

The Toy Example - ICHPS 2018

Thomas E. Love, Ph.D.

Version: 2018-01-08

Contents

0.1	Setup: Loading Packages	3
1	The Toy Data at Load-In	3
1.1	The Data Set	3
1.2	The Codebook	4
1.3	Numerical Summaries	4
1.4	Table 1	5
2	Data Management and Cleanup	5
2.1	Range Checks for Quantitative (continuous) Variables	5
2.2	Restating Categorical Information in Helpful Ways	6
2.3	Dealing with Variables including More than Two Categories	7
2.4	Creating the Transformation and Product Terms	9
3	Data Set After Cleaning	9
3.1	Glimpse	9
3.2	Summary	9
3.3	Table 1	10
4	The 13 Tasks We'll Tackle in this Example	11
5	Task 1. Ignoring covariates, estimate the effect of treatment vs. control on...	12
5.1	Outcome 1 (a continuous outcome)	12
5.2	Outcome 2 (a binary outcome)	13
5.3	Outcome 3 (a time-to-event outcome with right censoring)	15
5.4	Unadjusted Estimates of Treatment Effect on Outcomes	17
6	Task 2. Fit the propensity score model, then plot the PS-treatment relationship	17
6.1	Comparing the Distribution of Propensity Score Across the Two Treatment Groups	18
7	Task 3. Rubin's Rules to Check Overlap Before Propensity Adjustment	20
7.1	Rubin's Rule 1	21
7.2	Rubin's Rule 2	21
7.3	Rubin's Rule 3	21
8	Task 4. Use 1:1 greedy matching on the linear PS, then check post-match balance	23
8.1	Balance Assessment (Semi-Automated)	24
8.2	Extracting, Tabulating and Plotting Standardized Differences (without <code>cobalt</code>)	28
8.3	Using <code>cobalt</code> to build a "Love Plot" after Matching	31
8.4	Extracting, Tabulating and Plotting Variance Ratios (without <code>cobalt</code>)	33
8.5	Creating a New Data Frame, Containing the Matched Sample (without <code>cobalt</code>)	35
8.6	Rubin's Rules to Check Balance After Matching	35
9	Task 5. After matching, estimate the causal effect of treatment on ...	38
9.1	Outcome 1 (a continuous outcome)	38
9.2	Outcome 2 (a binary outcome)	41

9.3	Outcome 3 (a time-to-event outcome)	42
9.4	Results So Far (After Propensity Matching)	44
10	Task 6. Subclassify by PS quintile, then display post-subclassification balance	45
10.1	Check Balance and Propensity Score Overlap in Each Quintile	45
10.2	Creating a Standardized Difference Calculation Function	47
10.3	Creating the Five Subsamples, by PS Quintile	47
10.4	Standardized Differences in Each Quintile, and Overall	48
10.5	Plotting the Standardized Differences	48
10.6	Checking Rubin's Rules Post-Subclassification	50
11	Task 7. After subclassifying, what is the estimated average treatment effect?	53
11.1	... on Outcome 1 [a continuous outcome]	53
11.2	... on Outcome 2 [a binary outcome]	54
11.3	... on Outcome 3 [a time to event]	55
11.4	Results So Far (After Matching and Subclassification)	56
12	Task 8. Execute weighting by the inverse PS, then assess covariate balance	57
12.1	ATT approach: Weight treated subjects as 1; control subjects as $ps/(1-ps)$	57
12.2	ATE Approach: Weight treated subjects by $1/ps$; Control subjects by $1/(1-PS)$	58
12.3	Assessing Balance after Weighting	58
12.4	Rubin's Rules after ATT weighting	63
12.5	Rubin's Rules after ATE weighting	64
13	Using TWANG for Alternative PS Estimation and ATT Weighting	64
13.1	Estimate the Propensity Score using Generalized Boosted Regression, and then perform ATT Weighting	64
13.2	Semi-Automated Love plot of Standardized Differences	68
13.3	Semi-Automated Love plot of Variance Ratios	68
14	Task 9. After weighting, what is the estimated average average causal effect of treatment?	69
14.1	... on Outcome 1 [a continuous outcome]	69
14.2	... on Outcome 2 [a binary outcome]	71
14.3	... on Outcome 3 [a time to event]	73
14.4	Results So Far (After Matching, Subclassification and Weighting)	76
15	Task 10. After direct adjustment for the linear PS, what is the estimated average causal treatment effect?	76
15.1	... on Outcome 1 [a continuous outcome]	76
15.2	... on Outcome 2 [a binary outcome]	77
15.3	... on Outcome 3 [a time-to-event outcome]	78
15.4	Results So Far (After Matching, Subclassification, Weighting, Adjustment)	80
16	Task 11. "Double Robust" Approach - Weighting + Adjustment, what is the estimated average causal effect of treatment?	81
16.1	... on Outcome 1 [a continuous outcome]	81
16.2	... on Outcome 2 [a binary outcome]	83
16.3	... on Outcome 3 [a time to event]	85
17	Task 12. Results	88
17.1	Treatment Effect Estimates	88
17.2	Quality of Balance: Standardized Differences and Variance Ratios	89
17.3	Quality of Balance: Rubin's Rules	89

18 What is a Sensitivity Analysis for Matched Samples?	90
18.1 Goal of a Formal Sensitivity Analysis for Matched Samples	90
18.2 The Sensitivity Parameter, Γ	91
18.3 Interpreting the Sensitivity Parameter, Γ	91
19 Task 13. Sensitivity Analysis for Matched Samples, Outcome 1, using rbounds	91
19.1 Rosenbaum Bounds for the Wilcoxon Signed Rank test (Quantitative outcome)	92
19.2 Specifying The Threshold Γ value	93
19.3 Interpreting Γ appropriately	93
19.4 An Alternate Approach - the Hodges-Lehman estimate	93
19.5 What about other types of outcomes?	94
19.6 What about when we match 1:2 or 1:3 instead of 1:1?	94
20 Wrapup	94
20.1 Session status	95

0.1 Setup: Loading Packages

```
library(pander); library(tableone); library(broom)
library(Epi); library(survival); library(arm)
library(Hmisc); library(Matching); library(lme4)
library(twang); library(survey); library(rbounds)
library(devtools); library(cobalt); library(tidyverse)
```

1 The Toy Data at Load-In

This document is about showing a (relatively) simple way to do things. In no way would I claim that the approaches provided here are optimal. This is just a first demonstration.

1.1 The Data Set

The Data Set is 100% fictional, and is available as `toy.csv` on the course website. It contains data on 200 subjects (70 treated - subjects 131-200 and 130 controls - subjects 1-130) on treatment status, six covariates, and three outcomes, with no missing observations anywhere. We assume that a logical argument suggests that the square of `covA`, as well as the interactions of `covB` with `covC` and with `covD` should be related to treatment assignment, and thus should be included in our propensity model.

Our objective is to estimate the average causal effect of treatment (as compared to control) on each of the three outcomes, without propensity adjustment, and then with propensity matching, subclassification, weighting and regression adjustment using the propensity score.

```
toy <- read_csv("toy.csv")
```

Parsed with column specification:

```
cols(
  subject = col_integer(),
  treated = col_integer(),
  covA = col_double(),
  covB = col_integer(),
  covC = col_double(),
  covD = col_double(),
```

```

covE = col_integer(),
covF = col_character(),
out1.cost = col_integer(),
out2.event = col_character(),
out3.time = col_integer()
)

toy$out2.event <- factor(toy$out2.event)
toy$covF <- factor(toy$covF)
toy

```

```

# A tibble: 200 x 11
  subject treated covA covB covC covD covE covF out1~ out2~ out3~
    <int>   <int> <dbl> <int> <dbl> <dbl> <int> <fctr> <int> <fct> <int>
1     1     1     0 0.120     1  7.60  6.70     8 3-High    46 Yes     99
2     2     2     0 0.0500    0 11.2  11.7     7 1-Low     44 Yes    112
3     3     3     0 0.680     0  7.70  9.30    12 2-Mid~    47 No     89
4     4     4     0 0.460     0  9.50  9.40    12 2-Mid~    42 No    121
5     5     5     0 0.340     1  9.90 10.7     9 2-Mid~    30 No    120
6     6     6     0 0.150     0 10.4 12.2     6 3-High    28 No     91
7     7     7     0 0.180     0 13.7  8.50     9 1-Low    60 Yes    101
8     8     8     0 0.280     1 11.0  9.20     6 2-Mid~    62 No    101
9     9     9     0 0.160     0  9.80 10.2    10 2-Mid~    48 No    103
10    10    10     0 0.120     1 10.8 12.1    11 1-Low    54 Yes     89
# ... with 190 more rows

```

1.2 The Codebook

Variable	Type	Notes
subject	Subject ID	1-130 = controls, 131-200 = treated
treated	2-level categorical (0/1)	0 = control, 1 = treated
covA	Quantitative (2 decimal places)	reasonable values range from 0 to 1
covB	2-level categorical (0/1)	0 = no, 1 = yes
covC	Quantitative (1 decimal place)	range 3-20
covD	Quantitative (1 decimal place)	range 3-20
covE	Integer	range 3-20
covF	3-level ordinal factor	1 = Low, 2 = Middle, 3 = High
out1.cost	Quantitative outcome	typical values 20-80
out2.event	Binary outcome (did event occur?)	Yes/No (note: event is bad)
out3.time	Time to event outcome	Time before event is observed or subject exits study (censored), range is 75-156 weeks

With regard to the `out3.time` variable, subjects with `out2.event = No` were censored, so that `out2.event = Yes` indicates an observed event.

1.3 Numerical Summaries

```
summary(toy)
```

```

subject      treated      covA      covB

```

Min. : 1.00	Min. :0.00	Min. :0.0400	Min. :0.00
1st Qu.: 50.75	1st Qu.:0.00	1st Qu.:0.2675	1st Qu.:0.00
Median :100.50	Median :0.00	Median :0.5100	Median :0.00
Mean :100.50	Mean :0.35	Mean :0.4995	Mean :0.37
3rd Qu.:150.25	3rd Qu.:1.00	3rd Qu.:0.7300	3rd Qu.:1.00
Max. :200.00	Max. :1.00	Max. :0.9900	Max. :1.00
covC	covD	covE	covF
Min. : 4.200	Min. : 5.500	Min. : 2.00	1-Low :67
1st Qu.: 8.600	1st Qu.: 8.500	1st Qu.: 8.00	2-Middle:84
Median : 9.850	Median : 9.900	Median :10.00	3-High :49
Mean : 9.714	Mean : 9.863	Mean :10.21	
3rd Qu.:10.725	3rd Qu.:11.100	3rd Qu.:12.00	
Max. :16.200	Max. :14.400	Max. :19.00	
out1.cost	out2.event	out3.time	
Min. :24.00	No :103	Min. : 79.0	
1st Qu.:39.00	Yes: 97	1st Qu.:101.0	
Median :49.00		Median :109.0	
Mean :50.42		Mean :109.2	
3rd Qu.:58.00		3rd Qu.:118.0	
Max. :80.00		Max. :151.0	

1.4 Table 1

```
varlist = c("covA", "covB", "covC", "covD", "covE", "covF",
            "out1.cost", "out2.event", "out3.time")
factorlist = c("covB", "covF", "out2.event")
CreateTableOne(vars = varlist, strata = "treated",
               data = toy, factorVars = factorlist)
```

	Stratified by treated			
	0	1	p	test
n	130	70		
covA (mean (sd))	0.45 (0.29)	0.59 (0.22)	<0.001	
covB = 1 (%)	40 (30.8)	34 (48.6)	0.020	
covC (mean (sd))	9.76 (2.02)	9.63 (2.02)	0.649	
covD (mean (sd))	9.74 (1.77)	10.10 (1.88)	0.183	
covE (mean (sd))	10.50 (3.54)	9.67 (2.74)	0.090	
covF (%)			0.266	
1-Low	48 (36.9)	19 (27.1)		
2-Middle	54 (41.5)	30 (42.9)		
3-High	28 (21.5)	21 (30.0)		
out1.cost (mean (sd))	44.94 (9.69)	60.61 (16.56)	<0.001	
out2.event = Yes (%)	58 (44.6)	39 (55.7)	0.177	
out3.time (mean (sd))	106.95 (11.94)	113.44 (10.75)	<0.001	

2 Data Management and Cleanup

2.1 Range Checks for Quantitative (continuous) Variables

Checking and cleaning the quantitative variables is pretty straightforward - the main thing I'll do at this stage is check the ranges of values shown to ensure that they match up with what I'm expecting. Here, all of

the quantitative variables have values that fall within the “permissible” range described by my codebook, so we’ll assume that for the moment, we’re OK on `subject` (just a meaningless code, really), `covA`, `covC`, `covD`, `covE`, `out1.cost` and `out3.time`, and we see no missingness.

2.2 Restating Categorical Information in Helpful Ways

The cleanup of the toy data focuses, as it usually does, on variables that contain **categories** of information, rather than simple counts or measures, represented in quantitative variables.

2.2.1 Re-expressing Binary Variables as Numbers and Factors

We have three binary variables (`treated`, `covB` and `out2.event`). A major issue in developing these variables is to ensure that the direction of resulting odds ratios and risk differences are consistent and that cross-tabulations are in standard epidemiological format.

It will be useful to define binary variables in two ways:

- as a numeric indicator variable taking on the values 0 (meaning “not having the characteristic being studied”) or 1 (meaning “having the characteristic being studied”)
- as a text factor - with the levels of our key exposure and outcomes arranged so that “having the characteristic” precedes “not having the characteristic” in R when you create a table, but the covariates should still be No/Yes.

So what do we currently have? From the output below, it looks like `treated` and `covB` are numeric, 0/1 variables, while `out2.event` is a factor with levels “No” and then “Yes”

```
toy %>% select(treated, covB, out2.event) %>% summary()
```

	treated		covB		out2.event
Min.	:0.00	Min.	:0.00	No	:103
1st Qu.	:0.00	1st Qu.	:0.00	Yes	: 97
Median	:0.00	Median	:0.00		
Mean	:0.35	Mean	:0.37		
3rd Qu.	:1.00	3rd Qu.	:1.00		
Max.	:1.00	Max.	:1.00		

So, we’ll create factors for `treated` and `covB`:

```
toy$treated_f <- factor(toy$treated, levels = c(1,0), labels = c("Treated", "Control"))
toy$covB_f <- factor(toy$covB, levels = c(0,1), labels = c("No B", "Has B"))
```

For `out2.event`, on the other hand, we don’t have either quite the way we might want it. As you see in the summary output, we have two codes for `out2.event` - either No or Yes, in that order. But we want Yes to precede No (and I’d like a more meaningful name). So I redefine the factor variable, as follows.

```
toy$out2_f <- factor(toy$out2.event, levels = c("Yes","No"), labels = c("Event","No Event"))
```

To obtain a numerical (0 or 1) version of `out2.event` we can use R’s `as.numeric` function - the problem is that this produces values of 1 (for No) and 2 (for Yes), rather than 0 and 1. So, I simply subtract 1 from the result, and we get what we need.

```
toy$out2 <- as.numeric(toy$out2.event) - 1
```

2.2.2 Testing Your Code - Sanity Checks

Before I move on, I'll do a series of sanity checks to make sure that our new variables are defined as we want them, by producing a series of small tables comparing the new variables to those originally included in the data set.

```
table(toy$treated_f, toy$treated)
```

	0	1
Treated	0	70
Control	130	0

```
table(toy$covB_f, toy$covB)
```

	0	1
No B	126	0
Has B	0	74

```
table(toy$out2_f, toy$out2.event)
```

	No	Yes
Event	0	97
No Event	103	0

```
table(toy$out2, toy$out2.event)
```

	No	Yes
0	103	0
1	0	97

```
table(toy$out2, toy$out2_f)
```

	Event	No	Event
0	0	103	
1	97	0	

Everything looks OK:

- `treated_f` correctly captures the information in `treated`, with the label Treated above the label Control in the rows of the table, facilitating standard epidemiological format.
- `covB_f` also correctly captures the `covB` information, placing “Has B” last.
- `out2_f` correctly captures and re-orders the labels from the original `out2.event`
- `out2` shows the data correctly (as compared to the original `out2.event`) with 0-1 coding.

2.3 Dealing with Variables including More than Two Categories

When we have a multi-categorical (more than two categories) variable, like `covF`, we will want to have

- both a text version of the variable with sensibly ordered levels, as a factor in R, as well as
- a series of numeric indicator variables (taking the values 0 or 1) for the individual levels.

```
summary(toy$covF)
```

1-Low 2-Middle 3-High

From the `summary` output, we can see that we're all set for the text version of `covF`, as what we have currently is a factor with three levels, labeled 1-Low, 2-Middle and 3-High. This list of variables should work out well for us, as it preserves the ordering in a table and permits us to see the names, too. If we'd used just Low, Middle and High, then when R sorted a table into alphabetical order, we'd have High, then Low, then Middle - not ideal.

2.3.1 A Brief Digression

Suppose, for the moment, that a different categorical variable had been included in our data set - this one, which we'll call `cat4`, has four levels, called (in the imported data: 1, 2, 3 and 4) - I'd turn this into a factor using this command:

```
cat4.f <- factor(cat4, levels=c(1,2,3,4), labels=c("Group 1", "Group 2", "Group 3", "Group 4"))
```

or, perhaps, instead, something like this:

```
cat4.f <- factor(cat4, levels=c(1,2,3,4), labels=c("1-Lowest", "2-Low", "3-High", "4-Highest"))
```

2.3.2 Preparing Indicator Variables for `covF`

So, all we need to do for `covF` is prepare indicator variables. We can either do this for all levels, or select one as the baseline, and do the rest. Here, I'll show them all.

```
## Re-expressing the Multi-Categorical Variable
toy$covF.Low <- as.numeric(toy$covF=="1-Low")
toy$covF.Middle <- as.numeric(toy$covF=="2-Middle")
toy$covF.High <- as.numeric(toy$covF=="3-High")
```

And now, some more sanity checks for the `covF` information:

```
table(toy$covF, toy$covF.Low)
```

```
      0  1
1-Low  0 67
2-Middle 84  0
3-High  49  0
```

```
table(toy$covF, toy$covF.Middle)
```

```
      0  1
1-Low  67  0
2-Middle  0 84
3-High  49  0
```

```
table(toy$covF, toy$covF.High)
```

```
      0  1
1-Low  67  0
2-Middle 84  0
3-High  0 49
```


2.4 Creating the Transformation and Product Terms

Remember that we have reason to believe that the square of `covA` as well as the interaction of `covB` with `covC` and also `covB` with `covD` will have an impact on treatment assignment. It will be useful to have these transformations in our data set for modeling and summarizing. I will use `covB` in its numeric (0,1) form (rather than as a factor - `covB.f`) when creating product terms, as shown below.

```
toy$Asqr <- toy$covA^2
toy$BC <- toy$covB*toy$covC
toy$BD <- toy$covB*toy$covD
```

3 Data Set After Cleaning

3.1 Glimpse

```
glimpse(toy)
```

```
Observations: 200
Variables: 21
$ subject    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,...
$ treated    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ covA       <dbl> 0.12, 0.05, 0.68, 0.46, 0.34, 0.15, 0.18, 0.28, 0....
$ covB       <int> 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,...
$ covC       <dbl> 7.6, 11.2, 7.7, 9.5, 9.9, 10.4, 13.7, 11.0, 9.8, 1...
$ covD       <dbl> 6.7, 11.7, 9.3, 9.4, 10.7, 12.2, 8.5, 9.2, 10.2, 1...
$ covE       <int> 8, 7, 12, 12, 9, 6, 9, 6, 10, 11, 4, 12, 16, 17, 1...
$ covF       <fctr> 3-High, 1-Low, 2-Middle, 2-Middle, 2-Middle, 3-Hi...
$ out1.cost  <int> 46, 44, 47, 42, 30, 28, 60, 62, 48, 54, 34, 38, 50...
$ out2.event <fctr> Yes, Yes, No, No, No, No, Yes, No, No, Yes, Yes, ...
$ out3.time  <int> 99, 112, 89, 121, 120, 91, 101, 101, 103, 89, 115,...
$ treated_f  <fctr> Control, Control, Control, Control, Control, Cont...
$ covB_f     <fctr> Has B, No B, No B, No B, Has B, No B, No B, Has B...
$ out2_f     <fctr> Event, Event, No Event, No Event, No Event, No Ev...
$ out2       <dbl> 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,...
$ covF.Low   <dbl> 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0,...
$ covF.Middle <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1,...
$ covF.High  <dbl> 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,...
$ Asqr       <dbl> 0.0144, 0.0025, 0.4624, 0.2116, 0.1156, 0.0225, 0....
$ BC         <dbl> 7.6, 0.0, 0.0, 0.0, 9.9, 0.0, 0.0, 11.0, 0.0, 10.8...
$ BD         <dbl> 6.7, 0.0, 0.0, 0.0, 10.7, 0.0, 0.0, 9.2, 0.0, 12.1...
```

3.2 Summary

```
summary(toy)
```

subject	treated	covA	covB
Min. : 1.00	Min. :0.00	Min. :0.0400	Min. :0.00
1st Qu.: 50.75	1st Qu.:0.00	1st Qu.:0.2675	1st Qu.:0.00
Median :100.50	Median :0.00	Median :0.5100	Median :0.00
Mean :100.50	Mean :0.35	Mean :0.4995	Mean :0.37
3rd Qu.:150.25	3rd Qu.:1.00	3rd Qu.:0.7300	3rd Qu.:1.00

Max. :200.00	Max. :1.00	Max. :0.9900	Max. :1.00
covC	covD	covE	covF
Min. : 4.200	Min. : 5.500	Min. : 2.00	1-Low :67
1st Qu.: 8.600	1st Qu.: 8.500	1st Qu.: 8.00	2-Middle:84
Median : 9.850	Median : 9.900	Median :10.00	3-High :49
Mean : 9.714	Mean : 9.863	Mean :10.21	
3rd Qu.:10.725	3rd Qu.:11.100	3rd Qu.:12.00	
Max. :16.200	Max. :14.400	Max. :19.00	
out1.cost	out2.event	out3.time	treated_f covB_f
Min. :24.00	No :103	Min. : 79.0	Treated: 70 No B :126
1st Qu.:39.00	Yes: 97	1st Qu.:101.0	Control:130 Has B: 74
Median :49.00		Median :109.0	
Mean :50.42		Mean :109.2	
3rd Qu.:58.00		3rd Qu.:118.0	
Max. :80.00		Max. :151.0	
out2_f	out2	covF.Low	covF.Middle
Event : 97	Min. :0.000	Min. :0.000	Min. :0.00
No Event:103	1st Qu.:0.000	1st Qu.:0.000	1st Qu.:0.00
	Median :0.000	Median :0.000	Median :0.00
	Mean :0.485	Mean :0.335	Mean :0.42
	3rd Qu.:1.000	3rd Qu.:1.000	3rd Qu.:1.00
	Max. :1.000	Max. :1.000	Max. :1.00
covF.High	Asqr	BC	BD
Min. :0.000	Min. :0.00160	Min. : 0.000	Min. : 0.000
1st Qu.:0.000	1st Qu.:0.07157	1st Qu.: 0.000	1st Qu.: 0.000
Median :0.000	Median :0.26020	Median : 0.000	Median : 0.000
Mean :0.245	Mean :0.32389	Mean : 3.462	Mean : 3.635
3rd Qu.:0.000	3rd Qu.:0.53290	3rd Qu.: 8.450	3rd Qu.: 9.300
Max. :1.000	Max. :0.98010	Max. :13.700	Max. :13.200

3.3 Table 1

Note that the factors I created for the out2 outcome are not well ordered for a Table 1, but are well ordered for other tables we'll fit later. So, in this case, I'll use the numeric version of the out2 outcome, but the new factor representations of covB and treated.

```
varlist = c("covA", "covB_f", "covC", "covD", "covE", "covF",
            "out1.cost", "out2", "out3.time")
factorlist = c("covB_f", "covF", "out2")
CreateTableOne(vars = varlist, strata = "treated_f",
               data = toy, factorVars = factorlist)
```

	Stratified by treated_f			
	Treated	Control	p	test
n	70	130		
covA (mean (sd))	0.59 (0.22)	0.45 (0.29)	<0.001	
covB_f = Has B (%)	34 (48.6)	40 (30.8)	0.020	
covC (mean (sd))	9.63 (2.02)	9.76 (2.02)	0.649	
covD (mean (sd))	10.10 (1.88)	9.74 (1.77)	0.183	
covE (mean (sd))	9.67 (2.74)	10.50 (3.54)	0.090	
covF (%)			0.266	
1-Low	19 (27.1)	48 (36.9)		
2-Middle	30 (42.9)	54 (41.5)		
3-High	21 (30.0)	28 (21.5)		

```

out1.cost (mean (sd))  60.61 (16.56)  44.94 (9.69)  <0.001
out2 = 1 (%)           39 (55.7)      58 (44.6)   0.177
out3.time (mean (sd)) 113.44 (10.75) 106.95 (11.94) <0.001

```

4 The 13 Tasks We'll Tackle in this Example

1. Ignoring the covariate information, what is the unadjusted point estimate (and 95% confidence interval) for the effect of the treatment on each of the three outcomes (`out1.cost`, `out2.event`, and `out3.time`)?
2. Assume that theory suggests that the square of `covA`, as well as the interactions of `covB` with `covC` and `covB` with `covD` should be related to treatment assignment. Fit a propensity score model to the data, using the six covariates (A-F) and the three transformations (A^2 , and the B-C and B-D interactions.) Plot the resulting propensity scores, by treatment group, in an attractive and useful way.
3. Use Rubin's Rules to assess the overlap of the propensity scores and the individual covariates prior to the use of any propensity score adjustments.
4. Use 1:1 greedy matching to match all 70 treated subjects to control subjects without replacement on the basis of the linear propensity for treatment. Evaluate the degree of covariate imbalance before and after propensity matching for each of the six covariates, and present the pre- and post-match standardized differences and variance ratios for the covariates, as well as the square term and interactions, as well as both the raw and linear propensity score in appropriate plots. Now, build a new data frame containing the propensity-matched sample, and use it to first check Rubin's Rules after matching.
5. Now, use the matched sample data set to evaluate the treatment's average causal effect on each of the three outcomes. In each case, specify a point estimate (and associated 95% confidence interval) for the effect of being treated (as compared to being a control subject) on the outcome. Compare your results to the automatic versions reported by the Matching package when you include the outcome in the matching process.
6. Now, instead of matching, instead subclassify the subjects into quintiles by the raw propensity score. Display the balance in terms of standardized differences by quintile for the covariates, their transformations, and the propensity score in an appropriate table or plot(s). Are you satisfied?
7. Regardless of your answer to the previous question, use the propensity score quintile subclassification approach to find a point estimate (and 95% confidence interval) for the effect of the treatment on each outcome.
8. Now using a reasonable propensity score weighting strategy, assess the balance of each covariate, the transformations and the linear propensity score prior to and after propensity weighting. Is the balance after weighting satisfactory?
9. Using propensity score weighting to evaluate the treatment's effect, developing a point estimate and 95% CI for the average causal effect of treatment on each outcome.
10. Finally, use direct adjustment for the linear propensity score on the entire sample to evaluate the treatment's effect, developing a point estimate and 95% CI for each outcome.
11. Now, try a double robust approach. Weight, then adjust for linear propensity score.
12. Compare your conclusions about the average causal effect obtained in the following six ways to each other. What happens and why? Which of these methods seems most appropriate given the available information?
 - without propensity adjustment,
 - after propensity matching,
 - after propensity score subclassification,
 - after propensity score weighting,
 - after adjusting for the propensity score directly, and
 - after weighting then adjusting for the PS, to each other.
13. Perform a sensitivity analysis for your matched samples analysis and the first outcome (`out1.cost`) if it turns out to show a statistically significant treatment effect.

5 Task 1. Ignoring covariates, estimate the effect of treatment vs. control on...

5.1 Outcome 1 (a continuous outcome)

Our first outcome describes a quantitative measure, cost, and we're asking what the effect of `treatment` as compared to `control` is on that outcome. Starting with brief numerical summaries:

```
toy %>%
  group_by(treated_f) %>%
  summarize(n = length(out1.cost), mean = mean(out1.cost),
            sd = sd(out1.cost), min = min(out1.cost),
            Q1 = quantile(out1.cost, 0.25), median = median(out1.cost),
            Q3 = quantile(out1.cost, 0.75), max = max(out1.cost))
```

```
# A tibble: 2 x 9
  treated_f      n mean   sd   min   Q1 median   Q3   max
  <fctr>    <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Treated      70  60.6  16.6  24.0  50.2  62.0  75.8  80.0
2 Control     130  44.9   9.69  24.0  38.0  47.0  51.0  78.0
```

It looks like the Treated group has higher costs than the Control group. To model this, we could use a linear regression model to obtain a point estimate and 95% confidence interval. Here, I prefer to use the numeric version of the `treated` variable, with 0 = “control” and 1 = “treated”.

```
unadj.out1 <- lm(out1.cost ~ treated, data=toy)
summary(unadj.out1); confint(unadj.out1, level = 0.95) ## provides treated effect and CI estimates
```

Call:

```
lm(formula = out1.cost ~ treated, data = toy)
```

Residuals:

Min	1Q	Median	3Q	Max
-36.614	-7.945	2.062	9.062	33.062

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	44.938	1.098	40.932	< 2e-16 ***
treated	15.676	1.856	8.447	6.36e-15 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.52 on 198 degrees of freedom

Multiple R-squared: 0.2649, Adjusted R-squared: 0.2612

F-statistic: 71.35 on 1 and 198 DF, p-value: 6.364e-15

	2.5 %	97.5 %
(Intercept)	42.7734	47.10352
treated	12.0162	19.33544

We can store these results in a data frame, with the `tidy` function from the `broom` package.

```
temp <- tidy(unadj.out1, conf.int = TRUE, conf.level = 0.95)
temp
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
------	----------	-----------	-----------	---------	----------	-----------

```
1 (Intercept) 44.93846 1.097891 40.931619 1.434004e-98 42.7734 47.10352
2 treated 15.67582 1.855775 8.447051 6.363978e-15 12.0162 19.33544
```

Our unadjusted treatment effect estimate is an increase of 15.68 in cost, with 95% confidence interval (12.02, 19.34).

I should mention that the `broom` package also has a useful function called `glance` which lets you get some detailed summaries of the model.

```
glance(unadj.out1)
```

```
   r.squared adj.r.squared   sigma statistic    p.value df    logLik
1 0.2649043   0.2611917 12.51788  71.35267 6.363978e-15  2 -788.2144
   AIC      BIC deviance df.residual
1 1582.429 1592.324 31026.09         198
```

5.2 Outcome 2 (a binary outcome)

5.2.1 Using a 2x2 table in standard epidemiological format

Thanks to our preliminary cleanup, it's relatively easy to obtain a table in standard epidemiological format comparing treated to control subjects in terms of `out2`:

```
table(toy$treated_f, toy$out2_f)
```

	Event	No Event
Treated	39	31
Control	58	72

Note that the exposure is in the rows, with “Having the Exposure” or “Treated” at the top, and the outcome is in the columns, with “Yes” or “Outcome Occurred” or “Event Occurred” on the left, so that the top left cell count describes people that had both the exposure and the outcome. That's *standard epidemiological format*, just what we need for the `twoby2` function in the `Epi` package.

```
twoby2(table(toy$treated_f, toy$out2_f))
```

2 by 2 table analysis:

Outcome : Event

Comparing : Treated vs. Control

	Event	No Event	P(Event)	95% conf. interval
Treated	39	31	0.5571	0.4398 0.6685
Control	58	72	0.4462	0.3631 0.5324

	95% conf. interval
Relative Risk: 1.2488	0.9406 1.6579
Sample Odds Ratio: 1.5617	0.8702 2.8028
Conditional MLE Odds Ratio: 1.5582	0.8354 2.9261
Probability difference: 0.1110	-0.0335 0.2489

Exact P-value: 0.1413

Asymptotic P-value: 0.1352

Eventually, we will be interested in at least two measures - the odds ratio and the risk (probability) difference estimates, and their respective confidence intervals.

- For a *difference in risk*, our unadjusted treatment effect estimate is an increase of 11.1 percentage points as compared to control, with 95% CI of (-3.4, +24.9) percentage points.
- For an *odds ratio*, our unadjusted treatment effect estimate is an odds ratio of 1.56 (95% CI = 0.87, 2.80) for the event occurring with treatment as compared to control.
- For neither measure is the observed unadjusted effect statistically significant at a 95% confidence level.

5.2.2 Using a logistic regression model

For the odds ratio estimate, we can use a simple logistic regression model to estimate the unadjusted treatment effect, resulting in essentially the same answer. We'll use the numerical (0/1) format to represent binary information, as follows.

```
unadj.out2 <- glm(out2 ~ treated, data=toy, family=binomial())
summary(unadj.out2)
```

Call:

```
glm(formula = out2 ~ treated, family = binomial(), data = toy)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.276	-1.087	-1.087	1.270	1.270

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-0.2162	0.1764	-1.226	0.220
treated	0.4458	0.2984	1.494	0.135

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 277.08 on 199 degrees of freedom
Residual deviance: 274.83 on 198 degrees of freedom
AIC: 278.83

Number of Fisher Scoring iterations: 3

```
exp(coef(unadj.out2)) # produces odds ratio estimate
```

(Intercept)	treated
0.8055556	1.5617353

```
exp(confint(unadj.out2)) # produces 95% CI for odds ratio
```

Waiting for profiling to be done...

	2.5 %	97.5 %
(Intercept)	0.5683258	1.136896
treated	0.8721205	2.816455

And, again, we can use the `tidy` function in the `broom` package to build a tibble of the key parts of the output. Note that by including the `exponentiate = TRUE` command, our results in the `treated` row describe the odds ratio, rather than the log odds.

```
tidy(unadj.out2, conf.int = TRUE, exponentiate = TRUE)
```

	term	estimate	std.error	statistic	p.value	conf.low	conf.high
1	(Intercept)	0.8055556	0.1764365	-1.225501	0.2203865	0.5683258	1.136896
2	treated	1.5617353	0.2983755	1.494082	0.1351541	0.8721205	2.816455

- Our odds ratio estimate remains about 1.56, with 95% confidence interval ranging from 0.9 to 2.8, again showing no evidence of a statistically significant impact of the treatment on the occurrence of the event described by `out2`.
- For practical purposes, the odds ratio and 95% confidence interval obtained here matches the methodology for the `twoby2` function.
- The approach implemented in the `twoby2` function produces slightly less conservative (i.e. narrower) confidence intervals for the effect estimate than does the approach used in the logistic regression model.

Glancing at the summaries of the model may be helpful in some settings, as well.

```
glance(unadj.out2)
```

	null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual
1	277.0788	199	-137.416	278.832	285.4286	274.832	198

5.3 Outcome 3 (a time-to-event outcome with right censoring)

Our `out3.time` variable is a variable indicating the time before the event described in `out2` occurred. This happened to 97 of the 200 subjects in the data set. For the other 103 subjects who left the study before their event occurred, we have the time before censoring. We can see the results of this censoring in the survival object describing each treatment group.

Here, for instance, is the survival object for the *treated* subjects - the third subject listed here is censored - had the event at some point after 124 weeks (124+) but we don't know precisely when after 124 weeks.

```
Surv(toy$out3.time, toy$out2.event == "Yes")[toy$treated == 1]
```

```
[1] 89 100 124+ 111 103+ 112 109 104 98+ 129 114 125 121+ 97
[15] 137+ 103 110 106+ 108 136+ 96 125+ 118+ 99+ 101 109 128 111+
[29] 120+ 118 115 129+ 118+ 106 106+ 118+ 121 120 107 119+ 105 112
[43] 106+ 132+ 126+ 130+ 115+ 108 99 122 110 102 114+ 120+ 100 108
[57] 120 118+ 117+ 123+ 109 122 126+ 126+ 118+ 125+ 103 94 106 105
```

- To see the controls, we could use `Surv(toy$out3.time, toy$out2.event=="Yes")[toy$treated==0]`

To deal with the right censoring, we'll use the `survival` package to fit a simple unadjusted Cox proportional hazards model to assess the relative hazard of having the event at a particular time point among treated subjects as compared to controls.

```
unadj.out3 <- coxph(Surv(out3.time, out2.event=="Yes") ~ treated, data=toy)
summary(unadj.out3) ## exp(coef) section indicates relative risk estimate and 95% CI
```

Call:

```
coxph(formula = Surv(out3.time, out2.event == "Yes") ~ treated,
      data = toy)
```

n= 200, number of events= 97

	coef	exp(coef)	se(coef)	z	Pr(> z)
treated	-0.1535	0.8577	0.2086	-0.736	0.462

```

      exp(coef) exp(-coef) lower .95 upper .95
treated    0.8577    1.166    0.5698    1.291

```

```

Concordance= 0.532 (se = 0.028 )
Rsquare= 0.003 (max possible= 0.988 )
Likelihood ratio test= 0.55 on 1 df, p=0.46
Wald test = 0.54 on 1 df, p=0.462
Score (logrank) test = 0.54 on 1 df, p=0.4615

```

The relative hazard rate is shown in the `exp(coef)` section of the output. Our unadjusted treatment model suggests that the hazard of the outcome is smaller (but not significantly smaller) in the treated group than in the control group. Our estimate is that this relative hazard rate for occurrence of the event associated with treatment as compared to control is 0.86 with a 95% confidence interval of (0.57, 1.29).

Yes, you can tidy this model, as well, using the `broom` package.

```
tidy(unadj.out3, exponentiate = TRUE)
```

```

      term estimate std.error statistic p.value conf.low conf.high
1 treated 0.8577207 0.2086392 -0.7356088 0.4619688 0.5698386 1.29104

```

Glancing at the summaries of the model may be helpful, too.

```
glance(unadj.out3)
```

```

      n nevent statistic.log p.value.log statistic.sc p.value.sc
1 200     97    0.5460054 0.4599545    0.5421382 0.4615479
      statistic.wald p.value.wald r.squared r.squared.max concordance
1          0.54    0.4619688 0.002726304    0.9884724    0.5318705
      std.error.concordance logLik AIC BIC
1          0.02796153 -446.0285 894.057 896.6317

```

It's wise, whenever fitting a Cox proportional hazards model, to assess the proportional hazards assumption. One way to do this is to run a simple test in R - from which we can obtain a plot, if we like. The idea is for the plot to show no clear patterns over time, and look pretty much like a horizontal line, while we would like the test to be non-significant - if that's the case, our proportional hazards assumption is likely OK.

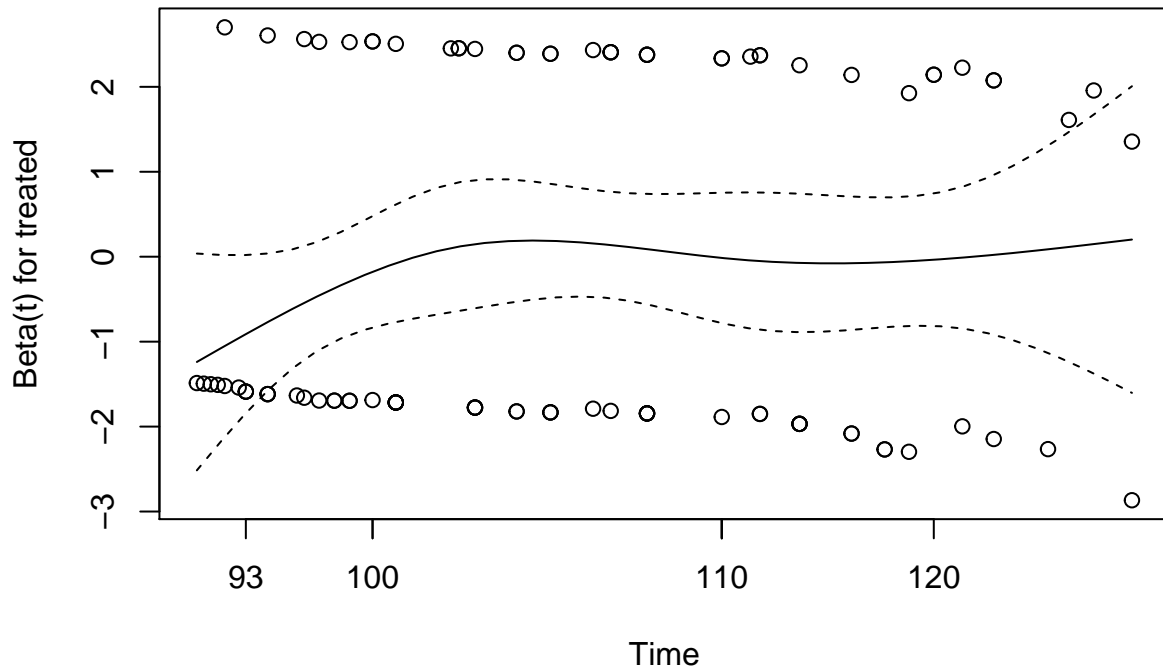
```
cox.zph(unadj.out3)
```

```

      rho chisq p
treated 0.118 1.34 0.247

```

```
plot(cox.zph(unadj.out3), var="treated")
```

If the proportional hazards assumption is clearly violated (here it isn't), call a statistician.

5.4 Unadjusted Estimates of Treatment Effect on Outcomes

So, our unadjusted average treatment effect estimates (in each case comparing treated subjects to control subjects) are thus:

Est. Treatment Effect (95% CI)	Outcome 1 (Cost diff.)	Outcome 2 (Risk diff.)	Outcome 2 (Odds Ratio)	Outcome 3 (Relative Hazard Rate)
No covariate adjustment (unadjusted)	15.7 (12.0, 19.3)	+0.11 (-0.03, +0.25)	1.56 (0.87, 2.82)	0.86 (0.57, 1.29)

6 Task 2. Fit the propensity score model, then plot the PS-treatment relationship

I'll use a logistic regression model

```
psmodel <- glm(treated ~ covA + covB + covC + covD + covE + covF +
               Asqr + BC + BD, family=binomial(), data=toy)
arm::display(psmodel)
```

```
glm(formula = treated ~ covA + covB + covC + covD + covE + covF +
     Asqr + BC + BD, family = binomial(), data = toy)
               coef.est coef.se
(Intercept)   -5.24      1.94
covA           10.23      3.11
covB            1.07      2.47
covC            0.01      0.11
covD            0.24      0.12
covE           -0.14      0.05
covF2-Middle   0.19      0.40
covF3-High     0.59      0.45
Asqr          -6.91      2.83
BC            -0.03      0.17
BD             0.01      0.20
---
n = 200, k = 11
residual deviance = 216.9, null deviance = 259.0 (difference = 42.1)
```

Having fit the model, my first step will be to save the raw and linear propensity score values to the main toy example tibble.

```
toy$ps <- psmodel$fitted
toy$linps <- psmodel$linear.predictors
```

6.1 Comparing the Distribution of Propensity Score Across the Two Treatment Groups

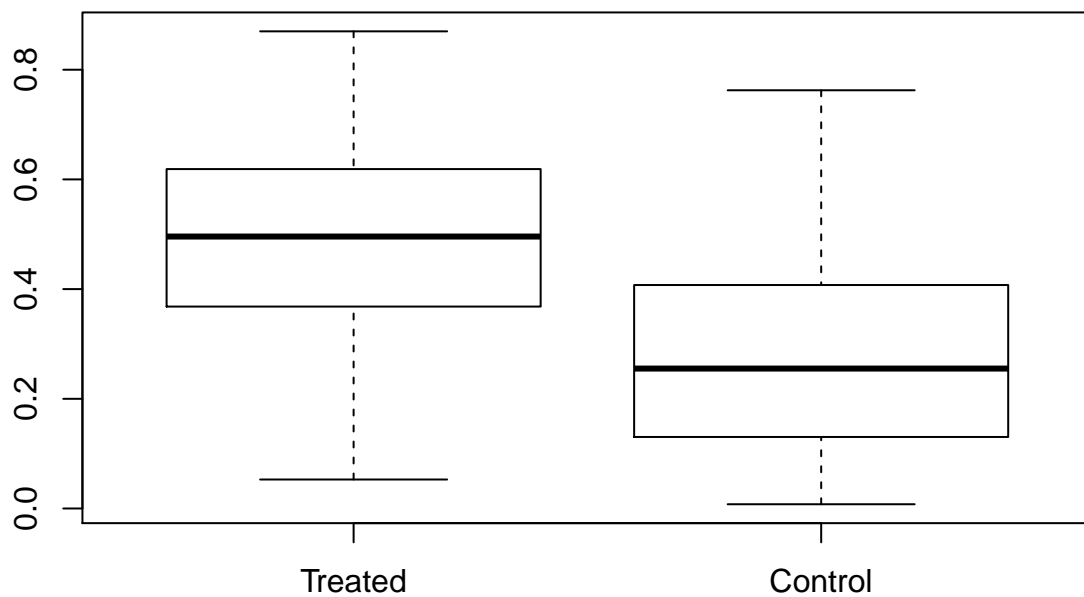
Now, I can use these saved values to assess the propensity model.

```
by(toy$ps, toy$treated_f, summary)
```

```
toy$treated_f: Treated
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.05289 0.36875 0.49584 0.47608 0.61527 0.86993
-----
toy$treated_f: Control
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.007769 0.132215 0.255136 0.282110 0.406386 0.762492
```

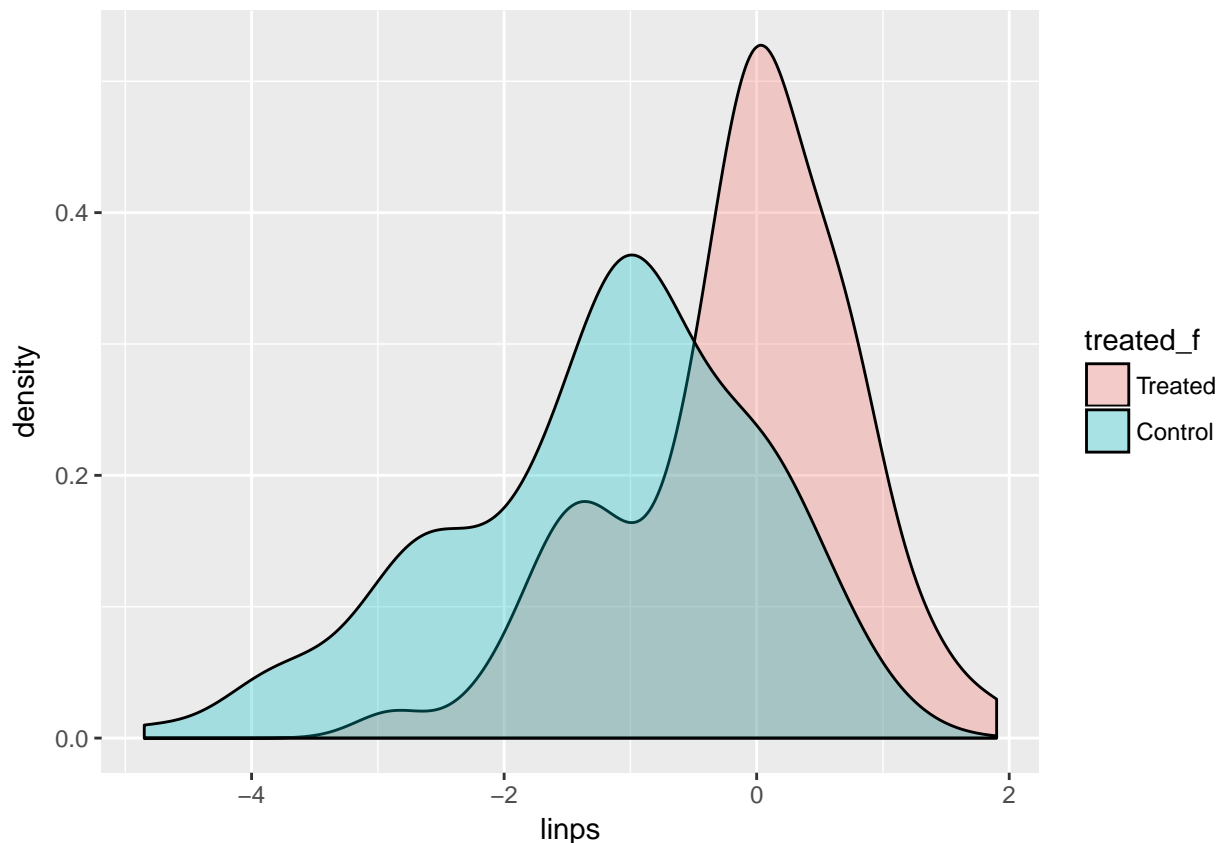
The simplest plot is probably a boxplot, but it's not very granular.

```
boxplot(toy$ps ~ toy$treated_f)
```



I'd rather get a fancier plot to compare the distributions of the propensity score across the two treatment groups, perhaps using a smoothed density estimate, as shown below. Here, I'll show the distributions of the linear propensity score, the log odds of treatment.

```
ggplot(toy, aes(x = linps, fill = treated_f)) +  
  geom_density(alpha = 0.3)
```



We see a fair amount of overlap across the two treatment groups. I'll use Rubin's Rules in the next section to help assess the amount of overlap at this point, before any adjustments for the propensity score.

7 Task 3. Rubin's Rules to Check Overlap Before Propensity Adjustment

In his 2001 article¹ about using propensity scores to design studies, as applied to studies of the causal effects of the conduct of the tobacco industry on medical expenditures, Donald Rubin proposed three “rules” for assessing the overlap / balance of covariates appropriately before and after propensity adjustment. Before an outcome is evaluated using a regression analysis (perhaps supplemented by a propensity score adjustment through matching, weighting, subclassification or even direct adjustment), there are three checks that should be performed.

When we do a propensity score analysis, it will be helpful to perform these checks as soon as the propensity model has been estimated, even before any adjustments take place, to see how well the distributions of covariates overlap. After using the propensity score, we hope to see these checks meet the standards below. In what follows, I will describe each standard, and demonstrate its evaluation using the propensity score model we just fit, and looking at the original toy data set, without applying the propensity score in any way to do adjustments.

¹Rubin DB 2001 Using Propensity Scores to Help Design Observational Studies: Application to the Tobacco Litigation. Health Services & Outcomes Research Methodology 2: 169-188..

7.1 Rubin's Rule 1

Rubin's Rule 1 states that the absolute value of the standardized difference of the linear propensity score, comparing the treated group to the control group, should be close to 0, ideally below 10%, and in any case less than 50%. If so, we may move on to Rule 2.

To evaluate this rule in the toy example, we'll run the following code to place the right value into a variable called `rubin1.unadj` (for Rubin's Rule 1, unadjusted).

```
rubin1.unadj <- with(toy,
  abs(100*(mean(linps[treated==1]) - mean(linps[treated==0]))/sd(linps)))
rubin1.unadj
```

```
[1] 88.11531
```

What this does is calculate the (absolute value of the) standardized difference of the linear propensity score comparing treated subjects to control subjects.

- We want this value to be close to 0, and certainly less than 50 in order to push forward to outcomes analysis without further adjustment for the propensity score.
- Clearly, here, with a value of 88%, we can't justify simply running an unadjusted regression model, be it a linear, logistic or Cox model - we've got observed selection bias, and need to actually apply the propensity score somehow in order to account for this.
- So, we'll need to match, subclassify, weight or directly adjust for propensity here.

Since we've failed Rubin's 1st Rule, in some sense, we're done checking the rules, because we clearly need to further adjust for observed selection bias - there's no need to prove that further through checking Rubin's 2nd and 3rd rules. But we'll do it here to show what's involved.

7.2 Rubin's Rule 2

Rubin's Rule 2 states that the ratio of the variance of the linear propensity score in the treated group to the variance of the linear propensity score in the control group should be close to 1, ideally between 4/5 and 5/4, but certainly not very close to or exceeding 1/2 and 2. If so, we may move on to Rule 3.

To evaluate this rule in the toy example, we'll run the following code to place the right value into a variable called `rubin2.unadj` (for Rubin's Rule 2, unadjusted).

```
rubin2.unadj <- with(toy, var(linps[treated==1])/var(linps[treated==0]))
rubin2.unadj
```

```
[1] 0.5835438
```

This is the ratio of variances of the linear propensity score comparing treated subjects to control subjects. We want this value to be close to 1, and certainly between 0.5 and 2. In this case, we pass Rule 2, if just barely.

7.3 Rubin's Rule 3

For Rubin's Rule 3, we begin by calculating regression residuals for each covariate of interest (usually, each of those included in the propensity model) regressed on a single predictor - the linear propensity score. We then look to see if the ratio of the variance of the residuals of this model for the treatment group divided by the variance of the residuals of this model for the control group is close to 1. Again, ideally this will fall between 4/5 and 5/4 for each covariate, but certainly between 1/2 and 2. If so, then the use of regression models seems well justified.

To evaluate Rubin's 3rd Rule, we'll create a little function to help us do the calculations.

```
## General function rubin3 to help calculate Rubin's Rule 3
rubin3 <- function(data, covlist, linps) {
  covlist2 <- as.matrix(covlist)
  res <- NA
  for(i in 1:ncol(covlist2)) {
    cov <- as.numeric(covlist2[,i])
    num <- var(resid(lm(cov ~ data$linps)))[data$exposure == 1]
    den <- var(resid(lm(cov ~ data$linps)))[data$exposure == 0]
    res[i] <- round(num/den, 3)
  }
  final <- data_frame(name = names(covlist), resid.var.ratio = res)
  return(final)
}
```

Now, then, applying the rule to our sample prior to propensity score adjustment, we get the following result. Note that I'm using the indicator variable forms for the covF information.

```
cov.sub <- dplyr::select(toy,
                        covA, covB, covC, covD, covE,
                        covF.Middle, covF.High, Asqr, BC, BD)

toy$exposure <- toy$treated

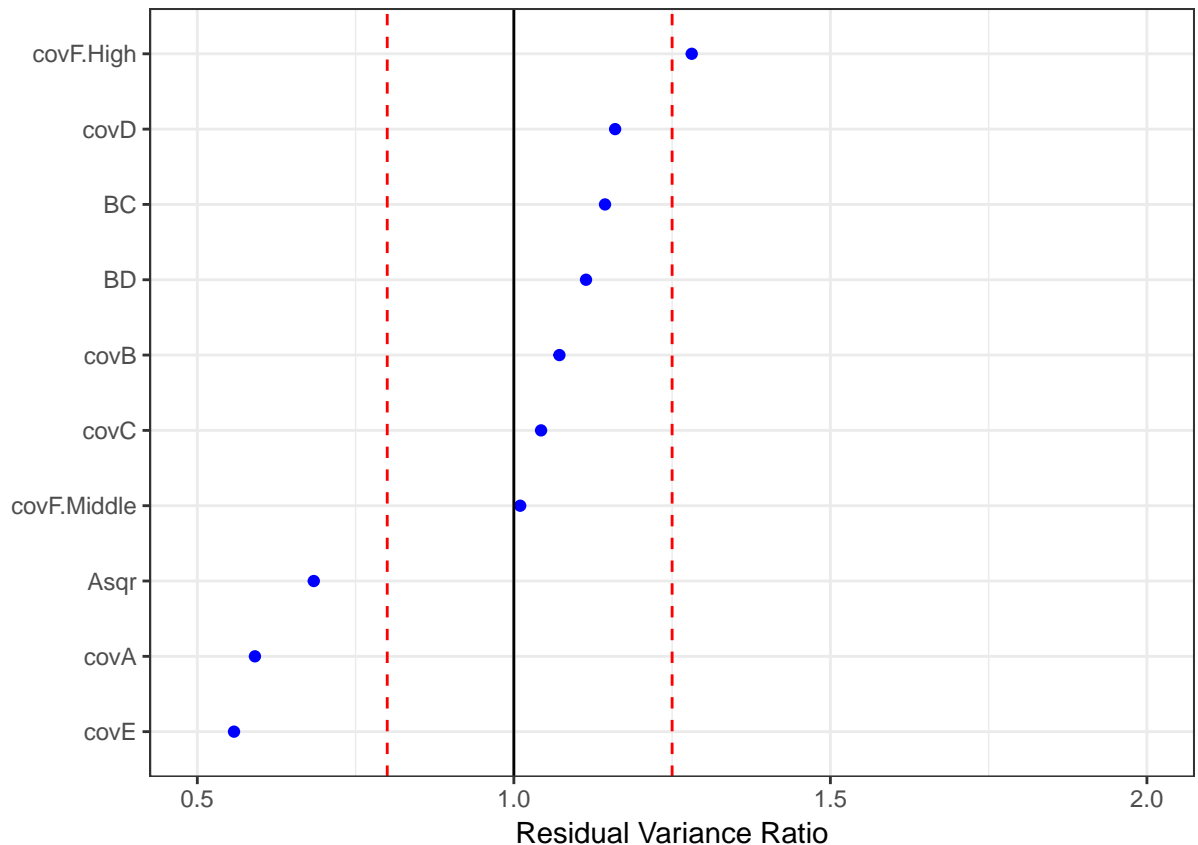
rubin3.unadj <- rubin3(data = toy, covlist = cov.sub, linps = linps)
rubin3.unadj
```

```
# A tibble: 10 x 2
  name      resid.var.ratio
  <chr>      <dbl>
1 covA      0.591
2 covB      1.07
3 covC      1.04
4 covD      1.16
5 covE      0.558
6 covF.Middle 1.01
7 covF.High  1.28
8 Asqr      0.684
9 BC        1.14
10 BD       1.11
```

Some of these covariates look to have residual variance ratios near 1, while others are further away, but all are within the (0.5, 2.0) range. So we'd pass Rule 3 here, although we'd clearly like to see some covariates (A and E, in particular) with ratios closer to 1.

7.3.1 A Cleveland Dot Chart of the Rubin's Rule 3 Results

```
ggplot(rubin3.unadj, aes(x = resid.var.ratio, y = reorder(name, resid.var.ratio))) +
  geom_point(col = "blue") +
  theme_bw() +
  xlim(0.5, 2.0) +
  geom_vline(aes(xintercept = 1)) +
  geom_vline(aes(xintercept = 4/5), linetype = "dashed", col = "red") +
  geom_vline(aes(xintercept = 5/4), linetype = "dashed", col = "red") +
  labs(x = "Residual Variance Ratio", y = "")
```



We see several values between 0.5 and 0.8, but nothing outside (0.5, 2).

8 Task 4. Use 1:1 greedy matching on the linear PS, then check post-match balance

As requested, we'll do 1:1 greedy matching on the linear propensity score without replacement and breaking ties randomly. To start, we won't include an outcome variable in our call to the `Match` function within the `Matching` package. We'll wind up with a match including 70 treated and 70 control subjects.

```
X <- toy$linps ## matching on the linear propensity score
Tr <- as.logical(toy$treated)
match1 <- Match(Tr=Tr, X=X, M = 1, replace=FALSE, ties=FALSE)
summary(match1)
```

```
Estimate... 0
SE..... 0
T-stat..... NaN
p.val..... NA
```

```
Original number of observations..... 200
Original number of treated obs..... 70
Matched number of observations..... 70
Matched number of observations (unweighted). 70
```

8.1 Balance Assessment (Semi-Automated)

Next, we'll assess the balance imposed by this greedy match on our covariates, and their transformations (A^2 and $B \cdot C$ and $B \cdot D$) as well as the raw and linear propensity scores. The default output from the `MatchBalance` function is extensive...

```
set.seed(5001)
mb1 <- MatchBalance(treated ~ covA + covB + covC + covD + covE + covF +
                    Asqr + BC + BD + ps + linps, data=toy,
                    match.out = match1, nboots=500)
```

```
***** (V1) covA *****
```

	Before Matching	After Matching
mean treatment.....	0.59071	0.59071
mean control.....	0.45046	0.55443
std mean diff.....	63.684	16.476
mean raw eQQ diff.....	0.15014	0.054286
med raw eQQ diff.....	0.175	0.05
max raw eQQ diff.....	0.28	0.14
mean eCDF diff.....	0.15106	0.059127
med eCDF diff.....	0.18352	0.042857
max eCDF diff.....	0.28352	0.2
var ratio (Tr/Co).....	0.58886	0.83347
T-test p-value.....	0.00016497	0.29257
KS Bootstrap p-value..	< 2.22e-16	0.084
KS Naive p-value.....	0.0013313	0.12159
KS Statistic.....	0.28352	0.2

```
***** (V2) covB *****
```

	Before Matching	After Matching
mean treatment.....	0.48571	0.48571
mean control.....	0.30769	0.38571
std mean diff.....	35.364	19.865
mean raw eQQ diff.....	0.18571	0.1
med raw eQQ diff.....	0	0
max raw eQQ diff.....	1	1
mean eCDF diff.....	0.089011	0.05
med eCDF diff.....	0.089011	0.05
max eCDF diff.....	0.17802	0.1
var ratio (Tr/Co).....	1.1805	1.0543
T-test p-value.....	0.01552	0.22225

```
***** (V3) covC *****
```

	Before Matching	After Matching
mean treatment.....	9.6257	9.6257
mean control.....	9.7623	9.7329

std mean diff.....	-6.7767	-5.3156
mean raw eQQ diff.....	0.18714	0.26429
med raw eQQ diff.....	0.1	0.2
max raw eQQ diff.....	1.7	1.7
mean eCDF diff.....	0.021871	0.031027
med eCDF diff.....	0.012088	0.028571
max eCDF diff.....	0.078022	0.1
var ratio (Tr/Co).....	0.99092	0.98676
T-test p-value.....	0.64881	0.76698
KS Bootstrap p-value..	0.87	0.804
KS Naive p-value.....	0.94461	0.8752
KS Statistic.....	0.078022	0.1

***** (V4) covD *****

	Before Matching	After Matching
mean treatment.....	10.096	10.096
mean control.....	9.7377	10.103
std mean diff.....	19.074	-0.38055
mean raw eQQ diff.....	0.43143	0.23286
med raw eQQ diff.....	0.4	0.1
max raw eQQ diff.....	1	1.5
mean eCDF diff.....	0.053676	0.024654
med eCDF diff.....	0.050549	0.028571
max eCDF diff.....	0.10989	0.071429
var ratio (Tr/Co).....	1.1249	1.2306
T-test p-value.....	0.19162	0.9803
KS Bootstrap p-value..	0.482	0.98
KS Naive p-value.....	0.64191	0.99407
KS Statistic.....	0.10989	0.071429

***** (V5) covE *****

	Before Matching	After Matching
mean treatment.....	9.6714	9.6714
mean control.....	10.5	10.029
std mean diff.....	-30.199	-13.017
mean raw eQQ diff.....	1.0429	0.87143
med raw eQQ diff.....	1	1
max raw eQQ diff.....	3	3
mean eCDF diff.....	0.058242	0.058095
med eCDF diff.....	0.05	0.057143
max eCDF diff.....	0.12857	0.11429
var ratio (Tr/Co).....	0.60225	0.61843
T-test p-value.....	0.068094	0.48672

KS Bootstrap p-value..	0.258	0.504
KS Naive p-value.....	0.43948	0.75053
KS Statistic.....	0.12857	0.11429

***** (V6) covF2-Middle *****

	Before Matching	After Matching
mean treatment.....	0.42857	0.42857
mean control.....	0.41538	0.38571
std mean diff.....	2.6456	8.5982
mean raw eQQ diff.....	0.014286	0.042857
med raw eQQ diff.....	0	0
max raw eQQ diff.....	1	1
mean eCDF diff.....	0.0065934	0.021429
med eCDF diff.....	0.0065934	0.021429
max eCDF diff.....	0.013187	0.042857
var ratio (Tr/Co).....	1.0152	1.0336
T-test p-value.....	0.85825	0.59098

***** (V7) covF3-High *****

	Before Matching	After Matching
mean treatment.....	0.3	0.3
mean control.....	0.21538	0.27143
std mean diff.....	18.332	6.1901
mean raw eQQ diff.....	0.085714	0.028571
med raw eQQ diff.....	0	0
max raw eQQ diff.....	1	1
mean eCDF diff.....	0.042308	0.014286
med eCDF diff.....	0.042308	0.014286
max eCDF diff.....	0.084615	0.028571
var ratio (Tr/Co).....	1.251	1.0619
T-test p-value.....	0.20201	0.65566

***** (V8) Asqr *****

	Before Matching	After Matching
mean treatment.....	0.39675	0.39675
mean control.....	0.28465	0.36475
std mean diff.....	45.623	13.023
mean raw eQQ diff.....	0.12648	0.063506
med raw eQQ diff.....	0.1308	0.06825
max raw eQQ diff.....	0.2544	0.1568
mean eCDF diff.....	0.15106	0.059127
med eCDF diff.....	0.18352	0.042857
max eCDF diff.....	0.28352	0.2

var ratio (Tr/Co).....	0.72913	0.77694
T-test p-value.....	0.004317	0.43078
KS Bootstrap p-value..	< 2.22e-16	0.084
KS Naive p-value.....	0.0013313	0.12159
KS Statistic.....	0.28352	0.2

***** (V9) BC *****

	Before Matching	After Matching
mean treatment.....	4.5843	4.5843
mean control.....	2.8569	3.5614
std mean diff.....	34.494	20.426
mean raw eQQ diff.....	1.78	1.0229
med raw eQQ diff.....	0	0
max raw eQQ diff.....	8.8	7.8
mean eCDF diff.....	0.099473	0.062458
med eCDF diff.....	0.099451	0.071429
max eCDF diff.....	0.18681	0.12857
var ratio (Tr/Co).....	1.2672	1.1369
T-test p-value.....	0.017031	0.19412
KS Bootstrap p-value..	0.026	0.348
KS Naive p-value.....	0.083513	0.60929
KS Statistic.....	0.18681	0.12857

***** (V10) BD *****

	Before Matching	After Matching
mean treatment.....	4.8157	4.8157
mean control.....	3	3.7786
std mean diff.....	35.245	20.132
mean raw eQQ diff.....	1.8629	1.0371
med raw eQQ diff.....	0	0
max raw eQQ diff.....	9.5	8.3
mean eCDF diff.....	0.10274	0.065306
med eCDF diff.....	0.1011	0.071429
max eCDF diff.....	0.18132	0.1
var ratio (Tr/Co).....	1.2538	1.1086
T-test p-value.....	0.014967	0.2157
KS Bootstrap p-value..	0.03	0.6
KS Naive p-value.....	0.10039	0.8752
KS Statistic.....	0.18132	0.1

***** (V11) ps *****

	Before Matching	After Matching
mean treatment.....	0.47608	0.47608
mean control.....	0.28211	0.40347

std mean diff.....	101.45	37.975
mean raw eQQ diff.....	0.19569	0.073618
med raw eQQ diff.....	0.21857	0.081407
max raw eQQ diff.....	0.26218	0.1385
mean eCDF diff.....	0.26132	0.1249
med eCDF diff.....	0.25165	0.14286
max eCDF diff.....	0.47253	0.25714
var ratio (Tr/Co).....	1.047	1.4126
T-test p-value.....	1.7258e-10	2.591e-06
KS Bootstrap p-value..	< 2.22e-16	0.014
KS Naive p-value.....	9.8491e-10	0.019182
KS Statistic.....	0.47253	0.25714

***** (V12) linps *****

	Before Matching	After Matching
mean treatment.....	-0.13912	-0.13912
mean control.....	-1.2152	-0.45294
std mean diff.....	117.3	34.211
mean raw eQQ diff.....	1.0997	0.3201
med raw eQQ diff.....	1.0798	0.33541
max raw eQQ diff.....	1.9834	0.73393
mean eCDF diff.....	0.26132	0.1249
med eCDF diff.....	0.25165	0.14286
max eCDF diff.....	0.47253	0.25714
var ratio (Tr/Co).....	0.58354	1.4158
T-test p-value.....	3.3878e-11	3.1952e-06
KS Bootstrap p-value..	< 2.22e-16	0.014
KS Naive p-value.....	9.8491e-10	0.019182
KS Statistic.....	0.47253	0.25714

Before Matching Minimum p.value: < 2.22e-16

Variable Name(s): covA Asqr ps linps Number(s): 1 8 11 12

After Matching Minimum p.value: 2.591e-06

Variable Name(s): ps Number(s): 11

The cobalt package has some promising tools for taking this sort of output and turning it into something useful. We'll look at that approach soon. For now, some old-school stuff...

8.2 Extracting, Tabulating and Plotting Standardized Differences (without cobalt)

We'll start by naming the covariates that the MatchBalance output contains...

```
covnames <- c("covA", "covB", "covC", "covD", "covE",
              "covF - Middle", "covF - High",
```

```
"A^2", "B*C", "B*D", "raw PS", "linear PS")
```

The next step is to extract the standardized differences (using the pooled denominator to estimate, rather than the treatment-only denominator used in the main output above.)

```
pre.szd <- NULL; post.szd <- NULL
for(i in 1:length(covnames)) {
  pre.szd[i] <- mb1$BeforeMatching[[i]]$sdiff.pooled
  post.szd[i] <- mb1$AfterMatching[[i]]$sdiff.pooled
}
```

Now, we can build a table of the standardized differences:

```
temp <- data.frame(pre.szd, post.szd, row.names=covnames)
print(temp, digits=3)
```

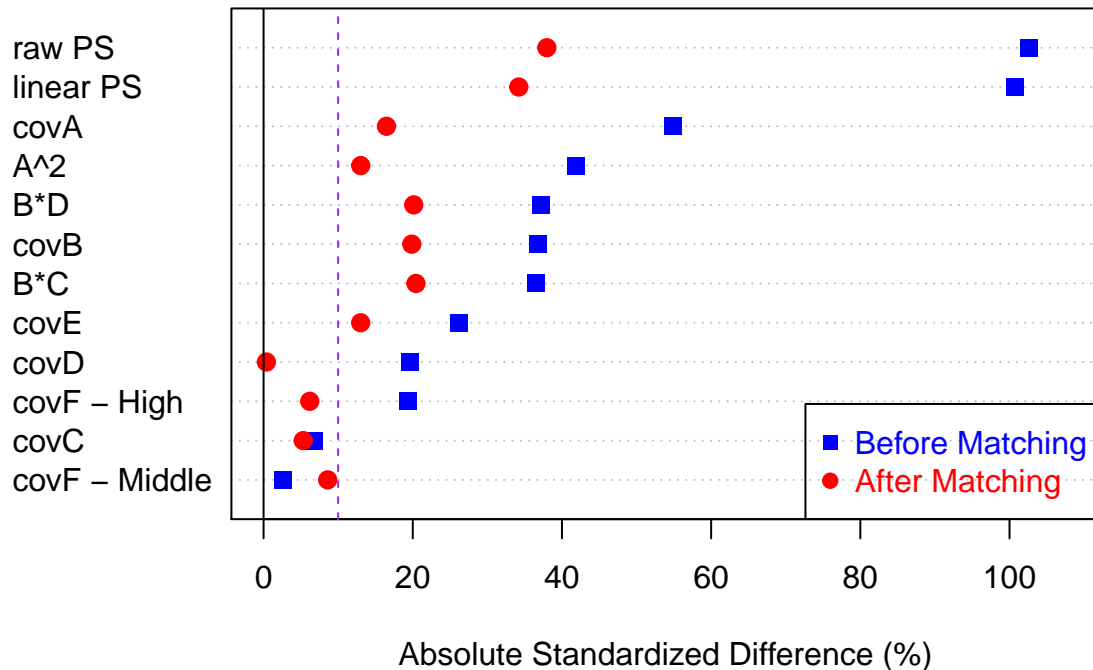
	pre.szd	post.szd
covA	54.83	16.476
covB	36.80	19.865
covC	-6.76	-5.316
covD	19.63	-0.381
covE	-26.18	-13.017
covF - Middle	2.66	8.598
covF - High	19.33	6.190
A^2	41.90	13.023
B*C	36.47	20.426
B*D	37.18	20.132
raw PS	102.60	37.975
linear PS	100.70	34.211

And then, more usefully, we can plot the absolute values of the standardized differences...

```
temp <- data.frame(pre.szd, post.szd, row.names=covnames)
tempsort <- temp[with(temp, order(abs(pre.szd))),]
high <- max(max(abs(pre.szd)), max(abs(post.szd)), 0.1)

dotchart(abs(tempsort$pre.szd), pch="", xlim=c(0, 1.05*high),
          labels=row.names(tempsort), main="Absolute Standardized Difference Plot",
          xlab="Absolute Standardized Difference (%)")
points(abs(tempsort$pre.szd), seq(1:length(tempsort$pre.szd)),
        pch=15, col="blue", cex=1.2)
points(abs(tempsort$post.szd), seq(1:length(tempsort$post.szd)),
        pch=19, col="red", cex=1.2)
abline(v=0, lty=1)
abline(v=10, lty=2, col="purple")
legend("bottomright", legend = c("Before Matching", "After Matching"),
       col=c("blue", "red"), text.col=c("blue", "red"), bty="o", pch = c(15, 19))
```

Absolute Standardized Difference Plot

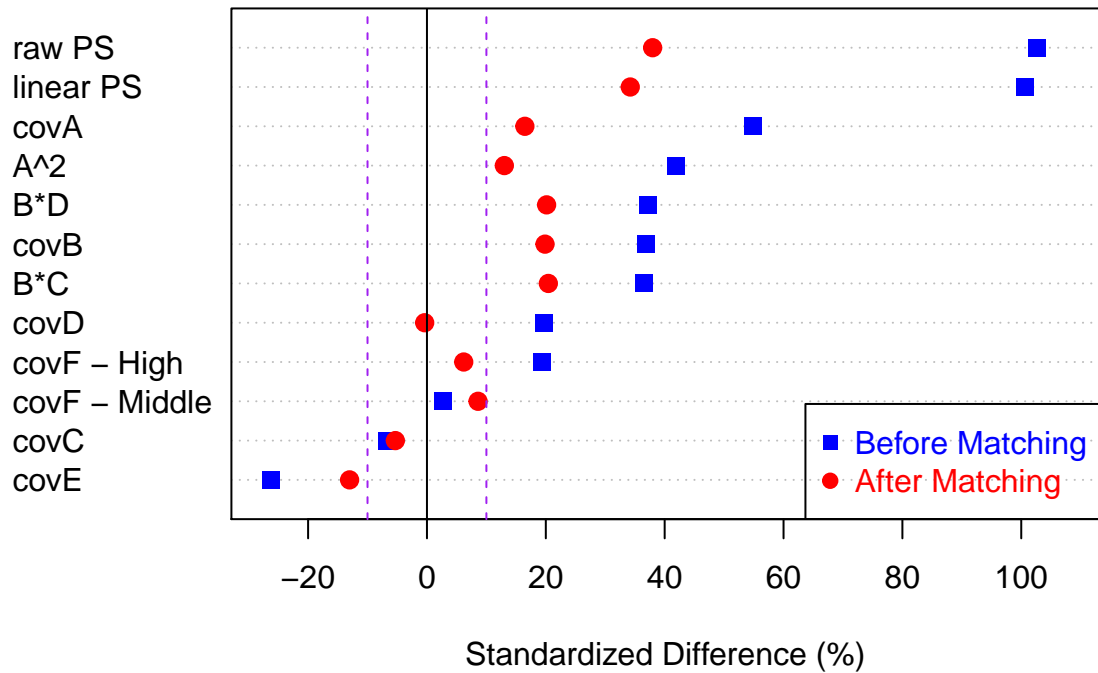


Or, we can plot the standardized differences with their signs intact...

```
temp <- data.frame(pre.szd, post.szd, row.names=covnames)
tempsort <- temp[with(temp, order(pre.szd)), ]
low <- min(min(pre.szd), min(post.szd), -0.1)
high <- max(max(pre.szd), max(post.szd), 0.1)

dotchart(tempsort$pre.szd, xlim=c(1.05*low, 1.05*high), pch="",
          labels=row.names(tempsort), main="Standardized Difference Plot",
          xlab="Standardized Difference (%)")
points(tempsort$pre.szd, seq(1:length(tempsort$pre.szd)),
       pch=15, col="blue", cex=1.2)
points(tempsort$post.szd, seq(1:length(tempsort$post.szd)),
       pch=19, col="red", cex=1.2)
abline(v=0, lty=1)
abline(v=10, lty=2, col="purple")
abline(v=-10, lty=2, col="purple")
legend("bottomright", legend = c("Before Matching", "After Matching"),
       col=c("blue", "red"), text.col=c("blue", "red"), bty="o", pch = c(15, 19))
```

Standardized Difference Plot



8.3 Using cobalt to build a “Love Plot” after Matching

```
b <- bal.tab(match1, treated ~ covA + covB + covC + covD + covE + covF +
  Asqr + BC + BD + ps + linps, data=toy, un = TRUE)
b
```

Balance Measures:

	Type	Diff.Un	Diff.Adj
covA	Contin.	0.6368	0.1648
covB	Binary	0.1780	0.1000
covC	Contin.	-0.0678	-0.0532
covD	Contin.	0.1907	-0.0038
covE	Contin.	-0.3020	-0.1302
covF_1-Low	Binary	-0.0978	-0.0714
covF_2-Middle	Binary	0.0132	0.0429
covF_3-High	Binary	0.0846	0.0286
Asqr	Contin.	0.4562	0.1302
BC	Contin.	0.3449	0.2043
BD	Contin.	0.3525	0.2013
ps	Contin.	1.0145	0.3798
linps	Contin.	1.1730	0.3421

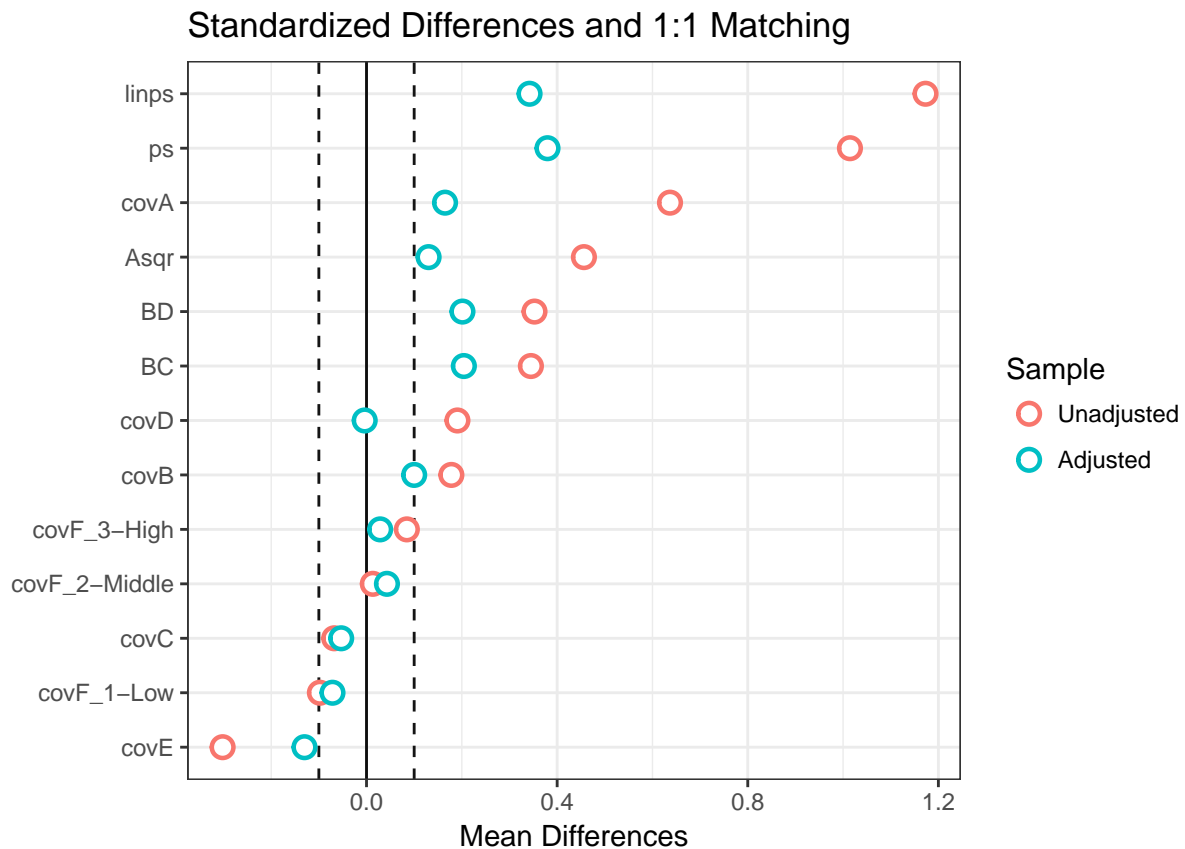
Sample sizes:

	Control	Treated
All	130	70

Matched	70	70
Unmatched	60	0

8.3.1 Building a Plot of Standardized Differences, with cobalt

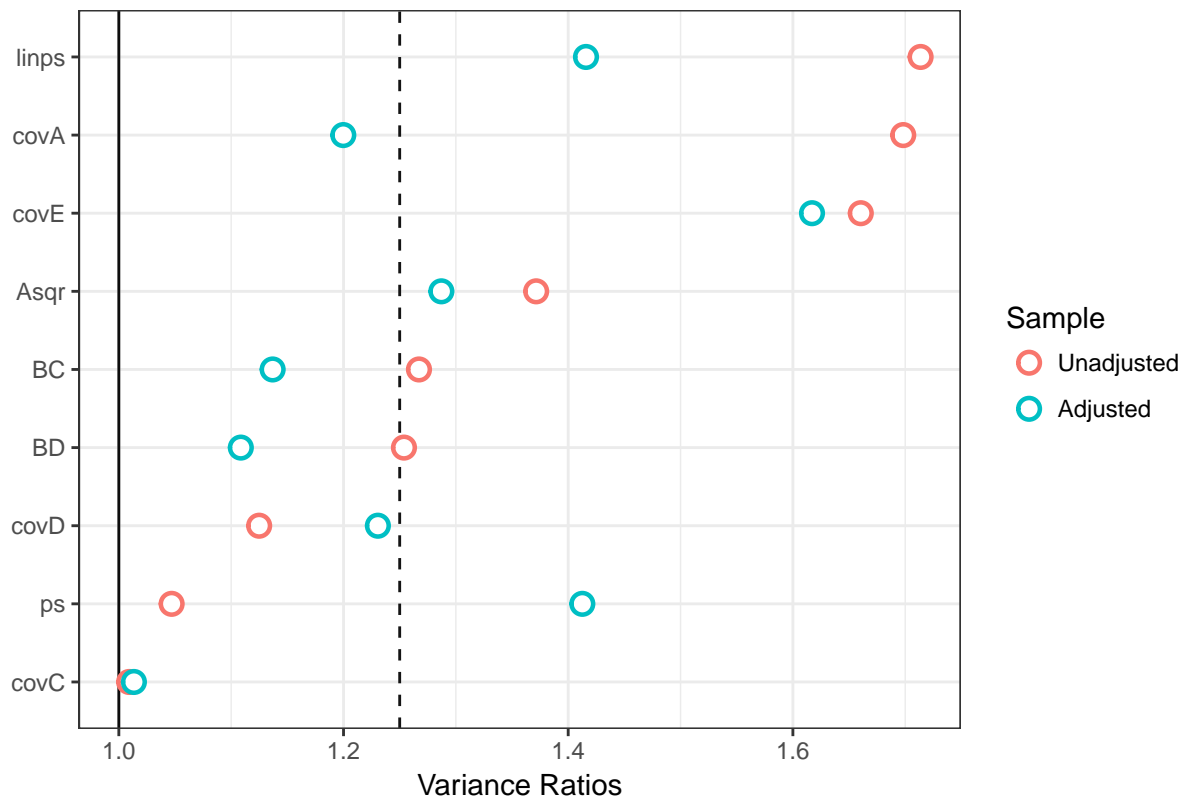
```
p <- love.plot(b, threshold = .1, size = 1.5,
               var.order = "unadjusted",
               title = "Standardized Differences and 1:1 Matching")
p + theme_bw()
```



8.3.2 Building a Plot of Variance Ratios, with cobalt

```
p <- love.plot(b, stat = "v",
               threshold = 1.25, size = 1.5,
               var.order = "unadjusted",
               title = "Variance Ratios and 1:1 Matching")
p + theme_bw()
```


Variance Ratios and 1:1 Matching



8.4 Extracting, Tabulating and Plotting Variance Ratios (without cobalt)

Next, we extract the variance ratios, build a table and then plot them.

```
pre.vratio <- NULL; post.vratio <- NULL
for(i in 1:length(covnames)) {
  pre.vratio[i] <- mb1$BeforeMatching[[i]]$var.ratio
  post.vratio[i] <- mb1$AfterMatching[[i]]$var.ratio
}

## Table of Variance Ratios
temp <- data.frame(pre.vratio, post.vratio, row.names=covnames)
print(temp, digits=2)
```

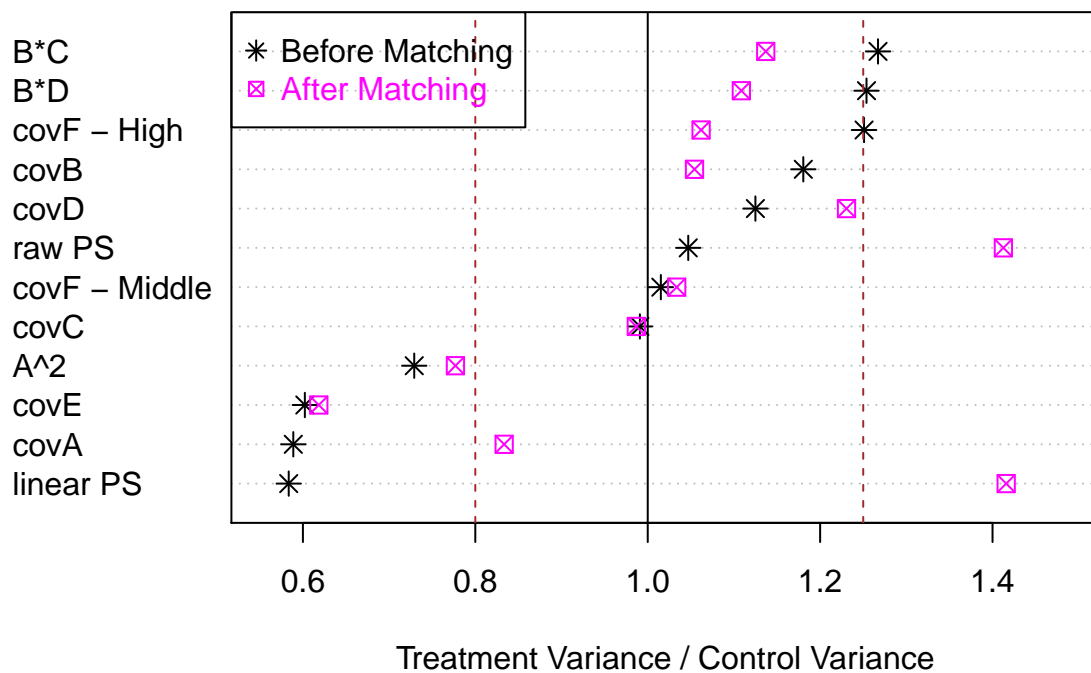
	pre.vratio	post.vratio
covA	0.59	0.83
covB	1.18	1.05
covC	0.99	0.99
covD	1.12	1.23
covE	0.60	0.62
covF - Middle	1.02	1.03
covF - High	1.25	1.06
A ²	0.73	0.78
B*C	1.27	1.14
B*D	1.25	1.11
raw PS	1.05	1.41

linear PS 0.58 1.42

```
## Variance Ratio Plot
temp <- data.frame(pre.vratio, post.vratio, row.names=covnames)
tempsort <- temp[with(temp, order(pre.vratio)), ]
low <- min(min(pre.vratio), min(post.vratio))
high <- max(max(pre.vratio), max(post.vratio))

dotchart(tempsort$pre.vratio, xlim=c(0.95*low, 1.05*high),
          pch="", labels=row.names(tempsort), main="Plot of Variance Ratios",
          xlab="Treatment Variance / Control Variance")
points(tempsort$pre.vratio, seq(1:length(tempsort$pre.vratio)),
       pch=8, col="black", cex=1.2)
points(tempsort$post.vratio, seq(1:length(tempsort$post.vratio)),
       pch=7, col="magenta", cex=1.2)
abline(v=1, lty=1)
abline(v=4/5, lty=2, col="brown")
abline(v=5/4, lty=2, col="brown")
legend("topleft", legend = c("Before Matching", "After Matching"),
       col=c("black", "magenta"), text.col=c("black", "magenta"),
       bty="n", pch = c(8, 7))
```

Plot of Variance Ratios



8.5 Creating a New Data Frame, Containing the Matched Sample (without cobalt)

Now, we build a new matched sample data frame in order to do some of the analyses to come. This will contain only the 140 matched subjects (70 treated and 70 control).

```
matches <- factor(rep(match1$index.treated, 2))
toy.matchedsample <- cbind(matches, toy[c(match1$index.control, match1$index.treated),])
```

Some sanity checks:

```
table(toy.matchedsample$treated_f)
```

```
Treated Control
      70      70
```

```
head(toy.matchedsample)
```

	matches	subject	treated	covA	covB	covC	covD	covE	covF	out1.cost
1	131	44	0	0.76	0	8.5	10.7	11	3-High	32
2	132	100	0	0.60	1	4.6	7.3	11	3-High	38
3	133	72	0	0.68	0	8.1	10.4	10	1-Low	24
4	134	84	0	0.61	1	9.5	7.4	13	3-High	35
5	135	108	0	0.22	0	11.1	10.1	7	2-Middle	49
6	136	18	0	0.58	1	10.1	11.5	9	1-Low	49

	out2.event	out3.time	treated_f	covB_f	out2_f	out2	covF.Low	covF.Middle
1	Yes	94	Control	No B	Event	1	0	0
2	Yes	98	Control	Has B	Event	1	0	0
3	No	95	Control	No B	No Event	0	1	0
4	No	122	Control	Has B	No Event	0	0	0
5	Yes	104	Control	No B	Event	1	0	1
6	Yes	101	Control	Has B	Event	1	1	0

	covF.High	Asqr	BC	BD	ps	linps	exposure
1	1	0.5776	0.0	0.0	0.5414953	0.1663638	0
2	1	0.3600	4.6	7.3	0.5587964	0.2362787	0
3	0	0.4624	0.0	0.0	0.4074310	-0.3745958	0
4	1	0.3721	9.5	7.4	0.4739821	-0.1041658	0
5	0	0.0484	0.0	0.0	0.1605378	-1.6542318	0
6	0	0.3364	10.1	11.5	0.6940724	0.8192280	0

8.6 Rubin's Rules to Check Balance After Matching

8.6.1 Rubin's Rule 1

Rubin's Rule 1 states that the absolute value of the standardized difference of the linear propensity score, comparing the treated group to the control group, should be close to 0, ideally below 10%, and in any case less than 50%. If so, we may move on to Rule 2.

Recall that our result without propensity matching (or any other adjustment) was

```
rubin1.unadj
```

```
[1] 88.11531
```

To run this for our matched sample, we use:

```
rubin1.match <- with(toy.matchedsample,
  abs(100*(mean(linps[treated==1])-mean(linps[treated==0]))/sd(linps)))
rubin1.match
```

```
[1] 36.54225
```

Here, we've at least got this value down below 50%, so we would pass Rule 1, although perhaps a different propensity score adjustment (perhaps by weighting or subclassification, or using a different matching approach) might improve this result by getting it closer to 0.

8.6.2 Rubin's Rule 2

Rubin's Rule 2 states that the ratio of the variance of the linear propensity score in the treated group to the variance of the linear propensity score in the control group should be close to 1, ideally between 4/5 and 5/4, but certainly not very close to or exceeding 1/2 and 2. If so, we may move on to Rule 3.

Recall that our result without propensity matching (or any other adjustment) was

```
rubin2.unadj
```

```
[1] 0.5835438
```

To run this for our matched sample, we use:

```
rubin2.match <- with(toy.matchedsample, var(linps[treated==1])/var(linps[treated==0]))
rubin2.match
```

```
[1] 1.415833
```

This is moderately promising - a substantial improvement over our unadjusted result, though not yet within our desired range of 4/5 to 5/4, but clearly within 1/2 to 2. We pass Rule 2, as well.

8.6.3 Rubin's Rule 3

For Rubin's Rule 3, we begin by calculating regression residuals for each covariate of interest (usually, each of those included in the propensity model) regressed on a single predictor - the linear propensity score. We then look to see if the ratio of the variance of the residuals of this model for the treatment group divided by the variance of the residuals of this model for the control group is close to 1. Again, ideally this will fall between 4/5 and 5/4 for each covariate, but certainly between 1/2 and 2. If so, then the use of regression models seems well justified.

Recall that our result without propensity matching (or any other adjustment) was

```
rubin3.unadj
```

```
# A tibble: 10 x 2
  name      resid.var.ratio
  <chr>          <dbl>
1 covA          0.591
2 covB          1.07
3 covC          1.04
4 covD          1.16
5 covE          0.558
6 covF.Middle   1.01
7 covF.High     1.28
8 Asqr          0.684
9 BC            1.14
10 BD           1.11
```

After propensity matching, we use this code to assess Rubin's 3rd Rule in our matched sample.

```
cov.sub <- dplyr::select(toy.matchedsample,
                        covA, covB, covC, covD, covE,
                        covF.Middle, covF.High, Asqr, BC, BD)

toy.matchedsample$exposure <- toy.matchedsample$treated

rubin3.matched <- rubin3(data = toy.matchedsample, covlist = cov.sub, linps = linps)

rubin3.matched
```

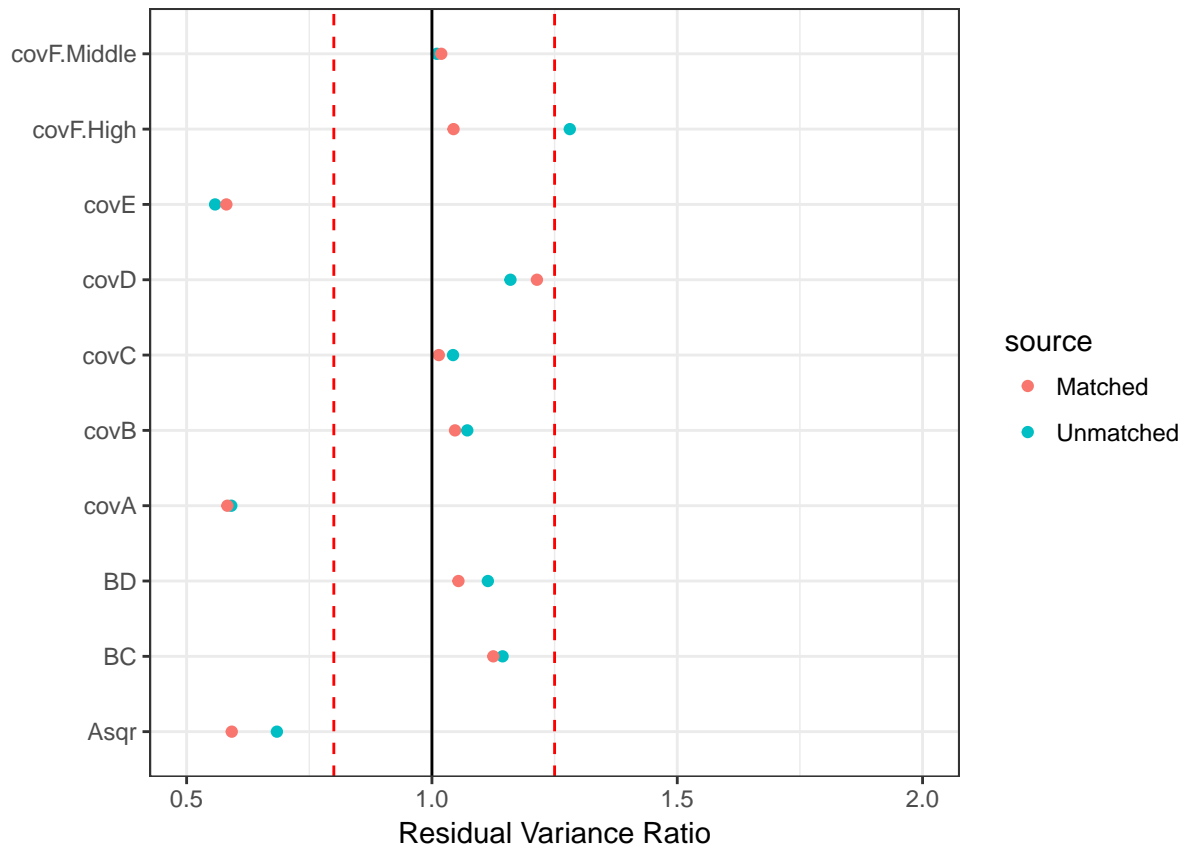
```
# A tibble: 10 x 2
  name      resid.var.ratio
  <chr>          <dbl>
1 covA          0.583
2 covB          1.05
3 covC          1.01
4 covD          1.21
5 covE          0.581
6 covF.Middle    1.02
7 covF.High      1.04
8 Asqr          0.592
9 BC            1.12
10 BD           1.05
```

It looks like the results are basically unchanged, except that `covF.High` is improved. The dotplot of these results comparing pre- to post-matching is shown below.

8.6.4 A Cleveland Dot Chart of the Rubin's Rule 3 Results Pre vs. Post-Match

```
rubin3.both <- bind_rows(rubin3.unadj, rubin3.matched)
rubin3.both$source <- c(rep("Unmatched",10), rep("Matched", 10))

ggplot(rubin3.both, aes(x = resid.var.ratio, y = name, col = source)) +
  geom_point() +
  theme_bw() +
  xlim(0.5, 2.0) +
  geom_vline(aes(xintercept = 1)) +
  geom_vline(aes(xintercept = 4/5), linetype = "dashed", col = "red") +
  geom_vline(aes(xintercept = 5/4), linetype = "dashed", col = "red") +
  labs(x = "Residual Variance Ratio", y = "")
```



Not a whole lot of improvement to report.

9 Task 5. After matching, estimate the causal effect of treatment on ...

9.1 Outcome 1 (a continuous outcome)

9.1.1 Approach 1. Automated Approach from the Matching Library - ATT Estimate

First, we'll look at the essentially automatic answer which can be obtained when using the `Matching` package and inserting an outcome `Y`. For a continuous outcome, this is often a reasonable approach.

```
X <- toy$linps ## matching on the linear propensity score
Tr <- as.logical(toy$treated)
Y <- toy$out1.cost
match1 <- Match(Y=Y, Tr=Tr, X=X, M = 1, replace=FALSE, ties=FALSE)
summary(match1)
```

```
Estimate... 15.557
SE..... 2.0397
T-stat..... 7.6273
p.val..... 2.3981e-14
```

```
Original number of observations..... 200
```

```
Original number of treated obs..... 70
Matched number of observations..... 70
Matched number of observations (unweighted). 70
```

We can obtain an approximate 95% confidence interval by adding and subtracting 1.96 times (or just double) the standard error (SE) to the point estimate, which is 15.6 here. Here, using the 1.96 figure, that would yields an approximate 95% CI of (11.6, 19.6).

9.1.2 Approach 2. Automated Approach from the Matching Library - ATE Estimate

```
match1.ATE <- Match(Y=Y, Tr=Tr, X=X, M = 1, replace=FALSE, ties=FALSE, estimand="ATE")
```

```
Warning in Match(Y = Y, Tr = Tr, X = X, M = 1, replace = FALSE, ties =
FALSE, : replace==FALSE, but there are more (weighted) control obs than
treated obs. Some control obs will not be matched. You may want to estimate
ATT instead.
```

```
summary(match1.ATE)
```

```
Estimate... 16.014
SE..... 1.5568
T-stat..... 10.287
p.val..... < 2.22e-16
```

```
Original number of observations..... 200
Original number of treated obs..... 70
Matched number of observations..... 140
Matched number of observations (unweighted). 140
```

```
rm(match1.ATE)
```

And our 95% CI for this ATE estimate would be $16.014 \pm 1.96(1.5614)$, or (13.0, 19.1), but we'll stick with the ATT estimate for now.

9.1.3 ATT vs. ATE: Definitions

- Informally, the **average treatment effect on the treated (ATT)** estimate describes the difference in potential outcomes (between treated and untreated subjects) summarized across the population of people who actually received the treatment.
 - In our initial match, we identified a unique and nicely matched control patient for each of the 70 people in the treated group. We have a 1:1 match on the treated, and thus can describe subjects across that set of treated patients reasonably well.
- On the other hand the **average treatment effect (ATE)** refers to the difference in potential outcomes summarized across the entire population, including those who did not receive the treatment.
 - In our ATE match, we have less success, in part because if we match to the treated patients in a 1:1 way, we'll have an additional 60 unmatched control patients, about whom we can describe results only vaguely. We could consider matching up control patients to treated patients, perhaps combined with a willingness to re-use some of the treated patients to get a better estimate across the whole population.

9.1.4 Approach 3. Mirroring the Paired T test in a Regression Model

We can mirror the paired t test result in a regression model that treats the match identifier as a fixed factor in a linear model, as follows. This takes the pairing into account, but treating pairing as a fixed, rather than random, factor, isn't really satisfactory as a solution, although it does match the paired t test.

```
adj.m.out1 <- lm(out1.cost ~ treated + factor(matches), data=toy.matchedsample)
coef(adj.m.out1)["treated"] # point estimate for treated effect
```

```
treated
15.55714
```

```
confint(adj.m.out1)["treated",1] # lower limit of 95% CI
```

```
[1] 11.45873
```

```
confint(adj.m.out1)["treated",2] # lower limit of 95% CI
```

```
[1] 19.65556
```

So, this regression approach produces an estimate that is exactly the same as the paired t test², but this isn't something I'm completely comfortable with.

9.1.5 Approach 4. A Mixed Model to account for 1:1 Matching

What I think of as a more appropriate result comes from a mixed model where the matches are treated as a random factor, but the treatment group is treated as a fixed factor. This is developed like this, using the lme4 package. Note that we have to create a factor variable to represent the matches, since that's the only thing that lme4 understands.

```
toy.matchedsample$matches.f <- as.factor(toy.matchedsample$matches)
## Need to use matches as a factor in R here

mixedmodel.out1 <- lmer(out1.cost ~ treated + (1 | matches.f), data=toy.matchedsample)
summary(mixedmodel.out1); confint(mixedmodel.out1)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: out1.cost ~ treated + (1 | matches.f)
Data: toy.matchedsample
```

```
REML criterion at convergence: 1122.5
```

```
Scaled residuals:
```

```
      Min       1Q   Median       3Q      Max
-2.3102 -0.5639  0.1046  0.6678  1.9395
```

```
Random effects:
```

```
Groups      Name      Variance Std.Dev.
matches.f (Intercept)  45.25     6.727
Residual                147.72    12.154
Number of obs: 140, groups: matches.f, 70
```

```
Fixed effects:
```

```
              Estimate Std. Error t value
(Intercept)   45.057      1.660  27.137
```

²I'll leave checking that this is true as an exercise for the curious.


```
treated      15.557      2.054      7.573
```

Correlation of Fixed Effects:

(Intr)

```
treated -0.619
```

Computing profile confidence intervals ...

```

          2.5 %    97.5 %
.sig01    0.503528  9.895039
.sigma    10.313755 14.378709
(Intercept) 41.802672 48.311613
treated    11.503977 19.610308
```

9.1.6 Practically, does any of this matter in this example?

Not in this example, no, as long as you stick to the ATT approaches.

	Approach	Effect Estimate	Standard Error	95% CI
	“Automated” ATT via <code>Match</code>	15.56	2.04	11.56, 19.55
	Linear Model (pairs as fixed factor)	15.56	2.05	11.46, 19.66
	Mixed Model (pairs as random factor)	15.57	2.05	11.50, 19.61

9.2 Outcome 2 (a binary outcome)

9.2.1 Approach 1. Automated Approach from the Matching Library (ATT)

First, we’ll look at the essentially automatic answer which can be obtained when using the `Matching` package and inserting an outcome `Y`. For a binary outcome, this is often a reasonable approach, especially if you don’t wish to adjust for any other covariate, and the result will be expressed as a risk difference, rather than as a relative risk or odds ratio. Note that I have used the 0-1 version of Outcome 2, rather than a factor version. The estimate produced is the difference in risk associated with `out2 = 1` (Treated subjects) minus `out2 = 0` (Controls.)

```

X <- toy$linps ## matching on the linear propensity score
Tr <- as.logical(toy$treated)
Y <- toy$out2
match1 <- Match(Y=Y, Tr=Tr, X=X, M = 1, replace=FALSE, ties=FALSE)
summary(match1)
```

```

Estimate... 0.1
SE..... 0.075997
T-stat..... 1.3158
p.val..... 0.18823
```

```

Original number of observations..... 200
Original number of treated obs..... 70
Matched number of observations..... 70
Matched number of observations (unweighted). 70
```

As in the continuous case, we obtain an approximate 95% confidence interval by adding and subtracting 1.96 times (or just double) the standard error (SE) to the point estimate, which is 0.1 (i.e. 10 percentage points)

here.

- Here, using the 1.96 figure, that would yields an approximate 95% CI of (-0.05, 0.25).
- Again, the estimated average causal effect is not statistically significant here, since 0 is contained in this confidence interval.

9.2.2 Approach 2. Using the matched sample to perform a conditional logistic regression

Since we have the matched sample available, we can simply perform a conditional logistic regression to estimate the treatment effect in terms of a log odds ratio (or, by exponentiating, an odds ratio.) Again, I use the 0/1 version of both the outcome and treatment indicator. The key modeling function `clogit` is part of the `survival` package.

```
adj.m.out2 <- clogit(out2 ~ treated + strata(matches), data=toy.matchedsample)
summary(adj.m.out2)
```

Call:

```
coxph(formula = Surv(rep(1, 140L), out2) ~ treated + strata(matches),
      data = toy.matchedsample, method = "exact")
```

```
n= 140, number of events= 71
```

	coef	exp(coef)	se(coef)	z	Pr(> z)
treated	0.4925	1.6364	0.3827	1.287	0.198

	exp(coef)	exp(-coef)	lower .95	upper .95
treated	1.636	0.6111	0.7729	3.465

```
Rsquare= 0.012 (max possible= 0.25 )
```

```
Likelihood ratio test= 1.71 on 1 df, p=0.1914
```

```
Wald test = 1.66 on 1 df, p=0.1982
```

```
Score (logrank) test = 1.69 on 1 df, p=0.1936
```

The odds ratio in the `exp(coef)` section above is the average causal effect estimate - it describes the odds of having an event (`out2`) occur associated with being a treated subject, as compared to the odds of the event when a control subject.

- Again, the result, though nominally fairly far from 1, is nowhere near statistically significant, according to our 95% confidence interval.
- Our estimate is 1.64, with 95% CI (0.77, 3.47).
- I'll use this conditional logistic regression approach to summarize the findings with regard to an odds ratio in my summary of matching results to come.

9.3 Outcome 3 (a time-to-event outcome)

9.3.1 Approach 1. Automated Approach from the Matching Library

Again, we'll start by thinking about the essentially automatic answer which can be obtained when using the `Match` function. The problem here is that this approach doesn't take into account the right censoring at all, and assumes that all of the specified times in Outcome 3 are observed. This causes the result (or the ATE version) to be non-sensical, given what we know about the data. So I don't recommend you use this approach when dealing with a time-to-event outcome.

And as a result, I won't even show it here.

9.3.2 Approach 2. A stratified Cox proportional hazards model

Since we have the matched sample, we can use a stratified Cox proportional hazards model to compare the treatment groups on our time-to-event outcome, while accounting for the matched pairs. The main results will be a relative hazard rate estimate, with 95% CI. Again, I use the 0/1 numeric version of the event indicator (`out2`), and of the treatment indicator (`treated`) here.

```
adj.m.out3 <- coxph(Surv(out3.time, out2) ~ treated + strata(matches), data=toy.matchedsample)
summary(adj.m.out3)
```

Call:

```
coxph(formula = Surv(out3.time, out2) ~ treated + strata(matches),
      data = toy.matchedsample)
```

```
n= 140, number of events= 71
```

	coef	exp(coef)	se(coef)	z	Pr(> z)
treated	-0.2877	0.7500	0.3118	-0.923	0.356

	exp(coef)	exp(-coef)	lower .95	upper .95
treated	0.75	1.333	0.4071	1.382

```
Concordance= 0.571 (se = 0.5 )
```

```
Rsquare= 0.006 (max possible= 0.34 )
```

```
Likelihood ratio test= 0.86 on 1 df, p=0.3537
```

```
Wald test = 0.85 on 1 df, p=0.3562
```

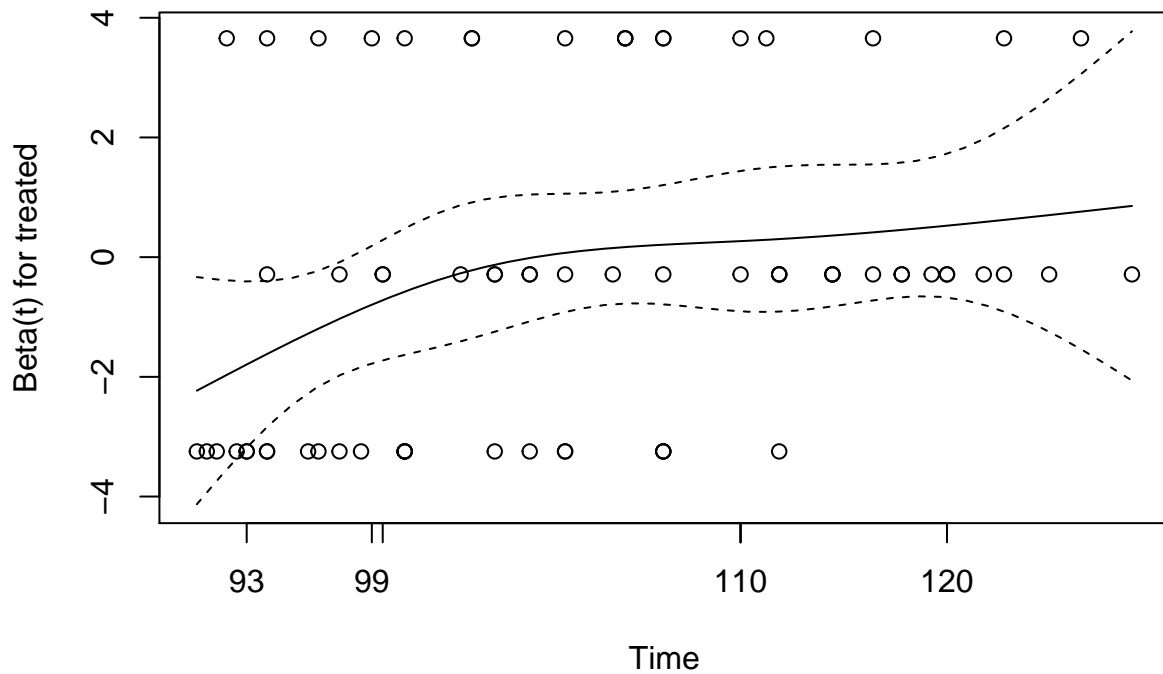
```
Score (logrank) test = 0.86 on 1 df, p=0.3545
```

The relative hazard rate (from the `exp(coef)` section of the output) is estimated to be 0.75, with 95% CI (0.41, 1.39) - again not statistically significant. Checking the proportional hazards assumption suggests a possible issue, as well...

```
cox.zph(adj.m.out3) # Quick check for proportional hazards assumption
```

	rho	chisq	p
treated	0.288	5.9	0.0152

```
plot(cox.zph(adj.m.out3), var="treated")
```



9.4 Results So Far (After Propensity Matching)

So, here's our summary again, now incorporating both our unadjusted results and the results after matching. Automated results and my favorite of our various non-automated approaches are shown. Note that I've left out the "automated" approach for a time-to-event outcome entirely, so as to discourage you from using it as presented above.

Est. Treatment Effect (95% CI)	Outcome 1 (Cost diff.)	Outcome 2 (Risk diff.)	Outcome 2 (Odds Ratio)	Outcome 3 (Relative Hazard Rate)
No covariate adjustment	15.7	+0.11	1.56	0.86
(unadjusted)	(12.0, 19.3)	(-0.03, +0.25)	(0.87, 2.82)	(0.57, 1.29)
After 1:1 PS Match	15.6	+0.11	N/A	N/A
(Match: Automated)	(11.6, 19.6)	(-0.05, +0.25)	N/A	N/A
After 1:1 PS Match	15.6	N/A	1.64	0.75
("Regression" Models)	(11.5, 19.6)	N/A	(0.77, 3.47)	(0.41, 1.38)

10 Task 6. Subclassify by PS quintile, then display post-subclassification balance

First, we divide the data by the propensity score into 5 strata of equal size using the `cut2` function from the `Hmisc` package. Then we create a `quintile` variable which specifies 1 = lowest propensity scores to 5 = highest.

```
toy$stratum <- cut2(toy$ps, g=5)

by(toy$ps, toy$stratum, summary) # sanity check

toy$stratum: [0.00777,0.147)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.007769 0.045272 0.068558 0.072741 0.097041 0.144572
-----
toy$stratum: [0.14670,0.260)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.1467 0.1736 0.2241 0.2089 0.2367 0.2597
-----
toy$stratum: [0.25985,0.415)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.2598 0.2892 0.3243 0.3298 0.3663 0.4081
-----
toy$stratum: [0.41456,0.543)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.4146 0.4694 0.4898 0.4872 0.5136 0.5415
-----
toy$stratum: [0.54295,0.870]
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.5430 0.5931 0.6485 0.6514 0.6865 0.8699

toy$quintile <- factor(toy$stratum, labels=1:5)

table(toy$stratum, toy$quintile) ## sanity check
```

```

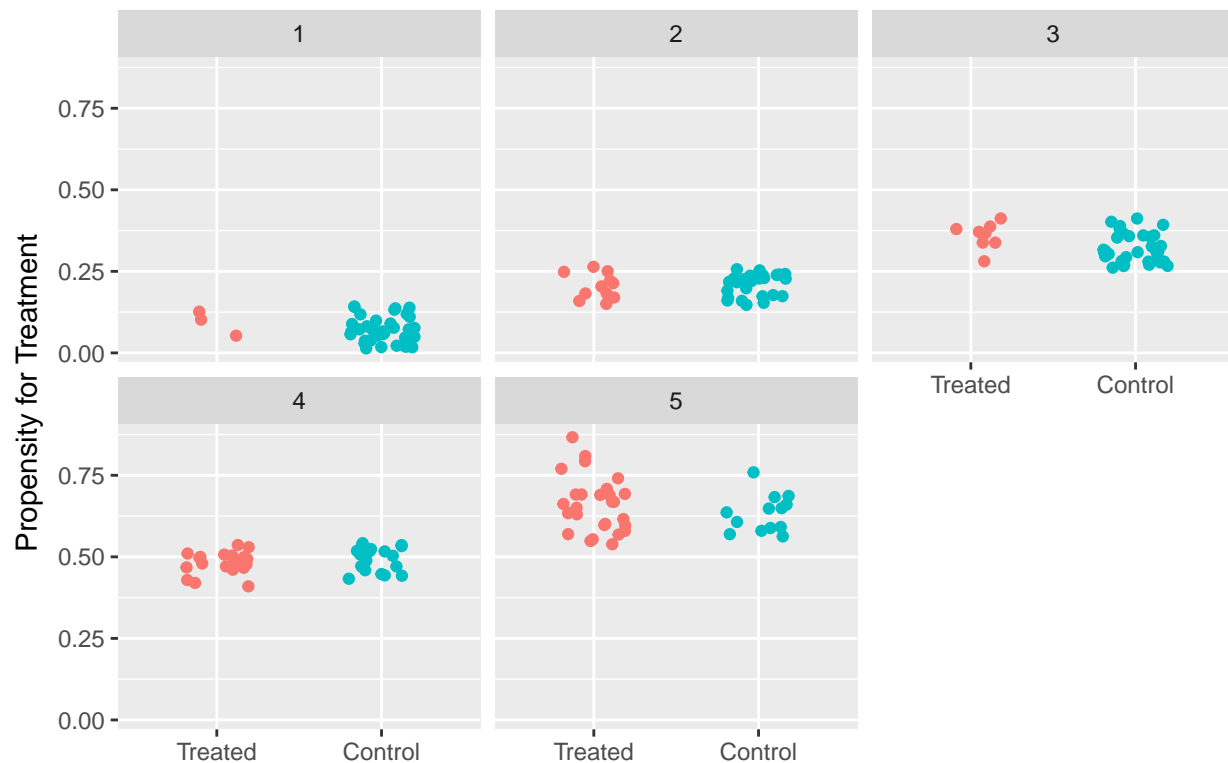
           1  2  3  4  5
[0.00777,0.147) 40  0  0  0  0
[0.14670,0.260)  0 40  0  0  0
[0.25985,0.415)  0  0 40  0  0
[0.41456,0.543)  0  0  0 40  0
[0.54295,0.870]  0  0  0  0 40
```

10.1 Check Balance and Propensity Score Overlap in Each Quintile

We want to check the balance and propensity score overlap for each quintile. I'll start with a set of faceted, jittered plots to look at overlap.

```
ggplot(toy, aes(x = treated_f, y = round(ps,2), group = quintile, color = treated_f)) +
  geom_jitter(width = 0.2) +
  guides(color = FALSE) +
  facet_wrap(~ quintile) +
  labs(x = "", y = "Propensity for Treatment",
       title = "Quintile Subclassification in the Toy Example")
```

Quintile Subclassification in the Toy Example



It can be helpful to know how many observations (by exposure group) are in each quintile.

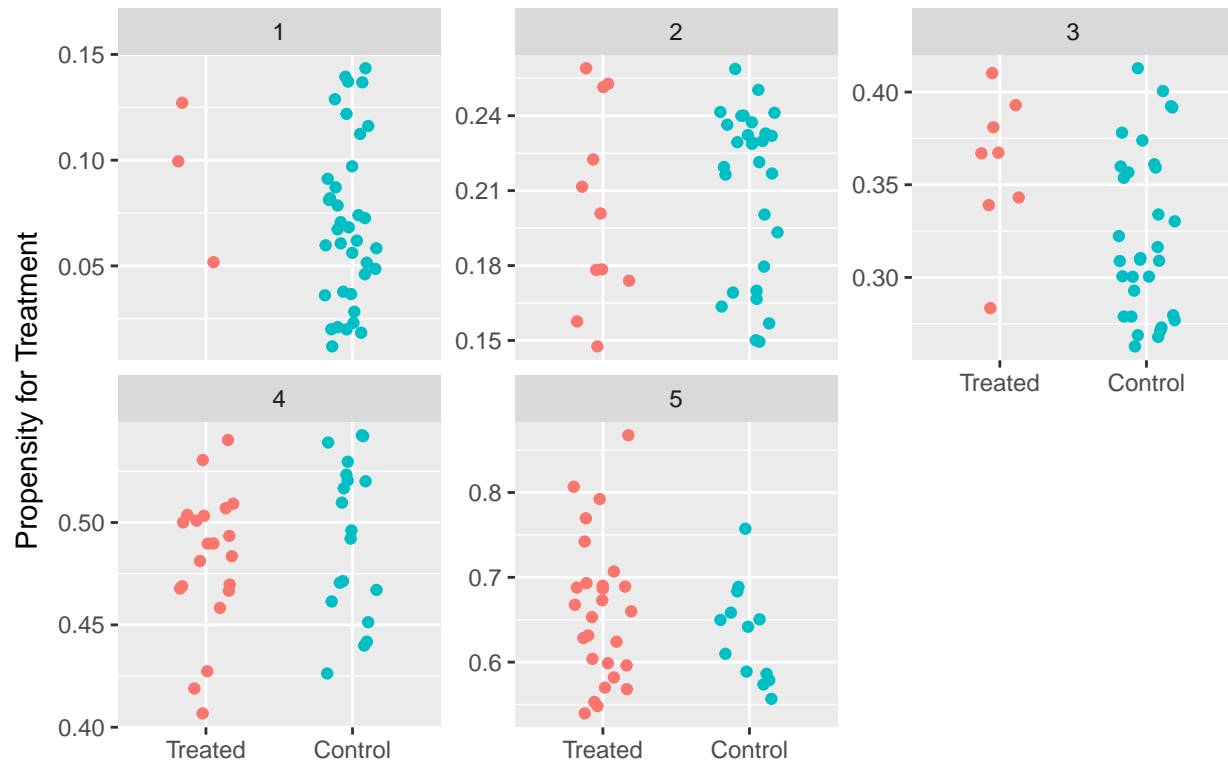
```
addmargins(table(toy$quintile, toy$treated_f))
```

	Treated	Control	Sum
1	3	37	40
2	11	29	40
3	8	32	40
4	21	19	40
5	27	13	40
Sum	70	130	200

The overlap may show a little better in the plot if you free up the y axes...

```
ggplot(toy, aes(x = treated_f, y = round(ps,2), group = quintile, color = treated_f)) +
  geom_jitter(width = 0.2) +
  guides(color = FALSE) +
  facet_wrap(~ quintile, scales = "free_y") +
  labs(x = "", y = "Propensity for Treatment",
       title = "Quintile Subclassification in the Toy Example")
```

Quintile Subclassification in the Toy Example



10.2 Creating a Standardized Difference Calculation Function

We'll need to be able to calculate standardized differences in this situation so I've created a simple `szd` function to do this - using the average denominator method.

```
szd <- function(covlist, g) {
  covlist2 <- as.matrix(covlist)
  g <- as.factor(g)
  res <- NA
  for(i in 1:ncol(covlist2)) {
    cov <- as.numeric(covlist2[,i])
    num <- 100*diff(tapply(cov, g, mean, na.rm=TRUE))
    den <- sqrt(mean(tapply(cov, g, var, na.rm=TRUE)))
    res[i] <- round(num/den,2)
  }
  names(res) <- names(covlist)
  res
}
```

10.3 Creating the Five Subsamples, by PS Quintile

Next, we split the complete sample into the five quintiles.

```
## Divide the sample into the five quintiles
quin1 <- filter(toy, quintile==1)
```

```

quin2 <- filter(toy, quintile==2)
quin3 <- filter(toy, quintile==3)
quin4 <- filter(toy, quintile==4)
quin5 <- filter(toy, quintile==5)

```

10.4 Standardized Differences in Each Quintile, and Overall

Now, we'll calculate the standardized differences within each quintile, as well as overall.

```

covs <- c("covA", "covB", "covC", "covD", "covE", "covF.Middle", "covF.High", "Asqr", "BC", "BD", "ps", "ps2")
d.q1 <- szd(quin1[covs], quin1$treated)
d.q2 <- szd(quin2[covs], quin2$treated)
d.q3 <- szd(quin3[covs], quin3$treated)
d.q4 <- szd(quin4[covs], quin4$treated)
d.q5 <- szd(quin5[covs], quin5$treated)
d.all <- szd(toy[covs], toy$treated)

toy.szd <- data_frame(covs, Overall = d.all, Q1 = d.q1, Q2 = d.q2, Q3 = d.q3, Q4 = d.q4, Q5 = d.q5)
toy.szd <- gather(toy.szd, "quint", "sz.diff", 2:7)
toy.szd

```

```

# A tibble: 72 x 3
  covs      quint  sz.diff
  <chr>    <chr>    <dbl>
1 covA    Overall    54.8
2 covB    Overall    36.8
3 covC    Overall   - 6.76
4 covD    Overall    19.6
5 covE    Overall   -26.2
6 covF.Middle Overall     2.66
7 covF.High Overall    19.3
8 Asqr     Overall    41.9
9 BC       Overall    36.5
10 BD      Overall    37.2
# ... with 62 more rows

```

10.5 Plotting the Standardized Differences

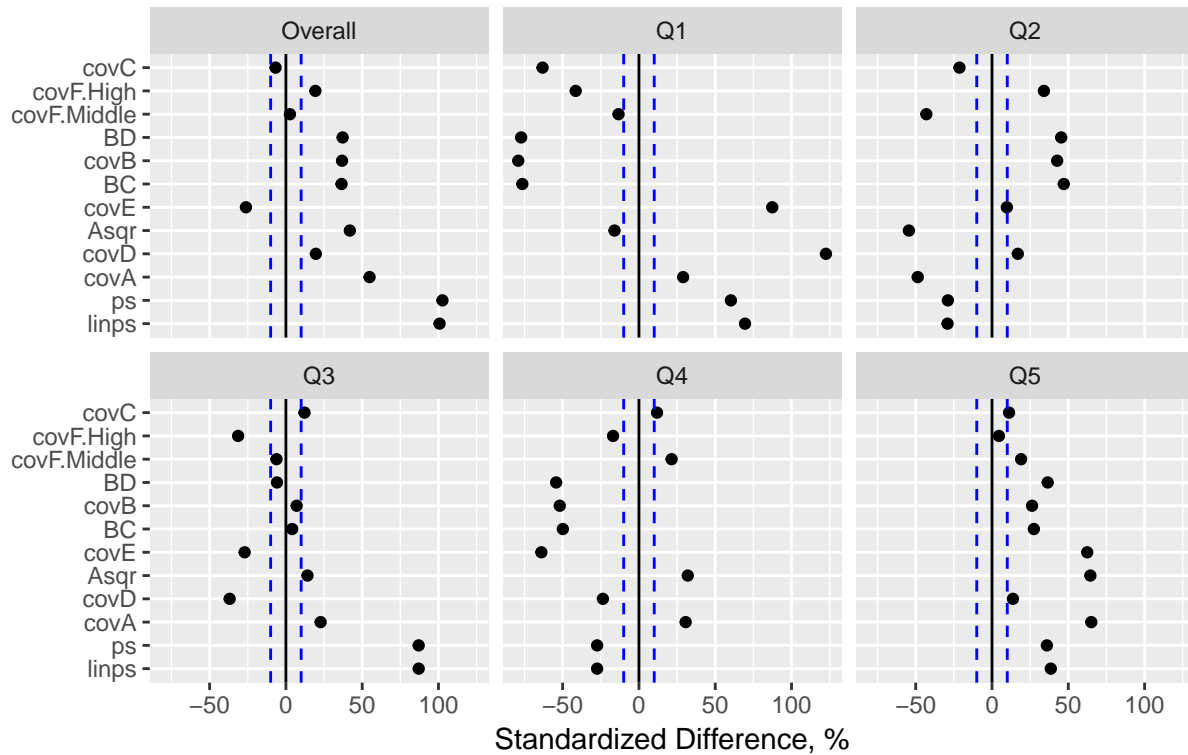
```

ggplot(toy.szd, aes(x = sz.diff, y = reorder(covs, -sz.diff), group = quint)) +
  geom_point() +
  geom_vline(xintercept = 0) +
  geom_vline(xintercept = c(-10,10), linetype = "dashed", col = "blue") +
  facet_wrap(~ quint) +
  labs(x = "Standardized Difference, %", y = "",
       title = "Comparing Standardized Differences by PS Quintile",
       subtitle = "The toy example")

```


Comparing Standardized Differences by PS Quintile

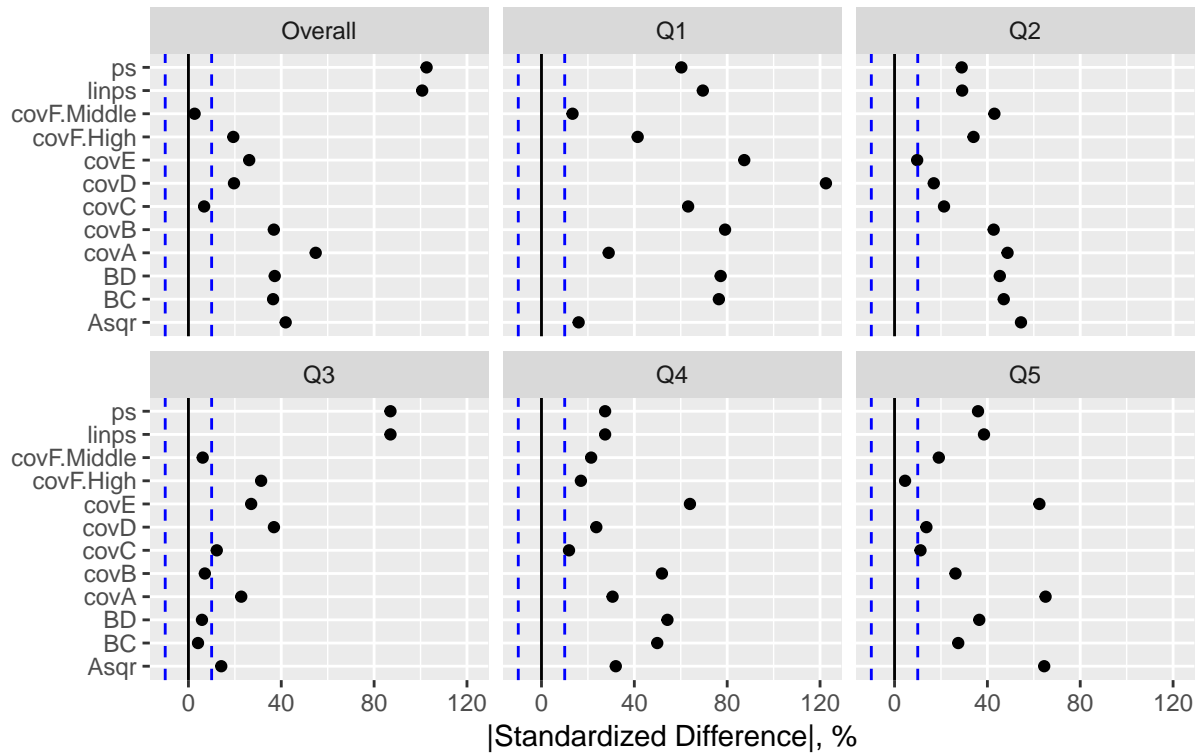
The toy example



```
ggplot(toy.sz, aes(x = abs(sz.diff), y = covs, group = quint)) +
  geom_point() +
  geom_vline(xintercept = 0) +
  geom_vline(xintercept = c(-10,10), linetype = "dashed", col = "blue") +
  facet_wrap(~ quint) +
  labs(x = "|Standardized Difference|, %", y = "",
       title = "Absolute Standardized Differences by PS Quintile",
       subtitle = "The toy example")
```

Absolute Standardized Differences by PS Quintile

The toy example



10.6 Checking Rubin's Rules Post-Subclassification

10.6.1 Rubin's Rule 1

As a reminder, prior to adjustment, Rubin's Rule 1 for the toy example was:

```
rubin1.unadj <- with(toy,
  abs(100*(mean(linps[treated==1]) -
    mean(linps[treated==0]))/sd(linps)))
rubin1.unadj
```

```
[1] 88.11531
```

After propensity score subclassification, we can obtain the same summary within each of the five quintiles...

```
rubin1.q1 <- with(quin1, abs(100*(mean(linps[treated==1]) -
  mean(linps[treated==0]))/sd(linps)))
rubin1.q2 <- with(quin2, abs(100*(mean(linps[treated==1]) -
  mean(linps[treated==0]))/sd(linps)))
rubin1.q3 <- with(quin3, abs(100*(mean(linps[treated==1]) -
  mean(linps[treated==0]))/sd(linps)))
rubin1.q4 <- with(quin4, abs(100*(mean(linps[treated==1]) -
  mean(linps[treated==0]))/sd(linps)))
rubin1.q5 <- with(quin5, abs(100*(mean(linps[treated==1]) -
  mean(linps[treated==0]))/sd(linps)))
```

```
rubin1.sub <- c(rubin1.q1, rubin1.q2, rubin1.q3, rubin1.q4, rubin1.q5)
names(rubin1.sub)=c("Q1", "Q2", "Q3", "Q4", "Q5")
```

```
rubin1.sub
```

```
      Q1      Q2      Q3      Q4      Q5
60.58052 29.90587 80.45697 27.58516 35.77173
```

Each quintile shows (in some cases, only a slightly) better result than the full data set. With a small sample size like this, and some subclasses having very few subjects in one exposure group, it was always a long shot that subclassification alone would reduce all of these values below 10%, but I had hoped to get more than 3/5 below 50%.

10.6.2 Rubin's Rule 2

As a reminder, prior to adjustment, Rubin's Rule 2 for the toy example was:

```
rubin2.unadj <- with(toy, var(linps[treated==1])/var(linps[treated==0]))
rubin2.unadj
```

```
[1] 0.5835438
```

After Subclassification, we can obtain the same summary within each of the five quintiles...

```
rubin2.q1 <- with(quin1, var(linps[treated==1])/var(linps[treated==0]))
rubin2.q2 <- with(quin2, var(linps[treated==1])/var(linps[treated==0]))
rubin2.q3 <- with(quin3, var(linps[treated==1])/var(linps[treated==0]))
rubin2.q4 <- with(quin4, var(linps[treated==1])/var(linps[treated==0]))
rubin2.q5 <- with(quin5, var(linps[treated==1])/var(linps[treated==0]))
```

```
rubin2.sub <- c(rubin2.q1, rubin2.q2, rubin2.q3, rubin2.q4, rubin2.q5)
names(rubin2.sub)=c("Q1", "Q2", "Q3", "Q4", "Q5")
```

```
rubin2.sub
```

```
      Q1      Q2      Q3      Q4      Q5
0.4780421 1.2041215 0.7905793 0.7950520 2.4852723
```

Some of these variance ratios are actually a bit further from 1 than the full data set. Again, with a small sample size like this, subclassification looks like a weak choice. At most, three of the quintiles (2-4) show OK variance ratios after propensity score subclassification.

10.6.3 Rubin's Rule 3

Prior to propensity adjustment, recall that Rubin's Rule 3 summaries were:

```
covs <- c("covA", "covB", "covC", "covD", "covE",
          "covF.Middle", "covF.High", "Asqr", "BC", "BD")
rubin3.unadj <- rubin3(data=toy, covlist=toy[covs])
```

After subclassification, then, Rubin's Rule 3 summaries within each quintile are:

```
rubin3.q1 <- rubin3(data=quin1, covlist=quin1[covs])
rubin3.q2 <- rubin3(data=quin2, covlist=quin2[covs])
rubin3.q3 <- rubin3(data=quin3, covlist=quin3[covs])
rubin3.q4 <- rubin3(data=quin4, covlist=quin4[covs])
rubin3.q5 <- rubin3(data=quin5, covlist=quin5[covs])
```

```

toy.rubin3 <- data_frame(covs, All = rubin3.unadj$resid.var.ratio,
                        Q1 = rubin3.q1$resid.var.ratio,
                        Q2 = rubin3.q2$resid.var.ratio,
                        Q3 = rubin3.q3$resid.var.ratio,
                        Q4 = rubin3.q4$resid.var.ratio,
                        Q5 = rubin3.q5$resid.var.ratio)

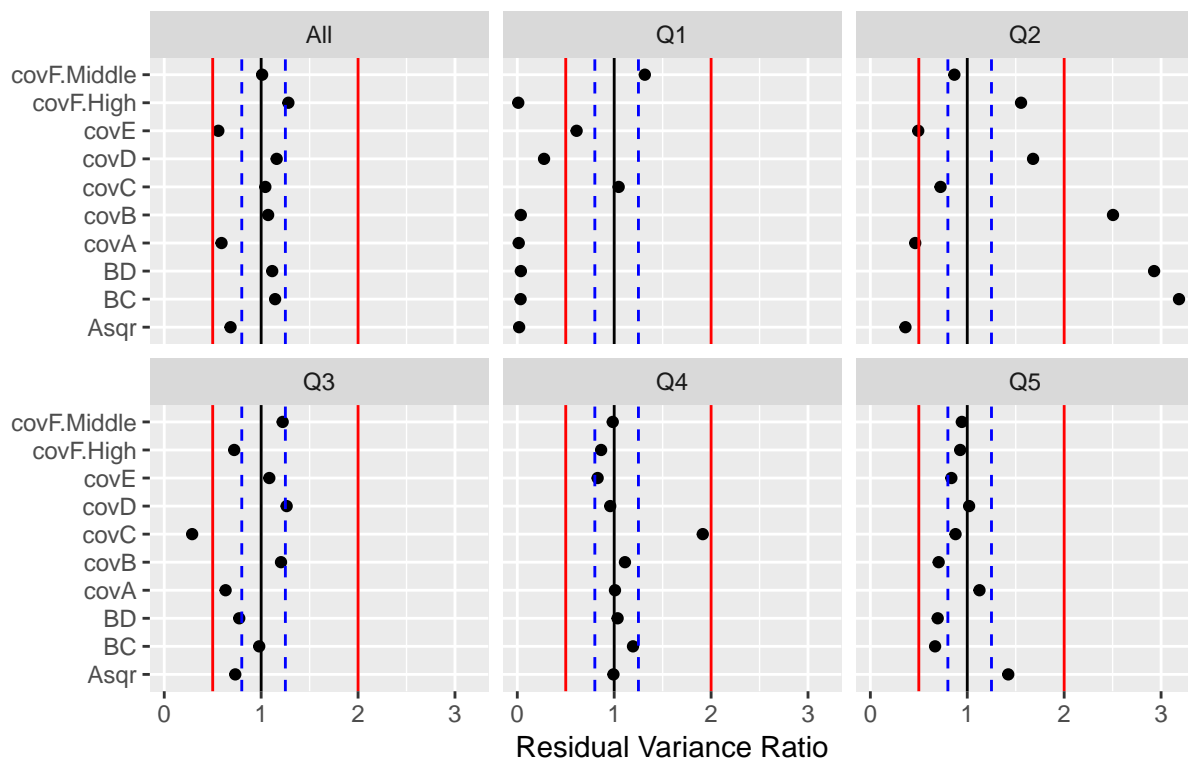
toy.rubin3 <- gather(toy.rubin3, "quint", "rubin3", 2:7)

ggplot(toy.rubin3, aes(x = rubin3, y = covs, group = quint)) +
  geom_point() +
  geom_vline(xintercept = 1) +
  geom_vline(xintercept = c(0.8, 1.25), linetype = "dashed", col = "blue") +
  geom_vline(xintercept = c(0.5, 2), col = "red") +
  facet_wrap(~ quint) +
  labs(x = "Residual Variance Ratio", y = "",
       title = "Residual Variance Ratios by PS Quintile",
       subtitle = "Rubin's Rule 3: The toy example")

```

Residual Variance Ratios by PS Quintile

Rubin's Rule 3: The toy example



Most of the residual variance ratios are in the range of (0.5, 2) but the results within quintiles vary widely. Quintiles 1 and 2 are especially problematic in this regard.

11 Task 7. After subclassifying, what is the estimated average treatment effect?

11.1 ... on Outcome 1 [a continuous outcome]

First, we'll find the estimated average causal effect (and standard error) within each quintile via linear regression.

```
quin1.out1 <- lm(out1.cost ~ treated, data=quin1)
quin2.out1 <- lm(out1.cost ~ treated, data=quin2)
quin3.out1 <- lm(out1.cost ~ treated, data=quin3)
quin4.out1 <- lm(out1.cost ~ treated, data=quin4)
quin5.out1 <- lm(out1.cost ~ treated, data=quin5)

coef(summary(quin1.out1)); coef(summary(quin2.out1)); coef(summary(quin3.out1)); coef(summary(quin4.out1)); coef(summary(quin5.out1))
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	45.45946	1.508909	30.127372	3.931690e-28
treated	-12.45946	5.509756	-2.261345	2.954424e-02

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	44.206897	1.850820	23.8850290	1.727762e-24
treated	1.065831	3.529376	0.3019884	7.643075e-01

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	44.375	2.035534	21.800171	4.412752e-23
treated	10.500	4.551593	2.306884	2.660602e-02

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	43.84211	2.895296	15.142532	1.105938e-17
treated	16.91980	3.995888	4.234303	1.400593e-04

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	48.07692	2.991778	16.069682	1.556770e-18
treated	23.44160	3.641476	6.437388	1.432788e-07

We could probably figure out a cleverer way to accomplish this using the `broom` package.

Next, we find the mean of the five quintile-specific estimated regression coefficients

```
est.st <- (coef(quin1.out1)[2] + coef(quin2.out1)[2] + coef(quin3.out1)[2] +
          coef(quin4.out1)[2] + coef(quin5.out1)[2])/5
est.st
```

```
treated
7.893553
```

To get the combined standard error estimate, we do the following:

```
se.q1 <- summary(quin1.out1)$coefficients[2,2]
se.q2 <- summary(quin2.out1)$coefficients[2,2]
se.q3 <- summary(quin3.out1)$coefficients[2,2]
se.q4 <- summary(quin4.out1)$coefficients[2,2]
se.q5 <- summary(quin5.out1)$coefficients[2,2]

se.st <- sqrt((se.q1^2 + se.q2^2 + se.q3^2 + se.q4^2 + se.q5^2)*(1/25))
se.st
```

```
[1] 1.926223
```

The resulting 95% confidence Interval for the average causal treatment effect is then:

```
temp.result1 <- c(est.st, est.st - 1.96*se.st, est.st + 1.96*se.st)
names(temp.result1) <- c("Estimate", "Low 95% CI", "High 95% CI")
temp.result1
```

```
Estimate Low 95% CI High 95% CI
7.893553 4.118156 11.668950
```

11.2 ... on Outcome 2 [a binary outcome]

First, we find the estimated average causal effect (and standard error) within each quintile via logistic regression:

```
quin1.out2 <- glm(out2 ~ treated, data=quin1, family=binomial())
quin2.out2 <- glm(out2 ~ treated, data=quin2, family=binomial())
quin3.out2 <- glm(out2 ~ treated, data=quin3, family=binomial())
quin4.out2 <- glm(out2 ~ treated, data=quin4, family=binomial())
quin5.out2 <- glm(out2 ~ treated, data=quin5, family=binomial())
```

```
coef(summary(quin1.out2)); coef(summary(quin2.out2)); coef(summary(quin3.out2)); coef(summary(quin4.out2)); coef(summary(quin5.out2))
```

```
      Estimate Std. Error    z value Pr(>|z|)
(Intercept) -0.05406722  0.3289181 -0.1643790 0.8694328
treated      -0.63907996  1.2681430 -0.5039495 0.6142969
```

```
      Estimate Std. Error    z value Pr(>|z|)
(Intercept) -0.7985077  0.4013863 -1.989374 0.04665989
treated      0.2388919  0.7442903  0.320966 0.74823614
```

```
      Estimate Std. Error    z value Pr(>|z|)
(Intercept) -0.2513144  0.3563439 -0.7052581 0.4806496
treated      0.2513144  0.7918213  0.3173878 0.7509494
```

```
      Estimate Std. Error    z value Pr(>|z|)
(Intercept)  0.3184537  0.4646602  0.6853476 0.4931246
treated      0.1670541  0.6463994  0.2584379 0.7960690
```

```
      Estimate Std. Error    z value Pr(>|z|)
(Intercept) -0.1541507  0.5563486 -0.2770757 0.7817220
treated      0.6847789  0.6843591  1.0006135 0.3170137
```

Next, we find the mean of the five quintile-specific estimated logistic regression coefficients

```
est.st <- (coef(quin1.out2)[2] + coef(quin2.out2)[2] + coef(quin3.out2)[2] +
          coef(quin4.out2)[2] + coef(quin5.out2)[2])/5
est.st ## this is the estimated log odds ratio
```

```
treated
0.1405919
```

```
## And we exponentiate this to get the overall odds ratio estimate
exp(est.st)
```

```
treated
1.150955
```

To get the combined standard error estimate across the five quintiles, we do the following:

```
se.q1 <- summary(quin1.out2)$coefficients[2,2]
se.q2 <- summary(quin2.out2)$coefficients[2,2]
se.q3 <- summary(quin3.out2)$coefficients[2,2]
se.q4 <- summary(quin4.out2)$coefficients[2,2]
se.q5 <- summary(quin5.out2)$coefficients[2,2]
se.st <- sqrt((se.q1^2 + se.q2^2 + se.q3^2 + se.q4^2 + se.q5^2)*(1/25))
se.st
```

```
[1] 0.3834222
```

```
## Of course, this standard error is also on the log odds ratio scale
```

Now, we obtain a 95% Confidence Interval for the Average Causal Effect of our treatment (as an Odds Ratio) through combination and exponentiation, as follows:

```
temp.result2 <- c(exp(est.st), exp(est.st - 1.96*se.st), exp(est.st + 1.96*se.st))
names(temp.result2) <- c("Estimate", "Low 95% CI", "High 95% CI")
temp.result2
```

```
      Estimate  Low 95% CI High 95% CI
1.1509548    0.5428536    2.4402474
```

11.3 ... on Outcome 3 [a time to event]

Subjects with `out2.event` = “Yes” are truly observed events, while those with `out2.event` == “No” are censored before an event can happen to them.

The Cox model comparing treated to control, stratifying on quintile, is...

```
adj.s.out3 <- coxph(Surv(out3.time, out2) ~ treated + strata(quintile), data=toy)
summary(adj.s.out3) ## exp(coef) gives relative hazard associated with treatment
```

Call:

```
coxph(formula = Surv(out3.time, out2) ~ treated + strata(quintile),
      data = toy)
```

```
n= 200, number of events= 97
```

```
              coef exp(coef) se(coef)      z Pr(>|z|)
treated -0.2411    0.7858   0.2436 -0.989   0.322
```

```
              exp(coef) exp(-coef) lower .95 upper .95
treated    0.7858      1.273    0.4874    1.267
```

```
Concordance= 0.538 (se = 0.056 )
Rsquare= 0.005 (max possible= 0.946 )
Likelihood ratio test= 0.98 on 1 df,  p=0.3212
Wald test               = 0.98 on 1 df,  p=0.3224
Score (logrank) test = 0.98 on 1 df,  p=0.3219
```

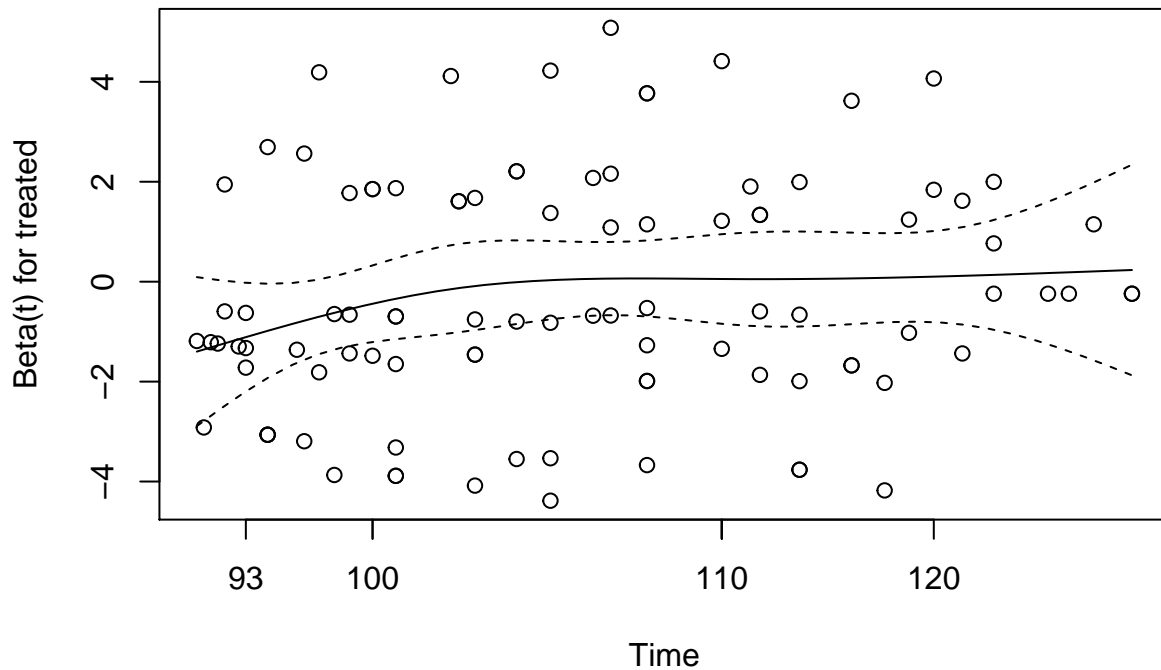
11.3.1 Checking the Proportional Hazards Assumption

The proportional hazards assumption looks fairly reasonable.

```
cox.zph(adj.s.out3) ## checking the proportional hazards assumption
```

```
      rho chisq      p
treated 0.161  2.39 0.122
```

```
plot(cox.zph(adj.s.out3), var="treated")
```



11.4 Results So Far (After Matching and Subclassification)

These subclassification results describe the average treatment effect, while the previous analyses we have completed describe the average treatment effect on the treated. This is one reason for the meaningful difference between the estimates.

Est. Treatment Effect (95% CI)	Outcome 1 (Cost diff.)	Outcome 2 (Risk diff.)	Outcome 2 (Odds Ratio)	Outcome 3 (Relative Hazard Rate)
No covariate adjustment (unadjusted)	15.7 (12.0, 19.3)	+0.11 (-0.03, +0.25)	1.56 (0.87, 2.82)	0.86 (0.57, 1.29)
After 1:1 PS Match (Match: Automated)	15.6 (11.6, 19.6)	+0.11 (-0.05, +0.25)	N/A	N/A
After 1:1 PS Match ("Regression" models)	15.6 (11.5, 19.6)	N/A	1.64 (0.77, 3.47)	0.75 (0.41, 1.38)
After PS Subclassification	7.9	N/A	1.15	0.79

Est. Treatment Effect (95% CI)	Outcome 1 (Cost diff.)	Outcome 2 (Risk diff.)	Outcome 2 (Odds Ratio)	Outcome 3 (Relative Hazard Rate)
("Regression" models, ATE)	(4.1, 11.7)	N/A	(0.54, 2.44)	(0.49, 1.27)

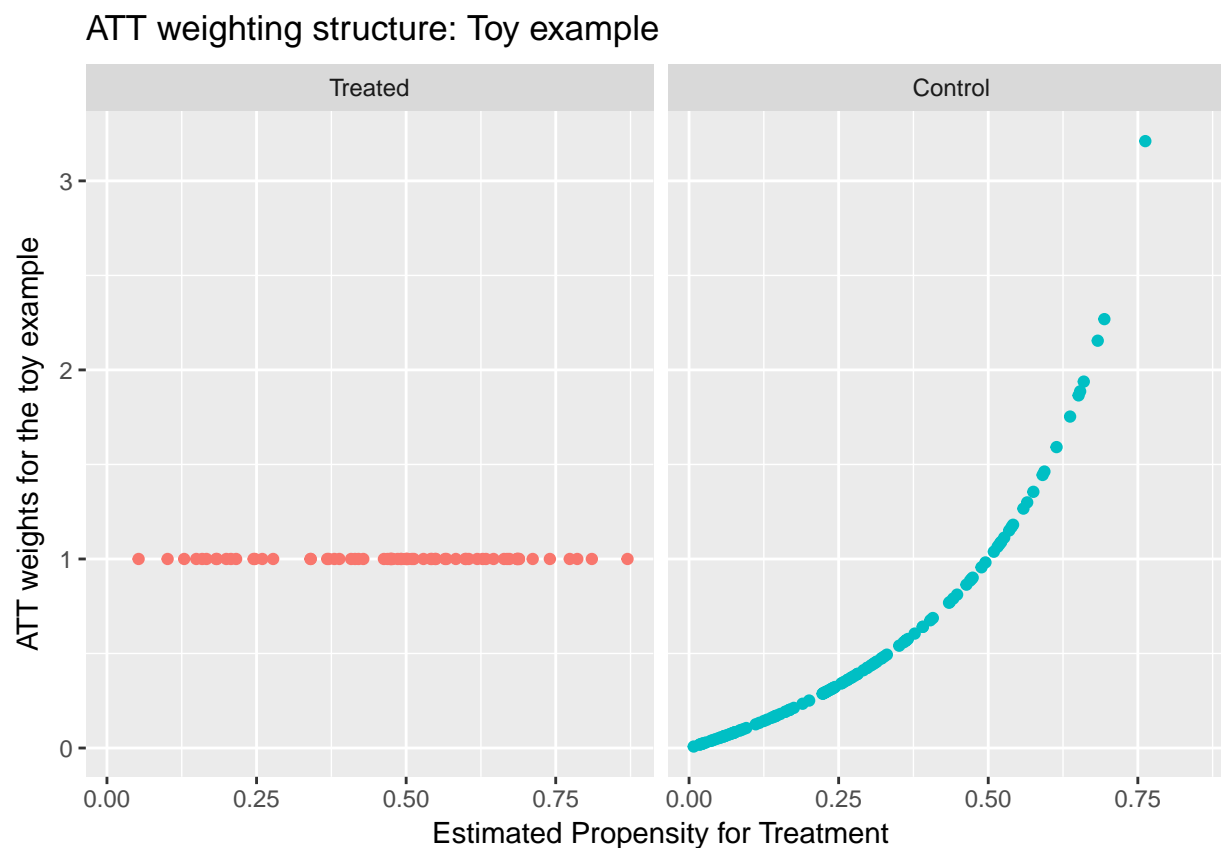
12 Task 8. Execute weighting by the inverse PS, then assess co-variate balance

12.1 ATT approach: Weight treated subjects as 1; control subjects as $ps/(1-ps)$

```
toy$wts1 <- ifelse(toy$treated==1, 1, toy$ps/(1-toy$ps))
```

Here is a plot of the resulting ATT (average treatment effect on the treated) weights:

```
ggplot(toy, aes(x = ps, y = wts1, color = treated_f)) +
  geom_point() +
  guides(color = FALSE) +
  facet_wrap(~ treated_f) +
  labs(x = "Estimated Propensity for Treatment",
       y = "ATT weights for the toy example",
       title = "ATT weighting structure: Toy example")
```



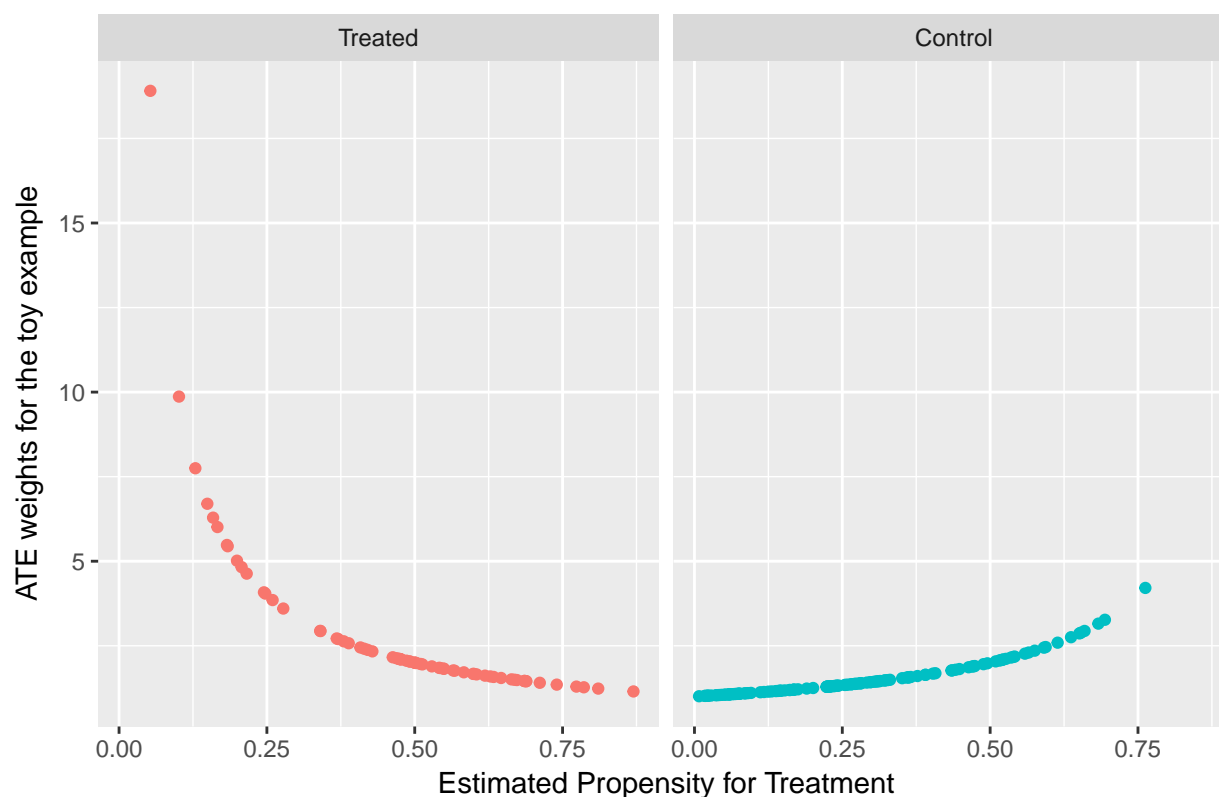
12.2 ATE Approach: Weight treated subjects by $1/ps$; Control subjects by $1/(1-PS)$

```
toy$wts2 <- ifelse(toy$treated==1, 1/toy$ps, 1/(1-toy$ps))
```

Here's a plot of the ATE (average treatment effect) weights...

```
ggplot(toy, aes(x = ps, y = wts2, color = treated_f)) +  
  geom_point() +  
  guides(color = FALSE) +  
  facet_wrap(~ treated_f) +  
  labs(x = "Estimated Propensity for Treatment",  
       y = "ATE weights for the toy example",  
       title = "ATE weighting structure: Toy example")
```

ATE weighting structure: Toy example



12.3 Assessing Balance after Weighting

The **twang** package provides several functions for assessing balance after weighting, in addition to actually doing the weighting using more complex propensity models. For this example, we'll demonstrate balance assessment for our two (relatively simple) weighting schemes. In other examples, we'll use **twang** to do more complete weighting work.

12.3.1 Reminder of ATT vs. ATE Definitions

- Informally, the **average treatment effect on the treated** (ATT) estimate describes the difference in potential outcomes (between treated and untreated subjects) summarized across the population of people who actually received the treatment. This is usually the estimate we work with in making causal estimates from observational studies.
- On the other hand, the **average treatment effect** (ATE) refers to the difference in potential outcomes summarized across the entire population, including those who did not receive the treatment.

12.3.2 For ATT weights (wts1)

```
toy_df <- data.frame(toy) # twang doesn't react well to tibbles

covlist <- c("covA", "covB", "covC", "covD", "covE", "covF", "Asqr", "BC", "BD", "ps", "linps")

# for ATT weights
bal.wts1 <- dx.wts(x=toy_df$wts1, data=toy_df, vars=covlist,
                  treat.var="treated", estimand="ATT")
bal.wts1
```

	type	n.treat	n.ctrl	ess.treat	ess.ctrl	max.es	mean.es	max.ks
1	unw	70	130	70	130.00000	1.1730321	0.40948475	0.4725275
2		70	130	70	62.42659	0.1259544	0.07031615	0.1998033

```
      mean.ks iter
1 0.19771767   NA
2 0.08785673   NA

bal.table(bal.wts1)
```

\$unw	tx.mn	tx.sd	ct.mn	ct.sd	std.eff.sz	stat	p	ks
covA	0.591	0.220	0.450	0.287	0.637	3.863	0.000	0.284
covB	0.486	0.503	0.308	0.463	0.354	2.461	0.015	0.178
covC	9.626	2.016	9.762	2.025	-0.068	-0.458	0.647	0.078
covD	10.096	1.877	9.738	1.770	0.191	1.317	0.189	0.110
covE	9.671	2.744	10.500	3.536	-0.302	-1.842	0.067	0.129
covF:1-Low	0.271	0.445	0.369	0.483	-0.220	1.317	0.269	0.098
covF:2-Middle	0.429	0.495	0.415	0.493	0.027	NA	NA	0.013
covF:3-High	0.300	0.458	0.215	0.411	0.185	NA	NA	0.085
Asqr	0.397	0.246	0.285	0.288	0.456	2.904	0.004	0.284
BC	4.584	5.008	2.857	4.449	0.345	2.427	0.016	0.187
BD	4.816	5.152	3.000	4.601	0.352	2.476	0.014	0.181
ps	0.476	0.191	0.282	0.187	1.014	6.922	0.000	0.473
linps	-0.139	0.917	-1.215	1.201	1.173	7.100	0.000	0.473

	ks.pval
covA	0.001
covB	0.098
covC	0.921
covD	0.598
covE	0.401
covF:1-Low	0.269
covF:2-Middle	0.269
covF:3-High	0.269
Asqr	0.001

```
BC          0.072
BD          0.087
ps          0.000
linps       0.000
```

```
[[2]]
```

	tx.mn	tx.sd	ct.mn	ct.sd	std.eff.sz	stat	p	ks
covA	0.591	0.220	0.565	0.216	0.116	0.740	0.460	0.200
covB	0.486	0.503	0.447	0.499	0.076	0.440	0.660	0.038
covC	9.626	2.016	9.477	1.960	0.074	0.442	0.659	0.086
covD	10.096	1.877	10.122	1.711	-0.014	-0.086	0.931	0.078
covE	9.671	2.744	9.495	3.430	0.064	0.323	0.747	0.125
covF:1-Low	0.271	0.445	0.300	0.458	-0.064	0.090	0.913	0.028
covF:2-Middle	0.429	0.495	0.396	0.489	0.067	NA	NA	0.033
covF:3-High	0.300	0.458	0.305	0.460	-0.010	NA	NA	0.005
Asqr	0.397	0.246	0.366	0.246	0.126	0.775	0.439	0.200
BC	4.584	5.008	4.170	4.844	0.083	0.482	0.630	0.081
BD	4.816	5.152	4.359	4.960	0.089	0.517	0.606	0.081
ps	0.476	0.191	0.463	0.177	0.069	0.410	0.682	0.093
linps	-0.139	0.917	-0.196	0.838	0.062	0.380	0.705	0.093

	ks.pval
covA	0.121
covB	1.000
covC	0.943
covD	0.972
covE	0.619
covF:1-Low	0.913
covF:2-Middle	0.913
covF:3-High	0.913
Asqr	0.121
BC	0.963
BD	0.963
ps	0.903
linps	0.903

The `std.eff.sz` shows the standardized difference, but as a proportion, rather than as a percentage. We'll create a data frame (tibble) so we can plot the data more easily.

```
bal.before.wts1 <- bal.table(bal.wts1)[1]
bal.after.wts1 <- bal.table(bal.wts1)[2]

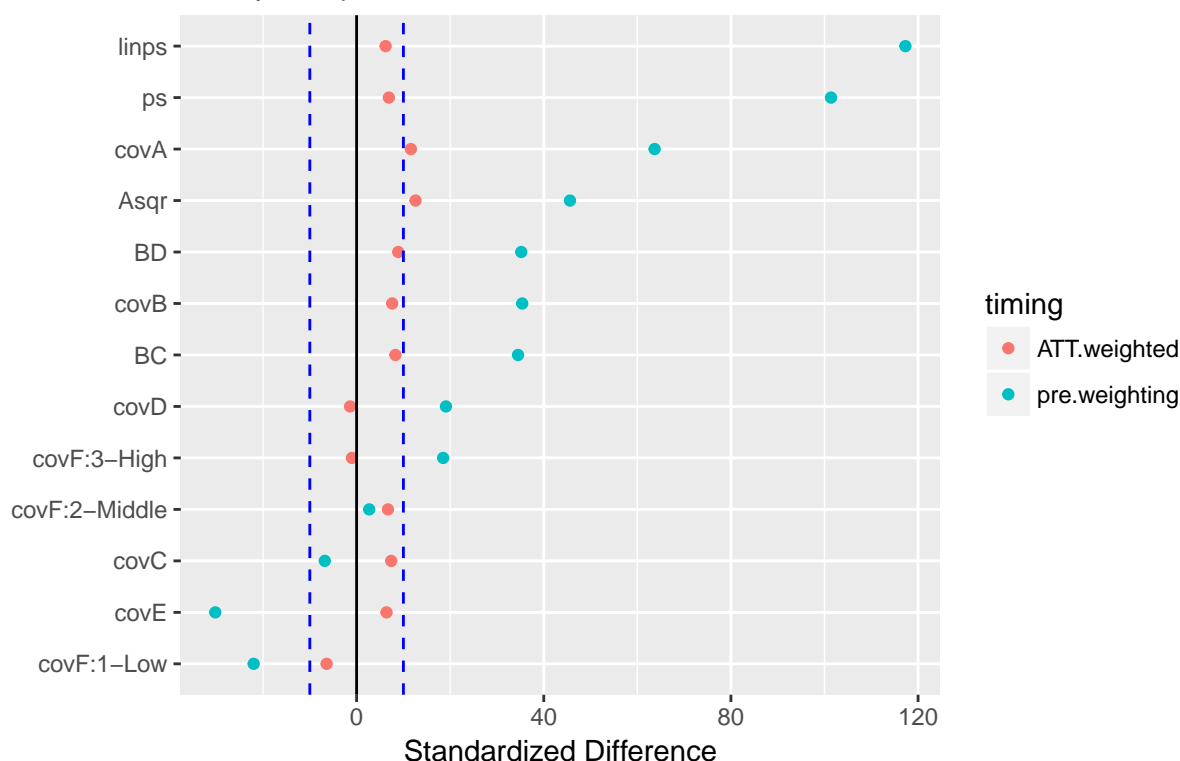
balance.att.weights <- data_frame(names = rownames(bal.before.wts1$unw),
                                   pre.weighting = 100*bal.before.wts1$unw$std.eff.sz,
                                   ATT.weighted = 100*bal.after.wts1[[1]]$std.eff.sz)
balance.att.weights <- gather(balance.att.weights, timing, szd, 2:3)
```

OK - here is the plot of standardized differences before and after ATT weighting.

```
ggplot(balance.att.weights, aes(x = szd, y = reorder(names, szd), color = timing)) +
  geom_point() +
  geom_vline(xintercept = 0) +
  geom_vline(xintercept = c(-10,10), linetype = "dashed", col = "blue") +
  labs(x = "Standardized Difference", y = "",
       title = "Standardized Difference before and after ATT Weighting",
       subtitle = "The toy example")
```

Standardized Difference before and after ATT Weighting

The toy example



12.3.3 For ATE weights (wts2)

```
bal.wts2 <- dx.wts(x=toy_df$wts2, data=toy_df, vars=covlist,
  treat.var="treated", estimand="ATE")
```

```
bal.wts2
```

	type	n.treat	n.ctrl	ess.treat	ess.ctrl	max.es	mean.es	max.ks
1	unw	70	130	70.00000	130.0000	0.9256094	0.36753949	0.4725275
2		70	130	38.96583	115.2132	0.1866526	0.05313253	0.1436468

```
  mean.ks iter
```

```
1 0.19771767 NA
```

```
2 0.07716551 NA
```

```
bal.table(bal.wts2)
```

```
$unw
```

	tx.mn	tx.sd	ct.mn	ct.sd	std.eff.sz	stat	p	ks
covA	0.591	0.220	0.450	0.287	0.513	3.863	0.000	0.284
covB	0.486	0.503	0.308	0.463	0.368	2.461	0.015	0.178
covC	9.626	2.016	9.762	2.025	-0.068	-0.458	0.647	0.078
covD	10.096	1.877	9.738	1.770	0.198	1.317	0.189	0.110
covE	9.671	2.744	10.500	3.536	-0.251	-1.842	0.067	0.129
covF:1-Low	0.271	0.445	0.369	0.483	-0.220	1.317	0.269	0.098
covF:2-Middle	0.429	0.495	0.415	0.493	0.027	NA	NA	0.013
covF:3-High	0.300	0.458	0.215	0.411	0.185	NA	NA	0.085
Asqr	0.397	0.246	0.285	0.288	0.403	2.904	0.004	0.284

BC	4.584	5.008	2.857	4.449	0.367	2.427	0.016	0.187
BD	4.816	5.152	3.000	4.601	0.373	2.476	0.014	0.181
ps	0.476	0.191	0.282	0.187	0.926	6.922	0.000	0.473
linps	-0.139	0.917	-1.215	1.201	0.881	7.100	0.000	0.473

	ks.pval
covA	0.001
covB	0.098
covC	0.921
covD	0.598
covE	0.401
covF:1-Low	0.269
covF:2-Middle	0.269
covF:3-High	0.269
Asqr	0.001
BC	0.072
BD	0.087
ps	0.000
linps	0.000

[[2]]

	tx.mn	tx.sd	ct.mn	ct.sd	std.eff.sz	stat	p	ks
covA	0.483	0.254	0.490	0.270	-0.025	-0.130	0.897	0.114
covB	0.347	0.479	0.356	0.481	-0.018	-0.109	0.913	0.009
covC	9.395	1.860	9.664	2.007	-0.139	-0.764	0.446	0.144
covD	10.208	1.860	9.870	1.759	0.187	1.087	0.278	0.133
covE	10.566	3.063	10.154	3.532	0.125	0.657	0.512	0.087
covF:1-Low	0.336	0.472	0.345	0.475	-0.020	0.011	0.988	0.010
covF:2-Middle	0.407	0.491	0.409	0.492	-0.002	NA	NA	0.001
covF:3-High	0.257	0.437	0.246	0.431	0.025	NA	NA	0.011
Asqr	0.297	0.260	0.313	0.277	-0.057	-0.322	0.748	0.114
BC	3.258	4.673	3.309	4.631	-0.011	-0.066	0.948	0.050
BD	3.426	4.852	3.468	4.772	-0.009	-0.051	0.959	0.048
ps	0.349	0.212	0.344	0.203	0.022	0.105	0.916	0.141
linps	-0.806	1.133	-0.864	1.193	0.050	0.225	0.822	0.141

	ks.pval
covA	0.802
covB	1.000
covC	0.541
covD	0.641
covE	0.963
covF:1-Low	0.988
covF:2-Middle	0.988
covF:3-High	0.988
Asqr	0.802
BC	1.000
BD	1.000
ps	0.566
linps	0.566

```
bal.before.wts2 <- bal.table(bal.wts2)[1]
```

```
bal.after.wts2 <- bal.table(bal.wts2)[2]
```

```
balance.ate.weights <- data_frame(names = rownames(bal.before.wts2$unw),
                                   pre.weighting = 100*bal.before.wts2$unw$std.eff.sz,
```

```

ATT.weighted = 100*bal.after.wts2[[1]]$std.eff.sz)
balance.ate.weights <- gather(balance.ate.weights, timing, szd, 2:3)

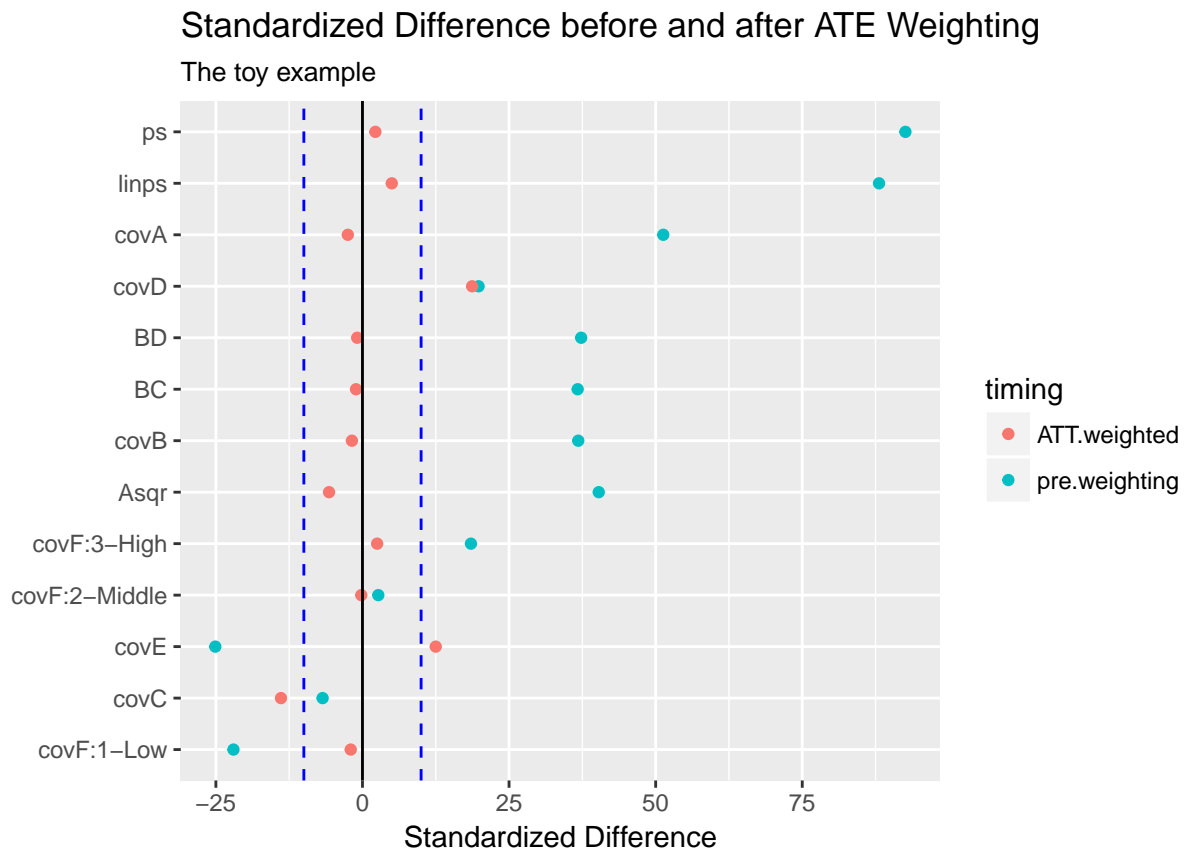
```

Here is the plot of standardized differences before and after ATE weighting.

```

ggplot(balance.ate.weights, aes(x = szd, y = reorder(names, szd), color = timing)) +
  geom_point() +
  geom_vline(xintercept = 0) +
  geom_vline(xintercept = c(-10,10), linetype = "dashed", col = "blue") +
  labs(x = "Standardized Difference", y = "",
       title = "Standardized Difference before and after ATE Weighting",
       subtitle = "The toy example")

```



12.4 Rubin's Rules after ATT weighting

For our weighted sample, our summary statistic for Rules 1 and 2 may be found from the `bal.table` output.

12.4.1 Rubin's Rule 1

We can read off the standardized effect size after weighting for the linear propensity score as 0.062. Multiplying by 100, we get 6.2%, so we would pass Rule 1.

12.4.2 Rubin's Rule 2

We can read off the standard deviations within the treated and control groups. We can then square each, to get the relevant variances, then take the ratio of those variances. Here, we have $0.917^2 / 0.838^2 = 1.1974314$, which is a substantial improvement over our unadjusted result, and now within our desired range of $4/5$ to $5/4$, as well as clearly within $1/2$ to 2 . Pass Rule 2, also.

12.4.3 Rubin's Rule 3

Rubin's Rule 3 requires some more substantial manipulation of the data. I'll skip that for now.

12.5 Rubin's Rules after ATE weighting

Again, our summary statistic for Rules 1 and 2 may be found from the `bal.table` output.

12.5.1 Rubin's Rule 1

The standardized effect size after ATE weighting for the linear propensity score is 0.050. Multiplying by 100, we get 5.0%, so we would pass Rule 1.

12.5.2 Rubin's Rule 2

We can read off the standard deviations within the treated and control groups from the ATE weights, then square to get the variances, then take the ratio. Here, we have $1.133^2 / 1.193^2 = 0.9019427$, which is also a substantial improvement over our unadjusted result, and within our desired range of $4/5$ to $5/4$. Pass Rule 2, also.

12.5.3 Rubin's Rule 3

Again, for now, I'm skipping Rubin's Rule 3 after weighting.

13 Using TWANG for Alternative PS Estimation and ATT Weighting

Here, I'll demonstrate the use of the `twang` package's functions to fit the propensity model and then perform ATT weighting, mostly using default options.

13.1 Estimate the Propensity Score using Generalized Boosted Regression, and then perform ATT Weighting

We can directly use the `twang` (toolkit for **w**eighting and **a**nalysis of **n**onequivalent **g**roups) package to weight our results, and even to re-estimate the propensity score using generalized boosted regression rather than a logistic regression model. The `twang` vignette is very helpful and found at this link.

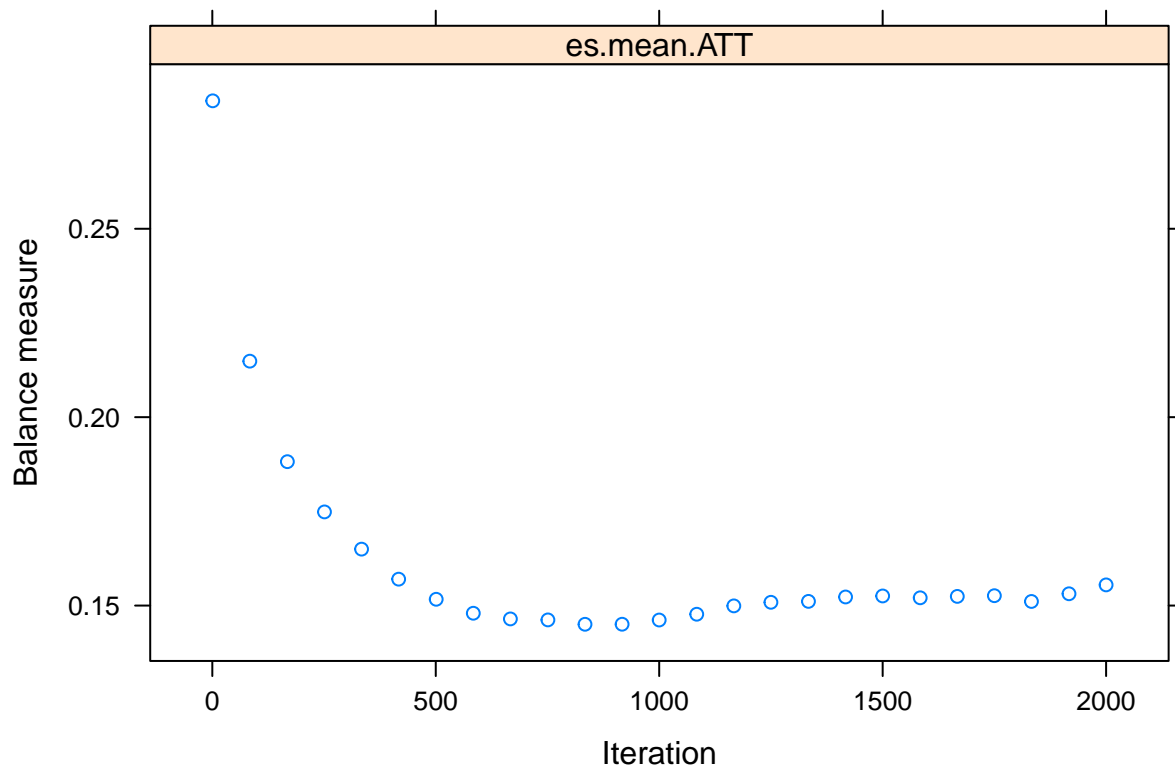
To begin, we'll estimate the propensity score using the `twang` function `ps`. This uses a *generalized boosted regression* approach to estimate the propensity score and produce material for checking balance.


```
# Recall that twang does not play well with tibbles,
# so we have to use the data frame version of the toy object

ps.toy <- ps(treated ~ covA + covB + covC + covD + covE + covF +
             Asqr + BC + BD,
             data = toy_df,
             n.trees = 2000,
             interaction.depth = 2,
             stop.method = c("es.mean"),
             estimand = "ATT",
             verbose = FALSE)
```

13.1.1 Did we let the simulations run long enough to stabilize estimates?

```
plot(ps.toy)
```



13.1.2 What is the effective sample size of our weighted results?

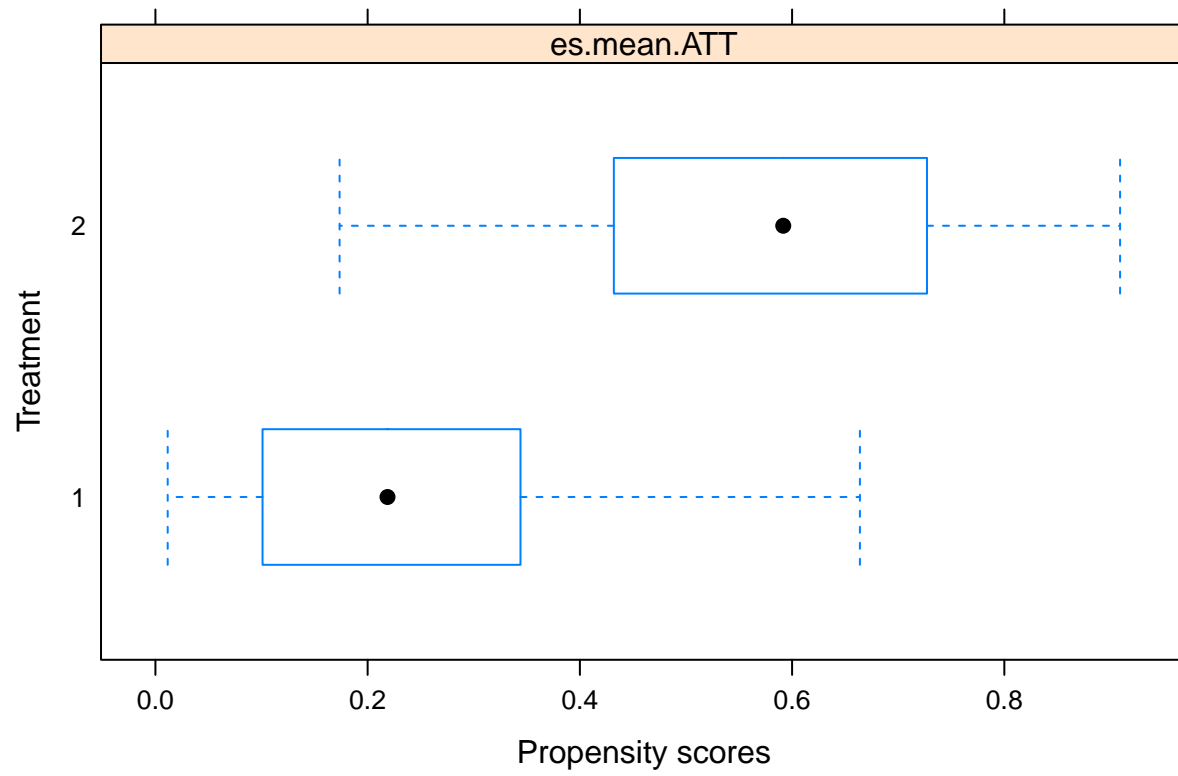
```
summary(ps.toy)
```

	n.treat	n.ctrl	ess.treat	ess.ctrl	max.es	mean.es
unw	70	130	70	130.0000	0.6368393	0.2850744
es.mean.ATT	70	130	70	73.3784	0.2340580	0.1448418

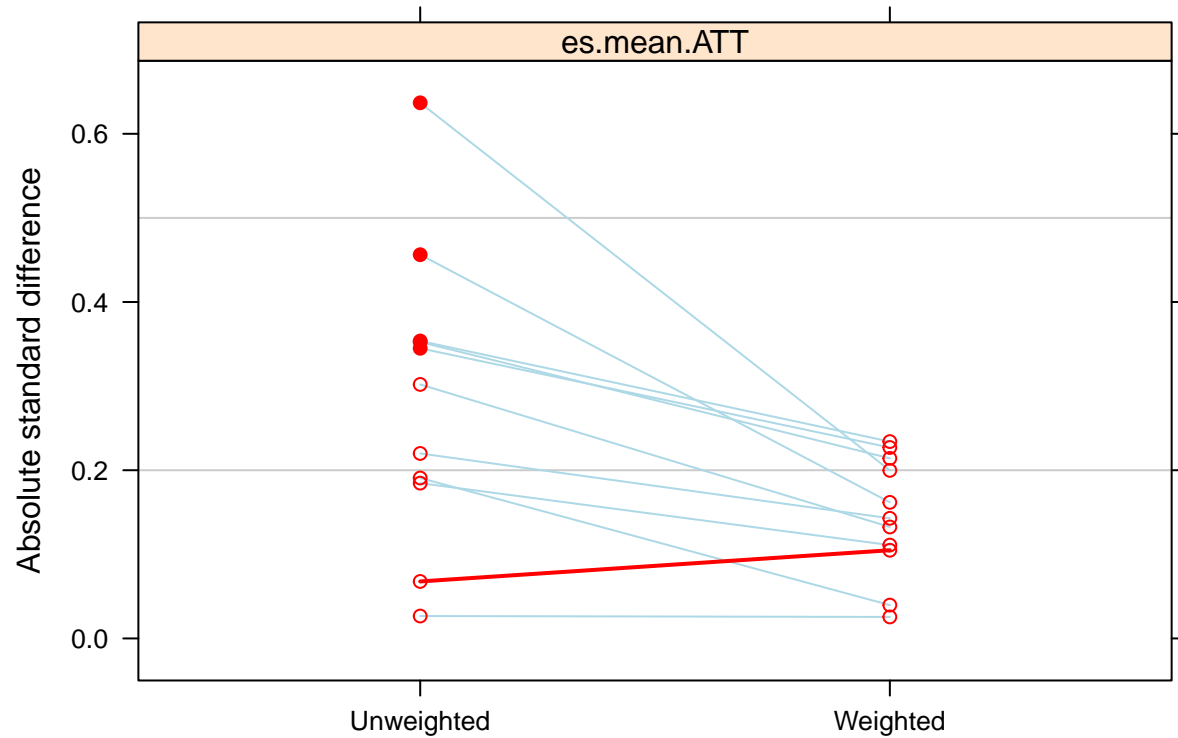
	max.ks	max.ks.p	mean.ks	iter
unw	0.2835165	NA	0.14775225	NA
es.mean.ATT	0.1524293	NA	0.09974725	870

13.1.3 How is the balance?

```
plot(ps.toy, plots = 2)
```



```
plot(ps.toy, plots = 3)
```



13.1.4 Assessing Balance with cobalt

```
bal.tab(ps.toy, full.stop.method = "es.mean.att")
```

Call:

```
ps(formula = treated ~ covA + covB + covC + covD + covE + covF +
    Asqr + BC + BD, data = toy_df, n.trees = 2000, interaction.depth = 2,
    verbose = FALSE, estimand = "ATT", stop.method = c("es.mean"))
```

Balance Measures:

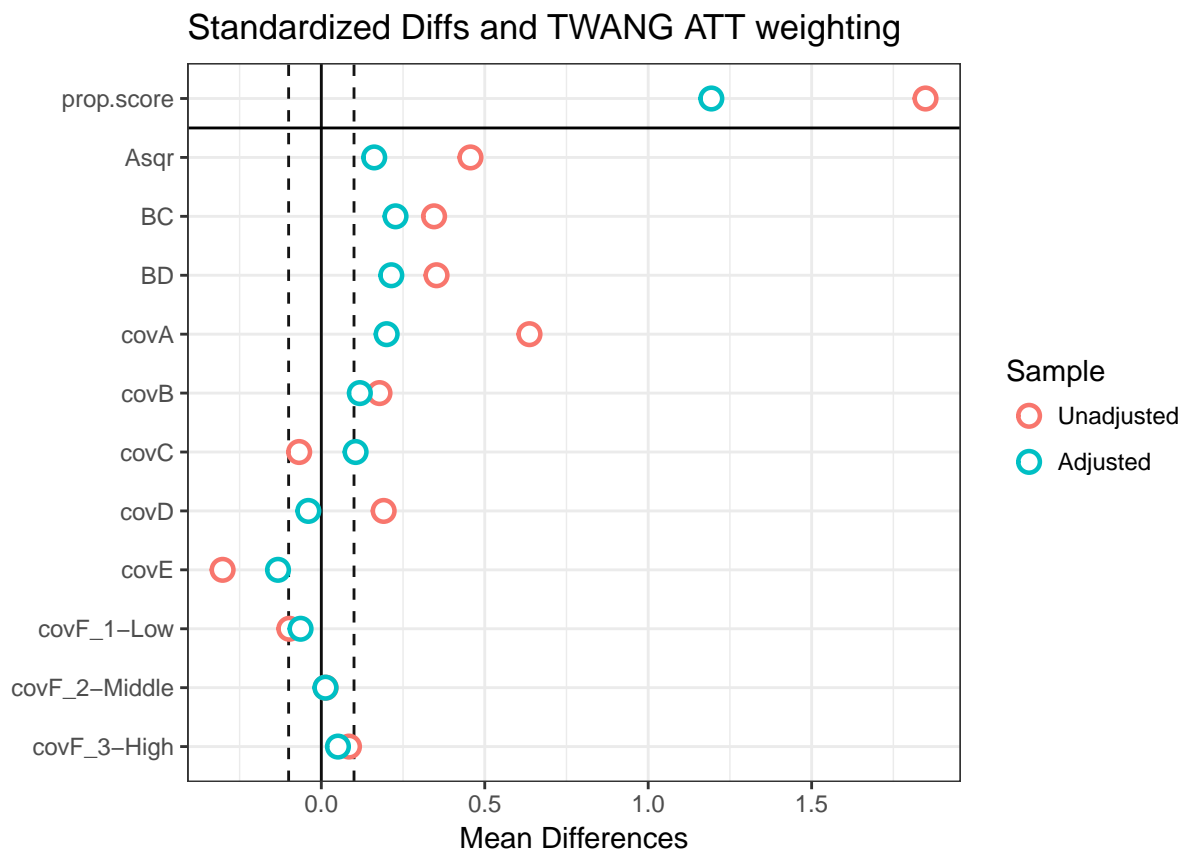
	Type	Diff.Adj
prop.score	Distance	1.1932
covA	Contin.	0.1998
covB	Binary	0.1178
covC	Contin.	0.1049
covD	Contin.	-0.0396
covE	Contin.	-0.1325
covF_1-Low	Binary	-0.0635
covF_2-Middle	Binary	0.0127
covF_3-High	Binary	0.0509
Asqr	Contin.	0.1618
BC	Contin.	0.2270
BD	Contin.	0.2142

Effective sample sizes:

	Control	Treated
Unadjusted	130.00	70
Adjusted	73.38	70

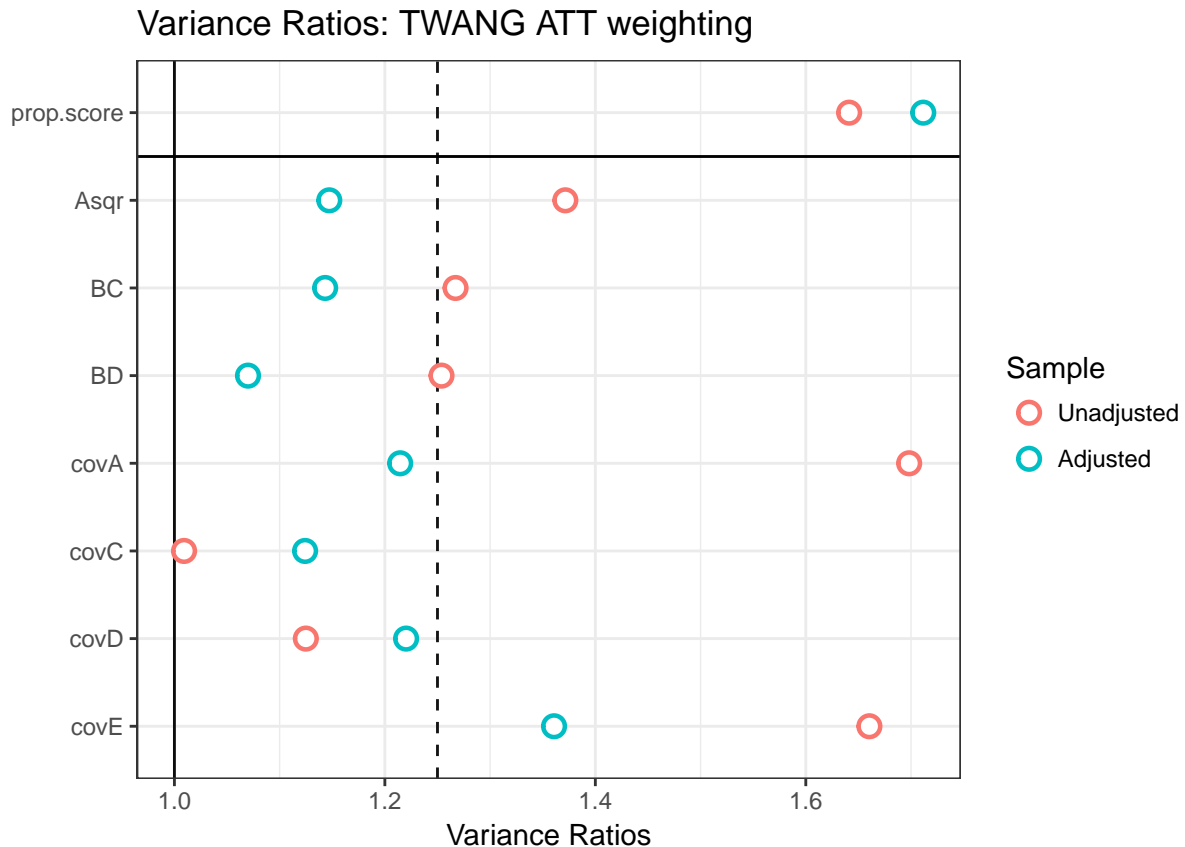
13.2 Semi-Automated Love plot of Standardized Differences

```
p <- love.plot(bal.tab(ps.toy),
               threshold = .1, size = 1.5,
               title = "Standardized Diffs and TWANG ATT weighting")
p + theme_bw()
```



13.3 Semi-Automated Love plot of Variance Ratios

```
p <- love.plot(bal.tab(ps.toy), stat = "v",
               threshold = 1.25, size = 1.5,
               title = "Variance Ratios: TWANG ATT weighting")
p + theme_bw()
```



14 Task 9. After weighting, what is the estimated average average causal effect of treatment?

14.1 ... on Outcome 1 [a continuous outcome]

14.1.1 with ATT weights

The relevant regression approach uses the `svydesign` and `svyglm` functions from the `survey` package.

```
toywt1.design <- svydesign(ids=~1, weights=~wts1, data=toy) # using ATT weights
```

```
adjout1.wt1 <- svyglm(out1.cost ~ treated, design=toywt1.design)
summary(adjout1.wt1); confint(adjout1.wt1)
```

Call:

```
svyglm(formula = out1.cost ~ treated, design = toywt1.design)
```

Survey design:

```
svydesign(ids = ~1, weights = ~wts1, data = toy)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	45.393	1.231	36.889	< 2e-16 ***

```

treated      15.221      2.323      6.553 4.78e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 189.3912)

Number of Fisher Scoring iterations: 2

            2.5 %    97.5 %
(Intercept) 42.98170 47.80528
treated      10.66821 19.77339

```

14.1.2 with ATE weights

```

toywt2.design <- svydesign(ids=~1, weights=~wts2, data=toy) # using ATE weights

adjout1.wt2 <- svyglm(out1.cost ~ treated, design=toywt2.design)
summary(adjout1.wt2); confint(adjout1.wt2)

```

Call:

```
svyglm(formula = out1.cost ~ treated, design = toywt2.design)
```

Survey design:

```
svydesign(ids = ~1, weights = ~wts2, data = toy)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	45.0951	0.9245	48.78	<2e-16 ***
treated	7.0474	3.5236	2.00	0.0469 *

```

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

(Dispersion parameter for gaussian family taken to be 223.8246)

Number of Fisher Scoring iterations: 2

	2.5 %	97.5 %
(Intercept)	43.2831607	46.90713
treated	0.1412766	13.95348

14.1.3 with TWANG ATT weights

```

toywt3.design <- svydesign(ids=~1,
                          weights=~get.weights(ps.toy,
                                                stop.method = "es.mean"),
                          data=toy) # using twang ATT weights

adjout1.wt3 <- svyglm(out1.cost ~ treated, design=toywt3.design)
summary(adjout1.wt3); confint(adjout1.wt3)

```

Call:

```
svyglm(formula = out1.cost ~ treated, design = toywt3.design)

Survey design:
svydesign(ids = ~1, weights = ~get.weights(ps.toy, stop.method = "es.mean"),
  data = toy)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   45.055      1.109  40.616 < 2e-16 ***
treated       15.560      2.261   6.882 7.6e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 202.799)

Number of Fisher Scoring iterations: 2

              2.5 %   97.5 %
(Intercept) 42.88061 47.22893
treated     11.12823 19.99080
```

14.2 ... on Outcome 2 [a binary outcome]

For a binary outcome, we build the outcome model using the quasibinomial, rather than the usual binomial family. We use the same `svydesign` information as we built for outcome 1.

14.2.1 Using ATT weights

```
adjout2.wt1 <- svyglm(out2 ~ treated, design=toywt1.design, family=quasibinomial())
summary(adjout2.wt1)
```

Call:

```
svyglm(formula = out2 ~ treated, design = toywt1.design, family = quasibinomial())
```

Survey design:

```
svydesign(ids = ~1, weights = ~wts1, data = toy)
```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.1611     0.2538  -0.635   0.526
treated       0.3907     0.3502   1.116   0.266
```

(Dispersion parameter for quasibinomial family taken to be 1.005025)

Number of Fisher Scoring iterations: 4

```
exp(summary(adjout2.wt1)$coef)
```

```
              Estimate Std. Error  t value Pr(>|t|)
(Intercept)  0.8512082   1.288958  0.5301153 1.692797
treated      1.4779751   1.419314  3.0515633 1.304634
```

```
exp(confint(adjout2.wt1))
```

```
              2.5 %    97.5 %  
(Intercept) 0.5175732 1.399909  
treated      0.7440438 2.935863
```

14.2.2 Using ATE weights

```
adjout2.wt2 <- svyglm(out2.event ~ treated, design=toywt2.design, family=quasibinomial())  
summary(adjout2.wt2)
```

Call:

```
svyglm(formula = out2.event ~ treated, design = toywt2.design,  
       family = quasibinomial())
```

Survey design:

```
svydesign(ids = ~1, weights = ~wts2, data = toy)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.1972	0.1877	-1.051	0.295
treated	0.2545	0.3713	0.685	0.494

(Dispersion parameter for quasibinomial family taken to be 1.005025)

Number of Fisher Scoring iterations: 3

```
exp(summary(adjout2.wt2)$coef)
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.821019	1.206466	0.3496972	1.342700
treated	1.289826	1.449650	1.9845839	1.638674

```
exp(confint(adjout2.wt2))
```

```
              2.5 %    97.5 %  
(Intercept) 0.5683120 1.186095  
treated      0.6229616 2.670553
```

14.2.3 with TWANG ATT weights

```
adjout2.wt3 <- svyglm(out2 ~ treated, design=toywt3.design,  
                    family=quasibinomial())  
summary(adjout2.wt3)
```

Call:

```
svyglm(formula = out2 ~ treated, design = toywt3.design, family = quasibinomial())
```

Survey design:

```
svydesign(ids = ~1, weights = ~get.weights(ps.toy, stop.method = "es.mean"),  
       data = toy)
```


Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.2581	0.2335	-1.105	0.270
treated	0.4877	0.3358	1.452	0.148

(Dispersion parameter for quasibinomial family taken to be 1.005025)

Number of Fisher Scoring iterations: 4

```
exp(summary(adjout2.wt3)$coef)
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.7725293	1.263056	0.331168	1.310547
treated	1.6285006	1.398989	4.273565	1.159466

```
exp(confint(adjout2.wt3))
```

	2.5 %	97.5 %
(Intercept)	0.4887992	1.220954
treated	0.8433287	3.144698

14.3 ... on Outcome 3 [a time to event]

As before, subjects with `out2.event = "Yes"` are truly observed events, while those with `out2.event == "No"` are censored before an event can happen to them.

14.3.1 Using ATT weights

The Cox model comparing treated to control, weighting by ATT weights (`wts1`), is...

```
adjout3.wt1 <- coxph(Surv(out3.time, out2) ~ treated, data=toy, weights=wts1)
summary(adjout3.wt1)
```

Call:

```
coxph(formula = Surv(out3.time, out2) ~ treated, data = toy,
      weights = wts1)
```

n= 200, number of events= 97

	coef	exp(coef)	se(coef)	z	Pr(> z)
treated	-0.1975	0.8208	0.2417	-0.817	0.414

	exp(coef)	exp(-coef)	lower .95	upper .95
treated	0.8208	1.218	0.511	1.318

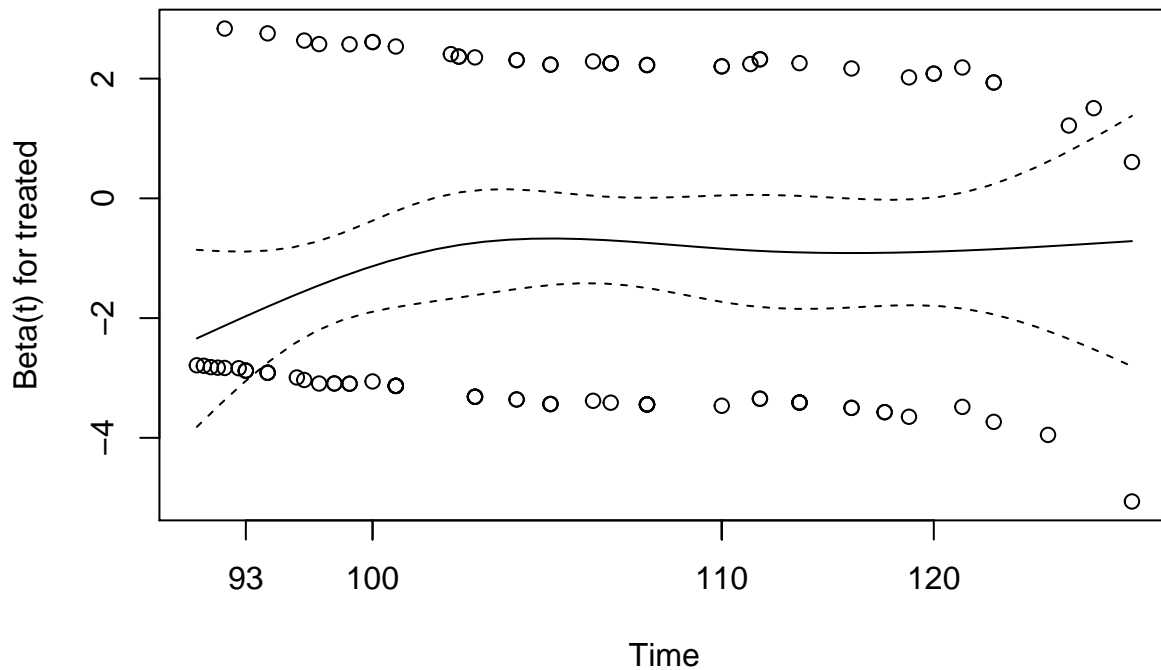
Concordance= 0.549 (se = 0.033)
Rsquare= 0.003 (max possible= 0.951)
Likelihood ratio test= 0.66 on 1 df, p=0.4158
Wald test = 0.67 on 1 df, p=0.4139
Score (logrank) test = 0.67 on 1 df, p=0.4132

The `exp(coef)` output gives the relative hazard of the event comparing treated subjects to control subjects.

And here's the check of the proportional hazards assumption...

```
cox.zph(adjout3.wt1); plot(cox.zph(adjout3.wt1), var="treated")
```

```
      rho chisq    p
treated 0.111  1.57 0.21
```



14.3.2 Using ATE weights

```
adjout3.wt2 <- coxph(Surv(out3.time, out2) ~ treated, data=toy, weights=wts2)
summary(adjout3.wt2)
```

Call:

```
coxph(formula = Surv(out3.time, out2) ~ treated, data = toy,
      weights = wts2)
```

n= 200, number of events= 97

	coef	exp(coef)	se(coef)	z	Pr(> z)
treated	-0.1814	0.8341	0.1451	-1.251	0.211

	exp(coef)	exp(-coef)	lower .95	upper .95
treated	0.8341	1.199	0.6276	1.108

Concordance= 0.547 (se = 0.02)

Rsquare= 0.008 (max possible= 1)

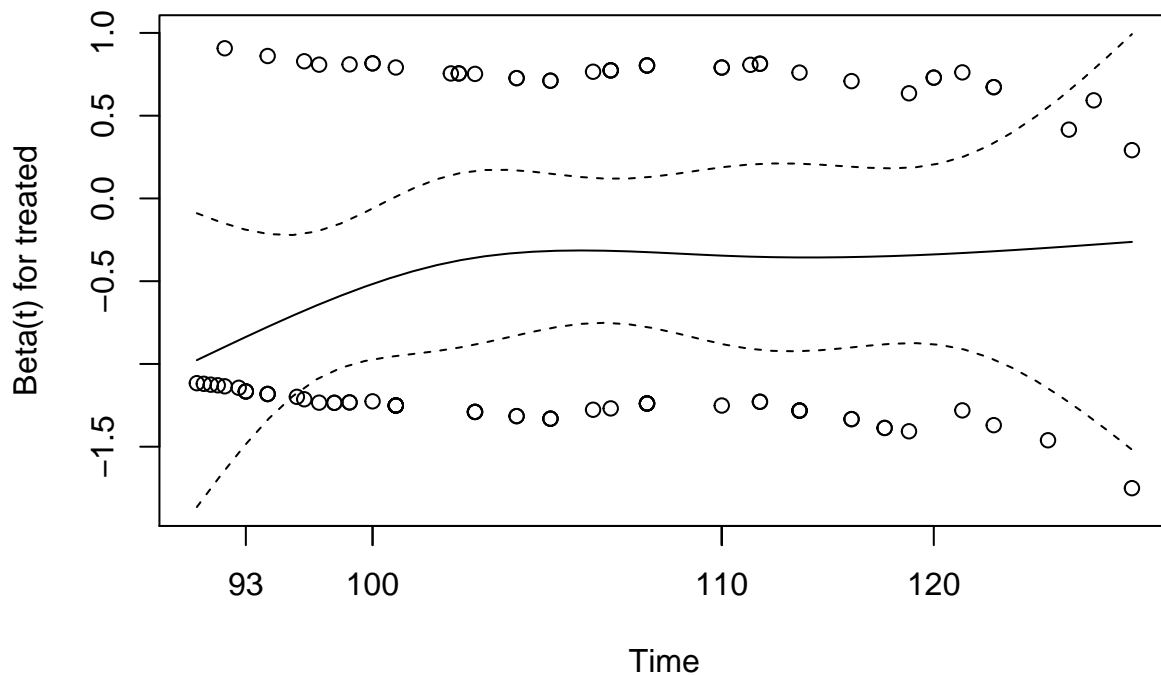
Likelihood ratio test= 1.56 on 1 df, p=0.2124

```
Wald test          = 1.56  on 1 df,   p=0.2111
Score (logrank) test = 1.57  on 1 df,   p=0.2105
```

And here's the check of the proportional hazards assumption...

```
cox.zph(adjout3.wt2); plot(cox.zph(adjout3.wt2), var="treated")
```

```
      rho chisq      p
treated 0.151  1.06 0.303
```



14.3.3 with TWANG ATT weights

```
wts3 <- get.weights(ps.toy, stop.method = "es.mean")

adjout3.wt3 <- coxph(Surv(out3.time, out2) ~ treated, data=toy, weights=wts3)
summary(adjout3.wt3)
```

Call:

```
coxph(formula = Surv(out3.time, out2) ~ treated, data = toy,
      weights = wts3)
```

n= 200, number of events= 97

	coef	exp(coef)	se(coef)	z	Pr(> z)
treated	-0.1056	0.8997	0.2758	-0.383	0.702

	exp(coef)	exp(-coef)	lower .95	upper .95
treated	0.8997	1.1115	0.3488	2.3888

```
treated    0.8997      1.111      0.524      1.545
```

```
Concordance= 0.539 (se = 0.034 )
Rsquare= 0.001 (max possible= 0.913 )
Likelihood ratio test= 0.15 on 1 df, p=0.7033
Wald test              = 0.15 on 1 df, p=0.7017
Score (logrank) test = 0.15 on 1 df, p=0.7016
```

14.4 Results So Far (After Matching, Subclassification and Weighting)

Est. Treatment Effect (95% CI)	Outcome 1 (Cost diff.)	Outcome 2 (Risk diff.)	Outcome 2 (Odds Ratio)	Outcome 3 (Relative Hazard Rate)
No covariate adjustment (unadjusted, ATT)	15.7 (12.0, 19.3)	+0.11 (-0.03, +0.25)	1.56 (0.87, 2.82)	0.86 (0.57, 1.29)
1:1 PS Match (Match: ATT)	15.6 (11.6, 19.6)	+0.11 (-0.05, +0.25)	N/A	N/A
1:1 PS Match ("Regression", ATT)	15.6 (11.5, 19.6)	N/A	1.64 (0.77, 3.47)	0.75 (0.41, 1.38)
PS Subclassification (ATE)	7.9 (4.1, 11.7)	N/A	1.15 (0.54, 2.44)	0.79 (0.49, 1.27)
ATT Weighting (ATT)	15.2 (10.7, 19.8)	N/A	1.48 (0.74, 2.94)	0.82 (0.51, 1.32)
ATE Weighting (ATE)	7.1 (0.1, 14.0)	N/A	1.29 (0.62, 2.67)	0.83 (0.63, 1.11)
twang ATT weights (ATT)	15.6 (11.1, 20.0)	N/A	1.63 (0.84, 3.14)	0.90 (0.52, 1.55)

15 Task 10. After direct adjustment for the linear PS, what is the estimated average causal treatment effect?

15.1 ... on Outcome 1 [a continuous outcome]

Here, we fit a linear regression model with `linps` added as a covariate.

```
adj.reg.out1 <- lm(out1.cost ~ treated + linps, data=toy)
summary(adj.reg.out1); confint(adj.reg.out1)
```

Call:

```
lm(formula = out1.cost ~ treated + linps, data = toy)
```

Residuals:

```
      Min       1Q   Median       3Q      Max
-32.071  -8.449   1.481   8.135  28.651
```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  48.5668      1.4186  34.235 < 2e-16 ***
```

```

treated      12.4629      1.9782      6.300 1.9e-09 ***
linps        2.9859      0.7746      3.855 0.000157 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.1 on 197 degrees of freedom
Multiple R-squared:  0.3165,    Adjusted R-squared:  0.3095
F-statistic: 45.6 on 2 and 197 DF,  p-value: < 2.2e-16

              2.5 %    97.5 %
(Intercept) 45.769157 51.364410
treated      8.561701 16.364085
linps        1.458316  4.513435
## provides treated effect and confidence interval estimates

```

15.2 ... on Outcome 2 [a binary outcome]

Here, fit a logistic regression with linps added as a covariate

```

adj.reg.out2 <- glm(out2 ~ treated + linps, data=toy, family=binomial())
summary(adj.reg.out2)

```

Call:

```
glm(formula = out2 ~ treated + linps, family = binomial(), data = toy)
```

Deviance Residuals:

```

      Min       1Q   Median       3Q      Max
-1.4003  -1.1256  -0.9414   1.1915   1.4281

```

Coefficients:

```

              Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.05119    0.23531  -0.218   0.828
treated      0.30064    0.32835   0.916   0.360
linps        0.13716    0.13027   1.053   0.292

```

(Dispersion parameter for binomial family taken to be 1)

```

Null deviance: 277.08  on 199  degrees of freedom
Residual deviance: 273.71  on 197  degrees of freedom
AIC: 279.71

```

Number of Fisher Scoring iterations: 4

```
exp(coef(adj.reg.out2)) # produces odds ratio estimate
```

```

(Intercept)      treated      linps
  0.9501023    1.3507214    1.1470102

```

```
exp(confint(adj.reg.out2)) # produces 95% CI for odds ratio
```

Waiting for profiling to be done...

```

              2.5 %    97.5 %
(Intercept) 0.5977473 1.508760
treated      0.7097588 2.580375

```

```
linps      0.8902357 1.487135
```

15.3 ... on Outcome 3 [a time-to-event outcome]

Again, subjects with `out2.event` No are right-censored, those with Yes for `out2.event` have their times to event observed.

We fit a Cox proportional hazards model predicting time to event (with `event=Yes` indicating non-censored cases) based on treatment group (`treated`) and now also the linear propensity score.

```
adj.reg.out3 <- coxph(Surv(out3.time, out2) ~ treated + linps, data=toy)
summary(adj.reg.out3)
```

Call:

```
coxph(formula = Surv(out3.time, out2) ~ treated + linps, data = toy)
```

```
n= 200, number of events= 97
```

	coef	exp(coef)	se(coef)	z	Pr(> z)
treated	-0.2274	0.7966	0.2343	-0.97	0.332
linps	0.0684	1.0708	0.1005	0.68	0.496

	exp(coef)	exp(-coef)	lower .95	upper .95
treated	0.7966	1.2553	0.5032	1.261
linps	1.0708	0.9339	0.8793	1.304

```
Concordance= 0.559 (se = 0.033 )
Rsquare= 0.005 (max possible= 0.988 )
Likelihood ratio test= 1.01 on 2 df, p=0.6022
Wald test = 1.01 on 2 df, p=0.6025
Score (logrank) test = 1.01 on 2 df, p=0.6021
```

The `exp(coef)` section indicates the relative hazard estimates and associated 95% CI.

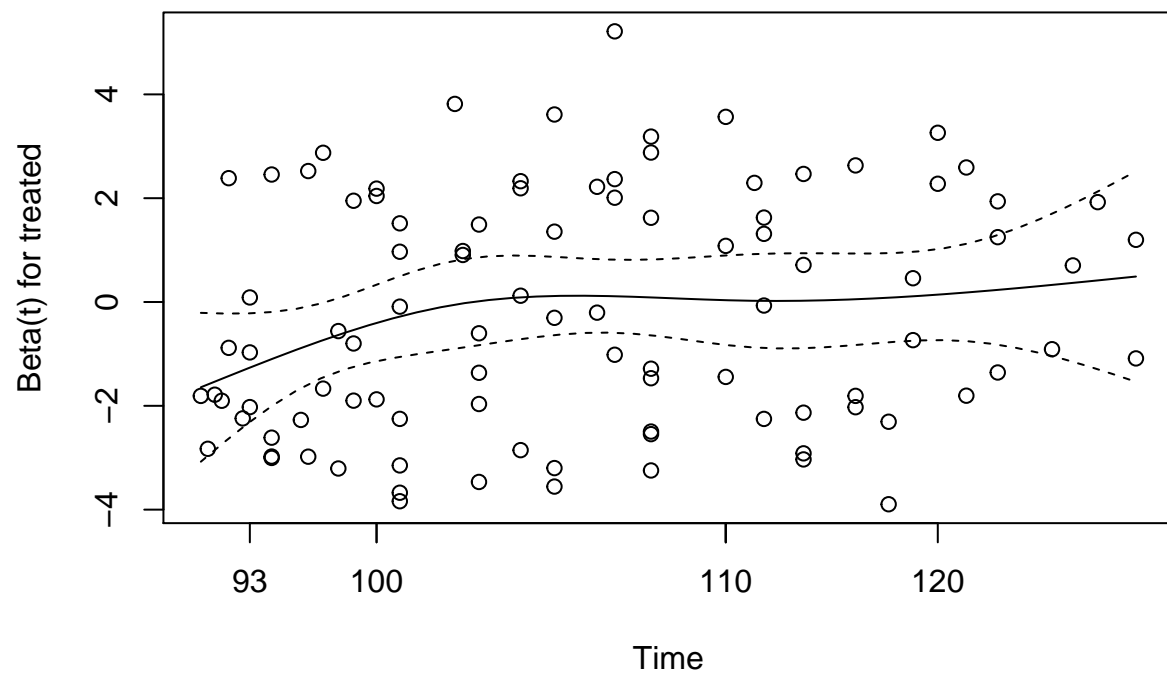
15.3.1 Check proportional hazards assumption

Not the best of news here. The results are close to being statistically significant.

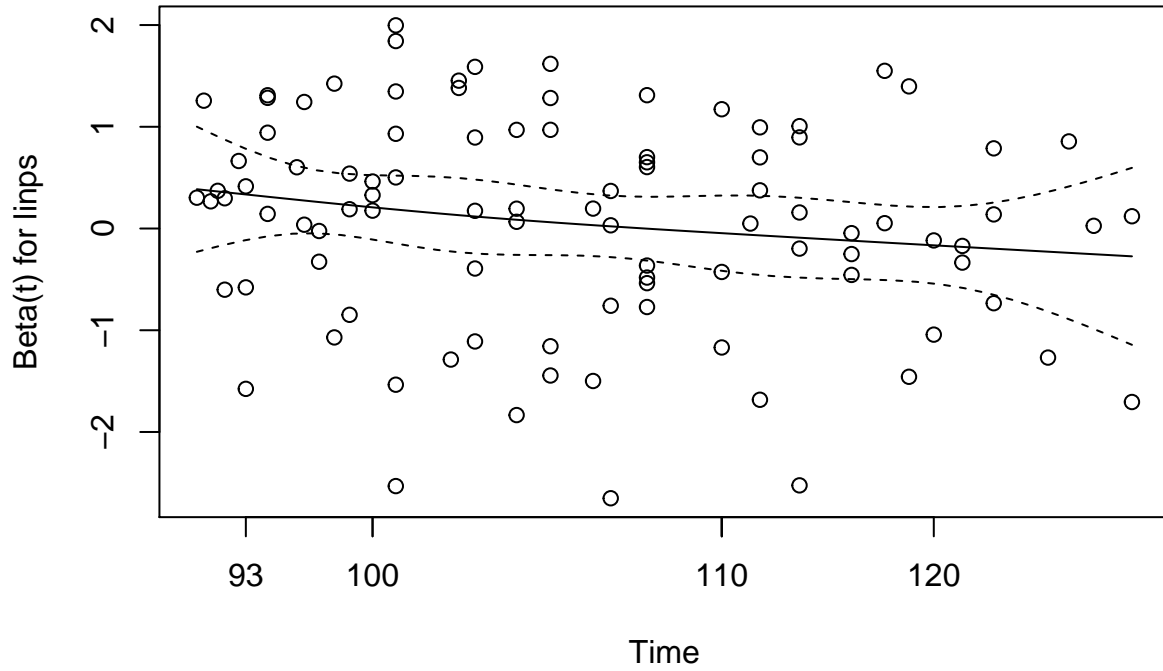
```
cox.zph(adj.reg.out3)
```

	rho	chisq	p
treated	0.193	3.40	0.0651
linps	-0.174	3.17	0.0751
GLOBAL	NA	4.52	0.1045

```
plot(cox.zph(adj.reg.out3), var="treated")
```



```
plot(cox.zph(adj.reg.out3), var="linps")
```



15.4 Results So Far (After Matching, Subclassification, Weighting, Adjustment)

Est. Treatment Effect (95% CI)	Outcome 1 (Cost diff.)	Outcome 2 (Risk diff.)	Outcome 2 (Odds Ratio)	Outcome 3 (Relative Hazard Rate)
No covariate adjustment	15.7	+0.11	1.56	0.86
(unadjusted, ATT)	(12.0, 19.3)	(-0.03, +0.25)	(0.87, 2.82)	(0.57, 1.29)
1:1 PS Match	15.6	+0.11	N/A	N/A
(Match: ATT)	(11.6, 19.6)	(-0.05, +0.25)	N/A	N/A
1:1 PS Match	15.6	N/A	1.64	0.75
(“Regression”, ATT)	(11.5, 19.6)	N/A	(0.77, 3.47)	(0.41, 1.38)
PS Subclassification	7.9	N/A	1.15	0.79
(“Regression”, ATE)	(4.1, 11.7)	N/A	(0.54, 2.44)	(0.49, 1.27)
ATT Weighting	15.2	N/A	1.48	0.82
(ATT)	(10.7, 19.8)	(0.74, 2.94)	(0.51, 1.32)	
ATE Weighting	7.1	N/A	1.29	0.83
(ATE)	(0.1, 14.0)	N/A	(0.62, 2.67)	(0.63, 1.11)
twang ATT weights	15.6	N/A	1.63	0.90
(ATT)	(11.1, 20.0)	N/A	(0.84, 3.14)	(0.52, 1.55)
Direct Adjustment	12.5	N/A	1.35	0.80
(with linps, ATT)	(8.56, 16.36)	N/A	(0.71, 2.58)	(0.50, 1.26)

16 Task 11. “Double Robust” Approach - Weighting + Adjustment, what is the estimated average causal effect of treatment?

This approach is essentially identical to the weighting analyses done in Task 9. The only change is to add `linps` to `treated` in the outcome models.

16.1 ... on Outcome 1 [a continuous outcome]

16.1.1 with ATT weights

The relevant regression approach uses the `svydesign` and `svyglm` functions from the `survey` package.

```
toywt1.design <- svydesign(ids=~1, weights=~wts1, data=toy) # using ATT weights

dr.out1.wt1 <- svyglm(out1.cost ~ treated + linps, design=toywt1.design)
summary(dr.out1.wt1); confint(dr.out1.wt1)
```

Call:

```
svyglm(formula = out1.cost ~ treated + linps, design = toywt1.design)
```

Survey design:

```
svydesign(ids = ~1, weights = ~wts1, data = toy)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	46.817	1.534	30.528	< 2e-16 ***
treated	14.808	2.122	6.980	4.41e-11 ***
linps	7.264	1.230	5.906	1.51e-08 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 148.853)

Number of Fisher Scoring iterations: 2

	2.5 %	97.5 %
(Intercept)	43.810924	49.82230
treated	10.649919	18.96660
linps	4.853559	9.67487

16.1.2 with ATE weights

```
toywt2.design <- svydesign(ids=~1, weights=~wts2, data=toy) # using ATE weights

dr.out1.wt2 <- svyglm(out1.cost ~ treated + linps, design=toywt2.design)
summary(dr.out1.wt2); confint(dr.out1.wt2)
```

Call:

```
svyglm(formula = out1.cost ~ treated + linps, design = toywt2.design)
```

Survey design:

```
svydesign(ids = ~1, weights = ~wts2, data = toy)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	50.420	1.695	29.750	< 2e-16 ***
treated	6.691	2.581	2.593	0.0102 *
linps	6.162	1.356	4.545	9.57e-06 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 172.751)

Number of Fisher Scoring iterations: 2

	2.5 %	97.5 %
(Intercept)	47.098373	53.741875
treated	1.633441	11.749311
linps	3.504662	8.818882

16.1.3 with twang based ATT weights

```
wts3 <- get.weights(ps.toy, stop.method = "es.mean")  
  
toywt3.design <- svydesign(ids=~1, weights=~wts3, data=toy) # twang ATT weights  
  
dr.out1.wt3 <- svyglm(out1.cost ~ treated + linps, design=toywt3.design)  
summary(dr.out1.wt3); confint(dr.out1.wt3)
```

Call:

```
svyglm(formula = out1.cost ~ treated + linps, design = toywt3.design)
```

Survey design:

```
svydesign(ids = ~1, weights = ~wts3, data = toy)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	48.694	1.717	28.367	< 2e-16 ***
treated	12.954	2.294	5.646	5.68e-08 ***
linps	7.435	1.260	5.900	1.56e-08 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 155.9117)

Number of Fisher Scoring iterations: 2

	2.5 %	97.5 %
(Intercept)	45.329842	52.058658
treated	8.457203	17.451434
linps	4.964688	9.904405

16.2 ... on Outcome 2 [a binary outcome]

For a binary outcome, we build the outcome model using the quasibinomial, rather than the usual binomial family. We use the same `svydesign` information as we built for outcome 1.

16.2.1 Using ATT weights

```
dr.out2.wt1 <- svyglm(out2 ~ treated + linps, design=toywt1.design,
                      family=quasibinomial())
summary(dr.out2.wt1)
```

Call:

```
svyglm(formula = out2 ~ treated + linps, design = toywt1.design,
       family = quasibinomial())
```

Survey design:

```
svydesign(ids = ~1, weights = ~wts1, data = toy)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.1186	0.2781	-0.427	0.670
treated	0.3817	0.3559	1.072	0.285
linps	0.2262	0.2052	1.102	0.272

(Dispersion parameter for quasibinomial family taken to be 1.005348)

Number of Fisher Scoring iterations: 4

```
exp(summary(dr.out2.wt1)$coef)
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.888133	1.320594	0.6527147	1.954483
treated	1.464723	1.427472	2.9223339	1.329576
linps	1.253772	1.227815	3.0099762	1.312371

```
exp(confint(dr.out2.wt1))
```

	2.5 %	97.5 %
(Intercept)	0.5149615	1.531727
treated	0.7291363	2.942405
linps	0.8385347	1.874632

16.2.2 Using ATE weights

```
dr.out2.wt2 <- svyglm(out2.event ~ treated + linps, design=toywt2.design,
                      family=quasibinomial())
summary(dr.out2.wt2)
```

Call:

```
svyglm(formula = out2.event ~ treated + linps, design = toywt2.design,
       family = quasibinomial())
```

Survey design:

```
svydesign(ids = ~1, weights = ~wts2, data = toy)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.11246	0.27201	-0.413	0.680
treated	0.24972	0.38320	0.652	0.515
linps	0.09896	0.20309	0.487	0.627

(Dispersion parameter for quasibinomial family taken to be 1.005037)

Number of Fisher Scoring iterations: 4

```
exp(summary(dr.out2.wt2)$coef)
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.8936347	1.312604	0.6613785	1.973369
treated	1.2836645	1.466966	1.9187497	1.674260
linps	1.1040237	1.225184	1.6278777	1.871245

```
exp(confint(dr.out2.wt2))
```

	2.5 %	97.5 %
(Intercept)	0.5243509	1.522994
treated	0.6057234	2.720375
linps	0.7414934	1.643802

16.2.3 Using twang ATT weights

```
dr.out2.wt3 <- svyglm(out2 ~ treated + linps, design=toywt3.design,  
                      family=quasibinomial())  
summary(dr.out2.wt3)
```

Call:

```
svyglm(formula = out2 ~ treated + linps, design = toywt3.design,  
       family = quasibinomial())
```

Survey design:

```
svydesign(ids = ~1, weights = ~wts3, data = toy)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.1707	0.2748	-0.621	0.535
treated	0.4271	0.3546	1.204	0.230
linps	0.1828	0.1930	0.947	0.345

(Dispersion parameter for quasibinomial family taken to be 1.005653)

Number of Fisher Scoring iterations: 4

```
exp(summary(dr.out2.wt3)$coef)
```

	Estimate	Std. Error	t value	Pr(> t)
--	----------	------------	---------	----------

```
(Intercept) 0.8430793 1.316301 0.5373532 1.707879
treated      1.5327500 1.425640 3.3343637 1.258506
linps        1.2006145 1.212918 2.5784289 1.411577
```

```
exp(confint(dr.out2.wt3))
```

```
                2.5 %    97.5 %
(Intercept) 0.4919674 1.444776
treated      0.7649226 3.071321
linps        0.8224259 1.752711
```

16.3 ... on Outcome 3 [a time to event]

As before, subjects with `out2.event = "Yes"` are truly observed events, while those with `out2.event == "No"` are censored before an event can happen to them.

16.3.1 Using ATT weights

The Cox model comparing treated to control, weighting by ATT weights (`wts1`), is...

```
dr.out3.wt1 <- coxph(Surv(out3.time, out2) ~ treated + linps, data=toy, weights=wts1)
summary(dr.out3.wt1)
```

Call:

```
coxph(formula = Surv(out3.time, out2) ~ treated + linps, data = toy,
      weights = wts1)
```

```
n= 200, number of events= 97
```

	coef	exp(coef)	se(coef)	z	Pr(> z)
treated	-0.2086	0.8118	0.2424	-0.861	0.39
linps	0.1045	1.1102	0.1445	0.723	0.47

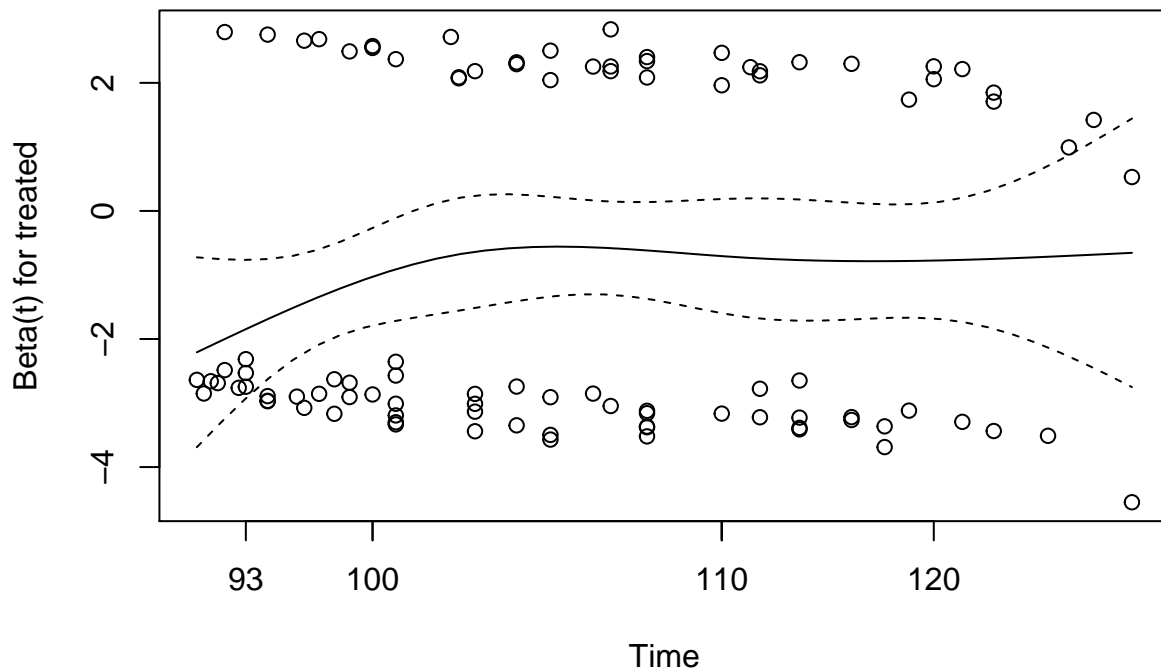
	exp(coef)	exp(-coef)	lower .95	upper .95
treated	0.8118	1.2319	0.5048	1.305
linps	1.1102	0.9007	0.8363	1.474

```
Concordance= 0.564 (se = 0.039 )
Rsquare= 0.006 (max possible= 0.951 )
Likelihood ratio test= 1.2 on 2 df, p=0.5501
Wald test = 1.19 on 2 df, p=0.5514
Score (logrank) test = 1.19 on 2 df, p=0.5508
```

The `exp(coef)` output gives the relative hazard of the event comparing treated subjects to control subjects. And here's the check of the proportional hazards assumption...

```
cox.zph(dr.out3.wt1); plot(cox.zph(dr.out3.wt1), var="treated")
```

	rho	chisq	p
treated	0.1151	1.539	0.215
linps	-0.0332	0.298	0.585
GLOBAL	NA	1.759	0.415



16.3.2 Using ATE weights

```
dr.out3.wt2 <- coxph(Surv(out3.time, out2) ~ treated + linps, data=toy, weights=wts2)
summary(dr.out3.wt2)
```

Call:

```
coxph(formula = Surv(out3.time, out2) ~ treated + linps, data = toy,
      weights = wts2)
```

n= 200, number of events= 97

	coef	exp(coef)	se(coef)	z	Pr(> z)
treated	-0.181102	0.834351	0.145821	-1.242	0.214
linps	-0.001565	0.998436	0.065906	-0.024	0.981

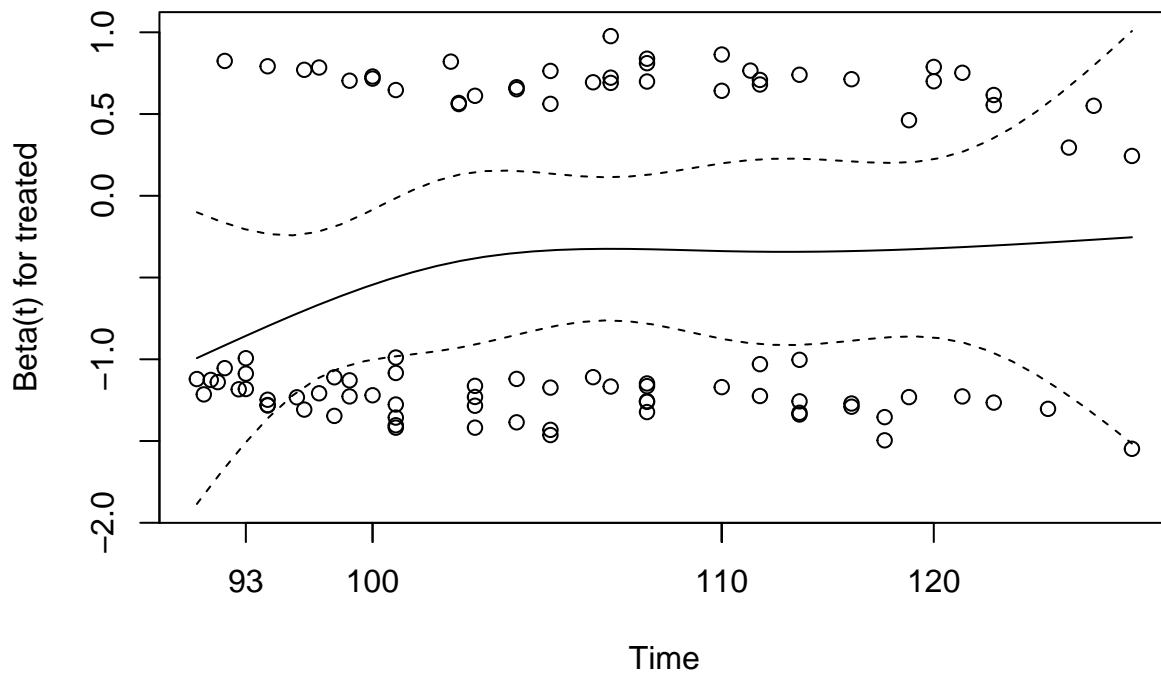
	exp(coef)	exp(-coef)	lower .95	upper .95
treated	0.8344	1.199	0.6269	1.110
linps	0.9984	1.002	0.8774	1.136

Concordance= 0.533 (se = 0.023)
 Rsquare= 0.008 (max possible= 1)
 Likelihood ratio test= 1.56 on 2 df, p=0.4594
 Wald test = 1.56 on 2 df, p=0.4574
 Score (logrank) test = 1.57 on 2 df, p=0.4564

And here's the check of the proportional hazards assumption...

```
cox.zph(dr.out3.wt2); plot(cox.zph(dr.out3.wt2), var="treated")
```

	rho	chisq	p
treated	0.173	1.27	0.260
linps	-0.136	1.02	0.313
GLOBAL	NA	2.08	0.353



16.3.3 Using twang ATT weights

```
dr.out3.wt3 <- coxph(Surv(out3.time, out2) ~ treated + linps,
  data=toy, weights=wts3)
summary(dr.out3.wt3)
```

Call:

```
coxph(formula = Surv(out3.time, out2) ~ treated + linps, data = toy,
  weights = wts3)
```

n= 200, number of events= 97

	coef	exp(coef)	se(coef)	z	Pr(> z)
treated	-0.12927	0.87874	0.28119	-0.460	0.646
linps	0.06495	1.06711	0.15181	0.428	0.669

	exp(coef)	exp(-coef)	lower .95	upper .95
treated	0.87874	1.13811	0.31111	2.41111
linps	1.06711	0.93777	0.71111	1.51111

treated	0.8787	1.1380	0.5064	1.525
linps	1.0671	0.9371	0.7925	1.437

```

Concordance= 0.552 (se = 0.042 )
Rsquare= 0.002 (max possible= 0.913 )
Likelihood ratio test= 0.33 on 2 df, p=0.8478
Wald test               = 0.33 on 2 df, p=0.8476
Score (logrank) test = 0.33 on 2 df, p=0.8475

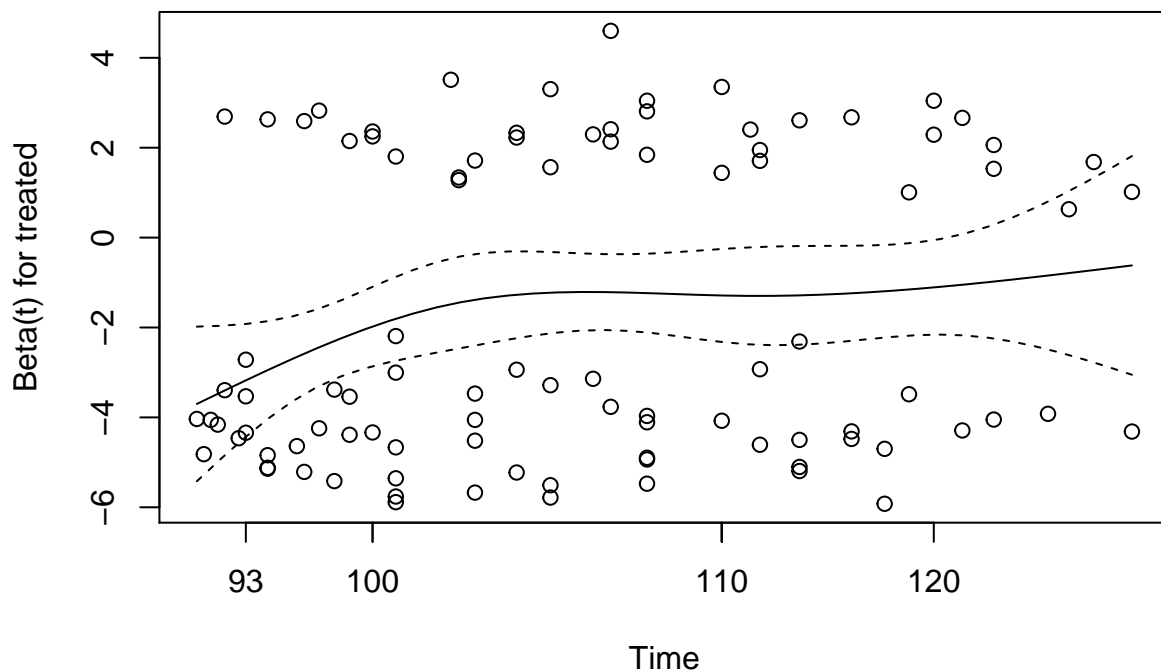
```

The `exp(coef)` output gives the relative hazard of the event comparing treated subjects to control subjects.

And here's the check of the proportional hazards assumption...

```
cox.zph(dr.out3.wt3); plot(cox.zph(dr.out3.wt3), var="treated")
```

	rho	chisq	p
treated	0.1931	5.26	0.0218
linps	-0.0745	1.52	0.2177
GLOBAL	NA	5.91	0.0521



17 Task 12. Results

17.1 Treatment Effect Estimates

We now can build the table of all of the outcome results we've obtained here.

Est. Treatment Effect (95% CI)	Outcome 1 (Cost diff.)	Outcome 2 (Risk diff.)	Outcome 2 (Odds Ratio)	Outcome 3 (Relative Hazard Rate)
No covariate adjustment	15.7	+0.11	1.56	0.86
(unadjusted, ATT)	(12.0, 19.3)	(-0.03, +0.25)	(0.87, 2.82)	(0.57, 1.29)
1:1 PS Match	15.6	+0.11	N/A	N/A
(Match: ATT)	(11.6, 19.6)	(-0.05, +0.25)	N/A	N/A
1:1 PS Match	15.6	N/A	1.64	0.75
(“Regression”, ATT)	(11.5, 19.6)	N/A	(0.77, 3.47)	(0.41, 1.38)
PS Subclassification	7.9	N/A	1.15	0.79
(“Regression”, ATE)	(4.1, 11.7)	N/A	(0.54, 2.44)	(0.49, 1.27)
ATT Weighting	15.2	N/A	1.48	0.82
(ATT)	(10.7, 19.8)	N/A	(0.74, 2.94)	(0.51, 1.32)
ATE Weighting	7.1	N/A	1.29	0.83
(ATE)	(0.1, 14.0)	N/A	(0.62, 2.67)	(0.63, 1.11)
twang ATT weights	15.6	N/A	1.63	0.90
(ATT)	(11.1, 20.0)	N/A	(0.84, 3.14)	(0.52, 1.55)
Direct Adjustment	12.5	N/A	1.35	0.80
(with linsps, ATT)	(8.6, 16.4)	N/A	(0.71, 2.58)	(0.50, 1.26)
Double Robust	14.8	N/A	1.46	0.81
(ATT wts + adj.)	(10.6, 19.0)	N/A	(0.73, 2.94)	(0.50, 1.31)
Double Robust	6.7	N/A	1.28	0.83
(ATE wts + adj.)	(1.6, 11.7)	N/A	(0.61, 2.72)	(0.63, 1.11)
Double Robust	13.0	N/A	1.53	0.88
(twang ATT wts + adj.)	(8.5, 17.5)	N/A	(0.76, 3.07)	(0.50, 1.53)

So, for outcome 1, we have a significant result (indicating higher costs with the treatment) with every approach, and with outcomes 2 and 3, we do not.

17.2 Quality of Balance: Standardized Differences and Variance Ratios

We’re looking at the balance across the following 10 covariates and transformations here: covA, covB, covC, covD, covE, covF[middle], covF[high], A squared, BxC and BxD ...

Approach	Standardized Diffs	Variance Ratios
Most Desirable Values	Between -10 and +10	Between 0.8 and 1.25
No Adjustments	-30 to 63	0.59 to 1.27
Subclassification Quintile 1	-79 to 123	0 to 1.35
Quintile 2	-54 to 47	0.40 to 2.99
Quintile 3	-37 to 23	0.32 to 1.22
Quintile 4	-64 to 32	0.84 to 1.85
Quintile 5	5 to 65	0.80 to 1.32
1:1 Propensity Matching	-13 to 20	0.62 to 1.23
Propensity Weighting, ATT	-6 to 13	0.64 to 1.20
Propensity Weighting, ATE	-14 to 19	0.86 to 1.12

17.3 Quality of Balance: Rubin’s Rules

Approach	Rubin 1	Rubin 2	Rubin 3
“Pass” Range, per Rubin	0 to 50	0.5 to 2.0	0.5 to 2.0
No Adjustments	88	0.58	0.59 to 1.28
Subclassification: Quintile 1	61	0.48	0.02 to 1.32
Quintile 2	30	1.20	0.36 to 3.19
Quintile 3	80	0.79	0.29 to 1.26
Quintile 4	28	0.80	0.83 to 1.91
Quintile 5	36	2.49	0.67 to 1.42
1:1 Propensity Matching	37	1.42	0.56 to 1.28
Propensity Weighting, ATT	6.2	1.20	Not evaluated
Propensity Weighting, ATE	5.0	0.90	Not evaluated

Clearly, the matching and propensity weighting show improvement over the initial (no adjustments) results, although neither is completely satisfactory in terms of all covariates. In practice, I would be comfortable with either a 1:1 match or a weighting approach, I think. It isn’t likely that the subclassification will get us anywhere useful in terms of balance. Rubin’s Rule 3 could also be applied after weighting on the propensity score.

18 What is a Sensitivity Analysis for Matched Samples?

We’ll study a formal sensitivity analysis approach for **matched** samples. Note well that this specific approach is appropriate only when we have

1. a statistically significant conclusion
2. from a matched samples analysis using the propensity score.

18.1 Goal of a Formal Sensitivity Analysis for Matched Samples

To replace a general qualitative statement that applies in all observational studies, like ...

the association we observe between treatment and outcome does not imply causation

or

hidden biases can explain observed associations

... with a quantitative statement that is specific to what is observed in a particular study, such as ...

to explain the association seen in a particular study, one would need a hidden bias of a particular magnitude.

If the association is strong, the hidden bias needed to explain it would be large.

- If a study is free of hidden bias (main example: a carefully randomized trial), this means that any two units (patients, subjects, whatever) that appear similar in terms of their observed covariates actually have the same chance of assignment to treatment.
- There is *hidden bias* if two units with the same observed covariates have different chances of receiving the treatment.

A **sensitivity analysis** asks: How would inferences about treatment effects be altered by hidden biases of various magnitudes? How large would these differences have to be to alter the qualitative conclusions of the study?

The methods for building such sensitivity analyses are largely due to Paul Rosenbaum, and as a result the methods are sometimes referred to as **Rosenbaum bounds**.

18.2 The Sensitivity Parameter, Γ

Suppose we have two units (subjects, patients), say, j and k , with the same observed covariate values \mathbf{x} but different probabilities p of treatment assignment (possibly due to some unobserved covariate), so that $\mathbf{x}_j = \mathbf{x}_k$ but that possibly $p_j \neq p_k$.

Units j and k might be *matched* to form a matched pair in our attempt to control overt bias due to the covariates \mathbf{x} .

- The odds that units j and k receive the treatment are, respectively, $\frac{p_j}{1-p_j}$ and $\frac{p_k}{1-p_k}$, and the odds ratio is thus the ratio of these odds.

Imagine that we knew that this odds ratio for units with the same \mathbf{x} was at most some number Γ , so that $\Gamma \geq 1$. That is,

$$\frac{1}{\Gamma} \leq \frac{p_j(1-p_k)}{p_k(1-p_j)} \leq \Gamma$$

We call Γ the **sensitivity parameter**, and it is the basis for our sensitivity analyses.

- If $\Gamma = 1$, then $p_j = p_k$ whenever $\mathbf{x}_j = \mathbf{x}_k$, so the study would be free of hidden bias, and standard statistical methods designed for randomized trials would apply.

If $\Gamma = 2$, then two units who appear similar in that they have the same set of observed covariates \mathbf{x} , could differ in their odds of receiving the treatment by as much as a factor of 2, so that one could be twice as likely as the other to receive the treatment.

So Γ is a value between 1 and ∞ where the size of Γ indicates the degree of a departure from a study free of hidden bias.

18.3 Interpreting the Sensitivity Parameter, Γ

Again, Γ is a measure of the degree of departure from a study that is free of hidden bias.

A sensitivity analysis will consider possible values of Γ and show how the inference for our outcomes might change under different levels of hidden bias, as indexed by Γ .

- A study is *sensitive* if values of Γ close to 1 could lead to inferences that are very different from those obtained assuming the study is free of hidden bias.
- A study is *insensitive* (a good thing here) if extreme values of Γ are required to alter the inference.

When we perform this sort of sensitivity analysis, we will specify different levels of hidden bias (different Γ values) and see how large a Γ we can have while still retaining the fundamental conclusions of the matched outcomes analysis.

19 Task 13. Sensitivity Analysis for Matched Samples, Outcome 1, using `rbounds`

In our matched sample analysis, for outcome 1 (cost) in the toy example, we saw a statistically significant result. A formal *sensitivity analysis* is called for, as a result, and we will accomplish one for this quantitative outcome, using the `rbounds` package.

The `rbounds` package is designed to work with the output from `Matching`, and can calculate Rosenbaum sensitivity bounds for the treatment effect, which help us understand the impact of hidden bias needed to invalidate our significant conclusions from the matched samples analysis.

19.1 Rosenbaum Bounds for the Wilcoxon Signed Rank test (Quantitative outcome)

We have already used the `Match` function from the `Matching` package to develop a matched sample. Given this, we need only run the `psens` function from the `rbounds` package to obtain sensitivity results.

```
X <- toy$linps ## matching on the linear propensity score
Tr <- as.logical(toy$treated)
Y <- toy$out1.cost
match1 <- Match(Tr=Tr, X=X, Y = Y, M = 1, replace=FALSE, ties=FALSE)
summary(match1)
```

```
Estimate... 15.557
SE..... 2.0397
T-stat..... 7.6273
p.val..... 2.3981e-14
```

```
Original number of observations..... 200
Original number of treated obs..... 70
Matched number of observations..... 70
Matched number of observations (unweighted). 70
```

```
psens(match1, Gamma = 5, GammaInc = 0.25)
```

Rosenbaum Sensitivity Test for Wilcoxon Signed Rank P-Value

```
Unconfounded estimate .... 0
```

Gamma	Lower bound	Upper bound
1.00	0	0.0000
1.25	0	0.0000
1.50	0	0.0000
1.75	0	0.0001
2.00	0	0.0003
2.25	0	0.0008
2.50	0	0.0022
2.75	0	0.0046
3.00	0	0.0087
3.25	0	0.0148
3.50	0	0.0233
3.75	0	0.0343
4.00	0	0.0480
4.25	0	0.0644
4.50	0	0.0833
4.75	0	0.1046
5.00	0	0.1280

Note: Gamma is Odds of Differential Assignment To Treatment Due to Unobserved Factors

If the study were free of hidden bias, that is, if $\Gamma = 1$, then there would be **strong** evidence that the treated patients had higher costs, and the specific Wilcoxon signed rank test we're looking at here shows a p value < 0.0001 . The sensitivity analysis we'll conduct now asks how this conclusion might be changed by hidden

biases of various magnitudes, depending on the significance level we plan to use in our test.

19.2 Specifying The Threshold Γ value

From the output above, find the Γ value where the upper bound for our p value slips from “statistically significant” to “not significant” territory.

- We’re doing a two-tailed test, with a 95% confidence level, so the Γ statistic for this situation is between 3.50 and 3.75, since that is the point where the upper bound for the p value crosses the threshold of $\alpha/2 = 0.025$.

So this study’s conclusion (that treated patients had significantly higher costs) would still hold even in the face of a hidden bias with $\Gamma = 3.5$, but not with $\Gamma = 3.75$.

The tipping point for the sensitivity parameter is a little over 3.5. To explain away the observed association between treatment and this outcome (cost), a hidden bias or unobserved covariate would need to increase the odds of treatment by more than a factor of $\Gamma = 3.5$.

Returning to the output:

- If instead we were doing a one-tailed test, with a 95% confidence level, then the Γ statistic for this situation is between 4 and 4.25, since that is the point where the upper bound for the p value crosses $\alpha = 0.05$.
- And if instead we were doing a one-tailed test with a 90% confidence level, then the Γ statistic would be between 4.75 and 5.0, since that is where the upper bound for the p value crosses $\alpha = 0.10$.

19.3 Interpreting Γ appropriately

Γ tells you only *how big a bias is needed to change the answer*. By itself, it says NOTHING about the likelihood that a bias of that size is present in your study, except that, of course, smaller biases hide more effectively than large ones, on average.

- In some settings, we’ll think of Γ in terms of small (< 1.5), modest (1.5 - 2.5), moderate (2.5 - 4) and large (> 4) hidden bias requirements. But these are completely arbitrary distinctions, and I can provide no good argument for their use.

The **only** defense against hidden bias affecting your conclusions is to try to reduce the potential for hidden bias in the first place. We work on this via careful design of observational studies, especially by including as many different dimensions of the selection problem as possible in your propensity model.

19.4 An Alternate Approach - the Hodges-Lehman estimate

```
hlsens(match1)
```

Rosenbaum Sensitivity Test for Hodges-Lehmann Point Estimate

Unconfounded estimate 16.5

Gamma	Lower bound	Upper bound
1	16.5	16.5
2	11.0	21.6
3	7.5	24.1
4	5.5	26.1
5	3.5	27.6

Note: Gamma is Odds of Differential Assignment To Treatment Due to Unobserved Factors

If the Γ value is 3.0, then this implies that the Hodges-Lehmann estimate might be as low as 7.5 or as high as 24.1 (it is 16.5 in the absence of hidden bias in this case - when $\Gamma = 0$.)

19.5 What about other types of outcomes?

The `rbounds` package can evaluate binary outcomes using the `binarysens` and `Fishersens` functions.

Survival outcomes can be assessed, too, but not, I believe, using `rbounds` unless there is no censoring. Some time back, I built a spreadsheet for this task, which I'll be happy to share.

19.6 What about when we match 1:2 or 1:3 instead of 1:1?

The `mcontrol` function in the `rbounds` package can be helpful in such a setting.

20 Wrapup

If you run this script, you'll wind up with a version of the `toy` tibble that contains 200 observations on 28 variables, along with a `codebook` list.

You'll also have two new functions, called `szd` and `rubin3`, that, with some modification, may be useful elsewhere.

To drop everything else in the global environment created by this Markdown file, run the code that follows.

```
rm(list = c("adj.m.out1", "adj.m.out2", "adj.m.out3",
  "adj.reg.out1", "adj.reg.out2", "adj.reg.out3",
  "adj.s.out3", "adjout1.wt1", "adjout1.wt2",
  "adjout2.wt1", "adjout2.wt2", "adjout3.wt1",
  "adjout3.wt2", "bal.after.wts1", "bal.after.wts2",
  "bal.before.wts1", "bal.before.wts2", "bal.wts1",
  "bal.wts2", "balance.ate.weights", "balance.att.weights",
  "cov.sub", "covlist", "covnames", "covs",
  "d.all", "d.q1", "d.q2", "d.q3", "d.q4", "d.q5",
  "dr.out1.wt1", "dr.out1.wt2", "dr.out2.wt1",
  "dr.out2.wt2", "dr.out3.wt1", "dr.out3.wt2", "est.st",
  "factorlist", "high", "i", "low", "match1", "matches",
  "mb1", "mixedmodel.out1", "post.szd", "post.vratio",
  "pre.szd", "pre.vratio", "psmodel", "quin1",
  "quin1.out1", "quin1.out2", "quin2", "quin2.out1",
  "quin2.out2", "quin3", "quin3.out1", "quin3.out2",
  "quin4", "quin4.out1", "quin4.out2", "quin5",
  "quin5.out1", "quin5.out2", "rubin1.match", "rubin1.q1",
  "rubin1.q2", "rubin1.q3", "rubin1.q4", "rubin1.q5",
  "rubin1.sub", "rubin1.unadj", "rubin2.match",
  "rubin2.q1", "rubin2.q2", "rubin2.q3", "rubin2.q4",
  "rubin2.q5", "rubin2.sub", "rubin2.unadj", "rubin3.both",
  "rubin3.matched", "rubin3.q1", "rubin3.q2", "rubin3.q3",
```

```
"rubin3.q4", "rubin3.q5", "rubin3.unadj", "se.q1",
"se.q2", "se.q3", "se.q4", "se.q5", "se.st", "temp",
"temp.result1", "temp.result2", "temp.sort",
"toy.matchedsample", "toy.rubin3", "toy.szd",
"toy_df", "toywt1.design", "toywt2.design", "Tr",
"unadj.out1", "unadj.out2", "unadj.out3", "varlist", "X", "Y"))
```

20.1 Session status

```
devtools::session_info()
```

```
Session info -----
```

```
setting  value
version  R version 3.4.3 (2017-11-30)
system   x86_64, mingw32
ui        RTerm
language (EN)
collate   English_United_States.1252
tz        America/New_York
date      2018-01-08
```

```
Packages -----
```

package	* version	date	source
abind	1.4-5	2016-07-21	CRAN (R 3.4.0)
acepack	1.4.1	2016-10-29	CRAN (R 3.4.0)
arm	* 1.9-3	2016-11-27	CRAN (R 3.4.2)
assertthat	0.2.0	2017-04-11	CRAN (R 3.4.0)
backports	1.1.2	2017-12-13	CRAN (R 3.4.3)
base	* 3.4.3	2017-11-30	local
base64enc	0.1-3	2015-07-28	CRAN (R 3.4.0)
bindr	0.1	2016-11-13	CRAN (R 3.4.0)
bindrcpp	* 0.2	2017-06-17	CRAN (R 3.4.0)
broom	* 0.4.3	2017-11-20	CRAN (R 3.4.2)
cellranger	1.1.0	2016-07-27	CRAN (R 3.4.0)
checkmate	1.8.5	2017-10-24	CRAN (R 3.4.2)
class	7.3-14	2015-08-30	CRAN (R 3.4.3)
cli	1.0.0	2017-11-05	CRAN (R 3.4.2)
cluster	2.0.6	2017-03-10	CRAN (R 3.4.3)
cmprsk	2.2-7	2014-06-17	CRAN (R 3.4.0)
cobalt	* 3.1.0	2017-11-12	CRAN (R 3.4.2)
coda	0.19-1	2016-12-08	CRAN (R 3.4.0)
colorspace	1.3-2	2016-12-14	CRAN (R 3.4.3)
compiler	3.4.3	2017-11-30	local
crayon	1.3.4	2017-09-16	CRAN (R 3.4.1)
data.table	1.10.4-3	2017-10-27	CRAN (R 3.4.2)
datasets	* 3.4.3	2017-11-30	local
devtools	* 1.13.4	2017-11-09	CRAN (R 3.4.3)
digest	0.6.13	2017-12-14	CRAN (R 3.4.3)
dplyr	* 0.7.4	2017-09-28	CRAN (R 3.4.2)
e1071	1.6-8	2017-02-02	CRAN (R 3.4.0)
Epi	* 2.19	2017-08-09	CRAN (R 3.4.2)
etm	0.6-2	2014-12-09	CRAN (R 3.4.0)

evaluate	0.10.1	2017-06-24	CRAN	(R 3.4.0)
forcats	* 0.2.0	2017-01-23	CRAN	(R 3.4.2)
foreign	0.8-69	2017-06-21	CRAN	(R 3.4.1)
Formula	* 1.2-2	2017-07-10	CRAN	(R 3.4.1)
gbm	* 2.1.3	2017-03-21	CRAN	(R 3.4.0)
ggplot2	* 2.2.1	2016-12-30	CRAN	(R 3.4.0)
glue	1.2.0	2017-10-29	CRAN	(R 3.4.2)
graphics	* 3.4.3	2017-11-30	local	
grDevices	* 3.4.3	2017-11-30	local	
grid	* 3.4.3	2017-11-30	local	
gridExtra	2.3	2017-09-09	CRAN	(R 3.4.2)
gtable	0.2.0	2016-02-26	CRAN	(R 3.4.0)
haven	1.1.0	2017-07-09	CRAN	(R 3.4.1)
Hmisc	* 4.1-1	2018-01-03	CRAN	(R 3.4.3)
hms	0.4.0	2017-11-23	CRAN	(R 3.4.2)
htmlTable	1.11.1	2017-12-27	CRAN	(R 3.4.3)
htmltools	0.3.6	2017-04-28	CRAN	(R 3.4.0)
htmlwidgets	0.9	2017-07-10	CRAN	(R 3.4.1)
httr	1.3.1	2017-08-20	CRAN	(R 3.4.1)
jsonlite	1.5	2017-06-01	CRAN	(R 3.4.0)
knitr	1.18	2017-12-27	CRAN	(R 3.4.3)
labeling	0.3	2014-08-23	CRAN	(R 3.4.0)
lattice	* 0.20-35	2017-03-25	CRAN	(R 3.4.3)
latticeExtra	* 0.6-28	2016-02-09	CRAN	(R 3.4.0)
lazyeval	0.2.1	2017-10-29	CRAN	(R 3.4.2)
lme4	* 1.1-15	2017-12-21	CRAN	(R 3.4.3)
lubridate	1.7.1	2017-11-03	CRAN	(R 3.4.2)
magrittr	1.5	2014-11-22	CRAN	(R 3.4.1)
MASS	* 7.3-48	2017-12-25	CRAN	(R 3.4.3)
Matching	* 4.9-2	2015-12-25	CRAN	(R 3.4.0)
Matrix	* 1.2-12	2017-11-20	CRAN	(R 3.4.3)
memoise	1.1.0	2017-04-21	CRAN	(R 3.4.0)
methods	* 3.4.3	2017-11-30	local	
minqa	1.2.4	2014-10-09	CRAN	(R 3.4.0)
mnormt	1.5-5	2016-10-15	CRAN	(R 3.4.0)
modelr	0.1.1	2017-07-24	CRAN	(R 3.4.1)
munsell	0.4.3	2016-02-13	CRAN	(R 3.4.0)
nlme	3.1-131	2017-02-06	CRAN	(R 3.4.3)
nloptr	1.0.4	2014-08-04	CRAN	(R 3.4.0)
nnet	7.3-12	2016-02-02	CRAN	(R 3.4.3)
numDeriv	2016.8-1	2016-08-27	CRAN	(R 3.4.0)
pander	* 0.6.1	2017-08-06	CRAN	(R 3.4.2)
parallel	* 3.4.3	2017-11-30	local	
pillar	1.0.1	2017-11-27	CRAN	(R 3.4.3)
pkgconfig	2.0.1	2017-03-21	CRAN	(R 3.4.0)
plyr	1.8.4	2016-06-08	CRAN	(R 3.4.0)
psych	1.7.8	2017-09-09	CRAN	(R 3.4.2)
purrr	* 0.2.4	2017-10-18	CRAN	(R 3.4.2)
R6	2.2.2	2017-06-17	CRAN	(R 3.4.0)
rbinds	* 2.1	2014-12-08	CRAN	(R 3.4.0)
RColorBrewer	* 1.1-2	2014-12-07	CRAN	(R 3.4.0)
Rcpp	0.12.14	2017-11-23	CRAN	(R 3.4.2)
readr	* 1.1.1	2017-05-16	CRAN	(R 3.4.0)
readxl	1.0.0	2017-04-18	CRAN	(R 3.4.0)

reshape2	1.4.3	2017-12-11	CRAN (R 3.4.3)
rlang	0.1.6	2017-12-21	CRAN (R 3.4.3)
rmarkdown	1.8	2017-11-17	CRAN (R 3.4.2)
rpart	4.1-11	2017-03-13	CRAN (R 3.4.3)
rprojroot	1.3-2	2018-01-03	CRAN (R 3.4.3)
rstudioapi	0.7	2017-09-07	CRAN (R 3.4.1)
rvest	0.3.2	2016-06-17	CRAN (R 3.4.0)
scales	0.5.0	2017-08-24	CRAN (R 3.4.1)
splines	* 3.4.3	2017-11-30	local
stats	* 3.4.3	2017-11-30	local
stringi	1.1.6	2017-11-17	CRAN (R 3.4.2)
stringr	* 1.2.0	2017-02-18	CRAN (R 3.4.0)
survey	* 3.32-1	2017-06-22	CRAN (R 3.4.0)
survival	* 2.41-3	2017-04-04	CRAN (R 3.4.2)
tableone	* 0.8.1	2017-06-17	CRAN (R 3.4.2)
tibble	* 1.4.1	2017-12-25	CRAN (R 3.4.3)
tidyr	* 0.7.2	2017-10-16	CRAN (R 3.4.2)
tidyselect	0.2.3	2017-11-06	CRAN (R 3.4.2)
tidyverse	* 1.2.1	2017-11-14	CRAN (R 3.4.3)
tools	3.4.3	2017-11-30	local
twang	* 1.5	2017-07-02	CRAN (R 3.4.1)
utf8	1.1.3	2018-01-03	CRAN (R 3.4.3)
utils	* 3.4.3	2017-11-30	local
withr	2.1.1	2017-12-19	CRAN (R 3.4.3)
xml2	1.1.1	2017-01-24	CRAN (R 3.4.0)
xtable	* 1.8-2	2016-02-05	CRAN (R 3.4.0)
yaml	2.1.16	2017-12-12	CRAN (R 3.4.3)
zoo	1.8-0	2017-04-12	CRAN (R 3.4.3)