Thomas E. Love, Ph.D.
ICHPS 2018 Workshop

Thomas.Love@case.edu
2018-01-10

The **TOY** Example

The Data Set is 100% fictional, and is available as **toy.csv** on the website. It contains data on 200 patients (70 treated – subjects 131-200 and 130 controls – subjects 1-130) on treatment status, six covariates, and three outcomes, with no missing observations anywhere.

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.

Some of the Data for the **toy** *Example*

subject	treated	covA	covB	covC	covD	covE	covF	out1.cost	out2.event	out3.time
1	0	0.12	1	7.6	6.7	8	3-High	46	Yes	99
2	0	0.05	0	11.2	11.7	7	1-Low	44	Yes	112
3	0	0.68	0	7.7	9.3	12	2-Middle	47	No	89
195 observations skipped										
199	1	0.41	0	10.5	9	14	3-High	38	Yes	106
200	1	0.66	1	7.9	9	13	2-Middle	68	Yes	105

A Basic Codebook for the **toy** *Example*

Variable	Type of Variable	Description
subject	Subject ID #	1-130 = control subjects; 131-200 = treated subjects
treated	Factor with two levels, 0 and 1	Coded as 0 = control subject; 1 = treated subject
covA	Continuous (2 decimal places)	Permissible values range from 0-1
covB	Factor with two levels, 0 and 1	0 = does not have factor B; 1 = has factor B
covC	Continuous (1 decimal place)	Permissible values range from 3 to 20
covD	Continuous (1 decimal place)	Permissible values range from 3 to 20
covE	Continuous (0 decimal places)	Permissible values range from 3 to 20
covF	Three-level ordinal factor	Coded as 1-Low, 2-Middle, or 3-High
out1.cost	Continuous outcome	Cost – values typically between 20 and 80
out2.event	Binary outcome – did "event" occur?	Whether a (bad) event occurs – coded as Yes or No
out3.time	Time to event outcome – time before "event" is observed or patient exits study (is censored.)	Observation Time range is 75 to 156 weeks. Patients with out2.event = No were censored. out2.event = Yes indicates an observed event.

Note: 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.

The Questions We'll Tackle in the **toy** Example

a) 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)?
b) Assume that theory suggests that the **square** of **covA**, as well as the **interactions** of **covB** with **covC** and 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.
c) Use 1:1 greedy matching to match all 70 treated patients to control patients 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.
d) Now, build a new data frame containing the propensity-matched sample, and use it 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 patient) on the outcome. Compare your results to the automatic versions reported by the Matching library when you include the outcome in the matching process.
e) Now, instead of matching, instead subclassify the patients 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?
f) Regardless of your answer to the previous question, use the propensity score quintile stratification approach to find a point estimate (and 95% confidence interval) for the effect of the treatment on each outcome.
g) 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?
h) 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.
i) 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.
j) Now, try a double robust approach. Weight, then adjust for linear propensity score.
k) Compare your conclusions about the average causal effect obtained [1] without propensity adjustment, [2] after propensity matching, [3] after propensity score subclassification, [4] after propensity score weighting, [5] after adjusting for the propensity score directly, and [6] after weighting then adjusting for the PS, to each other. What happens and why? Which of these methods seems most appropriate given the information available?

Table of Contents

The R Script; Detailed Description of Code & Results

The complete script is available at the website as **toy_script.R** - this script makes use of the following libraries:

Epi, survival, sm, Matching, lme4, reshape2, epibasix, Hmisc, lattice, and **survey**

The **toy.csv** data set contains 200 observations and 11 variables, as noted previously.

subject	treated	covA	covB	covC	covD	covE	covF	out1.cost	out2.event	out3.time
1	0	0.12	1	7.6	6.7	8	3-High	46	Yes	99
2	0	0.05	0	11.2	11.7	7	1-Low	44	Yes	112
3	0	0.68	0	7.7	9.3	12	2-Middle	47	No	89
195 observations skipped										
199	1	0.41	0	10.5	9	14	3-High	38	Yes	106
200	1	0.66	1	7.9	9	13	2-Middle	68	Yes	105

Preliminary Data Cleanup

My first step will be to summarize the data after import, and evaluate what needs to be done.

```
> 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
```

Range Checks for Continuous Variables

Checking and cleaning the **continuous** 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 continuous variables have values that fall within the "permissible" range described by my codebook on page 1, 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.**

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 continuous variables.

Re-expressing Binary Variables

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:

a) 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")
b) as a text factor – with the levels arranged so that "having the characteristic" precedes "not having the characteristic" in R when you create a table.

Here, for **treated** and again for **covB**, we already have the numeric indicator as part of the data import (you can check this by looking at the summary output – if you get quartiles and a median, and the minimum and maximum are 0 and 1, then R thinks it's a number. So, we need only to create the factor, with the levels ordered as 1 then 0, and labels that are meaningful.

```
> toy$treated.f <- factor(toy$treated,
     levels=c(1,0),labels=c("Treated","Control"))
> toy$covB.f <- factor(toy$covB, levels=c(1,0), labels=c("Has
     B", "No 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 Occurred", "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 # the -1 at the end
    changes the default 1/2 code to 0/1
```

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.

```
> ## Sanity Checks
> table(toy$treated.f, toy$treated)
           0   1
  Treated  0  70
  Control 130   0

> table(toy$covB.f, toy$covB)
         0   1
  Has B  0  74
  No B 126   0

> table(toy$out2.f, toy$out2.event)
                  No Yes
  Event Occurred   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 Occurred 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" first.
- **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.

Dealing with More than Two Categories in a Variable

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

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

From the summary output, we can see that we're all set for version A of **covF** is currently 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.

> *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"))
```

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.

```
> 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, again, some sanity checks on these indicator variables, for instance, looking at the Low group. I'll spare you the output for the Middle and High indicator variables, which are also OK.

```
> table(toy$covF, toy$covF.Low)
           0  1
  1-Low    0 67
  2-Middle 84  0
  3-High   49  0
```

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 transformation in our data set for modeling and summarizing. You need to 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
```

At this point, if you've followed the script precisely, your version of the **toy** data set should contain 200 observations on 21 variables, specifically:

```
> names(toy)
 [1] "subject"      "treated"      "covA"         "covB"
 [5] "covC"         "covD"         "covE"         "covF"
 [9] "out1.cost"    "out2.event"   "out3.time"    "treated.f"
[13] "covB.f"       "out2.f"       "out2"         "covF.Low"
[17] "covF.Middle"  "covF.High"    "Asqr"         "BC"
[21] "BD"

> dim(toy)
[1] 200  21
```

Question A. Ignoring covariates, what is the estimated effect of treatment vs control on…

…Outcome 1 (a continuous outcome)

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

```
> by(toy$out1.cost, toy$treated.f, summary)
toy$treated.f: Treated
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  24.00   50.25   62.00   60.61   75.75   80.00
------------------------------------------------
toy$treated.f: Control
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  24.00   38.00   47.00   44.94   51.00   78.00
```

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)

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
```

Our unadjusted treatment effect estimate is an increase of **15.7** in cost, with 95% CI (**12.0, 19.3**)

…Outcome 2 (a binary outcome)

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

```
> table(toy$treated.f, toy$out2.f)
           Event Occurred 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…

```
> library(Epi)
> twoby2(table(toy$treated.f, toy$out2.f))

2 by 2 table analysis:
------------------------------------------------------
Outcome    : Event Occurred
Comparing : Treated vs. Control

           Event
         Occurred No Event P(Event Occurred)       95% Conf. Int.
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, as highlighted.

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 effect statistically significant at a 95% confidence level.

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. When modeling, it's better to 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)) # produced 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
```

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.

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.

…Outcome 3 (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 patients in the data set. For the other 103 patients 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 are the treated patients – the third patient 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
[13] 121+  97  137+ 103  110  106+ 108  136+  96  125+ 118+  99+
[25] 101  109  128  111+ 120+ 118  115  129+ 118+ 106  106+ 118+
[37] 121  120  107  119+ 105  112  106+ 132+ 126+ 130+ 115+ 108
[49]  99  122  110  102  114+ 120+ 100  108  120  118+ 117+ 123+
[61] 109  122  126+ 126+ 118+ 125+ 103   94  106  105
```

To see the controls, use…

```
> Surv(toy$out3.time, toy$out2.event=="Yes")[toy$treated==0]
```

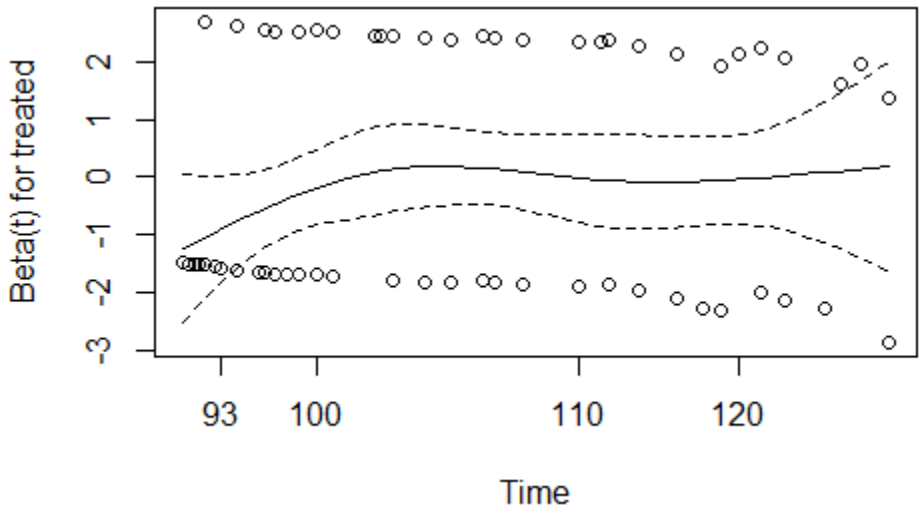
To deal with this right censoring, we'll use the survival library to fit a simple unadjusted Cox proportional hazards model to assess the relative hazard of having the event at a particular time point among treated patients as compared to controls.

```
> library(survival)
> unadj.out3 <- coxph(Surv(out3.time, out2.event=="Yes") ~
    treated, data=toy)
> summary(unadj.out3)
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 highlighted **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**).

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.

Unadjusted Estimates of Treatment Effect on Outcomes

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

Causal Effect of Treatment Estimates, with (95% CI)	**Outcome 1** Cost difference	**Outcome 2** Risk Difference	**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)

Question B. Fit the propensity score model, then plot the PS-treatment relationship…

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

My very first step, will be to save the raw and linear PS values to the toy example.

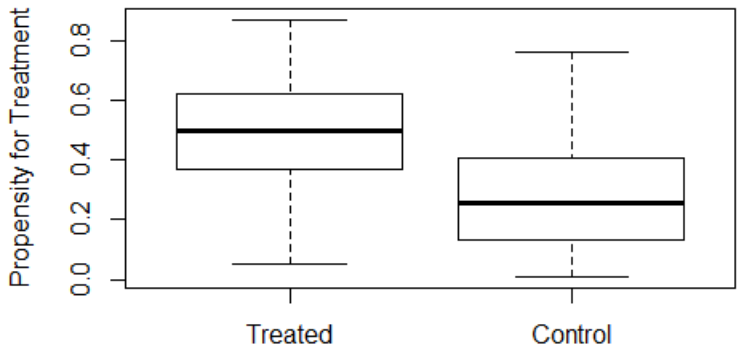
```
> toy$ps <- psmodel$fitted
> toy$linps <- psmodel$linear.predictors
```

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

```
> by(toy$ps, toy$treated.f, summary)

toy$treated.f: Treated
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.05289 0.36880 0.49580 0.47610 0.61530 0.86990
------------------------------------------------------------
toy$treated.f: Control
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.007769 0.132200 0.255100 0.282100 0.406400 0.762500
```

OK – it looks like the patients who were actually treated have somewhat higher propensity for treatment than do the controls. So that makes sense.

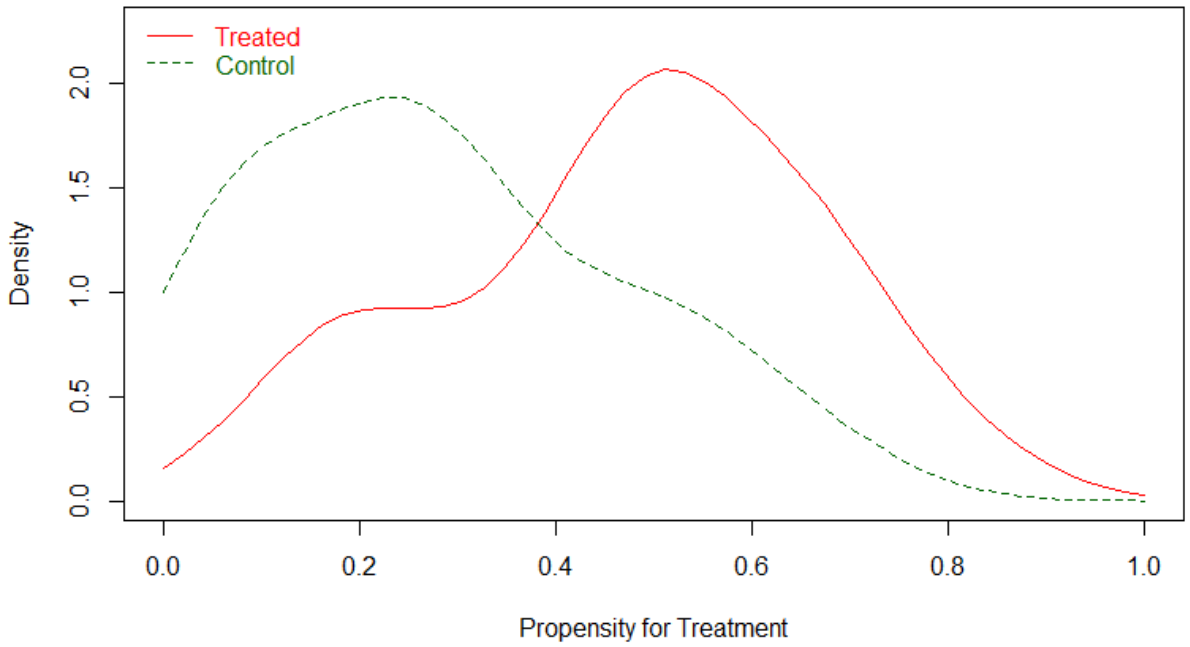
```
> plot(toy$ps ~ toy$treated.f, ylab="Propensity for Treatment",
      xlab="")
```



We could tweak the boxplot further, but 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.

Note that you need to use the **factor** version of the treated variable (i.e. **treated.f**) here in both the plot call and the legend in order to get the plot to label things correctly.

```
> library(sm)
> sm.density.compare(psmodel$fitted, toy$treated.f,
      xlab="Propensity for Treatment", main="Propensity Score
      Comparison", xlim=c(0,1), col=c("red", "dark green"),
      lty=1)
> legend("topleft", legend=levels(toy$treated.f), lty=c(1,2),
      lwd=1, col=c("red","dark green"), text.col=c("red","dark
      green"), bty="n")
```



Note that, here, I have changed the script from the prior versions of this example to call for the legend to be placed on the plot at the top left.

If, instead, you make use of the locator function (an approach which is commented out in the newest version of the toy script) this means that you must click on the plot to place the legend after running this command or R will stop executing commands.

Interlude: Rubin's Rules to Check Overlap Before Propensity Adjustment

In his 2001 article 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[1], 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.

Rubin's Rule 1: First, 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 little script 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
```

Again, what this does is calculate the (absolute value of the) standardized difference of the linear propensity score comparing treated patients to control patients. 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, stratify, weight or directly adjust for propensity here.

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

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.

Rubin's Rule 2: Second, 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 between 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 script 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
```

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

Rubin's Rule 3: Third, we calculate 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 general 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$linps))[data$treated==1])
+     den <- var(resid(lm(cov ~ data$linps))[data$treated==0])
+     res[i] <- round(num/den, 3)
+   }
+   names(res) <- names(covlist)
+   print(res)
+ }
```

Now, then, applying the rule to our sample prior to propensity score adjustment, we get the following result:

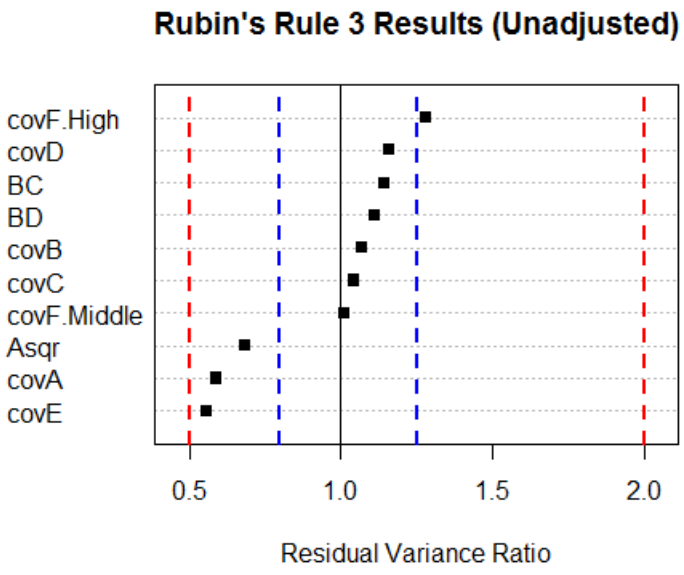
```
> rubin3.unadj <- rubin3(data=toy, covlist=toy[c("covA", "covB",
     "covC", "covD", "covE", "covF.Middle", "covF.High",
     "Asqr","BC", "BD")])

 covA  covB  covC  covD  covE covF.Middle covF.High  Asqr    BC
0.591 1.072 1.043 1.160 0.558       1.010     1.281 0.684 1.144
   BD
1.114
```

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.

Building a Dotplot of the Rubin's Rule 3 results…

```
> d0 <- rubin3(data=toy, covlist=toy[c("covA", "covB", "covC",
     "covD", "covE", "covF.Middle", "covF.High", "Asqr","BC",
     "BD")])
> d <- sort(d0)
> low <- min(min(d), 0.45)
> high <- max(max(d), 2.05)

> dotchart(d, pch=15, col="black", main="Rubin's Rule 3 Results
     (Unadjusted)", xlab="Residual Variance Ratio", xlim=c(low,
     high))
>  abline(v=1, lty=1)
>  abline(v=0.8, lty=2,
     lwd=2, col="blue")
>  abline(v=1.25, lty=2,
     lwd=2, col="blue")
>  abline(v=0.5, lty=2,
     lwd=2, col="red")
>  abline(v=2, lty=2,
     lwd=2, col="red")
```

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



Question C. Use 1:1 greedy matching on the linear PS, then check balance with plots…

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 library. We'll wind up with a match including 70 treated and 70 control patients.

```
> library(Matching)
> X <- toy$linps
> Tr <- as.logical(toy$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
```

Next, we'll assess the balance imposed by this greedy match on our covariates, and their transformations (A^2 and B*C and B*D) as well as the raw and linear propensity scores.

```
> 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

var ratio (Tr/Co).....    0.58886           0.83347
T-test p-value........ 0.00016497           0.29257
KS Bootstrap p-value.. < 2.22e-16             0.086
KS Naive p-value......  0.0013313           0.12159
KS Statistic..........    0.28352               0.2
```

Output edited severely…

```
***** (V12) psmodel$linear.predictors *****
                        Before Matching        After Matching
mean treatment........   -0.13912           -0.13912
mean control.........    -1.2152            -0.45294
std mean diff.........     117.3              34.211

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              0.026
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 psmodel$fitted
psmodel$linear.predictors  Number(s): 1 8 11 12

After Matching Minimum p.value: 2.591e-06
Variable Name(s): psmodel$fitted  Number(s): 11
```

Here's a table of results I might actually review from the full output. I built this by painstaking copying and pasting – just the sort of old-world craftsmanship that's hard to find nowadays.

Table of Key Summaries from the MatchBalance Output

Before Match	**Raw PS**: 0.48 for 70 treated patients, 0.28 in 130 controls
After Match	**Raw PS**: 0.48 for 70 treated patients, 0.40 in 70 controls

Covariate	Standardized Difference (%)		Variance Ratio		P value (K-S bootstrap or T)	
	Before Match	After Match	Before Match	After Match	Before Match	After Match
(1) covA	63	16	.59	.83	0	.09
(2) covB	35	20	1.18	1.05	.02	.22
(3) covC	-7	-5	.99	.99	.88	.82
(4) covD	19	-0.3	1.12	1.23	.51	.97
(5) covE	-30	-13	.60	.62	.23	.50
(6) covF-Middle	3	9	1.02	1.03	.86	.59
(7) covF-High	18	6	1.25	1.06	.20	.66
(8) A^2	46	13	0.73	0.78	0	.09
(9) BC	35	20	1.27	1.14	.02	.32
(10) BD	35	20	1.25	1.11	.03	.59
(11 raw PS	101	38	1.05	1.41	0	.03
(12) linear PS	117	34	0.58	1.42	0	.03

To plot this stuff so we can look at it more carefully, we'll need to name 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")
```

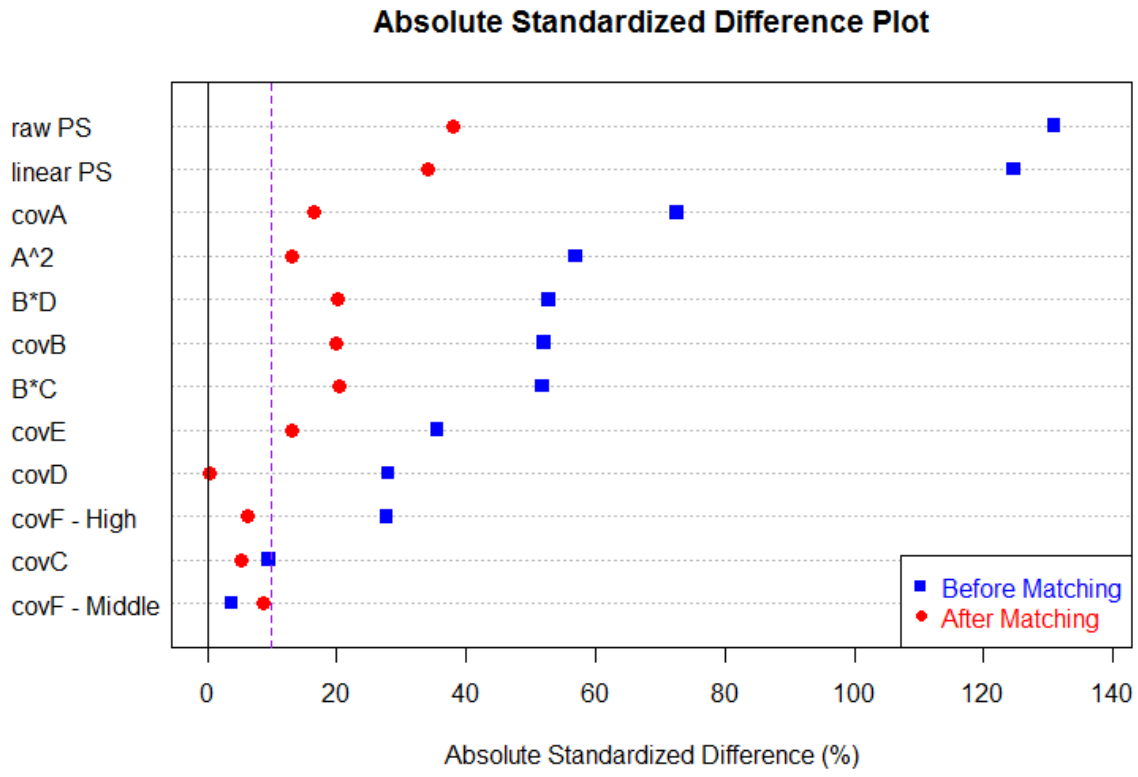
Extracting, Tabulating and Plotting Standardized Differences

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) then create a table and, finally and more usefully, plot them…

```
> 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
+ }

> temp <- data.frame(pre.szd, post.szd, row.names=covnames)
> print(temp, digits=3)
              pre.szd post.szd
covA            72.57   16.476
covB            52.02   19.865
covC            -9.57   -5.316
covD            27.95   -0.381
covE           -35.54  -13.017
covF - Middle    3.77    8.598
covF - High     27.75    6.190
A^2             56.95   13.023
B*C             51.84   20.426
B*D             52.77   20.132
raw PS         130.90   37.975
linear PS      124.61   34.211
```

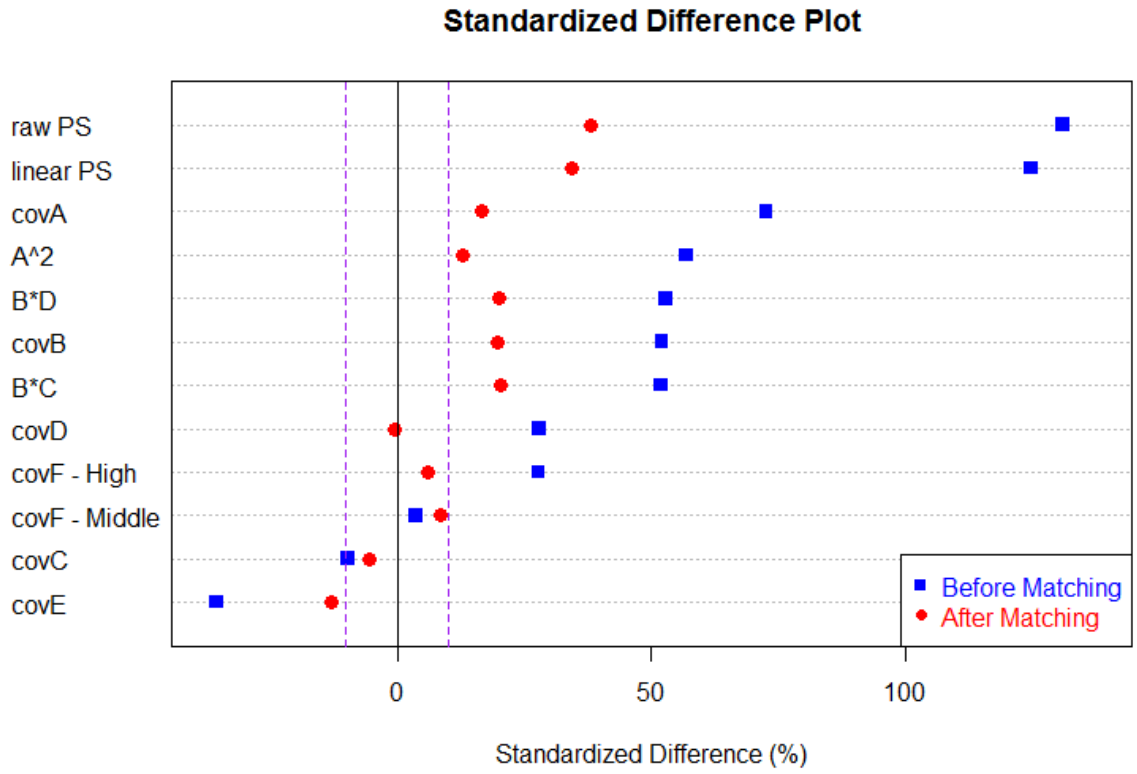
Plot the absolute values of the standardized differences…

```
> 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))
```



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))
```

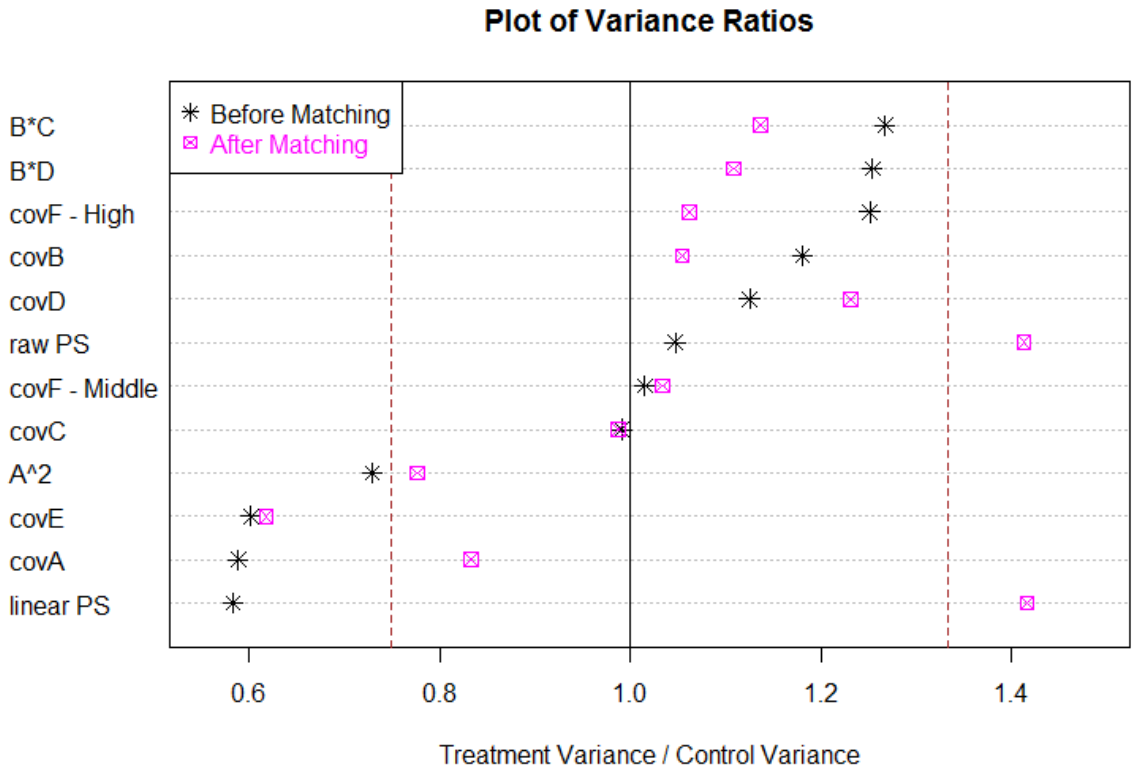


Extracting, Tabulating and Plotting Variance Ratios

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
+ }

> 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

> 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=3/4, lty=2, col="brown")
> abline(v=4/3, lty=2, col="brown")
> legend("topleft", legend = c("Before Matching", "After
     Matching"), col=c("black", "magenta"), text.col=c("black",
     "magenta"), bty="o", pch = c(8, 7))
```

Thomas.Love@case.edu



Creating a New Data Frame, Containing the Matched Sample

Now, we build a new matched sample data frame in order to do some of the analyses to come.

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

And this new data frame contains the 140 patients (70 treated, 70 control) we need.

```
> table(toy.matchedsample$treated.f)
Treated Control
     70      70
> head(toy.matchedsample)
    matches subject treated covA
44      131      44       0 0.76
100     132     100       0 0.60
72      133      72       0 0.68    etc.
84      134      84       0 0.61
108     135     108       0 0.22
```

Interlude: Rubin's Rules to Check Balance After Matching

Here are the results we obtain for each of the three standards.

> Rubin's Rule 1: First, 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.

> Rubin's Rule 2: Second, 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 between 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
```

After matching, our Rule 2 result is:

```
> 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 ½ to 2.

We pass Rule 2, as well.

Rubin's Rule 3: Third, we calculate 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 <- rubin3(data=toy, covlist=toy[c("covA", "covB",
     "covC", "covD", "covE", "covF.Middle", "covF.High",
     "Asqr","BC", "BD")])

 covA  covB  covC  covD  covE covF.Middle covF.High  Asqr    BC
0.591 1.072 1.043 1.160 0.558       1.010     1.281 0.684 1.144

   BD
1.114
```

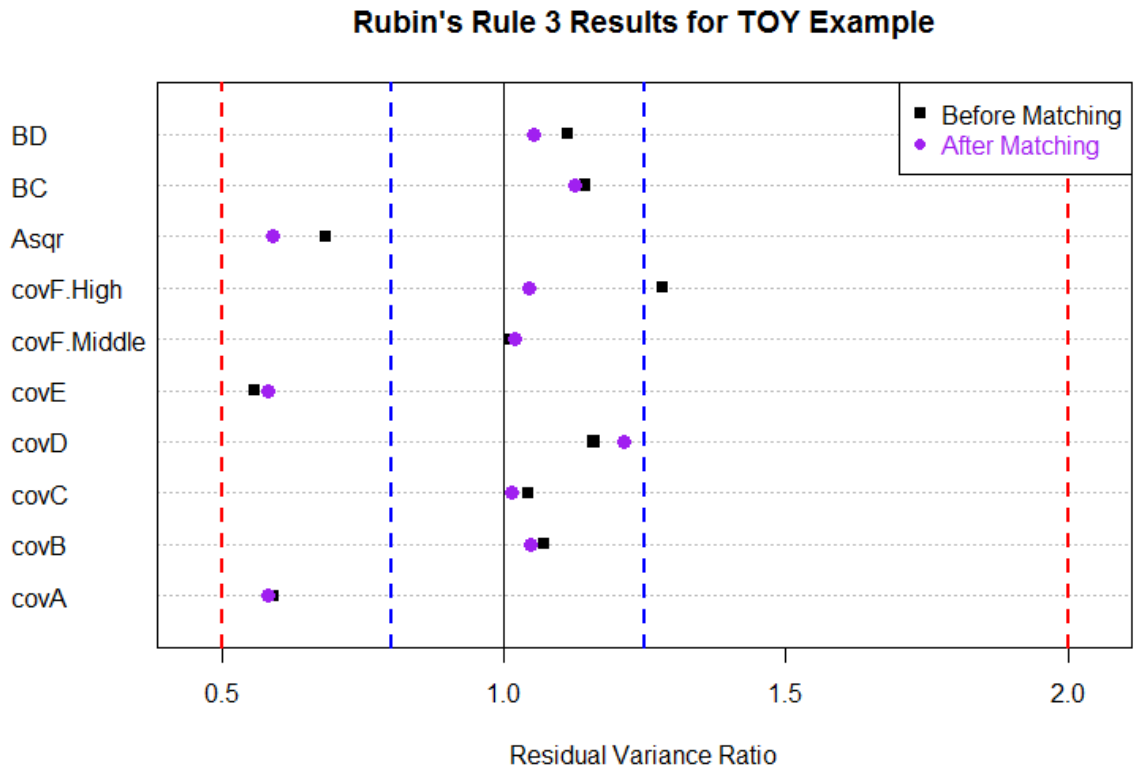
After propensity matching, we use our matched sample, created earlier, which includes the linear propensity score

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

 covA  covB  covC  covD  covE covF.Middle covF.High  Asqr    BC
0.583 1.047 1.014 1.214 0.581       1.019     1.044 0.592 1.125

   BD
1.054
```

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…

Building a Dotplot of the Rubin's Rule 3 results Comparing Pre- to Post-Match…

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

> low <- min(d.unadj, d.match, 0.45)
> high <- max(d.unadj, d.match, 2.05)

> dotchart(d.unadj, pch=15, col="black", main="Rubin's Rule 3
     Results for TOY Example", xlab="Residual Variance Ratio",
     xlim=c(low, high))
> points(d.match, seq(1:length(d.match)), pch=19, col="purple",
     cex=1.2)
> abline(v=1, lty=1)
> abline(v=0.8, lty=2, lwd=2, col="blue")
> abline(v=1.25, lty=2, lwd=2, col="blue")
> abline(v=0.5, lty=2, lwd=2, col="red")
> abline(v=2, lty=2, lwd=2, col="red")
> legend("topright", legend=c("Before Matching", "After
     Matching"), col=c("black", "purple"), text.col=c("black",
     "purple"), bty="o", pch = c(15, 19))
```



Question D. After matching, estimate the causal effect of treatment vs control on…

…Outcome 1 (a continuous outcome)

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 library 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**).

Approach 2. Automated Approach from the Matching Library – ATE Estimate

Alternatively, we can get an ATE estimate from the Matching library, as follows...

```
> match1.ATE <- Match(Y=Y, Tr=Tr, X=X, M = 1, replace=FALSE,
     ties=FALSE, estimand="ATE")
Warning message: You may want to estimate ATT instead.
> summary(match1.ATE)
Estimate...  16.014    SE.........  1.5614
T-stat.....  10.257    p.val......  < 2.22e-16
```

And our 95% CI would be 16.014 ± 1.96(1.5614), or (13.0, 19.1)

Approach 3. Using the matched sample to perform a Paired T test

The problem with the "automated" approach to a continuous outcome is that it doesn't quite take into account the fact that pairing is involved, but it's a different sort of pairing than we'd get in some other studies. We can essentially mimic the automated result with a paired t test…

```
> by(toy.matchedsample$out1.cost, toy.matchedsample$treated.f,
     summary)  ## description doesn't account for paired samples
toy.matchedsample$treated.f: Treated
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  24.00   50.25   62.00   60.61   75.75   80.00
------------------------------------------------------------
toy.matchedsample$treated.f: Control
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  24.00   36.50   46.50   45.06   52.75   78.00
```

Of course, this little summary doesn't account for the pairing. To do so, run a paired t test:

```
> t.test(toy.matchedsample$out1.cost ~
     toy.matchedsample$treated.f, paired=TRUE)
Paired t-test
data:  toy.matchedsample$out1.cost by
toy.matchedsample$treated.f
t = 7.5726, df = 69, p-value = 1.208e-10
alt. hypothesis: true difference in means is not equal to 0
95 percent confidence interval: 11.45873 19.65556
sample estimates: mean of the differences = 15.55714
```

Approach 4. 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)
> summary(adj.m.out1); confint(adj.m.out1)

Call: lm(formula = out1.cost ~ treated + factor(matches), data =
toy.matchedsample)

Residuals:     Min      1Q  Median      3Q     Max
           -23.279  -4.779   0.000   4.779  23.279
```

```
Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          3.472e+01  8.655e+00   4.012 0.000151 ***
treated              1.556e+01  2.054e+00   7.573 1.21e-10 ***
factor(matches)132   8.000e+00  1.215e+01   0.658 0.512587
factor(matches)133  -1.000e+00  1.215e+01  -0.082 0.934664
```

etc. etc.

```
factor(matches)199  -1.000e+00  1.215e+01  -0.082 0.934664
factor(matches)200   1.050e+01  1.215e+01   0.864 0.390628
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.15 on 69 degrees of freedom
Multiple R-squared:  0.7096,  Adjusted R-squared:  0.415
F-statistic: 2.409 on 70 and 69 DF,  p-value: 0.0001647

                         2.5 %   97.5 %
(Intercept)          17.4545085 51.98835
treated              11.4587300 19.65556
factor(matches)132  -16.2465372 32.24654
factor(matches)133  -25.2465372 23.24654
```

etc. etc.

```
factor(matches)199  -25.2465372 23.24654
factor(matches)200  -13.7465372 34.74654
```

Recall that when R says 1.556 e+01, it's just its cute way of saying 15.56 (or, more precisely, 1.556×10^1). Or, if you just want to look at the **treated** variable, as we do here, you can use…

```
> coef(adj.m.out1)["treated"] # point estimate
 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 test, but this isn't something I'm completely comfortable with.

Approach 5. 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** library. Note that we have to create a factor variable to represent the matches, since that's the only think **lme4** understands.

```
> library(lme4)
> toy.matchedsample$matches.f <-
     as.factor(toy.matchedsample$matches)
> 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: 1122.453

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
treated           15.557      2.054   7.573

Correlation of Fixed Effects:
        (Intr)
treated -0.619
                 2.5 %    97.5 %
.sig01        0.503528  9.895039
.sigma       10.313755 14.378709
(Intercept)  41.802672 48.311613
treated      11.503977 19.610308
```

So, let's see. Does any of this really matter? Well, no, not within the ATT approaches in this case. I'll use the mixed model later, but it really doesn't matter in any meaningful way…

	Effect Estimate	Standard Error	95% CI
"Automated" ATT Approach via Match	15.56	2.04	11.56, 19.55
Paired t test	15.56	2.05	11.46, 19.66
Linear model (pairs a fixed factor)	15.56	2.05	11.46, 19.66
Mixed model (pairs a random factor)	15.57	2.05	11.50, 19.61

…Outcome 2 (a binary outcome)

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 library 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 patients) 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.

Approach 2. Automated Approach from the Matching Library (ATE)

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

Estimate...  0.085714    SE.........  0.053914
T-stat.....  1.5898      p.val......  0.11187
```

And our 95% CI would be 0.0857 ± 1.96(0.0539), or (-0.02, 0.19)

Approach 3. 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.

```
> library(survival)
> 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 highlighted above is the average causal effect estimate – it describes the odds of having an event (out2) occur associated with being a treated patient, as compared to the odds of the event when a control patient. 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.

Approach 4. Using the matched sample to perform McNemar's test

If we are not planning any further covariate adjustment, we can obtain a McNemar's test to compare outcomes from these paired samples. First, we reshape the data into a form the McNemar test can deal with. In reshaping the data, you'll note that I use the factor version of both the treatment indicator (i.e. **treated.f**) and the outcome (i.e. **out2.f**). The treatment indicator (**treated.f**) takes on the values Treated and Control, so that's what appears in the third and fourth lines of this code. The levels of the outcome as a factor (**out2.f**) are Event Occurred and No Event, so that's what appears in the levels section of lines 3 and 4 below. In the labels section of lines 3 and 4, you can abbreviate as you like. Our goal here is to obtain a 2x2 table, with one cell count for each *pair* of observations – so that our 140 matched patients should be represented here by a total *n* of 70 – the number of matched pairs.

```
> library(reshape2)
> wide.out2 <- dcast(toy.matchedsample, matches ~ treated.f,
     value.var="out2.f")
> wide.out2$Treated <- factor(wide.out2$Treated, levels=c("Event
     Occurred","No Event"), labels=c("Treated, Event", "Treated,
     No Event"))
> wide.out2$Control <- factor(wide.out2$Control, levels=c("Event
     Occurred","No Event"), labels=c("Ctrl, Event", "Ctrl, No
     Event"))
> table(wide.out2$Treated, wide.out2$Control)
                    Ctrl, Event Ctrl, No Event
  Treated, Event             21             18
  Treated, No Event          11             20
```

And 21 + 18 + 11 + 20 = 70, so we're OK. 21 pairs of patients had both their Treated and Control patients have the Event. 20 pairs had both not have the Event – these are called the concordant pairs, and don't tell us anything about the treatment effect on this binary outcome. Instead, we learn from the discordant pairs, where exactly one of the two members has the Event. We have 18 pairs where only the Treated patient has the event, as compared to 11 pairs where only the Control patient does. Here's the complete McNemar result:

```
> library(epibasix)
> mcNemar(as.matrix(table(wide.out2$Treated,
     wide.out2$Control)))
Error in mcNemar(as.matrix(table(wide.out2$Treated,
     wide.out2$Control))) : The number of discordant pairs, must
     be greater than 30 to ensure validity.
```

Oops - this won't give an answer since we have < 30 discordant pairs. 29 in fact.

To force it to show us the answer anyway, we can use

```
> mcNemar(as.matrix(table(wide.out2$Treated,
     wide.out2$Control)), force=TRUE)

Matched Pairs Analysis: McNemar's Chi^2 Statistic and Odds Ratio

McNemar's Chi^2 Statistic (corrected for continuity) = 1.241
which has a p-value of: 0.265

McNemar's Odds Ratio (b/c): 1.636
95% Confidence Limits for the OR are: [0.745, 4.392]
```

The McNemar Odds Ratio is just the ratio of the counts of discordant pairs, or 18/11 = **1.64**, just as the conditional logistic regression gave us, and this is associated with a somewhat wider 95% CI of (**0.75**, **4.39**), as compared to the conditional logistic regression approach.

…Outcome 3 (a time-to-event outcome)

Approach 1. Automated Approach from the Matching Library

Again, we'll start by looking at 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.

```
> X <- toy$linps ## matching on the linear propensity score
> Tr <- as.logical(toy$treated)
> Y <- toy$out3.time
> match1 <- Match(Y=Y, Tr=Tr, X=X, M = 1, replace=FALSE,
     ties=FALSE)
> summary(match1)
Estimate...  7.4571
SE.........  1.7877
T-stat.....  4.1714
p.val......  3.0271e-05

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

Approach 2. A stratified Cox proportional hazards model

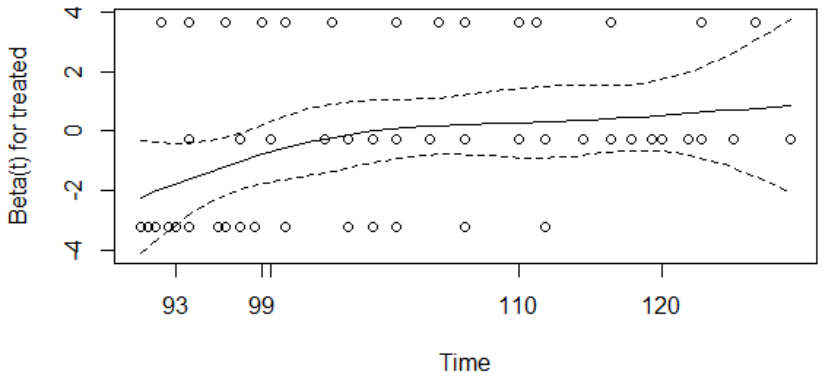
Since we have the matched sample, we should use a stratified Cox proportional hazards model to compare the treatment groups on our time-to-event outcome, while accounting for the matched pairs. The main results will be a relative hazard rate estimate, with 95% CI. Again, I use the 0/1 numeric version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

```
> library(survival)
> 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 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)
          rho chisq      p
treated 0.288   5.9 0.0152
```

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



Summary of Results 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.

Causal Effect of Treatment Estimates, with (95% CI)	**Outcome 1** Cost difference	**Outcome 2** Risk Difference	**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 Propensity Match (Match's Automatic ATT Results)	**15.6** (11.6, 19.6)	**+0.10** (-0.05, +0.25)	**N/A**	**N/A**
After 1:1 Propensity Match (Match's Automatic ATE Results)	**16.0** **(13.0, 19.1)**	**+0.09** (-0.02, +0.19)	**N/A**	**N/A**
After 1:1 Propensity Match (Detailed "Regression" Models)	**15.6** **(11.5, 19.6)**	**N/A**	**1.64** **(0.77, 3.47)**	**0.75** **(0.41, 1.38)**

Question E. Now subclassify into quintiles by the linear PS, and display/assess balance…

We'll start by running the subclassification through the **cut2** function in the **Hmisc** library. Since there are 200 subjects overall, we should wind up with 40 in each of the five quintiles.

```
> library(Hmisc)
> toy$stratum <- cut2(toy$ps, g=5)
> toy$quintile <- factor(toy$stratum, labels=1:5)
> table(toy$stratum, toy$quintile) ## quick 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
```

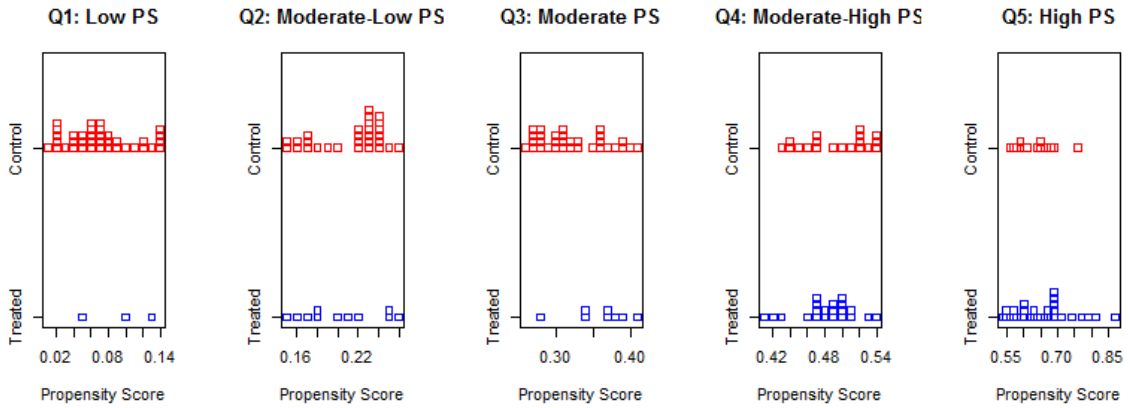
Next, we'll split the original data frame into the five quintiles…

```
> quin1 <- subset(toy, quintile==1)
> quin2 <- subset(toy, quintile==2)
Etc.
```

A Plot of Overlap in the Propensity Score Quintiles

Within each quintile, we'll check the balance and level of propensity score overlap through plots. We can check the overlap in the raw propensity scores from a set of stripcharts…

```
> par(mfrow=c(1,5))
> stripchart(round(ps,2) ~ treated.f, data=quin1, main="Q1: Low
    PS", method="stack", cex=1, offset=1/2, col=c("blue",
    "red"), xlab="Propensity Score")
> stripchart(round(ps,2) ~ treated.f, data=quin2, main="Q2:
    Moderate-Low PS", method="stack", cex=1, offset=1/2,
    col=c("blue", "red"), xlab="Propensity Score")
> stripchart(round(ps,2) ~ treated.f, data=quin3, main="Q3:
    Moderate PS", method="stack", cex=1, offset=1/2,
    col=c("blue", "red"), xlab="Propensity Score")
> stripchart(round(ps,2) ~ treated.f, data=quin4, main="Q4:
    Moderate-High PS", method="stack", cex=1, offset=1/2,
    col=c("blue", "red"), xlab="Propensity Score")
> stripchart(round(ps,2) ~ treated.f, data=quin5, main="Q5: High
    PS", method="stack", cex=1, offset=1/2, col=c("blue",
    "red"), xlab="Propensity Score")
> par(mfrow=c(1,1))
```



This amplifies the results from a simple table looking at treatment status within each quintile. We're spreading the 70 treated observations fairly thinly, especially in the first three quintiles.

```
> 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 szd Function – A General Approach to Calculating Standardized Differences

We need to be able to calculate standardized differences in this setting, rather than having the MatchBalance function to do it for us. I've built a function to help, called **szd** …

```
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
}
```

The `szd` function uses the following definition of the standardized difference…

$$\text{standardized difference} = \frac{100(\bar{x}_T - \bar{x}_C)}{\sqrt{\frac{s_T^2 + s_C^2}{2}}}$$

Our next step is to collect the standardized differences within each quintile (and across the full sample) for the list of all covariates, transformations, and the raw and linear propensity score…

```
> covs <- c("covA", "covB", "covC", "covD", "covE",
     "covF.Middle", "covF.High", "Asqr","BC", "BD", "ps",
     "linps")
> 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)

> d.q1
  covA   covB   covC   covD   covE covF.Middle covF.High
 28.95 -79.09 -63.18 122.55  87.36      -13.37    -41.44
  Asqr     BC     BD     ps   linps
-16.01 -76.42 -77.19  60.26   69.51
```

```
> d.q2
  covA   covB   covC   covD   covE covF.Middle covF.High
-48.68  42.71 -21.32  16.91   9.75      -43.02     34.02
  Asqr     BC     BD     ps  linps
-54.47  47.05  45.35 -28.92 -29.13

> d.q3
  covA   covB   covC   covD   covE covF.Middle covF.High
 22.76   7.07  12.21 -36.84 -27.03       -6.15    -31.32
  Asqr     BC     BD     ps  linps
 14.14   4.12  -5.83  87.03  87.05

> d.q4
  covA   covB   covC   covD   covE covF.Middle covF.High
 30.63 -51.90  11.86 -23.62 -63.96       21.39    -16.98
  Asqr     BC     BD     ps  linps
 32.01 -49.87 -54.26 -27.41 -27.42

> d.q5
  covA   covB   covC   covD   covE covF.Middle covF.High
 65.07  26.26  11.17  13.73  62.40       19.08      4.53
  Asqr     BC     BD     ps  linps
 64.48  27.44  36.51  35.97  38.54

> d.all
  covA   covB   covC   covD   covE covF.Middle covF.High
 54.83  36.80  -6.76  19.63 -26.18        2.66     19.33
  Asqr     BC     BD     ps  linps
 41.90  36.47  37.18 102.60 100.70
```

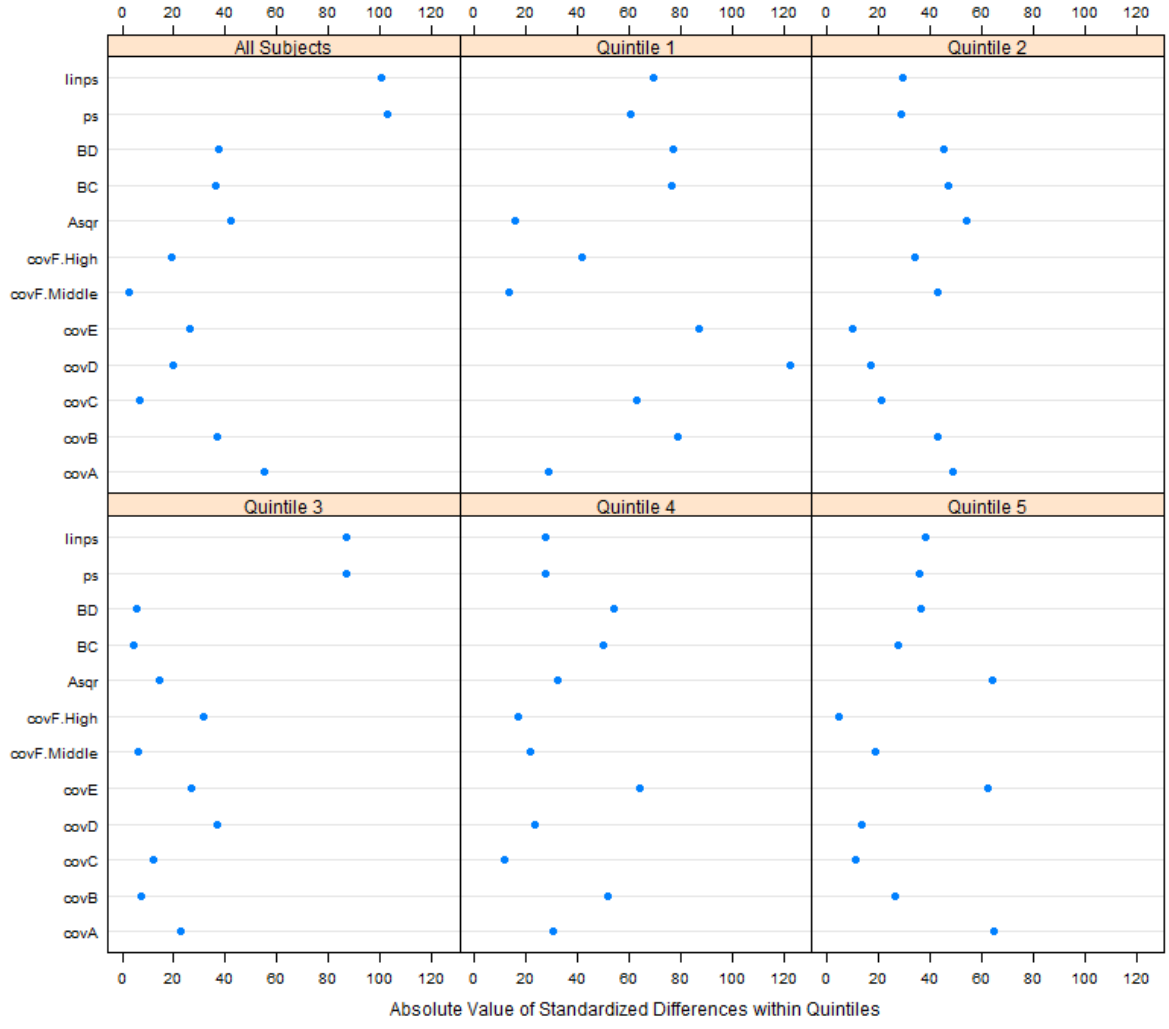
We can build a pretty table of this stuff, but it can be lots of copy and pasting work. It's usually much better to look at plots.

Variable	Quintile 1	Quintile 2	Quintile 3	Quintile 4	Quintile 5	All Subjects
# Treated	3	11	8	21	27	70
# Control	37	29	32	19	13	130
szd PS	60	-29	87	-27	36	103
szd Linear PS	70	-29	87	-27	39	101

Plotting Standardized Differences across Quintiles with lattice

The **lattice** package in R lets us plot multiple small, similarly indexed plots – which can be very helpful in this setting. For instance, consider the following plot, which describes the absolute standardized differences for each of our covariates within each quintile.

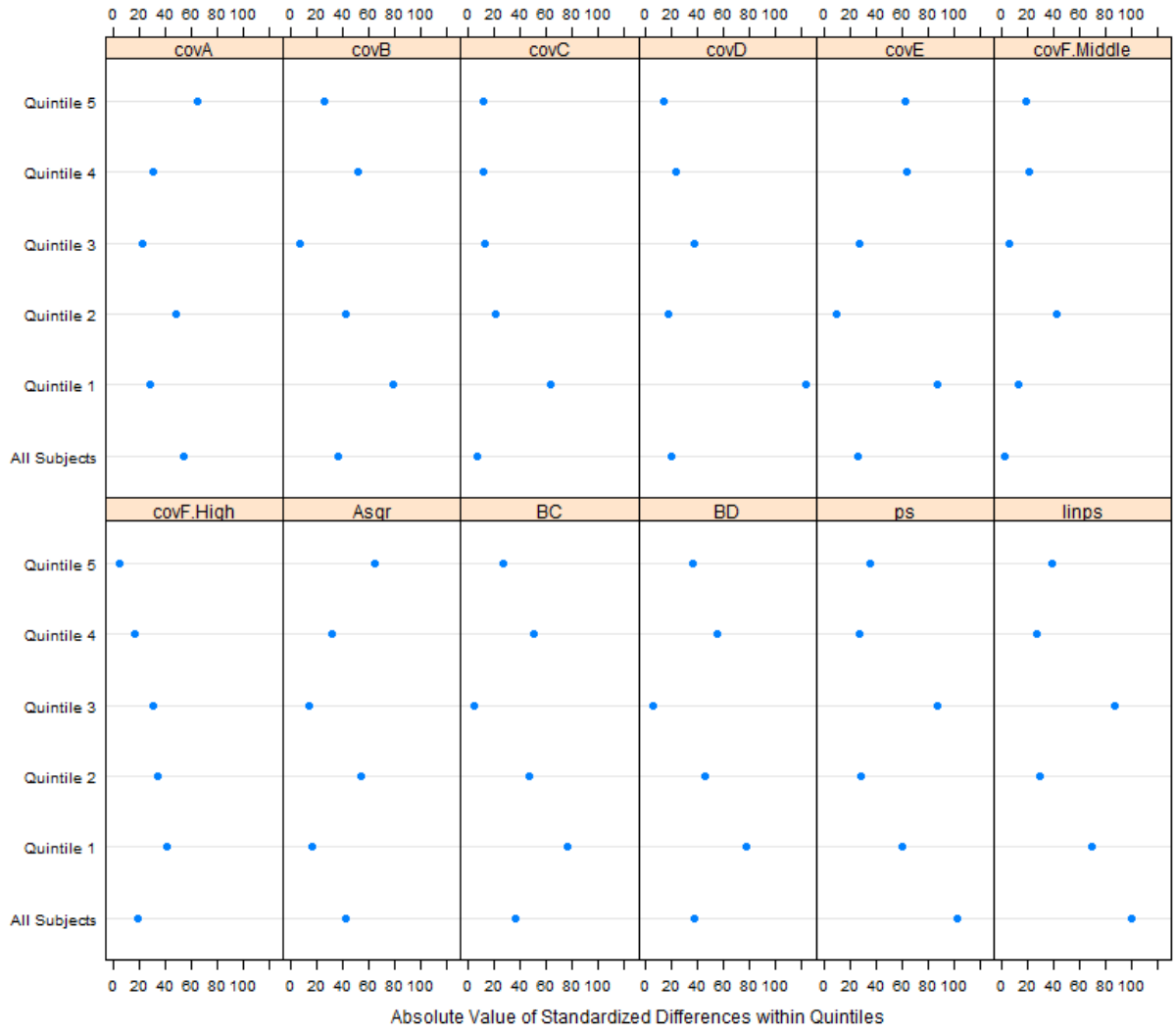
```
> library(lattice)
> temp <- matrix(c(abs(d.all), abs(d.q1), abs(d.q2), abs(d.q3),
      abs(d.q4), abs(d.q5)), ncol=6)
> dimnames(temp) <- list(names(d.q1), c("All Subjects",
      "Quintile 1", "Quintile 2", "Quintile 3", "Quintile 4",
      "Quintile 5"))
> dotplot(temp, groups=F, xlab="Absolute Value of Standardized
      Differences within Quintiles",
      scales = list(alternating=3), as.table=TRUE)
```



As you can see, none of the quintiles is especially well balanced across all covariates.

Another plotting approach is even better, in my view, for looking at the balance of individual covariates, but, like the prior plot, it can get to be a problem if you have lots of covariates.

```
> dotplot(t(temp), groups=F, xlab="Absolute Value of
    Standardized Differences within Quintiles",
    scales = list(alternating=3), as.table=TRUE)
```



OK – enough. Either of these approaches is just fine, if a little hard to fight through.

Looking at Variance Ratios across Quintiles

I built a silly function in R to calculate the variance ratios for this set of covariates for any particular subset – like the quintiles.

```
> vratio.toy <- function(data)
 {
   covA.vr <- with(data, var(covA[treated==1])/var(covA[treated==0]))
   covB.vr <- with(data, var(covB[treated==1])/var(covB[treated==0]))
   covC.vr <- with(data, var(covC[treated==1])/var(covC[treated==0]))
   covD.vr <- with(data, var(covD[treated==1])/var(covD[treated==0]))
   covE.vr <- with(data, var(covE[treated==1])/var(covE[treated==0]))
   covF.Middle.vr <- with(data,
     var(covF.Middle[treated==1])/var(covF.Middle[treated==0]))
   covF.High.vr <- with(data,
     var(covF.High[treated==1])/var(covF.High[treated==0]))
   Asqr.vr <- with(data, var(Asqr[treated==1])/var(Asqr[treated==0]))
   BC.vr <- with(data, var(BC[treated==1])/var(BC[treated==0]))
   BD.vr <- with(data, var(BD[treated==1])/var(BD[treated==0]))
   ps.vr <- with(data, var(ps[treated==1])/var(ps[treated==0]))
   linps.vr <- with(data,
     var(linps[treated==1])/var(linps[treated==0]))
   res.vr <- round(c(covA.vr, covB.vr, covC.vr, covD.vr, covE.vr,
     covF.Middle.vr, covF.High.vr, Asqr.vr, BC.vr, BD.vr, ps.vr,
     linps.vr),3)
   names(res.vr) <- c("A", "B", "C", "D", "E", "F-middle", "F-high",
     "A^2", "BC", "BD", "ps", "linear ps")
   res.vr
    }
```

I then gathered the quintile-specific variance ratios, by calling this function multiple times.

```
> vratio.toy(quin1)
    A     B     C     D     E F-mid  F-hi   A^2    BC    BD    ps linps
0.102 0.000 0.716 0.183 0.566 1.345 0.000 0.021 0.000 0.000 0.964 0.478
```

Across the five quintiles, I get the following summaries of variance ratios:

Quintile	Range across 10 covariates A, B, C, D, E, F-mid, F-hi, A^2, BC, BD	Propensity Score	Linear PS
1	0.00 to 1.35	0.96	0.48
2	0.40 to 2.99	1.24	1.20
3	0.32 to 1.22	0.82	0.79
4	0.84 to 1.85	0.79	0.80
5	0.80 to 1.32	2.05	2.49
All of the Data	**0.59 to 1.27**	**1.05**	**0.58**

None of these look very good – perhaps quintile 4 is OK.

Checking Rubin's Rules After Subclassification on the Propensity Score

> Rubin's Rule 1: First, 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 adjustment was

```
> rubin1.unadj
[1] 88.11531
```

We begin by checking Rubin's Rule 1 in each quintile.

```
> 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
```

Recall that we want this value to be as close to 0 as possible, and certainly less than 50. These results are not very promising, with quintiles 1 and 3 especially poorly balanced, while 2 and 4 are better than our cutoff (50%). All are at least a bit improved from our original value across the entire data set (88%).

Rubin's Rule 2: Second, 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 between 1/2 and 2. If so, we may move on to Rule 3.

Recall that our result without propensity adjustment was

```
> rubin2.unadj
[1] 0.5835438
```

Again, we'll just check the result within each quintile.

```
> 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
```

In quintiles 1 and 5, the variance ratio is not between ½ and 2, so we fail Rule 2.

In the other quintiles, the results aren't great – we'd be comfortable if all of these were between 4/5 (0.8) and 5/4 (1.25) – we're close on the interior quintiles to these limits.

Rubin's Rule 3: Third, we calculate 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

```
> covs <- c("covA", "covB", "covC", "covD", "covE",
     "covF.Middle", "covF.High", "Asqr","BC", "BD")
> rubin3.unadj <- rubin3(data=toy, covlist=toy[covs])
  covA     covB     covC     covD     covE covF.Middle
 0.591    1.072    1.043    1.160    0.558       1.010
  covF.High     Asqr       BC       BD
      1.281    0.684    1.144    1.114
```

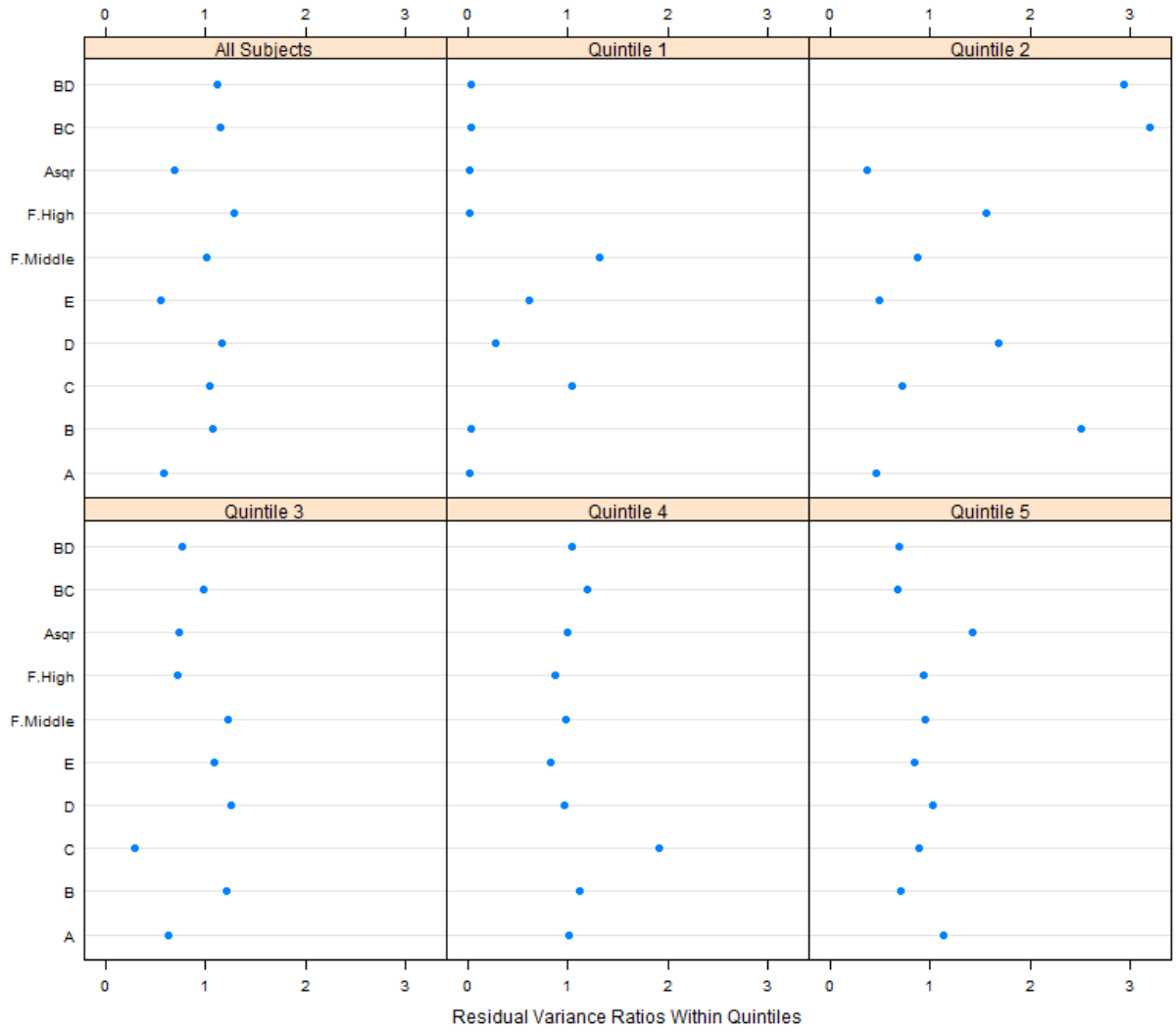
Again, we check each quintile separately. I've marked in red those outside the ½, 2 range.

```
> rubin3.q1 <- rubin3(data=quin1, covlist=quin1[covs])
  covA     covB     covC     covD     covE covF.Middle
 0.015    0.036    1.046    0.276    0.612       1.316
  covF.High     Asqr       BC       BD
      0.010    0.019    0.034    0.037
> rubin3.q2 <- rubin3(data=quin2, covlist=quin2[covs])
  covA     covB     covC     covD     covE covF.Middle
 0.463    2.504    0.724    1.679    0.494       0.867
  covF.High     Asqr       BC       BD
      1.555    0.362    3.185    2.928
> rubin3.q3 <- rubin3(data=quin3, covlist=quin3[covs])
  covA     covB     covC     covD     covE covF.Middle
 0.633    1.205    0.288    1.263    1.084       1.222
  covF.High     Asqr       BC       BD
      0.722    0.733    0.980    0.774
> rubin3.q4 <- rubin3(data=quin4, covlist=quin4[covs])
  covA     covB     covC     covD     covE covF.Middle
 1.008    1.111    1.914    0.959    0.829       0.985
  covF.High     Asqr       BC       BD
      0.865    0.992    1.194    1.035
> rubin3.q5 <- rubin3(data=quin5, covlist=quin5[covs])
  covA     covB     covC     covD     covE covF.Middle
 1.12     0.705    0.879    1.019    0.836       0.944
  covF.High     Asqr       BC       BD
      0.927    1.423    0.668    0.695
```

And we can see some real problems (values outside of the 0.5 to 2.0 range) especially in the lower PS quintiles. The worst is likely covF.High in Quintile 1, which has a variance ratio of 0.1.

Building a Lattice Plot to look at Rubin's Rule 3 results after Subclassification

```
> library(lattice)
> temp <- matrix(c(rubin3.unadj, rubin3.q1, rubin3.q2,
     rubin3.q3, rubin3.q4, rubin3.q5), ncol=6)
> names(rubin3.unadj) <- c("A", "B", "C", "D", "E", "F.Middle",
     "F.High", "Asqr","BC", "BD")
> dimnames(temp) <- list(names(rubin3.unadj), c("All Subjects",
     "Quintile 1", "Quintile 2", "Quintile 3", "Quintile 4",
     "Quintile 5"))
> dotplot(temp, groups=F, xlab="Residual Variance Ratios Within
     Quintiles", scales = list(alternating=3), as.table=TRUE)
```



Question F. After subclassification, estimate the causal effect of treatment vs control on…

…Outcome 1 (a continuous outcome)

Combining Quintile-Specific Linear Regression Models

First, we find and store the estimated average causal effect (and standard error) within each quintile via a series of linear regression models

```
> 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
```

Or, in a more presentable fashion…

Quintile	Treatment Effect Estimate	Standard Error	# Treated	# Control
1	-12.46	5.5	3	37
2	1.07	3.5	11	29
3	10.50	4.6	8	32
4	16.92	4.0	21	19
5	23.44	3.6	27	13

Those five estimates really don't look like they are succeeding in estimating the same thing. Some of this is because of the modest sample size – subclassification is more effective if the minimum number of patients in any of the treated or control cells is still pretty large.

The five resulting treatment effect estimates are very different – in practice, I don't think it would be sensible to combine them. Here, I still want to show you how it's done.

Since each quintile contains the same number of patients, a weighted average would weight them equally at 1/5 of the total. So our estimate is just the average of the estimates.

```
> 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 estimate of the standard error, we square each quintile-specific standard error to get a variance, divide by the number of quintiles squared – i.e. $5^2 = 25$ – and then take the square root to get back to the standard error scale. The standard error over the combined sample will typically, as here, be smaller than the standard error for the individual quintiles.

```
> 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 effect is thus

```
> 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
```

And our estimate after subclassification on the propensity score is that the average causal effect on cost for treated as compared to control patients is an increase of **7.9**, with 95% CI of (**4.1**, **11.7**)

…Outcome 2 (a binary outcome)

Approach 1: Quintile-Specific Logistic Regression, yielding Odds Ratios

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

```
> 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))
              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
```

and so on, for the other four quintiles. This is a log odds ratio – the odds ratio is $e^{-0.639} = 0.53$

```
> exp(-0.639)
[1] 0.52782
```

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 ## estimated log odds ratio
  treated
0.1405919
```

Exponentiate this log odds ratio estimate to get the overall odds ratio estimate

```
> exp(est.st)
 treated
1.150955
```

To get the combined standard error estimate:

```
> 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))
```

Of course, this standard error is also on the log odds ratio scale, so we must apply it to the original log odds ratio estimate and then exponentiate to get the 95% Confidence Interval for Average Causal Effect (as an Odds Ratio) which is ...

```
> 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
```

So, after subclassification, we estimate the odds ratio for having the event given treatment as compared to control as **1.15**, with 95% CI (**0.54**, **2.44**)

Approach 2: Individual 2x2 Comparisons by Quintile to look at Risk Differences

To look at risk differences (or odds ratios, I suppose) we can look at the output from the twoby2 function for each quintile separately.

```
> library(Epi)
> twoby2(table(quin1$treated.f, quin1$out2.f))
2 by 2 table analysis:
------------------------------------------------------
Outcome    : Event Occurred
Comparing : Treated vs. Control

          Event
        Occurred No Event  P(Event Occurred) 95% conf. interval
Treated        1        2            0.3333    0.0434   0.8465
Control       18       19            0.4865    0.3321   0.6435

                                   95% conf. interval
             Relative Risk:  0.6852    0.1337   3.5118
         Sample Odds Ratio:  0.5278    0.0440   6.3372
Conditional MLE Odds Ratio:  0.5360    0.0085  11.1408
    Probability difference: -0.1532   -0.4659   0.3304

             Exact P-value: 1
        Asymptotic P-value: 0.6143
------------------------------------------------------
```

And we can run the same comparison for each of the other quintiles.

Here are some of the results from the twoby2 function, after artisanal crafting into a more attractive Word table. As you can seem the risk differences, at least in quintiles 1 and 5, don't look very comparable to each other.

Quintile	Risk Difference Estimate	Risk Difference 95% CI
1	-0.153	(-0.466, 0.330)
2	0.053	(-0.226, 0.368)
3	0.063	(-0.269, 0.387)
4	0.040	(-0.243, 0.317)
5	0.168	(-0.142, 0.445)

To combine these risk difference estimates, we'd need to estimate standard errors. Happily, this is approximately one-quarter of the width of the 95% confidence interval. So, for instance, the width of the interval in quintile 1 is just under 0.8, and dividing this by four, we get…

```
> (.330-(-.466))/4
[1] 0.199
```

Across the five quintiles, then, the estimated standard errors associated with the risk difference estimates are:

Quintile	Estimate	95% CI	Width of CI	Std. Error (Width/4)	Variance (SE^2)
1	-0.153	(-0.466, 0.330)	.796	0.199	0.0396
2	0.053	(-0.226, 0.368)	.594	0.149	0.0221
3	0.063	(-0.269, 0.387)	.656	0.164	0.0269
4	0.040	(-0.243, 0.317)	.560	0.140	0.0196
5	0.168	(-0.142, 0.445)	.587	0.147	0.0215
Mean	0.034			**Mean**	0.0259

So our estimated risk difference is 0.034, with a standard error equal to the square root of 0.0259, which is 0.161, yielding an approximate 95% CI of **0.034** ± 2(0.161), or (**-0.29**, **0.36**)

We could use the same approach (averaging the effect estimates and the variances, then taking the square root of the average variance to get a standard error) to get results for the odds ratio estimate, based on the confidence intervals provided in the twoby2 output. I prefer the much simpler logistic regression approach for that task. The results here will be a little different – if you work it out from the cross-product odds ratios, you get an overall point estimate of 1.25 rather than the 1.15 we saw in the logistic regression models. You get closer, at least in this case, to the logistic regression answer by working with the conditional MLE odds ratios.

…Outcome 3 (a time-to-event outcome)

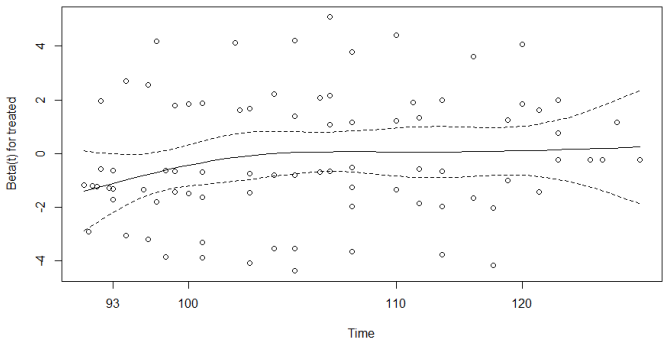
Cox Proportional Hazards Model to Estimate Relative Hazard Rate

We're going to run another Cox model here, comparing treated to control on our **out3.time** values, while stratifying on the propensity score quintiles.

```
> library(survival)
> adj.s.out3 <- coxph(Surv(out3.time, out2) ~ treated +
     strata(quintile), data=toy)
> summary(adj.s.out3)
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
```

The relative hazard rate estimate is **0.79**, with 95% CI (**0.49**, **1.27**).

A quick check of the proportional hazards assumption doesn't raise any major alerts. The plot (shrunken and shown at right) doesn't show any especially meaningful deviations from a horizontal line.




```
> cox.zph(adj.s.out3)
          rho chisq     p
treated 0.161  2.39 0.122
```

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

Results from Propensity Score Stratification by Quintile

So, here's our summary again, now incorporating the subclassification results, which, as we've noted, don't seem to balance the covariates sufficiently in this case.

Causal Effect of Treatment Estimates, with (95% CI)	**Outcome 1** Cost difference	**Outcome 2** Risk Difference	**Outcome 2** Odds Ratio	**Outcome 3** Relative Hazard Rate
After Stratification on the Quintiles of the Linear PS	**7.9** (4.1, 11.7)	**+0.03** (-0.29, 0.36)	**1.15** (0.54, 2.44)	**0.79** (0.49, 1.27)

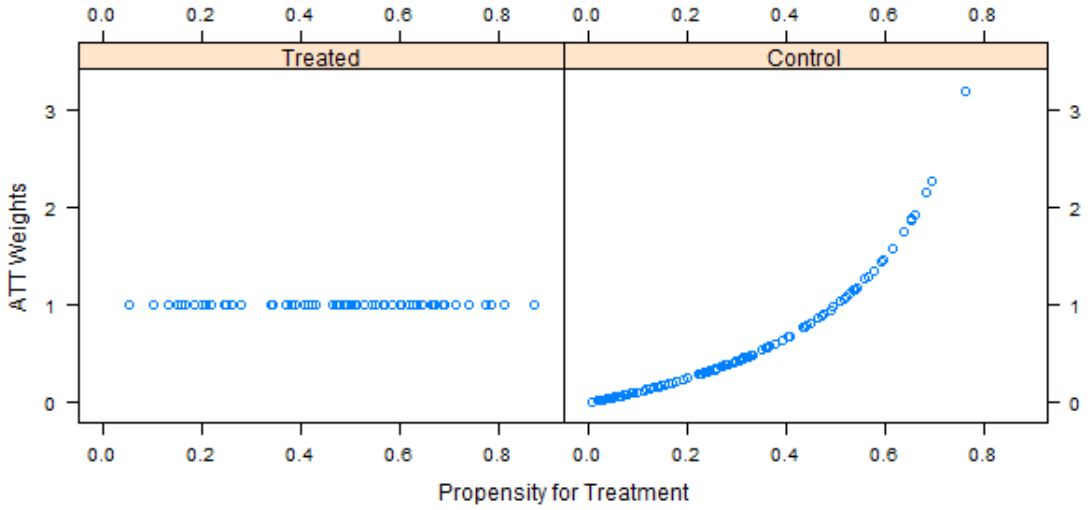
Question G. Now weight the observations by the PS, and display/assess balance…

There are at least three ways to do weighting in R. We'll look at two of them here, and save the third (which uses the **twang** library) for later discussion.

Approach 1. ATT Weights: 1 for treated patients, PS/(1-PS) for controls

We can take an average treatment effect on the treated approach (ATT) and weight each treated patient as 1, then each control patient as [propensity score / (1 – propensity score)].

```
> toy$wts1 <- ifelse(toy$treated==1, 1, toy$ps/(1-toy$ps))
> library(lattice)
> xyplot(wts1 ~ ps | treated.f, data=toy, ylab="ATT Weights",
xlab="Propensity for Treatment", scales = list(alternating=3))
```



Evaluating the Balance Imposed by ATT Weights via the twang package

The **twang** package (**t**ool kit for **w**eighting and **a**nalysis of **n**on-equivalent **g**roups) contains a series of functions for causal modeling through various kinds of propensity weighting. We'll just look at a piece of that tool kit here – assessing covariate balance imposed by our weighting. To begin, we specify the covariates whose balance we want to assess – note that variables already listed as factors (like covF) need not be specified at the level of indicator variables.

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

Next, we ask for information on the balance imposed by our ATT weights as follows. Within the dx.wts function, we have several arguments – **x** provides the weights, **data** indicates the data set, **vars** identifies the covariates and **treat.var** specifies the 0/1 treatment indicator. Here we are using ATT weights, and so will use an **ATT** **estimand**.

```
> bal.wts1 <- dx.wts(x=toy$wts1, data=toy, vars=covlist,
     treat.var="treated", estimand="ATT")
```

Next, we will ask for the details of the result here, which yields a table summarizing, for both the type=**unw** (unweighted) sample and, in the bottom row, our sample after weighting by wts1, the following elements:

n.treat and **n.ctrl**	Unweighted sample sizes in the treated and control groups, respectively
ess.treat and **ess.ctrl**	Effective sample size in treated and control groups (approximate # of observations from a simple random sample that yields an estimate with sampling variation equal to the sampling variation obtained with the weighted comparison observations.) Essentially, **ess.ctrl** estimates the # of comparison participants that are comparable to the treatment group.
max.es and **mean.es**	Maximum and mean absolute standardized mean difference of **e**ffect **s**ize, across all covariates (1/100 our usual standardized difference)
max.ks and **mean.ks**	Maximum and mean Kolmogorov-Smirnov (KS) statistic, summarized across all covariates

```
> bal.wts1
  type n.treat n.ctrl ess.treat  ess.ctrl    max.es    mean.es
1  unw      70    130        70 130.00000 1.1730321 0.40948475
2           70    130        70  62.42659 0.1259544 0.07031615
     max.ks    mean.ks iter
1 0.4725275 0.19771767   NA
2 0.1998033 0.08785673   NA
```

Repeating:

```
> bal.wts1
  type n.treat n.ctrl ess.treat  ess.ctrl    max.es    mean.es
1  unw      70    130        70 130.00000 1.1730321 0.40948475
2           70    130        70  62.42659 0.1259544 0.07031615
     max.ks    mean.ks iter
1 0.4725275 0.19771767   NA
2 0.1998033 0.08785673   NA
```

Here, we have 70 treated and 130 control observations before weighting, with a maximum effect size of 1.17 (i.e. a standardized difference of 117%) across the list of covariates we fed to the dx.wts function.

After weighting by our ATT weights, we see that we have an effective sample size of 70 treated patients (since ATT weighting weights each treated patient as 1, the effective sample size in ATT estimation for the treated group will always be the same as it was without weighting) and a little over 62 control patients, with a maximum effect size of 0.126 (i.e. a maximum standardized difference of 12.6%). This looks like a solid improvement in covariate balance, but we'd also like to look at similar results for each of the individual covariates.

Happily, we can do that by applying the `bal.table` function to our new balance results. Here is the complete table, which is quite long…

```
> bal.table(bal.wts1)
$unw
               tx.mn tx.sd  ct.mn ct.sd std.eff.sz   stat     p
covA           0.591 0.220  0.450 0.287      0.637  3.863 0.000
covB           0.486 0.503  0.308 0.463      0.354  2.461 0.015
covC           9.626 2.016  9.762 2.025     -0.068 -0.458 0.647
covD          10.096 1.877  9.738 1.770      0.191  1.317 0.189
covE           9.671 2.744 10.500 3.536     -0.302 -1.842 0.067
covF:1-Low     0.271 0.445  0.369 0.483     -0.220  1.317 0.269
covF:2-Middle  0.429 0.495  0.415 0.493      0.027     NA    NA
covF:3-High    0.300 0.458  0.215 0.411      0.185     NA    NA
Asqr           0.397 0.246  0.285 0.288      0.456  2.904 0.004
BC             4.584 5.008  2.857 4.449      0.345  2.427 0.016
BD             4.816 5.152  3.000 4.601      0.352  2.476 0.014
ps             0.476 0.191  0.282 0.187      1.014  6.922 0.000
linps         -0.139 0.917 -1.215 1.201      1.173  7.100 0.000
```

```
                ks ks.pval
covA         0.284   0.001
covB         0.178   0.098
covC         0.078   0.921
covD         0.110   0.598
covE         0.129   0.401
covF:1-Low   0.098   0.269
covF:2-Middle 0.013   0.269
covF:3-High  0.085   0.269
Asqr         0.284   0.001
BC           0.187   0.072
BD           0.181   0.087
ps           0.473   0.000
linps        0.473   0.000

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

                ks ks.pval
covA         0.200   0.121
covB         0.038   1.000
covC         0.086   0.943
covD         0.078   0.972
covE         0.125   0.619
covF:1-Low   0.028   0.913
covF:2-Middle 0.033   0.913
covF:3-High  0.005   0.913
Asqr         0.200   0.121
BC           0.081   0.963
BD           0.081   0.963
ps           0.093   0.903
linps        0.093   0.903
```

To explain the individual elements, let's just focus on the results before and after weighting for two variables – covA and the raw propensity score.

```
$unw   tx.mn tx.sd  ct.mn ct.sd std.eff.sz   stat     p    ks ks.pval
covA  0.591 0.220  0.450 0.287      0.637  3.863 0.000 0.284   0.001
ps    0.476 0.191  0.282 0.187      1.014  6.922 0.000 0.473   0.000

[[2]] tx.mn tx.sd  ct.mn ct.sd std.eff.sz   stat     p    ks ks.pval
covA  0.591 0.220  0.565 0.216      0.116  0.740 0.460 0.200   0.121
ps    0.476 0.191  0.463 0.177      0.069  0.410 0.682 0.093   0.903
```

tx.mn, tx.sd, ct.mn, ct.sd are the mean and standard deviation of the treated and control groups on the covariate in question.

std.eff.sz is the standardized effect size (1/100 of the standardized difference) – which uses the treatment group standard deviation as the denominator for ATT estimates, and the pooled standard deviation as the denominator for ATE estimates.

stat and **p** are the test statistic (t test if continuous, χ^2 if categorical) and associated p value

ks and **ks.pval** are the Kolmogorov-Smirnov test statistic and associated p value. For multi-categorical variables, this is just the χ^2 test p value across the categories.

So, we see, for instance, that the propensity scores prior to weighting have a standardized difference of 101%, based on means of .476 in the treated patients and .282 in the controls. After weighting, the standardized difference of the propensity score is just 6.9%, with a weighted mean of .463 in the control patients.

Here's a balance summary for each of these covariates and elements after ATT Weighting:

Variable	Standardized Difference	Treated Variance / Control Variance
covA	100*`std.eff.sz` = 11.6	[$.220^2$ / $.206^2$] = 1.04
covB	7.6	1.02
covC	7.4	1.06
covD	-1.4	1.20
covE	6.4	0.64
covF: 1-Low	-6.4	0.94
covF: 2-Middle	6.7	1.02
covF: 3-High	-1.0	0.99
A squared	12.6	1.00
B*C	8.3	1.07
B*D	8.9	1.08
Propensity Score	6.9	1.16
Linear PS	6.2	1.20

Rubin's Rules 1 and 2 to Check Balance After ATT Weighting

Here are the results we obtain for each of the three standards.

> Rubin's Rule 1: First, 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.

For our weighted sample, our summary statistic may be found from the bal.table output…

```
> bal.table(bal.wts1)
Edited…
[[2]]
        tx.mn tx.sd  ct.mn ct.sd std.eff.sz
linps  -0.139 0.917 -0.196 0.838       0.062
```

Here, 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.

> Rubin's Rule 2: Second, 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 between 1/2 and 2. If so, we may move on to Rule 3.

Again, after weighting, our summary statistic may be calculated from the bal.table output…

```
> bal.table(bal.wts1)
Edited…
[[2]]
        tx.mn tx.sd  ct.mn ct.sd std.eff.sz
linps  -0.139 0.917 -0.196 0.838       0.062
```

Here, 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 variance.

```
> 0.917^2/0.838^2
[1] 1.197431
```

This is promising – a substantial improvement over our unadjusted result, and now within our desired range of 4/5 to 5/4, as well as clearly within ½ to 2.

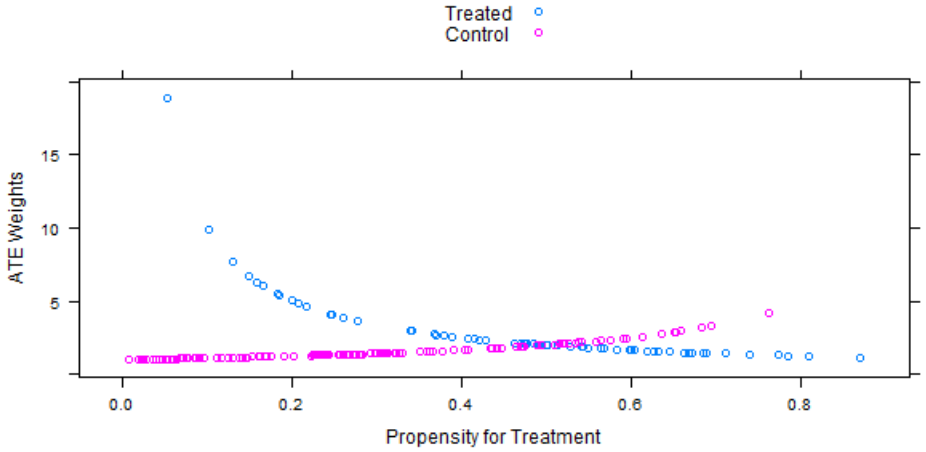
We pass Rule 2, as well.

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

Approach 2. ATE Weights: 1/PS for treated patients, 1/(1-PS) for controls

We can take an average treatment effect approach (ATE) and weight each treated patient as 1/(propensity score), then each control patient as 1/(1 – propensity score). This time, we'll plot the resulting weights on the same graph, rather than in a panel of graphs.

```
> toy$wts2 <- ifelse(toy$treated==1, 1/toy$ps, 1/(1-toy$ps))
> library(lattice)
> xyplot(wts2 ~ ps, groups=treated.f, data=toy, ylab="ATE
    Weights", xlab="Propensity for Treatment", auto.key=TRUE)
```



You can see that this weighting approach places greater emphasis on those treated patients with lower propensity scores, and on control patients with higher propensity scores, i.e. increased weight on patients whose propensity scores are unusual given their true exposure.

Evaluating the Balance Imposed by ATE Weights via twang

Note the use of the "ATE" estimand here, since these are ATE weights.

```
> bal.wts2 <- dx.wts(x=toy$wts2, data=toy, vars=covlist,
    treat.var="treated", estimand="ATE")
> bal.wts2
  type n.treat n.ctrl ess.treat ess.ctrl    max.es    mean.es
1  unw      70    130  70.00000 130.0000 0.9256094 0.36753949
2           70    130  38.96583 115.2132 0.1866526 0.05313253
    max.ks    mean.ks iter
1 0.4725275 0.19771767   NA
2 0.1436468 0.07716551   NA
```

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

Heavily edited here to show just covA and ps again – in practice, we'd look at all covariates.

```
$unw
      tx.mn tx.sd  ct.mn ct.sd std.eff.sz   stat     p    ks ks.pval
covA  0.591 0.220  0.450 0.287      0.513  3.863 0.000 0.284   0.001
ps    0.476 0.191  0.282 0.187      0.926  6.922 0.000 0.473   0.000

[[2]]
      tx.mn tx.sd  ct.mn ct.sd std.eff.sz   stat     p    ks ks.pval
covA  0.483 0.254  0.490 0.270     -0.025 -0.130 0.897 0.114   0.802
ps    0.349 0.212  0.344 0.203      0.022  0.105 0.916 0.141   0.566
```

Again, the weighting seems to have done a pretty good job – dropping the maximum standardized difference of 92.6% down to 18.7%, and balancing the propensity score well.

In fact, we can build a whole table of balance summaries just as we did before. If we do, we find that the range of standardized differences is -14 to +19, and the range of variance ratios is 0.86 to 1.12 across our main set of 10 covariates.

Rubin's Rules 1 and 2 to Check Balance After ATE Weighting

To check both Rule 1 and Rule 2, we'll need the output from bal.table to describe the linear PS.

```
> bal.table(bal.wts2)
[[2]]
               tx.mn tx.sd  ct.mn ct.sd std.eff.sz
linps         -0.806 1.133 -0.864 1.193      0.050
```

Rule 1. Again, we can read off the standardized effect size after weighting for the linear propensity score as 0.050. Multiplying by 100, we get 5.0%, so we would pass Rule 1.

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 variance.

```
> 1.133^2/1.193^2
[1] 0.9019427
```

Again, we see a substantial improvement over our unadjusted result, and now within our desired range of 4/5 to 5/4, as well as clearly within ½ to 2. We pass Rule 2, as well.

Rubin's Rule 3 requires some more substantial manipulation of the data. **I'll skip it.**

Question H. After weighting, estimate the causal effect of treatment vs control on…

Using the Weights via a Survey Design Tool

First, we arrange the weights (either ATT or ATE) into a survey design, then use the resulting weighted information to fit weighted linear regression models.

```
> library(survey)
> toywt1.design <- svydesign(ids=~1, weights=~wts1, data=toy)
     # using ATT weights
> toywt2.design <- svydesign(ids=~1, weights=~wts2, data=toy)
     # using ATE weights
```

…Outcome 1 (a continuous outcome)

ATT Weights: Linear Regression using svyglm

```
> 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
```

ATE Weights: Linear Regression using svyglm

```
> 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
```

…Outcome 2 (a binary outcome)

ATT Weights: Logistic Regression using svyglm and the quasibinomial family

We fit a logistic regression model, and exponentiate the log odds ratio treatment effect estimate to obtain an odds ratio estimate of the average causal effect of treatment.

```
> 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
```

ATE Weights: Logistic Regression using svyglm and the quasibinomial family

```
> 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
```

…Outcome 3 (a time-to-event outcome)

ATT Weights: Weighted Cox Proportional Hazards Model

```
> library(survival)
> 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

> cox.zph(adjout3.wt1); plot(cox.zph(adjout3.wt1),
     var="treated")
          rho chisq    p
treated 0.111  1.57 0.21
```

ATE Weights: Weighted Cox Proportional Hazards Model

```
> adjout3.wt2 <- coxph(Surv(out3.time, out2) ~ treated,
     data=toy, weights=wts2)
> summary(adjout3.wt2) ## exp(coef) output gives relative hazard
     of treated compared to control
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

> cox.zph(adjout3.wt2)
          rho chisq     p
treated 0.151  1.06 0.303
```

Thomas.Love@case.edu

Propensity Score Weighting Results

Causal Effect of Treatment Estimates, with (95% CI)	**Outcome 1** Cost difference	**Outcome 2** Risk Difference	**Outcome 2** Odds Ratio	**Outcome 3** Relative Hazard Rate
Propensity Score Weighting, Using ATT Weights	**15.2** (10.7, 19.8)	**N/A**	**1.48** (0.74, 2.94)	**0.82** (0.51, 1.32)
Propensity Score Weighing, Using ATE Weights	**7.0** (0.1, 14.0)	**N/A**	**1.29** (0.62, 2.67)	**0.83** (0.63, 1.11)

Question I. Estimate causal effect of treatment vs control (adjusting for propensity) on…

…Outcome 1 (a continuous outcome)

Linear Regression Model with linear PS 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
```

…Outcome 2 (a binary outcome)

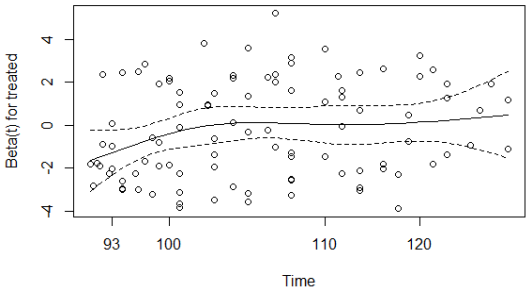
Logistic Regression Model with linear PS as a covariate

```
> 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)) # produced 95% CI for odds ratio

                2.5 %   97.5 %
(Intercept) 0.5977473 1.508760
treated     0.7097588 2.580375
linps       0.8902357 1.487135
```

…Outcome 3 (a time-to-event outcome)

Cox proportional hazards Model with linear PS as a covariate

```
> library(survival)
> 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


> 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")
```



Question J. Double Robust [Weight then Adjust] …

To do a double robust analysis of this form, we simply add the linear propensity score to the right hand side of the outcome regression models we built after weighting.

Using the Weights via a Survey Design Tool

Again, we arrange the weights (either ATT or ATE) into a survey design, then use the resulting weighted information to fit weighted linear regression models.

```
> library(survey)
> toywt1.design <- svydesign(ids=~1, weights=~wts1, data=toy)
     # using ATT weights
> toywt2.design <- svydesign(ids=~1, weights=~wts2, data=toy)
     # using ATE weights
```

…Outcome 1 (a continuous outcome)

ATT Weights: Linear Regression with survey weights adjusting for Linear PS

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

ATE Weights: Linear Regression with survey weights adjusting for Linear PS

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

…Outcome 2 (a binary outcome)

ATT Weights: Logistic Regression adjusting for Linear PS

We fit a logistic regression model, and exponentiate the log odds ratio treatment effect estimate to obtain an odds ratio estimate of the average causal effect of treatment.

```
> adjout2.wt1 <- svyglm(out2 ~ treated + linps,
     design=toywt1.design, family=quasibinomial())
> summary(adjout2.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(adjout2.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(adjout2.wt1))
                2.5 %   97.5 %
(Intercept) 0.5149615 1.531727
treated     0.7291363 2.942405
linps       0.8385347 1.874632
```

ATE Weights: Logistic Regression adjusting for Linear PS

```
> adjout2.wt2 <- svyglm(out2.event ~ treated + linps,
     design=toywt2.design, family=quasibinomial())
> summary(adjout2.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(adjout2.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(adjout2.wt2))
                2.5 %   97.5 %
(Intercept) 0.5243509 1.522994
treated     0.6057234 2.720375
linps       0.7414934 1.643802
```

…Outcome 3 (a time-to-event outcome)

ATT Weights: Weighted Cox Model adjusting for Linear PS

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

> cox.zph(adjout3.wt1)
            rho chisq     p
treated  0.1151 1.539 0.215
linps   -0.0332 0.298 0.585
GLOBAL       NA 1.759 0.415
```

ATE Weights: Weighted Cox Model adjusting for Linear PS

```
> adjout3.wt2 <- coxph(Surv(out3.time, out2) ~ treated,
     data=toy, weights=wts2)
> summary(adjout3.wt2) ## exp(coef) output gives relative hazard
     of treated compared to control
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

> cox.zph(adjout3.wt2)
           rho chisq     p
treated  0.173  1.27 0.260
linps   -0.136  1.02 0.313
GLOBAL      NA  2.08 0.353
```

Double Robust (Propensity Score Weighting, then Adjustment) Results

Causal Effect of Treatment Estimates, with (95% CI)	**Outcome 1** Cost difference	**Outcome 2** Risk Difference	Odds Ratio	**Outcome 3** Relative Hazard Rate
Propensity Score Weighting + Adjustment, ATT Weights	**14.8** (10.6, 19.0)	**N/A**	**1.46** (0.73, 2.94)	**0.81** (0.50, 1.31)
Propensity Score Weighting + Adjustment, ATE Weights	**6.7** (1.6, 11.7)	**N/A**	**1.28** (0.61, 2.72)	**0.83** (0.63, 1.11)

Question K. Summarize results – which approach might be best…

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**, c**ovF[middle]**, **covF[high]**, **A squared**, **B*C** and **B*D**…

Approach	Standardized Differences Across 10 covariates	Variance Ratios Across 10 covariates
Most Desirable Values	Between -10 and +10	Between 0.8 and 1.25
No Adjustments	-30 to 63	0.59 to 1.27
1:1 Propensity Matching	-13 to 20	0.62 to 1.23
Propensity 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
Propensity Weighting, ATT	-6 to 13	0.64 to 1.20
Propensity Weighting, ATE	-14 to 19	0.86 to 1.12

Quality of Balance as Assessed by Rubin's Rules

Approach	Rubin's Rule 1 (mean bias)	Rubin's Rule 2 (var. ratio)	Rubin's Rule 3 (residual var.ratio)
"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
1:1 Propensity Matching	37	1.42	0.58 to 1.21
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
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 Rules could also be applied after weighting on the propensity score.

Final Summary of All Results regarding Outcomes

Causal Effect of Treatment Estimates, with (95% CI)	**Outcome 1** Cost difference	**Outcome 2** Risk Difference	**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 Propensity Match (Match's Automatic ATT Results)	**15.6** (11.6, 19.6)	**+0.10** (-0.05, +0.25)	**N/A**	**N/A**
After 1:1 Propensity Match (Match's Automatic ATE Results)	**16.0 (13.0, 19.1)**	**+0.09** (-0.02, +0.19)		
After 1:1 Propensity Match (Detailed "Regression" Models)	**15.6** (11.5, 19.6)	**N/A**	**1.64** (0.77, 3.47)	**0.75** (0.41, 1.38)
After Stratification on the Quintiles of the Linear PS	**7.9** (4.1, 11.7)	**+0.03** (-0.29, 0.36)	**1.15** (0.54, 2.44)	**0.79** (0.49, 1.27)
Propensity Score Weighting, Using ATT Weights	**15.2** (10.7, 19.8)	**N/A**	**1.48** (0.74, 2.94)	**0.82** (0.51, 1.32)
Propensity Score Weighing, Using ATE Weights	**7.0** (0.1, 14.0)		**1.29** (0.62, 2.67)	**0.83** (0.63, 1.11)
After Direct Adjustment for the Linear Propensity Score	**12.5** (8.6, 16.4)		**1.35** (0.71, 2.58)	**0.80** (0.50, 1.26)
Double Robust: Weighting via ATT Weights then Adjustment	**14.8** (10.6, 19.0)		**1.46** (0.73, 2.94)	**0.81** (0.50, 1.31)
Double Robust: Weighting via ATE Weights then Adjustment	**6.7** (1.6, 11.7)		**1.28** (0.61, 2.72)	**0.83** (0.63, 1.11)

Again, your decision between these methods should be made before you look at the outcome results, on the basis of the quality of balance after the propensity score has been applied.

As mentioned, in this case, as in most settings, to obtain an ATT estimate, I would use either propensity matching (though not necessarily the 1:1 approach shown here – other approaches might lead to better balance results) or propensity weighting.