CSSS 569: Visualizing Data

Visualizing Inference

Christopher Adolph[*]

University of Washington, Seattle

February 23, 2011

[*]Assistant Professor, Department of Political Science and Center for Statistics and the Social Sciences.

Presenting Estimated Models in Social Science

Most empirical work in social science is regression model-driven

These models are

- full of covariates
- often non-linear
- often involve interactions & transformations

If there is anything we need to visualize well, it is our models

Yet most people just print off tables of parameter estimates from their software

Today: an improved, generic approach to model presentation

Emphasis on algorthms/programming: Lots of code for future study.

Model presentation as a step in research design

Ask a research question: Why do people vote?

Operationalize: What variables predict voting, and to what extent?

Analyze: How do the data answer these questions?

Present results: **?**

How do we present our analysis as an answer to the research question?

Answer in the terms of the question and variables, instead of statistical jargon

Model presentation as a step in research design

Desired style of answer:

The variables which best explain who votes are age and education. Each year increase in age increases the probability of voting by ___, plus or minus ___, and . . .

But wait! Probability is a nonlinear function of covariates!

A cryptic, weaker, but more accurate answer:

Higher ages are associated with greater probability of voting. The logit coefficient for age was ___ with a standard error of ___.

Weakest; bordering on non-responsive:

Variable ___ was the most significant predictor of voting, with a p-value less than 0.001, and a positive association with voting.

Coefficients are not enough

Social scientists' love affair with regression coefficient output is bizarre:

- Everything written in terms of arcane intermediate quantites (e.g., probit coefficients)
- Little effort to transform these results back to the scale of the quantities of interest
- Little effort to make informative statements about estimation uncertainty
- Little visualization at all, or graphs with low data-ink ratios

Why do we focus on non-linear coefficients?

Ultimately, we aren't interested in the βs, except in linear regression

There, the β's are partial derivatives, and easy to interpret
(as long as there aren't polynomials or interactions . . .
which are very common)

In other models, βs are parameters in a non-linear calculation.
We're interested in the *result* of that calculation.

That is, we want to know the behavior of $\mathrm{E}(y|x)$ as we vary x.

An alternative

1. Run your model as normal. Treat the output as an intermediate step.

2. Translate your model results back into the scale of the response variable

   - Modeling war?
     Show the change in probability of war associated with X

   - Modeling counts of crimes committed?
     Show how those counts vary with X

   - Unemployment rate time series?
     Show how a change in X shifts unemployment over the following t years

   These results are the **Quantity of Interest (QoI)**.

3. Graph as many scenarios of the quantity of interest as you need to fully explain your findings.

A bit more formally. . .

We want to know the behavior of $\mathrm{E}(y|x)$ as we vary x.

In non-linear models with multiple regressors, this gets tricky.

The effect of x_1 depends on $x_{\sim 1}$ & $\hat{\beta}_{\sim 1}$

So let's put them all together and calculate $\hat{Y}$

Generally, we will need to make a set of "counterfactual" assumptions:

$x_1 = a,\ x_2 = b,\ x_3 = c,\ \ldots$

- Choose $a, b, c \ldots$ to match some particular counterfactual case of interest *or*

- Hold all but one of the x's at their mean values (or other reference baseline), then vary the remaining x systematically.

The same trick works if we are after differences in y related to changes in x,
$\mathrm{E}(y_{\text{scen1}} - y_{\text{scen2}}|x_{\text{scen1}}, x_{\text{scen2}})$

Calculating quantities of interest

Our goal to obtain "quantities of interest," like

- Expected Values: $\mathrm{E}(Y|X_c)$
- First Differences: $\mathrm{E}(Y|X_{c2}) - \mathrm{E}(Y|X_{c1})$
- Relative Risks: $\mathrm{E}(Y|X_{c2})/\mathrm{E}(Y|X_{c1})$
- Any function of the above

for some counterfactual X_c's.

Voting example

Suppose we are modeling the probability of voting, using data from the 2000 NES

We have whether a respondent voted, their age, and their education (less than high school high school, or college)

We fit the following model:

$$\begin{aligned} \text{Vote}_i &\sim \text{Bernoulli}(\pi_i) \\ \pi_i &= \text{logit}^{-1}(\beta_0 + \beta_1 \text{Age}_i + \beta_2 \text{Age}_i^2 + \beta_3 \text{HSdeg}_i + \beta_4 \text{CollDeg}_i) \end{aligned}$$

Include a quadratic term in case the very old have more trouble voting

HSdeg and CollDeg are cumulative:
High school drop-outs get zeros for each,
High school grads get a 1 for HSdeg and a 0 for CollDeg,
and college grads get 1s for both

Voting example

	est.	s.e.	p-value
Age	0.07	0.02	0.000
Age^2	-0.000431	0.000165	0.009
High School Grad	1.17	0.18	0.000
College Grad	1.09	0.13	0.000
Constant	-3.05	0.42	0.000
log likelihood	-1040.6		
N	1798		

Table 1: *Determinants of voting in the 2000 presidential election.* Entries are logit coefficients. Data taken from the 2000 American National Election Survey.

How do we read these results? Would odds ratios help?

Can you tell me $\Pr(\text{Vote}|\text{Age} = 50)$?
Can you tell me $\Pr(\text{Vote}|\text{Age} = 50) - \Pr(\text{Vote}|\text{Age} = 25)$?
Confidence intervals around these quantities?

Calculating quantities of interest (QoIs)

Not really. We need to calculate these quantities for the reader

Some notation:

$X_{\text{Age}=50}$ refers to a scenario with every X at it's mean, except Age, which is 50

$X_{\text{Age}=25}$ refers to a scenario with every X at it's mean, except Age, which is 25

Now calculate

$$\begin{aligned}
\Delta \widehat{\Pr(\text{Voting})} &= \frac{1}{1+\exp(-X_{\text{Age}=50}\hat{\beta})} - \frac{1}{1+\exp(-X_{\text{Age}=25}\hat{\beta})} \\
&= 0.719 - 0.476 \\
&= 0.243
\end{aligned}$$

In `R`:

```
y.lo <- 1/(1 + exp(-t(x.lo)%*%betas))
y.hi <- 1/(1 + exp(-t(x.hi)%*%betas))
y.fd <- y.hi - y.lo
```

Simulating quantities of interest

All we have done is taken some intermediate quantities (the model parameters) and put them back into terms we care about: changes in the response

We could do this for any model:
just plug X_c and $\hat{\beta}$ into the systematic component of the model

This is what the command `predict(...,newdata=x.lo)` does for us

But we also want the confidence intervals of these quantities,
which are tiresome or impossible to calculate analytically,
(and which `predict()` doesn't always provide)

and we want a method that works for any model, not just logit.

Simulating quantities of interest

We need a generic simulation method for models like:

$$Y_i \sim f(\mu_i, \alpha) \qquad \mu_i = g(\beta, X_i)$$

(for convenience, $\theta = \text{vec}(\beta, \alpha)$)

Note that this is framework I introduced last week.

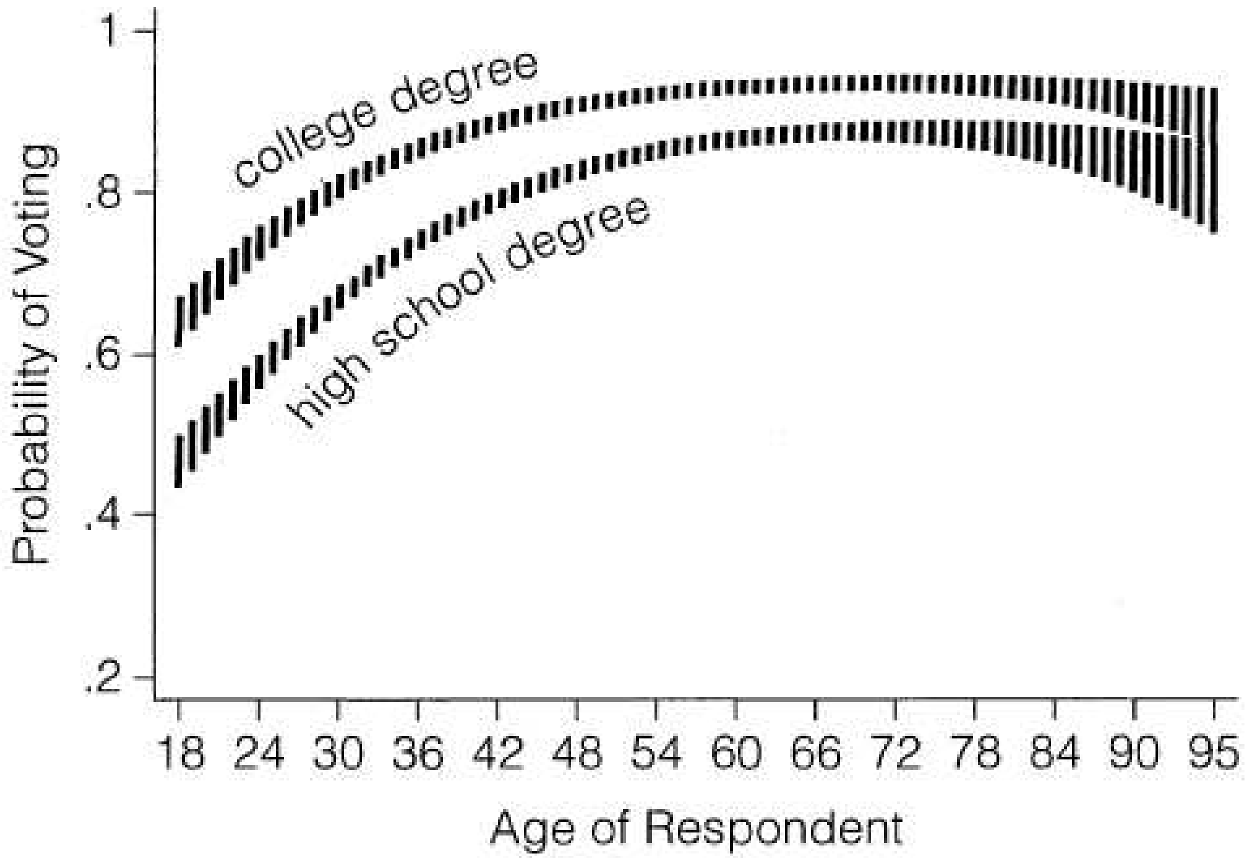
It's very general; most models you know nest within it.

King, Tomz, Wittenberg (*AJPS* 2000) provide a generic algorthm

Apply it to these data and produce. . .

When simulation is the only reasonable option

FIGURE 1 **Probability of Voting by Age**



Vertical bars indicate 99-percent confidence intervals

Algorthm for simulating EVs and their confidence intervals

1. Choose a counterfactual X_c
2. Estimate the model and obtain the parameter vector, $\hat{\boldsymbol{\theta}}$, and its variance covariance matrix, $\hat{\mathrm{V}}(\hat{\boldsymbol{\theta}})$.
3. Draw θ from the multivariate normal $f_{\mathcal{MVN}}\left(\hat{\boldsymbol{\theta}}, \hat{\mathrm{V}}(\hat{\boldsymbol{\theta}})\right)$.
4. Calculate $\tilde{\mu} = g(\tilde{\boldsymbol{\beta}}, X)$
5. Draw $\tilde{Y} \sim f(\tilde{\mu}, \tilde{\alpha})$, from the *model's* distribution. This is a *predicted value*.
6. Repeat steps 3 to 5 many times, averaging the results to get one *expected value*.
7. Repeat step 6 many times to obtain a vector of expected values.
8. Summarize this vector using means, std dev., or confidence intervals.

The algorithm applied to logit

1. Choose a counterfactual X_c

2. Estimate the model and obtain the parameter vector, $\hat{\boldsymbol{\beta}}$, and its variance covariance matrix, $\hat{V}(\hat{\boldsymbol{\beta}})$.

3. Draw $\tilde{\boldsymbol{\beta}}$ from the multivariate normal $f_{\mathcal{MVN}}\left(\hat{\boldsymbol{\beta}}, \hat{V}(\hat{\boldsymbol{\beta}})\right)$.

4. Calculate $\tilde{\mu} = 1/(1 + \exp(-X_c\tilde{\boldsymbol{\beta}})$

5. Draw $\tilde{Y} \sim f_{\text{Bernoulli}}(\tilde{\mu})$. This is a *predicted value* (a 1 or a 0).

6. Repeat steps 3 to 5 many times, averaging the results to get one *expected value* (i.e., a π).

7. Repeat step 6 many times to obtain a vector of expected values.

8. Summarize this vector using means, std dev., or confidence intervals.

A shortcut

It's a good idea to simulate at least 1,000 PVs to get a single EV.

And at least 1,000 EVs to get its distribution.

Which means doing a million sims for each X_c.

There is a shortcut. If $E(Y) = \mu$ (which is the case for linear regression, logit, probit, Poisson, etc.), then you can skip steps 5 and 6, and just repeat step 4 to get your EVs.

If you want PVs, though, you have to draw from the model distribution, to inclunde "fundamental uncertainty".

EVs include only "estimation uncertainty". PVs include both.

Several packages implement this algorithm: `Zelig` from King & coauthors and `simcf`, which is bundled with `tile`

Could also do by hand, or use `predict` in some cases

Graphing simulations from models

Challenges for presentation:

1. Our QoIs and their CIs trace out non-linear functions
2. Contain lots of information: easy to produce hundreds of data points
3. Often, counterfactuals producing QoIs vary on multiple dimensions

Once we simulate our quantities of interest and confidence intervals, we need visual methods for presenting them

Graphing simulations from models

Two proposed general graphical methods:

- **lineplot**: For models with at least one continuously varying counterfactual variable, use a line varying over that counterfactual, and as many lines as you have discrete counterfactuals

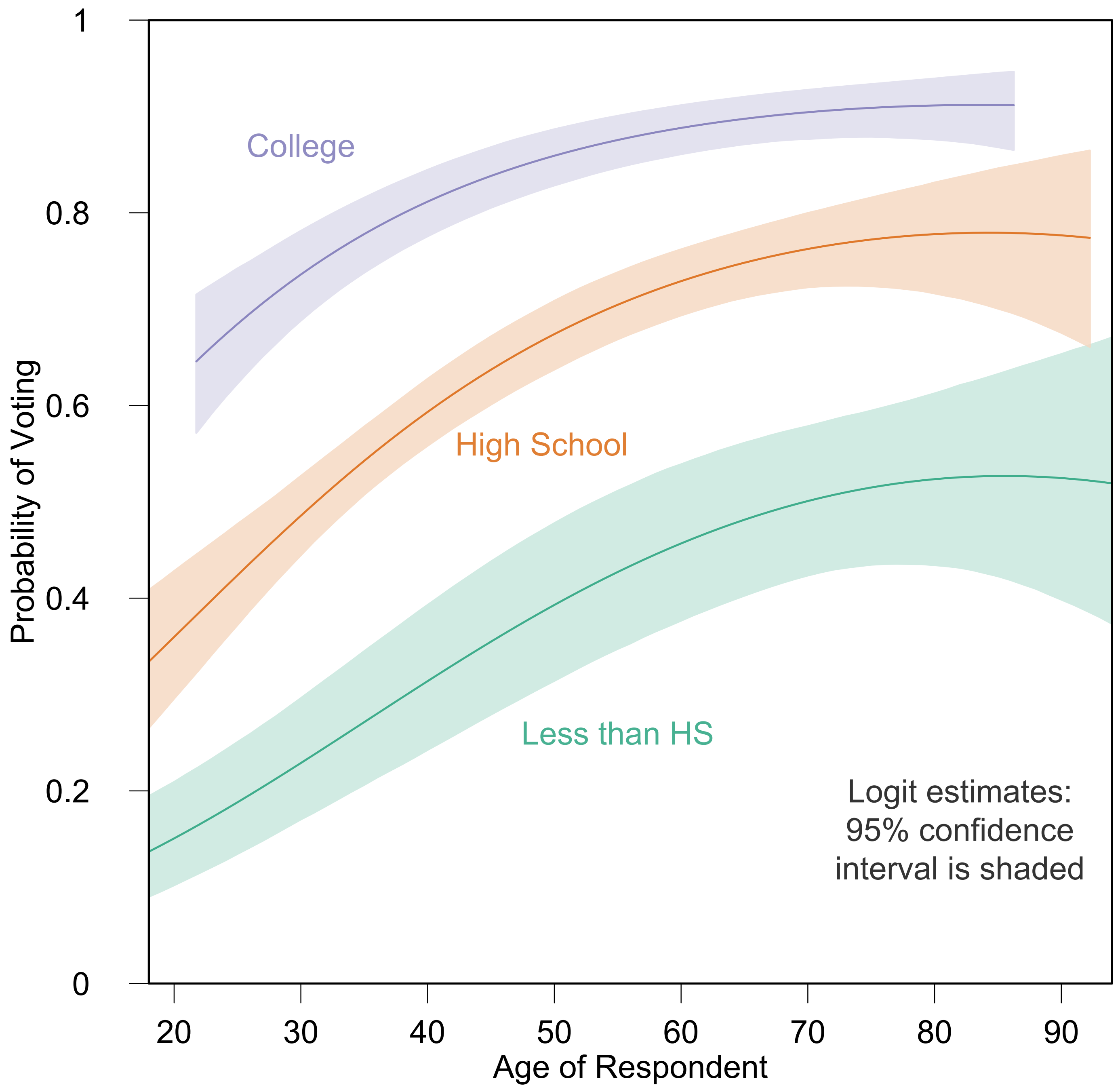
  Note this will focus attention on you continuous counterfactuals, which may not be your intention

- **ropeladder**: For any model, including those with only discrete counterfactual variables, or a combination of discrete and continuous

  ropeladders put equal attention on covariates that vary continuously and those that vary discretely.

  Can accommodate large numbers of different counterfactual scenarios

Let's use lineplots for our voting example, since we have one continuous and one categorical covariate


1
0.8
0.6
0.4
0.2
0
Probability of Voting
20
30
40
50
60
70
80
90
Age of Respondent
College
High School
Less than HS
Logit estimates:
95% confidence
interval is shaded

Lineplot example

```
# Load libraries
library(MASS)
library(simcf)
library(tile)
library(RColorBrewer)

# Load data
file <- "nes00.csv"
data <- read.csv(file, header=TRUE)

# Set up model formula and model specific data frame
model <- vote00 ~ age + I(age^2) + hsdeg + coldeg
mdata <- extractdata(model, data, na.rm=TRUE)
```

Notes:

`extractdata()` from `simcf` simplifies listwise deletion

We keep all data in a dataframe while deleting only based on model variables

Lineplot example

```
# Run logit & extract results
logit.result <- glm(model, family=binomial, data=mdata)
pe <- logit.result$coefficients  # point estimates
vc <- vcov(logit.result)         # var-cov matrix

# Simulate parameter distributions
sims <- 10000
simbetas <- mvrnorm(sims, pe, vc)
```

Notes:

We summarize the model using draws from the predictive distributions of the estimated parameters

This is approximately MVN for all MLEs, regardless of the distribution of the response variable

Lineplot example

```
# Set up counterfactuals:  all ages, each of three educations
xhyp <- seq(18,97,1)
nscen <- length(xhyp)
nohsScen <- hsScen <- collScen <- cfMake(model, mdata, nscen)
```

Notes:

We set up counterfactual levels of the continuous covariate of interest, age

This determines the number of scenarios for our lineplots

We then make counterfactual objects using `simcf`

Each of these objects has a many scenarios as there are counterfactuals of age

We need three sets of these counterfactuals—one for each level of education

Lineplot example

```
for (i in 1:nscen) {
  # No High school scenarios (loop over each age)
  nohsScen <- cfChange(nohsScen, "age", x = xhyp[i], scen = i)
  nohsScen <- cfChange(nohsScen, "hsdeg", x = 0, scen = i)
  nohsScen <- cfChange(nohsScen, "coldeg", x = 0, scen = i)
}
```

Notes:

We loop over each level of age

At each iteration, we need to set the appropriate values of the covariates in our counterfactuals

`cfChange()` takes a counterfactual scenario under construction, the name of a covariate, a new level for the variable, and the number of the scenario to be changed to that level

We apply `cfChange()` repeatedly to setup all our scenarios

We could set in addition `xpre` to do first differences between `x` and `xpre`

Lineplot example

```
for (i in 1:nscen) {
  # No High school scenarios (loop over each age)
  nohsScen <- cfChange(nohsScen, "age", x = xhyp[i], scen = i)
  nohsScen <- cfChange(nohsScen, "hsdeg", x = 0, scen = i)
  nohsScen <- cfChange(nohsScen, "coldeg", x = 0, scen = i)

  # HS grad scenarios (loop over each age)
  hsScen <- cfChange(hsScen, "age", x = xhyp[i], scen = i)
  hsScen <- cfChange(hsScen, "hsdeg", x = 1, scen = i)
  hsScen <- cfChange(hsScen, "coldeg", x = 0, scen = i)

  # College grad scenarios (loop over each age)
  collScen <- cfChange(collScen, "age", x = xhyp[i], scen = i)
  collScen <- cfChange(collScen, "hsdeg", x = 1, scen = i)
  collScen <- cfChange(collScen, "coldeg", x = 1, scen = i)
}
```

Lineplot example

```
# Simulate expected probabilities for all scenarios
nohsSims <- logitsimev(nohsScen, simbetas, ci=0.95)
hsSims <- logitsimev(hsScen, simbetas, ci=0.95)
collSims <- logitsimev(collScen, simbetas, ci=0.95)
```

Notes:

We run each set of counterfactuals through an appropriate simulator from `simcf`

Since we are using a logit model, and want expected probabilities, we use `logitsimev`, which stands for Logit Simulate Expected Values

The output is a list with point estimates `pe`, lower bounds `lower`, and upper bounds `upper`

Lineplot example

```
# Get 3 nice colors for traces
col <- brewer.pal(3,"Dark2")

# Set up lineplot traces of expected probabilities
nohsTrace <- lineplot(x=xhyp,
                      y=nohsSims$pe,
                      lower=nohsSims$lower,
                      upper=nohsSims$upper,
                      col=col[1],
                      extrapolate=list(data=mdata,
                                       cfact=nohsScen$x,
                                       omit.extrapolated=TRUE),
                      plot=1)
```

Notes:

We put the simulated responses, CIs, and corresponding hypothetical ages into a trace

We need one trace for each education level

Lineplot example

```
# Get 3 nice colors for traces
col <- brewer.pal(3,"Dark2")

# Set up lineplot traces of expected probabilities
nohsTrace <- lineplot(x=xhyp,
                      y=nohsSims$pe,
                      lower=nohsSims$lower,
                      upper=nohsSims$upper,
                      col=col[1],
                      extrapolate=list(data=mdata,
                                       cfact=nohsScen$x,
                                       omit.extrapolated=TRUE),
                      plot=1)
```

Notes:

Optionally, we include instructions to omit extrapolations outside the convex hull

Inputs to `extrapolate` are very picky: we need the model data, and the hypothetical data in exactly the same order of columns

Lineplot example

```
hsTrace <- lineplot(x=xhyp,
                    y=hsSims$pe,
                    lower=hsSims$lower,
                    upper=hsSims$upper,
                    col=col[2],
                    extrapolate=list(data=mdata,
                                       cfact=hsScen$x,
                                       omit.extrapolated=TRUE),
                    plot=1)

collTrace <- lineplot(x=xhyp,
                      y=collSims$pe,
                      lower=collSims$lower,
                      upper=collSims$upper,
                      col=col[3],
                      extrapolate=list(data=mdata,
                                         cfact=collScen$x,
                                         omit.extrapolated=TRUE),
                      plot=1)
```

Lineplot example

```
# Set up traces with labels and legend
labelTrace <- textTile(labels=c("Less than HS",
                                "High School",
                                "College"),
                       x=c( 55,    49,     30),
                       y=c( 0.26,  0.56,   0.87),
                       col=col,
                       plot=1)

legendTrace <- textTile(labels=c("Logit estimates:",
                                 "95% confidence",
                                 "interval is shaded"),
                        x=c(82, 82, 82),
                        y=c(0.2, 0.16, 0.12),
                        plot=1)
```

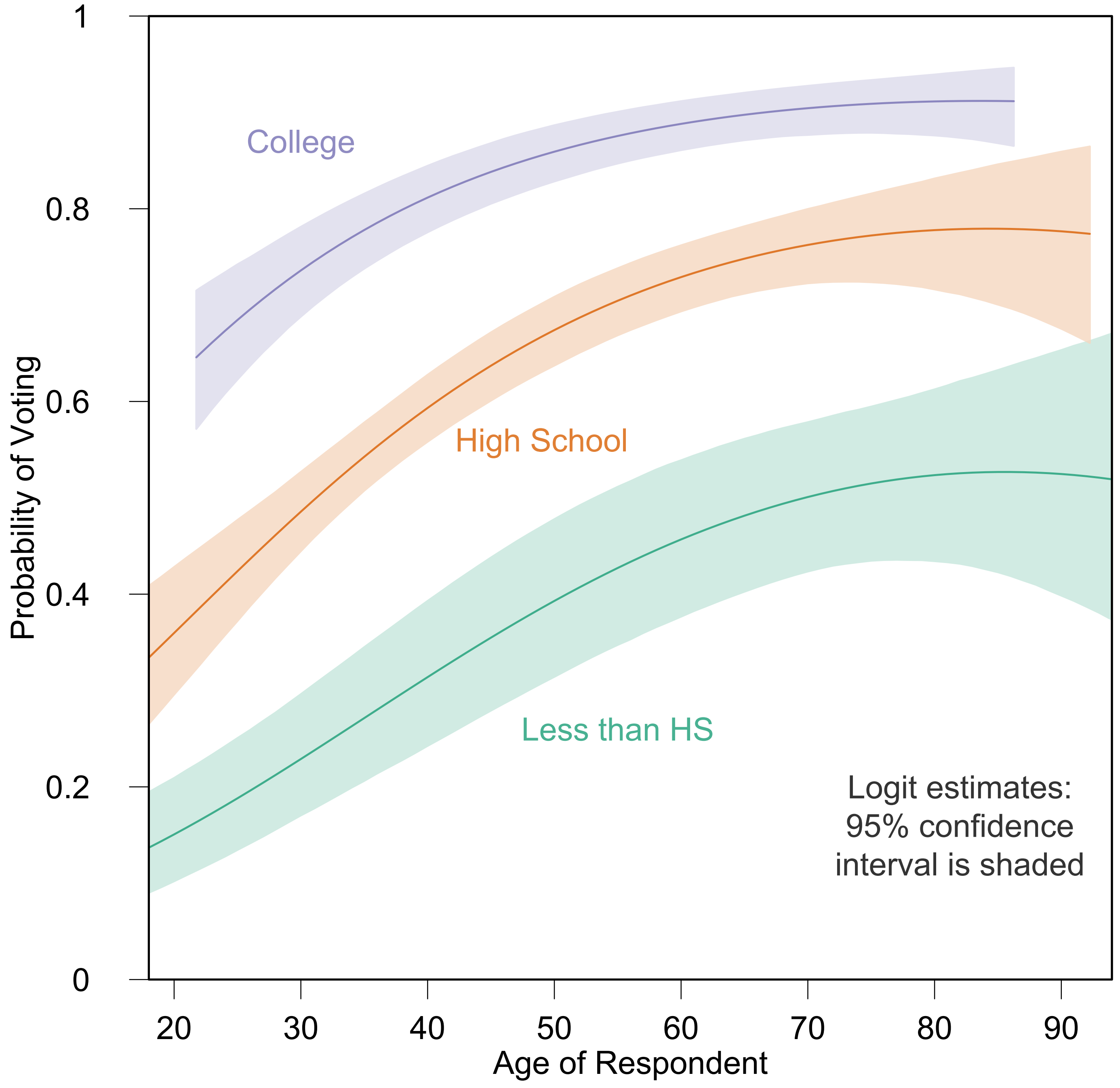
Lineplot example

```
# Plot traces using tile
tile(nohsTrace,
     hsTrace,
     collTrace,
     labelTrace,
     legendTrace,
     width=list(null=5),
     limits=c(18,94,0,1),
     xaxis=list(at=c(20,30,40,50,60,70,80,90)),
     xaxistitle=list(labels="Age of Respondent"),
     yaxistitle=list(labels="Probability of Voting"),
     frame=TRUE
     )
```

Notes:

The `width` argument lets us tweak the layout.

Setting `null=5` stretches the size of each plot in the graphic to 5 inches from a default of 2


1
0.8
0.6
0.4
0.2
0
20
30
40
50
60
70
80
90
Probability of Voting
Age of Respondent
College
High School
Less than HS
Logit estimates:
95% confidence
interval is shaded