```
#*************************************************************************
# Date: 12/01/2022
# Comment: Monte-Carlo Example for RDD bandwidth selector
# Used for: Jiawei Yang's Honor Thesis - Section 3
# Data File: N/A
# Data Source: Monte-Carlo, simulated based on Hill et al.(2018) pp350
#*************************************************************************

# Clear all variables and prior sessions
rm(list=ls(all=TRUE))

# Load packages and set seed
library('data.table')
library('ggplot2')
SEED_ID = 101

# Simulate dataset
  #running var: X ~ N(50, 25^2) for 500 obs
set.seed(SEED_ID)
x <- rnorm(500, mean = 50, sd = 25)
dt <- data.table(x = x)
  #parametric underlying data generating process
  #y = 100 - 3.6*x + 9/250*x^2
dt[, y:= 100 - 3.6*x + 9/250*x^2]
  #underlying Gaussian noise
  #y = f(x) + e such that e ~ N(0, 6^2)
set.seed(SEED_ID)
noise <- rnorm(200, mean = 0, sd = 6)
dt[, y_n:= y + noise]

# Initial visualization
ggplot(dt[y_n<200], aes(x = x, y = y_n, group = d)) +
  geom_point(size = 0.3) +
  geom_vline(xintercept=65, linetype="dashed", color = "black", alpha = 0.
6)+
  theme_classic()+
  xlab('X')+
  ylab('Y')+
  theme(text = element_text(family = "Times New Roman"))

# Assign treatment label: at threshold = 65
  #no underlying treatment effect: tau = 0
dt[, d := (x>=65)]

# Instantiate: regression results holder dataframe
dt_reg <- data.table(h = numeric(), rsq_l = numeric(), rsq_q = numeric(),
rsq_c = numeric())

#Compute fit proxy: raw r-squared
  # Fit regressions: single-sided polynomials - linear, quadratic, cubic
for( h in 3:65){
  #print(h)
  lm_q_temp <- summary(lm(y_n~poly(x,2, raw = TRUE), dt[d == 0 & x > 65-
h]))
  lm_l_temp <- summary(lm(y_n~poly(x,1, raw = TRUE), dt[d == 0 & x > 65-
h]))
  lm_c_temp <- summary(lm(y_n~poly(x,3, raw = TRUE), dt[d == 0 & x > 65-
h]))
```

```
  #print(lm_sum_temp$r.squared)
  dt_reg<-rbind(dt_reg, list(h, lm_l_temp$r.squared, lm_q_temp$r.squared, 
lm_c_temp$r.squared))
}

# Reg results: visual of r2 as a function of window sizes
ggplot(dt_reg, aes(x = h)) +
  geom_line(aes(y = rsq_l), size = 0.7, color = 'red') +
  geom_line(aes(y = rsq_q), size = 0.7, color = 'green') +
  geom_line(aes(y = rsq_c), size = 0.7, color = 'blue') +
  theme_classic()+
  xlab('Neighborhood window size (k)')+
  ylab('r-squared')+
  theme(text = element_text(family = "Times New Roman"))

# Compute locality score: percentage share of points included
s_count <- numeric()

for( h in 3:65){
  #print(nrow(dt[d==0 & x > 65-h]))
  s_count <- c(s_count, nrow(dt[d==0 & x > 65-h]))
}

# Compute aggregate objective score: Obj = locality * fit
dt_reg$s_count = s_count
control_ttl <- nrow(dt[d==0])
dt_reg[,pct_in_window := s_count/control_ttl]
dt_reg[,local_proxy := 1-pct_in_window]
dt_reg[,obj := local_proxy*rsq_q ]

# Visual: Objective as a function of window sizes
ggplot(dt_reg, aes(x = h)) +
  geom_line(aes(y = local_proxy), size = 0.5, linetype ='dashed', 
show.legend = TRUE) +
  geom_line(aes(y = rsq_q), size = 0.5, linetype ='dotdash', show.legend = 
TRUE) +
  geom_line(aes(y = obj), size = 0.5, linetype ='solid' , show.legend = 
TRUE) +
  scale_color_discrete(name = 'Linetypes')+
  theme_classic()+
  xlab('Neighborhood window size (k)')+
  ylab('Objective function and components')+
  geom_vline(xintercept=10, linetype="dashed", color = "red", alpha = 0.6)
+
  theme(text = element_text(family = "Times New Roman"))
```