# Interaction-Categorical Predictors

Paul E. Johnson[1] [2]

[1]Department of Political Science

[2]Center for Research Methods and Data Analysis, University of Kansas

2014

# Outline

# Outline

# Categorical Predictors

- Dichotomous: M or F
- Polychotomy: R, D, C, S, I, .... (not ordered)
- Ordinal Variable: Lo, Med, Hi

# Creating Contrasts

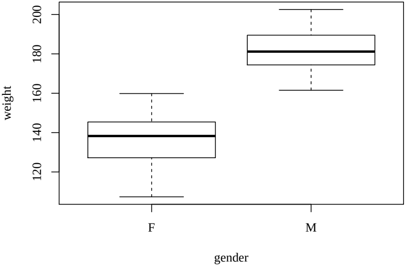
- A design emphasis in R is that users should not “create dummy variables” manually.
- Estimation routines should recognize ’factor’ variables and create suitable contrasts automatically
- A menu of available contrast schemes is available (and specified to environment as options)

# Outline

# Plot a fitted model with Gender in {M,F}

| | M1 | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | 136.452*** | (1.099) |
| genderM | 45.443*** | (1.555) |
| N | 200 | |
| RMSE | 10.993 | |
| $R^2$ | 0.812 | |

$*p \leq 0.05 ** \ p \leq 0.01 *** p \leq 0.001$

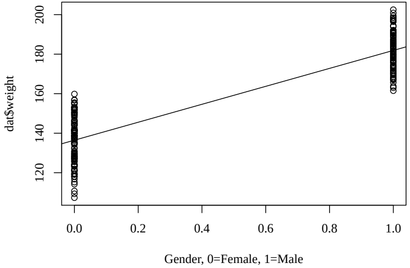



Intercept is mean weight for Female
"genderM" estimate is difference between Mean of Female and Male

# Maybe You are Not a Fan of the Box and Whisker Plot?

The "Contrasts" created for the Gender variable are

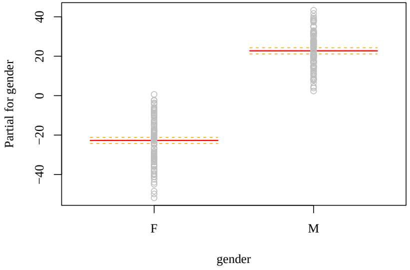
```
  M
F 0
M 1
```




Seems dishonest to allow x axis to take on a continuum of values.

# But the ”abline” is deceptive. Right?

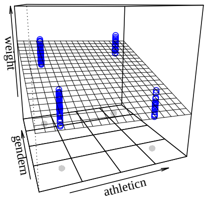
The termplot function tries to show only the meaningful information about the predictor



# Another Dichotomy: Are You Athletic?

| | M1 | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | 137.886*** | (1.165) |
| genderM | 45.443*** | (1.520) |
| athleticNo | -5.518** | (1.733) |
| N | 200 | |
| RMSE | 10.748 | |
| $R^2$ | 0.821 | |
| adj $R^2$ | 0.819 | |

$*p \leq 0.05 ** \; p \leq 0.01 *** p \leq 0.001$



# What if we include an ”interaction”

- Previous model assumed

$$weight_i = \beta_0 + \beta_1 genderM_i + \beta_2 athleticNo_i + e_i$$

The effect of genderM is the same for athletic people
The effect of athleticNo is the same for male and female

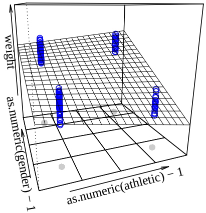
- Suppose instead that the effect of being athletic is different for M and F

$$\begin{aligned} weight_i &= \beta_0 + \beta_1 genderM_i + \beta_2 athleticNo_i + \beta_3 genderM_i \cdot athleticNo_i + e_i \\ &= \beta_0 + \beta_1 genderM_i + (\beta_2 + \beta_3 genderM_i) \cdot athleticNo_i + e_i \end{aligned}$$

# Estimate a Model that Allows Interaction.

| | M1 | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | 140.791*** | (1.113) |
| genderM | 39.633*** | (1.573) |
| athleticNo | -16.690*** | (2.182) |
| genderM:athleticNo | 22.345*** | (3.086) |
| N | 200 | |
| RMSE | 9.571 | |
| $R^2$ | 0.859 | |
| adj $R^2$ | 0.857 | |

$*p \leq 0.05 ** \ p \leq 0.01 *** p \leq 0.001$



# Predicted Values are Same as Means of Subgroups

```
newx <- expand.grid(gender = levels(dat$gender), athletic = levels(
     dat$athletic))
(newx$pred <- predict(m3, newdata = newx))
```

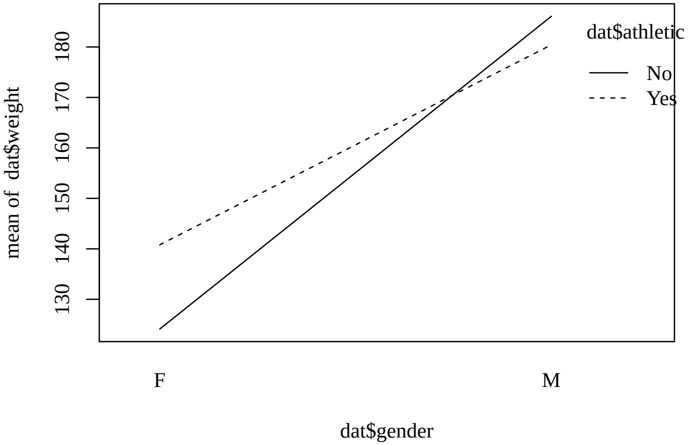
```
       1        2        3        4
140.7913 180.4243 124.1009 186.0789
```

```
grpmeans <- aggregate(dat$weight, by = list("gender" = dat$gender,
      "athletic" = dat$athletic), FUN = mean)
grpmeans
```

```
  gender athletic        x
1      F      Yes 140.7913
2      M      Yes 180.4243
3      F       No 124.1009
4      M       No 186.0789
```

# Try Out interaction.plot to Display

interaction.plot does not calculate regressions. It simply "connects the dots" of observed means.



# I Admit I'm a Neanderthal

- You should take a class on analysis of variance, where they will explain why I'm wrong, but
- All coding schemes that lead to the same predicted values for the subgroups are equivalent. (Same $R^2$, etc)
- "treatment contrasts" or "effects contrasts" or whatever change the "free hypothesis tests" that are provided with the printout
- Follow up hypothesis tests can be used to compare parameters when needed

# Outline

# I Like This

- Suppose data collected in M categories
- Can fit M separate regression models
- Can stack data from M sets into one, estimate with categorical interaction

# Fit 3 Regressions, one for each "type"

Use car package's Prestige dataset

```
library(car)
m1by <- by(Prestige, Prestige$type, function(x) {lm(prestige ~
     education, data=x)})
(lapply(m1by, summary))
```

```
$bc

Call:
lm(formula = prestige ~ education, data = x)

Residuals:
     Min       1Q   Median       3Q      Max
-19.7095  -6.0923   0.5828   6.4920  16.1411

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -4.294      9.331  -0.460    0.648
education      4.764      1.106   4.308 9.71e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.447 on 42 degrees of freedom
```

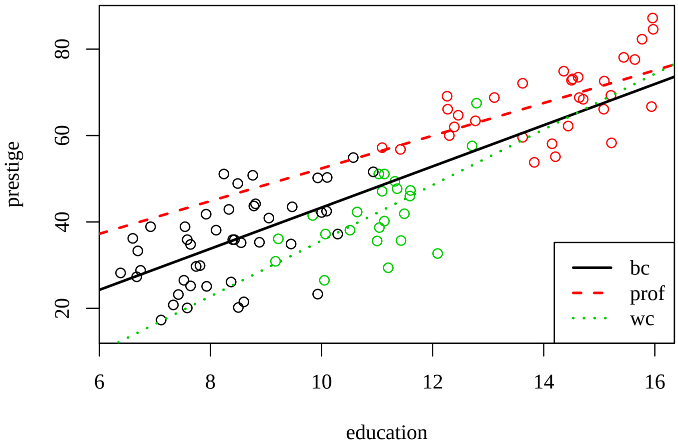
# Fit 3 Regressions, one for each "type" ...

```
Multiple R^2:  0.3064,  Adjusted R^2:  0.2899
F-statistic: 18.56 on 1 and 42 DF,  p-value: 9.709e-05


$prof

Call:
lm(formula = prestige ~ education, data = x)

Residuals:
     Min       1Q   Median       3Q      Max
-13.8450  -4.1613   0.6782   4.8756  12.2557

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  14.5701    12.9892   1.122 0.271190
education     3.7828     0.9179   4.121 0.000288 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.009 on 29 degrees of freedom
Multiple R^2:  0.3693,  Adjusted R^2:  0.3476
F-statistic: 16.98 on 1 and 29 DF,  p-value: 0.0002876
```

# Fit 3 Regressions, one for each "type" ...

```
$wc

Call:
lm(formula = prestige ~ education, data = x)

Residuals:
    Min      1Q  Median      3Q     Max
-16.417  -3.509   1.081   4.865  13.879

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -28.677     19.428  -1.476  0.15477
education      6.435      1.757   3.663  0.00145 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.608 on 21 degrees of freedom
Multiple R^2:  0.3898, Adjusted R^2:  0.3607
F-statistic: 13.41 on 1 and 21 DF,  p-value: 0.001451
```

# Predicted Values from 3 Separate Regressions



# lme4's lmList function offers a quick/convenient way

- pool=F keeps estimates completely separate

```
library(lme4)
lml1 <- lmList( prestige ~ education | type, data=Prestige, pool=F)
summary(lml1)
```

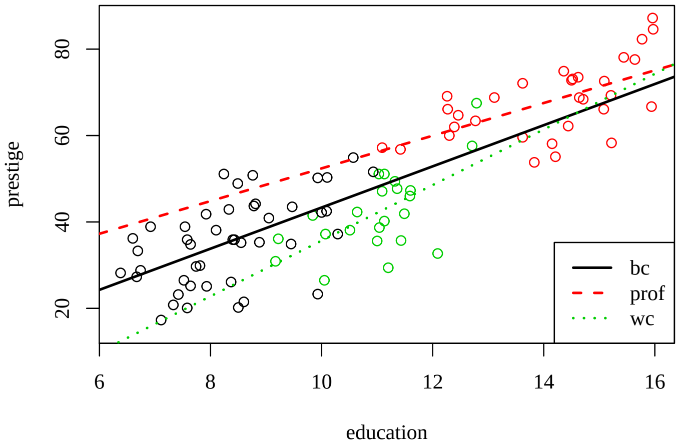
```
Call:
  Model: prestige ~ education | NULL
   Data: Prestige

Coefficients:
   (Intercept)
       Estimate Std. Error    t value  Pr(>|t|)
bc     -4.29360   9.331097 -0.4601388 0.6477900
prof   14.57006  12.989195  1.1217058 0.2711899
wc    -28.67688  19.428032 -1.4760567 0.1547669
   education
      Estimate Std. Error  t value     Pr(>|t|)
bc    4.763651  1.1058084 4.307845 9.708717e-05
prof  3.782846  0.9179125 4.121140 2.876492e-04
wc    6.434589  1.7568147 3.662645 1.451257e-03
```

# lme4's lmList with pooled std. errors.

```
library(lme4)
lml2 <- lmList( prestige ~ education | type, data=Prestige)
summary(lml2)
```

```
Call:
  Model: prestige ~ education | NULL
   Data: Prestige

Coefficients:
   (Intercept)
      Estimate Std. Error    t value  Pr(>|t|)
bc    -4.29360    8.64702 -0.4965409 0.6206972
prof  14.57006   14.50648  1.0043829 0.3178285
wc   -28.67688   19.98744 -1.4347448 0.1547505
   education
     Estimate Std. Error  t value     Pr(>|t|)
bc   4.763651   1.024740 4.648644 1.112981e-05
prof 3.782846   1.025135 3.690096 3.794766e-04
wc   6.434589   1.807400 3.560135 5.889605e-04

Residual standard error: 7.827301 on 92 degrees of freedom
```

# Predicted Values from 3 Separate Regressions



# Why Put 3 Regressions Into One?

- Predicted values are the same, but...
- Smaller standard errors because of “bigger N”
- Easier Hypo Tests to compare group differences on intercept and slope
- But: assumes “homoskedasticity”–same variance of error among 3 data groups

# How to Put 3 regressions into one?

- Keep slopes the same, allow different intercepts
- Allow different slopes and intercepts

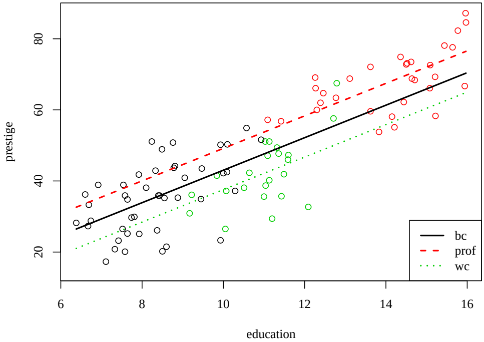
# Keep Slopes the Same

Fit:

```
lm(prestige ~ education + type,
     data=Prestige)
```

| | M1 | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | -2.698 | (5.736) |
| education | 4.573*** | (0.672) |
| typeprof | 6.142 | (4.259) |
| typewc | -5.458* | (2.691) |
| N | 98 | |
| RMSE | 7.814 | |
| $R^2$ | 0.798 | |
| adj $R^2$ | 0.791 | |

$*p \leq 0.05 ** \ p \leq 0.01 *** p \leq 0.001$



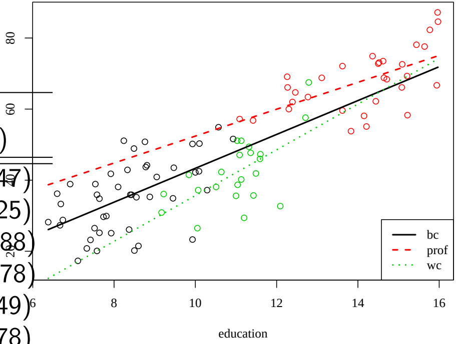
# Interaction Allows Slope and Intercept To Differ

Fit:

```
lm(prestige ~ education * type,
    data=Prestige)
```

| | M1 | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | -4.294 | ( 8.647) |
| education | 4.764*** | ( 1.025) |
| typeprof | 18.864 | (16.888) |
| typewc | -24.383 | (21.778) |
| education:typeprof | -0.981 | ( 1.449) |
| education:typewc | 1.671 | ( 2.078) |
| N | 98 | |
| RMSE | 7.827 | |
| $R^2$ | 0.801 | |
| adj $R^2$ | 0.790 | |

$*p \le 0.05 ** \ p \le 0.01 *** p \le 0.001$



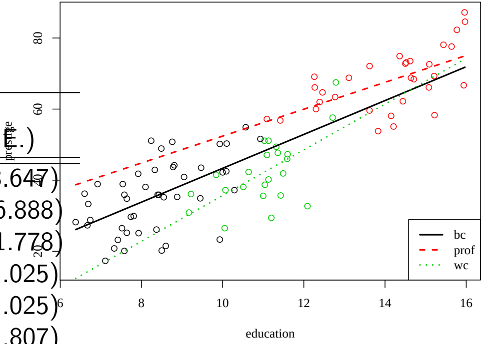
# Another Contrast Coding of Same Model

Fit:

```
lm(prestige ~ type / education,
   data=Prestige)
```

| | M1 | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | -4.294 | ( 8.647) |
| typeprof | 18.864 | (16.888) |
| typewc | -24.383 | (21.778) |
| typebc:education | 4.764*** | ( 1.025) |
| typeprof:education | 3.783*** | ( 1.025) |
| typewc:education | 6.435*** | ( 1.807) |
| N | 98 | |
| RMSE | 7.827 | |
| $R^2$ | 0.801 | |
| adj $R^2$ | 0.790 | |

$*p \leq 0.05 ** \ p \leq 0.01 *** p \leq 0.001$



# Previous plots produced in the "old fashioned way"

```
plot( prestige ~ education , data=Prestige , col=Prestige$type)
m7 <- lm(prestige ~ type / education , data=Prestige)
nd2 <- expand.grid(education=range(Prestige$education), type =
     levels(Prestige$type))
nd2$pred <- predict(m7, newdata=nd2)
for(i in 1:3){
 with(nd2[nd2$type %in% levels(nd2$type)[i], ],
    lines(education , pred , col=i , lty=i , lwd=2))
 }
legend("bottomright", legend=levels(Prestige$type), lty=1:3, col
     =1:3, lwd=2)
outreg(m7, tight=F)
```

# A regression model must have a "predict" method

- Want to run

```
predict(m6, newdata=someDataFrame)
```

- someDataFrame must be a valid data frame with
  - all predictors from m6
  - variables must have EXACT same names and be of same type (numeric, factor)
  - To ascertain names, I often run

```
m6mf <- model.frame(m6)
colnames(m6mf)
```

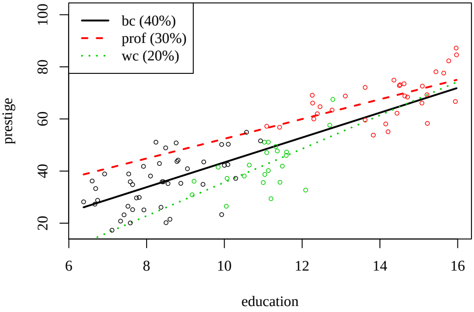
- Note problems if the regression formula has functions in it like "as.factor" or "as.numeric".

```
m7trouble <- lm(log(prestige) ~ as.numeric(education) + as.factor(
     type), data=Prestige)
colnames( model.frame(m7trouble) )
```

```
[1] "log(prestige)"          "as.numeric(education)" "as.factor(type)
 "
```

# Newer rockchalk plotSlopes automates that kind of plot

```
m6ps <- plotSlopes(m6, plotx="education", modx="type")
```



# testSlopes tests "simple slope" lines against 0

```
testSlopes(m6ps)
```

```
These are the straight-line "simple slopes" of the variable
    education
 for the selected moderator values.
                    "type"     slope Std. Error  t value     Pr(>|t|)
bc             education 4.763651   1.024740 4.648644 1.112981e-05
prof  education:typeprof 3.782846   1.025135 3.690096 3.794766e-04
wc      education:typewc 6.434589   1.807400 3.560135 5.889605e-04
```

# How is "plotSlopes" helping? See vignette "rockchalk" in 1.5.4+

- Automatically manipulate "example values" to fill out the newdata object
  - set all "non plotted" variables at "some level"
  - Get "example values" for each one.

# "plotSlopes" WRITES OUT the newdata object it uses

```
m6ps
```

```
$call
plotSlopes.lm(model = m6, plotx = "education", modx = "type")

$newdata
  education type      fit
1      6.38   bc 26.09849
2      6.38 prof 38.70461
3      6.38   wc 12.37580
4     15.97   bc 71.78190
5     15.97 prof 74.98210
6     15.97   wc 74.08350

$modxVals
  bc (40%) prof (30%)   wc (20%)
        bc       prof         wc
Levels: bc prof wc

$col
  bc prof   wc
   1    2    3

$lty
```

# "plotSlopes" WRITES OUT the newdata object it uses ...

```
  bc  prof    wc
   1     2     3

attr(,"class")
[1] "plotSlopes" "rockchalk"
```

# Outline

# MC Problem

- 2 variables "X" and "X*Z" are likely to be correlated with each other
- Consequence: higher standard errors than you might like, smaller t tests
- This is a fundamental problem, whether Z is numeric or categorical. Imagine $X * Z$ if $Z = c(0, 0, 0, 0, 1, 1, 1, 1)$.

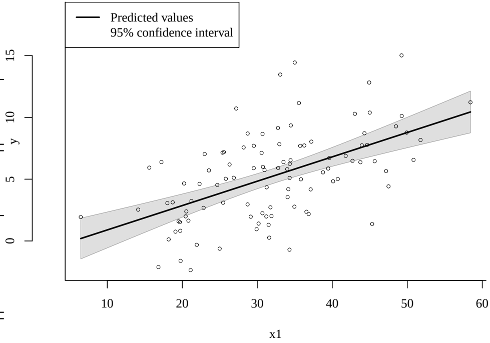
# Mean-Centering Proposed as "Solution"

- West and Aiken propose to fit the regression after replacing
  - $X$, the numeric predictor, with
  - $X - mean(X)$, the "mean centered" predictor.
- Regression printouts with mean centered IVs may seem to have "better" t-tests.

# Remember that the CI Hourglass?

| | m1 | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | -1.080 | (1.016) |
| x1 | 0.197*** | (0.030) |
| N | 100 | |
| RMSE | 3.009 | |
| $R^2$ | 0.302 | |

$*p \leq 0.05 ** \ p \leq 0.01 *** p \leq 0.001$




At the y-axis, the standard error of $\hat{\beta}_0$ and the standard error of the predicted line exactly coincide.

# Repeat: $s.e.(\hat{y})$ when $x1 = 0$

Repeat: At the y-axis, the standard error of $\hat{\beta}_0$ and the standard error of the predicted line exactly coincide.

| | m1 | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | -1.080 | (1.016) |
| x1 | 0.197*** | (0.030) |
| N | 100 | |
| RMSE | 3.009 | |
| $R^2$ | 0.302 | |

$*p \le 0.05 ** p \le 0.01 *** p \le 0.001$

```
predictOMatic(m1, predVals = list("x1" = c
    (0, 10, 20, 30, 40)), se.fit = TRUE,
    interval = "confidence")
```

```
  x1        fit        lwr       upr   fit$
    se.fit
1  0  -1.0800436  -3.0955963  0.935509   1
   .0156642
2 10   0.8916973  -0.5613787  2.344773   0
   .7322247
3 20   2.8634382   1.9246751  3.802201   0
   .4730554
4 30   4.8351792   4.2252686  5.445090   0
   .3073422
5 40   6.8069201   6.0429997  7.570841   0
   .3849498
```

The se's should match where $x1 = 0$. As $x1$ varies from left to right, the se.fit changes.

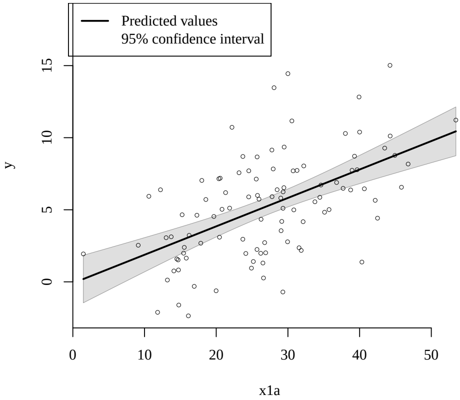
Tempting to talk about row 4. See why?

# Want $s.e.(\hat{y})$ even smaller?

Move the y axis: Subtract 5 from $x1$, move y-axis to the right

| | m2 | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | -0.094 | (0.872) |
| x1a | 0.197*** | (0.030) |
| N | 100 | |
| RMSE | 3.009 | |
| $R^2$ | 0.302 | |

$*p \le 0.05 ** \ p \le 0.01 *** p \le 0.001$

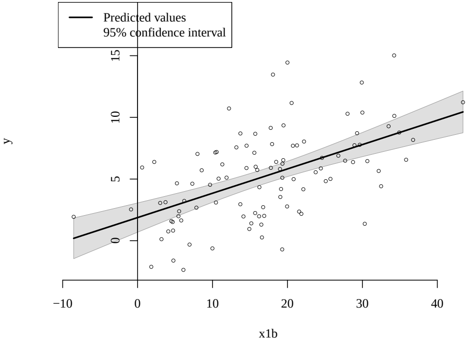


# Push y axis a little bit more to the right

Subtract 15 from $x1$ (15 chosen "from top of my head")

| | m3 | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | 1.878** | (0.598) |
| x1b | 0.197*** | (0.030) |
| N | 100 | |
| RMSE | 3.009 | |
| $R^2$ | 0.302 | |

$*p \le 0.05 ** \ p \le 0.01 *** p \le 0.001$

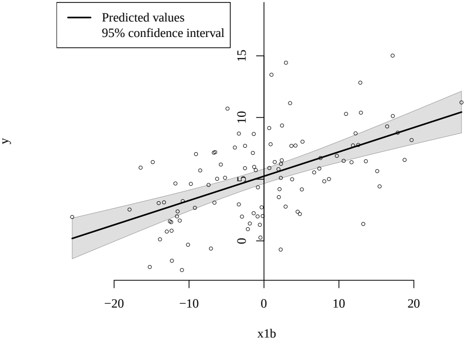



At the y-axis, the standard error of $\hat{\beta}_0$ and the standard error of the predicted line exactly coincide.

# Push y axis to the mean of $x1$

| | m4 | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | 5.242*** | (0.301) |
| x1b | 0.197*** | (0.030) |
| N | 100 | |
| RMSE | 3.009 | |
| $R^2$ | 0.302 | |

$*p \leq 0.05 ** \ p \leq 0.01 *** p \leq 0.001$



# What are you supposed to conclude from that?

- A numeric predictor's slope does not change when we subtract $K$ from it
- But it does change the estimate of the intercept–and the $s.e.(\hat{\beta}_0)$.
- There's no magic in this, however. From model 1, I can estimate the predicted value "at the mean" and I'll have exactly the same substantively important values as I get from model 4.

# Lets check centering in the Prestige regression

| | Not Centered | | Centered | |
|---|---|---|---|---|
| | Estimate | (S.E.) | Estimate | (S.E.) |
| (Intercept) | -4.294 | ( 8.647) | 47.130*** | (2.761) |
| education | 4.764*** | ( 1.025) | . | |
| typeprof | 18.864 | (16.888) | 8.276 | (4.579) |
| typewc | -24.383 | (21.778) | -6.345 | (3.233) |
| education:typeprof | -0.981 | ( 1.449) | . | |
| education:typewc | 1.671 | ( 2.078) | . | |
| educationc | . | | 4.764*** | (1.025) |
| educationc:typeprof | . | | -0.981 | (1.449) |
| educationc:typewc | . | | 1.671 | (2.078) |
| N | 98 | | 98 | |
| RMSE | 7.827 | | 7.827 | |
| $R^2$ | 0.801 | | 0.801 | |
| adj $R^2$ | 0.790 | | 0.790 | |

$*p \leq 0.05 ** \ p \leq 0.01 *** p \leq 0.001$

# Detour about a Mistake I Made While Recoding

- My first effort to create a "mean centered" regression was actually an interesting mistake. I tried this:

```
Prestige$educcenter <- Prestige$education - mean(Prestige$education
    , na.rm=TRUE)
m1 <- lm(prestige ~ education * type, data = Prestige)
m2 <- lm(prestige ~ educcenter * type, data = Prestige)
outreg(list("Not Centered" = m1, "Centered Wrongly" = m2), tight =
    FALSE)
```

- I'm going to call m2 "centered wrongly", but it is not "wrong", so much as evidence of the point to be made later.

# Detour: output

| | Not Centered | | Centered Wrongly | |
|---|---|---|---|---|
| | Estimate | (S.E.) | Estimate | (S.E.) |
| (Intercept) | -4.294 | ( 8.647) | 46.859*** | (2.708) |
| education | 4.764*** | ( 1.025) | . | |
| typeprof | 18.864 | (16.888) | 8.332 | (4.591) |
| typewc | -24.383 | (21.778) | -6.441* | (3.203) |
| education:typeprof | -0.981 | ( 1.449) | . | |
| education:typewc | 1.671 | ( 2.078) | . | |
| educcenter | . | | 4.764*** | (1.025) |
| educcenter:typeprof | . | | -0.981 | (1.449) |
| educcenter:typewc | . | | 1.671 | (2.078) |
| N | 98 | | 98 | |
| RMSE | 7.827 | | 7.827 | |
| $R^2$ | 0.801 | | 0.801 | |
| adj $R^2$ | 0.790 | | 0.790 | |

$*p \leq 0.05 ** \; p \leq 0.01 *** p \leq 0.001$

# Detour: output ...

Its not exactly wrong, but just more evidence you can subtract anything you want and leave the model the same, but superficially different.

# Here's what's wrong about that

- The m2 parameters are not what I expected. It took a long time to understand what was wrong. Why?
- Answer: I calculated the mean with the WRONG data! mean(Prestige$education) used the whole sample Prestige. In contrast, m1 was fit with the "listwise deletion" dataset, where missings on type and education were omitted. We should mean-center with the data that is actually used in the model.
- I should do this:

```
m1mf <- model.frame(m1)
m1mf[ , "education"] <- m1mf[, "education"] - mean(m1mf[ , "
      education"], na.rm = TRUE)
m3 <- lm( prestige ~ education * type, data=m1mf)
summary(m3)
```

# rockchalk meanCenter function avoids that mistake

```
m1mc <- meanCenter(m1)
summary(m1mc)
```

```
These variables were mean-centered before any transformations were
    made on the design matrix.
[1] "educationc"
The centers and scale factors were
       educationc
mean      10.7951
scale      1.0000
The summary statistics of the variables in the design matrix (after
    centering).
                          mean  std.dev.
prestige              47.32755 17.09491
educationc             0.00000  2.74894
typeprof               0.31633  0.46743
typewc                 0.23469  0.42599
educationc:typeprof    1.04043  1.72183
educationc:typewc      0.05319  0.45019

The following results were produced from:
meanCenter.default(model = m1)

Call:
```

# rockchalk meanCenter function avoids that mistake ...

```
lm(formula = prestige ~ educationc * type, data = stddat)

Residuals:
     Min       1Q    Median      3Q      Max
-19.7095  -5.3938  0.8125  5.3968  16.1411

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)           47.1305     2.7609  17.071  < 2e-16 ***
educationc             4.7637     1.0247   4.649 1.11e-05 ***
typeprof               8.2758     4.5791   1.807   0.0740 .
typewc                -6.3453     3.2333  -1.962   0.0527 .
educationc:typeprof   -0.9808     1.4495  -0.677   0.5003
educationc:typewc      1.6709     2.0777   0.804   0.4233
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.827 on 92 degrees of freedom
Multiple R^2:  0.8012, Adjusted R^2:  0.7904
F-statistic: 74.14 on 5 and 92 DF,  p-value: < 2.2e-16
```

# Compare the 3 models

x

| | Centered: Not | | Wrongly | | meanCenter | |
|---|---|---|---|---|---|---|
| | Estimate | (S.E.) | Estimate | (S.E.) | Estimate | (S.E.) |
| (Intercept) | -4.294 | ( 8.647) | 46.859*** | (2.708) | 47.130*** | (2.761) |
| education | 4.764*** | ( 1.025) | . | | . | |
| typeprof | 18.864 | (16.888) | 8.332 | (4.591) | 8.276 | (4.579) |
| typewc | -24.383 | (21.778) | -6.441* | (3.203) | -6.345 | (3.233) |
| education:typeprof | -0.981 | ( 1.449) | . | | . | |
| education:typewc | 1.671 | ( 2.078) | . | | . | |
| educcenter | . | | 4.764*** | (1.025) | . | |
| educcenter:typeprof | . | | -0.981 | (1.449) | . | |
| educcenter:typewc | . | | 1.671 | (2.078) | . | |
| educationc | . | | . | | 4.764*** | (1.025) |
| educationc:typeprof | . | | . | | -0.981 | (1.449) |
| educationc:typewc | . | | . | | 1.671 | (2.078) |
| N | 98 | | 98 | | 98 | |
| RMSE | 7.827 | | 7.827 | | 7.827 | |
| $R^2$ | 0.801 | | 0.801 | | 0.801 | |
| adj $R^2$ | 0.790 | | 0.790 | | 0.790 | |

$*p \leq 0.05 ** \; p \leq 0.01 *** p \leq 0.001$

x

# There's an interesting flaw here

- The one that is "wrongly centered" has more stars!
- Makes you wonder, if you fiddle around subtracting constants from your predictors, could you make more stars appear?
- Don't bother, next slide will explain

# But They Are All Actually The Same Model!

All of these models, even the wrongly centered one, generate the same predicted values

```
predictOMatic(m1, predVals = c("
    education"))
```

```
  education type      fit
1     6.380   bc 26.09849
2     8.445   bc 35.93543
3    10.605   bc 46.22492
4    12.755   bc 56.46677
5    15.970   bc 71.78190
```

```
predictOMatic(m2, predVals = c("
    educcenter"))
```

```
  educcenter type      fit
1 −4.3580392   bc 26.09849
2 −2.2930392   bc 35.93543
3 −0.1330392   bc 46.22492
4  2.0169608   bc 56.46677
5  5.2319608   bc 71.78190
```

```
predictOMatic(m1mc, predVals = c("
    educationc"))
```

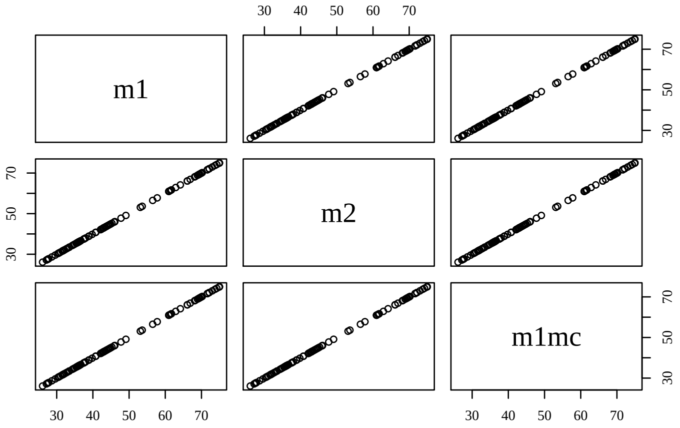
```
  educationc type      fit
1  −4.415102   bc 26.09849
2  −2.350102   bc 35.93543
3  −0.190102   bc 46.22492
4   1.959898   bc 56.46677
5   5.174898   bc 71.78190
```

Note, the predictor values are "shifted", but predictions identical.

# But They Are All Actually The Same Model! ...

Even for the "wrongly centered" model. You can subtract anything you want from any predictor, and the predicted value ends up the same!

# Plot the predicted values against each other



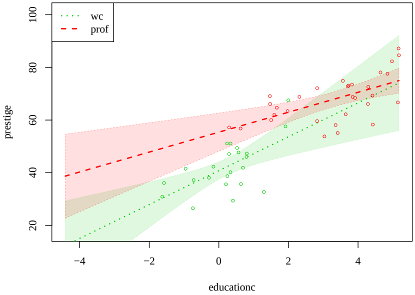
# But They Are Actually The Same Model!

```
anova(m2, m1, m1mc, test = "F")
```

```
Analysis of Variance Table

Model 1: prestige ~ educcenter * type
Model 2: prestige ~ education * type
Model 3: prestige ~ educationc * type
  Res.Df    RSS Df    Sum of Sq F Pr(>F)
1      92 5636.5
2      92 5636.5  0   9.0949e-13
3      92 5636.5  0  -9.0949e-13
```

# So Why Do They Seem Different?

- Centering–subtracting a constant from ALL cases in a dataset–moves the y axis.
- If you re-position the y-axis, you get a new "snapshot" estimate of the intercept, or group-specific intercept shifts.
- Here we have 3 "types" but they are all centered by same mean value.



# So Why Do They Seem Different? ...

- Centering by the "grand mean" does not necessarily put the estimate for a particular subgroup at the "most significant spot".
- rockchalk install has "examples" folder has full worked example "centeredRegression.R"

# Outline

# Problems

1. I'm working on an R function to automatically plot interactions involving categorical variables. The function is currently called "catplot" and it is circulating in a file called "plotCategorical.R". Please try that out.
2. There are several functions available in R packages to draw plots of categorical interactions. Try these:
    1. In package HH, the function "ancova" makes a plot that is interesting. Here's some code that works, and it saves a copy of the hotdog data for you in a file "hotdog.RData".

```
library(HH)
hotdog <- read.table(hh("datasets/hotdog.dat
    "), header=TRUE)
save(hotdog, file="hotdog.RData")
## This is the usual usage for NO
    interaction
## y ~ x + a or y ~ a + x
```

# Problems ...

```
## constant slope, different intercepts
ancova(Sodium ~ Calories + Type, data=hotdog
    )
ancova(Sodium ~ Type + Calories , data=hotdog
    )

## y ~ x * a or y ~ a * x for an interaction
## different slopes, and different
    intercepts
ancova(Sodium ~ Calories * Type, data=hotdog
    )
ancova(Sodium ~ Type * Calories , data=hotdog
    )
```

I mention that one because it gives you the hotdog data set.

2. In the base graphics package's there is "coplot". The "car" package has a nice little panel plugin to plot lm models. See if this is fun:

# Problems ...

```
coplot(Sodium ~ Calories | Type, data=hotdog
   )
coplot(Sodium ~ Calories | Type, data=hotdog
   , panel = function(x, y, ...)
   panel.smooth(x, y, span = .8, ...) )
library(car)
coplot(Sodium ~ Calories | Type, panel=
   panel.car, lowess.line=F, col=c("blue"),
   data=hotdog)
coplot(Sodium ~ Calories | Type, panel=
   panel.car, lowess.line=T, col=c("blue"),
   data=hotdog)
coplot(Sodium ~ Calories | Type, panel=
   panel.car, rows=1, lowess.line=F, col=c("
   blue"), data=hotdog)
```

# Problems ...

3. I wish you'd learn how to do these plots with the lattice and the ggplot2 packages. The xyplot function in lattice is made for this kind of thing, but for some reason I just can't concentrate hard enough to master all of the options. ggplot2 also makes lovely plots, but I can't wrap my mind around the term "aesthetic" as it is used in that documentation.