Regression Discontinuity Designs and Difference-in-Differences

Jake Bowers, Ben Hansen & Tom Leavitt

August 12, 2018

Contents

1 Regression Discontinuity Design

1.1 General Setup

Today we are going to use the data on close US House of Representatives races 1942–2008 used in Caughey and Sekhon (2011).[1] Caughey and Sekhon (2011) engage in a debate whose participants seek to identify the causal effect of the so-called "incumbency advantage." That is, what effect does a candidate's status as an incumbent have on whether or not that candidate wins an election? Obviously, whether or not a candidate is an incumbent is *not* randomly assigned.

Let's first load the data:

[1] The full replication data is available for download here. But we read it directly below.

```
rm(list = ls())

rdd_data <- read_dta("http://jakebowers.org/Matching/RDReplication.dta") %>% filter(Use == 1)  ## Use is indicator for
```

The Running Variable

The "running variable" is called DifDPct, which is defined as the Democratic margin of victory or defeat in the election; in other words, DifDPct is the difference between the percentage of all votes that were cast for the leading Democrat in the race and the percentage cast for the leading non-Democrat. Races in which no Democrat ran or in which the top two vote-getters were both Democrats are coded as missing.

```
running_var <- matrix(c("DifDPct", "Democrat Margin of Victory"), ncol = 2, byrow = TRUE)

dimnames(running_var) <- list(1, c("Running Variable", "Description"))

kable(running_var)
```

Running Variable	Description
DifDPct	Democrat Margin of Victory

The Treatment Variable

The treatment variable is whether or not the Democratic candidate wins the election or not. If the candidate wins the election, then that candidate is assigned to "treatment." If the candidate loses the election, then he or she is assigned to "control."

```
treatment <- matrix(c("DemWin", "Democrat Wins Election"), ncol = 2, byrow = TRUE)

dimnames(treatment) <- list(1, c("Treatment", "Description"))

kable(treatment)
```

Treatment	Description
DemWin	Democrat Wins Election

Now let's quickly look at the empirical distribution of the treatment variable:

```
table(rdd_data$DemWin)
```

Outcome Variables

In Caughey and Sekhon (2011), the primary outcome variables of interest are as follows: whether a democrat wins the next election, the proportion voting for a democrat in the next election, and the

	Democrat Wins Election
0	4507
1	5677

Table 1: The Treatment Variable

democratic vote margin in the next election.

```
dvs <- matrix(c("DWinNxt", "Dem Win t + 1", "DPctNxt", "Dem % t + 1", "DifDPNxt", "Dem % Margin t + 1"),
    ncol = 2, byrow = TRUE)

dimnames(dvs) <- list(seq(from = 1, to = 3, by = 1), c("Outcome", "Description"))

kable(dvs)
```

Outcome	Description
DWinNxt	Dem Win t + 1
DPctNxt	Dem % t + 1
DifDPNxt	Dem % Margin t + 1

Baseline Covariates

The relevant baseline (i.e., pre-treatment) covariates about the races are:

Covariate	Description
DWinPrv	Dem Win t - 1
DPctPrv	Dem % t - 1
DifDPPrv	Dem % Margin t - 1
IncDWNOM1	Inc's D1 NOMINATE
DemInc	Dem Inc in Race
NonDInc	Rep Inc in Race
PrvTrmsD	Dem's # Prev Terms
PrvTrmsO	Rep's # Prev Terms
RExpAdv	Rep Experience Adv
DExpAdv	Dem Experience Adv
ElcSwing	Partisan Swing
CQRating3	CQ Rating {-1, 0, 1}
DSpndPct	Dem Spending %
DDonaPct	Dem Donation %
SoSDem	Dem Sec of State
GovDem	Dem Governor
DifPVDec	Dem Pres % Margin
DemOpen	Dem-held Open Seat
NonDOpen	Rep-held Open Seat
OpenSeat	Open Seat
VtTotPct	Voter Turnout %
GovWkPct	Pct Gov't Worker
UrbanPct	Pct Urban
BlackPct	Pct Black
ForgnPct	Pct Foreign Born

1.2 RDD Framework

Let the index $i \in \{1, \ldots, n\}$ run over the n experimental units. In the context of a regression discontinuity design, let R_i denote the random "score variable" (also known as "running variable"). The random assignment variable, $Z_i \in \{0, 1\}$, which indicates whether subject i is assigned to treatment, $Z_i = 1$, or to control, $Z_i = 0$, is a deterministic function of R_i. In the context of the incumbency advantage study, we can define Z_i as follows:

$$Z_i \equiv \mathbf{I}\left[R_i > 0\right],$$

where $\mathbf{I}[\cdot]$ is an indicator function that is 1 if the argument to the function is true and 0 if false. In this particular study, if the margin of victory is greater than 0, then candidate i is treated ($Z_i = 1$), and if not, then candidate i is assigned to control ($Z_i = 0$).

Let $W_0 = [\underline{r}, \overline{r}]$, where $\underline{r} < r_0 < \overline{r}$, denote the window around the cutpoint (or threshold value), r_0, that sorts units into treatment or control.

1.3 Optimal Bandwidth Selection

We know from Berger et al. (1988); Hansen and Sales (2015); Rosenbaum (2008) that "if a researcher pre-specifies a sequence of hypotheses and corresponding level-α tests, tests those hypotheses in order, and stops testing after the first non-rejected hypothesis, then the probability of incorrectly rejecting at least one correct hypothesis is at most α" (Hansen and Sales, 2015, p. 185).

As applied to bandwidth selection in the RDD context, the SIUP implies that one should start with a set of candidate bandwidths and sequentially test for covariate balance (beginning from either the largest candidate bandwidth or the smallest candidate bandwidth).

Let's specify a set of candidate bandwidths and then sequentially test covariate balance. Before actually testing, though, we want to specify a balance criterion and then maximize effective sample size subject to that criterion.

```
bal_fmla <- reformulate(covs[1:25], response = "DemWin")

candidate_bands <- seq(from = -5, to = 5, by = 0.1)
```

Now let's first filter our dataset and check for balance in the largest candidate bandwidth spanning from -5 to 5.

```
lower_bound <- seq(from = -5, to = -0.1, by = 0.1)

upper_bound <- seq(from = 0.1, to = 5, by = 0.1) %>% sort(decreasing = TRUE)

rdd_dataA <- rdd_data
rdd_dataA %<>% filter(DifDPct > lower_bound[1] & DifDPct < upper_bound[1])

rdd_dataA %$% summary(DifDPct)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -4.99   -2.50   -0.13    0.04    2.81    4.99
```

```
xBalance(fmla = bal_fmla, data = rdd_dataA, report = "chisquare.test")
```

```
---Overall Test---
        chisquare df p.value
unstrat       569 45 8.9e-92
---
Signif. codes:  0 '***' 0.001 '** ' 0.01 '*  ' 0.05 '.  ' 0.1 '   ' 1
```

```
xBalance(fmla = bal_fmla, data = rdd_dataA, report = "all")
```

```
                strata  unstrat
                stat   DemWin=0 DemWin=1 adj.diff adj.diff.null.sd std.diff     z
vars
DWinPrv                3.5e-01  5.2e-01  1.8e-01          3.4e-02  3.6e-01  5.2e+00  ***
DPctPrv                4.7e+01  5.1e+01  3.6e+00          9.1e-01  2.8e-01  4.0e+00  ***
DifDPPrv               -5.4e+00 2.2e+00  7.6e+00          1.8e+00  3.0e-01  4.3e+00  ***
IncDWNOM1              9.3e-02  -6.2e-03 -1.0e-01         2.3e-02  -3.0e-01 -4.3e+00 ***
DemInc                 2.5e-01  4.1e-01  1.5e-01          3.2e-02  3.3e-01  4.7e+00  ***
NonDInc                4.6e-01  3.2e-01  -1.4e-01         3.3e-02  -2.9e-01 -4.3e+00 ***
PrvTrmsD               3.7e-01  1.8e+00  1.4e+00          1.7e-01  6.0e-01  8.4e+00  ***
PrvTrmsO               2.0e+00  1.1e+00  -9.3e-01         1.5e-01  -4.2e-01 -6.0e+00 ***
RExpAdv                4.9e-01  3.0e-01  -2.0e-01         3.3e-02  -4.1e-01 -5.9e+00 ***
DExpAdv                2.7e-01  4.4e-01  1.7e-01          3.3e-02  3.7e-01  5.3e+00  ***
ElcSwing               1.6e+00  1.1e+00  -4.8e-01         5.5e-01  -6.0e-02 -8.8e-01
CQRating3              -1.2e-01 1.1e-01  2.3e-01          4.3e-02  3.7e-01  5.3e+00  ***
DSpndPct               4.7e+01  5.1e+01  3.8e+00          6.6e-01  4.0e-01  5.7e+00  ***
DDonaPct               4.8e+01  5.1e+01  3.8e+00          7.1e-01  3.7e-01  5.4e+00  ***
SoSDem                 4.5e-01  4.7e-01  2.2e-02          3.4e-02  4.3e-02  6.3e-01
GovDem                 4.3e-01  4.9e-01  5.4e-02          3.4e-02  1.1e-01  1.6e+00
DifPVDec               -6.4e-02 -5.8e-02 5.8e-03         1.1e-02  3.7e-02  5.3e-01
DemOpen                8.0e-02  1.1e-01  3.0e-02          2.0e-02  1.0e-01  1.5e+00
NonDOpen               1.1e-01  9.8e-02  -1.2e-02         2.1e-02  -3.9e-02 -5.7e-01
OpenSeat               1.9e-01  2.1e-01  1.3e-02          2.7e-02  3.3e-02  4.8e-01
VtTotPct               3.9e+01  3.9e+01  -6.0e-01         5.5e-01  -7.4e-02 -1.1e+00
GovWkPct               4.5e+00  4.6e+00  4.3e-02          1.7e-01  1.8e-02  2.6e-01
UrbanPct               6.3e+01  6.6e+01  3.0e+00          1.6e+00  1.3e-01  1.9e+00  .
BlackPct               5.3e+00  6.0e+00  6.2e-01          5.1e-01  8.4e-02  1.2e+00
ForgnPct               3.7e+00  3.8e+00  6.4e-02          3.0e-01  1.5e-02  2.1e-01
DWinPrv.NATRUE         3.4e-02  1.7e-02  -1.8e-02         1.1e-02  -1.1e-01 -1.6e+00
DPctPrv.NATRUE         6.9e-02  5.3e-02  -1.6e-02         1.6e-02  -6.8e-02 -9.9e-01
DifDPPrv.NATRUE        3.4e-02  2.1e-02  -1.3e-02         1.1e-02  -7.8e-02 -1.1e+00
IncDWNOM1.NATRUE       6.9e-03  1.7e-18  -6.9e-03         4.0e-03  -1.2e-01 -1.7e+00 .
DemInc.NATRUE          4.1e-02  2.6e-02  -1.5e-02         1.2e-02  -8.3e-02 -1.2e+00
NonDInc.NATRUE         7.8e-02  6.0e-02  -1.8e-02         1.7e-02  -7.2e-02 -1.0e+00
PrvTrmsD.NATRUE        6.2e-01  3.3e-15  -6.2e-01         3.2e-02  -1.8e+00 -1.9e+01 ***
PrvTrmsO.NATRUE        0.0e+00  6.6e-01  6.6e-01          3.2e-02  2.0e+00  2.1e+01  ***
RExpAdv.NATRUE         7.6e-02  6.7e-02  -8.7e-03         1.8e-02  -3.4e-02 -4.9e-01
DExpAdv.NATRUE         8.2e-02  6.9e-02  -1.3e-02         1.8e-02  -5.0e-02 -7.3e-01
ElcSwing.NATRUE        6.9e-03  1.7e-18  -6.9e-03         4.0e-03  -1.2e-01 -1.7e+00 .
CQRating3.NATRUE       4.1e-01  3.9e-01  -2.0e-02         3.4e-02  -4.2e-02 -6.1e-01
DSpndPct.NATRUE        5.6e-01  5.3e-01  -2.4e-02         3.4e-02  -4.8e-02 -7.0e-01
DDonaPct.NATRUE        6.4e-01  6.8e-01  4.7e-02          3.2e-02  9.8e-02  1.4e+00
SoSDem.NATRUE          6.9e-03  9.5e-03  2.7e-03          6.2e-03  3.0e-02  4.4e-01
DifPVDec.NATRUE        6.4e-02  5.7e-02  -6.8e-03         1.6e-02  -2.8e-02 -4.2e-01
DemOpen.NATRUE         2.3e-02  7.2e-03  -1.6e-02         8.4e-03  -1.3e-01 -1.9e+00 .
NonDOpen.NATRUE        5.9e-02  4.1e-02  -1.9e-02         1.5e-02  -8.7e-02 -1.3e+00
OpenSeat.NATRUE        6.4e-02  4.5e-02  -1.9e-02         1.6e-02  -8.2e-02 -1.2e+00
VtTotPct.NATRUE        1.7e-01  1.6e-01  -1.2e-02         2.5e-02  -3.1e-02 -4.6e-01
GovWkPct.NATRUE        9.4e-02  9.1e-02  -3.1e-03         2.0e-02  -1.1e-02 -1.6e-01
UrbanPct.NATRUE        9.6e-02  9.1e-02  -5.4e-03         2.0e-02  -1.9e-02 -2.7e-01
```

```
BlackPct.NATRUE         9.4e-02  9.1e-02  -3.1e-03         2.0e-02  -1.1e-02 -1.6e-01
ForgnPct.NATRUE         9.4e-02  9.1e-02  -3.1e-03         2.0e-02  -1.1e-02 -1.6e-01
---Overall Test---
        chisquare df p.value
unstrat       569 45 8.9e-92
---
Signif. codes:  0 '***' 0.001 '** ' 0.01 '*  ' 0.05 '.  ' 0.1 '   ' 1
```

Question for Students:

- What can we infer from the results of the Chi-Squared balance test above?

Now let's write a function to perform this same procedure over all candidate bandwidth sizes beginning with the largerst candidate bandwidth and subsequently testing smaller and smaller bandwidths in order.

```
chi_squared_balance <- function(i, running_var, bal_fmla, data) {

    # Preliminaries
    suppressMessages(stopifnot(require(dplyr, quietly = TRUE)))
    suppressMessages(stopifnot(require(parallel, quietly = TRUE)))
    suppressMessages(stopifnot(require(magrittr, quietly = TRUE)))
    suppressMessages(stopifnot(require(RItools, quietly = TRUE)))

    lower_bound <- seq(from = -5, to = -0.1, by = 0.1)

    upper_bound <- seq(from = 0.1, to = 5, by = 0.1) %>% sort(decreasing = TRUE)

    data %<>% filter(running_var > lower_bound[i] & running_var < upper_bound[i])

    # Effective Sample Size
    ess <- nrow(data)

    p_value <- xBalance(fmla = bal_fmla, data = data, report = "chisquare.test")$overall[[3]]

    bands <- cbind(ess, p_value)

    return(bands)

}

is <- seq(from = 1, to = length(seq(from = -5, to = -0.1, by = 0.1)), by = 1)

cl <- makeCluster(parallel::detectCores())

band_df <- data.frame(t(parSapply(cl, is, chi_squared_balance, running_var = rdd_data$DifDPct, bal_fmla = bal_fmla,
    data = rdd_data))) %>% rename(ess = X1, p_value = X2)

parallel::stopCluster(cl)
```

```
kable(band_df)
```

ess	p_value
856	0.000
843	0.000
826	0.000
811	0.000
797	0.000
782	0.000
767	0.000
751	0.000
725	0.000
701	0.000
675	0.000
652	0.000
626	0.000
613	0.000
600	0.000
586	0.000
565	0.000
547	0.000
525	0.000
515	0.000
499	0.000
474	0.000
456	0.000
429	0.000
418	0.000
404	0.000
387	0.000
375	0.000
357	0.000
339	0.000
322	0.001
302	0.005
287	0.002
274	0.004
257	0.006
238	0.031
219	0.082
204	0.088
190	0.107
176	0.101
168	0.132
146	0.141
136	0.178
122	0.248
107	0.357
85	0.369
70	0.578
55	0.507
37	0.606
19	0.456

Questions for Students:

- There's something interesting going on as we make the window around the cut point smaller and smaller. What is it?
- And what are its implications, which are also mentioned by Caughey and Sekhon (2011)?

1.4 Local Random Assignment Does *Not* Imply Exclusion Restriction

Now let's assume that within a certain bandwidth around the cutpoint, W_0, the assumption of a local randomized experiment—described in (**??**) above—obtains. Yet even if this assumption obtains, the running variable, R_i, might still relate to potential outcomes through a mechanism other than $Z_i W_0$.

Question for Students:

- Explain why this would be a violation of the *exclusion restriction*?

The exclusion restriction is a strong assumption in the RDD context since Z_i is a deterministic function of the running variable, R_i, which implies that treatment and control groups, by construction, will be imbalanced on the running variable. Thus, if R_i relates to (y_{ic}, y_{it}), through a mechanism other than Z_{iW_0}, then the exlcusion restriction is violated.

If we think the exclusion restriction holds, then we can simply perform outcome analysis within a window around the cutpoint where local randomization presumably holds:

```
rdd_data %<>% filter(DifDPct > lower_bound[47] & DifDPct < upper_bound[47])

rdd_data %>% nrow


[1] 70


coef(lm(formula = DPctNxt ~ DemWin, data = rdd_data))[["DemWin"]]


[1] 11.9
```

What can we do if the exclusion restriction is violated? One approach is to model potential outcomes as a function of the running variable, and then to "de-trend" (or "transform") the outcome variable and to subsequently make the claim of "residual ignorability."

More specifically, we postulate that:

$$Y_{c_i} = \alpha_0 + \beta R_i + \epsilon_{c_i},\ \epsilon_{c_i} \sim \mathcal{N}\left(0, \sigma^2\right)$$
$$Y_{t_i} = \alpha_0 + (\alpha_1 - \alpha_0)\, Z_i \beta R_i + \epsilon_{c_i},\ \epsilon_{c_i} \sim \mathcal{N}\left(0, \sigma^2\right)$$

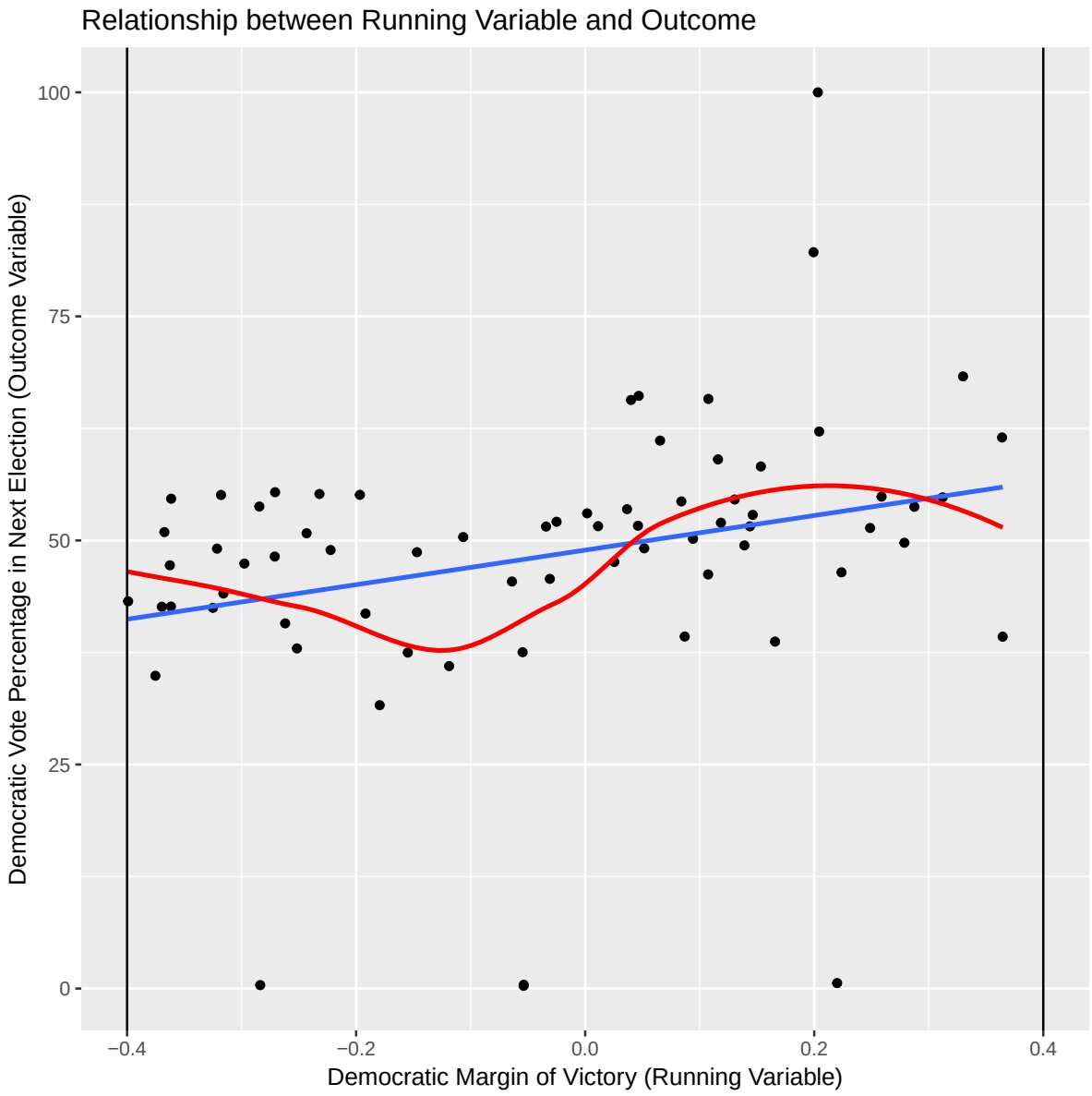
```
tmp_lm <- lm(DPctNxt ~ DifDPct, data = rdd_data)

rdd_data %<>% mutate(resid_DPctNxt = resid(tmp_lm))
```

Question for Students:

- Do we think modeling the outcome variable as a linear function of the running variable is appropriate?

```
ggplot(rdd_data, aes(DifDPct, DPctNxt)) + geom_point() + geom_smooth(method = lm, se = FALSE) + geom_smooth(se = FALSE
    colour = "red") + xlab("Democratic Margin of Victory (Running Variable)") + ylab("Democratic Vote Percentage in Ne
    ggtitle("Relationship between Running Variable and Outcome") + geom_vline(xintercept = c(-0.4, 0.4))
```



1.5 Outcome Analysis

We can now perform outcome analysis on the detrended outcome variable:

```
coeftest(x = lm(formula = resid_DPctNxt ~ DemWin, data = rdd_data), vcov. = vcovHC(x = lm(formula = resid_DPctNxt ~
    DemWin, data = rdd_data), type = "HC2"))[2, 1]


[1] 4.57


source("http://jakebowers.org/ICPSR/confintHC.R")
confint.HC(lm(formula = resid_DPctNxt ~ DemWin, data = rdd_data), parm = "DemWin", thevcov = vcovHC(lm(formula = resid
    DemWin, data = rdd_data), type = "HC2"))
```

```
       2.5 % 97.5 %
DemWin -2.64   11.8
```

?, however, propose robust regression, which is less sensitive to violations of the regression model's assumptions.

```
tmp_rlm <- rlm(DPctNxt ~ DifDPct, data = rdd_data)

rdd_data %<>% mutate(rlm_resid_DPctNxt = resid(tmp_rlm))

coeftest(x = lm(formula = rlm_resid_DPctNxt ~ DemWin, data = rdd_data), vcov. = vcovHC(x = lm(formula = rlm_resid_DPct
    DemWin, data = rdd_data), type = "HC2"))[2, 1]


[1] 5.87


confint.HC(lm(formula = rlm_resid_DPctNxt ~ DemWin, data = rdd_data), parm = "DemWin", thevcov = vcovHC(lm(formula = r
    DemWin, data = rdd_data), type = "HC2"))


       2.5 % 97.5 %
DemWin  -1.3     13
```

References

Berger, R. L., D. D. Boos, and F. M. Guess (1988). Tests and confidence sets for comparing two mean residual life functions. *Biometrics*, 103–115. 5

Caughey, D. and J. S. Sekhon (2011). Elections and the regression discontinuity design: Lessons from close us house races, 1942–2008. *Political Analysis 19*(4), 385–408. 1, 2, 10

Hansen, B. B. and A. Sales (2015). Comment on cochran's "observational studies". *Observational Studies*, 184–193. 5

Rosenbaum, P. R. (2008). Testing hypotheses in order. *Biometrika*. 5