Regression Presentations: Plots

Paul E. Johnson[1] [2]

[1]Department of Political Science

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

2013

Presenting Regressions

We Need

- Nice Looking Regression Tables (that's a separate set of notes. Look for regression-tables-1)
- Calculate & plot predicted values and confidence intervals
- This lecture is about 2-d plots
- For 3-d plots, look for plots-3d

Orientation #1

- One of the primary points of emphasis in the S & R languages is the development of a consistent framework for the estimation and probing of regression models.
- We fit a model, then "interrogate" it.
- A predict method should be supplied with *any* regression model.
  - provides predicted values (and, hopefully, confidence intervals) for user-specified input values.
  - If there is no predict method, then we are in the stone age. We have to manually extract the slope coefficients (`coef(m1)`), create a matrix of interesting predictor values (`X`), and calculate `X %*% coef(m1)`.

predict method: key arguments

predict() is a generic function, each regression model is responsible to implement predict."model"()

object:	a fitted regression model
newdata:	values of predictors for which predictions are sought
interval:	possibly a confidence or prediction interval (may not be available, depending on model type)
type:	Predictions may be supplied on various scales. In GLM, for example "response" or "link"

The creation of a "newdata" object is a struggle

- If `X %*% coef(m1)` is the stone age, then predict with newdata is the golden age of Rock and Roll.
- Various suggestions have been offered to streamline the creation of newdata objects and calculate predictions. These things put us in the Space Age.

  Space Age: technological things work well, except when they don't, and we all feel helpless when that happens.

- Because the Space Age tools sometimes fail, we still teach the "older ways" of predict and newdata. But we wish we didn't have to because it makes the students cry.

Glimpse the Space Age

- Let's stop right here to get an orientation. Lets all run this:

```
library(rockchalk)
example(predictOMatic)
## Or create a fresh example for yourself with:
dat <- genCorrelatedData2(1000,
      means = c(100,200,100, 40),
      sds = c(10, 20, 20, 5),
      rho = 0.2,
      beta = c(0.1, 0.3, 0.3, 0.1, -0.14, 0, 0.025),
      stde = 140)
dat$xcat1 <- factor(sample(c("No","Yes","Yes"), 1000, replace =
      TRUE))
m1 <- lm(y ~ x1 + x2 + x3 + x4 + xcat1, data = dat)
summary(m1)
predictOMatic(m1)
predictOMatic(m1, divider = "std.dev.")
predictOMatic(m1, divider = "seq", n = 7)
```

- In the following, I want to help you understand what's going on in there and why it is useful.

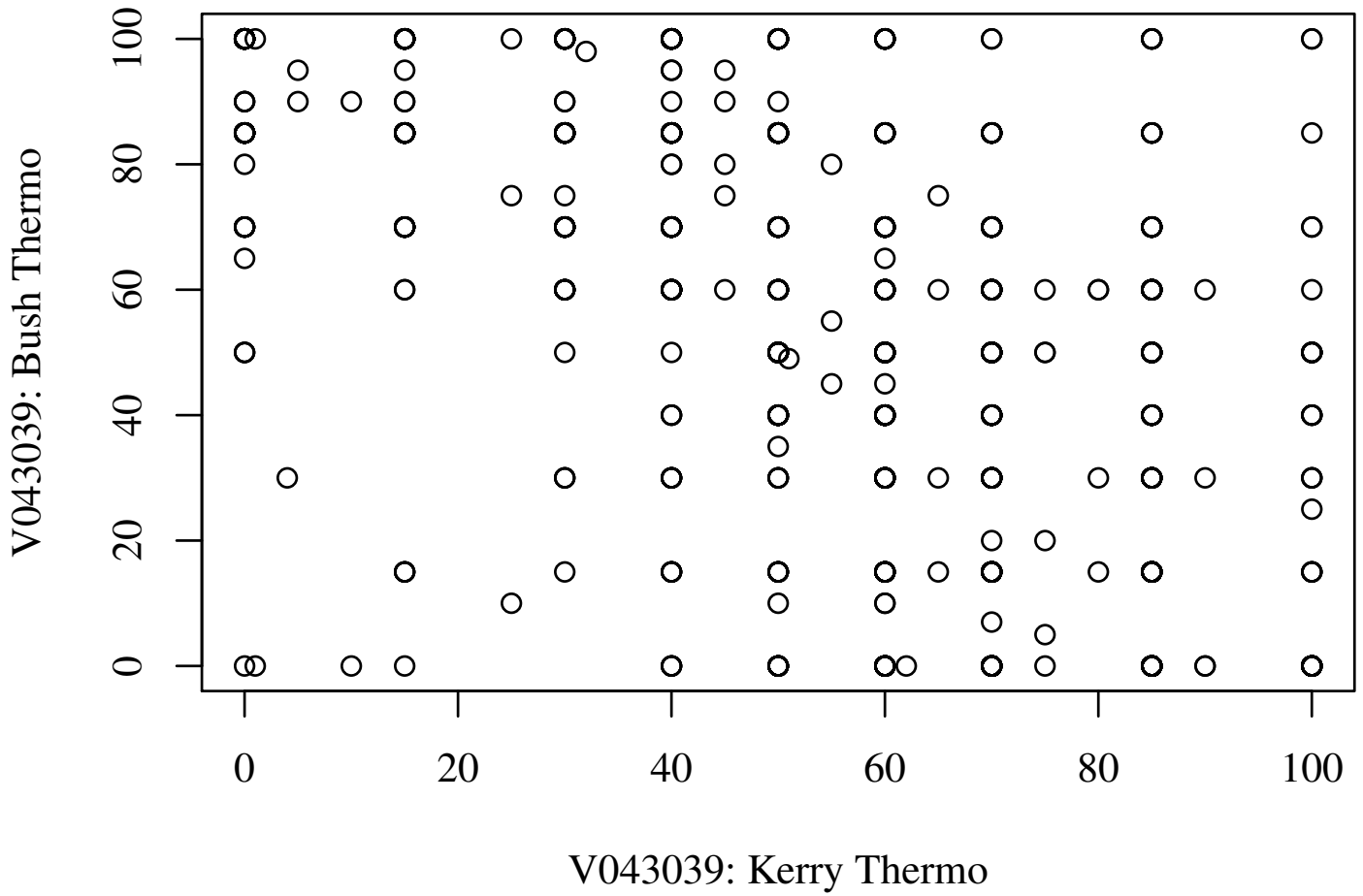
Example Data Used in these Notes

- American National Election Study (2004), recoding details follow.
- Chile dataset in car package
- Simulated data (rockchalk package genCorrelatedData)

Re-use the Coding for the American National Election Study 2004

```
## V043038      B1a. Feeling Thermometer: GW Bush
## V043039      B1b. Feeling Thermometer: John Kerry
## V043210      R1. R position on gay marriage
## V043213      S3. National economy better/worse since GW
     Bush took ofc
## V045117      G4a. Liberal/conservative 7-point scale:
     self-placement
## V045145X     H5x. Summary: Pre-Post US flag makes R feel
## V041109A     HHListing.9a. Respondent gender
## V043116      J1x. Summary: R party ID
## V043250      Y1x. Summary: Respondent age
```

Thermometer Scores for Bush and Kerry



Create a New Dependent Variable

The difference in thermometer scores:

```
mydta1$th.bush.kerry <- mydta1$V043038 - mydta1$V043039
```

Clean up a bunch of variables & value labels

```
##Party
mydta1$V043116 <- mydta1$V043116[,drop = TRUE]
levels(mydta1$V043116) <- c("SD","WD","ID","I","IR","WR","SR","O")
mydta1$V043116[ mydta1$V043116 %in% levels(mydta1$V043116)[8] ] <-
    NA
mydta1$V043116 <- mydta1$V043116[, drop=TRUE]
table(mydta1$V043116)
```

```
 SD  WD  ID   I  IR  WR  SR
203 179 210 118 138 154 193
```

```
##IDEO
mydta1$V045117 <- mydta1$V045117[ ,drop = TRUE]
levels(mydta1$V045117) <- c("EL","L","SL","M","SC","C","EC")
table(mydta1$V045117)
```

Create a New Dependent Variable ...

```
 EL   L  SL   M  SC   C  EC
 20 103 125 279 143 166  31
```

```
##Gender
levels(mydta1$V041109A) <- c("M","F")
## Gay Marriage
levels(mydta1$V043210)
```

```
[1] "1. Should be allowed"                                          "3.
    Should not be allowed"
[3] "5. Should not be allowed to marry but should be allowed" "VOL"
[5] "8. Don't know"                                                 "9.
    Refused"
```

Create a New Dependent Variable ...

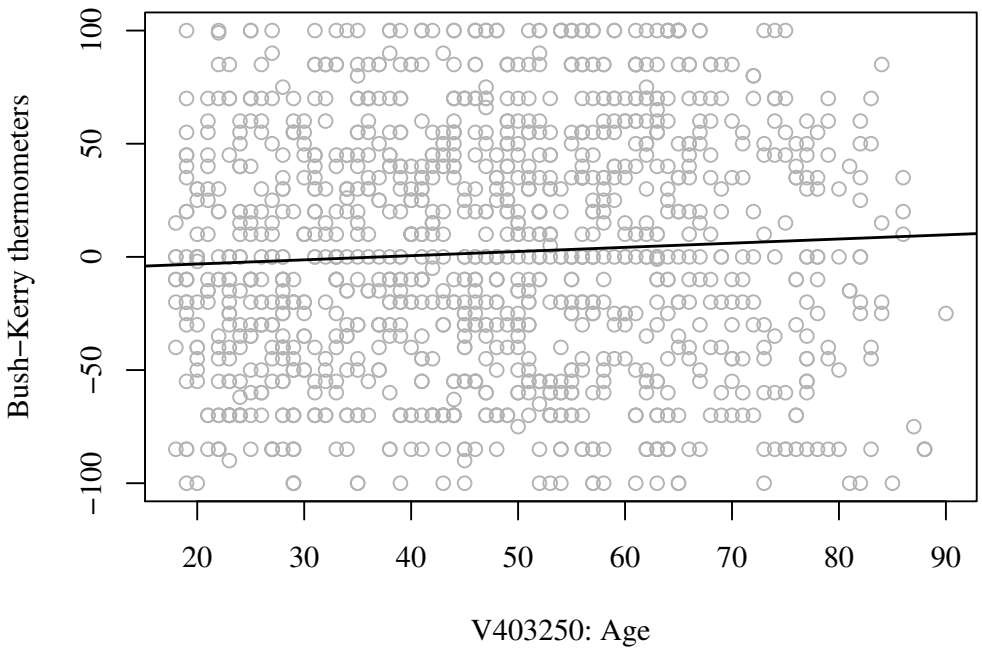
```
mydta1$V043210[ mydta1$V043210 %in% levels(mydta1$V043210)[4:10] ]
    <-NA
mydta1$V043210 <- mydta1$V043210[, drop = TRUE]
levels(mydta1$V043210) <- c("Allow","No","Med")
## Economy
mydta1$V043213 <- mydta1$V043213[ ,drop = TRUE]
l <- levels(mydta1$V043213)
econnew <- factor(mydta1$V043213, levels=c(l[2],l[3],l[1]), labels=
    c("Worse","Same","Better"))
table(mydta1$V043213, econnew)
```

```
             econnew
              Worse Same Better
 1. Better        0    0    190
 3. Worse       668    0      0
 5. The same      0  343      0
```

```
mydta1$V043213 <- econnew
rm(econnew)
##Flag
mydta1$V045145X <- mydta1$V045145X[,drop = TRUE]
```

Plotting Regressions: Here is the General Plan

- Put a continuous predictor on the horizontal axis ($x1_i$)
- Put the dependent variable on the vertical axis (y_i)
- Calculate $\hat{y}_i = \hat{b}_0 + \hat{b}_1 x1_i$ and plot that "predicted line"



Outline

Previous plot requires 3 Easy Steps (That *Almost Every* R user has taken)

1) lm 2) plot 3) abline.
Example:

```
## Step 1
mod1age <- lm(th.bush.kerry ~ V043250, data = mydta1)
## Step 2
plot(th.bush.kerry ~ V043250, type = "p", xlab = "V403250: Age",
     ylab = "Bush-Kerry thermometers", data = mydta1, col = gray(.7)
     )
## Step 3
abline(mod1age, lwd = 1.5)
```

abline: behind the scenes.

- abline() is a general purpose "straight line drawing tool". It can receive input in the form of coefficients, or, if you provide it with a regression object, it will notice and try to get the coefficients from the regression object.
- At the command line, run abline, you'll see it. Look for "if (is.object(a) || is.list(a)) {"

```
abline
```

abline: behind the scenes. ...

```
function (a = NULL, b = NULL, h = NULL, v = NULL, reg = NULL,
    coef = NULL, untf = FALSE, ...)
{
    int_abline <- function(a, b, h, v, untf, col = par("col"),
        lty = par("lty"), lwd = par("lwd"), ...) .External.graphics(
            C_abline,
        a, b, h, v, untf, col, lty, lwd, ...)
    if (!is.null(reg)) {
        if (!is.null(a))
            warning("'a' is overridden by 'reg'")
        a <- reg
    }
    if (is.object(a) || is.list(a)) {
        p <- length(coefa <- as.vector(coef(a)))
        if (p > 2)
            warning(gettextf("only using the first two of %d
                regression coefficients",
                p), domain = NA)
        islm <- inherits(a, "lm")
        noInt <- if (islm)
            !as.logical(attr(stats::terms(a), "intercept"))
        else p == 1
        if (noInt) {
            a <- 0
```

abline: behind the scenes. ...

```
            b <- coefa[1L]
        }
        else {
            a <- coefa[1L]
            b <- if (p >= 2)
                coefa[2L]
            else 0
        }
    }
    if (!is.null(coef)) {
        if (!is.null(a))
            warning("'a' and 'b' are overridden by 'coef'")
        a <- coef[1L]
        b <- coef[2L]
    }
    int_abline(a = a, b = b, h = h, v = v, untf = untf, ...)
    invisible()
}
<bytecode: 0x338ea60>
<environment: namespace:graphics>
```

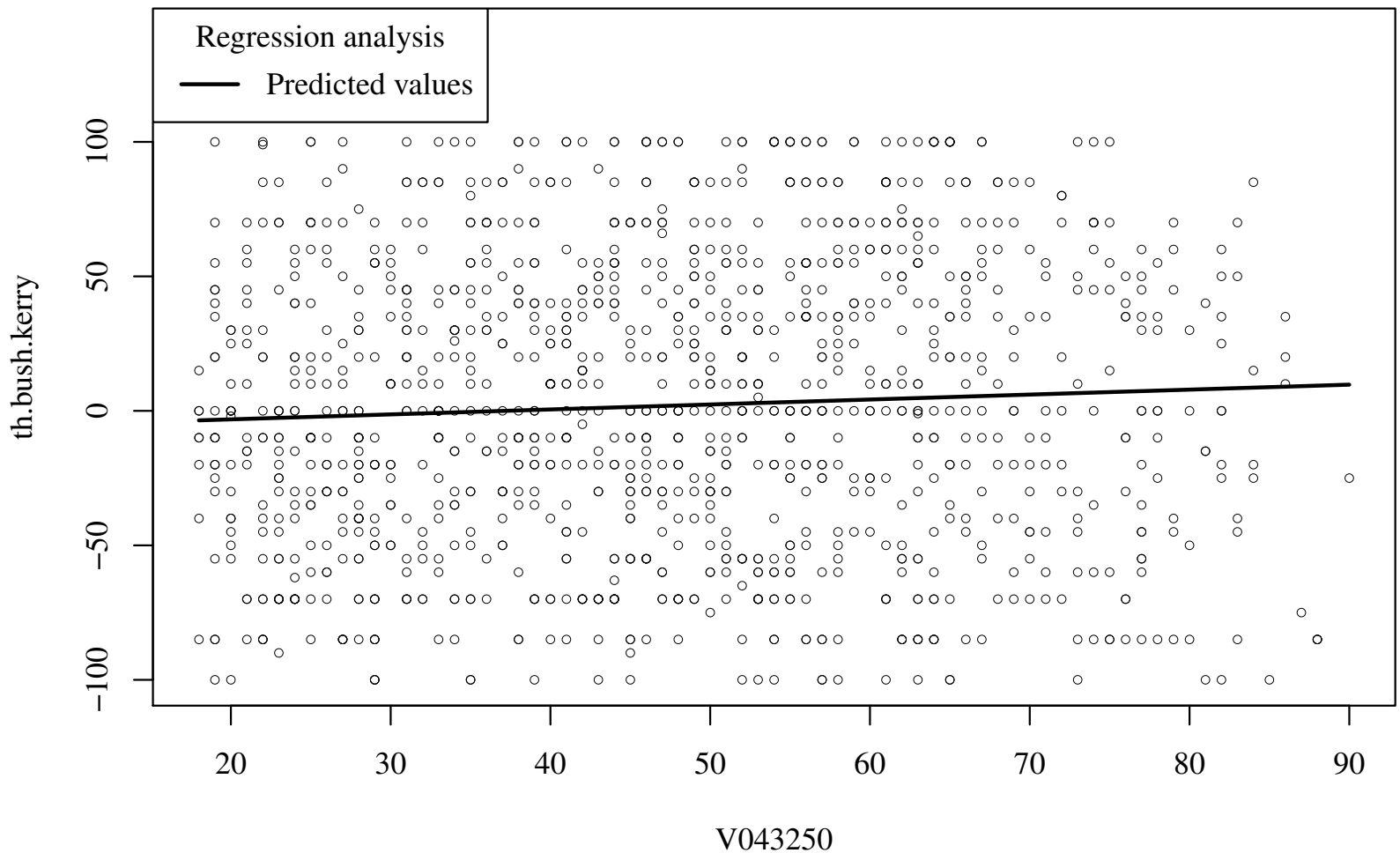
Getting away from abline()

This lecture, it turns out, is mostly about getting away from the limitations of abline()!

- limitations of abline()
  1. Doesn't work when there are more predictors.
  2. Doesn't work with nonlinear models or glm or ....
- The R paradigm suggests instead a 3 step sequence.
  1. newdata: create an exemplar data set of interesting values for which we might want predictions
  2. predict w/newdata: calculate predictions
  3. plot!

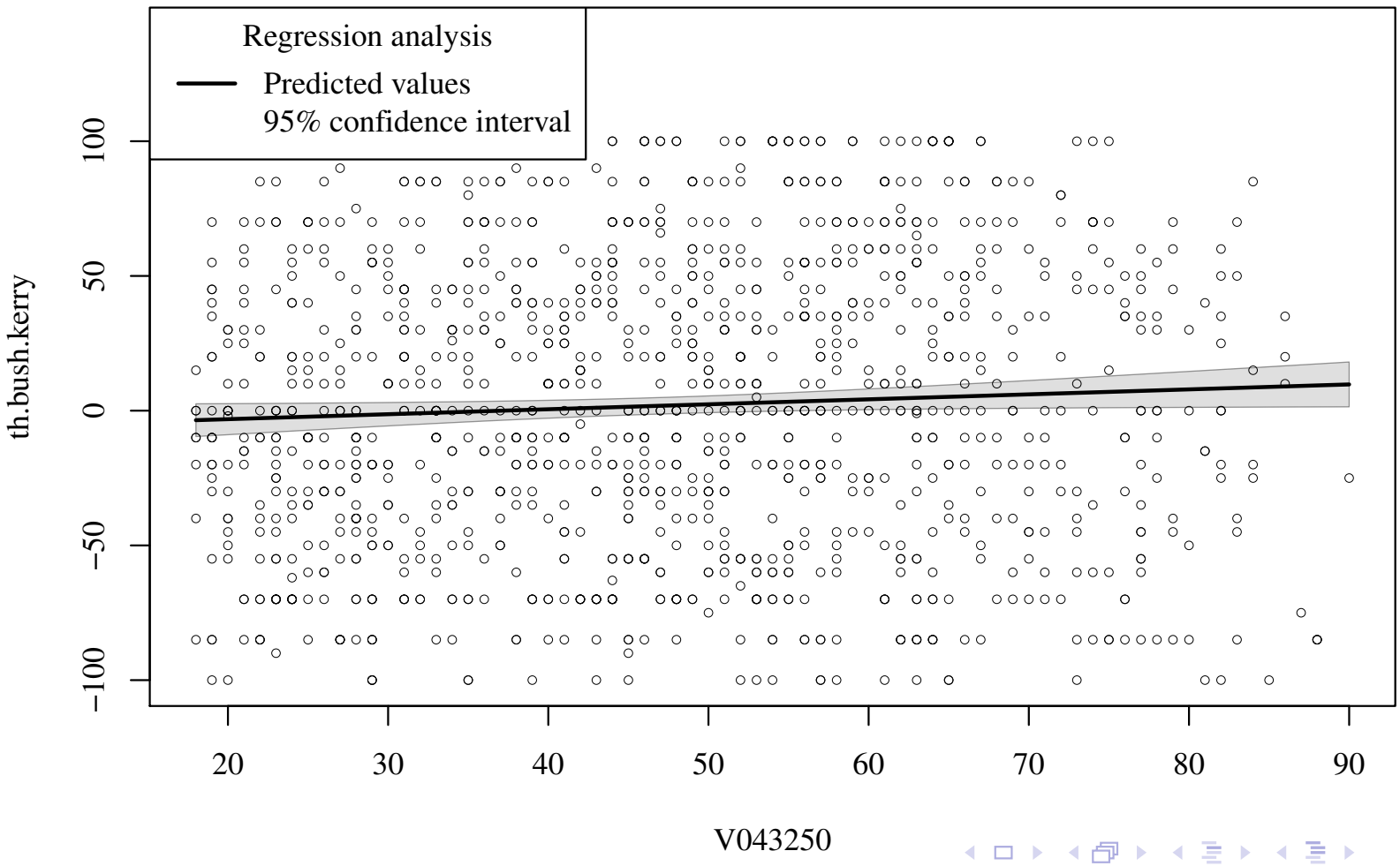
We want to produce a plot like this

```
plotSlopes(mod1age, plotx = "V043250", lwd = 0.3)
```



Request the 95% Confidence Interval

```
plotSlopes(mod1age, plotx = "V043250", interval = "confidence", lwd
    = 0.3)
```



The newdata -> predict -> plot

I made that with rockchalk::plotSlopes(). In a model with only 1 predictor, it is pretty easy to make a plot like that.

1. Create a set of "focal values": possible values of "x" for which we need predictions.
   - In a one-predictor linear model, "two points define a straight line". We might as well take the target values at the edges of the data:
     - `V03250r <- range(mydta1$V03250, na.rm = TRUE)`
   - However, if we want confidence intervals, we'll need an evenly spaced sequence of points, something like
     - `V03250r <- rockchalk::plotSeq(mydta1$V03250, length.out = 40)`
2. Create a new data frame using the new sequence as the independent variable

```
ndf <- data.frame(V03250 = V03250r)
```

The newdata -> predict -> plot ...

3. Use predict to get predictions, one for each row in the new data frame.

```
predict(m0, newdata = ndf)
```

or possibly

```
predict(m0, newdata = ndf, interval = "confidence")
```

This Code Would Work

```
ndf <- data.frame(V043250 = range(mydta1$V043250))
ndf$fit <- predict(mod1age, newdata = ndf )
head(ndf)
```

```
  V043250        fit
1      18 -3.524611
2      90  9.742260
```

```
plot(th.bush.kerry ~ V043250, data = mydta1, xlab = "V403250: Age",
     ylab = "th.bush.kerry = Bush-Kerry thermometers", col = gray(
    .7)) ## Scatterplot
lines(fit ~ V043250, data = ndf, lwd = 1.5)
```

Outline

Confidence Intervals

- predict methods for some regression models can calculate 95% intervals of various type.
- lm:
  - interval="confidence" to obtain the 95% confidence intervals for the estimate of the expected value of y given the predictors
  - interval="predict" to obtain the 95% confidence intervals for the outcome y_i given the predictors.
- When intervals are requested, predict returns a matrix with columns called

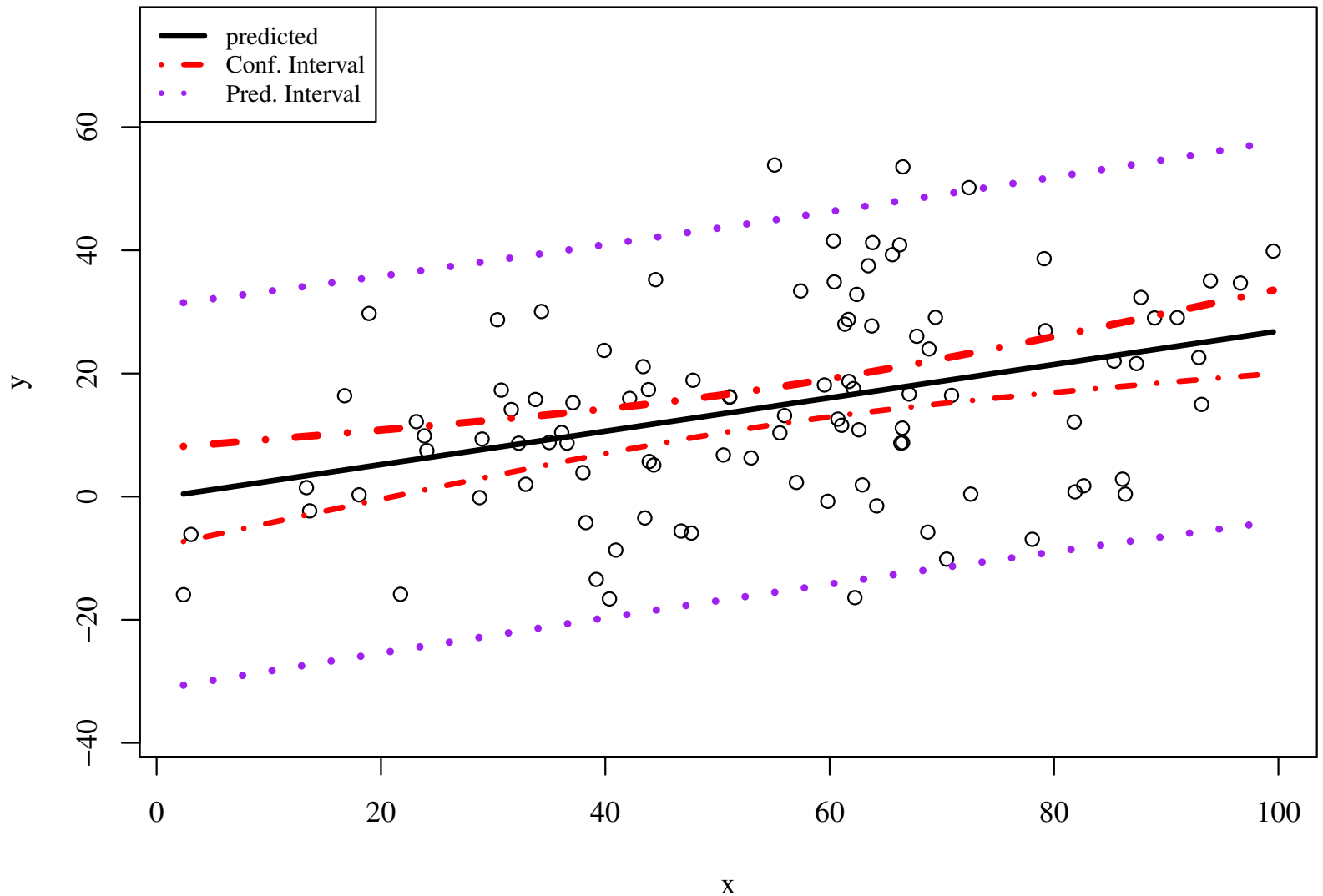
```
fit lwr upr
```

  Predicted ("fitted") values, and the lower and upper limits of the intervals.
- R's predict.glm does not supply intervals (complicated statistical controversy behind that), but they will provide standard errors of fitted values, from which we can create our own confidence intervals

Simple example to plot Confidence and Prediction Intervals

```
x <- rnorm(100, m = 50, s = 20)
y <- 3 + 0.2 * x + 15 * rnorm(100)
mp1 <- lm(y ~ x)
ndf <- data.frame(x = plotSeq(x,40))
p1 <- as.data.frame(predict(mp1, interval="conf", newdata = ndf))
p2 <- as.data.frame(predict(mp1, interval="pred", newdata = ndf))
plot(x, y, ylim = magRange(y, 1.3))
lines(ndf$x, p1$fit, col = "black", lwd=3)
lines(ndf$x, p1$lwr, col = "red", lwd = 3, lty = 4)
lines(ndf$x, p1$upr, col = "red", lwd = 4, lty = 4)
lines(ndf$x, p2$lwr, col = "purple", lwd = 4, lty = 3)
lines(ndf$x, p2$upr, col = "purple", lwd = 4, lty = 3)
legend("topleft", legend = c("predicted","Conf. Interval", "Pred.
    Interval"), col = c("black", "red", "purple"), lty = c(1, 4,3),
     cex = 0.8, lwd = 3)
```

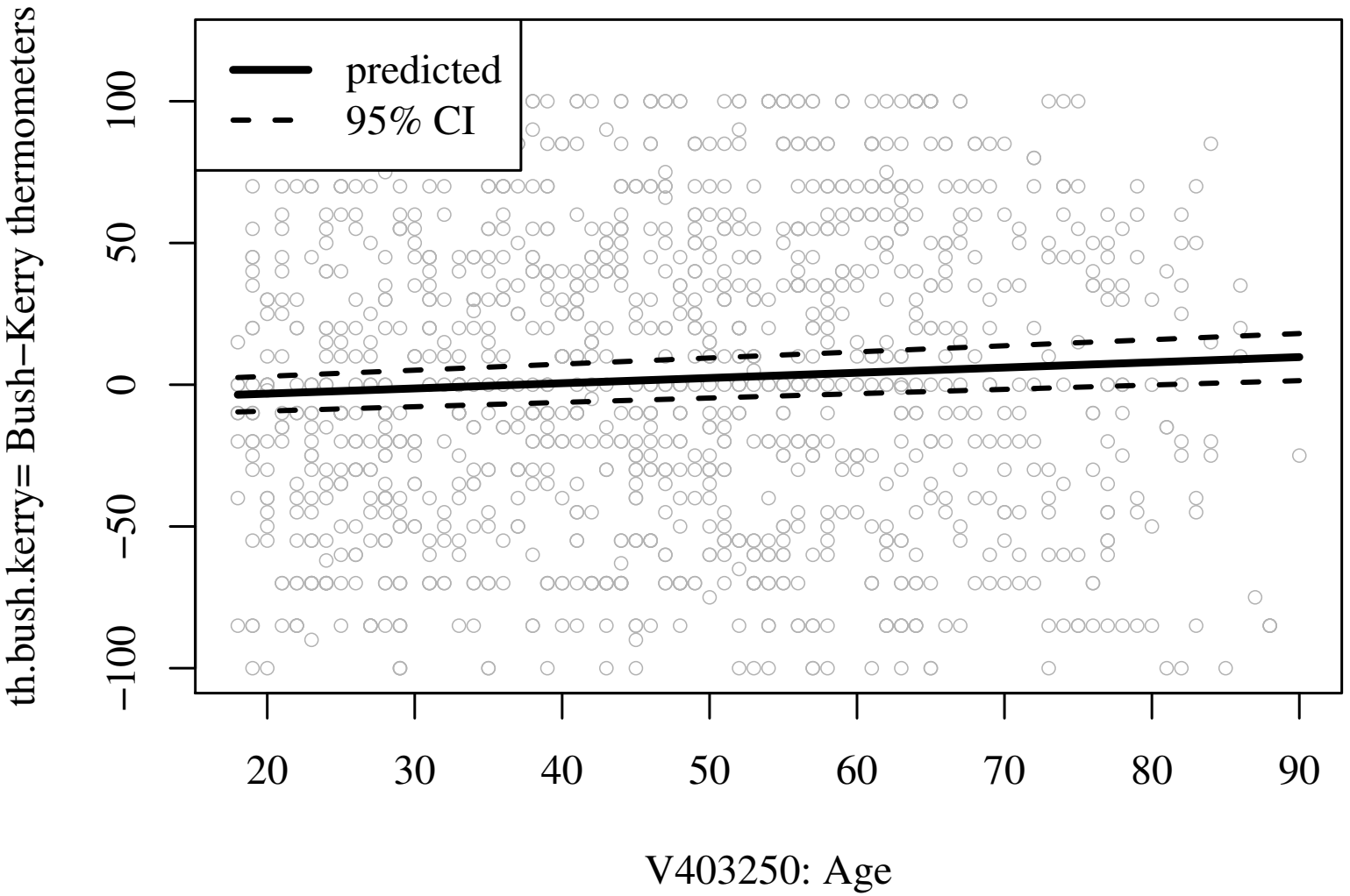
Difference b/t Conf and Pred Intervals



Confidence Intervals for the Bush-Kerry Regression

```
ndf1 <- data.frame(V043250 = range(mydta1$V043250))
mod1pred <- predict(mod1age, newdata = ndf1, interval="confidence")
ndf1 <- cbind(ndf1, mod1pred)
lines(fit ~ V043250, data = ndf1, lwd = 3, lty = 1)
lines(lwr ~ V043250, data = ndf1, lwd = 2, lty = 2)
lines(upr ~ V043250, data = ndf1, lwd = 2, lty = 2)
legend("topleft", legend = c("predicted","95% CI"), lty = c(1,2),
    lwd = c(3,2), bg = "white")
```

The Fitted (Predicted) Value and 95% Confidence Interval



Outline

What's that "newdata" argument in predict?

- Run predict without 'newdata', what do you get?
  - one prediction for each row in the data, same as output from fitted()

```
fit <- predict(mod1age)
head(fit)
```

```
         1          2          3          4          5          6
2.37177573 1.81898943 -0.02363155 6.24127979 4.58292091 2.92456202
```

```
head(fitted(mod1age))
```

```
         1          2          3          4          5          6
2.37177573 1.81898943 -0.02363155 6.24127979 4.58292091 2.92456202
```

- That does not simplify anything.

The Transition to Multiple Regression

- We want a prediction for a particular combination of predictors, to answer "what would happen if?" questions.
  - e.g., I need to know what young left-handed women feel about something
- How to choose particular values of predictors?
  - quantiles?
  - mean +/- standard deviation?
- Making a newdata objects that work requires some discipline because values must be included for all variables.

Building a newdata object

- Step 1. For each variable in the model, create a collection of one or more example values for which predictions are to be calculated (can use quantiles, means, medians, or anything else).
- Step 2. R's expand.grid() will create a "mix and match" data frame.

```
focalValues <- data.frame(x1 = c(1, 2, 3, 4), x2 = c(100,200))
mAndM <- expand.grid(focalValues)
mAndM[1:8, ]
```

```
  x1  x2
1  1 100
2  2 100
3  3 100
4  4 100
5  1 200
6  2 200
7  3 200
8  4 200
```

Tricky Thing about newdata argument

predict(m1, newdata = mAndM) will fail if

- newdata does not include one named column for each variable included in the model (here, m1)
- any values for factor variables included in mAndM are not valid labels.
- The model was specified in an unexpected way.

BAD: m1 <−lm(dat$y ∼dat$x1 + dat$x2)
GOOD: m1 <−lm(y ∼x1 + x2, data = dat)

- The model includes transformations like `as.numeric(x)` or `as.factor(x)`. These complicate the creation of newdata quite a bit.

Here's an example that works with the Chile data

- Using the Chile data from the car package, with 2 numeric (income, age) and 1 factor (sex).

```
require(car)
data(Chile)
Chile$income <- Chile$income/1000
m2 <- lm(statusquo ~ income + age + sex, data=Chile)
```

- statusquo : scale score of support for the status quo
- sex: M or F
- income (pesos, rescaled to 1000s of pesos)
- region (5 regions of Chile)

- Could use means, medians, or whatever else strikes our fancy. This creates just one example row

```
newdf <- data.frame(income = mean(Chile$income, na.rm=TRUE),
     age=median(Chile$age, na.rm = TRUE), sex = "M")
newdf$fit <- predict(m2, newdata = newdf)
newdf
```

Here's an example that works with the Chile data ...

```
    income age sex        fit
1 33.87586  36   M -0.1040288
```

Incidentally, Concerning Factors in newdata

- If we spell the factor levels incorrectly, this won't run.
- Thus there is an argument in favor of not spelling them, but rather asking for the valid levels and then using them in the newdata

```
sexlev <- levels(Chile$sex)
newdf <- data.frame(income=c(mean(Chile$income, na.rm = T)),
     age = c(55), sex = sexlev[1])
newdf$fit <- predict(m1, newdata = newdf)
```

- The valid levels are held in sexlev, and sexlev[1] is the first valid level.

Getting good results with newdata will require practice

- Be patient, it can be made to work.
- Lets pause for some practice. There are several WorkingExamples that have newdata elements. I suggest that everybody should step through number 1, then pick randomly between 2-4.
  1. regression-plot-factor-01.R
  2. regression-predictedPlots-01.R
  3. regression-predictedPlots-02.R
  4. regression-quadratic-1.R
- Please open one and "step through" to see what happens

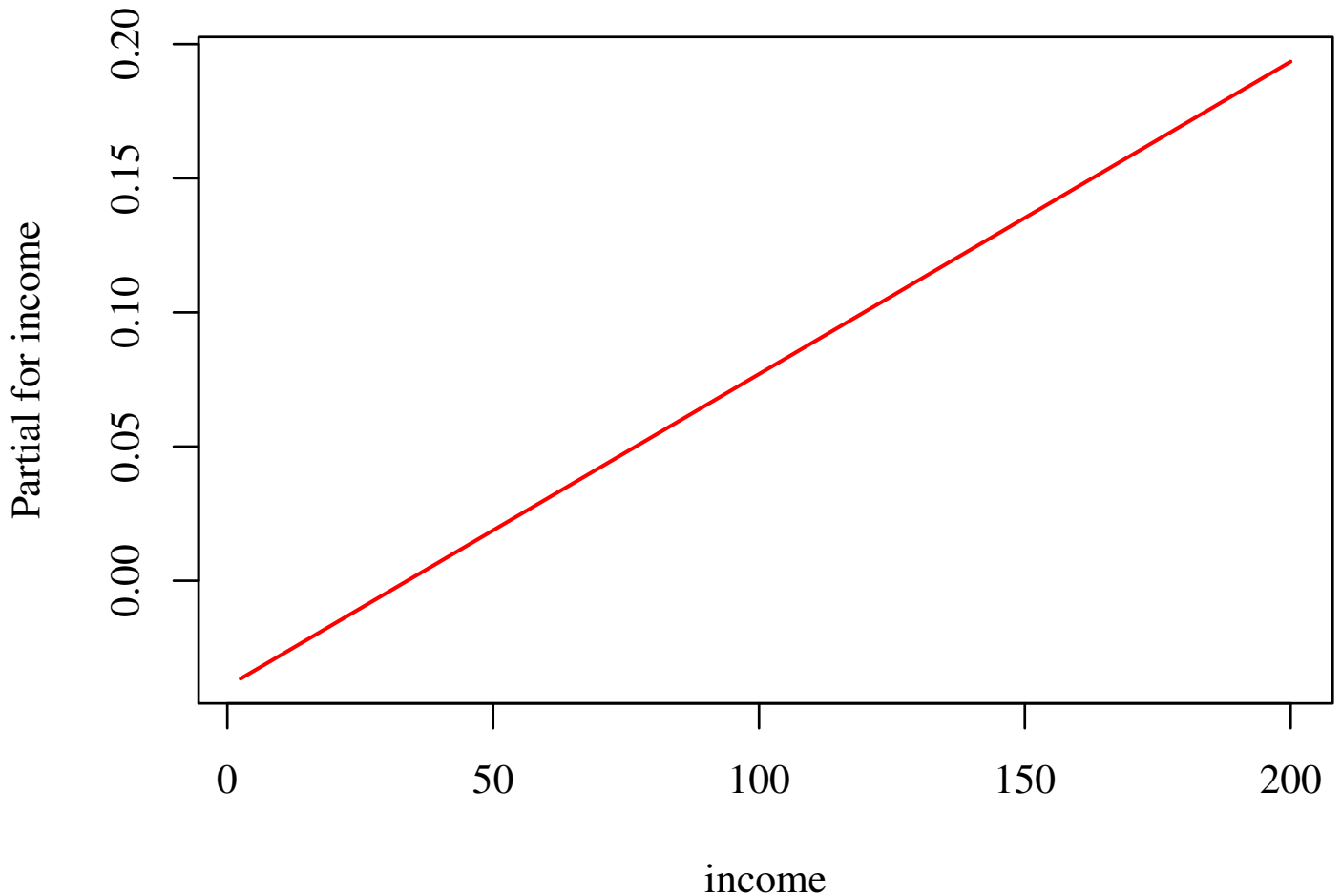
termplot is provided with R

- Because termplot is provided with R, it is my favorite tool for a "first quick look" at the regression model.
- It
  - has proper displays for both numeric and factor predictors
  - has many arguments for customization of plots
  - Automatically graphs each predictor, one after the other
- Disadvantages
  - does not handle models with interaction effects
  - does not respond correctly to some complicated model formulae

Look at a small model on the Chile data

```
termplot(m2)
```

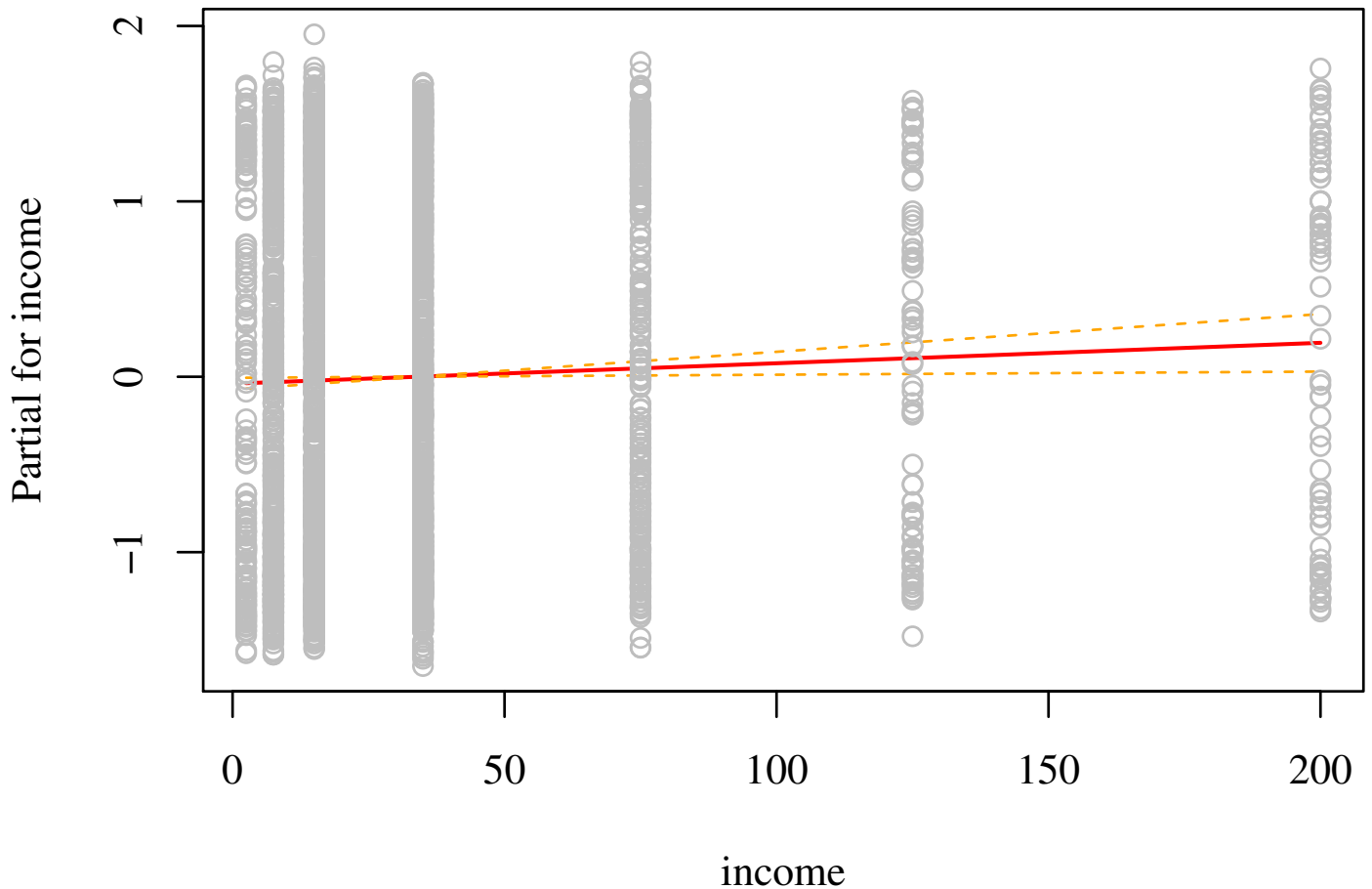
Will prepare a simple line plot for each variable, I'll show you "income"



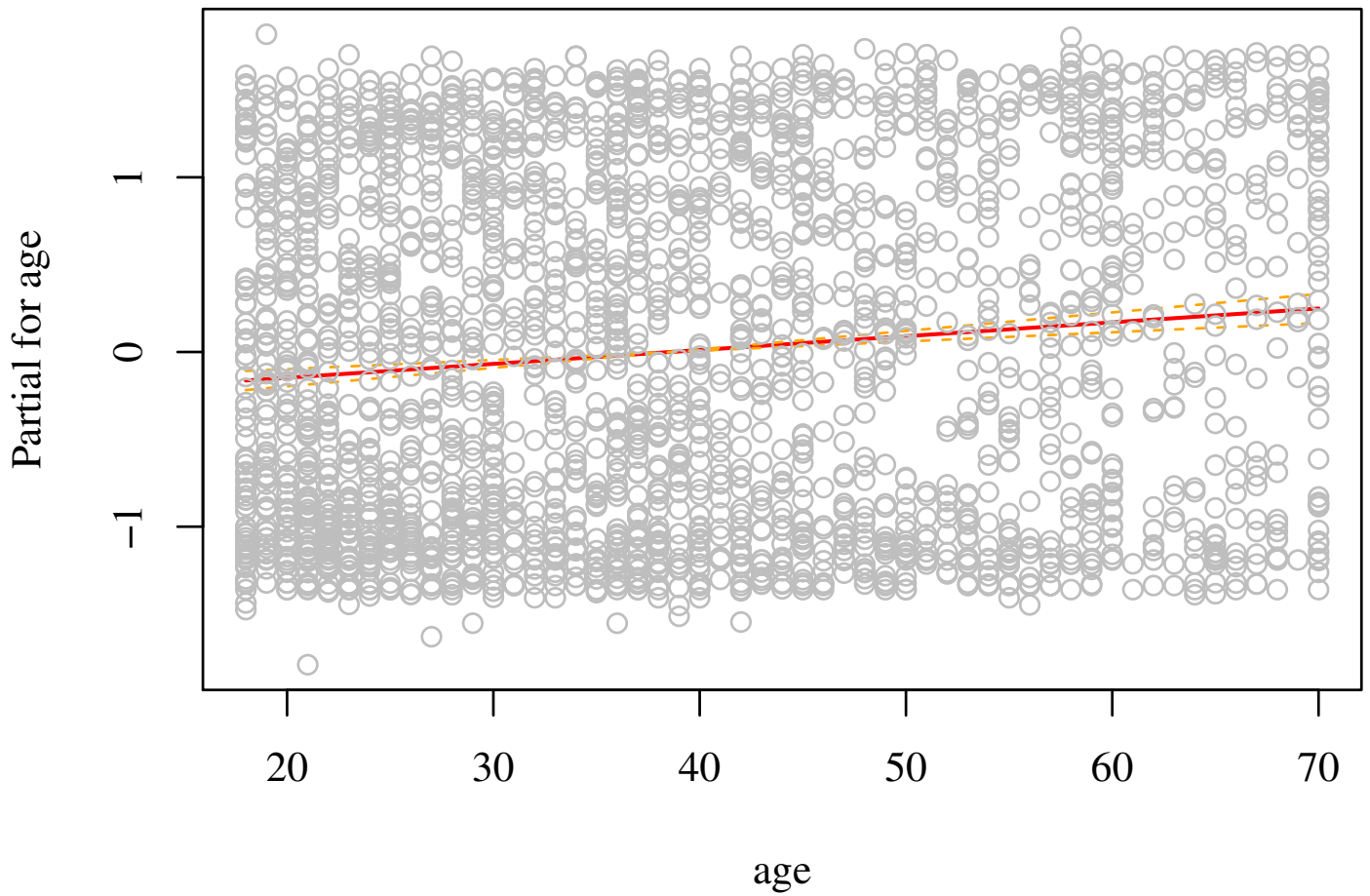
I almost always add se and partial.resid arguments

Richer display results

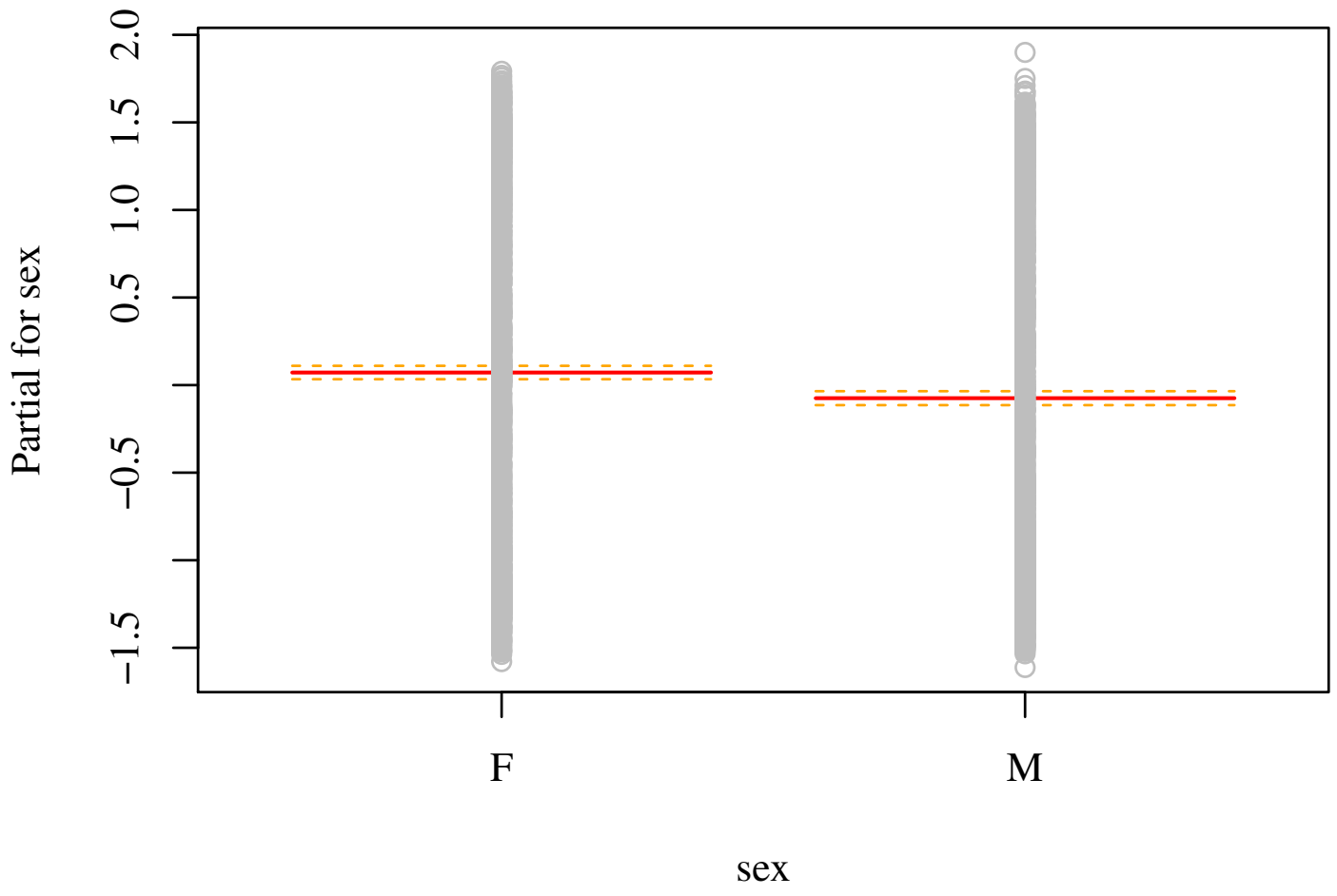
```
termplot(m2, se = TRUE, partial.resid = TRUE)
```



Look at a small model on the Chile data



Look at a small model on the Chile data



Run a big model

```
m3 <- lm (th.bush.kerry ~ V045117 + V043116 + V043210 + V043213 +
     V045145X + V041109A, data = mydta1)
```

- The interesting thing about this is that the model generates a huge set of coefficients, but
- termplot notices which variable corresponds to each set and makes a graceful plot

The Fitted Regression Model

	Bush-Kerry Thermometer Differential Estimate	(S.E.)
(Intercept)	-54.647*	(8.06)
Liberal	-10.788	(7.931)
Slight Lib.	2.375	(7.933)
Moderate	5.612	(7.819)
Slight Con.	10.141	(8.257)
Conservative	17.499*	(8.341)
Ext. Cons.	26.398*	(9.783)
Weak Dem.	24.605*	(4.032)
Indep. Lean Dem.	22.365*	(3.765)
Independent	40.605*	(5.165)
Indep. Lean Rep.	65.212*	(4.59)
Weak Rep.	67.239*	(4.515)
Str. Repub.	82.348*	(4.722)
No Gay Marr	7.911*	(2.615)
V043210Med	6.781	(5.84)
Econ. Same	17.701*	(2.821)
Econ. Better	25.083*	(3.278)
V045145X2. Very good	-7.623*	(2.528)
V045145X3. Somewhat good	-14.505*	(3.387)
V045145X4. Not very good	-14.672*	(6.141)
V045145X7. Don't feel anything VOL	-26.238*	(8.668)
Female	0.284	(2.19)
N	803	
RMSE	29.95	
R^2	0.712	

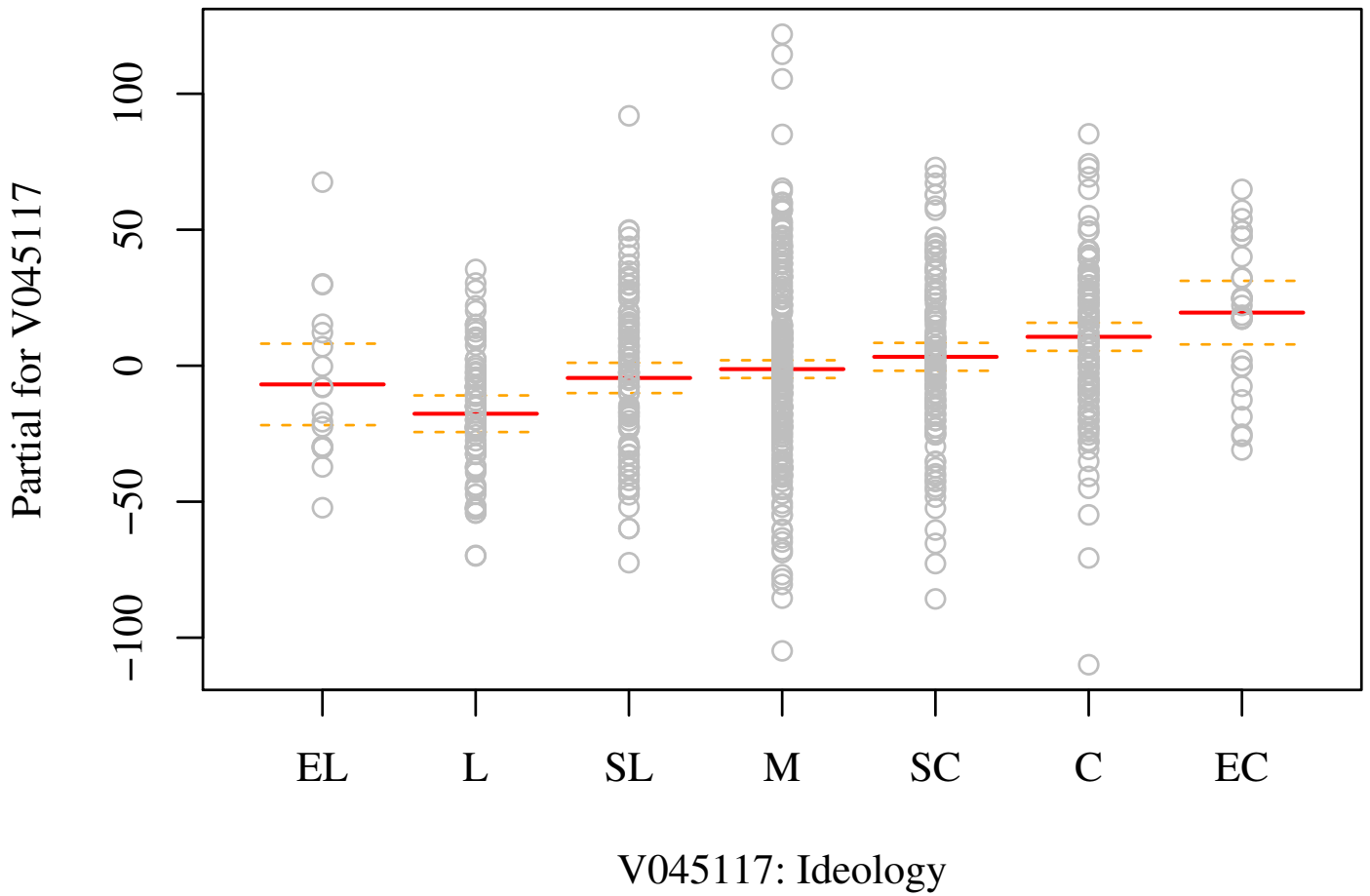
termplot will try to show these to you, one at a time

- You run

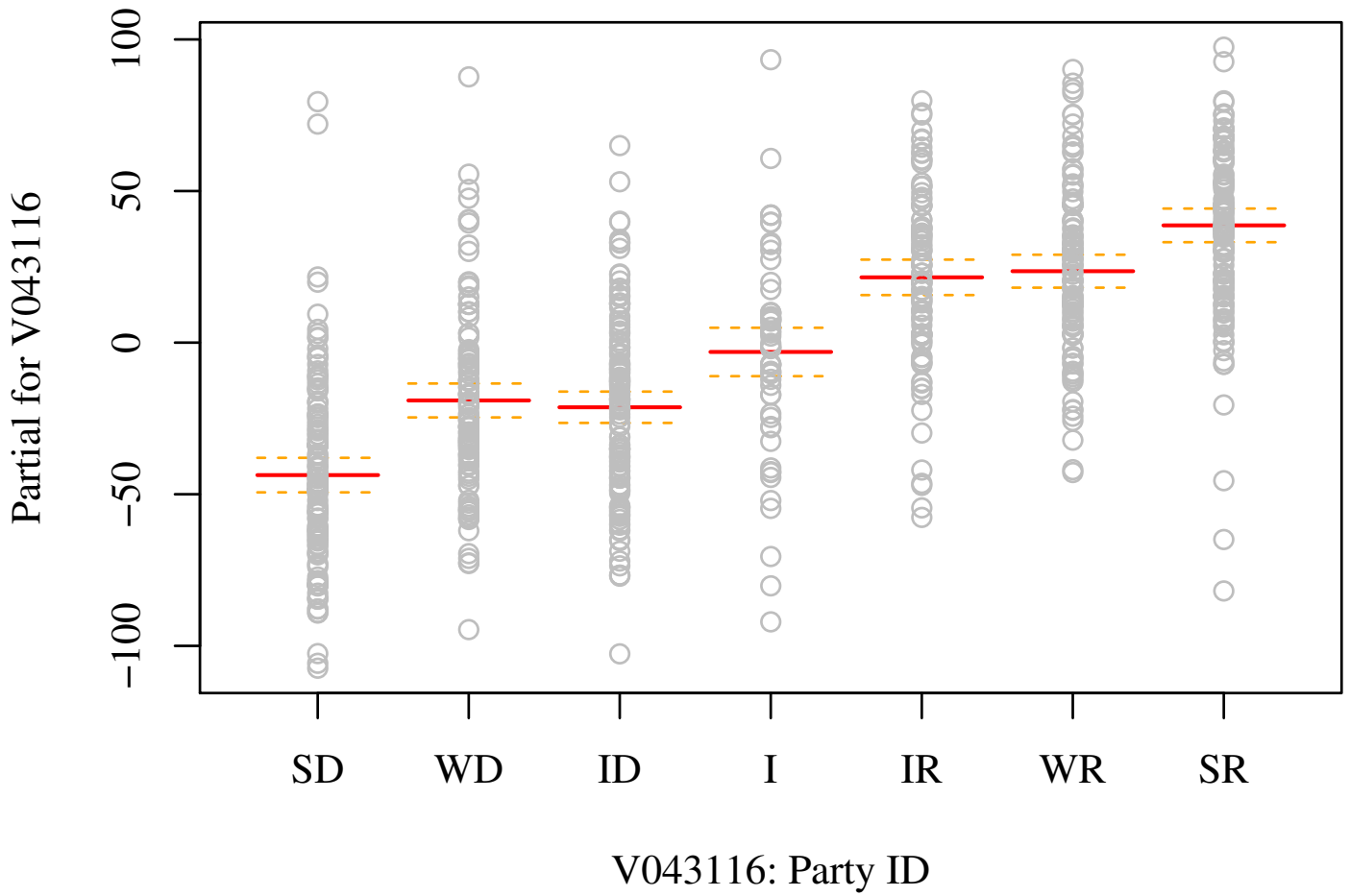
```
termplot(m3, se = TRUE, partial.resid = TRUE)
```

  to see all of them.
  I have to ask "by name" in order to put this presentation together
- se=T prints the 95% "confidence intervals" for the predicted values
- partial.resid=T plots the "partial residuals"

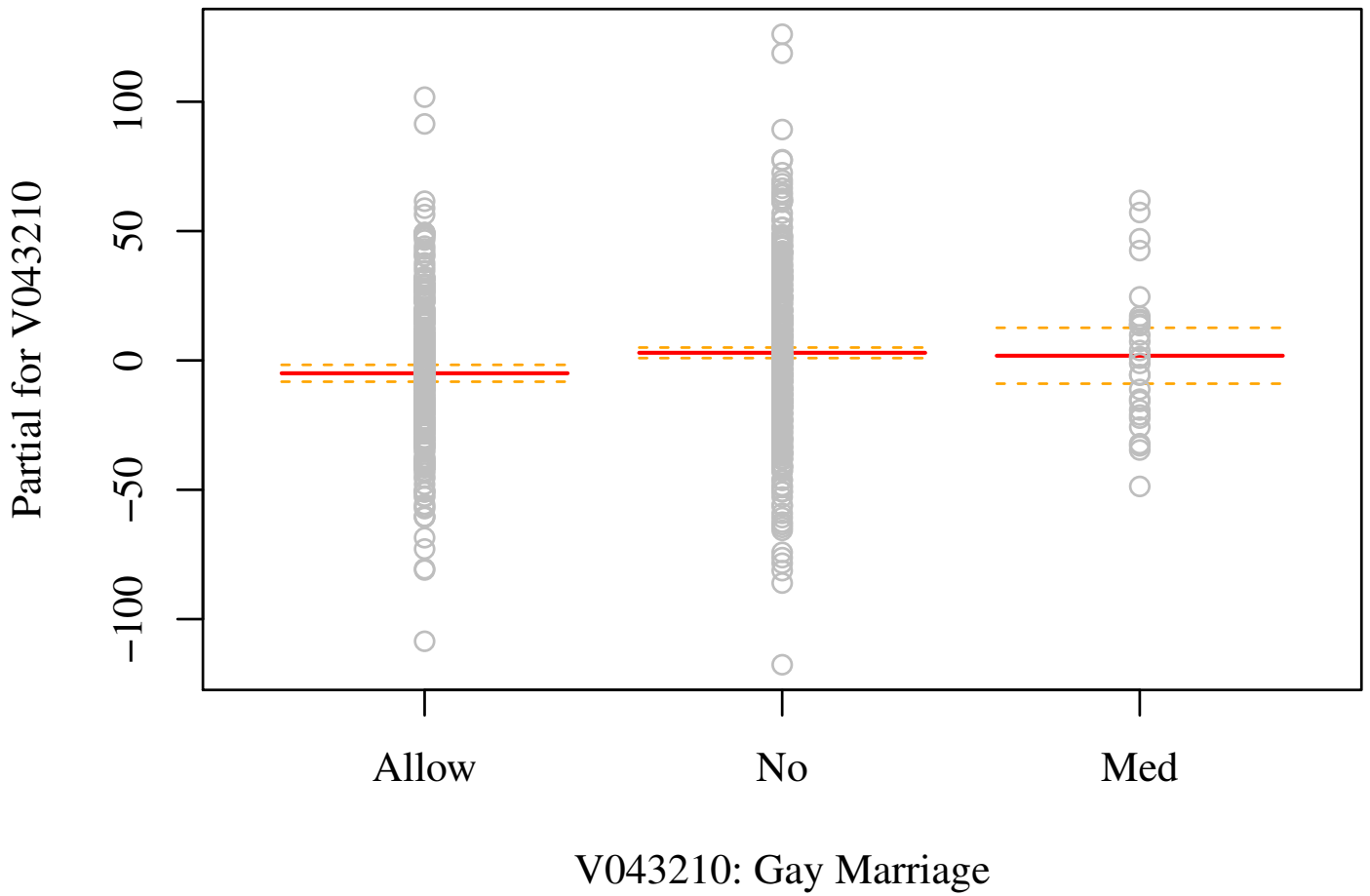
termplot: Ideology



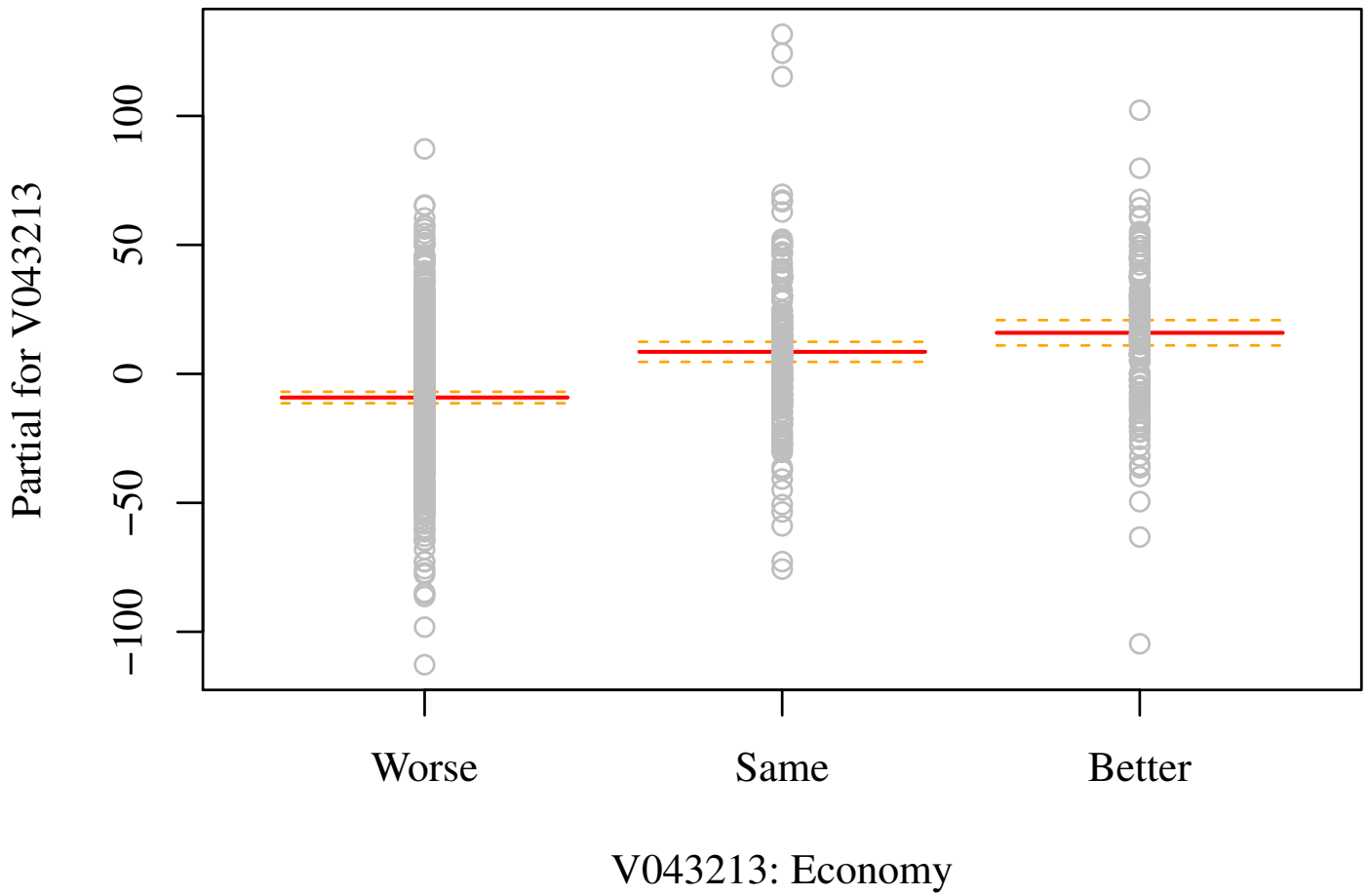
termplot: Party ID



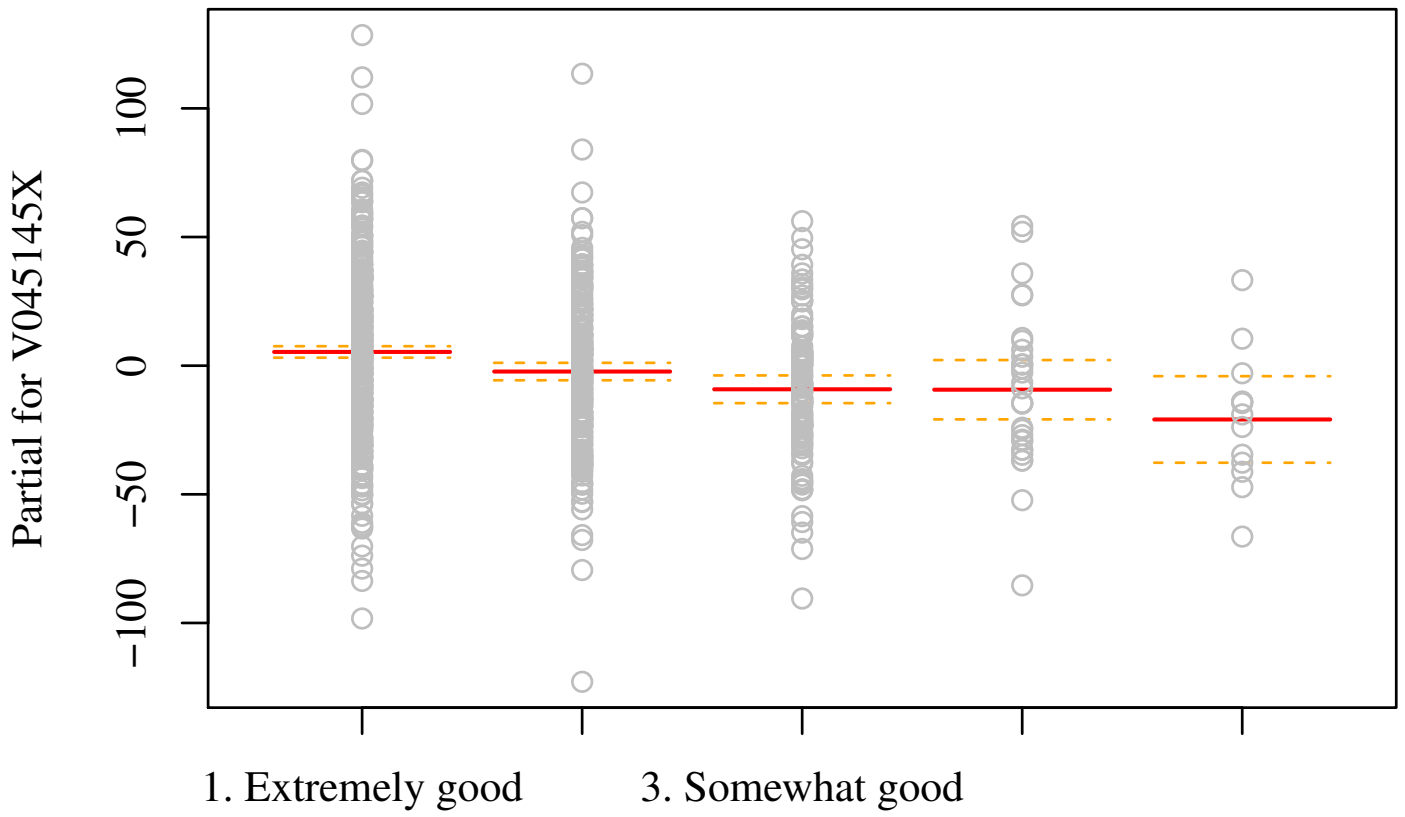
termplot: Gay Marriage



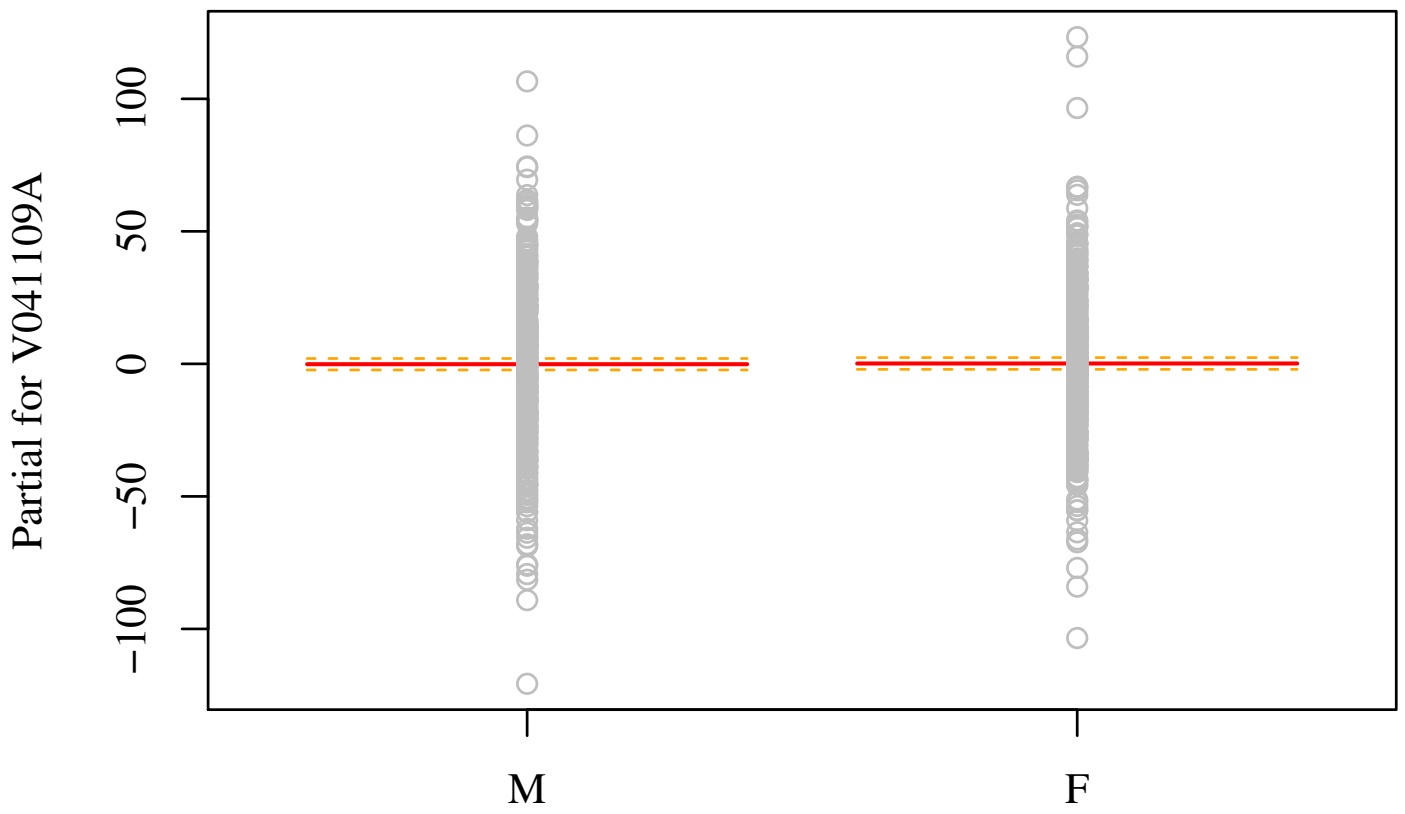
termplot: National Economy



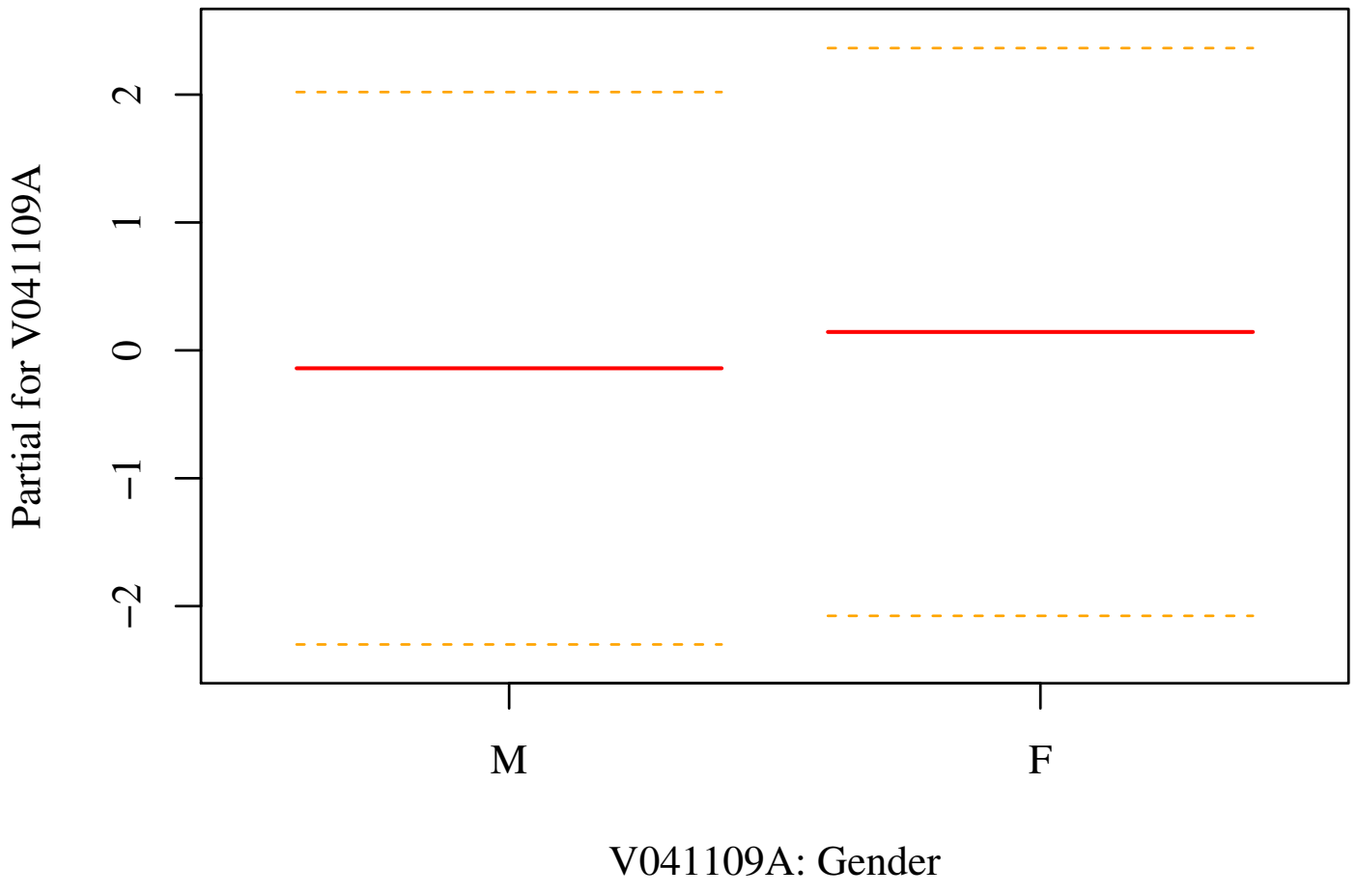
termplot: The Flag



termplot: Gender



termplot: Gender w/o partial residuals



Frustrating Detour: Why is the Vertical Axis called "partial"?

- The ?termplot manual is difficult to understand, the meaning of a "term predictions" and "partial residual" is not stated.
- So I went and read the code. (Then we are more sympathetic to the manual writer, because this is a complicated idea.)
- Summary of the idea: Aim is to plot y_i on $X1_i$, after "subtracting out" effects of $X2_i$, $X3_i$
- A "Partial Residual" is what is left unaccounted for by one variable when the predictive effects of all of the other input variables have been taken into account.
- If you are satisfied with that "story", skip the next few slides. Otherwise, be brave and keep trying.

Huh? What is the Partial Predicted Value?

- Consider model

```
mod <- lm(y ~ X1 + X2 + X3, data = dat)
```

- The ordinary "predicted value" is

$$\hat{y}_i = \hat{b}_0 + \hat{b}_1 X1_i + \hat{b}_2 X2_i + \hat{b}_3 X3_i$$

- The "mean prediction" is:

$$\hat{b}_0 + \hat{b}_1 \overline{X1} + \hat{b}_2 \overline{X2} + \hat{b}_3 \overline{X3}$$

- One old math professor trick is to add and subtract the same thing from one side of an equation and then rearrange so that some new insight flows out. Here, I will add and subtract the mean prediction from the predicted value.

Huh? What is the Partial Predicted Value? ...

- After re-grouping, we see the "full model" is the sum of the "mean prediction" and "term" predictions

$$\begin{aligned}\hat{y}_i = & \{\hat{b}_0 + \hat{b}_1\overline{X1} + \hat{b}_2\overline{X2} + \hat{b}_3\overline{X3}\} \\ & + \{\hat{b}_1(X1_i - \overline{X1}) + \hat{b}_1(X2_i - \overline{X2}) + \hat{b}_3(X3_i - \overline{X3})\} \end{aligned} \quad (1)$$

- In R, predict(mod, type="terms") returns the last "term predictions" as separate columns:

$$\overbrace{\hat{b}_1(X1_i - \overline{X1})}^{X1} \quad , \quad \overbrace{\hat{b}_2(X2_i - \overline{X2})}^{X2} \quad , \quad \overbrace{\hat{b}_3(X3_i - \overline{X3})}^{X3} \quad (2)$$

- These are the predicted values used by termplot in its internal calculations

And a partial residual is...

- begin with the "full model residual"

$$y_i - \hat{y}_i = y_i - \{\hat{b}_0 + \hat{b}_1 X1_i + \hat{b}_2 X2_i + \hat{b}_3 X3_i\} \quad (3)$$

- Get the *partial residual*$_i$ by replacing $X1_i$ with $\overline{X1}$:

$$y - \{\hat{b}_0 + \hat{b}_1 \overline{X1} + \hat{b}_2 X2_i + \hat{b}_3 X3_i\} \quad (4)$$

  - If we "do not allow variations in $X1$ to play a predictive role, what remains unexplained?"
- Another interpretation (See the source code in "residuals.lm"):
  - the partial residual = "full residual" + "term prediction"

$$\begin{aligned} \textit{partial residual}(X1_i) &= y_i - \{\hat{b}_0 + \hat{b}_1 X1_i + \hat{b}_2 X2_i + \hat{b}_3 X3_i\} \\ &\quad + \{\hat{b}_1 (X1_i - \overline{X1})\} \\ &= y_i - \{\hat{b}_0 + \hat{b}_1 \overline{X1} + \hat{b}_2 X2_i + \hat{b}_3 X3_i\} \end{aligned}$$

Reminder

- termplot is a wonderful tool, but it suffers from one crippling flaw: it does handle interaction terms.
- That lead to
  - many hours of teaching about predict and newdata in our intro regression class
  - the creation of the rockchalk package to automate the management of some common cases.

rockchalk functions

plotSlopes was the first effort there, and later it became apparent that a general approach to automatically create newdata objects would be a tremendous help (and allows predictOMatic to "just work").

newdata: scan regression model and construct "newdata" objects that can be used by predict methods
This is user friendly, flexible. The focal values of several predictors can be selected (by quantile, by standard deviations, a sequence that walks across the observed values).

predictOMatic: Calculates predicted values for various predictor values

plotCurves: handles nonlinear formulae and interactions among nonlinear terms.

newdata()

- newdata() tries to do the right thing, allowing easy customization.
- Consider the Chile data, with statusquo ~ income + age + sex

```
newdata(m2, predVals = c("income"))
```

```
  income      age sex
1    7.5 38.54872   F
2   15.0 38.54872   F
3   35.0 38.54872   F
```

```
newdata(m2, predVals = c("income"), n = 5)
```

```
  income      age sex
1    2.5 38.54872   F
2    7.5 38.54872   F
3   15.0 38.54872   F
4   35.0 38.54872   F
5  200.0 38.54872   F
```

newdata() ...

```
## income data not continuous, barely numeric
sort(unique(Chile$income))
```

```
[1]   2.5   7.5  15.0  35.0  75.0 125.0 200.0
```

```
newdata(m2, predVals = list(income = "table", "sex" = c("M","F")),
      n = 5)
```

```
   income sex      age
1    15.0   M 38.54872
2    35.0   M 38.54872
3     7.5   M 38.54872
4    75.0   M 38.54872
5     2.5   M 38.54872
6    15.0   F 38.54872
7    35.0   F 38.54872
8     7.5   F 38.54872
9    75.0   F 38.54872
10    2.5   F 38.54872
```

```
newdata(m2, predVals = list(income = "quantile", age = "std.dev."),
      n = 3)
```

newdata() ...

```
  income    age sex
1    7.5  23.79   F
2   15.0  23.79   F
3   35.0  23.79   F
4    7.5  38.55   F
5   15.0  38.55   F
6   35.0  38.55   F
7    7.5  53.31   F
8   15.0  53.31   F
9   35.0  53.31   F
```

predictOMatic shows marginal effects

If no particulars are requested, predictOMatic shows the marginal effects of each separate predictor, keeping the other values at their means (numeric variables) or modes (factor variables).

```
predictOMatic(m2)
```

```
$income
      income      age sex         fit
0%       2.5 38.54872   F 0.02609745
25%      7.5 38.54872   F 0.03191981
50%     15.0 38.54872   F 0.04065334
75%     35.0 38.54872   F 0.06394277
100%   200.0 38.54872   F 0.25608057

$age
      age   income sex         fit
0%     18 33.87586   F -0.10021358
25%    26 33.87586   F -0.03681408
50%    36 33.87586   F  0.04243529
75%    49 33.87586   F  0.14545948
100%   70 33.87586   F  0.31188316

$sex
```

predictOMatic shows marginal effects ...

```
        sex    income       age          fit
F (50%)   F 33.87586 38.54872  0.06263375
M (50%)   M 33.87586 38.54872 -0.08383037

attr(,"pnames")
[1] "income" "age"    "sex"
```

```
predictOMatic(m3)
```

```
$V045117
         V045117 V043116 V043210 V043213          V045145X V041109A
                    fit
M (30%)        M      ID      No   Worse 1. Extremely good        F
     -18.476144
C (20%)        C      ID      No   Worse 1. Extremely good        F
      -6.589442
SC (20%)      SC      ID      No   Worse 1. Extremely good        F
     -13.947726
SL (10%)      SL      ID      No   Worse 1. Extremely good        F
     -21.712903
L (10%)        L      ID      No   Worse 1. Extremely good        F
     -34.875946
```

predictOMatic shows marginal effects ...

```
$V043116
         V043116 V045117 V043210 V043213          V045145X V041109A
                    fit
ID (20%)      ID       M      No   Worse 1. Extremely good        F
     -18.47614
SD (20%)      SD       M      No   Worse 1. Extremely good        F
     -40.84086
SR (20%)      SR       M      No   Worse 1. Extremely good        F
      41.50739
WD (10%)      WD       M      No   Worse 1. Extremely good        F
     -16.23570
WR (10%)      WR       M      No   Worse 1. Extremely good        F
      26.39827

$V043210
            V043210 V045117 V043116 V043213          V045145X
               V041109A       fit
No (60%)          No       M      ID   Worse 1. Extremely good
            F -18.47614
Allow (30%)    Allow       M      ID   Worse 1. Extremely good
            F -26.38670
Med (0%)         Med       M      ID   Worse 1. Extremely good
            F -19.60610

$V043213
```

predictOMatic shows marginal effects ...

```
                V043213 V045117 V043116 V043210          V045145X
                    V041109A          fit
Worse (60%)      Worse        M       ID       No 1. Extremely good
              F -18.4761440
Same (30%)        Same        M       ID       No 1. Extremely good
              F  -0.7752059
Better (20%)    Better        M       ID       No 1. Extremely good
              F   6.6065102

$V045145X
                                                          V045145X
                                                              V045117
                                                              V043116
                                                              V043210
                                                              V043213
                                                              V041109A
                                                                    fit
1. Extremely good (50%)                                 1. Extremely good
           M       ID       No    Worse          F -18.47614
2. Very good (30%)                                        2. Very good
           M       ID       No    Worse          F -26.09885
3. Somewhat good (20%)                                  3. Somewhat good
           M       ID       No    Worse          F -32.98141
4. Not very good (0%)                                   4. Not very good
           M       ID       No    Worse          F -33.14812
```

predictOMatic shows marginal effects ...

```
7. Don't feel anything {VOL} (0%) 7. Don't feel anything {VOL}
          M      ID      No   Worse         F -44.71372

$V041109A
        V041109A V045117 V043116 V043210 V043213          V045145X
                    fit
F (50%)        F       M      ID      No   Worse 1. Extremely good
     -18.47614
M (50%)        M       M      ID      No   Worse 1. Extremely good
     -18.76006

attr(,"pnames")
[1] "V045117"  "V043116"  "V043210"  "V043213"  "V045145X" "V041109A
    "
```

plotSlopes uses that framework

- We want to estimate an interactive model, by changing this:

```
th.bush.kerry ~ V043250 + V041109A
```

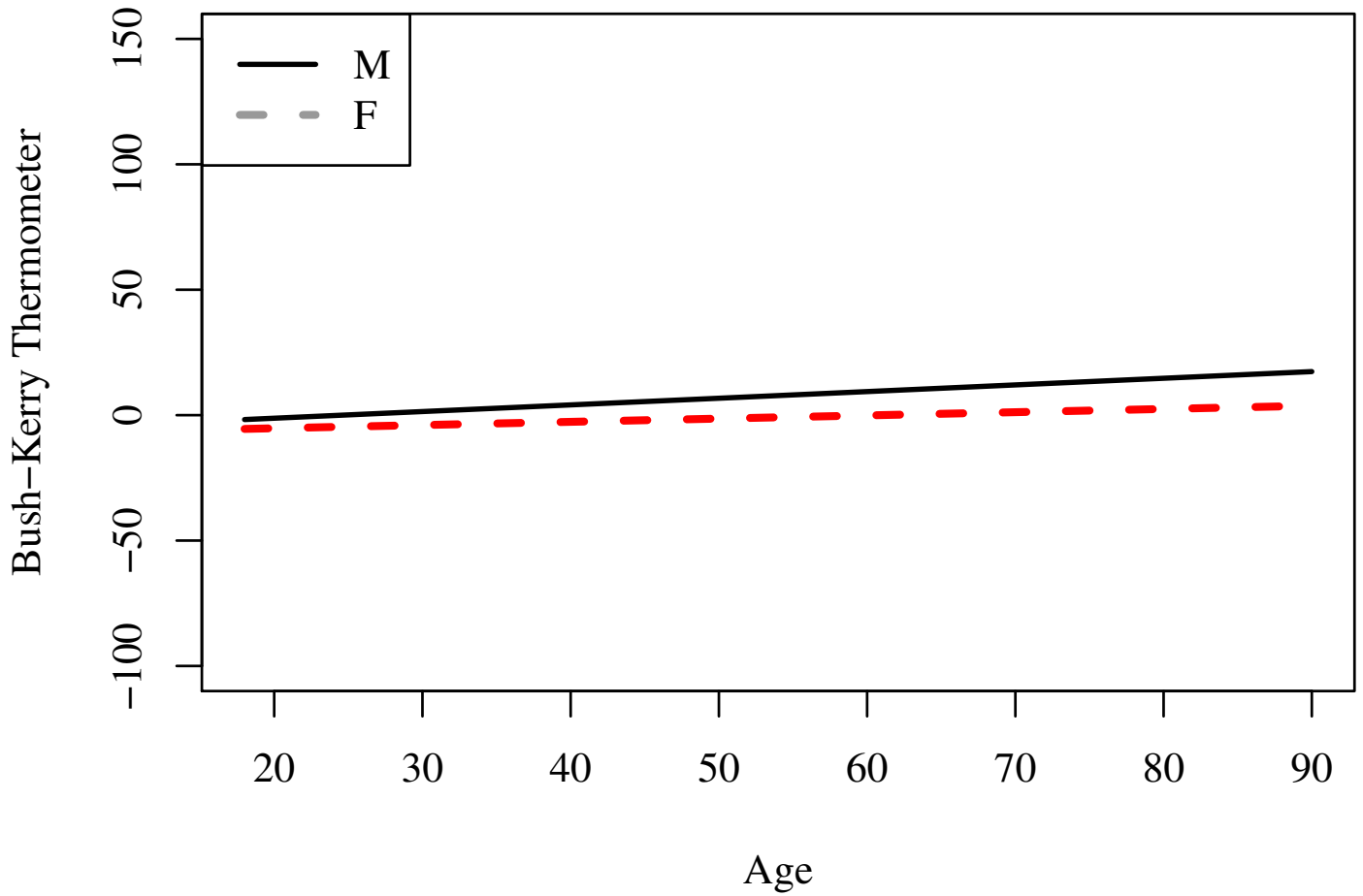
  $V041109A$ is respondent gender, coded "M" and "F". $V043250$ is age.

- To this:

```
th.bush.kerry ~ V043250 * V041109A
```

- We can plot an age, thermometer line for each "value" of gender
  - one line for "M"
  - one line for "F".

plotSlopes uses that information



How Did I Make That Plot?

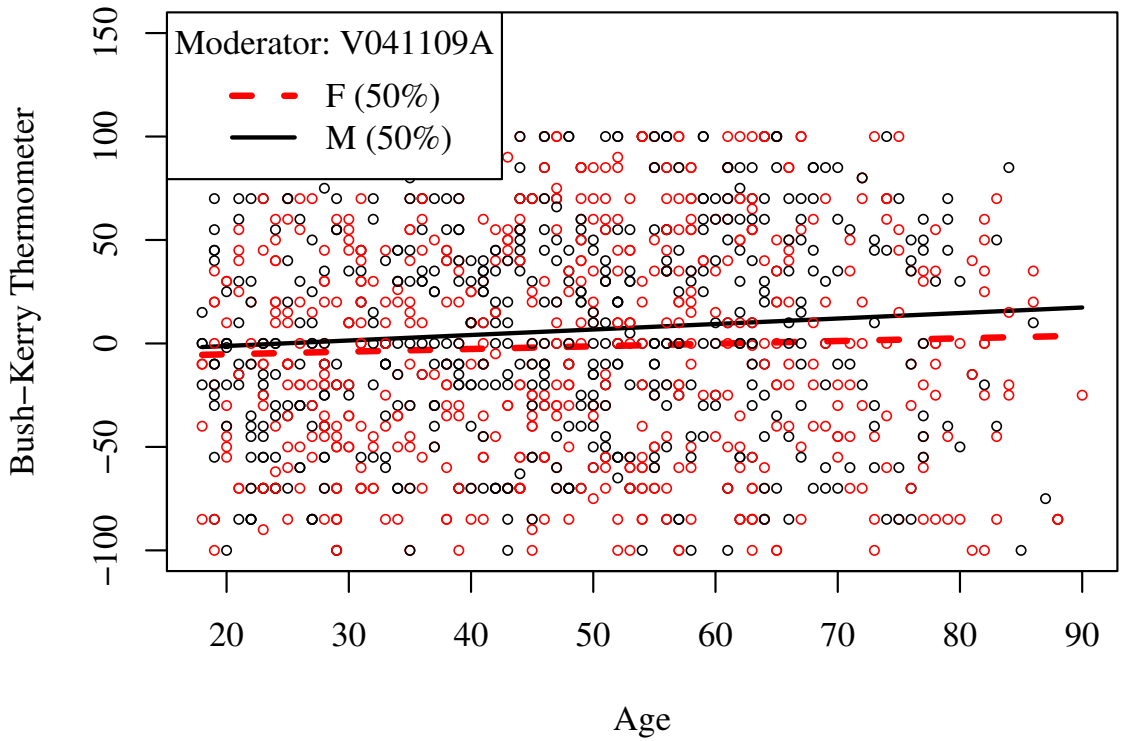
- Students thought it was very tedious to calculate predictions for various moderators
- TaDa! plotSlopes rockchalk simplifies this work.

```
plotSlopes(m2, plotx = "numericPredictor", modx = "
    aModeratorVariable")
```

- It figures out which values of aModeratorVariable should be considered, calculates predicted values for subgroups and draws them.
- It returns an object that includes the newdata object.

```
m6 <- lm(th.bush.kerry ~ V043250 * V041109A, data=mydta1)
plotSlopes(m6, plotx = "V043250", modx = "V041109A", plotPoints =
    FALSE, plotLegend = FALSE,  xlab = "Age", ylab = "Bush-Kerry
    Thermometer", ylim = c(-100,150), col = c("black","gray60"),
    llwd=c(2,3))
legend("topleft", legend = levels(mydta1$V041109A), col=c("black","
    gray60"), lty=c(1,2), lwd=c(2,3))
```

Sometimes We Want To See The Points



Notice the non-beautified "automatic" legend

Write that out completely, without plotSlopes

```
m6 <- lm(th.bush.kerry~V043250 + V041109A, data=mydta1)
mycols <- c(gray(.7), rgb(0.7, 0.1, 0.1), "black", "red")
V043250r <- range(mydta1$V043250)
newdf1 <- expand.grid(V043250 = V043250r,  V041109A = levels(mydta1
    $V041109A))
newdf1$fit <- predict(m6, newdata = newdf1)
plot(th.bush.kerry ~ V043250, type = "n", data = mydta1, xlab = "
    V043250: Age (years)", ylab = "Bush-Kerry Thermometer
    Differential")
points(th.bush.kerry ~ V043250, data = mydta1,  lwd = 0.6, col =
    ifelse(V041109A %in% c("M"), mycols[1], mycols[2]))
by(newdf1, newdf1$V041109A, function(x) lines(fit~V043250, data = x
    , col = mycols[2+as.numeric(x$V041109A)], lty = as.numeric(x$
    V041109A), lwd = 3 ))
```

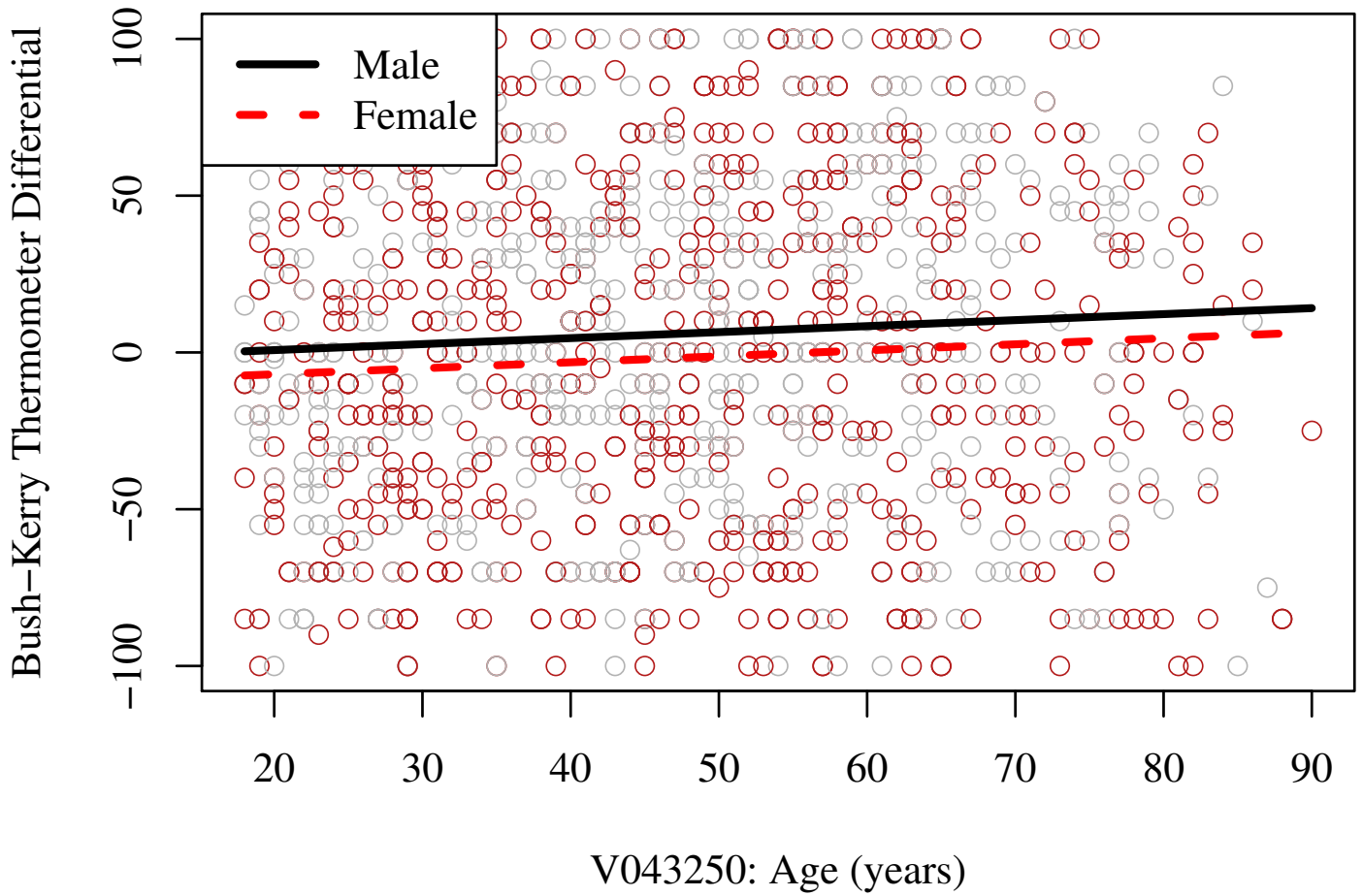
```
newdf1$V041109A: M
NULL
```

```
newdf1$V041109A: F
NULL
```

Write that out completely, without plotSlopes ...

```
legend("topleft", legend = c("Male", "Female"), col = mycols[3:4],
       lty = c(1,2), bg = "white", lwd = 3)
```

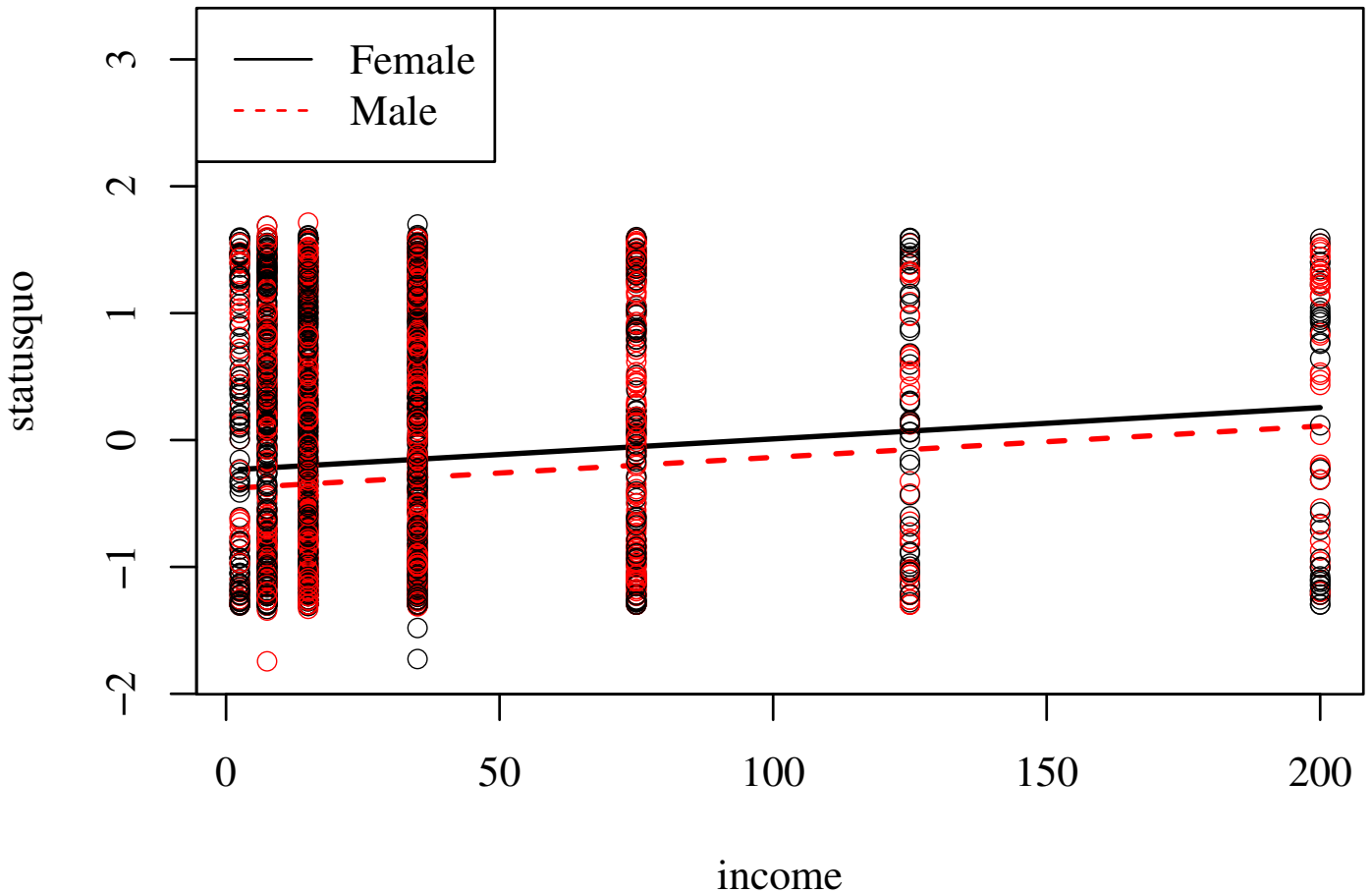
Plot Predicted values by Gender



Chile data: statusquo ~ income * region + sex

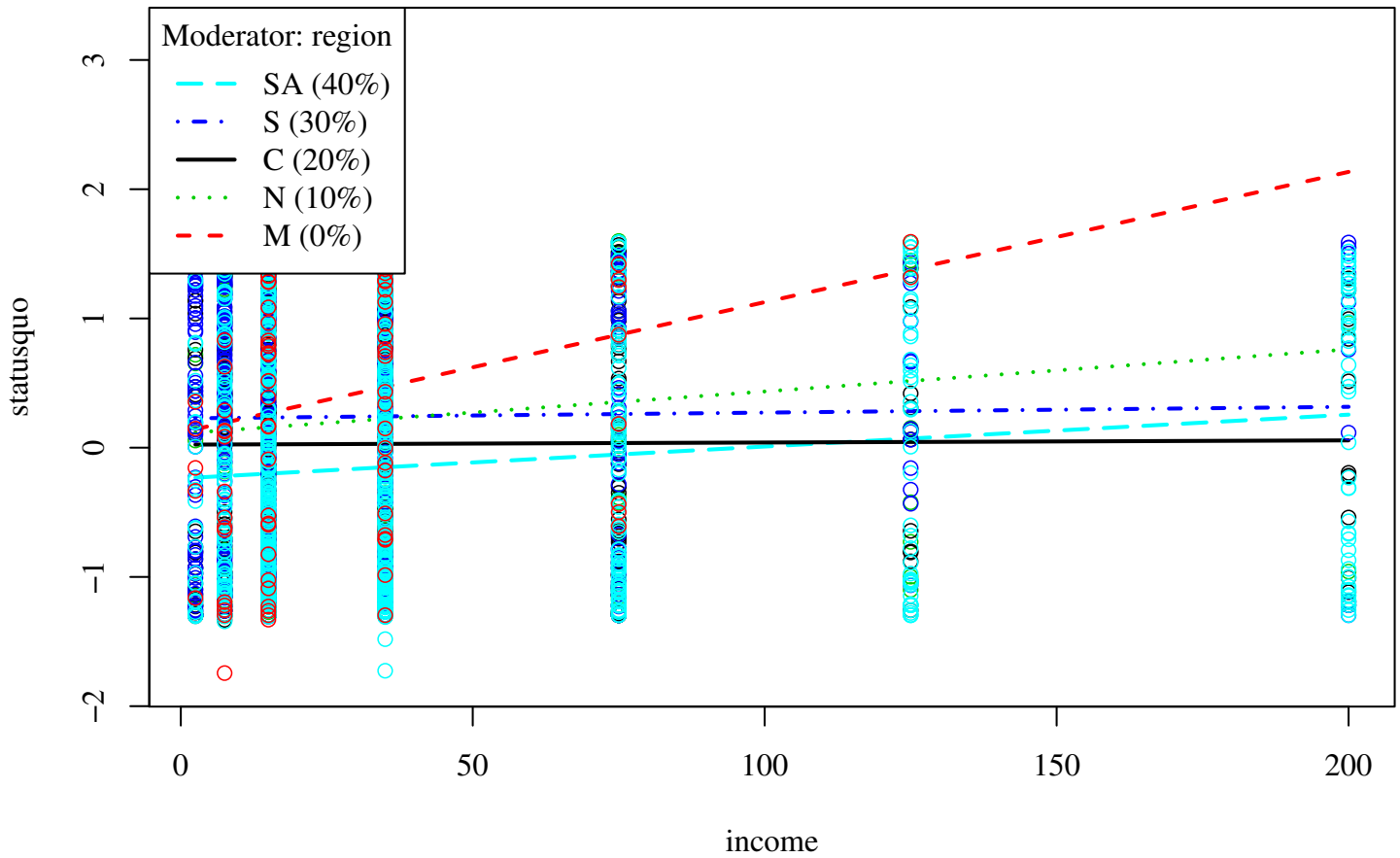
```
m6 <- lm(statusquo ~ income * region + sex, data = Chile)
plotSlopes(m6, plotx = "income", modx = "sex", plotLegend = FALSE,
     ylim = magRange(Chile$statusquo, c(1,1.3)), llwd = c(2,2), cex
     = 1.0, lwd = 0.5)
legend("topleft", c("Female", "Male"), col = c(1, 2), lty = 1:2, bg
     = "white")
```

Gender, Income, and Chilean Voter Attitudes



```
m6ps <- plotSlopes(m6, plotx = "income", modx = "region", ylim =
    magRange(Chile$statusquo, c(1, 1.3)), llwd = c(2, 2), cex = 1.0
    )
m6ps$newdata
```

	region	income	sex	fit	lwr	upr
1	SA	2.5	F	−0.23158507	−0.32180224	−0.1413679
2	S	2.5	F	0.22786977	0.12974686	0.3259927
3	C	2.5	F	0.02405915	−0.08384606	0.1319644
4	N	2.5	F	0.11756374	−0.03728760	0.2724151
5	M	2.5	F	0.14317264	−0.14128334	0.4276286
6	SA	200.0	F	0.25558282	0.03236837	0.4787973
7	S	200.0	F	0.31625024	−0.07781512	0.7103156
8	C	200.0	F	0.05668369	−0.32438301	0.4377504
9	N	200.0	F	0.76059507	0.12329258	1.3978975
10	M	200.0	F	2.13398804	0.67646454	3.5915115


Moderator: region
SA (40%)
S (30%)
C (20%)
N (10%)
M (0%)
statusquo
income
3
2
1
0
−1
−2
0
50
100
150
200

effects Package

- There are several other R packages that help us to explore predicted values.
- One of the first packages for this purpose was effects, by John Fox, Sanford Weisberg, and others.
- rockchalk is doing same/similar set of calculations, but user interface of rockchalk is (perhaps, just IMHO) more flexible & pleasant.
- plots from effects are not so nice as plots from termplot.
- Other packages that have a similar orientation are rms and Zelig

effects Package

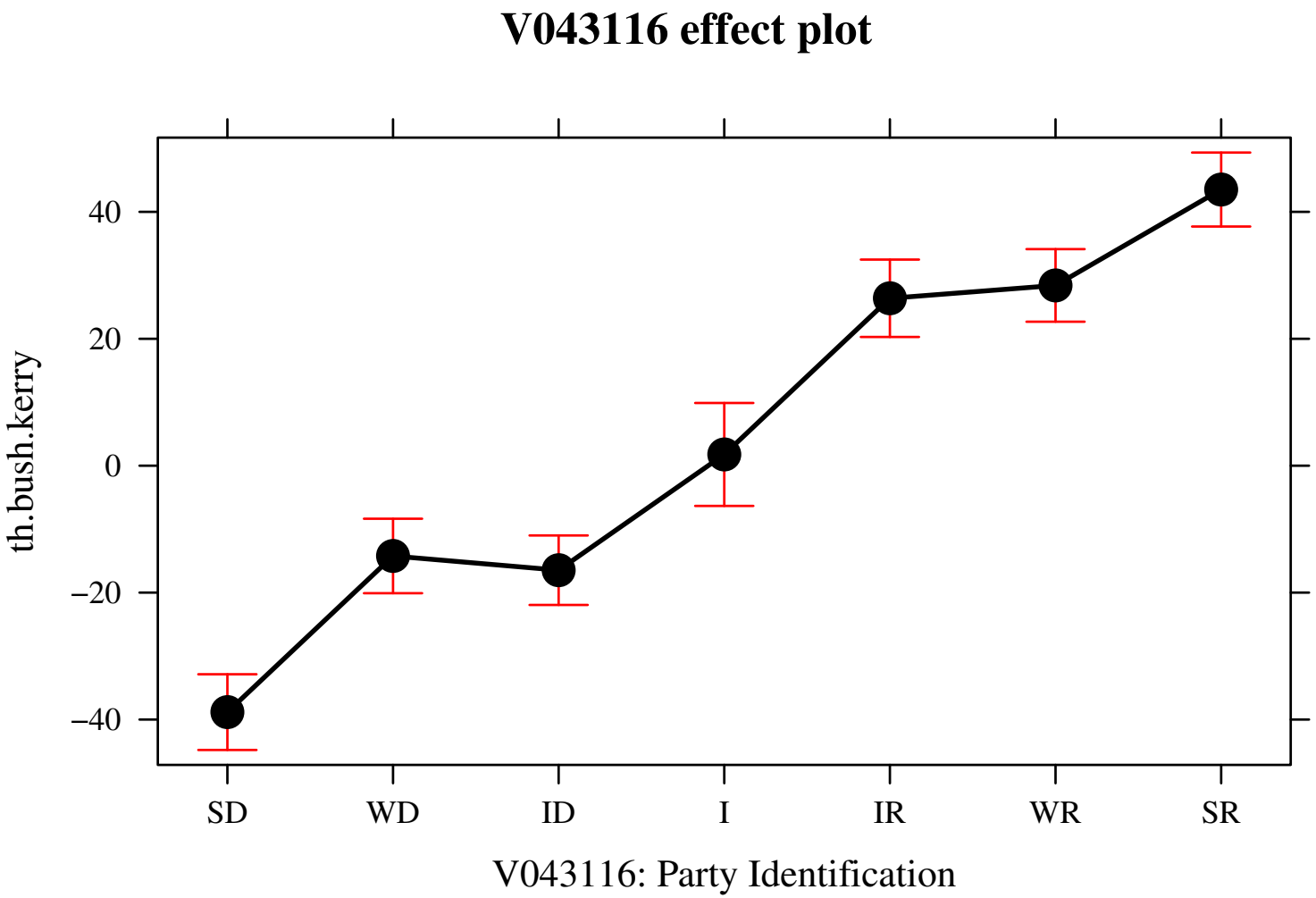
```
library(effects)
mod1.ae <- allEffects(mod1)
plot(mod1.ae, "V041109A")
```

Effects Plot: Gender

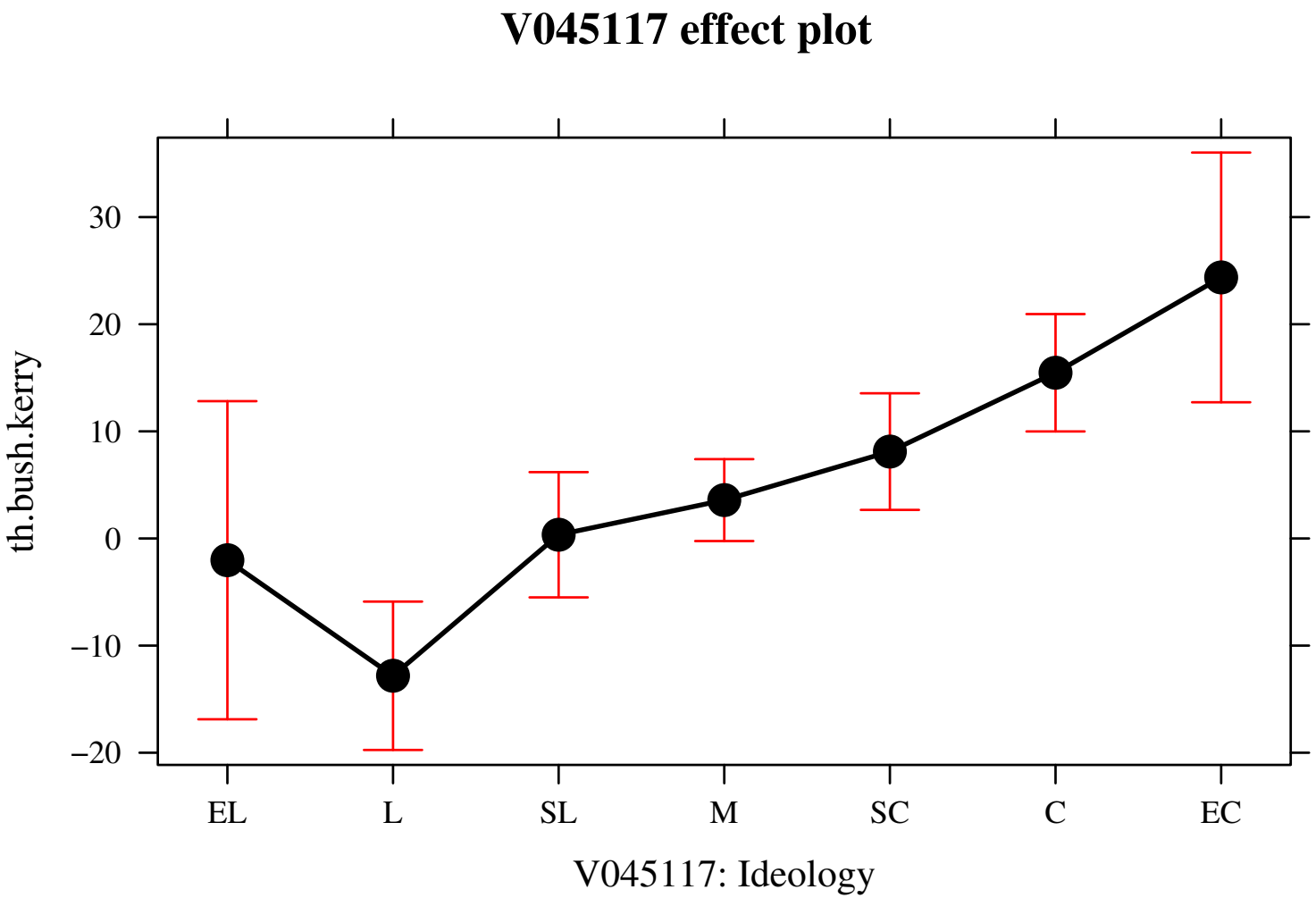
effects Package

```
plot(mod1.ae, "V043116", xlab="V043116: Party Identification")
```

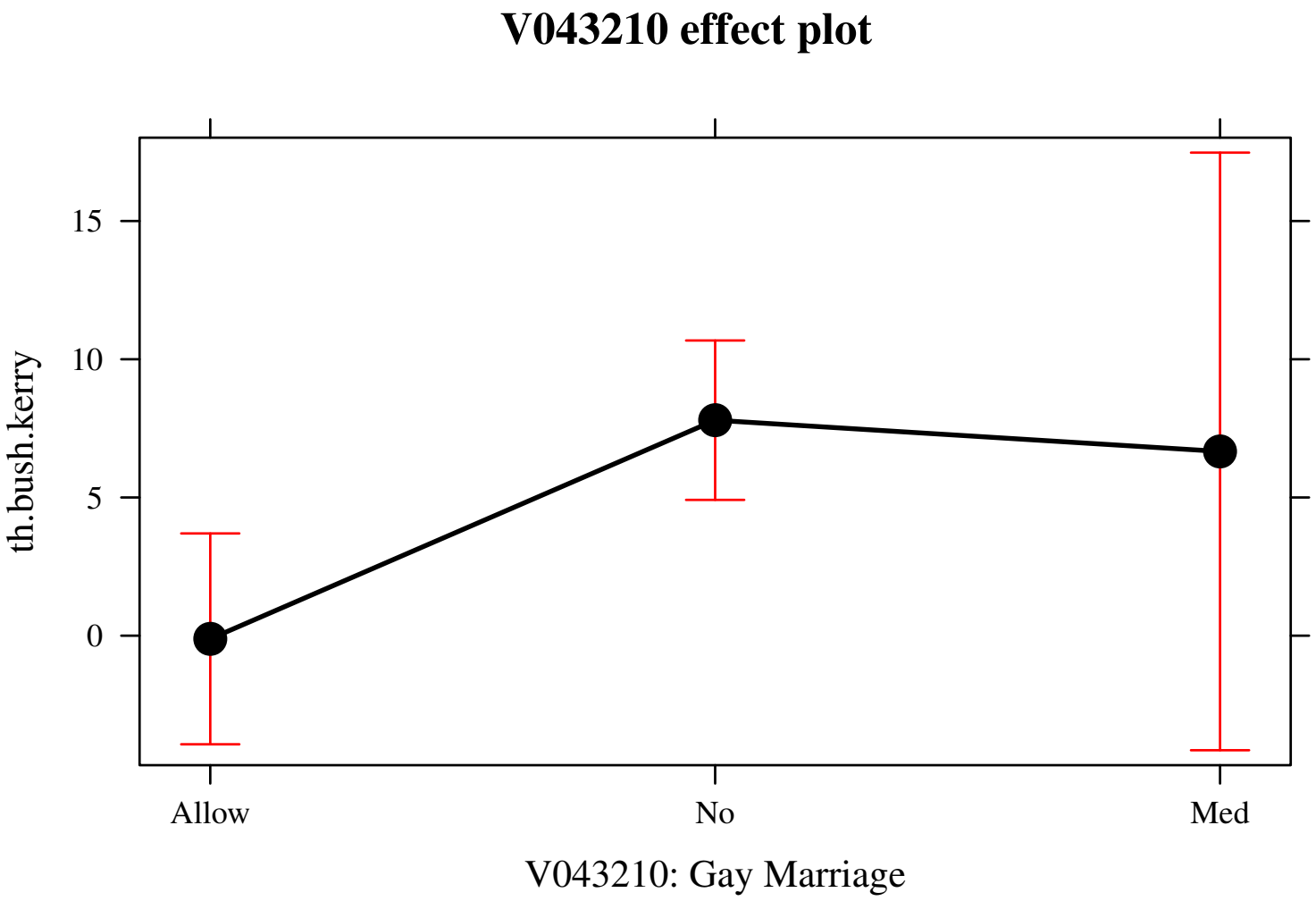
Effects Plot: Party Identification



Effects Plot: Ideology



Effects Plot: Gay Marriage



R Emphasizes Factors

- Importing Data: Any non-numeric score -> "factor" treatment
- Age-old question: Is a 7 point scale numeric or merely ordinal?
- Plots & regressions seem tidier if we use numeric scorings
- Possible to formally test the numeric versus the factor scorings

Create Numeric Predictors

```
mydta1$V045117n <- as.numeric(mydta1$V045117)
mydta1$V043116n <- as.numeric(mydta1$V043116)
mydta1$V043210n <- as.numeric(mydta1$V043210)
mydta1$V043213n <- as.numeric(mydta1$V043213)
mydta1$V045145Xn <- as.numeric(mydta1$V045145X)
m3n <- lm (th.bush.kerry ~ V045117n + V043116n + V043210n +
     V043213n + V045145Xn + V041109A, data=mydta1)
```

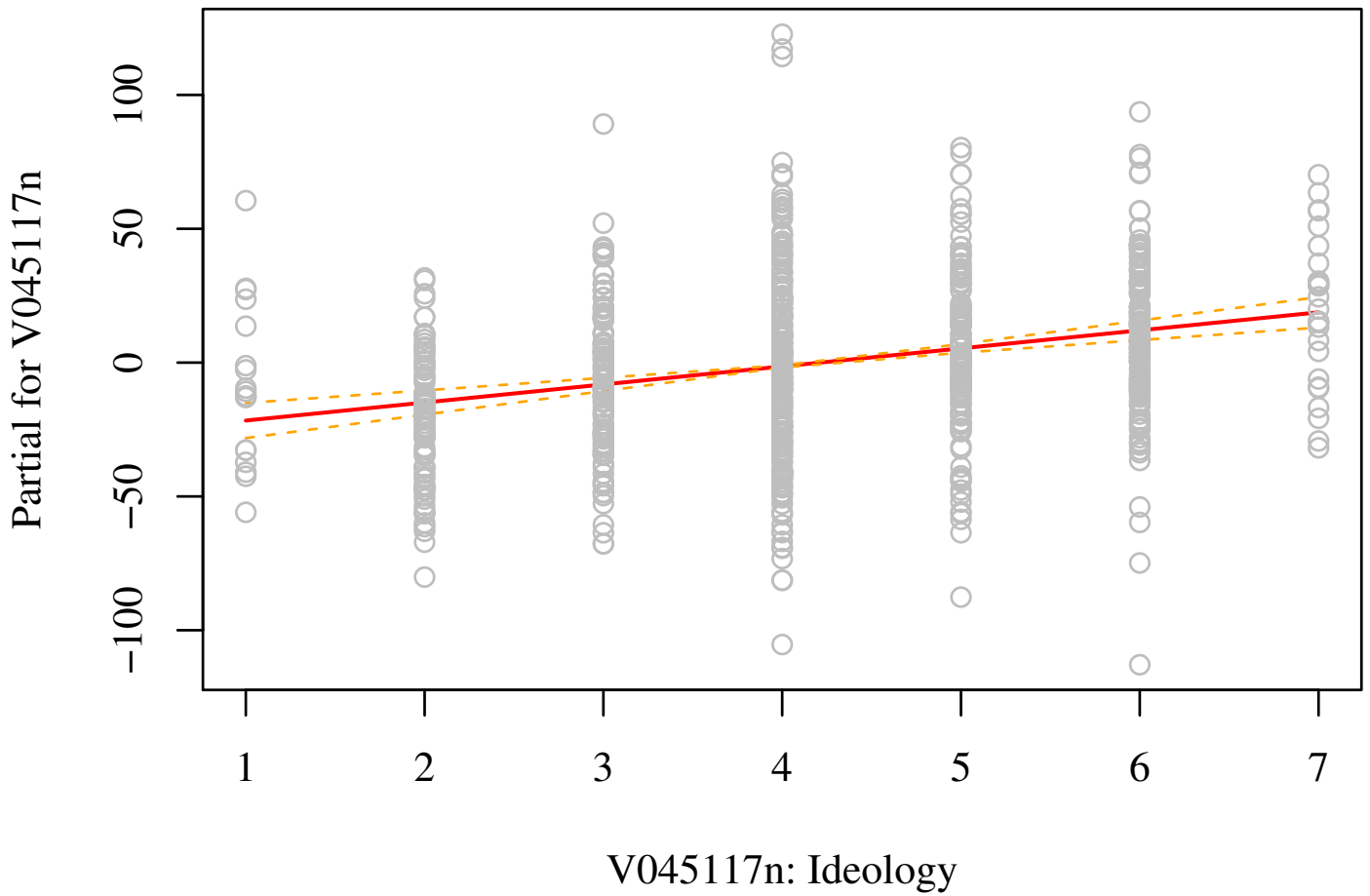
```
outreg(m3n, tight=FALSE, modelLabels=c("Numeric Predictors"),
     varLabels=list(V045117n="Ideology", V043116n="Party ID",
     V043210n="AntiGay", V043213n="Economy", V045145Xn="Flag Love",
     V041109AF="Female"))
```

Beautiful Outreg Output

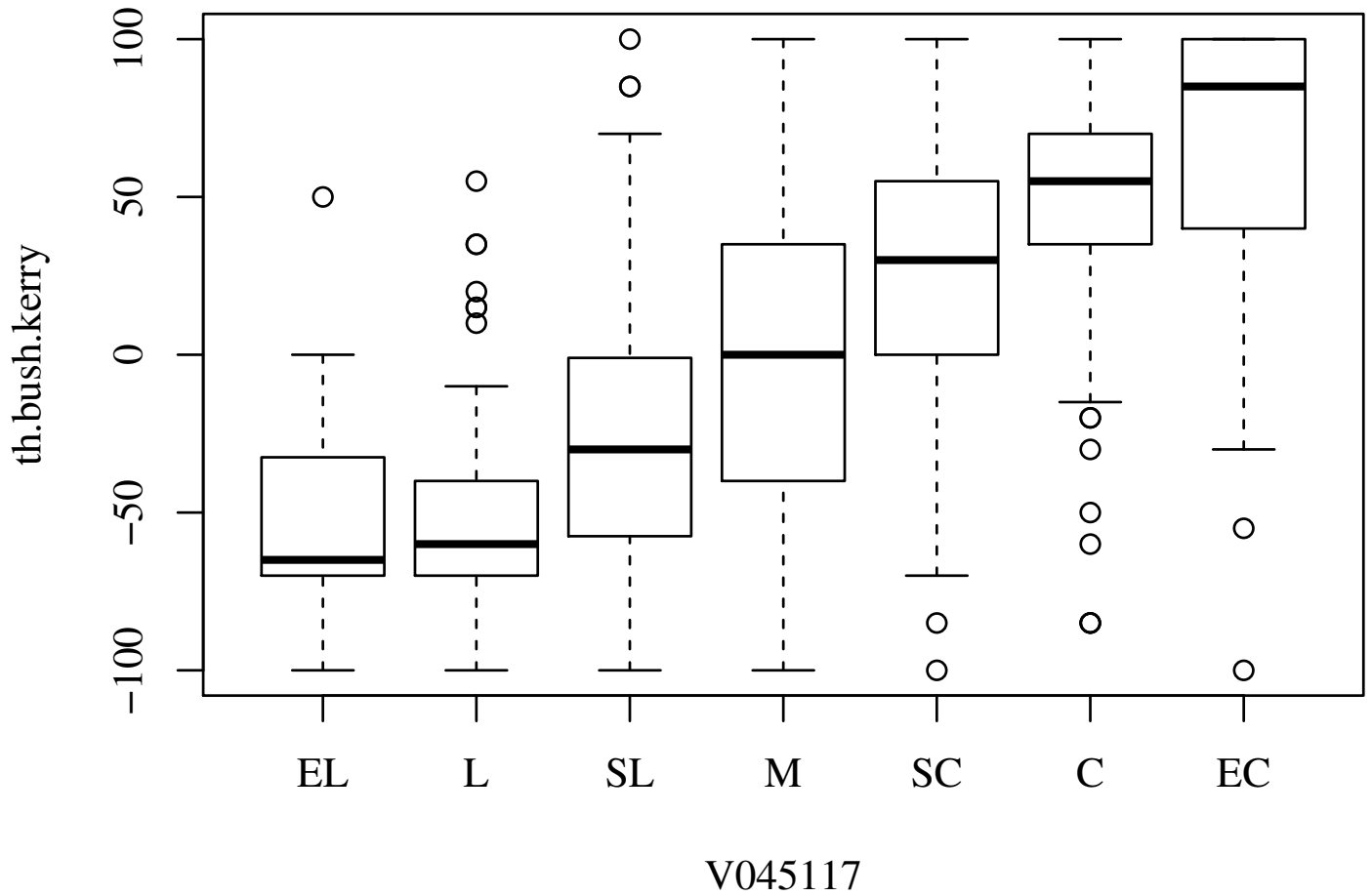
	Numeric Predictors	
	Estimate	(S.E.)
(Intercept)	-99.868*	(5.825)
Ideology	6.739*	(1.021)
Party ID	13.161*	(0.695)
AntiGay	7.207*	(2.201)
Economy	13.721*	(1.605)
Flag Love	-6.665*	(1.249)
Female	0.437	(2.169)
N	803	
RMSE	30.469	
R^2	0.696	
adj R^2	0.694	

$*p \leq 0.05$

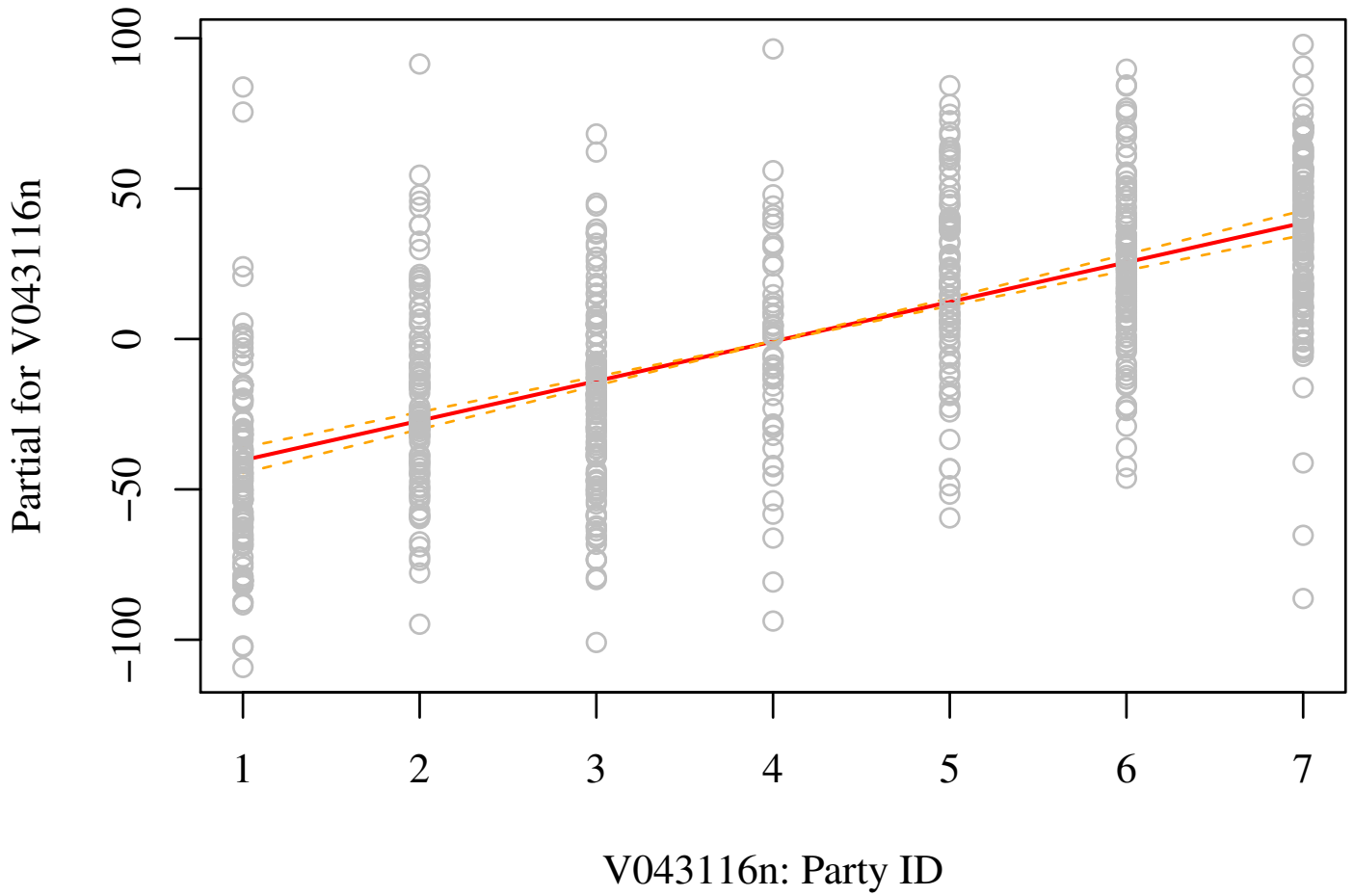
termplot: Ideology



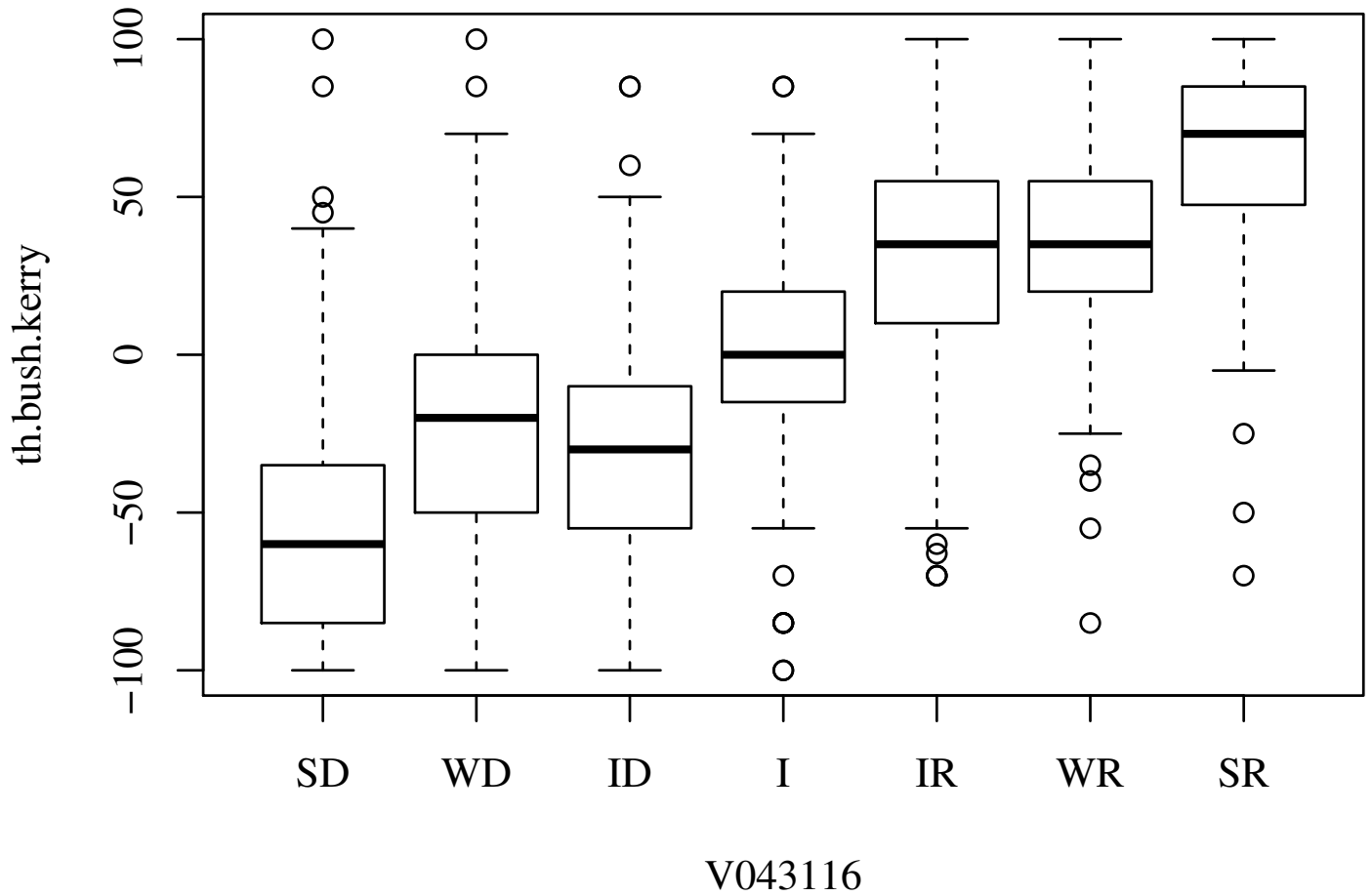
Compare to ordinary R plot(bush-kerry,ideology)



termplot: Party ID



Compare to ordinary R plot(bush-kerry, party))



Should you use the Numeric Model?

- My humble opinion: not without checking first
- Look at "termplot" output to see if the relationship really is linear
- Conduct hypothesis tests

An F test will work (these are nested models)

```
anova(m3,m3n, test="F")
```

```
Analysis of Variance Table

Model 1: th.bush.kerry ~ V045117 + V043116 + V043210 + V043213 +
    V045145X +
    V041109A
Model 2: th.bush.kerry ~ V045117n + V043116n + V043210n + V043213n +
    V045145Xn +
    V041109A
  Res.Df    RSS  Df Sum of Sq      F    Pr(>F)
1    781 700542
2    796 738959 -15    -38417 2.8553 0.0002215 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What does that Significance Test Mean?

- Something's wrong, at last one variable should not be treaded as a linear effect.
- Difficult to say with confidence what to do next.
- I would look at individual variables.

Party, for example

```
mod2 <- lm (th.bush.kerry ~ V045117n + V043116 + V043210n +
     V043213n + V045145Xn + V041109A, data=mydta1)
mod2n <- lm (th.bush.kerry ~ V045117n + V043116n + V043210n +
     V043213n + V045145Xn + V041109A, data=mydta1)
```

```
outreg(list(mod2, mod2n), tight=FALSE, modelLabels=c("Party Factor"
     ,"Numeric Predictors"), varLabels=list(V045117n="Ideology",
     V043116="Party ID", V043210n="AntiGay", V043213n="Economy",
     V045145Xn="Flag Love", V041109AF="Female"))
```

Beautiful Outreg Output

	Party Factor		Numeric Predictors	
	Estimate	(S.E.)	Estimate	(S.E.)
(Intercept)	-87.181*	(6.154)	-99.868*	(5.825)
Ideology	6.067*	(1.023)	6.739*	(1.021)
V043116WD	25.878*	(3.986)	.	
V043116ID	23.6*	(3.716)	.	
V043116I	41.993*	(5.017)	.	
V043116IR	66.697*	(4.505)	.	
V043116WR	68.496*	(4.393)	.	
V043116SR	83.971*	(4.654)	.	
AntiGay	6.611*	(2.178)	7.207*	(2.201)
Economy	13.211*	(1.591)	13.721*	(1.605)
Flag Love	-6.376*	(1.238)	-6.665*	(1.249)
Female	0.666	(2.17)	0.437	(2.169)
V043116n	.		13.161*	(0.695)
N	803		803	
RMSE	30.043		30.469	
R^2	0.707		0.696	
adj R^2	0.703		0.694	

$*p \leq 0.05$

An F test will work

```
anova(mod2, mod2n, test="F")
```

```
Analysis of Variance Table

Model 1: th.bush.kerry ~ V045117n + V043116 + V043210n + V043213n +
    V045145Xn +
    V041109A
Model 2: th.bush.kerry ~ V045117n + V043116n + V043210n + V043213n +
    V045145Xn +
    V041109A
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1    791 713920
2    796 738959 -5    -25039 5.5485 4.945e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Kinds of 3d Plot

- Please see the separate lecture notes on plotting in 3 dimensions. After inserting that material into this file, it simply became too unwieldy for editing.