Class 5—Political Science 531
Controlling For, Matrices, Good Tests!

Jake Bowers

February 25, 2013

Covariance Adjustment: Residualization Approximating Stratification

As of last time we learned both about one criteria for assessing estimators (bias) and one criteria for assessing methods of statistical inference (like hypothesis tests and confidence intervals) (false positive rate or Type I error rate). Let us now turn our attention to the question of "Controlling for".

1. Load data and functions:

```
source("http://jakebowers.org/PS531Data/ps531fns.R")
load(url("http://jakebowers.org/PS531Data/chile90.rda"))  ## Using the Chile 1990 Survey again
library(car)
```

Some recoding here:

```
## milworry=1 if mentioned worry in either variable P202 or P201 about conflicts with the military, 0 otherwise This trick relies
## on the fact that R converts TRUE to 1 and FALSE to 0 when appropriate.
chile90$milworry <- (chile90$P202 == 6) + (chile90$P201 == 6)
table(chile90$milworry, useNA = "ifany")


  0   1
853 334

chile90$age3 <- car::recode(chile90$P26, "1:2='18-34';3:4='35-54';5:6='55ymas'", as.factor.result = TRUE)
```

2. Use the full dataset (`chile90`) to estimate the difference of means in `milworry` between the youngest group and the oldest group in the survey. What story might we tell to make sense of this relationship?

```
(fit1 <- fit.please(milworry ~ age3, thedata = chile90))

              [,1]
(Intercept)  0.308
age335-54   -0.024
age355ymas  -0.107


## Another way to show that the least squares estimates are just means and differences of means The with() is also a nice way to
## get some of the benefits of attach().  See also within() for recoding and such
with(chile90, tapply(milworry, age3, mean))

 18-34  35-54 55ymas
 0.308  0.284  0.201

## or
tapply(chile90$milworry, chile90$age3, mean)

 18-34  35-54 55ymas
 0.308  0.284  0.201


diff(c(0, with(chile90, tapply(milworry, age3, mean))))

  18-34   35-54  55ymas
 0.3081 -0.0240 -0.0831
```

About 32% of the youngest group is worried about near term conflict with the military whereas about 32-14=18% of the oldest group reports such worry.

Why could this be? How about a hunch about the willingness of people who have just spent about 20 years of their lives under an authoritarian dictatorship to talk about the military government to strangers (i.e. survey interviewers)? The older group lived through and suffered the transition to authoritarianism and related violence most directly. The younger group basically don't remember it. Thus, the younger group may be less guarded/feel less fear and be more likely to more honestly express themselves. Just a made up hunch here but it would be supported by the work of Timur Kuran, Vaclav Havel, among others on civil society during and after authoritarian/totalitarian regimes.

3. Upon hearing your story, what might a critic say? What alternative explanation might she propose? (I am not asking you to delve into the codebook to assess this explanation here since I don't assume you all know Spanish! But I wanted you to practice thinking about alternative explanations for relationships that you've uncovered.)

   Maybe this relationship is real: since the question was about worry and fear, the old might be telling us that they just worry less about it even if they might judge the probability of another coup as the same as the younger group — i.e. the younger group have more to lose from a regime transition since they have a longer future and thus they ought to worry more.

4. How about this for yet another alternative explanation: People of different ages had very different experiences with the military regime that took power in 1973 and ruled until 1989: the older folks might, in fact, be deeply divided into two groups — one of which supported the military regime (and perhaps directly benefited from it) and the other of which did not (and perhaps directly suffered from it). And what we want here is a comparison among age groups uncontaminated by the ideology of the person — perhaps to reflect something else about age. So, now, let's "control for" or "adjust" our estimate of the difference between the oldest and the youngest for ideology. Choose either of the two variables recoded below (and make sure to recode them as I did, or some other way that makes sense to you). Now, estimate the mean difference that you did above, but now adjusted for one of the two measures of ideology. Now, what do you find? Interpret the results. *Hint:* Use `fit.please()` or `lm()`.

```
  ## P11L: p11l De 1 a 7 una opinion favorable o desfavorable (A.PINOCHET)
## '1' 1 '2' 2 '3' 3 '4' 4 '5' 5 '6' 6 '7' 7 '9' NO SABE/NO CONTESTA
## 11. Las siguientes personas tienen o han tenido diversos grados de
## notoriedad en la vida publica nacional. Indique con una nota de 1 a
## 7 el grado en que Ud. tiene una opinion favorable segun esta escala
## de evaluacion (TRATE DE USAR TODOS LOS RANGOS DE LA ESCALA, NO SOLO
## LOS EXTREMOS) (ENTREVISTADOR: SI CONTESTA NO CONOCE MARCAR
## 9. PARTA ALTERNADAMENTE DE ARRIBA HACIA ABAJO O DE ABAJO HACIA
## ARRIBA, O DEL CENTRO, COMPLETANDO TODA LA LISTA. INDIQUE POR DONDE
## PARTIO) PARTIDA

## Note: In Chilean schools a 7=A and 1=F. And a "grade" is a "nota".
## So, the respondents are being asked to give Pinochet a grade.
```

```
chile90$like.pinocho <- as.numeric(chile90$P11L > 3)  ## 1 if likes Pinochet, 0 if not.
```

```
## OR

## p22 Con Cual posicion identifica o simpatiza Ud. mas?
## '1' DERECHA
## '2'   CENTRO DERECHA
## '3' CENTRO
## '4'   CENTRO IZQUIERDA
## '5' IZQUIERDA
## '6' INDEPENDIENTE
## '7' NINGUNA
## '8'   NO SABE
```

```
chile90$ideology <- ifelse(chile90$P22 == 8, NA, chile90$P22)  ## Code DK as missing for now
chile90$right.wing <- as.numeric(chile90$ideology <= 2)
```

```
(fit2 <- fit.please(milworry ~ age3 + right.wing, thedata = chile90))

                [,1]
(Intercept)  0.3266
age335-54   -0.0249
age355ymas  -0.1059
right.wing  -0.1135

## compare to fit1
fit1
```

```
              [,1]
(Intercept)  0.308
age335-54   -0.024
age355ymas  -0.107
```

“Controlling for” right-wing self-identification does not seem to make much difference in the estimated proportion of people who worry about the military. Right-wingers are less worried about the military than center- and left-wingers.

5. Now, let’s simplify the model:

```
chile90$oldest <- ifelse(chile90$age3 == "55ymas", 1, 0)
table(chile90$oldest, chile90$age3, exclude = c())


       18-34 35-54 55ymas <NA>
  0      555   433      0    0
  1        0     0    199    0
  <NA>     0     0      0    0


(fit3 <- lm(milworry ~ oldest + right.wing, data = chile90))


Call:
lm(formula = milworry ~ oldest + right.wing, data = chile90)

Coefficients:
(Intercept)       oldest   right.wing
     0.3157      -0.0951      -0.1129
```

And, let’s make sure that we understand what is going on here. I’m going to propose an alternative method that I claim will give you the same answer as above [this is tricky, see the hint]: (1) regress `milworry` on your ideology variable; (2) record the residuals from the `milworry~your.ideology` regression — these represent worry with the mean differences in worry by ideology removed (i.e. a form of mean deviation); (3) regress `oldest` on your chosen ideology variable ; (4) record the residuals from the `oldest~your.ideology.var` (these are age with mean differences in age due to ideology); (5) regress the residuals from `milworry~your.ideology` on the residuals from `oldest~your.ideology.var`. How might you explain the residuals from `milworry~your.ideology` regression? What is the scale of those residuals? *Hint:* Use `lm()` rather than `fit.please()` for this. I recommend using `resid()` on the `lm()` objects to get the residuals quickly and easily without having to calculate predicted values and then subtracting them from the observed values. Here are some recommended ways to run these regressions. Make sure you understand what is going on with these lines of code [swap your own ideology measure for “right.wing” if you want].

```
(fit3 <- fit.please(milworry ~ oldest + right.wing, thedata = chile90))

## Now here I use lm() because it has a nice residuals() function already defined for it.

(mwrw.fit <- lm(milworry ~ right.wing, data = chile90))
(agerw.fit <- lm(oldest ~ right.wing, data = chile90))

mwrw.e <- resid(mwrw.fit)
agerw.e <- resid(agerw.fit)

lm(mwrw.e ~ agerw.e)
```

Notice that we got the same coefficient of roughly .126 from this “residualization” procedure as we did using multiple regression (using either `lm()` or `fit.please()`).

Now, let’s simplify this a bit. I think we got wrapped up in the subsetting and such before — all of which is good R practice but which may have obscured the statistical points. So, first I’m just going to look at a smaller version of the chile90 dataset — a version excluding the folks with missing values on right.wing and folks in the 25–64 age range

```
## Make a new dataset only for the youngest and oldest groups, and excluding missing values from the right.wing variable
chile90small <- subset(chile90, subset = age3 != "25-64" & !is.na(right.wing), select = c(milworry, age3, oldest, right.wing), drop
## The following command is equivalent to the subset() command chile90small<-chile90[chile90$age3!='25-64'&
## !is.na(chile90$right.wing), c('milworry','age3','right.wing'), drop=TRUE]

## Check our subsetting:
summary(chile90small)
```

```
    milworry          age3          oldest         right.wing
 Min.   :0.000   18-34 :541   Min.   :0.000   Min.   :0.000
 1st Qu.:0.000   35-54 :419   1st Qu.:0.000   1st Qu.:0.000
 Median :0.000   55ymas:184   Median :0.000   Median :0.000
 Mean   :0.287                Mean   :0.161   Mean   :0.122
 3rd Qu.:1.000                3rd Qu.:0.000   3rd Qu.:0.000
 Max.   :1.000                Max.   :1.000   Max.   :1.000

## Annoying that the middle level of the factor lingers despite efforts to drop=TRUE it!
chile90small$age3 <- factor(chile90small$age3)
summary(chile90small)

    milworry          age3          oldest         right.wing
 Min.   :0.000   18-34 :541   Min.   :0.000   Min.   :0.000
 1st Qu.:0.000   35-54 :419   1st Qu.:0.000   1st Qu.:0.000
 Median :0.000   55ymas:184   Median :0.000   Median :0.000
 Mean   :0.287                Mean   :0.161   Mean   :0.122
 3rd Qu.:1.000                3rd Qu.:0.000   3rd Qu.:0.000
 Max.   :1.000                Max.   :1.000   Max.   :1.000
```

Now do the two fits: the one looking at the mean difference in worry by age, and the other “controlling for” right.wing identification.

The difference in coefficient estimates between fit1 and fit1a arises from excluding the missing right.wing folks. The difference in estimates on right.wing between fit2 and fit2a arises from excluding the middle group Notice no difference in the estimate for the age effect between fit2 and fit2a

Calculate mean proportions of worry by the different age groups (i.e. the proportion for the excluded category of age and the differences between the excluded category and the other categories for the other categories)

```
(fit1a <- fit.please(milworry ~ age3, thedata = chile90small))

               [,1]
(Intercept)  0.3124
age335-54   -0.0236
age355ymas  -0.1059


## Now, 'control for' right.wing.
(fit2a <- fit.please(milworry ~ age3 + right.wing, thedata = chile90small))

               [,1]
(Intercept)  0.3266
age335-54   -0.0249
age355ymas  -0.1059
right.wing  -0.1135


## Now calculate differences of means:
(worrybyage.means <- with(chile90small, tapply(milworry, age3 == "55ymas", mean)))

FALSE  TRUE
0.302 0.207


(worrybyrw.means <- with(chile90small, tapply(milworry, right.wing, mean, na.rm = TRUE)))

    0     1
0.300 0.187


(agebyrw.means <- with(chile90small, tapply(age3 == "55ymas", right.wing, mean, na.rm = TRUE)))

    0     1
0.160 0.165


## Diff of means in worry due to age [youngest vs. oldest] (equiv to fit1a)
(unadjdiff <- worrybyage.means[2] - worrybyage.means[1])

   TRUE
-0.0956

## I'm calling this the 'unadjusted' difference since this is the difference of substantive concern.  How much of this difference
## is due to right-wing?
## Diff of means in worry due to right-wing [left/center vs. right] (equiv to lm(milworry~right.wing,data=chile90small))
worrybyrw.means[2] - worrybyrw.means[1]

     1
-0.113
```

So, we see, yet again, that the regression is giving us differences of means. But the idea of "adjusting" one difference of means for another one (or set of them) is where the real power of the linear model comes from and now I'll try to show a more direct (but more computationally burdensome) route that doesn't require least squares fitting at all.

First, subtract off the the extent to which the means in milworry differ by right.wing. This creates something I'm calling `mymwresid`.

```
## Here, I've got mean worry by right.wing from observed milworry from each person.
mymwresid <- with(chile90small, milworry - worrybyrw.means[as.character(right.wing)])
table(mymwresid)

mymwresid
-0.300497512437811  -0.18705035971223  0.699502487562189   0.81294964028777
               703                113                302                 26

## Notice that rather than just two values, we now have four:
with(chile90small, table(milworry, right.wing))

        right.wing
milworry   0   1
       0 703 113
       1 302  26


## And that this is the same as taking the residuals from the mwrw.fit regression
table(signif(resid(mwrw.fit), 5))


 -0.3005 -0.18705   0.6995  0.81295
     703      113      302       26
```

For, for folks with no worry but who are right.wing, we subtract off .179, but for folks without worry, but who are not right.wing, we substract off .307.

This, is what "worry without the effect of right.wing" or "worry without right.wing" means in the context of least squares:

Some of this difference of .127 due to right.wing is due to age. So, subtracting off this amount would be too much. We only want to take off the difference due to right.wing that is not related to age. So, how much of the difference in proportion "old" is due to right. wing? [Seems very strange to put it that way, huh? Not like announcing right.wing ideology (or center or left-wing ideology) can change your age. But, rather, we are asking how much ideology appears related to age — and by "related" we mean something very specific — the difference in means of proportion "old" between "right wing" and "not right wing" people.

```
## Diff of means in age due to right-wing.  Right-wing folks are older [30% are in the oldest group versus 27% in the youngest
## group] (equiv to lm(I(age3=='55ymas')~right.wing,data=chile90small))
agebyrw.means[2] - agebyrw.means[1]

      1
0.00527

myageresid <- with(chile90small, (age3 == "55ymas") - agebyrw.means[as.character(right.wing)])

table(myageresid)

myageresid
-0.165467625899281 -0.160199004975124  0.834532374100719  0.839800995024876
               116                844                 23                161

## or
table(signif(agerw.e, 5))


-0.16547  -0.1602  0.83453   0.8398
     116      844       23      161

## Notice that rather than just two values, we now have four:
with(chile90small, table((age3 == "55ymas"), right.wing))
```

```
      right.wing
         0   1
  FALSE 844 116
  TRUE  161  23
```

So, a small part of the mean difference in age can be due to differences in right.wingedness [where "due to" is not causal but merely descriptive of the sizes of mean differences].

Now, we can regress those residuals on each other to get the slope that we care about just as we did above:

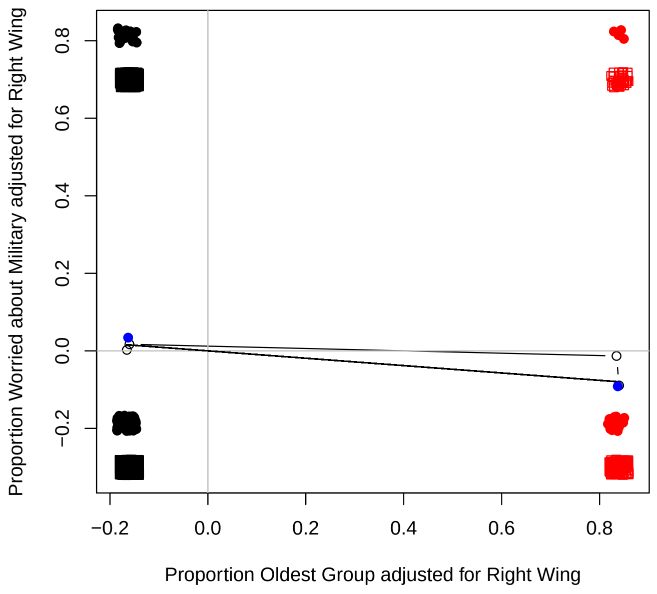
```
## Using lm() because fit.please() demands a dataset
thelm <- lm(mymwresid ~ myageresid - 1)  ## Excluding intercept because I know it is zero
coef(thelm)

myageresid
   -0.0951
```

Now, notice that we can do this adjustment without least squares:

```
## Mean of milworry for each value of age [both residualized for right.wing]
(therwadjmeans <- tapply(mymwresid, myageresid, mean))

-0.165467625899281 -0.160199004975124  0.834532374100719  0.839800995024876
            0.0026             0.0170            -0.0131            -0.0893


## Now, the least squares line smooths over those means just calculated.  See the next plot: Least squares is like taking a
## weighted mean of the points for the young [the black points] versus the old [the red points]

## The weights are proportionate to the number of people at each value:
(therwadjns <- tapply(mymwresid, myageresid, length))

-0.165467625899281 -0.160199004975124  0.834532374100719  0.839800995024876
               116                844                 23                161


## Here are the weighted mean calculations.  The adjusted worry among the young is a weighted average of the two means among the
## young [where each mean is separately calculated for the right.wing==1 and right.wing==0].

## So the adjusted worry among the young is:
(adj.worry.young <- (-0.0313 * (27/(27 + 203))) + (0.0429 * (203/(27 + 203))))

[1] 0.0342


## And among the old it is:
(adj.worry.old <- (0.0705 * (12/(12 + 74))) + (-0.1177 * (74/(12 + 74))))

[1] -0.0914


adj.worry.old - adj.worry.young

[1] -0.126

## And recall the least squares result from regressing residuals on residuals:
coef(thelm)

myageresid
   -0.0951

## Or the multiple regression result:
fit2a

               [,1]
(Intercept)  0.3266
age335-54   -0.0249
age355ymas  -0.1059
right.wing  -0.1135
```

Now the plot comparing the least squares line (the straight sloped black line), the line of means (the broken line with open circles), and the two adjusted means (in blue).

```
## I jitter the points to show how many people are in each group
## This makes the weighting that produces the least squares lines more clear
plot(jitter(myageresid,amount=.02),
     jitter(mymwresid,amount=.02),
     col=c("black","red")[(chile90small$age3=="55ymas")+1], # red for old
     pch=c(0,19)[(chile90small$right.wing==1)+1], # filled circle for right.wing
     xlab="Proportion Oldest Group adjusted for Right Wing",
     ylab="Proportion Worried about Military adjusted for Right Wing"
     )

## Now, plot the line of means --- this is the object of presumed substantive interest,
## the object which will be smoothed.
## Notice that "means" in this case, are "proportions" since
## a mean of a binary variable is a proportion of 1s.
lines(as.numeric(names(therwadjmeans)),
      therwadjmeans,type="b")

## Notice that because these are both residuals (i.e. observed minus
## some summary, here a mean), they both have means of zero:
summary(myageresid)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -0.165  -0.160  -0.160   0.000  -0.160   0.840

summary(mymwresid)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -0.300  -0.300  -0.300   0.000   0.700   0.813

abline(v=0,h=0,lty=1,col="gray") ## Mark the means.

## This is the line from the least squares fit.
lines(fitted(thelm)~myageresid)

## Now, add the points that we calculated directly from the adjusted
## means.
points(mean(sort(unique(myageresid))[1:2]),
       adj.worry.young,pch=19,col="blue")
points(mean(sort(unique(myageresid))[3:4]),
       adj.worry.old,pch=19,col="blue")
```



So, we've seen how to "control for" ideology using multiple regression, and what this means: it means "taking away" the "relationship" between right.wing and worry about the military and the relationship between right.wing and age and then assessing the relationship between "worry without right.wing" and "age without right.wing". By "relationship" we've meant "difference of means" (and this has been made particularly clear because we've used binary variables — with continuous variables, "relationship" is the "smoothed difference of means" where the smoothing is done by least squares). And by "taking away" we've meant adjusting the values of the variables (subtracting means from them) such that the new variable would have no difference of means when compared with the adjustor or control variable.

This seems very complicated, and it is, if you do it by hand as we've done it here. What is cool about least squares is that it happens automagically and nearly instantaneously for any number of controls.

6. Now, did this residualization do what we really wanted it to do? Try "holding constant" more directly: fit your model separately for each subgroup or strata on your ideology measure. Interpret the results of the two regressions. *Hint:* You can either use the `subset` argument to `lm()` or directly select the appropriate rows of the the chile90 dataframe for use in `fit.please()` by doing something like `chile90[my.ideology.var==somevalue,]`.

```
(fit.rw1 <- fit.please(milworry ~ age3, thedata = chile90, subset = right.wing == 1))

              [,1]
(Intercept) 0.1618
age335-54   0.0674
age355ymas  0.0121

(fit.rw0 <- fit.please(milworry ~ age3, thedata = chile90, subset = right.wing == 0))

               [,1]
(Intercept)  0.3340
age335-54   -0.0375
age355ymas  -0.1229
```

Wow! "Controlling for" didn't do what one might have thought. The older folks are *more* worried than the younger folks among the right.wing identifiers but *less* worried than the younger folks among the left/center-identifiers! The hidden assumption that ideology works the same way in the old as in the young [or equivalently, the assumption that age and worry about the military work the same way regardless of ideology] was incorrect.

7. So you've seen that `fit.please()` automagically does this regressing on residuals stuff (or optimally choosing coefficients to minimize sums of squared residuals). It turns out, that it can also handle this stratification or subgroup version of "controlling for" too. Try fitting your model using the following formula (adjusted for your own ideology variable): `milworry~oldest*your.ideology.var`. You should be able to recover the very same numbers from this regression as you did when you stratified. Interpret these results: what is the proportion who worry about the military among the old,right-wingers? what is the proportion who worry about the military among the young, right-wingers?

```
(fit4 <- fit.please(milworry ~ age3 * right.wing, thedata = chile90)[, 1])

          (Intercept)             age335-54            age355ymas            right.wing  age335-54:right.wing age355ymas:right.win
               0.3340               -0.0375               -0.1229               -0.1723                0.1049                0.135

## Proportion worried among youngest/not-right.wing
fit4["(Intercept)"]

(Intercept)
      0.334

## Proportion worried among youngest/right.wing
fit4["(Intercept)"] + fit4["right.wing"]

(Intercept)
      0.162

## Proportion worried among the oldest/right.wing
fit4["(Intercept)"] + fit4["age355ymas"] + fit4["right.wing"] + fit4["age355ymas:right.wing"]

(Intercept)
      0.174


## Difference in proportion worried between the oldest/right.wing and the youngest/not-right.wing
(fit4["(Intercept)"] + fit4["age355ymas"] + fit4["right.wing"] + fit4["age355ymas:right.wing"]) - (fit4["(Intercept)"])

(Intercept)
      -0.16

## same as
fit4["age355ymas"] + fit4["right.wing"] + fit4["age355ymas:right.wing"]

age355ymas
     -0.16
```

```
## Difference in proportion worried between the oldest/right.wing and youngest/right.wing
(fit4["(Intercept)"] + fit4["age355ymas"] + fit4["right.wing"] + fit4["age355ymas:right.wing"]) - (fit4["(Intercept)"] + fit4["rig

(Intercept)
     0.0121

## same as
fit4["age355ymas"] + fit4["age355ymas:right.wing"]

age355ymas
    0.0121


## Difference in proportion worried between oldest/right.wing and oldest/not-right wing
(fit4["(Intercept)"] + fit4["age355ymas"] + fit4["right.wing"] + fit4["age355ymas:right.wing"]) - (fit4["(Intercept)"] + fit4["age3

(Intercept)
    -0.0373

## Same as
fit4["right.wing"] + fit4["age355ymas:right.wing"]

right.wing
   -0.0373
```

So, we see the same results that we saw when we estimated the mean differences in worry separately for subgroups defined by right.wing: about 10% more of the older group is worried about the military among the right.wingers than the younger group.

This is often called an "interaction term" regression, or "interactions with dummy variables". It is a way to trick least squares into doing the subgroup-specific or stratified estimation all in one step: it works the same as the subgroup/stratification/subsetting approach when the subgroups are all binary or categorical. They are continuous, then `x*continuous.var` is not the same as `subset=continuous.var==somevalue` since the later breaks the continous variable into parts and the former assumes a smooth and linear interaction.

8. How much more prevalent is worry about the military among the oldest, right-wing group than it is among the youngest, right-wing group? *Hint:* First, you'll need to add coefficients to get the estimated proportion of worry about the military among each group. Second, you'll have to subtract those estimates (try subtracting the youngest group from the oldest group). Is there a simpler way to get this difference? What is it?

   See above.

9. What do you conclude about the interaction of ideology and age as they matter for fears about the military in newly post-transition Chile?

   The older right wingers are more afraid than the young right wingers. The older left wingers are less afraid than the young left wingers. But, the left wingers are more afraid than the right wingers in general.

10. Finally, did we really succeed in "controlling for" when we used the "covariance adjustment" method? Produce a plot or table assessing the extent to which the results depend on the linear functional form (and thus interpolation and/or extrapolation as explained in the Gelman and Hill reading) versus are supported by data. *Hint:* I think that here a table would be fine and more informative than a plot since we are not "controlling for" a continuous covariate as they did in the Gelman and Hill reading.

    When we use the full dataset, we do have data support for all of the relevant combinations. When we use the 100 person dataset, we cannot claim to control for "right.wing" for the 35-54 year old group at all (none of them are right.wing), yet the regression still provides us numbers (because we assumed a global straight line fit).

```
set.seed(20100205)
library(mosaic)
chile90$ideology <- ifelse(chile90$P22 == 8, NA, chile90$P22)
chile90s <- resample(chile90[, c("age3", "milworry", "ideology")], size = 100, replace = FALSE)
```

```
chile90s$right.wing <- as.numeric(chile90s$ideology <= 2)

lm(milworry ~ age3 + right.wing, data = chile90s)


Call:
lm(formula = milworry ~ age3 + right.wing, data = chile90s)
```

```
Coefficients:
(Intercept)    age335-54   age355ymas   right.wing
      0.367       -0.160       -0.154       -0.136

## But
with(chile90s, ftable(milworry, age3, right.wing, col.vars = "milworry", row.vars = c("right.wing", "age3")))

                  milworry  0  1
right.wing age3
0          18-34           24 15
           35-54           23  6
           55ymas          18  4
1          18-34            3  0
           35-54            0  0
           55ymas           3  1


lm(milworry ~ age3 + right.wing, data = chile90)


Call:
lm(formula = milworry ~ age3 + right.wing, data = chile90)

Coefficients:
(Intercept)    age335-54   age355ymas   right.wing
     0.3266      -0.0249      -0.1059      -0.1135

## But
with(chile90, ftable(milworry, age3, right.wing, col.vars = "milworry", row.vars = c("right.wing", "age3")))

                  milworry   0   1
right.wing age3
0          18-34           315 158
           35-54           261 110
           55ymas          127  34
1          18-34            57  11
           35-54            37  11
           55ymas           19   4
```

11. Would this problem get worse or better as we add controls? Or if we had smaller samples?

    Worse in each case.

12. **Extra practice with bootstrap and interactions:** Use the bootstrap to produce a simple 95% percentile confidence interval for this difference. Plot your bootstrap estimated/approximated sampling distribution for this difference. *Hint:* Bootstrap the entire regression but now your addition and subtraction will be done on columns of the results matrix.

```
## First see how to get the value of interest:
fit5 <- coef(lm(milworry ~ age3 * right.wing, data = chile90))
fit5["age355ymas"] + fit5["age355ymas:right.wing"]  ## this sum of coefs is what we want

age355ymas
    0.0121
```

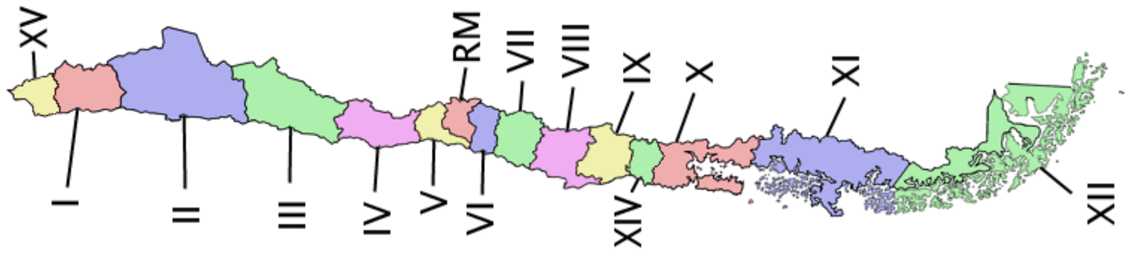
```
newfit.bs <- do(500) * lm(milworry ~ age3 * right.wing, data = resample(chile90))
```

```
summary(newfit.bs$age355ymas + newfit.bs$"age355ymas:right.wing")

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.2290 -0.0562  0.0073  0.0086  0.0711  0.3100


## The simple percentile bootstrap confidence interval
(bs.ci <- quantile(newfit.bs$age355ymas + newfit.bs$"age355ymas:right.wing", c(0.025, 0.975)))

  2.5%  97.5%
-0.162  0.200


## The estimated standard error of the sum of coefficients:
(bs.se <- sd(newfit.bs$age355ymas + newfit.bs$"age355ymas:right.wing"))

[1] 0.0935
```

The estimated sampling distribution of the sum of the two coefficients is shown below (after the stratified sampling bootstrap).

So, the difference could plausibly range between -.2 and 4. OR, we wouldn't reject the null of no mean difference at the $\alpha = .05$ significance level (i.e. that null would have a p-value of greater than or equal to .05, making that null a relatively plausible story for our data).

13. **Extra practice with bootstrap and interactions and design:** Now, close inspection of the codebook tells us that, in fact, the survey sampling was done separately in four places in the country. First, the researchers chose 4 metro areas — not a sample of metro areas, just the 4 largest ones. Then, they sampled within those areas. What should we do to change our resampling procedure given this information? Now, let's create a simple 95% confidence interval for the proportion of the left-wing, youngest group worried about the military using this new, more appropriate sampling plan. Any difference? Why might there be a difference between the sampling distribution for the intercept taking into account the actual design and not? *Hint:* Use `resample(chile90,within=chile90$REGION)`. To think about why there might be differences across the regions look at `prop.table(table(chile90$milworry,chile90$REGION),margin=2)`. In Chile, by the way, 8th Region (Concepcion and Talcahuano) has long been a hot-bed of leftism and labor activism where as the North (the 2nd region) has tended to be more right-wing (despite plenty of copper and nitrate-mining labor activism there, too). (RM=Region 13).



```
chile90$regionF <- factor(chile90$REGION)
newfit.stratA <- do(500) * lm(milworry ~ age3 * right.wing, data = resample(chile90, within = chile90$REGION))
```

```
(bs.stratA.ci <- quantile(newfit.stratA$age355ymas + newfit.stratA$"age355ymas:right.wing", c(0.025, 0.975)))

  2.5%  97.5%
-0.150  0.189


## The estimated standard error of the sum of coefficients:
(bs.stratA.se <- sd(newfit.stratA$age355ymas + newfit.stratA$"age355ymas:right.wing"))

[1] 0.0906


bs.ci

  2.5%  97.5%
-0.162  0.200

bs.se

[1] 0.0935
```

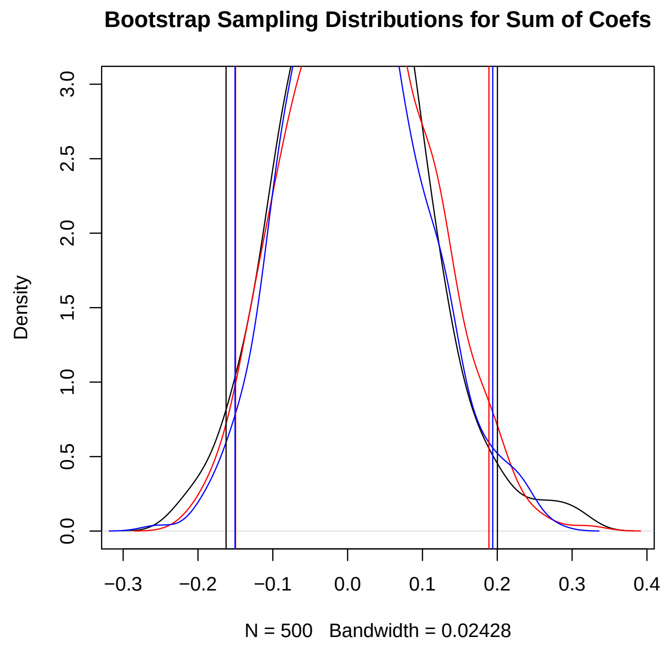
Notice that including the information about the research design has some effect (a small one here) on the statistical inference: namely producing a sampling distribution that is narrower. Now, if we thought that worry about the military varied much across the regions, we'd also want to adjust for region in the regression by allowing different intercepts for each region:

```
(fit.region <- lm(milworry ~ age3 * right.wing + regionF, data = chile90))


Call:
lm(formula = milworry ~ age3 * right.wing + regionF, data = chile90)

Coefficients:
          (Intercept)             age335-54            age355ymas            right.wing              regionF5
               0.3307               -0.0406               -0.1204               -0.1675                0.0398
             regionF8              regionF13  age335-54:right.wing  age355ymas:right.wing
               0.0667               -0.0512                0.0994                0.1286
```

```
newfit.stratB <- do(500) * lm(milworry ~ age3 * right.wing + regionF, data = resample(chile90, within = chile90$REGION))
```

```
## ## The simple percentile bootstrap confidence interval
(bs.stratB.ci <- quantile(newfit.stratB$age355ymas + newfit.stratB$"age355ymas:right.wing", c(0.025, 0.975), na.rm = TRUE))

  2.5%  97.5%
-0.150  0.194

## ## The estimated standard error of the sum of coefficients:
(bs.stratB.se <- sd(newfit.stratB$age355ymas + newfit.stratB$"age355ymas:right.wing", na.rm = TRUE))

[1] 0.0855
```

```
plot(density(newfit.bs$age355ymas + newfit.bs$"age355ymas:right.wing"), ylim = c(0, 3), main = "Bootstrap Sampling Distributions fo
lines(density(newfit.stratA$age355ymas + newfit.stratA$"age355ymas:right.wing"), col = "red")
lines(density(newfit.stratB$age355ymas + newfit.stratB$"age355ymas:right.wing"), col = "blue")

abline(v = bs.ci)
abline(v = bs.stratA.ci, col = "red")
abline(v = bs.stratB.ci, col = "blue")
```



So, taking the actual sampling design into account improved our statistical inference slightly, but adjusting the estimates for region did not appreciably make our confidence intervals shorter (or the standard error of our estimates smaller).

The Joy of Matrices

Over the last couple of weeks we've realized that we can represent what least-squares is doing in terms of differences of means, but, `fit.please()` doesn't obviously calculate differences of means.[1] We've also shown that we can calculate least-squares solutions when they are not a simple difference of means, by searching for coefficients which minimize the sum of squared residuals function (using `optim()` remember? and remember how you modified those functions with annoying long lists of coefficients for your own homework?)

Similarly, we've learned that "controlling for" in a simple, additive fashion is also a kind of difference of means, but it is a difference of adjusted means. In my comments on handout4 I illustrated this adjustment process and in your work last time you did it using least squares — where the residuals represented the variables of interest after this kind of linear, additive, adjustment for a third variable. This adjustment stuff took several lines of code, and, if you stopped to think about it, would quickly be quite cumbersome if we had more than one or two variables to adjust and for which to adjust.

[1] We've stayed away from continuous covariates only because we'd have to talk about "smoothed" conditional means or "smoothed" or "weighted" differences of means, not because there is something qualitatively different about what least-squares provides in each case.

And recall when you fed different fitting objective functions to an optimizer? Imagine now writing functions for possibly hundreds of coefficients? That would be a big pain!

Yet, `fit.please()` has only one line of code which is labeled "do least squares". What is happening here?

So, `fit.please()` does least squares using linear algebra. To do smart, creative applied quantitative political science, you don't need to be an expert on linear algebra (aka matrix algebra), but you ought to be familiar with some basic features of it so that you can read journal articles, and, most importantly, compute effectively and efficiently. So, today we'll first play around with the elements on which matrix algebra operates (matrices, vectors, and scalars) and, I hope, you'll gain a somewhat deeper understand of what is happening when we tell R to do least squares with the matrix $\mathbf{X}$ and vector $\mathbf{y}$ via $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ using the command `solve(t(X)%*%X)%*%(t(X)%*%y)`. Where `t(X)`$\equiv \mathbf{X}^T$ (meaning the transpose of $\mathbf{X}$) and `solve(X)`$\equiv \mathbf{X}^{-1}$ (meaning the inverse of $\mathbf{X}$).

0. First, do general setup (load the data, load the functions (namely, `fit.please()` in the file `ps531fns.R`).

```
news.df <- read.csv("http://jakebowers.org/PS531Data/news.df.csv")
# this next loads helpful functions into your R session
source("http://jakebowers.org/PS531Data/ps531fns.R")

print(news.df)  #Just looking at the data

          city state  r z medhhi1000 rpre s
1      Saginaw    MI 16 0       26.5   17 1
2   Sioux City    IA 22 1       37.4   21 1
3 Battle Creek    MI 14 0       35.5   13 2
4      Midland    MI  7 1       48.4   12 2
5       Oxford    OH 23 0       25.2   26 3
6       Lowell    MA 27 1       39.2   25 3
7       Yakima    WA 58 0       29.5   48 4
8     Richland    WA 61 1       53.1   41 4
```

1. First, let's create a $\mathbf{X}$ matrix using the newspapers data to do a regression of the form turnout∼z+baseline.turnout (where baseline.turnout is `rpre` and turnout is `r`). Here is how one might do this in R for both $\mathbf{X}$ and $\mathbf{y}$ (where, **bold** represents matrices or vectors):

```
X <- as.matrix(cbind(1, news.df[, c("z", "rpre")]))
y <- matrix(news.df$r, ncol = 1)  # not strictly necessary, y<-news.df$r would also work

# Look at the dimensions of the objects: number rows by number columns
dim(X)

[1] 8 3

dim(y)

[1] 8 1
```

(1) Explain how we created the $\mathbf{X}$ matrix [*Hint:* The column of 1s has to do with the intercept or constant term.] We just stuck the columns from the dataset onto a column of 1s for the intercept. (2) What do the columns of $\mathbf{X}$ represent? Variables. Attributes of cities. (3) What do the rows represent? Cities. Cases. (4) Look at the code for `fit.please()`. Verify that the three lines before the least squares calculation in `fit.please()` produce your $\mathbf{X}$ matrix and $\mathbf{y}$ vector[*Hint:* If your copy of `fit.please()` as some "..." in it, delete them before you run `model.frame()` etc.. The formula here is r∼z+rpre. And the dataset is news.df.]. Notice that to verify the creation of $\mathbf{X}$ and $\mathbf{y}$ above, you don't need to actually run `fit.please()` but you should just run each of the three lines that create $\mathbf{X}$ and $\mathbf{y}$ in `fit.please()` after suitably editing them to show that your $\mathbf{X}$ created above is the same as the $\mathbf{X}$ created by the `model.matrix` command. That is, you should pull those lines from `fit.please` out of the function and into your class work document for use on their own.

```
print(fit.please)

function (thefmla, thedata, ...)
{
    themf <- model.frame(thefmla, data = thedata, drop.unused.levels = TRUE,
        ...)
    X <- model.matrix(thefmla, data = themf)
    y <- model.response(themf)
    thebs <- solve(t(X) %*% X) %*% (t(X) %*% y)
    return(thebs)
}

thefmla <- r ~ z + rpre  #the formula
themf <- model.frame(thefmla, data = news.df, drop.unused.levels = TRUE)  #make a dataset with only the variables from the formula
newX <- model.matrix(thefmla, data = themf)  #convert the dataset into an appropriate matrix
```

```
newy <- model.response(themf)  #grab the response/outcome vector

# Looks good
newX == X

  (Intercept)    z rpre
1        TRUE TRUE TRUE
2        TRUE TRUE TRUE
3        TRUE TRUE TRUE
4        TRUE TRUE TRUE
5        TRUE TRUE TRUE
6        TRUE TRUE TRUE
7        TRUE TRUE TRUE
8        TRUE TRUE TRUE

newy == y

     [,1]
[1,] TRUE
[2,] TRUE
[3,] TRUE
[4,] TRUE
[5,] TRUE
[6,] TRUE
[7,] TRUE
[8,] TRUE


# But, the two objects have different attributes
all.equal(newX, X)

[1] "Attributes: < Names: 2 string mismatches >"
[2] "Attributes: < Length mismatch: comparison on first 2 components >"
[3] "Attributes: < Component 1: Numeric: lengths (3, 2) differ >"
[4] "Attributes: < Component 2: Modes: numeric, list >"
[5] "Attributes: < Component 2: target is numeric, current is list >"

all.equal(newy, y)

[1] "names for target but not for current"
[2] "Attributes: < names for current but not for target >"
[3] "Attributes: < Length mismatch: comparison on first 0 components >"
[4] "target is numeric, current is matrix"


# Same class
class(newX)

[1] "matrix"

class(X)

[1] "matrix"


# But names and such differ
str(X)

 num [1:8, 1:3] 1 1 1 1 1 1 1 1 0 1 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:3] "1" "z" "rpre"

attributes(X)

$dim
[1] 8 3

$dimnames
$dimnames[[1]]
NULL

$dimnames[[2]]
[1] "1"    "z"    "rpre"

str(newX)
```

```
 num [1:8, 1:3] 1 1 1 1 1 1 1 1 0 1 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:8] "1" "2" "3" "4" ...
  ..$ : chr [1:3] "(Intercept)" "z" "rpre"
 - attr(*, "assign")= int [1:3] 0 1 2

attributes(newX)

$dim
[1] 8 3

$dimnames
$dimnames[[1]]
[1] "1" "2" "3" "4" "5" "6" "7" "8"

$dimnames[[2]]
[1] "(Intercept)" "z"           "rpre"


$assign
[1] 0 1 2


str(y)

 int [1:8, 1] 16 22 14 7 23 27 58 61

attributes(y)

$dim
[1] 8 1

str(newy)

 Named int [1:8] 16 22 14 7 23 27 58 61
 - attr(*, "names")= chr [1:8] "1" "2" "3" "4" ...

attributes(newy)

$names
[1] "1" "2" "3" "4" "5" "6" "7" "8"
```

So, those lines of `fit.please` amount to setting up an $\mathbf{X}$ and a $\mathbf{y}$ matrix.

2. First, addition and subtraction: Try each of the following lines of math in R and explain to yourself (or your colleagues) what happened, and what this means about matrix math. I did the first one as an example.

   Explain what is happening with each of the following

```
X + 2

     1 z rpre
[1,] 3 2   19
[2,] 3 3   23
[3,] 3 2   15
[4,] 3 3   14
[5,] 3 2   28
[6,] 3 3   27
[7,] 3 2   50
[8,] 3 3   43
```

   "When you add a single number (aka a scalar) to a matrix, the scalar is added to each entry in the matrix."

   Notice: If we didn't have matrix math, here is what we'd have to do to add a scalar to a matrix

```
Xplus2 <- matrix(NA, nrow = 8, ncol = 3)  # Initialize an empty matrix for results
# Loop over rows and then over columns
for (row.entry in 1:8) {
    for (col.entry in 1:3) {
        # Add each element to 2 and record in the Xplus2 matrix
        Xplus2[row.entry, col.entry] <- X[row.entry, col.entry] + 2
    }
}

(X + 2) == Xplus2  # Same entries

        1    z rpre
```

```
[1,] TRUE TRUE TRUE
[2,] TRUE TRUE TRUE
[3,] TRUE TRUE TRUE
[4,] TRUE TRUE TRUE
[5,] TRUE TRUE TRUE
[6,] TRUE TRUE TRUE
[7,] TRUE TRUE TRUE
[8,] TRUE TRUE TRUE

# An easier check on whether two objects are the same (except for names and such)
all.equal((X + 2), Xplus2, check.attributes = FALSE)

[1] TRUE


X - 2  # Subtraction and addition of a scalar with a matrix are the same: they operate on each element

      1  z rpre
[1,] -1 -2   15
[2,] -1 -1   19
[3,] -1 -2   11
[4,] -1 -1   10
[5,] -1 -2   24
[6,] -1 -1   23
[7,] -1 -2   46
[8,] -1 -1   39
```

So, adding or subtracting a scalar (i.e. a single number) to a matrix, involves adding that number to each entry of the matrix.

```
twovec<-matrix(c(2,2,2),nrow=1,ncol=3)  # make a vector of 2s
twomat<-matrix(2,nrow=8,ncol=3)  # make a matrix of 2s
```

You'll see some errors appear below. Your job is to explain why R failed or made an error [I had to surround these lines in `try()` to prevent R from stopping at the error]

```
try(X + twovec)
try(X + t(twovec))  # Here, you need to explain what t(twovec) does.
```

`t(twovec)` transposes the vector. Both fail because addition and subtraction are elementwise operations and R is confused about what elements of the 8x3 matrix (X) to add to which elements of the 3x1 (or 1x3) vector/matrix (twovec)

```
opts_current$set(tidy.opts = list(keep.comment = FALSE))
options(keep.comment = FALSE)
X + twomat

     1 z rpre
[1,] 3 2   19
[2,] 3 3   23
[3,] 3 2   15
[4,] 3 3   14
[5,] 3 2   28
[6,] 3 3   27
[7,] 3 2   50
[8,] 3 3   43

# Addition/Subtraction is elementwise so two matrices/vectors of the same dimensions can be added/subtracted easily.
(X + twomat) == (X + 2)

        1    z rpre
[1,] TRUE TRUE TRUE
[2,] TRUE TRUE TRUE
[3,] TRUE TRUE TRUE
[4,] TRUE TRUE TRUE
[5,] TRUE TRUE TRUE
[6,] TRUE TRUE TRUE
[7,] TRUE TRUE TRUE
[8,] TRUE TRUE TRUE

# Adding a matrix full of 2s and adding a scalar 2 is the same thing.
all.equal((X + twomat), (X + 2), check.attributes = FALSE)

[1] TRUE
```

```
all.equal((X + twomat), (twomat + X), check.attributes = FALSE)

[1] TRUE


X + X

     1 z rpre
[1,] 2 0   34
[2,] 2 2   42
[3,] 2 0   26
[4,] 2 2   24
[5,] 2 0   52
[6,] 2 2   50
[7,] 2 0   96
[8,] 2 2   82

# X can be added to itself since it amounts to an operation with two matrices of the same size

colmeansvec <- colMeans(X)  # get the means of the columsn of X

colmeansmat <- matrix(rep(colmeansvec, 8), ncol = 3, byrow = TRUE)
# Fill a matrix with those means: repeat each row 8 times and stack them
```

Why would we want a matrix like the following?

```
X-colmeansmat

     1    z    rpre
[1,] 0 -0.5  -8.375
[2,] 0  0.5  -4.375
[3,] 0 -0.5 -12.375
[4,] 0  0.5 -13.375
[5,] 0 -0.5   0.625
[6,] 0  0.5  -0.375
[7,] 0 -0.5  22.625
[8,] 0  0.5  15.625

# Subtract the column means from each element of X
# That is, we are mean-deviating or centering the columns of X

t(apply(X,1,function(x){x-colmeansvec}))

     1    z    rpre
[1,] 0 -0.5  -8.375
[2,] 0  0.5  -4.375
[3,] 0 -0.5 -12.375
[4,] 0  0.5 -13.375
[5,] 0 -0.5   0.625
[6,] 0  0.5  -0.375
[7,] 0 -0.5  22.625
[8,] 0  0.5  15.625

# This is another way to do the mean-deviating, apply() repeats the
# same vector subtraction on each row of X, and t(apply()) transposes
# the result so that it looks like the other results.

# And here is another way to do it:
sweep(X,2,colmeansvec)  # See the help page for sweep

     1    z    rpre
[1,] 0 -0.5  -8.375
[2,] 0  0.5  -4.375
[3,] 0 -0.5 -12.375
[4,] 0  0.5 -13.375
[5,] 0 -0.5   0.625
[6,] 0  0.5  -0.375
[7,] 0 -0.5  22.625
[8,] 0  0.5  15.625
```

Mean-deviating variables eliminates intercepts (sets intercepts to zero by construction):

```
news.df.md <- scale(news.df[, c("r", "z", "rpre")], center = TRUE, scale = FALSE)
fit.please(r ~ z + rpre - 1, thedata = as.data.frame(news.df.md))

     [,1]
```

```
z    3.39
rpre 1.51
```

```
fit.please(r ~ z + rpre, thedata = as.data.frame(news.df.md))
```

```
            [,1]
(Intercept) 0.00
z           3.39
rpre        1.51
```

Recall that, in a bivariate scatterplot, the globally fit least squares line must go through the point of means (i.e. the point for $\bar{x}$ and $\bar{y}$.) If we center the variables at that point, the new origin of (0,0) is defined to be ($\bar{x}$, $\bar{y}$).

3. Second, multiplication. Notice that the symbols for scalar multiplication and matrix multiplication are not the same. Try each of the following lines of math in R and explain to yourself (or your colleagues) what happened, and what this means about matrix math.

```
options(keep.comment = FALSE)
X * 2
```

```
     1 z rpre
[1,] 2 0   34
[2,] 2 2   42
[3,] 2 0   26
[4,] 2 2   24
[5,] 2 0   52
[6,] 2 2   50
[7,] 2 0   96
[8,] 2 2   82
```

```
all.equal((X * 2), (2 * X))
```

```
[1] TRUE
```

```
X^2
```

```
     1 z rpre
[1,] 1 0  289
[2,] 1 1  441
[3,] 1 0  169
[4,] 1 1  144
[5,] 1 0  676
[6,] 1 1  625
[7,] 1 0 2304
[8,] 1 1 1681
```

```
X^0.5
```

```
     1 z rpre
[1,] 1 0 4.12
[2,] 1 1 4.58
[3,] 1 0 3.61
[4,] 1 1 3.46
[5,] 1 0 5.10
[6,] 1 1 5.00
[7,] 1 0 6.93
[8,] 1 1 6.40
```

```
sqrt(X)
```

```
     1 z rpre
[1,] 1 0 4.12
[2,] 1 1 4.58
[3,] 1 0 3.61
[4,] 1 1 3.46
[5,] 1 0 5.10
[6,] 1 1 5.00
[7,] 1 0 6.93
[8,] 1 1 6.40
```

Now, let's get a vector of coefficients to make this matrix math link even more tightly with fitting models to data:

```
b <- fit.please(r ~ z + rpre, thedata = news.df)
dim(b)
```

```
[1] 3 1
```

Now, let's do matrix multiplication the tedious way. What does this function tell us about the rule for doing matrix multiplication?

```
X.times.b <- matrix(NA, nrow = 8, ncol = 1)
for (row.entry in 1:8) {
    temp <- vector(length = 3)
    for (col.entry in 1:3) {
        temp[col.entry] <- X[row.entry, col.entry] * b[col.entry, ]
    }
    X.times.b[row.entry, ] <- sum(temp)
}
X.times.b

      [,1]
[1,] 14.14
[2,] 23.58
[3,]  8.09
[4,]  9.97
[5,] 27.75
[6,] 29.63
[7,] 61.01
[8,] 53.82
```

It shows us that matrix multiplication occurs row-by-column: first row multiplying first column, etc.. [And then matrix multiplication involves adding up the sums produced in this row-by-column multiplying.]

Now, doing part of it by hand:

```
(X[1,1] * b[1])+(X[1,2] * b[2])+(X[1,3] * b[3])  # Matrix multiplication is sum of row-times-column multiplication.

   1
14.1


(X[2,1] * b[1])+(X[2,2] * b[2])+(X[2,3] * b[3])

   1
23.6
```

And now a little faster (multiplying vectors rather than scalars and summing): You can break matrix multiplication into separate vector multiplication tasks since vector multiplication also goes sum-of-row-times-column.

```
X[1,] %*% b # First row of X by b

     [,1]
[1,] 14.1

X[2,] %*% b # Second row of X by b [b is a single column]

     [,1]
[1,] 23.6
```

And doing it very fast: This is direct matrix multiplication. So nice and clean compared to the previous! Don't we love matrix multiplication?

```
X %*% b

      [,1]
[1,] 14.14
[2,] 23.58
[3,]  8.09
[4,]  9.97
[5,] 27.75
[6,] 29.63
[7,] 61.01
[8,] 53.82
```

How does `fitted(thelm)` relate to `X %*% b`? What is

```
thelm<-lm(r~z+rpre,data=news.df)
fitted(thelm)

    1     2     3     4     5     6     7     8
14.14 23.58  8.09  9.97 27.75 29.63 61.01 53.82
```

This is the same as `X %*% b` where `b<-coef(thelm)` or `b<-fit.please(r~z+rpre,data=news.df)`.

So, when you see people write things like $\mathbf{y} = \mathbf{X}\mathbf{b}$ or $E(\mathbf{y}) = \mathbf{X}\hat{\boldsymbol{\beta}}$ you know that they are talking about the fitted values (or perhaps predicted values) from a linear model [linear because $\mathbf{Xb}$ is something like $b_0 + b_1x_1 + b_2x_2 + \ldots$ —- a linear combination of the x's weighted by or the b's.]

This will also come in handy when you are using a model fitting technique that does not have a convenient `fitted()` or `predict()` method.

Recall that `predict()` is a kind of generalized way to get fitted *or* predicted values:

```
all.equal(fitted(thelm), predict(thelm))

[1] TRUE

predict(thelm, newdata = data.frame(z = 0, rpre = c(5, 90)))

  1   2
 -4 125

newX <- data.frame(intercept = 1, z = 0, rpre = c(5, 90))
as.matrix(newX) %*% b

     [,1]
[1,]   -4
[2,]  125
```

4. How would you use matrix addition/subtraction to get the residuals once you had $\mathbf{Xb}$?

   Recall that the residuals are observed values minus fitted values (i.e. model values or predicted values at the observed values of the explanatory variables)

```
yhat <- X %*% b
cbind(yhat, fitted(thelm))

   [,1]  [,2]
1 14.14 14.14
2 23.58 23.58
3  8.09  8.09
4  9.97  9.97
5 27.75 27.75
6 29.63 29.63
7 61.01 61.01
8 53.82 53.82

ehat <- y - yhat
cbind(ehat, resid(thelm))

   [,1]  [,2]
1  1.86  1.86
2 -1.58 -1.58
3  5.91  5.91
4 -2.97 -2.97
5 -4.75 -4.75
6 -2.63 -2.63
7 -3.01 -3.01
8  7.18  7.18
```

   So, now you know not only how to get the vector of least squares coefficients, but also how to produce fitted values and residuals.

   The act of producing fitted values, by the way, means that you can play "what-if" games with your fitted model, asking, "What would happen if, say, I had a city with a baseline turnout of BLAH in the control group, etc..." You can do this by making $\mathbf{X}$ not contain the observed values of the treatment and baseline outcome/covariate, but by making a new $\mathbf{X}$ matrix reflecting values of $\mathbf{X}$ that you'd like to assess using your fitted model.

5. Say, we fit a different model, one in which the treatment is allowed to interact with baseline turnout such that the model formula would be r∼z*rpre.

   Fit the model using either `fit.please()` or `lm()` and interpret the results.

   Now, let's try to illustrate on a plot how the relationship between turnout and baseline turnout may depend on advertisements using our linear fit: (1) Use matrix multiplication to calculate the predicted values of turnout. (2) Plot the lines on a scatterplot of turnout against baseline turnout.

   Hint: So, you have the new coefficient vector (let's call it `b2`) and you've seen how to produce fitted values using matrix multiplication with the original $\mathbf{X}$ matrix. Now, however, we cannot merely matrix multiply $\mathbf{X}$ and —b2—, but we need a column in $\mathbf{X}$ representing the interaction of rpre and z (i.e. the column z multiplied by the column rpre). So, first you need to create a new $\mathbf{X}$ matrix with a column representing the interaction term. Next, you need to produce a vector of $\hat{y}_i$ or $\hat{\mathbf{y}}$ (i.e. fitted or predicted values). Third, you need to plot —r— (on y-axis) against —rpre— as a simple scatterplot. And fourth, you need to add lines to the plot to represent how turnout depends on both treatment and baseline turnout.

Hint 2: Say you use `newfit<-newX %*% newb`. Is the first entry in —newfit— for a control or a treated observation? When you plot, you'll want to make one line for the control observations and one line for the treated observations. To pull out the values of newfit for control observations, I might do something like `newfit[newX[,"z"]==0]`.

Bonus: Notice that you didn't really need to produce a predicted value for each and every observation in the dataset — these are straight lines and thus only need two points to connect. How might you reduce the number of points plotted (reduce the computation required above)? [try `expand.grid()`].[2]

```
fit2 <- lm(r ~ z * rpre, data = news.df)
(b2 <- coef(fit2))

(Intercept)           z        rpre      z:rpre
      -5.90      -11.25        1.29        0.58

(newX <- cbind(X, X[, 2] * X[, 3]))

     1 z rpre
[1,] 1 0   17  0
[2,] 1 1   21 21
[3,] 1 0   13  0
[4,] 1 1   12 12
[5,] 1 0   26  0
[6,] 1 1   25 25
[7,] 1 0   48  0
[8,] 1 1   41 41

(yhat <- newX %*% b2)

      [,1]
[1,] 16.10
[2,] 22.22
[3,] 10.92
[4,]  5.35
[5,] 27.75
[6,] 29.72
[7,] 56.22
[8,] 59.71

cbind(newX, yhat)

     1 z rpre
[1,] 1 0   17  0 16.10
[2,] 1 1   21 21 22.22
[3,] 1 0   13  0 10.92
[4,] 1 1   12 12  5.35
[5,] 1 0   26  0 27.75
[6,] 1 1   25 25 29.72
[7,] 1 0   48  0 56.22
[8,] 1 1   41 41 59.71
```

```
plot(r ~ rpre, data = news.df, xlab = "Baseline Outcome", ylab = "Outcome", pch = 19)
lines(newX[newX[, "z"] == 0, "rpre"], yhat[newX[, "z"] == 0], col = "blue", type = "b")
lines(newX[newX[, "z"] == 1, "rpre"], yhat[newX[, "z"] == 1], col = "orange", type = "b")
```

twidthtwidth

For the bonus and to make life easier when you are working with thousands of observations and not just 8 of them:

```
(preddat <- expand.grid(z = c(0, 1), rpre = range(news.df$rpre)))

  z rpre
1 0   12
2 1   12
3 0   48
4 1   48
```

[2]You can see some discussion of making such plots here: http://stackoverflow.com/questions/1395147/best-way-to-plot-interaction-effects-from-a-linear-model.

```
(predmat <- model.matrix(~z * rpre, data = preddat))

  (Intercept) z rpre z:rpre
1           1 0   12      0
2           1 1   12     12
3           1 0   48      0
4           1 1   48     48
attr(,"assign")
[1] 0 1 2 3

(preddat$fits <- predmat %*% b2)

   [,1]
1  9.63
2  5.35
3 56.22
4 72.84

preddat

  z rpre  fits
1 0   12  9.63
2 1   12  5.35
3 0   48 56.22
4 1   48 72.84
```

This plot has the same points as the previous one but the lines (which are also the same) are generated using only two points each and thus thus is easier to print, plot, display, etc.. [this mainly comes to matter when you have thousands of cases]

```
plot(r ~ rpre, data = news.df, xlab = "Baseline Outcome", ylab = "Outcome", pch = 19)
lines(fits ~ rpre, data = preddat, subset = z == 0, col = "blue", type = "b")
lines(fits ~ rpre, data = preddat, subset = z == 1, col = "orange", type = "b")
```

twidthtwidth

So, it appears the the treatment effect is different at different values of the baseline [but, recall this is assuming a globally smooth and linear relationship].

6. Now, let's meet another important matrix:

```
try(X %*% X)
```

Recall that matrix multiplication is row-by-column. In this case the first row of $\mathbf{X}$ has 3 elements but the first column of $\mathbf{X}$ has 8 elements: you can't do matrix multiplication unless you have the same number of elements to multiply.

```
t(X)  # Transpose of X

     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
1       1    1    1    1    1    1    1    1
z       0    1    0    1    0    1    0    1
rpre   17   21   13   12   26   25   48   41


XtX<-t(X) %*% X  # Aha! t(x) is 3x8 and X is 8x3, so XtX is 3x3

XtX

       1  z rpre
1      8  4  203
z      4  4   99
rpre 203 99 6329
```

To make our lives easier, let's mean deviate all of the variables (i.e. set it up so that all of them have mean=0). Now the `XtX` matrix will be easier to understand:

```
X.md<-X-colmeansmat
y.md<-y-mean(y)
XtX.md<-t(X.md) %*% X.md
XtX.md

     1    z   rpre
1    0  0.0    0.0
z    0  2.0   -2.5
rpre 0 -2.5 1177.9
```

Explain what each entry in `XtX.md` is: if you can, relate those numbers to quantities like variances, covariances, sums (of squares or not), sample sizes, do so.

Here another representation of `XtX` that might help you explain and reproduce the entries above.

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} n & \sum_{i=1}^n z_i & \sum_{i=1}^n \text{rpre}_i \\ \sum_{i=1}^n z_i & \sum_{i=1}^n {z_i}^2 & \sum_{i=1}^n z_i\text{rpre}_i \\ \sum_{i=1}^n \text{rpre}_i & \sum_{i=1}^n z_i\text{rpre}_i & \sum_{i=1}^n {\text{rpre}_i}^2 \end{bmatrix}$$

Try some of the following commands to get some other help in understanding what XtX is:

```
options(keep.comment = FALSE)
sum(X.md[, 1])

[1] 0

sum(X.md[, 2]^2)

[1] 2

sum((X.md[, 2] - mean(X.md[, 2]))^2)

[1] 2

sum((X.md[, 2] - mean(X.md[, 2]))^2)/(8 - 1)

[1] 0.286

var(X.md[, 2])

[1] 0.286

sum(X.md[, 2] * X.md[, 3])

[1] -2.5

sum(X.md[, 2] * X.md[, 3])/(8 - 1)

[1] -0.357

cov(X.md)

     1      z    rpre
1    0  0.000   0.000
z    0  0.286  -0.357
rpre 0 -0.357 168.268

XtX.md/(8 - 1)

     1      z    rpre
1    0  0.000   0.000
z    0  0.286  -0.357
rpre 0 -0.357 168.268
```

So, XtX is basically the variance-covariance matrix of the variables on the right-hand side of the linear model. The diagonal entries are the variances of each element (of z and rpre respectively). The off-diagonal entries are the covariances between z and rpre. When the equation has a constant, then some of the entries are simple sums of the corresponding variables.

What about $\mathbf{X}^T\mathbf{y}$? Explain the entries in `Xty.md`

```
options(keep.comment = FALSE)
t(X.md)

      [,1]  [,2]  [,3]  [,4]   [,5]   [,6] [,7] [,8]
1     0.00  0.00   0.0   0.0  0.000  0.000  0.0  0.0
z    -0.50  0.50  -0.5   0.5 -0.500  0.500 -0.5  0.5
rpre -8.38 -4.38 -12.4 -13.4  0.625 -0.375 22.6 15.6

y.md

      [,1]
[1,] -12.5
[2,]  -6.5
[3,] -14.5
[4,] -21.5
[5,]  -5.5
[6,]  -1.5
```

```
[7,]  29.5
[8,]  32.5

Xty.md <- t(X.md) %*% y.md
Xty.md

     [,1]
1       0
z       3
rpre 1772

cov(cbind(X.md, y.md))

     1      z    rpre
1    0  0.000   0.000   0.000
z    0  0.286  -0.357   0.429
rpre 0 -0.357 168.268 253.214
     0  0.429 253.214 404.286

Xty.md/7

        [,1]
1      0.000
z      0.429
rpre 253.214
```

The following is a verbal formula for a covariance: deviations of x from its mean times deviations of y from its mean divided by n-1 (i.e. roughly the average of the product of the deviations)

```
sum((X[,2]-mean(X[,2]))*(y-mean(y)))

[1] 3

sum((X[,2]-mean(X[,2]))*(y-mean(y)))/(8-1)

[1] 0.429


# Same as above because we've removed the means from X.md and y.md
sum((X.md[,2])*(y.md))/(8-1)

[1] 0.429


Xty<-t(X) %*% y
Xty

     [,1]
1     228
z     117
rpre 7558

Xty/(8-1)

       [,1]
1      32.6
z      16.7
rpre 1079.7
```

The entries in $(\mathbf{X}^T\mathbf{y})$ are the covariances between the columns of $\mathbf{X}$ and the vector $\mathbf{y}$. Notice, that, for the intercept, the covariance is 0 (and in XtX.md, if we included an intercept, we'd have a 0 entry for the variance in the intercept and 0s for the covariances between the intercept and the other variables). Later on we'll see that the least squares coefficients can emerge from dividing the $(\mathbf{X}^T\mathbf{y})$ matrix by the $(\mathbf{X}^T\mathbf{X})$ matrix — and thus, with mean-deviated variables, we'd have a division by 0 problem if we forced an intercept into the equation.

7. And finally division: Try each of the following lines of math in R and explain to yourself (or your colleagues) what happened (ideally relate what happened to ideas about variances, covariances, sums, etc..)

```
options(keep.comment = FALSE)
X/2

       1   z rpre
[1,] 0.5 0.0  8.5
[2,] 0.5 0.5 10.5
[3,] 0.5 0.0  6.5
[4,] 0.5 0.5  6.0
[5,] 0.5 0.0 13.0
```

```
[6,] 0.5 0.5 12.5
[7,] 0.5 0.0 24.0
[8,] 0.5 0.5 20.5
```

```
1/X
```

```
     1   z   rpre
[1,] 1 Inf 0.0588
[2,] 1   1 0.0476
[3,] 1 Inf 0.0769
[4,] 1   1 0.0833
[5,] 1 Inf 0.0385
[6,] 1   1 0.0400
[7,] 1 Inf 0.0208
[8,] 1   1 0.0244
```

```
X^(-1)
```

```
     1   z   rpre
[1,] 1 Inf 0.0588
[2,] 1   1 0.0476
[3,] 1 Inf 0.0769
[4,] 1   1 0.0833
[5,] 1 Inf 0.0385
[6,] 1   1 0.0400
[7,] 1 Inf 0.0208
[8,] 1   1 0.0244
```

```
1/X == (X^(-1))
```

```
        1    z rpre
[1,] TRUE TRUE TRUE
[2,] TRUE TRUE TRUE
[3,] TRUE TRUE TRUE
[4,] TRUE TRUE TRUE
[5,] TRUE TRUE TRUE
[6,] TRUE TRUE TRUE
[7,] TRUE TRUE TRUE
[8,] TRUE TRUE TRUE
```

```
try(solve(X))
solve(XtX)
```

```
           1        z      rpre
1     0.8254 -0.27767 -0.022132
z    -0.2777  0.50133  0.001064
rpre -0.0221  0.00106  0.000851
```

```
dim(XtX)
```

```
[1] 3 3
```

```
dim(Xty)
```

```
[1] 3 1
```

```
solve(XtX) %*% Xty
```

```
       [,1]
1    -11.56
z      3.39
rpre   1.51
```

```
try(solve(XtX.md))
XtX.md <- t(X.md[, 2:3]) %*% X.md[, 2:3]
try(solve(XtX.md))
```

```
           z     rpre
z    0.50133 0.001064
rpre 0.00106 0.000851
```

```
Xty.md <- t(X.md[, 2:3]) %*% y.md
solve(XtX.md) %*% Xty.md
```

```
     [,1]
z    3.39
rpre 1.51
```

```
(1/XtX) %*% Xty

      [,1]
1     95.0
z    162.6
rpre   3.5

fit.please(I(r - mean(r)) ~ I(z - mean(z)) + I(rpre - mean(rpre)) - 1, thedata = news.df)

                       [,1]
I(z - mean(z))        3.39
I(rpre - mean(rpre)) 1.51

fit.please(scale(r, scale = FALSE) ~ scale(z, scale = FALSE) + scale(rpre, scale = FALSE) - 1, thedata = news.df)

                             [,1]
scale(z, scale = FALSE)      3.39
scale(rpre, scale = FALSE) 1.51

lm(y.md ~ X.md[, "z"] + X.md[, "rpre"] - 1)


Call:
lm(formula = y.md ~ X.md[, "z"] + X.md[, "rpre"] - 1)

Coefficients:
   X.md[, "z"]  X.md[, "rpre"]
          3.39            1.51

coef(lm(r ~ z + rpre, data = news.df))

(Intercept)            z        rpre
     -11.56        3.39        1.51

fit.please(r ~ z + rpre, thedata = news.df)

              [,1]
(Intercept) -11.56
z             3.39
rpre          1.51
```

8. So, the vector of least squares coefficients is the result of dividing what kind of matrix by what kind of matrix? (what kind of information by what kind of information)? *Hint:* This perspective of "accounting for covariation" is another valid way to think about what least squares is doing [in addition to smoothing conditional means]. They are mathematically equivalent.

The least squares coefficients, $\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ or $\hat{\boldsymbol{\beta}}$ represent the covariances between $\mathbf{y}$ and the columns of $\mathbf{X}$ divided by the variances and covariances of $\mathbf{X}$.

So, we have, in some sense, information about what proportion of some covariance (which is like a correlation) is accounted for by some variance.

Why should covariances divided by variances amount to differences of means, let alone adjusted differences of means?

Here are some definitions of covariance and variance:

$$\text{Cov}(X,Y) = \frac{\sum_i^n (X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$
$$\text{Var}(X) = \text{Cov}(X,X) = \frac{\sum_i^n (X_i - \bar{X})(X_i - \bar{X})}{n-1} = \frac{\sum_i^n (X_i - \bar{X})^2}{n-1}$$

So, first,

$$\frac{\text{Cov}(X,Y)}{\text{Var}(X)} = \frac{\sum_i^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_i^n (X_i - \bar{X})^2}$$

because the (n-1) cancels out. (Thus, we had to divide X^TX and X^Ty by n-1 in the sections above to get the analogous covariances/variances). So, this is the bivariate case with $y = \beta_0 + \beta_1 x_1$. What about $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$?

This becomes notationally messy fast. I'll try to post a scan from Hanushek and Jackon or some other place where they present a nice scalar-form version of the three variable regression. Already, however, you can get a sense for the idea of deviations from the mean being a key ingredient in these calculations.

Size versus Level of a Test

The simulations that we did last time ask, in essence, whether the promise of a Type I error rate of a test (the level of the test) is actually fulfilled in practice: whether the level is the same as the size. Here are some quotes from the "Glossary" to Rosenbaum's 2010 book to help us with language to talk about how well our procedures operate:

consistent estimate A consistent estimate is one that gets it right if the sample size is large enough. An estimator $\hat{\theta}$ is a consistent estimate of a parameter θ if $\hat{\theta}$ converges in probability to θ as the sample size increases. See 'convergence in probability' in this glossary.

unbiased estimate An unbiased estimate is one that is right on average. An estimator $\hat{\theta}$ is an unbiased estimate of a parameter θ if the expectation of $\hat{\theta}$ equals θ .

consistent test A consistent test is one that gets it right if the sample size is large enough. A test of a null hypothesis H_0 is consistent against an alternative hypothesis H_A if the power of the test tends to one as the sample size increases when H_A is true.

level α test A statistical test has level α if, in the worst case, it rejects a true hypothesis with probability at most α, that is, if the supremum over true hypotheses of the probability of rejection is less than or equal to α. Compare the level of a test with the size α of a test as defined later in the glossary. In typical use, the level of the test is a promise about the tests performance and the size is a fact about its performance, where the achieved fact may be better than the promised performance. Suppose that a randomization test, such as Wilcoxons signed rank test, is applied in a paired randomized experiment, as in Chapter 2, and the null hypothesis is rejected if the P-value is less than or equal to 0.05. Then the test will have level $\alpha = 0.05$. However, if $|\zeta|$ is not divisible by 20, then the size of the test will be a little smaller than 0.05. In this case, one is advertising false rejections of true hypotheses at a 5% rate, but the rate is a tad better, a bit lower, than 5%. See E.L. Lehmanns book Testing Statistical Hypotheses, New York: John Wiley (1959).

power (of a test) The power of a test is the probability that the test will reject the null hypothesis when the null hypothesis is false. You would like the power to be high. If the test rejects when the P-value is less than 0.05, then the power of the test is the chance that the P-value is less than 0.05 when the null hypothesis is false. The power depends upon many things. In particular, the power depends upon what is true. If the null hypothesis is false, then something else is true. For instance, consider the power of a test of the null hypothesis that the treatment has no effect, where the test rejects when the P-value is less than 0.05. If the actual treatment effect is very small, the power may be only 0.06, but if the actual treatment effect is very large, the power may be 0.99. In other words, there may be only a 6% chance of rejecting the null hypothesis if the treatment effect is very small, but a 99% chance of rejection if the effect is very large. (Remember, if you test a true null hypothesis and reject when the P-value is less than 0.05, then you typically run a 5% chance of rejecting the true null hypothesis; so, 6% power is no reason for pride.) The power also depends upon the sample size. Typically, the power is larger when the sample size is larger.

size α test A statistical test has size α if, in the worst case, it rejects a true hypothesis with probability α, that is, if the supremum over true hypotheses of the probability of rejection is equal to α. Compare this with a level α test. See E.L. Lehmanns book Testing Statistical Hypotheses, New York: John Wiley (1959).

unbiased test Speaking informally, a test is unbiased if it is more likely to reject false hypotheses than to reject true hypotheses. An α level test is unbiased against a set of alternative hypotheses if the power of the test against these alternatives is at least α. See level α test in this glossary.

(pages 363–368, Rosenbaum 2009)

So, let's look again at tests, but this time as applied to the small field experiment data:

```r
perm.dist.fn <- function(y, z, strata) {
    newz <- shuffle(z, within = strata)
    coef(lm(y ~ newz + factor(strata)))
}
perm.sim.p.fn <- function(n.shuf, y, z, strata = rep(1, length(z))) {
    newz <- shuffle(z, within = strata)
    obs.coef <- coef(lm(y ~ newz + factor(strata)))
    perm.dist <- do(n.shuf) * perm.dist.fn(y = y, z = z, strata = strata)
    theps <- sapply(1:length(obs.coef), function(which.coef) {
        upper.p <- mean(perm.dist[, which.coef] >= obs.coef[which.coef])
        lower.p <- mean(perm.dist[, which.coef] <= obs.coef[which.coef])
        return(2 * min(upper.p, lower.p))
    })
    names(theps) <- names(obs.coef)
    return(theps)
}
set.seed(20110220)
library(parallel)
numcores <- detectCores()
sim.perm.p <- simplify2array(mclapply(1:100, function(i) {
    perm.sim.p.fn(500, y = news.df$r, z = news.df$z, strata = news.df$s)
}, mc.cores = numcores))
```

```
str(sim.perm.p)

 num [1:5, 1:100] 0.968 1.036 0.204 0.34 0.676 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:5] "(Intercept)" "newz" "factor(strata)2" "factor(strata)3" ...
  ..$ : NULL

apply(sim.perm.p, 1, summary)

        (Intercept)  newz factor(strata)2 factor(strata)3 factor(strata)4
Min.          0.080 0.080           0.096           0.096           0.088
1st Qu.       0.307 0.307           0.286           0.374           0.442
Median        0.580 0.590           0.610           0.494           0.730
Mean          0.550 0.553           0.570           0.577           0.647
3rd Qu.       0.805 0.798           0.799           0.795           0.941
Max.          1.050 1.060           1.060           1.110           1.100

apply(sim.perm.p, 1, function(x) {
    mean(x < 0.12)
})

    (Intercept)            newz factor(strata)2 factor(strata)3 factor(strata)4
           0.08            0.08            0.01            0.01            0.05

apply(sim.perm.p, 1, function(x) {
    quantile(x, c(0.025, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.975))
})

      (Intercept)   newz factor(strata)2 factor(strata)3 factor(strata)4
2.5%       0.0954 0.0954           0.152           0.152           0.104
5%         0.1120 0.1120           0.175           0.175           0.120
10%        0.1276 0.1276           0.227           0.272           0.151
25%        0.3070 0.3070           0.286           0.374           0.442
50%        0.5800 0.5900           0.610           0.494           0.730
75%        0.8050 0.7980           0.799           0.795           0.941
90%        0.9608 0.9612           0.990           1.012           1.012
95%        0.9960 1.0246           1.020           1.044           1.052
97.5%      1.0200 1.0463           1.036           1.060           1.060
```

```
lm.sim.p.fn <- function(y, z, strata) {
    newz <- shuffle(z, within = strata)
    summary(lm(y ~ newz + factor(strata)))$coef[, "Pr(>|t|)"]
}
set.seed(20110220)
sim.lm.p <- replicate(10000, lm.sim.p.fn(y = news.df$r, z = news.df$z, s = news.df$s))
```

```
str(sim.lm.p)

 num [1:5, 1:10000] 0.00375 0.21838 0.07596 0.15642 0.00105 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:5] "(Intercept)" "newz" "factor(strata)2" "factor(strata)3" ...
  ..$ : NULL

apply(sim.lm.p, 1, summary)

        (Intercept)  newz factor(strata)2 factor(strata)3 factor(strata)4
Min.       0.000234 0.012         0.00713          0.0188       0.0000712
1st Qu.    0.005140 0.314         0.09410          0.1860       0.0014000
Median     0.008900 0.526         0.12100          0.2270       0.0019900
Mean       0.007930 0.529         0.10500          0.1970       0.0017400
```

```
3rd Qu.    0.011100 0.879         0.14000          0.2540        0.0024700
Max.       0.011500 1.000         0.14100          0.2560        0.0025000
```

```
apply(sim.lm.p, 1, function(x) {
    mean(x < 0.12)
})
```

```
    (Intercept)             newz factor(strata)2 factor(strata)3 factor(strata)4
          1.000            0.122           0.378           0.122           1.000
```

```
apply(sim.lm.p, 1, function(x) {
    quantile(x, c(0.025, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.975))
})
```

```
      (Intercept)  newz factor(strata)2 factor(strata)3 factor(strata)4
2.5%     0.000234 0.012         0.00713          0.0188       0.0000712
5%       0.000234 0.012         0.00713          0.0188       0.0000712
10%      0.000515 0.012         0.00713          0.0188       0.0000712
25%      0.005137 0.314         0.09414          0.1861       0.0013999
50%      0.008896 0.526         0.12099          0.2268       0.0019899
75%      0.011103 0.879         0.14010          0.2539       0.0024715
90%      0.011178 1.000         0.14135          0.2557       0.0025047
95%      0.011452 1.000         0.14135          0.2557       0.0025047
97.5%    0.011452 1.000         0.14135          0.2557       0.0025047
```

Recall that if the level and size of the tests are the same, then we would expect the points in the following plot to following the diagonal line closely. We can still talk about a level .05 test if the points are not on the diagonal but are underneath it — at $p = .1$ we have fewer than 10% of the values (recall Rosenbaum's definition of the level of a test). That is, if fewer than 10% of the p-values are less than .1 we can still say that our promise of rejecting the true null no more than 10% of the time is met.

```
par(mfrow = c(1, 2), pty = "s", oma = rep(0, 4))
plot(ecdf(sim.lm.p["newz", ]), xlim = c(0, 1))
abline(0, 1)
abline(h = 0.12, v = 0.12)
plot(ecdf(sim.perm.p["newz", ]))
abline(0, 1)
abline(h = 0.12, v = 0.12)
```

twidthtwidth

So, what we see here is that permutation based test fulfills the promise of level — the size of the test is never greater than the level of the test. This means that the confidence intervals might be conservative (i.e. too wide), although we can also see that it is not that conservative.

The large-sample iid theory p-values (canned in `lm`) do not consistently have the size equal the level. [Notice here if you are playing around with this that it makes a big difference to have the strata dummies enter into the model.

We can see this here with the large-sample, iid nominally 88% confidence intervals (which behave correctly regarding the inclusion of important new information into the model even if they might not really be 88% intervals: meaning that we claim that the level of the tests used to create these confidence intervals is $\alpha = .12$ but, in fact, the size of those tests might be other than .12.

```
confint(lm(r ~ z, data = news.df), parm = "z", level = 0.88)
```

```
    6 % 94 %
z -26.3 29.3
```

```
confint(lm(r ~ z + factor(s), data = news.df), parm = "z", level = 0.88)
```

```
    6 % 94 %
z -4.76 7.76
```