```
########## R EXAMPLES #########

# 1
install.packages("causalweight")  # install causalweight package
library(causalweight)             # load causalweight package
data(JC)                          # load JC data
?JC                               # call documentation for JC data
D=JC$assignment                   # define treatment (assignment to JC)
Y=JC$earny4                       # define outcome (earnings in fourth year)
mean(Y[D==1])-mean(Y[D==0])       # compute the ATE

# 2
library(causalweight)             # load causalweight package
library(lmtest)                   # load lmtest package
library(sandwich)                 # load sandwich package
data(JC)                          # load JC data
D=JC$assignment                   # define treatment (assignment to JC)
Y=JC$earny4                       # define outcome (earnings in fourth year)
ols=lm(Y~D)                       # run OLS regression
coeftest(ols, vcov=vcovHC)        # output with heteroscedasticity-robust se

# 3
bs=function(data, indices) {     # defines function bs for bootstrapping
  dat=data[indices,]             # creates bootstrap sample according to indices
  coefficients=lm(dat)$coef      # estimates coefficients in bootstrap sample
  return(coefficients)           # returns coefficients
}                                # closes the function bs
library(boot)                    # load boot package
bootdata=data.frame(Y,D)         # data frame with Y,D for bootstrap procedure
set.seed(1)                      # set seed
results = boot(data=bootdata, statistic=bs, R=1999) # 1999 bootstrap estimations
results                          # displays the results
tstat=results$t0[2]/sd(results$t[,2])  # compute the t-statistic
```

```
2*pnorm(-abs(tstat))                   # compute the p-value

# 4
library(causalweight)              # load causalweight package
library(lmtest)                    # load lmtest package
library(sandwich)                  # load sandwich package
data(wexpect)                      # load wexpect data
?wexpect                           # call documentation for wexpect data
D1=wexpect$treatmentinformation    # define first treatment (wage information)
D2=wexpect$treatmentorder          # define second treatment (order of questions)
Y=wexpect$wexpect2                 # define outcome (wage expectations)
ols=lm(Y~D1+D2)                    # run OLS regression
coeftest(ols, vcov=vcovHC)         # output with heteroscedasticity robust se

# 5
library(datarium)                  # load datarium package
library(np)                        # load np package
data(marketing)                    # load marketing data
?marketing                         # call documentation for marketing data
D=marketing$newspaper              # define treatment (newspaper advertising)
Y=marketing$sales                  # define outcome (sales)
results=npregbw(Y~D)               # kernel regression
plot(results, plot.errors.method="asymptotic") # plot regression function
plot(results, gradients=TRUE, plot.errors.method="asymptotic") # plot effects

# 6
library(causalweight)              # load causalweight package
library(lmtest)                    # load lmtest package
library(sandwich)                  # load sandwich package
data(coffeeleaflet)                # load coffeeleaflet data
attach(coffeeleaflet)              # store all variables in own objects
?coffeeleaflet                     # call documentation for coffeeleaflet data
D=treatment                        # define treatment (leaflet)
Y=awarewaste                       # define outcome (aware of waste production)
X=cbind(mumedu,sex)                # define covariates (grade, gender, age)
ols=lm(Y~D+X)                      # run OLS regression
```

```
coeftest(ols, vcov=vcovHC)        # output with heteroscedasticity robust se

# 7
library(Matching)                       # load Matching package
library(Jmisc)                          # load Jmisc package
library(lmtest)                         # load lmtest package
library(sandwich)                       # load sandwich package
data(lalonde)                           # load lalonde data
attach(lalonde)                         # store all variables in own objects
?lalonde                                # call documentation for lalonde data
D=treat                                 # define treatment (training)
Y=re78                                  # define outcome
X=cbind(age,educ,nodegr,married,black,hisp,re74,re75,u74,u75) # covariates
DXdemeaned=D*demean(X)                  # interaction of D and demeaned X
ols=lm(Y~D+X+DXdemeaned)                # run OLS regression
coeftest(ols, vcov=vcovHC)              # output

# 8
library(Matching)                       # load Matching package
data(lalonde)                           # load lalonde data
attach(lalonde)                         # store all variables in own objects
D=treat                                 # define treatment (training)
Y=re78                                  # define outcome
X=cbind(age,educ,nodegr,married,black,hisp,re74,re75,u74,u75) # covariates
pairmatching=Match(Y=Y, Tr=D, X=X)      # pair matching
summary(pairmatching)                   # matching output

# 9
matching=Match(Y=Y, Tr=D, X=X, M=3, BiasAdjust = TRUE) # 1:M matching
summary(matching)                                      # matching output

# 10
library(Matching)                       # load Matching package
data(lalonde)                           # load lalonde data
attach(lalonde)                         # store all variables in own objects
D=treat                                 # define treatment (training)
```

```
Y=re78                                  # define outcome
X=cbind(age,educ,nodegr,married,black,hisp,re74,re75,u74,u75) # covariates
ps=glm(D~X,family=binomial)$fitted      # estimate the propensity score by logit
psmatching=Match(Y=Y, Tr=D, X=ps, BiasAdjust = TRUE) # propensity score matching
summary(psmatching)                     # matching output

# 11
library(Matching)                       # load Matching package
library(boot)                           # load boot package
data(lalonde)                           # load lalonde data
attach(lalonde)                         # store all variables in own objects
D=treat                                 # define treatment (training)
Y=re78                                  # define outcome
X=cbind(age,educ,nodegr,married,black,hisp,re74,re75,u74,u75) # covariates
bs=function(data, indices) {            # defines function bs for bootstrapping
  dat=data[indices,]                    # bootstrap sample according to indices
  ps=glm(dat[,2:ncol(dat)],data=dat,family=binomial)$fitted # propensity score
  effect=Match(Y=dat[,1], Tr=dat[,2], X=ps, BiasAdjust = TRUE)$est # ATET
  return(effect)                        # returns the estimated ATET
}                                       # closes the function bs
bootdata=data.frame(Y,D,X)              # data frame for bootstrap procedure
set.seed(1)                             # set seed
results = boot(data=bootdata, statistic=bs, R=999) # 999 bootstrap estimations
results                                 # displays the results
tstat=results$t0/sd(results$t)          # compute the t-statistic
2*pnorm(-abs(tstat))                    # compute the p-value

# 12
library(causalweight)                   # load causalweight package
library(COUNT)                          # load COUNT package
data(lbw)                               # load lbw data
attach(lbw)                             # store all variables in own objects
D=smoke                                 # define treatment (mother smoking)
Y=bwt                                   # outcome (birthweight in grams)
X=cbind(race==1, age, lwt, ptl, ht, ui, ftv) # covariates
set.seed(1)                             # set seed
```

```
ipw=treatweight(y=Y,d=D,x=X, boot=999)  # run IPW with 999 bootstraps
ipw$effect                              # show ATE
ipw$se                                  # show standard error
ipw$pval                                # show p-value

# 13
library(CBPS)                           # load CBPS package
library(lmtest)                         # load lmtest package
library(sandwich)                       # load sandwich package
cbps=CBPS(D~X, ATT = 0)                 # covariate balancing for ATE estimation
results=lm(Y~D, weights=cbps$weights)   # weighted regression
coeftest(results, vcov = vcovHC)        # show results

# 14
library(drgee)                          # load drgee package
library(COUNT)                          # load COUNT package
data(lbw)                               # load lbw data
attach(lbw)                             # store all variables in own objects
D=smoke                                 # define treatment (mother smoking)
Y=bwt                                   # outcome (birthweight in grams)
X=cbind(race==1, age, lwt, ptl, ht, ui, ftv) # covariates
dr=drgee(oformula=formula(Y~X), eformula=formula(D~X), elink="logit") # DR reg
summary(dr)                             # show results

# 15
library(COUNT)                  # load COUNT package
library(kdensity)               # load kdensity package
data(lbw)                       # load lbw data
attach(lbw)                     # store all variables in own objects
D=smoke                         # define treatment (mother smoking)
Y=bwt                           # outcome (birthweight in grams)
X=cbind(race==1, age, lwt, ptl, ht, ui, ftv) # covariates
ps=glm(D~X,family=binomial)$fitted # estimate the propensity score by logit
psdens1=kdensity(ps[D==1])      # density of propensity score among treated
psdens0=kdensity(ps[D==0])      # density of propensity score among non-treated
par(mfrow=c(2,2))               # specify a figure with four graphs (2X2)
```

```
plot(psdens1)                    # plot density for treated
plot(psdens0)                    # plot density for non-treated
hist(ps[D==1])                   # plot histogram of p-score for treated
hist(ps[D==0])                   # plot histogram of p-score for non-treated
summary(ps[D==1])                # summary statistics for p-scores among treated
summary(ps[D==0])                # summary statistics p-scores among non-treated

# 16
library(MatchIt)                   # load MatchIt package
output=matchit(D~X)                # pair matching (ATET) on propensity score
plot(output,type="hist")           # plot common support before/after matching
summary(output,standardize=TRUE)

# 17
library(Matching)                       # load Matching package
output1=Match(Y=Y, Tr=D, X=ps)          # pair matching (ATET) on p-score
MatchBalance(D~ptl, match.out=output1) # covariate balance before/after matching
output2=Match(Y=Y, Tr=D, X=ps, CommonSupport=TRUE) # pair matching (ATET)
MatchBalance(D~lwt, match.out=output2) # covariate balance before/after matching
summary(output1)                       # ATET without common support
summary(output2)                       # ATET with common support

# 18
library(causalweight)                   # load causalweight package
set.seed(1)                             # set seed to 1
ipw=treatweight(y=ptl,d=D,x=X, boot=999)# run IPW with 999 bootstraps
ipw$effect                              # show mean difference in X
ipw$pval                                # show p-value

# 19
set.seed(1)                                       # set seed to 1
ipw=treatweight(y=ptl,d=D,x=X, trim=0.1, boot=999)# run IPW with 999 bootstraps
ipw$effect                                        # show mean difference in X
ipw$pval                                          # show p-value
ipw$ntrimmed                                      # number of trimmed units
```

```
# 20
library(causalweight)                     # load causalweight package
library(devtools)                         # load devtools package
install_github("ehkennedy/npcausal")      # install npcausal package
library(npcausal)                         # load npcausal package
data(games)                               # load games data
games_nomis=na.omit(games)                # drop observations with missings
attach(games_nomis)                       # attach data
X=cbind(year,userscore, genre=="Action")  # define covariates
D=metascore                               # define treatment
Y=sales                                   # define outcome
results=ctseff(y=Y, a=D, x=X, bw.seq=seq(from=1,to=5,by=0.5)) # DR estimation
plot.ctseff(results)                      # potential outcome-treatment relation

# 21
library(qte)                              # load qte package
D=metascore>75                            # define binary treatment (score>75)
dat=data.frame(Y,D,X)                     # create data frame
QTE=ci.qte(Y~D, x=X, data=dat)            # estimate QTE across different ranks
ggqte(QTE)                                # plot QTEs across ranks (tau)

# 22
library(causalweight)            # load causalweight package
data(JC)                         # load JC data
X0=JC[,2:29]                     # define pre-treatment covariates X0
X1=JC[,30:36]                    # define post-treatment covariates X1
D1=JC[,37]                       # define treatment (training) in first year D1
D2=JC[,38]                       # define treatment (training) in second year D2
Y2=JC[,44]                       # define outcome (earnings in fourth year) Y2
output=dyntreatDML(y2=Y2,d1=D1,d2=D2,x0=X0,x1=X1) # doubly robust estimation
output$effect; output$se; output$pval # effect, standard error, p-value

# 23
output=dyntreatDML(y2=Y2,d1=D1,d2=D2,x0=X0,x1=X1, d2treat=0) # estimation
output$effect; output$se; output$pval # effect, standard error, p-value
```

```
# 24
library(causalweight)                                # load causalweight package
data(wexpect)                                        # load wexpect data
attach(wexpect)                                      # attach data
X=cbind(age,swiss,motherhighedu,fatherhighedu) # define covariates
D=male                                               # define treatment
M=cbind(business,econ,communi,businform)       # define mediator
Y=wexpect2                                           # define outcome
medDML(y=Y, d=D, m=M, x=X)                           # estimate causal mechanisms

# 25
library(causalweight)            # load causalweight package
data(JC)                         # load JC data
X0=JC[,2:29]                     # define pre-treatment covariates X0
X1=JC[,30:36]                    # define post-treatment covariates X1
D1=JC[,37]                       # define treatment (training) in first year D1
D2=JC[,38]                       # define treatment (training) in second year D2
Y2=JC[,44]                       # define outcome (earnings in fourth year) Y2
output=dyntreatDML(y2=Y2,d1=D1,d2=D2,x0=X0,x1=X1) # doubly robust estimation
output$effect; output$se; output$pval # effect, standard error, p-value

# 26
library(causalweight)            # load causalweight package
data(JC)                         # load JC data
X=JC[,2:29]                      # define covariates
D=JC[,37]                        # define treatment (training) in first year
Y=JC[,46]                        # define outcome (health state after 4 years)
output=treatDML(y=Y, d=D, x=X)   # double machine learning
output$effect; output$se; output$pval # effect, standard error, p-value

# 27
output=treatDML(y=Y,d=D,x=X,MLmethod="randomforest") # double machine learning
output$effect; output$se; output$pval # effect, standard error, p-value

# 28
library(grf)                      # load grf package
```

```
library(causalweight)             # load causalweight package
data(JC)                          # load JC data
X=JC[,2:29]                       # define covariates
D=JC[,37]                         # define treatment (training) in first year
Y=JC[,40]                         # outcome (proportion employed in third year)
set.seed(1)                       # set seed
cf=causal_forest(X=X, Y=Y, W=D)   # run causal forest
ATE=average_treatment_effect(cf)  # compute ATE
pval=2*pnorm(-abs(ATE[1]/ATE[2])) # compute the p-value
ATE; pval                         # provide ATE, standard error, and p-value

# 29
CATE=cf$predictions               # store CATEs in own variable
hist(CATE)                        # distribution of CATEs

# 30
library(lmtest)                                     # load lmtest package
library(sandwich)                                   # load sandwich package
highCATE=CATE>median(CATE)                          # dummy for high CATE
ols=lm(JC$age~highCATE)                             # regress CATEs on gender
coeftest(ols, vcov=vcovHC)                          # output

# 31
best_linear_projection(forest=cf,A=JC$female) # regression of function on gender

# 32
library(randomForest)                     # load randomForest package
dat=data.frame(CATE,X)                    # define data frame
randomf=randomForest(CATE~. ,data=dat)    # predict CATE as a function of X
importance(randomf)                       # show predictive importance of X

# 33
library(Matching)                       # load Matching package
library(policytree)                     # load policytree package
library(DiagrammeR)                     # load DiagrammeR package
data(lalonde)                           # load lalonde data
```

```
attach(lalonde)                           # store all variables in own objects
D=factor(treat)                           # define treatment (training)
Y=re78                                    # define outcome
X=cbind(age,educ,nodegr,married,black,hisp,re74,re75,u74,u75) # covariates
forest=multi_arm_causal_forest(X=X, Y=Y, W=D) #estimate treatment+outcome models
influence=double_robust_scores(forest)    # obtain efficient influence functions
Xpol=cbind(age,educ,nodegr)               # relevant X for optimal policy
tree=policy_tree(X=Xpol, Gamma=influence, depth=2) # policies for 4 subgroups
plot(tree)                                # plot the tree with optimal policies

# 34
library(causalweight)             # load causalweight package
data(JC)                          # load JC data
Z=JC$assignment                   # define instrument (assignment to JC)
D=JC$trainy1                      # define treatment (training in 1st year)
Y=JC$earny4                       # define outcome (earnings in fourth year)
ITT=mean(Y[Z==1])-mean(Y[Z==0])   # estimate intention-to-treat effect (ITT)
first=mean(D[Z==1])-mean(D[Z==0]) # estimate first stage effect (complier share)
LATE=ITT/first                    # compute LATE
ITT; first; LATE                  # show ITT, first stage effect, and LATE

# 35
library(AER)                      # load AER package
LATE=ivreg(Y~D|Z)                 # run two stage least squares regression
summary(LATE,vcov = vcovHC)       # results with heteroscedasticity-robust se

# 36
library(LARF)                     # load LARF package
library(causalweight)             # load causalweight package
data(c401k)                       # load 401(k) pension data
D=c401k[,3]                       # treatment: participation in pension plan
Z=c401k[,4]                       # instrument: eligibility for pension plan
Y=c401k[,2]                       # outcome: net financial assets in 1000 USD
X=as.matrix(c401k[,5:11])         # covariates
set.seed(1)                       # set seed
LATE=lateweight(y=Y, d=D, z=Z, x=X, boot=299)  # compute LATE (299 bootstraps)
```

```
LATE$effect; LATE$se.effect; LATE$pval.effect # show LATE results
LATE$first; LATE$se.first; LATE$pval.first # show first stage results

# 37
library(npcausal)                # load npcausal package
set.seed(1)                      # set seed
ivlate(y=Y, a=D, z=Z, x=X)       # estimate LATE by double machine learning

# 38
library(localIV)                                 # load localIV package
data(toydata)                                    # load toydata
D=toydata$d                                      # define binary treatment
Z=toydata$z                                      # define continuous instrument
Y=toydata$y                                      # define outcome
X=toydata$x                                      # define covariate
MTE=mte(selection=D~X+Z, outcome=Y~X)            # LIV estimation of MTE
MTEs=mte_at(u=seq(0.05, 0.95, 0.01), model=MTE)  # predict MTEs at mean of X
plot(x=MTEs$u,y=MTEs$value,xlab="p(Z, mean X)",ylab="MTE at mean X") #plot

# 39
library(wooldridge)                         # load wooldridge package
library(multiwayvcov)                       # load multiwayvcov package
library(lmtest)                             # load lmtest package
data(kielmc)                                # load kielmc data
attach(kielmc)                              # attach data
Y=rprice                                    # define outcome
D=nearinc                                   # define treatment group
T=y81                                       # define period dummy
interact=D*T                                # treatment-period interaction
did=lm(Y~D+T+interact)                      # DiD regression
vcovCL=cluster.vcov(model=did, cluster=cbd) # cluster: distance to center (cbd)
coeftest(did, vcov=vcovCL)                  # DiD results with cluster st.error

# 40
library(causalweight)                                # load causalweight package
X=cbind(area, rooms, baths)                          # define covariates
```

```
set.seed(1)                                                # set seed to 1
out=didweight(y=Y,d=D,t=T,x=X,boot=399,cluster=cbd) # DiD with cluster se
out$effect; out$se; out$pvalue                             # effect, se, and p-value

# 41
library(did)                                            # load did package
data(mpdta)                                             # load mpdta data
out=att_gt(yname="lemp", tname="year", gname="first.treat", idname="countyreal",
 xformla=~lpop, clustervars="countyreal", data=mpdta) # doubly robust did
summary(out)                                            # group-time-specific ATETs
ggdid(out)                                              # plot DiD results
meanATET=aggte(out)                                     # generate averages over ATETs
summary(meanATET)                                       # report averaged ATETs

# 42
library(qte)                                              # load qte package
library(wooldridge)                                       # load wooldridge package
data(kielmc)                                              # load kielmc data
cic=CiC(rprice~nearinc,t=1981,tmin1=1978,tname="year",data=kielmc) # run CiC
ggqte(cic)                                                # plot QTETs

# 43
library(devtools)                                           # load devtools package
install_github("synth-inference/synthdid")                  # install synthdid package
library(synthdid)                                           # load synthdid package
data(california_prop99)                                     # load smoking data
dat=panel.matrices(california_prop99)                       # prepare data
set.seed(1)                                                 # set seed
out=synthdid_estimate(Y=dat$Y, N0=dat$N0, T0=dat$T0) # synthetic DiD
se = sqrt(vcov(out, method='placebo'))                      # placebo standard error
out[1]; se                                                  # show results

# 44
set.seed(1)                                                 # set seed
out=synthdid_estimate(Y=dat$Y, N0=dat$N0,T0=dat$T0, omega.intercept=FALSE,
    weights=list(lambda=rep(0,dat$T0)))                     # synthetic control
```

```
se = sqrt(vcov(out, method='placebo'))              # placebo standard error
out[1]; se                                          # show results
plot(out)                                           # plot effects over time

# 45
library(rdrobust)                       # load rdrobust library
data(rdrobust_RDsenate)                 # data on elections for US Senate
Y=rdrobust_RDsenate$vote                # outcome is vote share of Democrats
R=rdrobust_RDsenate$margin              # running variable is margin of winning
results=rdrobust(y=Y, x=R)              # sharp RDD
summary(results)                        # show results
rdplot(y=Y, x=R)                        # plot outcome against running variable

# 46
library(rdd)                            # load rdd library
DCdensity(runvar=R)                     # run the McCrary (2008) sorting test

# 47
library(devtools)                   # load devtools package
install_github("kolesarm/RDHonest") # install RDHonest package
library(RDHonest)                   # load RDHonest package
data(rcp)                           # load rcp data
Y=rcp$cn                            # outcome is expenditures on non-durables
R=rcp$elig_year                     # running var based on eligibility to retire
D=rcp$retired                       # treatment is retirement status
results=rdrobust(y=Y, x=R, fuzzy=D) # fuzzy RDD
summary(results)                    # show results

# 48
library(haven)                                # load haven package
data=read_dta("C:/finaldata.dta")             # load data
Y=data$pers_total                             # define outcome (total personnel)
R=data$forcing                                # define running variable
D=data$costequalgrants                        # define treatment (grants)
results=rdrobust(y=Y, x=R, fuzzy=D, deriv=1)  # run fuzzy RKD
summary(results)                              # show results
```

```
# 49
library(bunching)              # load bunching package
data(bunching_data)            # load bunching data
Y=bunching_data$kink_vector    # define outcome (with bunching at value 10000)
set.seed(1)                    # set seed
b=bunchit(z_vector=Y,zstar=10000,binwidth=50,bins_l=20,bins_r=20,t0=0,t1=.2)#est
b$B; b$B_sd; b$plot            # show results

# 50
library(experiment)                       # load experiment package
library(causalweight)                     # load causalweight package
data(JC)                                  # load JC data
treat=JC$assignment                       # random treatment (assignment to JC)
outcome=JC$earny4                         # define outcome (earnings in 4. year)
selection=JC$pworky4>0                    # sample selection: employed in 4. year
outcome[selection==0]=NA                  # recode non-selected outcomes as NA
dat=data.frame(treat,selection,outcome)  # generate data frame
results=ATEbounds(outcome~factor(treat),data=dat) # compute worst case bounds
results$bounds; results$bonf.ci          # bounds on ATE + confidence intervals

# 51
library(devtools)                         # load devtools package
install_github("vsemenova/leebounds")    # install leebounds package
library(leebounds)                        # load leebounds package
results=leebounds(dat)                    # bounds (monotonic selection in treat)
results$lower_bound; results$upper_bound # bounds on ATE under monotonicity

# 52
library(rbounds)                            # load rbounds package
library(Matching)                           # load Matching package
data(lalonde)                               # load lalonde data
attach(lalonde)                             # store all variables in own objects
D=treat                                     # define treatment (training)
Y=re78                                      # define outcome
X=cbind(age,educ,nodegr,married,black,hisp,re74,re75,u74,u75) # covariates
```

```
set.seed(1)                                # set seed
output=Match(Y=Y, Tr=D, X=X, replace=FALSE)# pair matching (ATET),no replacement
hlsens(output, Gamma=2, GammaInc = 0.25)   # sensitivity analysis

# 53
library(devtools)                         # load devtools package
install_github("szonszein/interference") # install interference package
library(interference)                     # load interference package
data=read.csv("C:/india.csv")             # load data
data=na.omit(data)                        # drop observations with missings
group=data$village_id                     # cluster id
group_tr=data$mech                        # indicator high treatment proportion
indiv_tr=data$treat                       # individual treatment (insurance)
obs_outcome=data$EXPhosp_1                # outcome  (hospital expenditure)
dat=data.frame(group,group_tr,indiv_tr,obs_outcome) # generate data frame
estimates_hierarchical(dat)               # run estimation
```