Assignment 1

Tanner Bivins, Emmett Saulnier, Brock Wilson

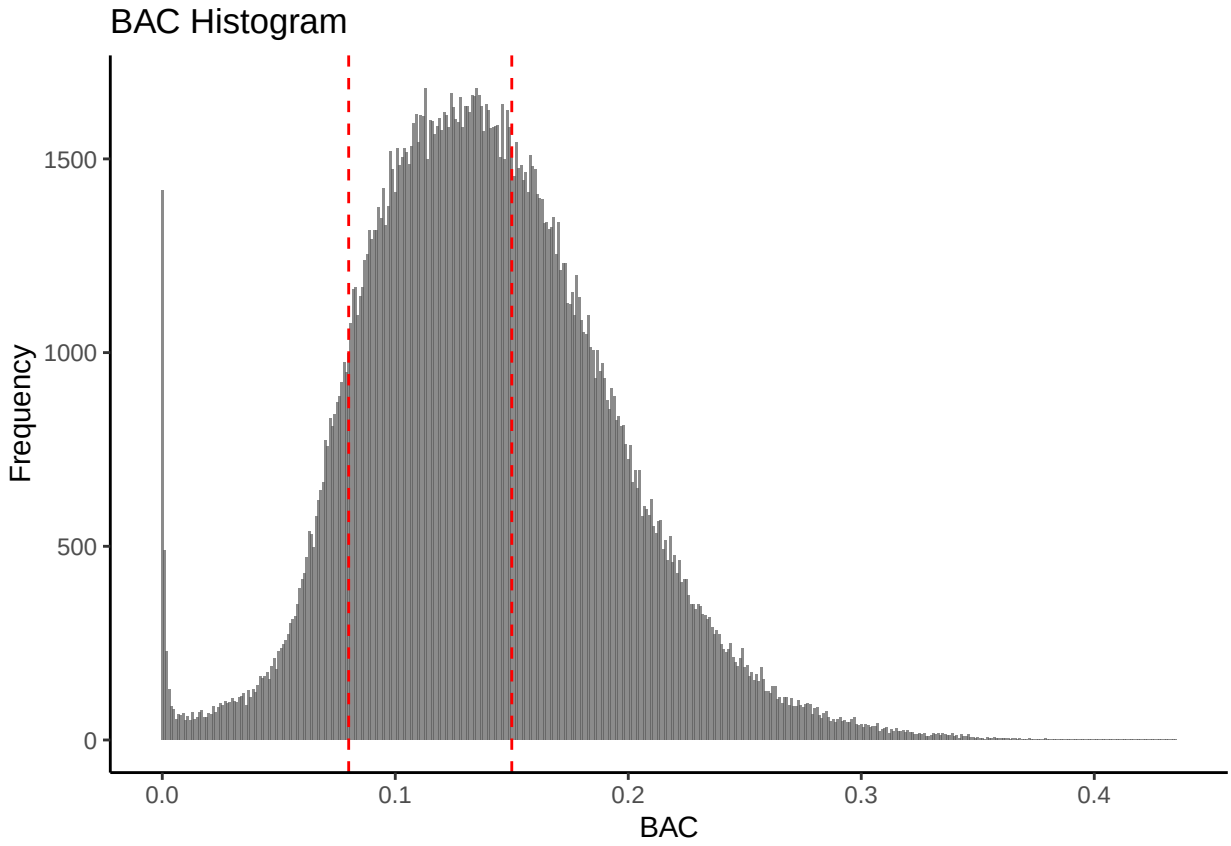
4/29/2021

For this problem you will estimate a regression discontinuity design to test whether having a BAC over the legal limits. You can compare this to the paper by yours truly about DUI punishments we read earlier.

Part A

Create a histogram of the running variable, BAC. Make sure you do it allowing for discrete bins. Is there evidence of clear sorting at the threshold?

```
# Loading Data
bac = data.table(read.csv(here("bac.csv")))

# Histogram of BAC
ggplot(data = bac, aes(x = bac)) +
  geom_histogram(binwidth = 0.001, alpha = 0.7) +
  geom_vline(xintercept = 0.08, linetype = "dashed", color = "red") +
  geom_vline(xintercept = 0.15, linetype = "dashed", color = "red") +
  labs(title ="BAC Histogram",
       x = "BAC",
       y = "Frequency") +
  theme_classic()
```



Solution

The histogram does not show evidence of sorting at either the 0.08 or 0.15 threshold, as the distribution is smooth on either side of both thresholds.

Part B

Get the R package rddensity. Perform a density test on the running variable. Is there evidence of sorting?

```
# Testing for sorting at both thresholds
rd_dens_08 = rddensity(X = bac$bac, c = 0.08, kernel = "uniform")
rd_dens_15 = rddensity(X = bac$bac, c = 0.15, kernel = "uniform")
```

Solution The density test does not provide evidence of sorting. For the 0.08 threshold, the t-stat is 1.00611060888731 with corresponding p-value of 0.314362365963106. For the 0.15 threshold, the t-stat is -0.00277848236185223 with corresponding p-value of 0.99778309467342. Thus, in both cases we fail to reject the null hypothesis that there is no sorting.

Part C

Next run a regression discontinuity model. To do so, create a dummy variable for a BAC over .08. Include that dummy variable, and the rescaled BAC (BAC-.08) as a control, and also include an interaction between that dummy variable and the running variable in model. First use age, gender, accident at the scene and race as outcomes. Do those factors shift at .08?

```
# Creating indicator variables for BAC over thresholds and rescaling
bac[,`:=`(
```

Table 1: Regression Discontinuity Estimates for the Impact of Exceeding DUI Threshold on Observable Characteristics

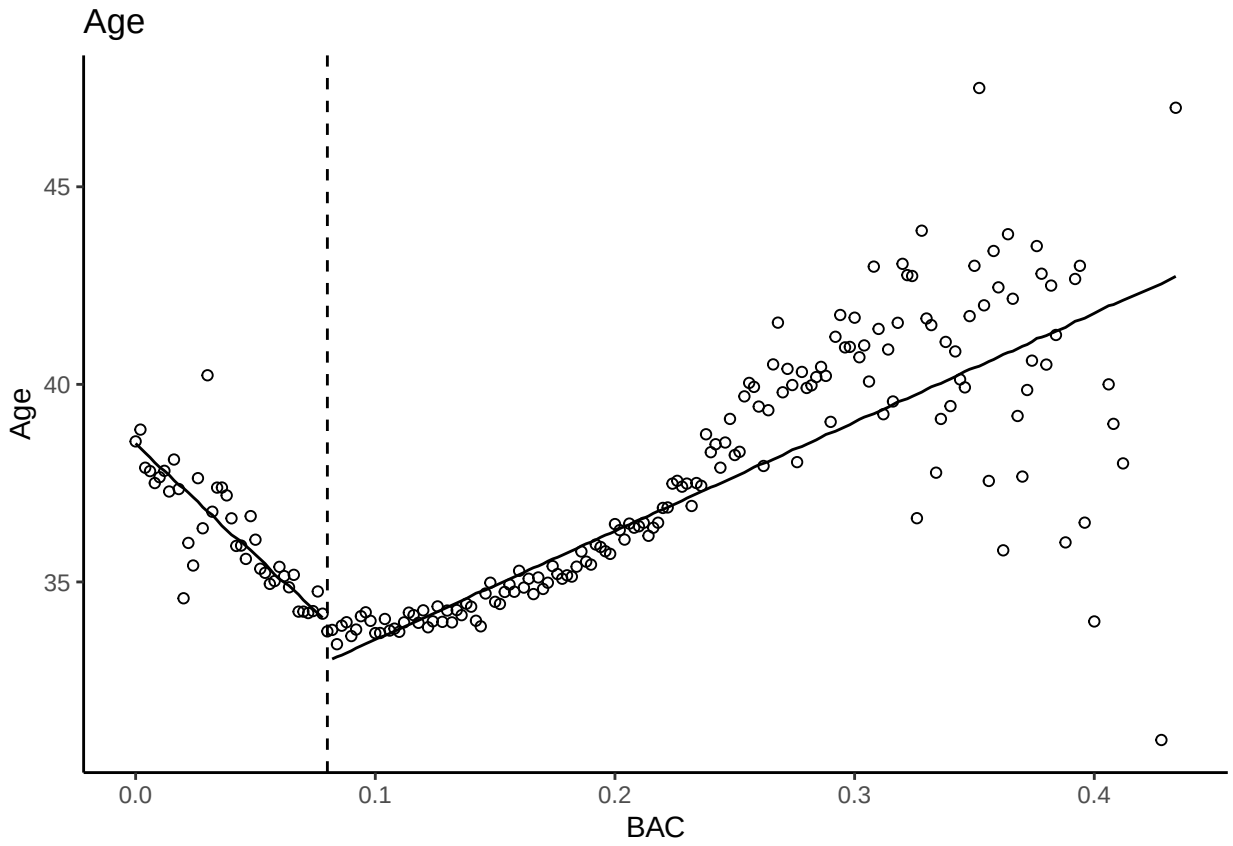
	Age	Male	Accident	White
(Intercept)	34.038***	0.791***	0.076***	0.849***
	(0.103)	(0.004)	(0.003)	(0.003)
DUI Threshold	-1.070***	0.006	-0.004	0.002
	(0.113)	(0.004)	(0.003)	(0.003)
BAC	-56.019***	0.210**	-1.545***	0.185**
	(2.977)	(0.106)	(0.091)	(0.090)
DUI × BAC	83.530***	-0.312***	2.674***	-0.010
	(3.032)	(0.108)	(0.093)	(0.092)
Num.Obs.	214558	214558	214558	214558
R2	0.012	0.000	0.021	0.001
R2 Adj.	0.012	0.000	0.021	0.001
F	904.630	10.264	1565.769	57.645

* p < 0.1, ** p < 0.05, *** p < 0.01

```
      drunk_08 = ifelse(bac > 0.08, 1, 0),
      drunk_15 = ifelse(bac > 0.15, 1, 0),
      rescaled_bac_08 = bac - 0.08,
      rescaled_bac_15 = bac - 0.15 )
]

# Regressions to check if there are differences in observables near threshold
obs_models_08 = list (
  "Age" = lm(data = bac, aged ~ drunk_08*rescaled_bac_08),
  "Male" = lm(data = bac, male ~ drunk_08*rescaled_bac_08),
  "Accident" = lm(data = bac, acc ~ drunk_08*rescaled_bac_08),
  "White" = lm(data = bac, white ~ drunk_08*rescaled_bac_08)
)


# Reporting regression results
modelsummary(
  obs_models_08,
  stars = TRUE,
  coef_rename = c("drunk_08"="DUI Threshold",
                  "rescaled_bac_08"="BAC",
                  "drunk_08:rescaled_bac_08" = "DUI × BAC"),
  gof_omit = 'AIC|BIC|Log.Lik',
  #coef_omit = "Intercept",
  title = "Regression Discontinuity Estimates for the Impact of Exceeding DUI Threshold on Observable C
)
```

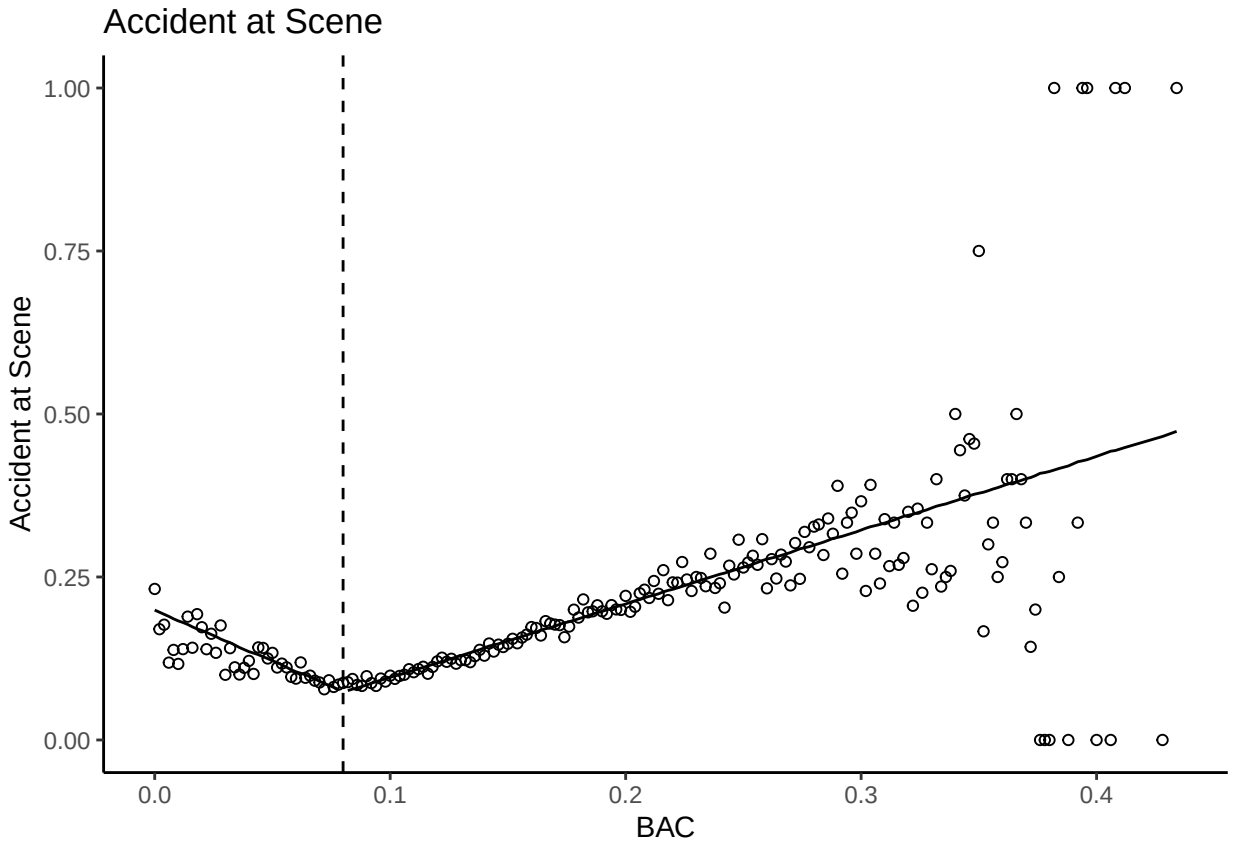
```
# Adding the fitted values for age, gender, and accident
bac$fitaged = predict(obs_models_08[["Age"]])
bac$fitmale = predict(obs_models_08[["Male"]])
bac$fitacc = predict(obs_models_08[["Accident"]])
bac$fitwhite = predict(obs_models_08[["White"]])
```

```
# Binning
binwidth = 0.002
for (x in seq(-0.082, 0.36, binwidth)) {
  bac[rescaled_bac_08 >= x & rescaled_bac_08 < x + binwidth, bac_bins_08 := x]
}

#bac$bac_bins_08 = cut(
#  x = bac$rescaled_bac_08,
#  breaks = seq(-0.082, 0.36, 0.002),
#  ordered_result = TRUE
#)


# Calculating averages for observables by bin
bac_average_08 = bac[,
    lapply(.SD, mean, na.rm =TRUE),
    .SDcols = c("aged","male","acc","white", "fitaged", "fitmale", "fitacc","fitwhite"),
    by = bac_bins_08]

bac_average_08[, cont_bin := bac_bins_08]


# Function that will generate RD plots
# Note that you must use "quote(dep_var)" in arguments for it to work
rd_plot = function(data, dep_var, fit_var, threshold) {
  p = ggplot(data = data, aes(x = cont_bin + threshold, y = eval(dep_var))) +
        geom_point(shape = 1) +
        geom_line(data = data[cont_bin < 0,], aes(y = eval(fit_var))) +
        geom_line(data = data[cont_bin > 0,], aes(y = eval(fit_var))) +
        xlab("BAC")+
        theme_classic() +
        geom_vline(xintercept = threshold, linetype = "dashed")
  return(p)
}

# Age
rd_plot(bac_average_08, quote(aged), quote(fitaged), 0.08)+
        labs(title = "Age", y = "Age")
```

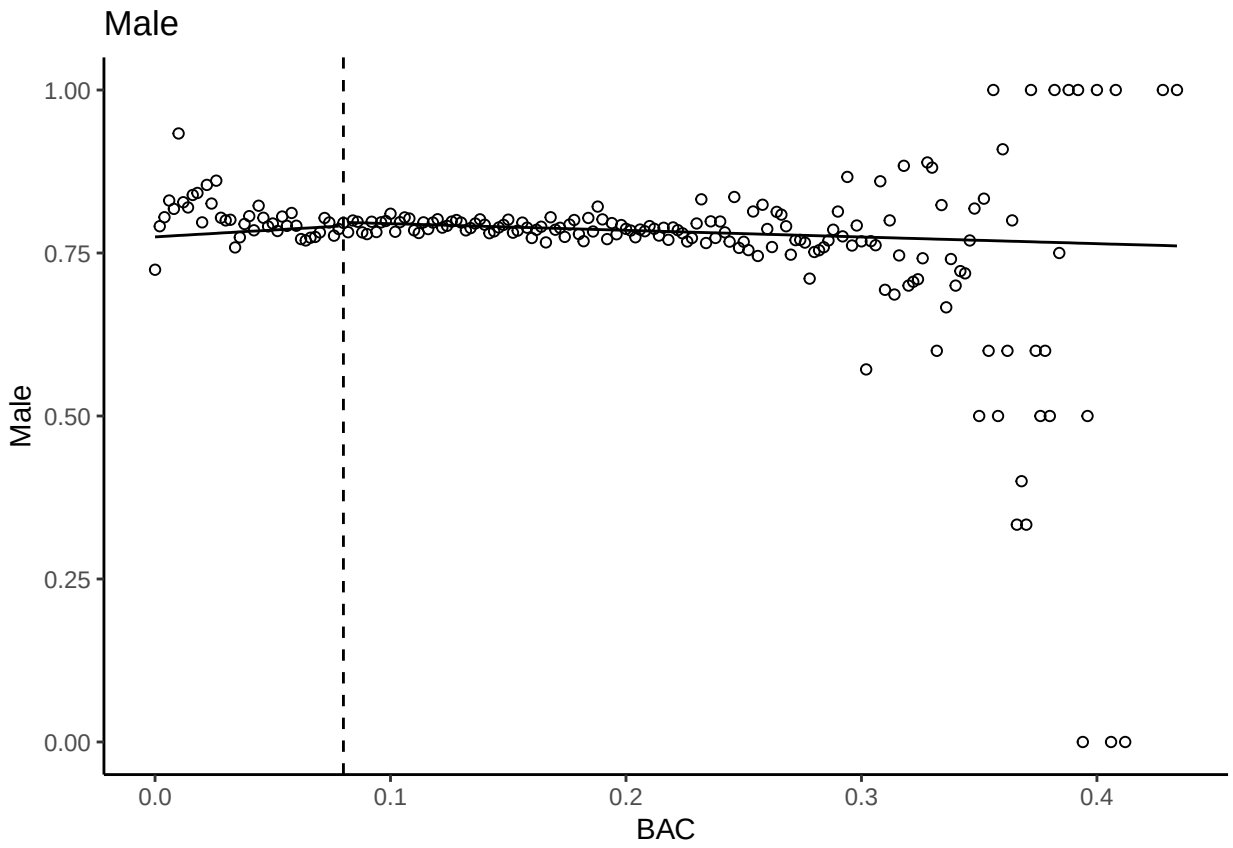


Solution

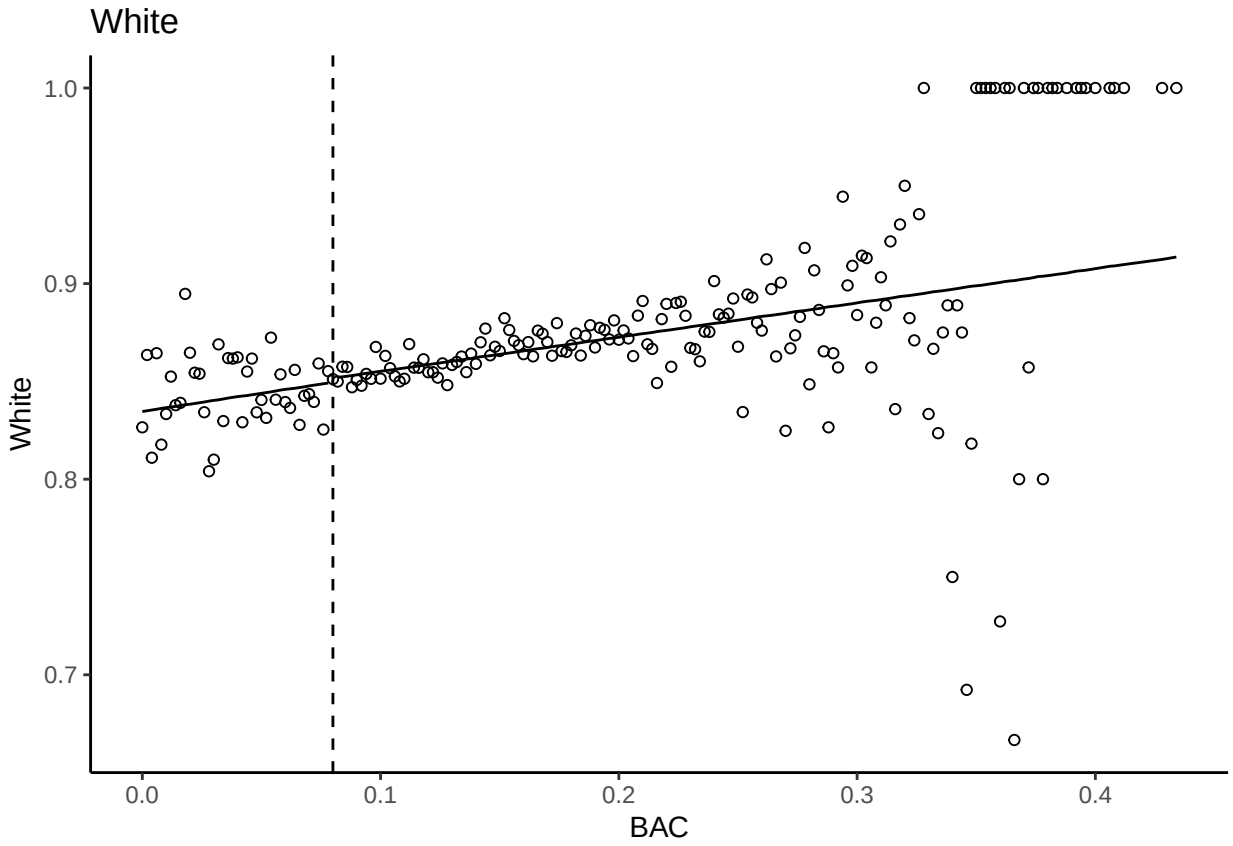
```
# Accident
rd_plot(bac_average_08, quote(acc), quote(fitacc), 0.08)+
        labs(title = "Accident at Scene", y = "Accident at Scene")
```




```
# Male
rd_plot(bac_average_08, quote(male), quote(fitmale), 0.08)+
        labs(title = "Male", y = "Male")
```




```
# White
rd_plot(bac_average_08, quote(white), quote(fitwhite), 0.08)+
        labs(title = "White", y = "White")
```



There is evidence that individuals who had a BAC above the DUI threshold (BAC > 0.08) are slightly younger than those below the threshold. Otherwise, there is no evidence of differences in gender, accident at the scene, or race.

The results for age seem to be driven by observations at the high end of the BAC distribution, and would likely disappear if the data was censored to be within a reasonable bandwidth of the threshold.

Part D

Now run a regression of recidivism on the same regression discontinuity design. What is your estimated effect using a bandwidth of .05, and a rectangular kernel (no weighting). Create a visualization of this by graphing the mean recidivism rate against the running variable. Show this for the whole BAC distribution, and the range from .03 to .13. Please include a fitted line.

```
# Censoring data using bandwidth of 0.05
bac_censored_08 = bac[rescaled_bac_08 <= 0.05 & rescaled_bac_08 >= -0.05,]

# Running RD models with and without controls
rd_models_08 = list(
  "No Controls" = lm(data = bac_censored_08,
           recidivism ~ drunk_08*rescaled_bac_08),
  "With Controls" = lm(data = bac_censored_08,
           recidivism ~ drunk_08*rescaled_bac_08 + aged + male + acc)
)

# Reporting regression results
modelsummary(
```

Table 2: Regression Discontinuity Estimates for the Impact of Exceeding DUI Threshold on Recidivism

	No Controls	With Controls
(Intercept)	0.112***	0.115***
	(0.003)	(0.005)
DUI Threshold	-0.019***	-0.019***
	(0.004)	(0.004)
BAC	-0.261	-0.308*
	(0.177)	(0.177)
DUI × BAC	0.684***	0.731***
	(0.195)	(0.195)
aged		-0.001***
		(0.000)
male		0.033***
		(0.002)
acc		0.003
		(0.003)
Num.Obs.	93792	93792
R2	0.001	0.003
R2 Adj.	0.001	0.003
F	16.739	52.018

* p < 0.1, ** p < 0.05, *** p < 0.01

```
  rd_models_08,
  stars = TRUE,
  coef_rename = c("drunk_08"="DUI Threshold",
                  "rescaled_bac_08"="BAC",
                  "drunk_08:rescaled_bac_08" = "DUI × BAC"),
  gof_omit = 'AIC|BIC|Log.Lik',
  #coef_omit = "Intercept",
  title = "Regression Discontinuity Estimates for the Impact of Exceeding DUI Threshold on Recidivism"
)
```

```
# Getting fitted values
bac_censored_08$fitrecid = predict(rd_models_08[["No Controls"]])

#### Graphing mean recidivism
bac_average = bac[,
    lapply(.SD, mean, na.rm =TRUE),
    .SDcols = c("recidivism"),
    by = bac_bins_08]

# Fitted data
bac_average2 = bac_censored_08[,
    lapply(.SD, mean, na.rm =TRUE),
    .SDcols = c("fitrecid"),
    by = bac_bins_08]

# Merging with rest of the data
bac_avg = merge(bac_average_08,bac_average, by="bac_bins_08") %>%
```

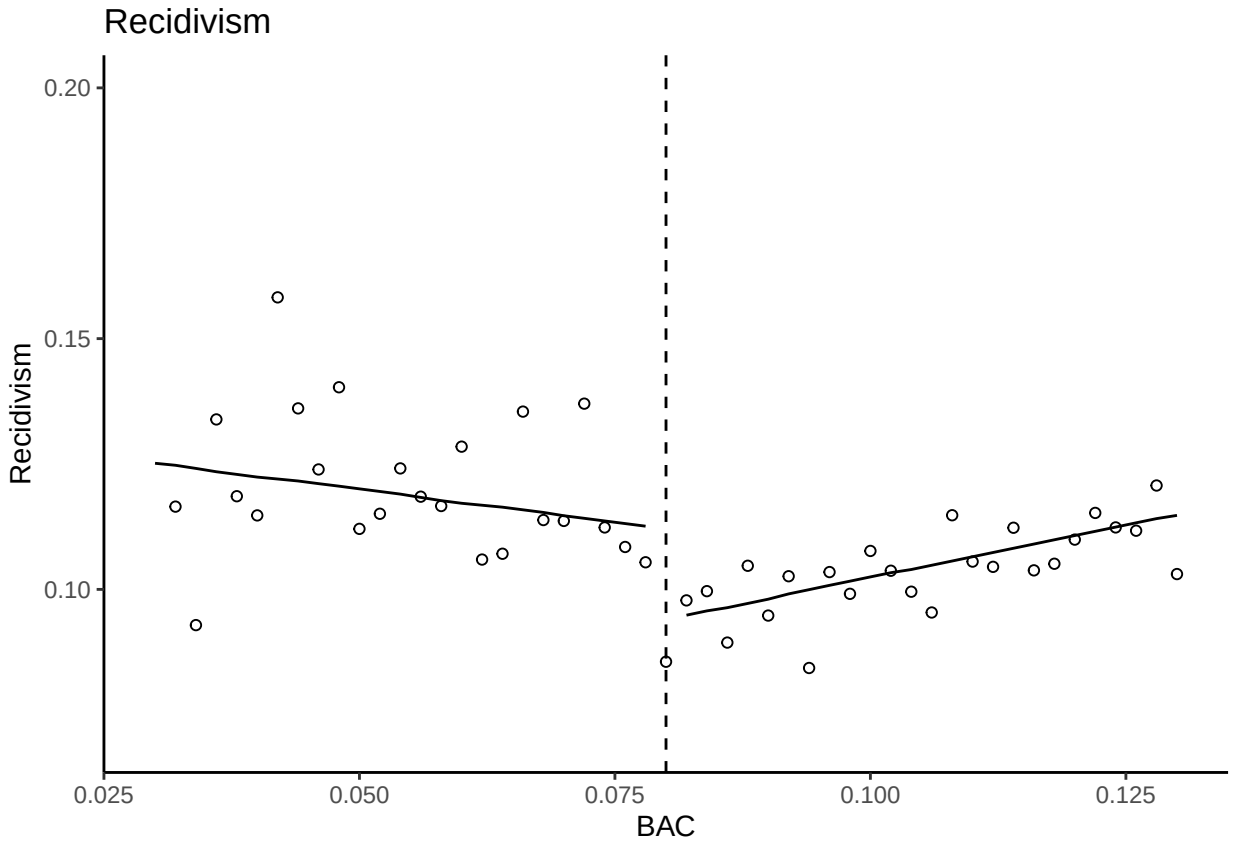
```
  merge(bac_average2, by="bac_bins_08", all.x = TRUE)


# Plotting the RD
rd_plot(bac_avg, quote(recidivism), quote(fitrecid), 0.08) +
  xlim(0.03,0.13) +
  ylim(0.07,0.2) +
  labs(title = "Recidivism", y = "Recidivism")
```

Solution

```
## Warning: Removed 153 rows containing missing values (geom_point).
```

```
## Warning: Removed 15 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 137 row(s) containing missing values (geom_path).
```

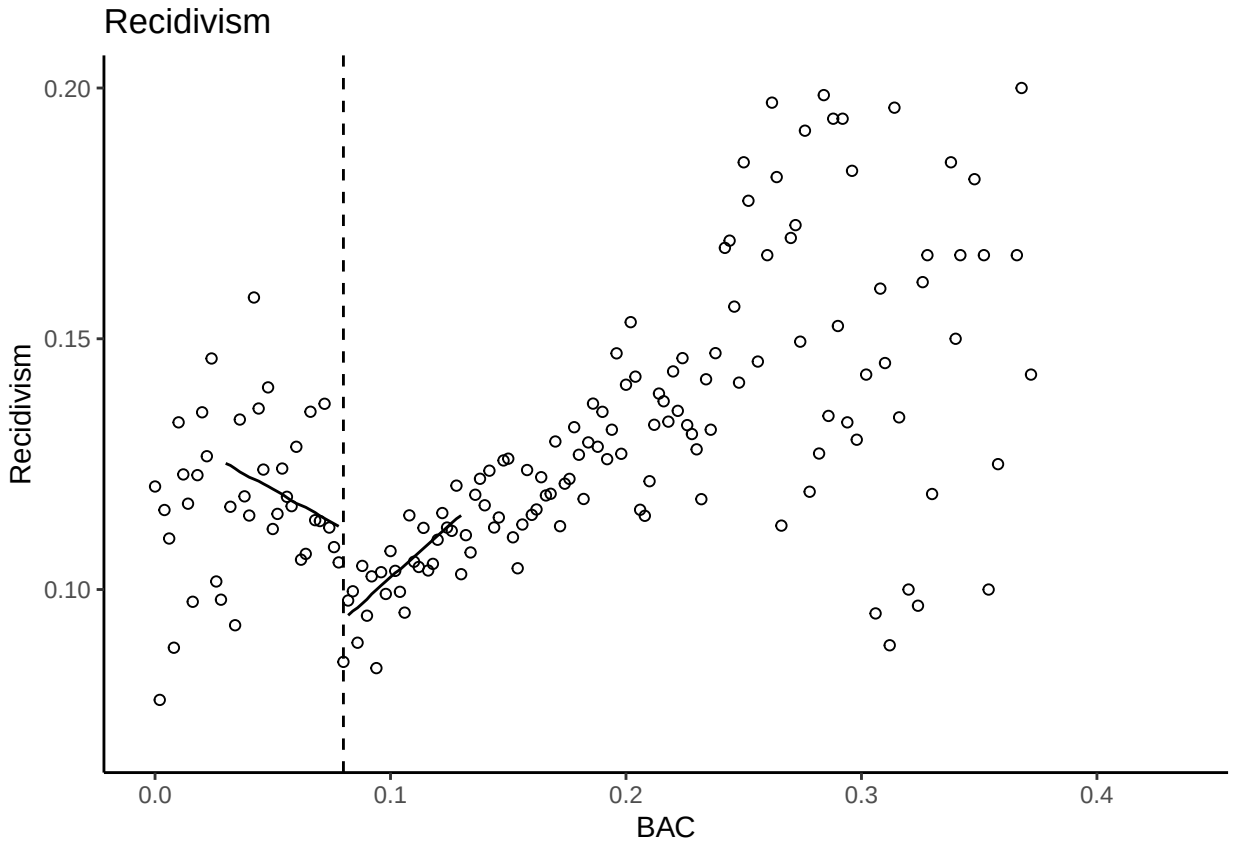



```
rd_plot(bac_avg, quote(recidivism), quote(fitrecid), 0.08) +
  ylim(0.07,0.2) +
  labs(title = "Recidivism", y = "Recidivism")
```

```
## Warning: Removed 37 rows containing missing values (geom_point).
```

```
## Warning: Removed 15 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 137 row(s) containing missing values (geom_path).
```



Part E

Do the same thing as part c and d but for the aggravated threshold of .151.

```
obs_models_15 = list (
  "Age" = lm(data = bac, aged ~ drunk_15*rescaled_bac_15),
  "Male" = lm(data = bac, male ~ drunk_15*rescaled_bac_15),
  "Accident" = lm(data = bac, acc ~ drunk_15*rescaled_bac_15),
  "White" = lm(data = bac, white ~ drunk_15*rescaled_bac_15)
)


# Reporting regression results
modelsummary(
  obs_models_15,
  stars = TRUE,
  coef_rename = c("drunk_15"="Aggr DUI Threshold",
                  "rescaled_bac_15"="BAC",
                  "drunk_15:rescaled_bac_15" = "Aggr DUI × BAC"),
  gof_omit = 'AIC|BIC|Log.Lik',
  #coef_omit = "Intercept",
  title = "Regression Discontinuity Estimates for the Impact of Exceeding Aggrevated DUI Threshold on O
)
```

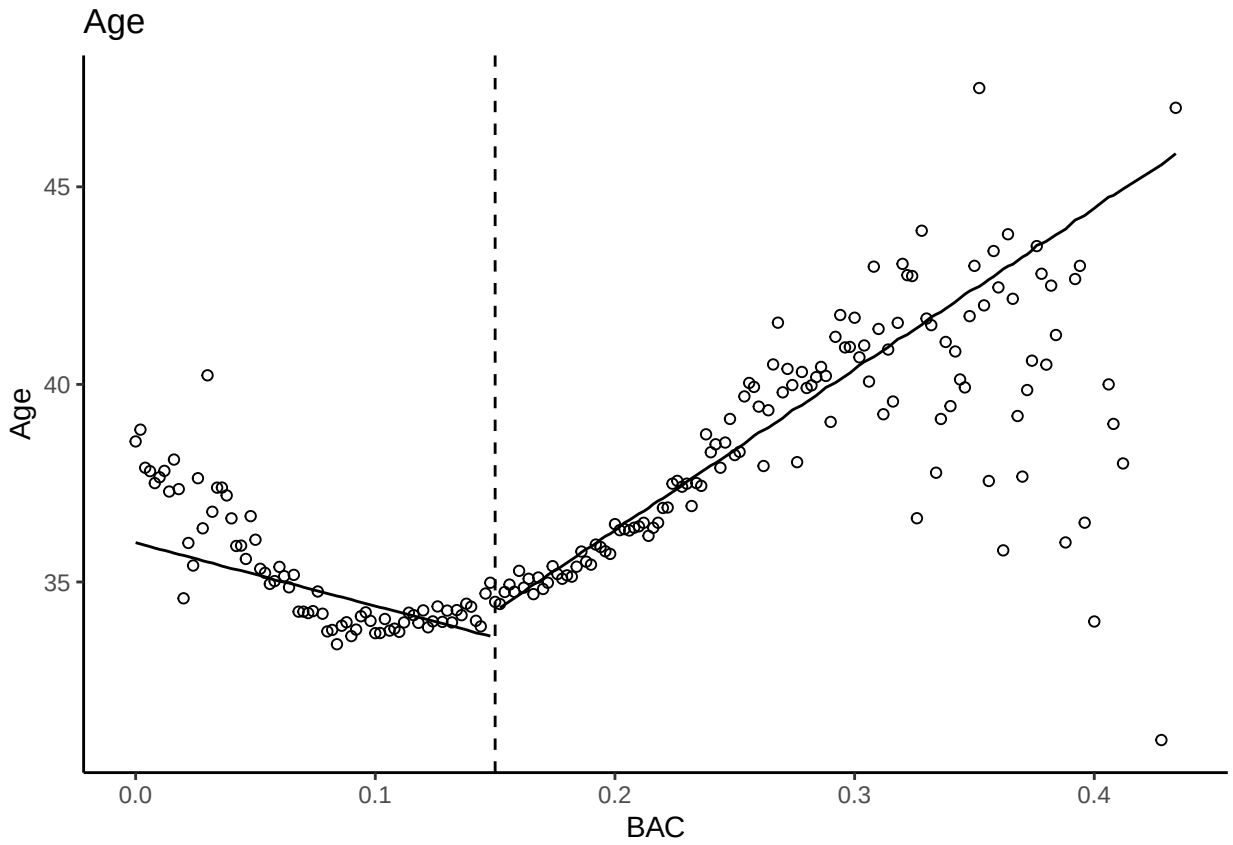
Table 3: Regression Discontinuity Estimates for the Impact of Exceeding Aggrevated DUI Threshold on Observable Characteristics

	Age	Male	Accident	White
(Intercept)	33.607***	0.796***	0.116***	0.864***
	(0.056)	(0.002)	(0.002)	(0.002)
Aggr DUI Threshold	0.638***	-0.004	0.038***	0.005*
	(0.082)	(0.003)	(0.003)	(0.002)
BAC	-15.952***	0.084**	0.082***	0.200***
	(1.010)	(0.036)	(0.031)	(0.030)
Aggr DUI × BAC	56.637***	-0.230***	1.017***	-0.112**
	(1.511)	(0.054)	(0.046)	(0.046)
Num.Obs.	214558	214558	214558	214558
R2	0.012	0.000	0.019	0.001
R2 Adj.	0.012	0.000	0.019	0.001
F	846.341	9.762	1390.344	60.543

* p < 0.1, ** p < 0.05, *** p < 0.01

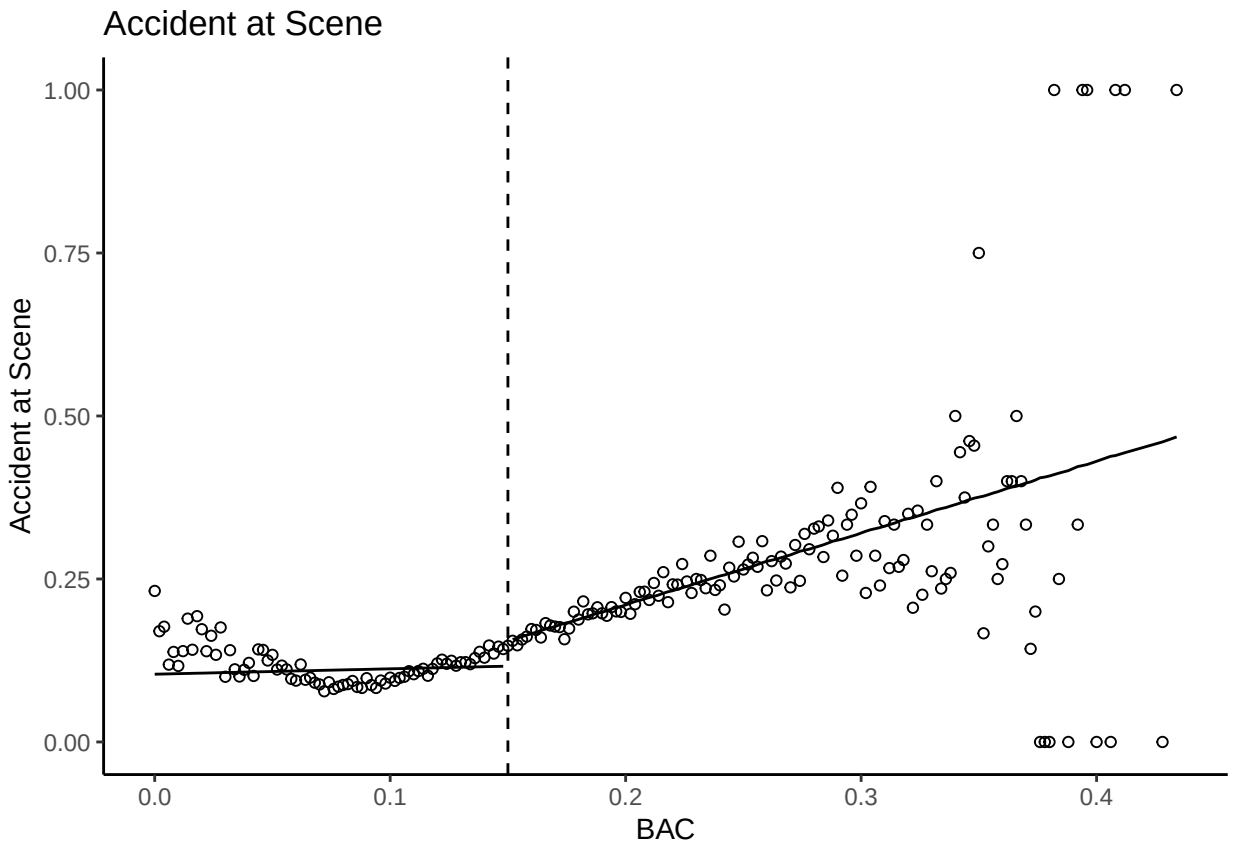
```
# Adding the fitted values for age, gender, and accident
bac$fitaged_15 = predict(obs_models_15[["Age"]])
bac$fitmale_15 = predict(obs_models_15[["Male"]])
bac$fitacc_15 = predict(obs_models_15[["Accident"]])
bac$fitwhite_15 = predict(obs_models_15[["White"]])


# Binning
binwidth = 0.002
for (x in seq(-0.152, 0.29, binwidth)) {
  bac[rescaled_bac_15 >= x & rescaled_bac_15 < x + binwidth, bac_bins_15 := x]
}

# Calculating averages for observables by bin
bac_average_15 = bac[,
    lapply(.SD, mean, na.rm =TRUE),
    .SDcols = c("aged","male","acc","white", "fitaged_15", "fitmale_15", "fitacc_15","fitwhite_15"),
    by = bac_bins_15]

bac_average_15[, cont_bin := bac_bins_15]


# Function that will generate RD plots
# Note that you must use "quote(dep_var)" in arguments for it to work
rd_plot = function(data, dep_var, fit_var, threshold) {
  p = ggplot(data = data, aes(x = cont_bin + threshold, y = eval(dep_var))) +
        geom_point(shape = 1) +
        geom_line(data = data[cont_bin < 0,], aes(y = eval(fit_var))) +
        geom_line(data = data[cont_bin > 0,], aes(y = eval(fit_var))) +
        xlab("BAC")+
        theme_classic() +
        geom_vline(xintercept = threshold, linetype = "dashed")
  return(p)
```

```
}

# Age
rd_plot(bac_average_15, quote(aged), quote(fitaged_15), 0.15)+
        labs(title = "Age", y = "Age")
```

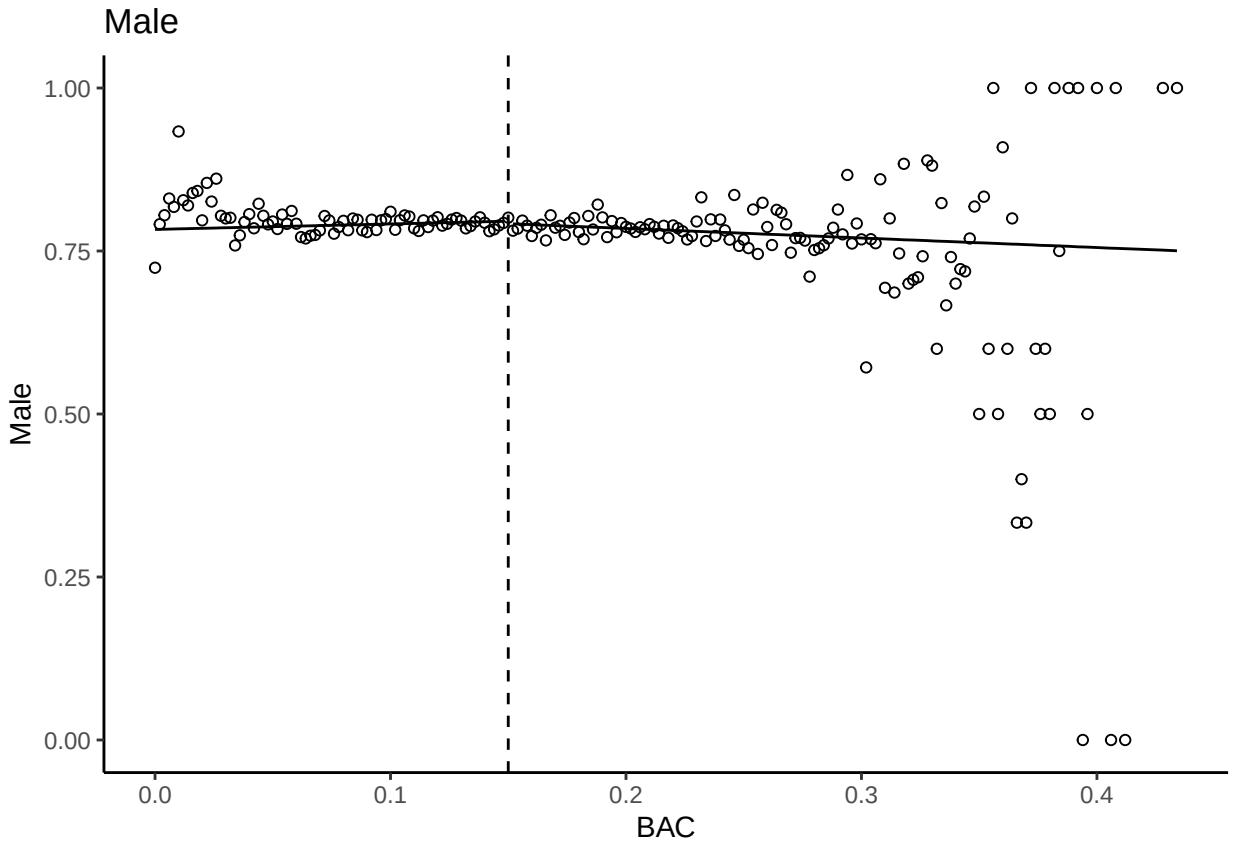


Solution

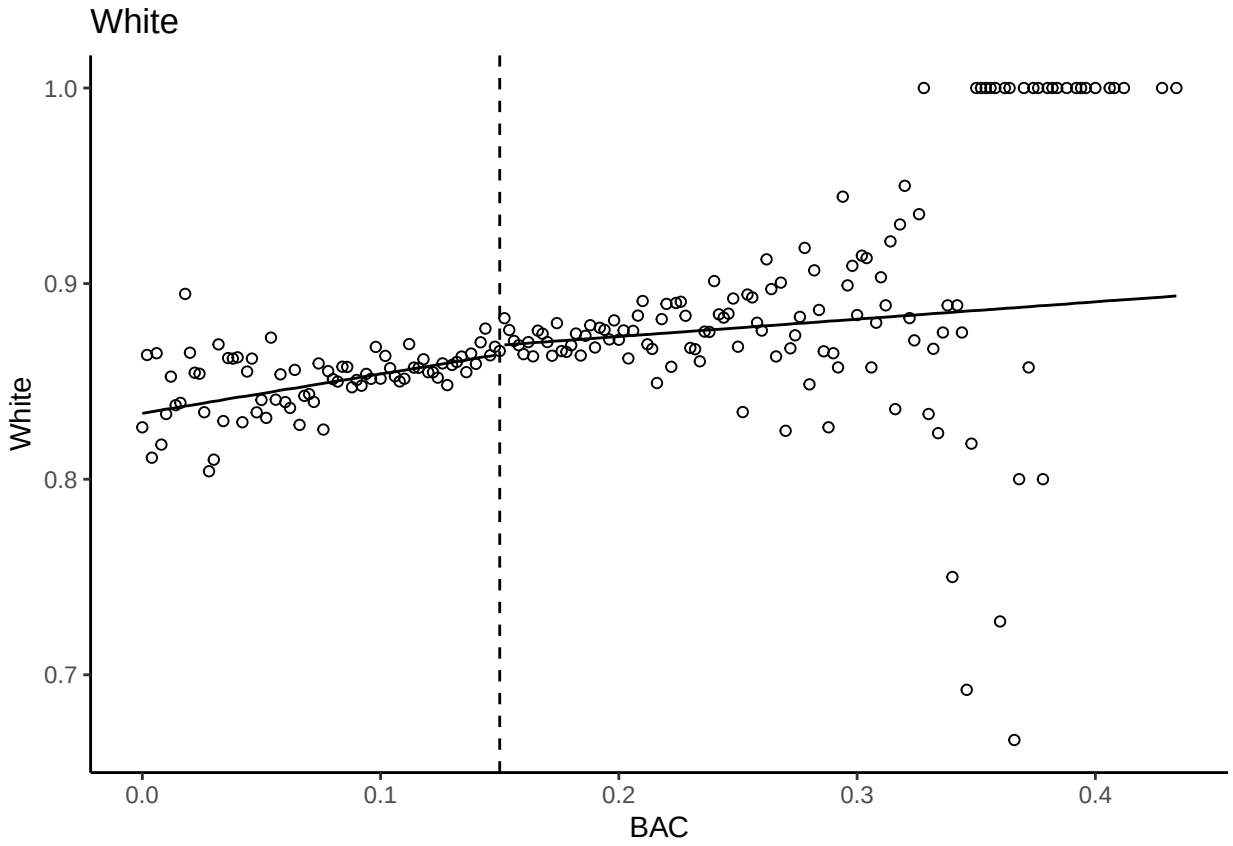
```
# Accident
rd_plot(bac_average_15, quote(acc), quote(fitacc_15), 0.15)+
        labs(title = "Accident at Scene", y = "Accident at Scene")
```




```
# Male
rd_plot(bac_average_15, quote(male), quote(fitmale_15), 0.15)+
        labs(title = "Male", y = "Male")
```




```
# White
rd_plot(bac_average_15, quote(white), quote(fitwhite_15), 0.15)+
        labs(title = "White", y = "White")
```




```
################################################
############ Effect on Recidivism ##############
################################################

# Censoring data using bandwidth of 0.05
bac_censored_15 = bac[rescaled_bac_15 <= 0.05 & rescaled_bac_15 >= -0.05,]

# Running RD models with and without controls
rd_models_15 = list(
  "No Controls" = lm(data = bac_censored_15,
           recidivism ~ drunk_15*rescaled_bac_15),
  "With Controls" = lm(data = bac_censored_15,
           recidivism ~ drunk_15*rescaled_bac_15 + aged + male + acc)
)

# Reporting regression results
modelsummary(
  rd_models_15,
  stars = TRUE,
  coef_rename = c("drunk_15"="Aggr DUI Threshold",
                  "rescaled_bac_15"="BAC",
                  "drunk_15:rescaled_bac_15" = "Aggr DUI × BAC"),
  gof_omit = 'AIC|BIC|Log.Lik',
  #coef_omit = "Intercept",
  title = "Regression Discontinuity Estimates for the Impact of Exceeding Aggrevated DUI Threshold on R
)
```

Table 4: Regression Discontinuity Estimates for the Impact of Exceeding Aggrevated DUI Threshold on Recidivism

	No Controls	With Controls
(Intercept)	0.121***	0.124***
	(0.002)	(0.004)
Aggr DUI Threshold	-0.008**	-0.007**
	(0.003)	(0.003)
BAC	0.364***	0.389***
	(0.079)	(0.079)
Aggr DUI × BAC	0.063	0.066
	(0.123)	(0.123)
aged		-0.001***
		(0.000)
male		0.029***
		(0.002)
acc		-0.008***
		(0.002)
Num.Obs.	138308	138308
R2	0.001	0.003
R2 Adj.	0.001	0.003
F	27.137	62.259

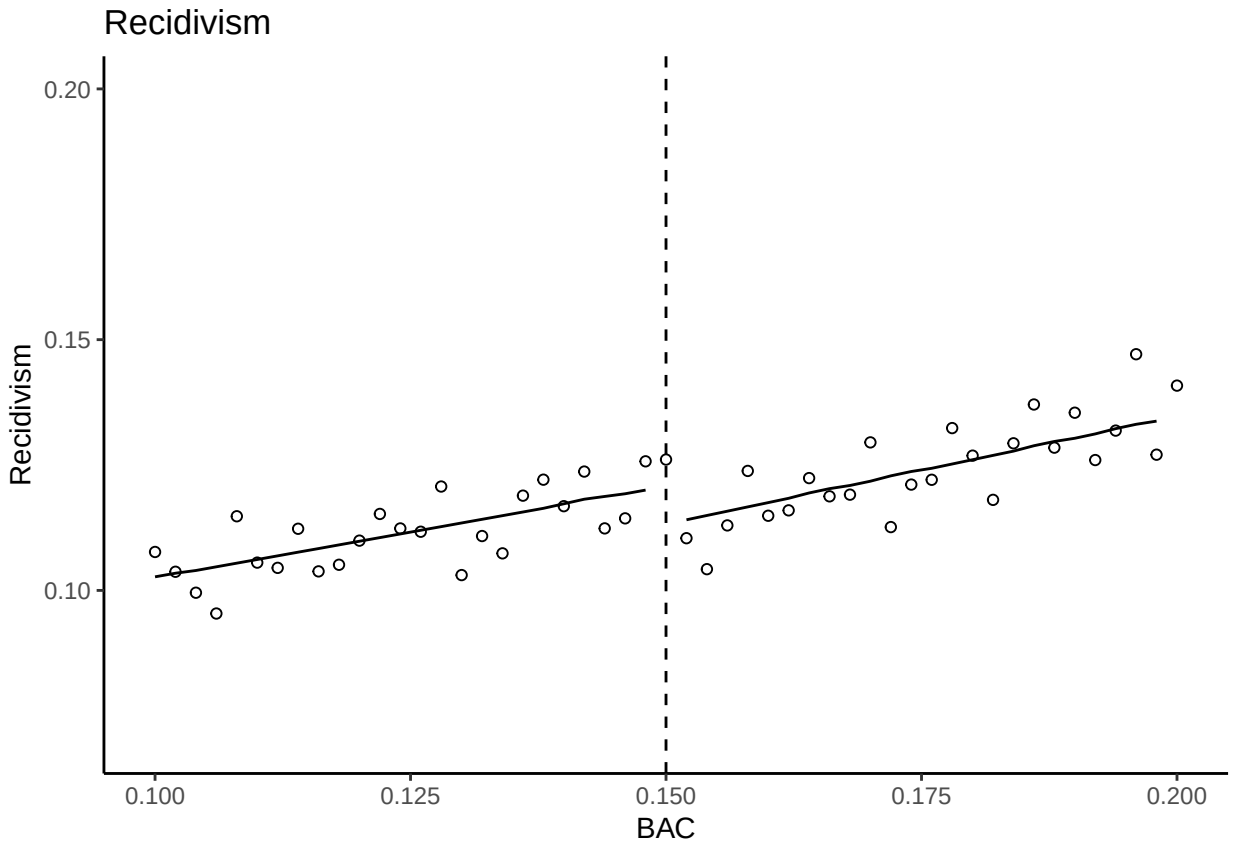
* p < 0.1, ** p < 0.05, *** p < 0.01

```
# Getting fitted values
bac_censored_15$fitrecid_15 = predict(rd_models_15[["No Controls"]])

#### Graphing mean recidivism
bac_average = bac[,
    lapply(.SD, mean, na.rm =TRUE),
    .SDcols = c("recidivism"),
    by = bac_bins_15]

# Fitted data
bac_average2 = bac_censored_15[,
    lapply(.SD, mean, na.rm =TRUE),
    .SDcols = c("fitrecid_15"),
    by = bac_bins_15]

# Merging with rest of the data
bac_avg = merge(bac_average_15,bac_average, by="bac_bins_15") %>%
  merge(bac_average2, by="bac_bins_15", all.x = TRUE)


# Plotting the RD
rd_plot(bac_avg, quote(recidivism), quote(fitrecid_15), 0.15) +
  xlim(0.10,0.20) +
  ylim(0.07,0.2) +
  labs(title = "Recidivism", y = "Recidivism")
```

```
## Warning: Removed 152 rows containing missing values (geom_point).
```

```
## Warning: Removed 50 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 103 row(s) containing missing values (geom_path).
```

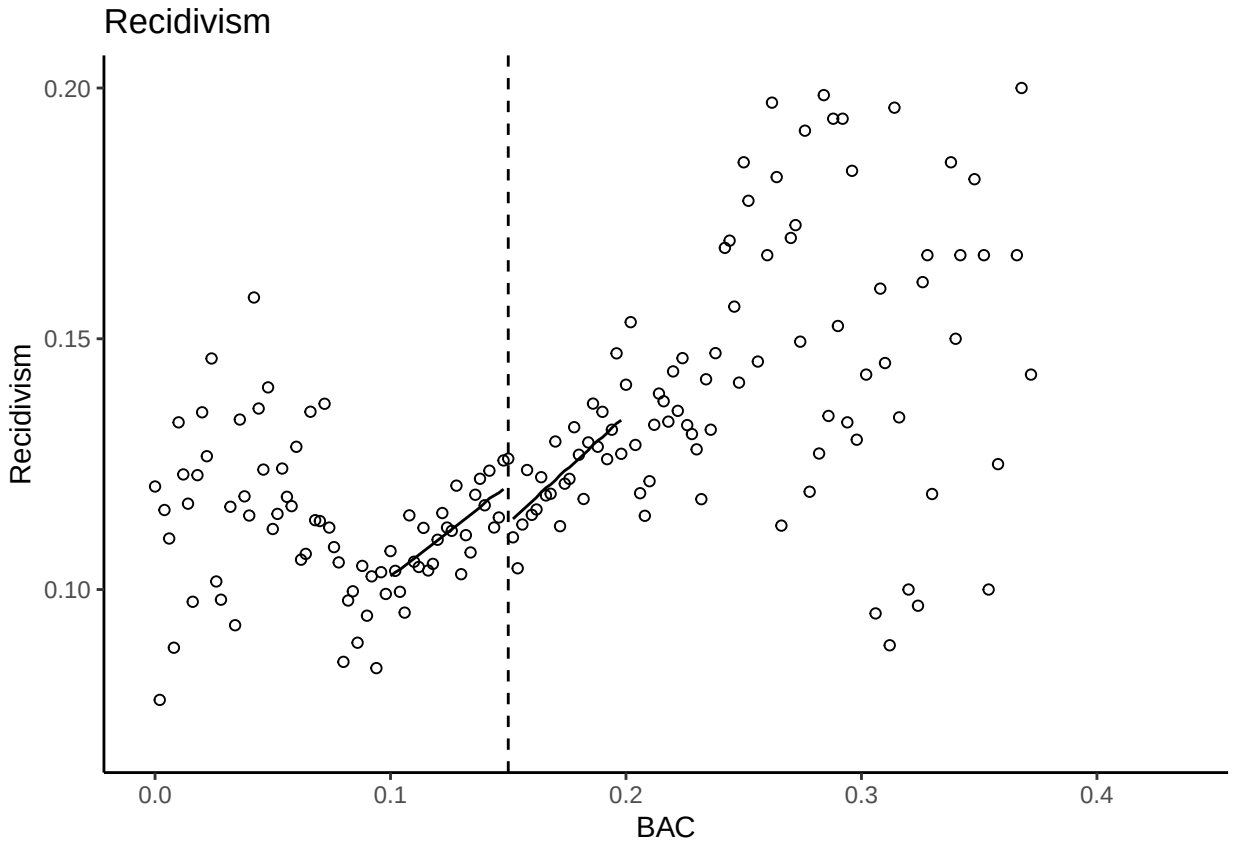



```
rd_plot(bac_avg, quote(recidivism), quote(fitrecid_15), 0.15) +
  ylim(0.07,0.2) +
  labs(title = "Recidivism", y = "Recidivism")
```

```
## Warning: Removed 37 rows containing missing values (geom_point).
```

```
## Warning: Removed 50 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 103 row(s) containing missing values (geom_path).
```

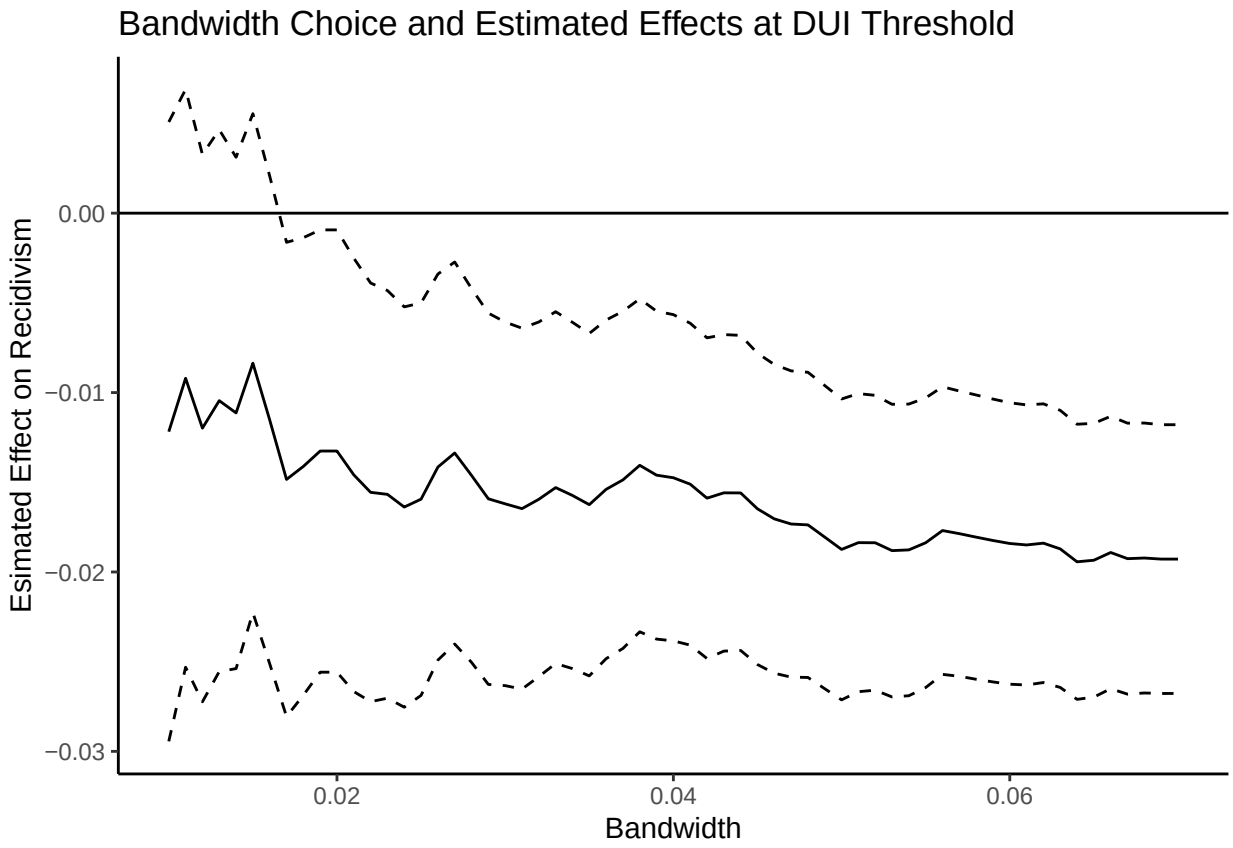


Part F

Now run this model for every possible bandwidth between .01 and .07. Store both the point estimates and lower and upper confidence intervals. Create a scatter plot of the confidence interval and the point estimates. Are the estimates robust? Create a visualization of this.

```
# Writing a function that estimates coefficients for given bandwidth and threshold
rd_bw = function(bw, threshold){

  # Rescaling and creating indicator based on threshold
  bac$rescaled_bac = bac$bac - threshold
  bac$drunk = ifelse(bac$rescaled_bac > 0, 1, 0)

  # Censoring data to within bandwidth
  bac2 = bac[rescaled_bac <= bw & rescaled_bac >= -bw,]

  # Running regression model
  mod_recid = lm(data = bac2, recidivism ~ drunk*rescaled_bac)

  # Collecting results
  estimate = broom::tidy(mod_recid)[2,2]
  t_stat = broom::tidy(mod_recid)[2,4]
  lower_bound = broom::tidy(mod_recid)[2,2] - 2* broom::tidy(mod_recid)[2,3]
  upper_bound = broom::tidy(mod_recid)[2,2] + 2* broom::tidy(mod_recid)[2,3]

  # Returning the results
```

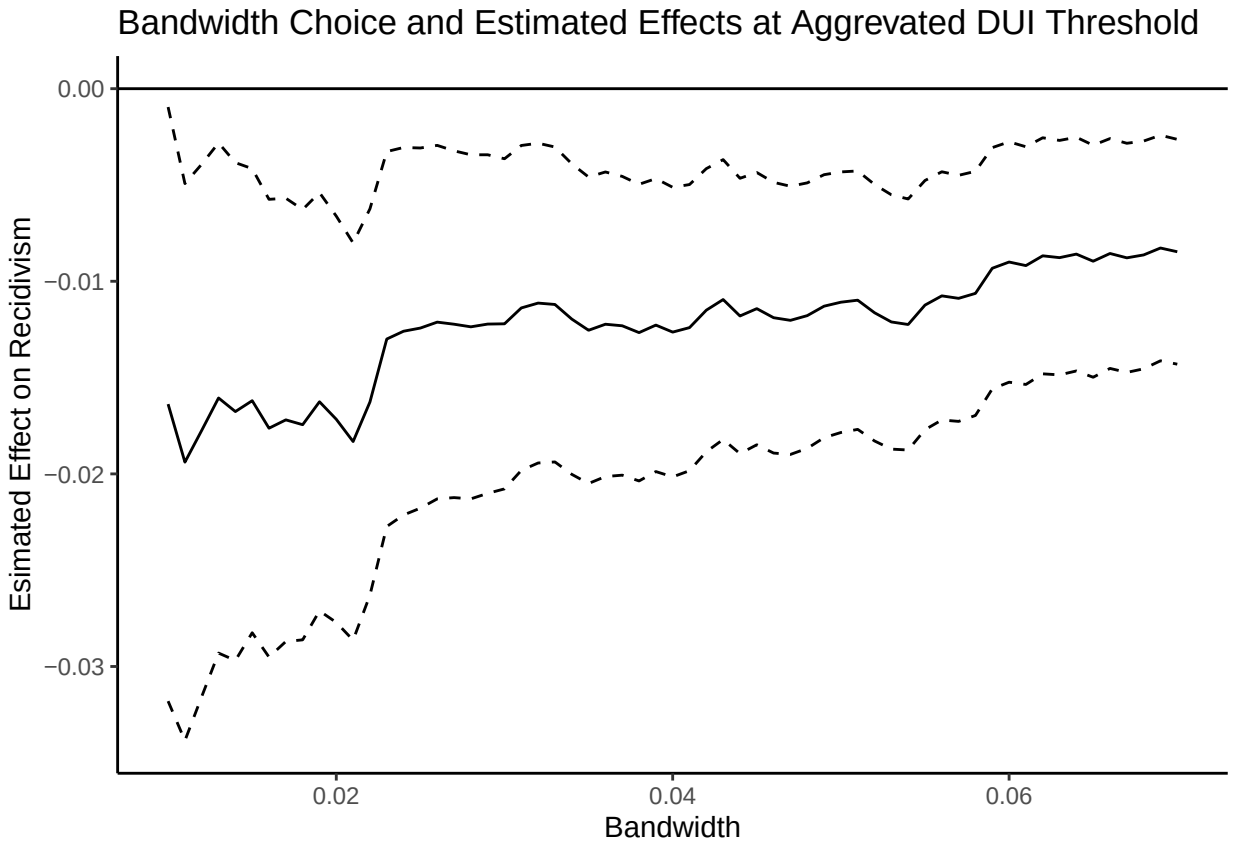
```
  output = data.frame(threshold, bw, estimate, lower_bound, upper_bound, t_stat)
  return(output)

}

# Threshold = 0.08
bandwidth_08 = map_dfr(.x = seq(0.01, 0.07, 0.001), .f = rd_bw, threshold = 0.08)
colnames(bandwidth_08) = c("threshold", "bw", "estimate", "lower_bound", "upper_bound")

# Plotting Estimates
ggplot(data = bandwidth_08, aes(x = bw, y = estimate)) +
  geom_line() +
  geom_line(aes(y = upper_bound), linetype = "dashed") +
  geom_line(aes(y = lower_bound), linetype = "dashed") +
  geom_hline(yintercept = 0)+
  theme_classic() +
  labs(title = "Bandwidth Choice and Estimated Effects at DUI Threshold",
       x = "Bandwidth",
       y = "Esimated Effect on Recidivism")
```



Solution

```
# Threshold = 0.151
bandwidth_15 = map_dfr(.x = seq(0.01, 0.07, 0.001), .f = rd_bw, threshold = 0.151)
colnames(bandwidth_15) = c("threshold", "bw", "estimate", "lower_bound", "upper_bound")


# Plotting Estimates
ggplot(data = bandwidth_15, aes(x = bw, y = estimate)) +
```

```
  geom_line() +
  geom_line(aes(y = upper_bound), linetype = "dashed") +
  geom_line(aes(y = lower_bound), linetype = "dashed") +
  geom_hline(yintercept = 0)+
  theme_classic() +
  labs(title = "Bandwidth Choice and Estimated Effects at Aggrevated DUI Threshold",
       x = "Bandwidth",
       y = "Esimated Effect on Recidivism")
```

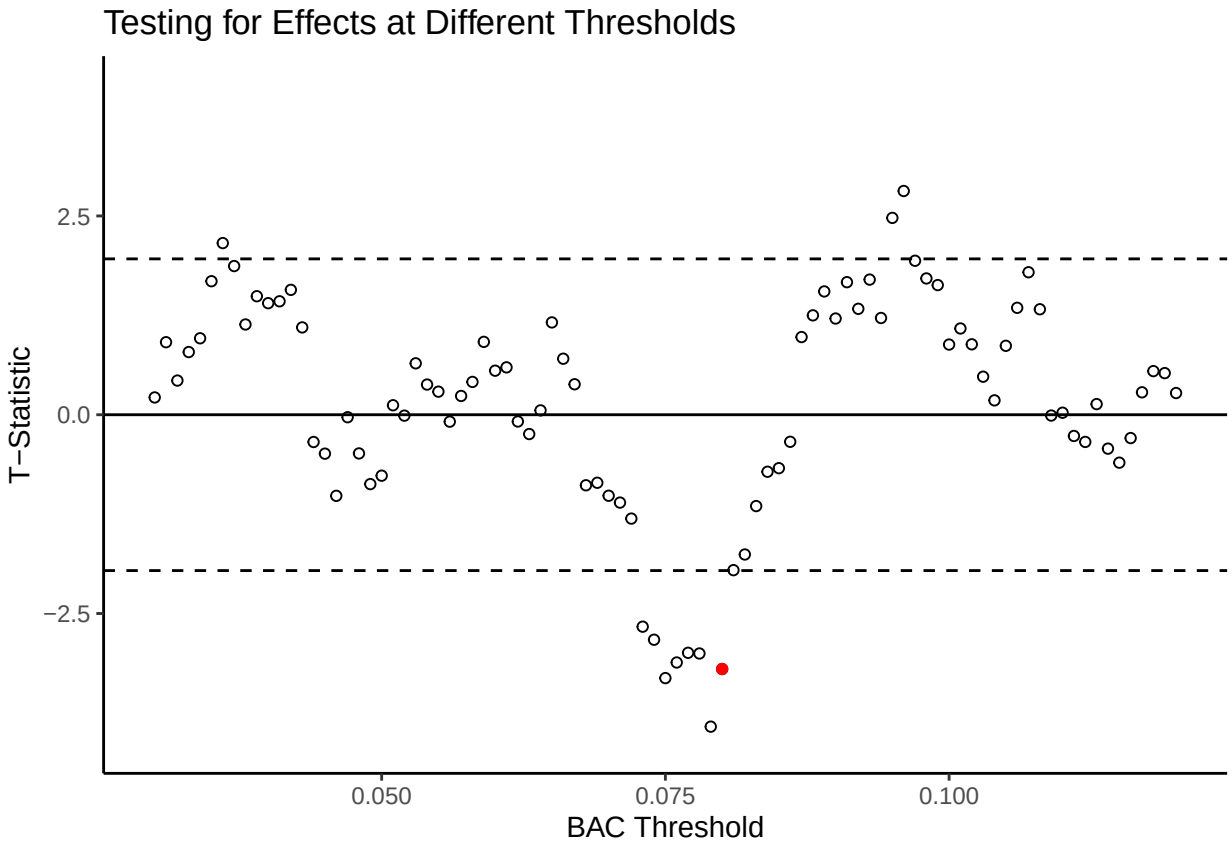


Part G

Finally, I want you to reestimate your models using instead of the .08 threshold, every other BAC as the threshold, keeping a bandwidth of .03, between .03 and .12 (i.e. .03, .031, .032, etc). Create a scatter plot of both the point estimates (yaxis) against the potential RD thresholds (x axis). Now create a scatter plot of your test statistic on the null hypothesis of your point estimates (z-stat/t-stat) with the test statistic on the y axis, and the threshold on the x axis. What is the rank of (1 being the largest) of your point estimate estimated at .08? How many estimates did you do? The rank divided by the number of tests is called an empirical p-value based on a form of permutation inference.

```
# reordering argument order so that we can still use map function
rd_bw2 = function(threshold, bw){
  return(rd_bw(bw, threshold))
}

# Testing at all thresholds between 0.03 and 0.12
```

```
output_g = map_dfr(.x = seq(0.03, 0.12, 0.001), .f = rd_bw2, bw = 0.03)
colnames(output_g) = c("threshold", "bw", "estimate", "lower_bound", "upper_bound", "t_stat")
output_g = data.table(output_g)

# Plot of T Stats
ggplot(data = output_g, aes(x = threshold, y = t_stat)) +
  geom_point(shape = 1) +
  geom_point(data = output_g[threshold==0.08], aes(x=threshold,y=t_stat), color = "red")+
  geom_hline(yintercept = 0) +
  geom_hline(yintercept = -1.96, linetype = "dashed") +
  geom_hline(yintercept = 1.96, linetype = "dashed") +
  ylim(-4.1,4.1)+
  theme_classic() +
  labs(title = "Testing for Effects at Different Thresholds",
       x = "BAC Threshold",
       y = "T-Statistic")
```



Solution:

```
# Number of Permutations
den = nrow(output_g)

# Rank of 0.08 Threshold
output_g$rank = rank(output_g$t_stat)
num = output_g[threshold == 0.08, "rank"]

# Permutation Test
p_value = num/den
```

The rank of the point estimate for a threshold of 0.08 is 3, where we have calculated 91 estimates. This gives a p-value for the permutation test of 0.032967032967033.