
          expr = bgtab<-rbind(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10),
          error = function(e) NULL
        )
        if (exists("bgtab")==FALSE){
          lag_val=0
        } else if (exists("bgtab")==TRUE){
          bgtab_select<-bgtab[V1<=0.1]
          if (nrow(bgtab_select)>=1){
            lag_val=max(bgtab_select$V2)
          } else if (lag_1>0.1){
            lag_val=0
          }
        }
```

```
        #Bruesch Pagan heteroskedasticity test
        bp_pval=bptest(lm_rdd)$p.value
        if (is.na(bp_pval)){
          bp_pval=100
        }
        #adjust errors if needed to account for autocorrelation or heteroskedasticity
        if (lag_val>0){
          nw_vcov <- NeweyWest(lm_rdd, order.by = data_model$kt, data=data_model, lag = lag_val, prewhite = F,
adjust = T)
          lmsum<-as.matrix(coeftest(lm_rdd, vcov = nw_vcov))
          lm_sum<-lmsum[2,]
        } else if (lag_val==0 & bp_pval<=0.1){
          hc_vcov <- vcovHC(lm_rdd)
          lmsum<-as.matrix(coeftest(lm_rdd, vcov = hc_vcov))
          lm_sum<-lmsum[2,]
        } else if (lag_val==0 & bp_pval>0.1){
          lmsum<-as.matrix(summary(lm_rdd)$coefficients)
          lm_sum<-lmsum[2,]
        }

        data_model_l<-data_model[time_variable_tp<0]
        data_model_r<-data_model[time_variable_tp>=0]
        totcas_l=sum(data_model_l$tot_casualties)
        totcas_r=sum(data_model_r$tot_casualties)
        n_year=length(unique(data_model$year))

        resultsline<-
data.table(i,t,j,lm_sum[[1]],lm_sum[[2]],lm_sum[[4]],nrow(data_model),nrow(data_model_l),nrow(data_model_r),
                                  as.numeric(bwl+j),as.numeric(bwr+j),as.numeric(lag_val),as.numeric(bp_pval),
                                  summary(lm_rdd)$r.squared, summary(lm_rdd)$adj.r.squared, BIC(lm_rdd),
AIC(lm_rdd),totcas_l,totcas_r,n_year)
        names(resultsline)<-c("model_no","tp","bw_adjust","coef","se", "pval",
"n_tot","n_left","n_right","bw_l","bw_r","lag","bp_pval",
                              "rsq","adj_rsq","bic","aic","totcas_l","totcas_r","n_year")
        return(resultsline)
      }}
```

```
    #run for different bandwiths as follows:
    seq_bw<-c(1,0,-1)
    results_table<-c()
    for (j in seq_bw){
      calcs<-rdd_bwtrial(j)
      results_table<-rbind(calcs,results_table)
    }
    return(results_table)
  }}


#Run functions and save output as csv files
param_table<-c()
for (i in seq_i){
  for (t in seq_t){
    data_raw<-eval(parse(text=paste0("data_",i)))
    data<-data_raw[tp==t]
    param_table<-rbind(param_table,inputparams(i,t))
    }}
write.csv(param_table, file="model_params.csv", row.names=FALSE)

poly_table<-c()
for (i in seq_i){
  for (t in seq_t){
    data_raw<-eval(parse(text=paste0("data_",i)))
    data<-data_raw[tp==t]
    mod_paras<-fread("model_params.csv")
    poly_table<-rbind(poly_table,rdd_calcs_poly(i,t))
  }}
write.csv(poly_table, file="results_polytrial.csv", row.names=FALSE)

bw_table<-c()
for (i in seq_i){
  for (t in seq_t){
    data_raw<-eval(parse(text=paste0("data_",i)))
    data<-data_raw[tp==t]
    mod_paras<-fread("model_params.csv")
```

```
    bw_table<-rbind(bw_table,rdd_calcs_bw(i,t))
}}
write.csv(bw_table, file="results_bwtrial.csv", row.names=FALSE)

################################################################################
#Script for placebo tests as per Guido W. Imbens and Thomas Lemieux. Regression discontinuity designs: A guide to
practice. Journal of Econometrics, 142:615–635, 2008.
#We use the same bandwidth as per the associated original models in file "model_params.csv" as generated in script
"rdd_models.R"

#attach packages
library(data.table)
library(DescTools)
library(rdrobust)
library(sandwich)
library(lmtest)

#set working directory
setwd("~/Documents/DST")

#Define sequences of data sets
#The STATS19 data have been segmented into separate files representing 7 easting and 13 northing bands and 1
aggregate data set (21 files in total)
#Datasets are named with same text prefix "data_", different numerical suffix "i"
#We define the sequence of datasets as seq_i, which refers to the 21 segmented datasets described above
#We define seq_q so that we can split each data set into pre- ("left") and post- ("right") DST for 2 placebo tests
per original model
#We define seq_t as the sequence of time periods ranging from 1 to 5
#The functions will iterate through all sequences and output results tables with all models' results
seq_i<-c(1:21)
seq_q<-c("left","right")
seq_t<-c(1:5)

#Define functions:
#rdd_calcs_poly function performs RDD with optimal bandwidth, and trials different polynomials for time
#rdd_calcs_bw function trials RDD with different bandwidths for linear in time form
```

```
rdd_calcs_poly<-function(i,q,t){
  #extract bandwidth info
  paras_iq<-mod_paras[model_no==i & tp==t]
  bwl=paras_iq$bw_mainl
  bwr=paras_iq$bw_mainr
  bwexists=paras_iq$bw_exists

  if (bwexists==1){
    #prepare data: need wt, kt and ktpost variables
    data_model<-data[time_variable_tp>=cut-bwl & time_variable_tp<cut+bwr]
    data_model<-data_model[time_variable_tp>=cut, wt:=1]
    data_model<-data_model[time_variable_tp<cut, wt:=0]
    mintp=min(data_model$time_variable_tp)
    interventiontp=abs(mintp)+1
    data_model<-data_model[,kt:=interventiontp+time_variable_tp]
    data_model<-data_model[wt==0, ktpost:=0]
    maxkt_wt0=max(data_model[wt==0]$kt)
    data_model<-data_model[wt==1, ktpost:=kt-maxkt_wt0]

    cols<-c("dow", "year")
    data_model<-data_model[, (cols):=lapply(.SD, as.factor), .SDcols=cols]

    rdd_polytrial<-function(j){

      tryCatch(
        expr=lm_rdd<-lm(tot_casualties ~ wt + poly(kt, degree=j) + poly(ktpost, degree=j)  + dow + year,
data=data_model),
        error=function(e) NULL
      )
      if(exists("lm_rdd")==TRUE){
        #Breusch Godfrey autocorrelation test up to lag 10
        for (l in c(1:10)){
          tryCatch(
            expr = assign(paste0("bgtest_",l), value=BreuschGodfreyTest(lm_rdd, order = l, order.by = data_model$kt,
type = "Chisq", data = data_model)),
            error = function(e) NULL
          )
```

```
  tryCatch(
    expr = assign(paste0("lag_",l), value=eval(parse(text=paste0("bgtest_",l,"$p.value")))),
    error = function(e) NULL
  )
  tryCatch(
    expr = assign(paste0("b",l),value=eval(parse(text=paste0("data.table(as.numeric(lag_",l,"),",l,")")))),
    error = function(e) NULL
  )
}
tryCatch(
  expr = bgtab<-rbind(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10),
  error = function(e) NULL
)
if (exists("bgtab")==FALSE){
  lag_val=0
} else if (exists("bgtab")==TRUE){
  bgtab_select<-bgtab[V1<=0.1]
  if (nrow(bgtab_select)>=1){
    lag_val=max(bgtab_select$V2)
  } else if (lag_1>0.1){
    lag_val=0
  }
}
#Bruesch Pagan heteroskedasticity test
bp_pval=bptest(lm_rdd)$p.value
if (is.na(bp_pval)){
  bp_pval=100
}
#adjust errors if needed to account for autocorrelation or heteroskedasticity
if (lag_val>0){
  nw_vcov <- NeweyWest(lm_rdd, order.by = data_model$kt, data=data_model, lag = lag_val, prewhite = F,
adjust = T)
  lmsum<-as.matrix(coeftest(lm_rdd, vcov = nw_vcov))
  lm_sum<-lmsum[2,]
} else if (lag_val==0 & bp_pval<=0.1){
  hc_vcov <- vcovHC(lm_rdd)
  lmsum<-as.matrix(coeftest(lm_rdd, vcov = hc_vcov))
```

```
          lm_sum<-lmsum[2,]
        } else if (lag_val==0 & bp_pval>0.1){
          lmsum<-as.matrix(summary(lm_rdd)$coefficients)
          lm_sum<-lmsum[2,]
        }

        data_model_l<-data_model[wt==0]
        data_model_r<-data_model[wt==1]

        resultsline<-
data.table(i,q,t,j,lm_sum[[1]],lm_sum[[2]],lm_sum[[4]],nrow(data_model),nrow(data_model_l),nrow(data_model_r),
                                as.numeric(bwl),as.numeric(bwr),as.numeric(lag_val),as.numeric(bp_pval),
                                summary(lm_rdd)$r.squared, summary(lm_rdd)$adj.r.squared, BIC(lm_rdd), AIC(lm_rdd))
        names(resultsline)<-c("model_no","data","tp","poly_deg","coef","se", "pval",
"n_tot","n_left","n_right","bw_l","bw_r","lag","bp_pval",
                              "rsq","adj_rsq","bic","aic")
        return(resultsline)
      }}
    #run for polynomials order 1 to 4
    results_table<-c()
    for (j in 1:4){
      calcs<-rdd_polytrial(j)
      results_table<-rbind(calcs,results_table)
    }
    return(results_table)
  }}

rdd_calcs_bw<-function(i,q,t){
  #extract bandwidth info
  paras_iq<-mod_paras[model_no==i & tp==t]
  bwl=paras_iq$bw_mainl
  bwr=paras_iq$bw_mainr
  bwexists=paras_iq$bw_exists

  if (bwexists==1){
    rdd_bwtrial<-function(j){
```

```
      #prepare data: need wt, kt and ktpost variables
      newcut=cut+j
      data_model<-data[time_variable_tp>=newcut-bwl & time_variable_tp<newcut+bwr]
      data_model<-data_model[time_variable_tp>=newcut, wt:=1]
      data_model<-data_model[time_variable_tp<newcut, wt:=0]
      mintp=min(data_model$time_variable_tp)
      interventiontp=abs(mintp)+1
      data_model<-data_model[,kt:=interventiontp+time_variable_tp]
      data_model<-data_model[wt==0, ktpost:=0]
      maxkt_wt0=max(data_model[wt==0]$kt)
      data_model<-data_model[wt==1, ktpost:=kt-maxkt_wt0]

      cols<-c("dow", "year")
      data_model<-data_model[, (cols):=lapply(.SD, as.factor), .SDcols=cols]

      tryCatch(
        expr=lm_rdd<-lm_rdd<-lm(tot_casualties ~ wt + poly(kt, degree=1) + poly(ktpost, degree=1) + dow + year,
data=data_model),
        error=function(e) NULL
      )
      if(exists("lm_rdd")==TRUE){
        #Breusch Godfrey autocorrelation test up to lag 10
        for (l in c(1:10)){
          tryCatch(
            expr = assign(paste0("bgtest_",l), value=BreuschGodfreyTest(lm_rdd, order = l, order.by = data_model$kt,
type = "Chisq", data = data_model)),
            error = function(e) NULL
          )
          tryCatch(
            expr = assign(paste0("lag_",l), value=eval(parse(text=paste0("bgtest_",l,"$p.value")))),
            error = function(e) NULL
          )
          tryCatch(
            expr = assign(paste0("b",l),value=eval(parse(text=paste0("data.table(as.numeric(lag_",l,"),",l,")")))),
            error = function(e) NULL
          )
        }
```

```
        tryCatch(
          expr = bgtab<-rbind(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10),
          error = function(e) NULL
        )
        if (exists("bgtab")==FALSE){
          lag_val=0
        } else if (exists("bgtab")==TRUE){
          bgtab_select<-bgtab[V1<=0.1]
          if (nrow(bgtab_select)>=1){
            lag_val=max(bgtab_select$V2)
          } else if (lag_1>0.1){
            lag_val=0
          }
        }
        #Bruesch Pagan heteroskedasticity test
        bp_pval=bptest(lm_rdd)$p.value
        if (is.na(bp_pval)){
          bp_pval=100
        }
        #adjust errors if needed to account for autocorrelation or heteroskedasticity
        if (lag_val>0){
          nw_vcov <- NeweyWest(lm_rdd, order.by = data_model$kt, data=data_model, lag = lag_val, prewhite = F,
adjust = T)
          lmsum<-as.matrix(coeftest(lm_rdd, vcov = nw_vcov))
          lm_sum<-lmsum[2,]
        } else if (lag_val==0 & bp_pval<=0.1){
          hc_vcov <- vcovHC(lm_rdd)
          lmsum<-as.matrix(coeftest(lm_rdd, vcov = hc_vcov))
          lm_sum<-lmsum[2,]
        } else if (lag_val==0 & bp_pval>0.1){
          lmsum<-as.matrix(summary(lm_rdd)$coefficients)
          lm_sum<-lmsum[2,]
        }

        data_model_l<-data_model[wt==0]
        data_model_r<-data_model[wt==1]
```

```
        resultsline<-
data.table(i,q,t,j,lm_sum[[1]],lm_sum[[2]],lm_sum[[4]],nrow(data_model),nrow(data_model_l),nrow(data_model_r),
                                as.numeric(bwl+j),as.numeric(bwr+j),as.numeric(lag_val),as.numeric(bp_pval),
                                summary(lm_rdd)$r.squared, summary(lm_rdd)$adj.r.squared, BIC(lm_rdd), AIC(lm_rdd))
        names(resultsline)<-c("model_no","data","tp","bw_adjust","coef","se", "pval",
"n_tot","n_left","n_right","bw_l","bw_r","lag","bp_pval",
                              "rsq","adj_rsq","bic","aic")
        return(resultsline)
      }}

    #run for different bandwiths as follows:
    seq_bw<-c(1,0,-1)
    results_table<-c()
    for (j in seq_bw){
      calcs<-rdd_bwtrial(j)
      results_table<-rbind(calcs,results_table)
    }
    return(results_table)
  }}


#Run functions and save output as csv files
poly_table<-c()
for (i in seq_i){
  for (q in seq_q){
    for (t in seq_t){
      data_raw<-eval(parse(text=paste0("data_",i)))
      data_tp<-data_raw[tp==t]
      if (q=="left"){
        data<-data_tp[time_variable_tp<0]
        cut=round(mean(data$time_variable_tp))
      } else if (q=="right"){
        data<-data_tp[time_variable_tp>=0]
        cut=round(mean(data$time_variable_tp))
      }
      mod_paras<-fread("model_params.csv")
      poly_table<-rbind(poly_table,rdd_calcs_poly(i,q,t))
```

```
    }}}
write.csv(poly_table, file="results_placebo_polytrial.csv", row.names=FALSE)

bw_table<-c()
for (i in seq_i){
  for (q in seq_q){
    for (t in seq_t){
      data_raw<-eval(parse(text=paste0("data_",i)))
      data_tp<-data_raw[tp==t]
      if (q=="left"){
        data<-data_tp[time_variable_tp<0]
        cut=round(mean(data$time_variable_tp))
      } else if (q=="right"){
        data<-data_tp[time_variable_tp>=0]
        cut=round(mean(data$time_variable_tp))
      }
      mod_paras<-fread("model_params.csv")
      bw_table<-rbind(bw_table,rdd_calcs_bw(i,t))
    }}}
write.csv(bw_table, file="results_placebo_bwtrial.csv", row.names=FALSE)
```