

## The Effects Package in R

Dave Armstrong

University of Wisconsin – Milwaukee  
Department of Political Science

e: armstrod@uwm.edu

1 / 59

## The Effects Package

The `effects` package is one of the most versatile and useful pieces of software for assessing non-linear/interactive effects in linear models or for assessing any effects in non-linear models.

- You can load the package by doing:  
`library(effects)`
- The main command that does all of the work is `effect()` which has `print` and `plot` methods

The function provides predicted values holding other variables at fixed values (along with confidence intervals)

2 / 59

## Non-linear Effects in Linear Models

The effects package can plot predictions of non-linear effects induced by:

- Non-linear transformations of the independent variables
- Orthogonal polynomials in the independent variables
- Cubic splines (natural or B-) on independent variables

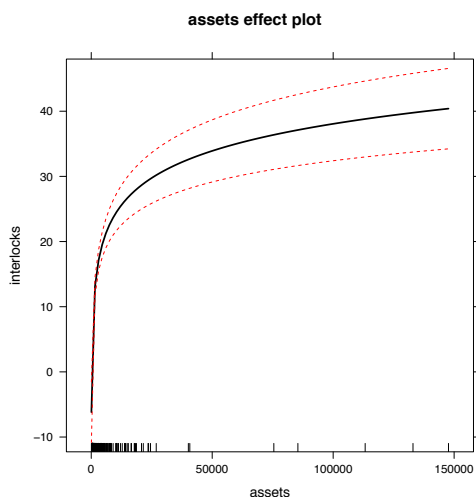
3 / 59

## Transformations

```
> library(car)
> library(effects)
> data(Ornstein)
> mod <- lm(interlocks ~ log(assets) +
+ sector + nation, data=Ornstein)
> eff <- effect("log(assets)", mod, default.levels=100)
> pdf("asset_eff.pdf", height=6, width=6)
> print(
+ plot(eff)
+ )
> invisible(dev.off())
```

4 / 59

## Transformations (2)



5 / 59

## Choosing Other Values of Covariates

By default, `effect()` holds other covariates constant at their means. If you wanted medians instead (which would do the “right” thing with dummy variables), you could do:

```
> eff2 <- effect("log(assets)", mod,
+ default.levels=100, typical = "median")
```

This still doesn't do the right things for categorical variables. Instead, you need to use the `given.values` argument, which allows you to provide values for each column of the model matrix.

- We know what values we need to provide by typing

```
> colnames(eff2$model.matrix)
[1] "(Intercept)" "log(assets)" "sectorBNK" "sectorCON" "sectorFIN"
[6] "sectorHLD" "sectorMAN" "sectorMER" "sectorMIN" "sectorTRN"
[11] "sectorWOD" "nationOTH" "nationUK" "nationUS"
```

6 / 59

## Choosing Other Values of Covariates

We probably would want to pick one sector and one nation as the baseline categories, which we could do as follows (setting the construction sector to 1 and the reference category to nation as 1):

```
> eff3 <- effect("log(assets)", mod,
+ default.levels=100,
+ given.value= c("sectorBNK" = 0,
+ "sectorCON" = 1, "sectorFIN" = 0,
+ "sectorHLD" = 0, "sectorMAN" = 0,
+ "sectorMER" = 0, "sectorMIN" = 0,
+ "sectorTRN" = 0, "sectorWOD" = 0,
+ "nationOTH" = 0, "nationUK" = 0,
+ "nationUS" = 0))
```

Then, you could do `plot(eff3)` and get the appropriate figure.

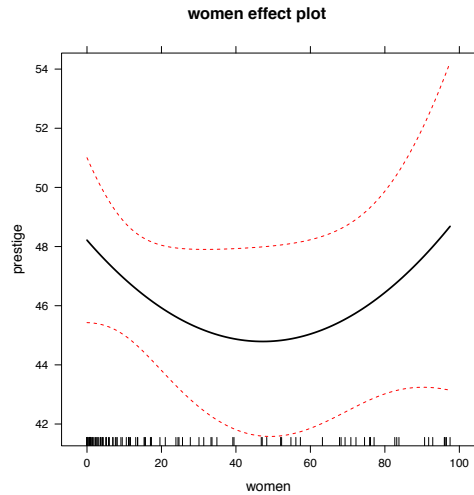
7 / 59

## Orthogonal Polynomials

```
> mod <- lm(prestige ~ income +
+ education + poly(women, 2),
+ data=Prestige)
> eff <- effect("poly(women, 2)", mod,
+ default.levels=100)
> pdf("womeff.pdf", height=6, width=6)
> plot(eff)
> invisible(dev.off())
```

8 / 59

## Orthogonal Polynomials (2)



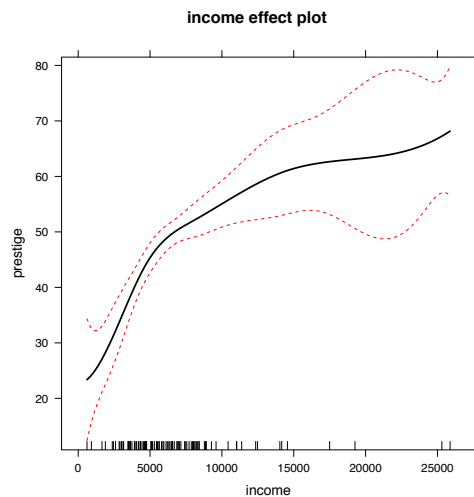
9 / 59

## Cubic Splines

```
> library(splines)
> mod <- lm(prestige ~ bs(income, df=6) +
+ education + women, data=Prestige)
> eff <- effect("bs(income, df=6)", mod,
+ default.levels=100)
> pdf("incoeff.pdf", height=6, width=6)
> plot(eff)
> invisible(dev.off())
```

10 / 59

## Cubic Splines



11 / 59

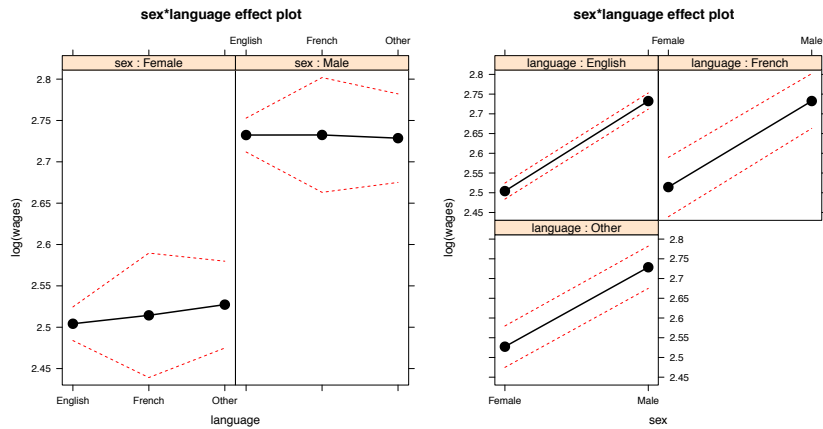
## Interaction Effects

The effects package is also good at plotting interaction effects. First, let's look at two categorical variables:

```
> data(SLID)
> mod <- lm(log(wages) ~ education + age +
+ sex*language, data=SLID)
> eff <- effect("sex*language", mod)
> pdf("sl1.pdf", height=6, width=6)
> print(plot(eff))
> invisible(dev.off())
> pdf("sl2.pdf", height=6, width=6)
> print(plot(eff, x.var="sex", as.table=T))
> invisible(dev.off())
```

12 / 59

## Two Categorical



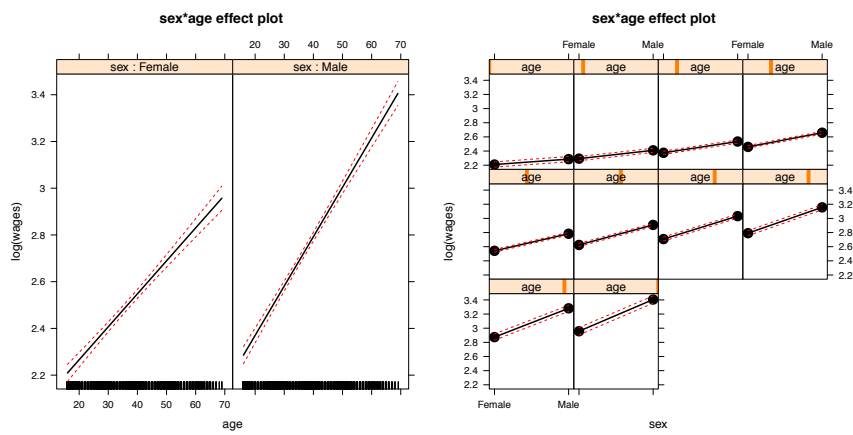
13 / 59

## One Categorical, One Continuous

```
> mod <- lm(log(wages) ~ education + language +
+ sex*age, data=SLID)
> eff <- effect("sex*age", mod)
> pdf("sl3.pdf", height=6, width=6)
> print(plot(eff, as.table=T))
> invisible(dev.off())
> pdf("sl4.pdf", height=6, width=6)
> print(plot(eff, x.var="sex", as.table=T))
> invisible(dev.off())
```

14 / 59

## One Categorical, One Continuous (2)



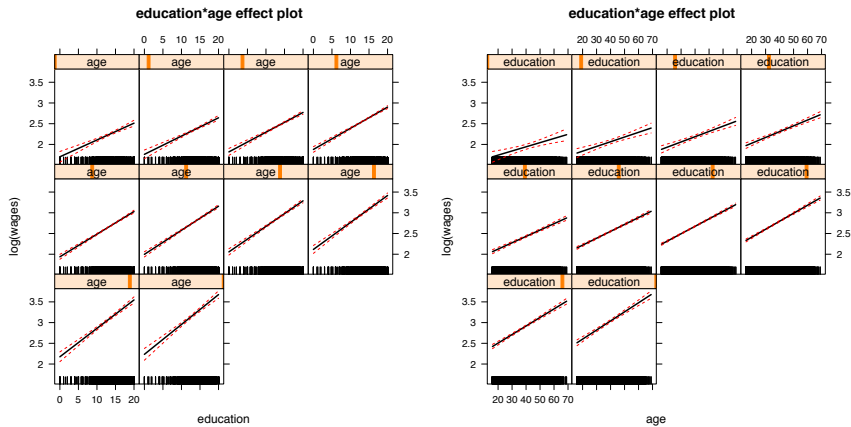
15 / 59

## Two Continuous

```
> mod <- lm(log(wages) ~ sex + language +
+ education*age, data=SLID)
> eff <- effect("education*age", mod)
> pdf("sl5.pdf", height=6, width=6)
> print(plot(eff, as.table=T))
> invisible(dev.off())
> pdf("sl6.pdf", height=6, width=6)
> print(plot(eff, x.var="age", as.table=T))
> invisible(dev.off())
```

16 / 59

## Two Continuous (2)



17 / 59

## Binary Logit/Probit

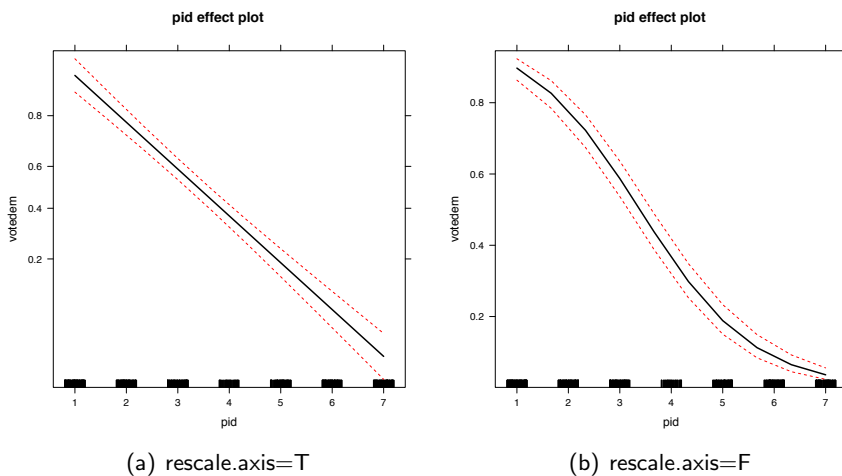
The Effects package is also great for plotting predicted probabilities.

```
> library(foreign)
> nes <- read.dta("anes1992.dta")
> mod <- glm(votedem ~ age + pid +
+ black + retnat, data=nes, family=binomial)
> eff <- effect("pid", mod,
+ given.values = c(
+ "pid" = 4,
+ "black" = 0,
+ "retnatsame" = 0,
+ "retnatworse" = 1))
> pdf("pid1.pdf", height=6, width=6)
> print(plot(eff))
> invisible(dev.off())
> pdf("pid2.pdf", height=6, width=6)
> print(plot(eff, rescale.axis=F))
> invisible(dev.off())
```

18 / 59

## Party ID Effects

Figure: Rescaling Axis



(a) rescale.axis=T

(b) rescale.axis=F

19 / 59

In our ordered data models, we usually model the cumulative probability.

- We are modeling the probability of being in a category *less than* the one of interest. In the ordered logit (the one we will be mostly talking about):

$$\begin{aligned} Pr(Y \leq m) &= \Lambda(\tau_m - \mathbf{X}\beta) \\ &= \frac{1}{1 + e^{-(\tau_m - \mathbf{X}\beta)}} \end{aligned}$$

We can also find the probability that  $Y$  is in any particular category:

$$\begin{aligned} Pr(Y = m) &= \Lambda(\tau_m - \mathbf{X}\beta) - \Lambda(\tau_{m-1} - \mathbf{X}\beta) \\ &= \frac{1}{1 + e^{-(\tau_m - \mathbf{X}\beta)}} - \frac{1}{1 + e^{-(\tau_{m-1} - \mathbf{X}\beta)}} \end{aligned}$$

20 / 59

## Identification

These models need to have some parameters fixed to be identified. Specifically,

- $\beta_0 = 0$  The intercept in the linear predictor is set to zero
- $\tau_0 = -\infty$
- $\tau_m = \infty$

In R, these models are estimated with the `polr` function in the `MASS` library.

21 / 59

## Example

```
> options(useFancyQuotes=F)
> library(foreign)
> fh <- read.dta("fh2006_ss.dta")
> library(MASS)
> mod <- polr(freedomstat ~ log(pop) + gdppc10k + civ, data=fh)
> summary(mod)
Call:
polr(formula = freedomstat ~ log(pop) + gdppc10k + civ, data = fh)

Coefficients:
                Value Std. Error t value
log(pop)         -0.2556   0.09261 -2.7596
gdppc10k          0.5394   0.24689  2.1848
civIslamic       -1.3684   0.52840 -2.5897
civLatin American 1.4775   0.57628  2.5638
civOrthodox       0.2807   0.66007  0.4253
civOther          0.8716   0.49292  1.7681
civWestern        4.1342   1.16519  3.5481

Intercepts:
                Value Std. Error t value
not free|partly free -2.5367   0.9004  -2.8172
partly free|free     -0.4965   0.8748  -0.5675

Residual Deviance: 264.0614
AIC: 282.0614
```

22 / 59

## Interpretation (1)

The first thing you might notice is that there are no  $p$ -values. This could make it difficult to do proper hypothesis tests. We could get  $p$ -values as follows:

```
> smod <- summary(mod)
> p.val <- 2*pt(abs(smod$coef[,3]),
+ mod$df.residual, lower.tail=F)
> round(cbind(smod$coef, p.val), 4)
```

	Value	Std. Error	t value	p.val
log(pop)	-0.2556	0.0926	-2.7596	0.0064
gdppc10k	0.5394	0.2469	2.1848	0.0303
civIslamic	-1.3684	0.5284	-2.5897	0.0104
civLatin American	1.4775	0.5763	2.5638	0.0112
civOrthodox	0.2807	0.6601	0.4253	0.6712
civOther	0.8716	0.4929	1.7681	0.0789
civWestern	4.1342	1.1652	3.5481	0.0005
not free partly free	-2.5367	0.9004	-2.8172	0.0054
partly free free	-0.4965	0.8748	-0.5675	0.5711

23 / 59

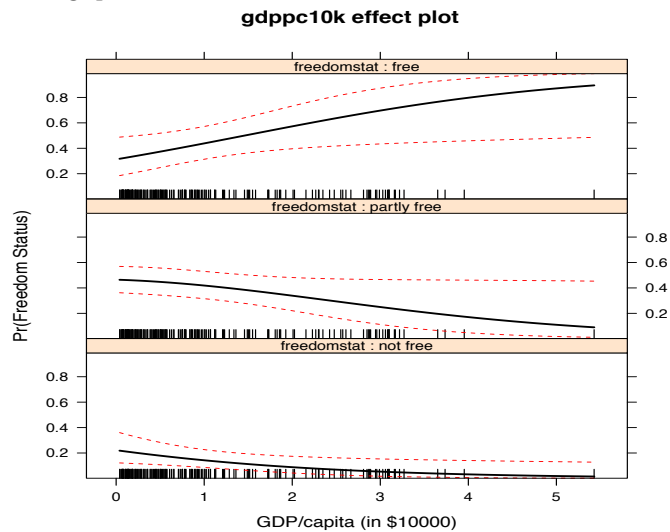
## Fitted Probabilities for Continuous Variables

We can use the `effects` package to get fitted probabilities for continuous covariates. For example:

```
> gdp.eff <- effect("gdppc10k", mod,
+ default.levels=100)
```

24 / 59

Using `plot(gdp.eff)` we get the following graph.



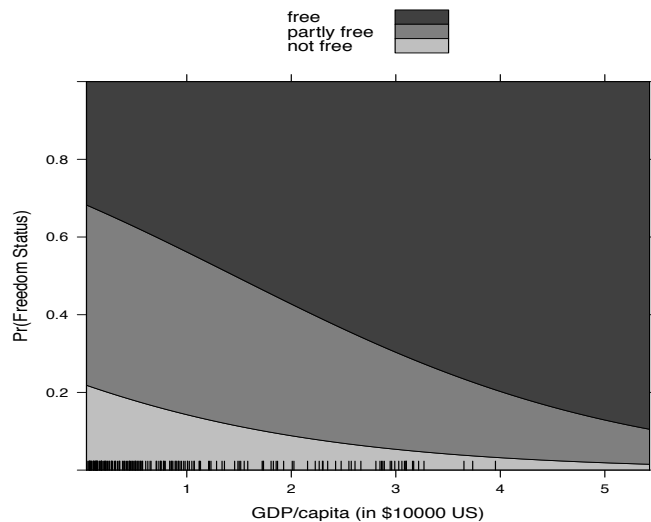
25 / 59

## Stacked Probabilities

```
> print(  
+ plot(gdp.eff, xlab="GDP/capita (in $10000 US)",  
+ ylab = "Pr(Freedom Status)", style="stacked",  
+ colors=c("gray75", "gray50", "gray25"),  
+ main = "")  
+ )
```

26 / 59

## Stacked Graph



27 / 59

## Extracting elements from the effects

If we wanted to modify the graph significantly, we would need to extract some of the elements of the `gdp.eff` object. Specifically, we want.

- Fitted probabilities
- 95% confidence bounds
- Values of `gdppc10k` used to calculate effects.

28 / 59

## Extracting... (2)

Here, we can take the various relevant pieces out of the effects object.

```
> plot.dat <- data.frame(
+   probs = c(gdp.eff$prob),
+   status = rep(levels(fh$freedomstat), each=100),
+   lower = c(gdp.eff$lower.prob),
+   upper = c(gdp.eff$upper.prob),
+   gdppc10k = c(gdp.eff$x)
```

29 / 59

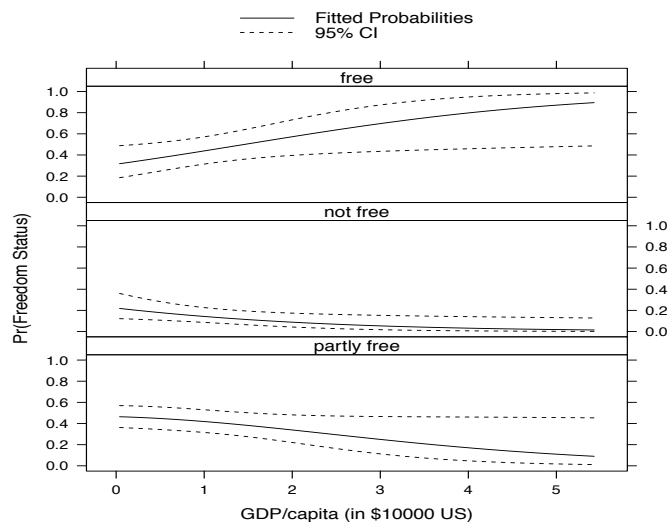
## Making the Graph

We can then make the graph ourselves if we like:

```
> trellis.par.set(strip.background = list(col=NA))
> print(
+   xyplot(probs ~ gdppc10k | status, data=plot.dat,
+     xlab = "GDP/capita (in $10000 US)",
+     ylab = "Pr(Freedom Status)",
+     layout=c(1,3), as.table=T,
+     ylim=c(-.05, 1.05),
+     panel = function(x,y,subscripts){
+       panel.lines(x,y,lty=1, col="black")
+       panel.lines(x,plot.dat$lower[subscripts],
+         lty=2, col="black")
+       panel.lines(x,plot.dat$upper[subscripts],
+         lty=2, col="black")},
+     key=list(space="top",
+       lines=list(col=c("black", "black"),
+         lty=c(1,2)),
+       text=list(c("Fitted Probabilities",
+         "95% CI"))))
+ )
```

30 / 59

## The Graph



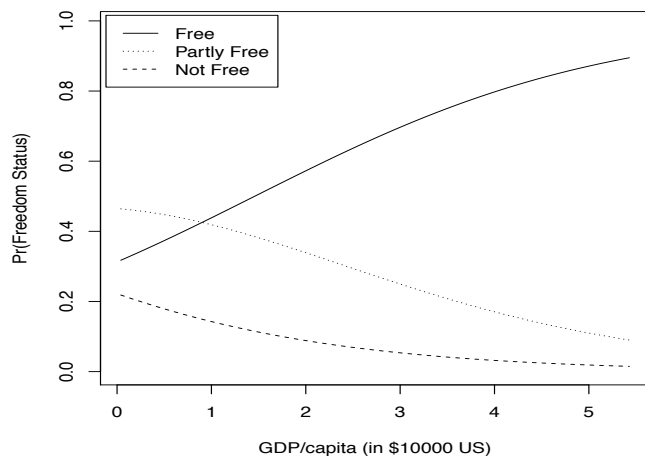
31 / 59

## Or, on the Same Panel

```
> plot(range(plot.dat$gdppc10k),
+   range(c(plot.dat$lower, plot.dat$upper)),
+   type = "n", xlab= "GDP/capita (in $10000 US)",
+   ylab = "Pr(Freedom Status)")
> sp <- split(plot.dat, plot.dat$status)
> lines(sp[[1]]$gdppc10k, sp[[1]]$probs, lty=1)
> lines(sp[[2]]$gdppc10k, sp[[2]]$probs, lty=2)
> lines(sp[[3]]$gdppc10k, sp[[3]]$probs, lty=3)
> legend("topleft", c("Free", "Partly Free", "Not Free"),
+   lty=c(1,3,2), inset=.01)
```

32 / 59

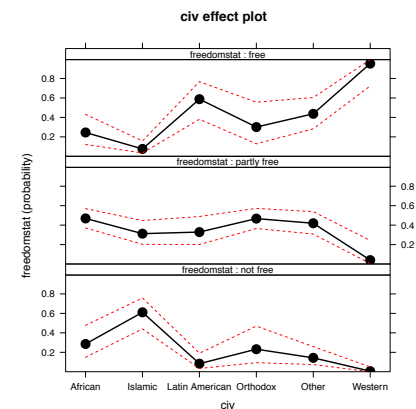
## The Other Graph



33 / 59

## The Effects Graph

```
> civ.eff <- effect(
+ "civ", mod)
> print(
+ plot(civ.eff)
+ )
```



34 / 59

## Effect for civilization

```
> plot.dat <- data.frame(
+ probs = c(civ.eff$prob),
+ lower = c(civ.eff$lower.prob),
+ upper = c(civ.eff$upper.prob),
+ status = rep(c("Not Free",
+ "Partly Free", "Free"), each=6),
+ civ = as.factor(rep(civ.eff$x[[1]], 3)))
```

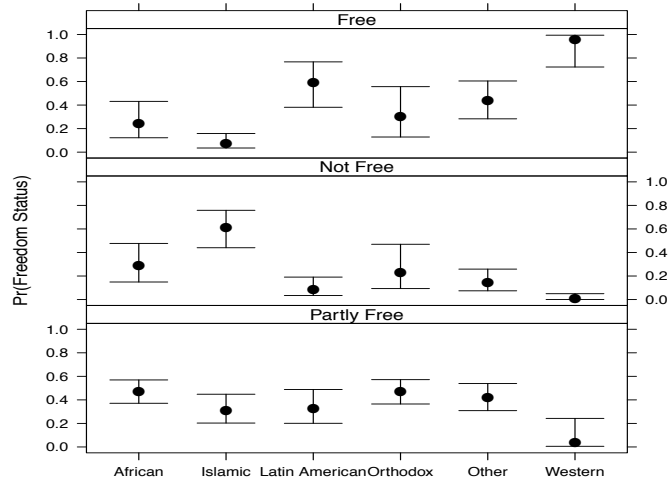
35 / 59

## Making the Graph

```
> trellis.par.set(strip.background =
+ list(col=NA))
> print(
+ xyplot(probs ~ civ | status,
+ data=plot.dat, as.table=T,
+ ylim=c(-.05, 1.05), xlab = "",
+ ylab = "Pr(Freedom Status)",
+ layout = c(1,3),
+ panel = function(x,y,subscripts){
+ panel.points(x,y,pch=16,
+ col="black", cex=1.25)
+ panel.arrows(
+ x, plot.dat$lower[subscripts],
+ x, plot.dat$upper[subscripts],
+ col="black", angle=90, code=3)})
+ )
```

36 / 59

## The Graph



37 / 59

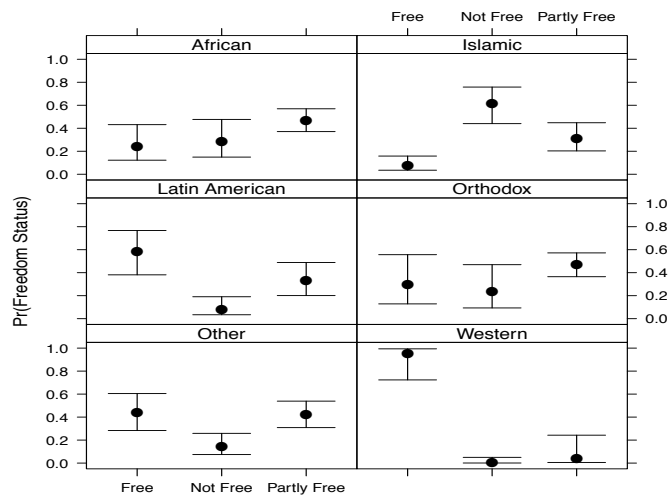
## Making the Graph by Civilizations

```

> trellis.par.set(strip.background=
+ list(col=NA))
> print(
+ xyplot(probs ~ status | civ,
+ data=plot.dat, as.table=T,
+ ylim=c(-.05, 1.05), xlab = "",
+ ylab = "Pr(Freedom Status)",
+ layout = c(2,3),
+ panel = function(x,y,subscripts){
+   panel.points(x,y,pch=16,
+     col="black", cex=1.25)
+   panel.arrows(
+     x, plot.dat$lower[subscripts],
+     x, plot.dat$upper[subscripts],
+     col="black", angle=90, code=3)})
+ )
    
```

38 / 59

## The Graph



39 / 59

## Multinomial Logit

For the MNL model, we can figure out the probability estimated to be in any category as:

$$Pr(Y = m) = \frac{e^{\mathbf{X}\beta_m}}{1 + \sum_m e^{\mathbf{X}\beta_m}}$$

For identification purposes, we have to fix one set of coefficients at zero. Usually this is:

$$\beta_1 = 0$$

40 / 59

## Voting Data in France

vote vote choice with party labels  
votenum numeric vote variable  
age exact age in years  
male dichotomous variable 1=male, 0=female  
lrsel left-right self-placement (1=left - 10=right)  
demsat satisfaction with democracy (1=very satisfied - 4=not at all satisfied)  
notrelig dichotomous variable 1=not religions, 0=religious  
urban size of community (1-5)  
eususp support for EU 1=positive, 2=ambivalent, 3=negative  
occupa2 dichotomous variable 1=lower non-manual workers, 0 otherwise  
occupa3 dichotomous variable 1=owhers and higher non-manual workers, 0 otherwise  
edulevel level of education 1=No HS, 2=some HS, 3=college, 4=post-college/grad  
retnat retrospective economic evaluations 1=better, 2=same, 3=worse.

41 / 59

## Reading in the Data and Running the Model

```
> library(foreign)
> frdat1 <- read.dta("fra94ds_rintro.dta")
> frdat <- read.dta("fra94ds_rintro.dta", convert.factors=F)
> frdat$vote <- frdat1$vote
> frdat$vote <- as.factor(as.character(frdat$vote))
> frdat$retnat <- factor(frdat$retnat,
+ levels=1:3, labels=c("better", "same", "worse"))
> rm(frdat1)
> library(nnet)
> multi.mod <- multinom(vote ~ lrsel + retnat + age + male +
+ edulevel + notrelig + urban + demsat + occupa2 + occupa3 +
+ eusup, data=frdat)

# weights: 70 (52 variable)
initial value 872.315349
iter 10 value 588.306537
iter 20 value 528.812279
iter 30 value 511.468623
iter 40 value 507.520375
iter 50 value 507.162740
iter 60 value 507.159631
final value 507.159622
converged
```

42 / 59

## Model Summary

```
> summary(multi.mod)
Call:
multinom(formula = vote ~ lrsel + retnat + age + male + edulevel +
notrelig + urban + demsat + occupa2 + occupa3 + eusup, data = frdat)

Coefficients:
(Intercept) lrsel retnatsame retnatworse age male
PCF 6.646025 -0.9186144 1.4496012 0.9267292 0.006637629 0.1883463
PS 7.212172 -0.6118981 -0.2458185 -0.3476691 0.022012486 0.4179970
RPR -4.662170 1.2431696 -0.9251467 -0.6497244 0.044693613 0.8297589
UDF -1.549249 1.0156698 0.1843344 0.3933059 0.042785077 1.3598392

edulevel notrelig urban demsat occupa2 occupa3
PCF -0.2589632 0.62543595 0.08472624 -0.7067206 0.2809062 -0.6755855
PS -0.6032898 -0.15973989 -0.02766268 -1.0633042 0.4329946 0.3391748
RPR -0.1661496 -0.39267746 -0.29934401 -0.6240611 0.1822301 0.5612063
UDF -0.5198723 0.01077893 -0.26006825 -1.0981277 0.4368764 1.2875438

eusup
PCF -1.2878848
PS -0.1096627
RPR -0.1117965
UDF -0.6900913

Std. Errors:
(Intercept) lrsel retnatsame retnatworse age male edulevel
PCF 1.936166 0.1680044 0.7366045 0.7680104 0.01495767 0.4479948 0.2746386
PS 1.544002 0.1274739 0.4562967 0.4919267 0.01160671 0.3389716 0.2034899
RPR 1.985625 0.1727249 0.5255688 0.5805700 0.01415554 0.4271482 0.2554229
UDF 1.921822 0.1678897 0.5436779 0.5949859 0.01397518 0.4234847 0.2522958

notrelig urban demsat occupa2 occupa3 eusup
PCF 0.4712602 0.1643484 0.3032612 0.4962543 0.6521656 0.3726496
PS 0.3377095 0.1194552 0.2403707 0.3902064 0.4477477 0.3117390
RPR 0.4271869 0.1495149 0.3079757 0.4984855 0.5619010 0.3880234
UDF 0.4223361 0.1470873 0.3100356 0.5030913 0.5455107 0.3771189

Residual Deviance: 1014.319
AIC: 1118.319
```

43 / 59

## Interpretation (1)

The first thing you might notice is that you don't get t-statistics or p-values with these things. You can make them easily enough though

```
> smulti <- summary(multi.mod)
> multi.t <- smulti$coefficients/smulti$standard.errors
> multi.p <- pt(abs(multi.t),
+ nrow(multi.mod$fitted.values)-multi.mod$edf,
+ lower.tail=F)
> b <- round(smulti$coefficients, 3)
> b[which(multi.p > .05, arr.ind=T)] <- ""
```

44 / 59

## Printing the Coefficients

```
> noquote(b)
      (Intercept) lrself retnatsame retnatworse age   male edulevel notrelig
PCF  6.646      -0.919  1.45
PS   7.212      -0.612                0.022   -0.603
RPR -4.662      1.243  -0.925                0.045  0.83
UDF  1.016      1.016                0.043  1.36  -0.52
      urban  demsat  occupa2  occupa3  eusup
PCF      -0.707                -1.288
PS      -1.063
RPR    -0.299  -0.624
UDF    -0.26  -1.098      1.288  -0.69
```

45 / 59

## Effects in MNL

Even though the ordered and multinomial logit models are quite different in specification, the effects package works very similarly in both settings.

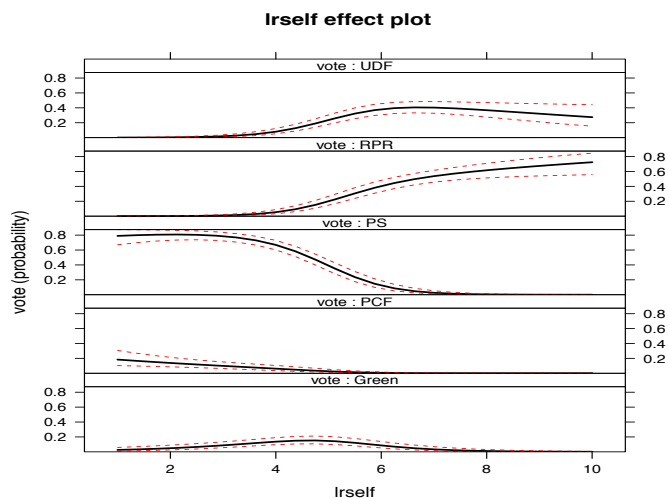
```
> lrs.eff <- effect("lrself",
+ multi.mod, default.levels=25)
```

We can use the effects package to make graphs for us, if we so choose

46 / 59

## Multi-panel Plot

```
> plot(
+ plot(lrs.eff, rug=FALSE)
+ )
```

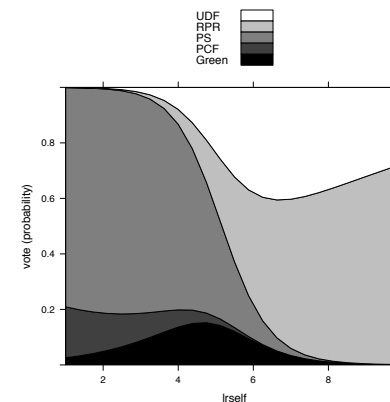


47 / 59

## Stacked MNL Graph

We could also make a stacked graph, just like with the OL model

```
> plot(
+ plot(lrs.eff,
+ rug=FALSE,
+ style="stacked",
+ colors =
+   c("gray0", "gray25",
+     "gray50", "gray75",
+     "gray100"),
+ main = "")
+ )
```



48 / 59

## More Control

If you want more control over the graph, you can take the data out of the effect output just like above.

```
> plot.dat <- data.frame(
+   prob = c(lrs.eff$prob),
+   lower = c(lrs.eff$lower.prob),
+   upper = c(lrs.eff$upper.prob),
+   lrs = c(lrs.eff$x[[1]]),
+   party = rep(lrs.eff$y.levels, each=25))
```

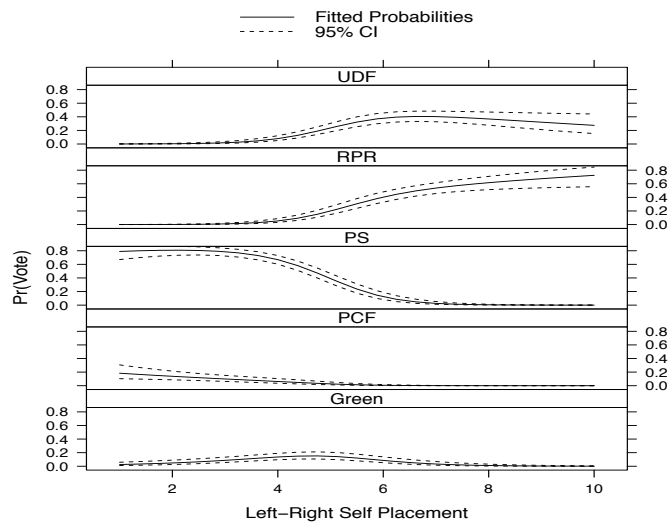
49 / 59

## Making the Graph

```
> trellis.par.set(
+   strip.background = list(col=NA))
> print(
+   xyplot(prob ~ lrs | party,
+   data = plot.dat, layout=c(1,5),
+   xlab = "Left-Right Self Placement",
+   ylab = "Pr(Vote)",
+   panel = function(x,y,subscripts){
+     panel.lines(x,y,col="black")
+     panel.lines(x, plot.dat$lower[subscripts],
+       col="black", lty=2)
+     panel.lines(x, plot.dat$upper[subscripts],
+       col="black", lty=2)},
+   key=list(space="top",
+     lines=list(lty=c(1,2),
+       col=c("black", "black")),
+     text=list(c("Fitted Probabilities",
+       "95% CI"))))
+ )
```

50 / 59

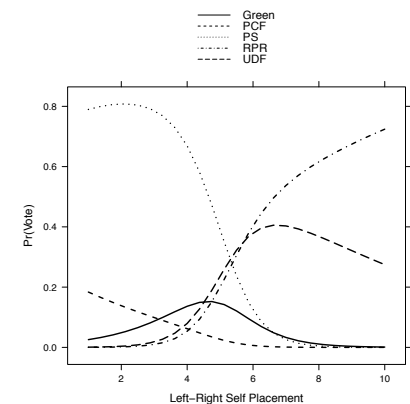
## The Graph



51 / 59

## All on One Panel

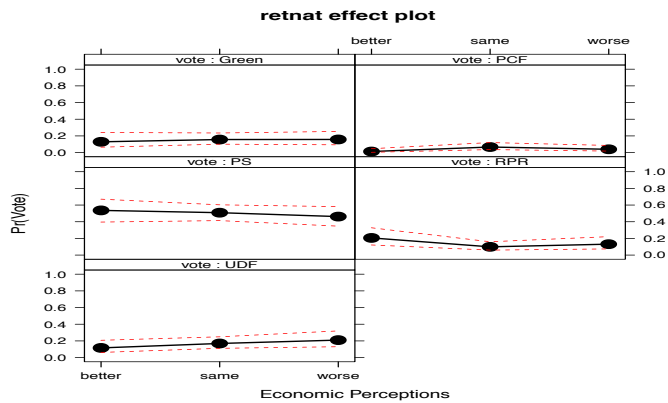
```
> print(
+   xyplot(prob ~ lrs,
+   groups=party,
+   type="l",
+   data = plot.dat,
+   xlab =
+   "Left-Right Self Placement",
+   ylab = "Pr(Vote)",
+   lty=c(1:5),
+   col="black",
+   lwd=2,
+   key=list(space="top",
+     lines=list(
+       lty=1:5,
+       col="black"),
+     text=list(
+       lrs.eff$y.levels)))
+ )
```



52 / 59

## Let's Look at the effect of the Economy

```
> ret.eff <- effect("retnat", multi.mod)
> print(
+ plot(ret.eff, as.table=T,
+ layout=c(2,3),
+ ylim = c(-.05, 1.05),
+ xlab = "Economic Perceptions",
+ ylab = "Pr(Vote)")
+ )
```



53 / 59

## Making the Graph Ourselves: Data

```
> plot.dat <- data.frame(
+ prob = c(ret.eff$prob),
+ lower = c(ret.eff$lower.prob),
+ upper = c(ret.eff$upper.prob),
+ ret = factor(c(ret.eff$x[[1]]),
+ levels=1:3,
+ labels=c("better", "same", "worse")),
+ party = rep(ret.eff$y.levels, each=3))
```

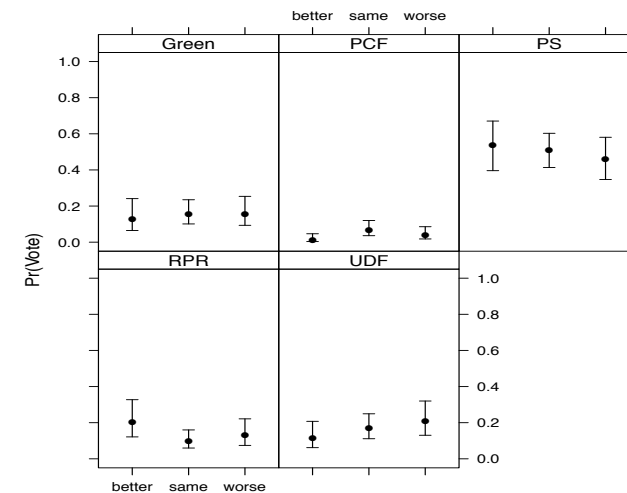
54 / 59

## Making the Graph

```
> trellis.par.set(
+ strip.background = list(col=NA))
> print(
+ xyplot(prob ~ ret | party,
+ data=plot.dat,
+ as.table=T,
+ xlab = "", ylab = "Pr(Vote)",
+ ylim= c(-.05, 1.05),
+ panel = function(x,y,subscripts){
+ panel.arrows(
+ x, plot.dat$lower[subscripts],
+ x, plot.dat$upper[subscripts],
+ code = 3, angle=90, length=.05)
+ panel.points(x,y,
+ pch=16, col="black")})
+ )
```

55 / 59

## The Graph



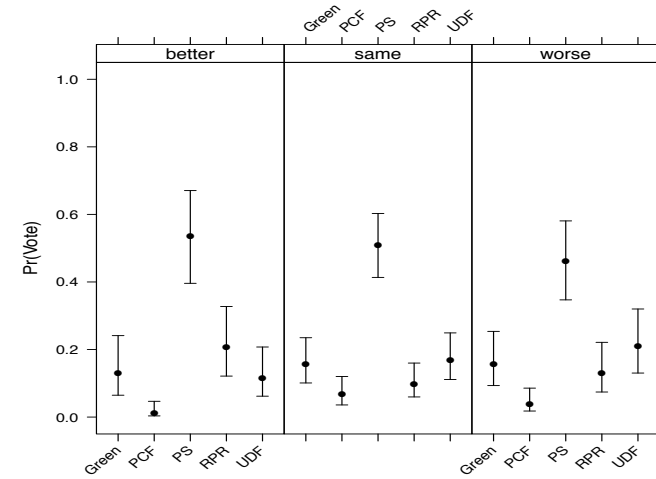
56 / 59

## Making the Graph, by Economy

```
> trellis.par.set(
+   strip.background = list(col=NA))
> print(
+   xyplot(prob ~ party | ret,
+     data=plot.dat,
+     as.table=T,
+     layout = c(3,1),
+     scales=list(x=list(rot=45)),
+     xlab = "", ylab = "Pr(Vote)",
+     ylim= c(-.05, 1.05),
+     panel = function(x,y,subscripts){
+       panel.arrows(
+         x, plot.dat$lower[subscripts],
+         x, plot.dat$upper[subscripts],
+         code = 3, angle=90, length=.05)
+       panel.points(x,y,
+         pch=16, col="black")}))
+ )
```

57 / 59

## The Graph



58 / 59