CSSS 569: Visualizing Data

Visualizing Robustness

Christopher Adolph*
University of Washington, Seattle

March 4, 2011

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

Robustness Checks

Last time: Presenting conditional expectations & differences from regressions

But are we confident that these were the "right" estimates?

The language of inference usually assumes we

- correctly specified our model
- correctly measured our variables
- chose the right probability model
- e.g., we don't have influential outliers
- etc.

We're never completely sure these assumptions hold.

Most people present one model, and argue it was the best choice

Sometimes, a few alternatives are displayed

The race of the variables

	Model 1	Model 2	Model 3	Model 4	Model 5
My variable of interest, X_1	X.XX (X.XX)	X.XX (X.XX)	X.XX (X.XX)	X.XX (X.XX)	
A control I "need"	X.XX (X.XX)	X.XX (X.XX)	X.XX (X.XX)	X.XX (X.XX)	X.XX (X.XX)
A control I "need"	X.XX (X.XX)	X.XX (X.XX)	X.XX (X.XX)	X.XX (X.XX)	X.XX (X.XX)
A candidate control		X.XX (X.XX)		X.XX (X.XX)	
A candidate control			X.XX (X.XX)	X.XX (X.XX)	
Alternate measure of X_1					X.XX (X.XX)

Robustness Checks

Problems with the approach above?

1. Lots of space to show a few permutations of the model

   Most space wasted or devoted to ancillary info

2. What if we're really interested in $\mathrm{E}(Y|X)$, not $\hat{\beta}$?

   E.g., because of nonlinearities, interactions, scale differences, etc.

3. The selection of permutations is ad hoc.

We'll try to fix 1 & 2.

Objection 3 is harder, but worth thinking about.

Robustness Checks: An algorthm

1. Identify a relation of interest between a concept $\mathcal{X}$ and a concept $\mathcal{Y}$

2. Choose:

   - a measure of $\mathcal{X}$, denoted X,
   - a measure of $\mathcal{Y}$, denoted Y,
   - a set of confounders, Z,
   - a functional form, $g(\cdot)$
   - a probability model of Y, $f(\cdot)$

3. Estimate the probability model $Y \sim f(\mu, \alpha)$, $\mu = g(\text{vec}(X, Z), \beta)$.

4. Simulate the quantity of interest, e.g., $\text{E}(Y|X)$ or $\text{E}(Y_1 - Y_2|X_1 - X_2)$, to obtain a point estimate and confidence interval.

5. Repeat above steps, changing at each iteration one of the choices in step 2.

6. Compile the results in a variant of the dot plot (ropeladder).

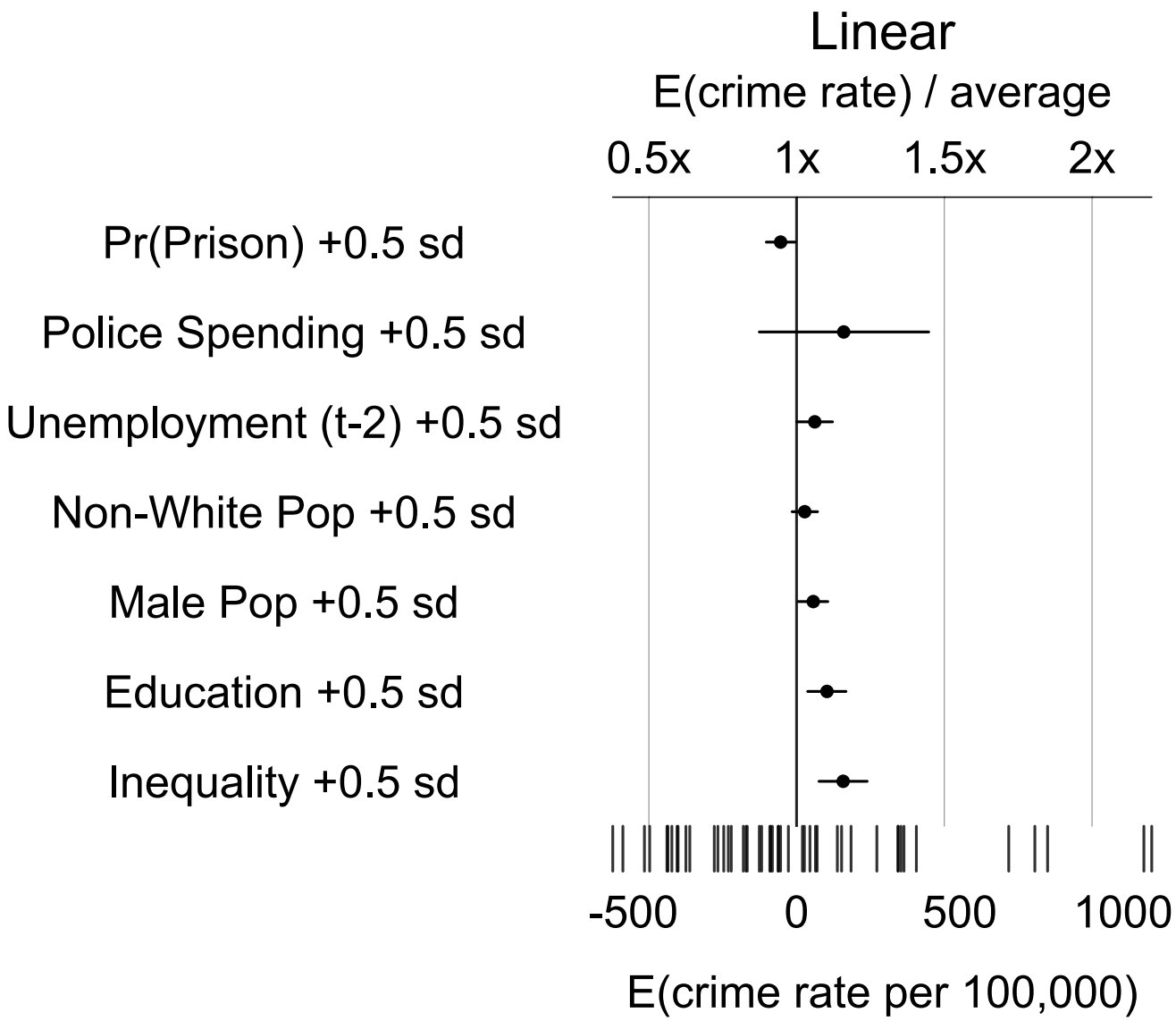
Kitchen sink models of 1960 US crime rates

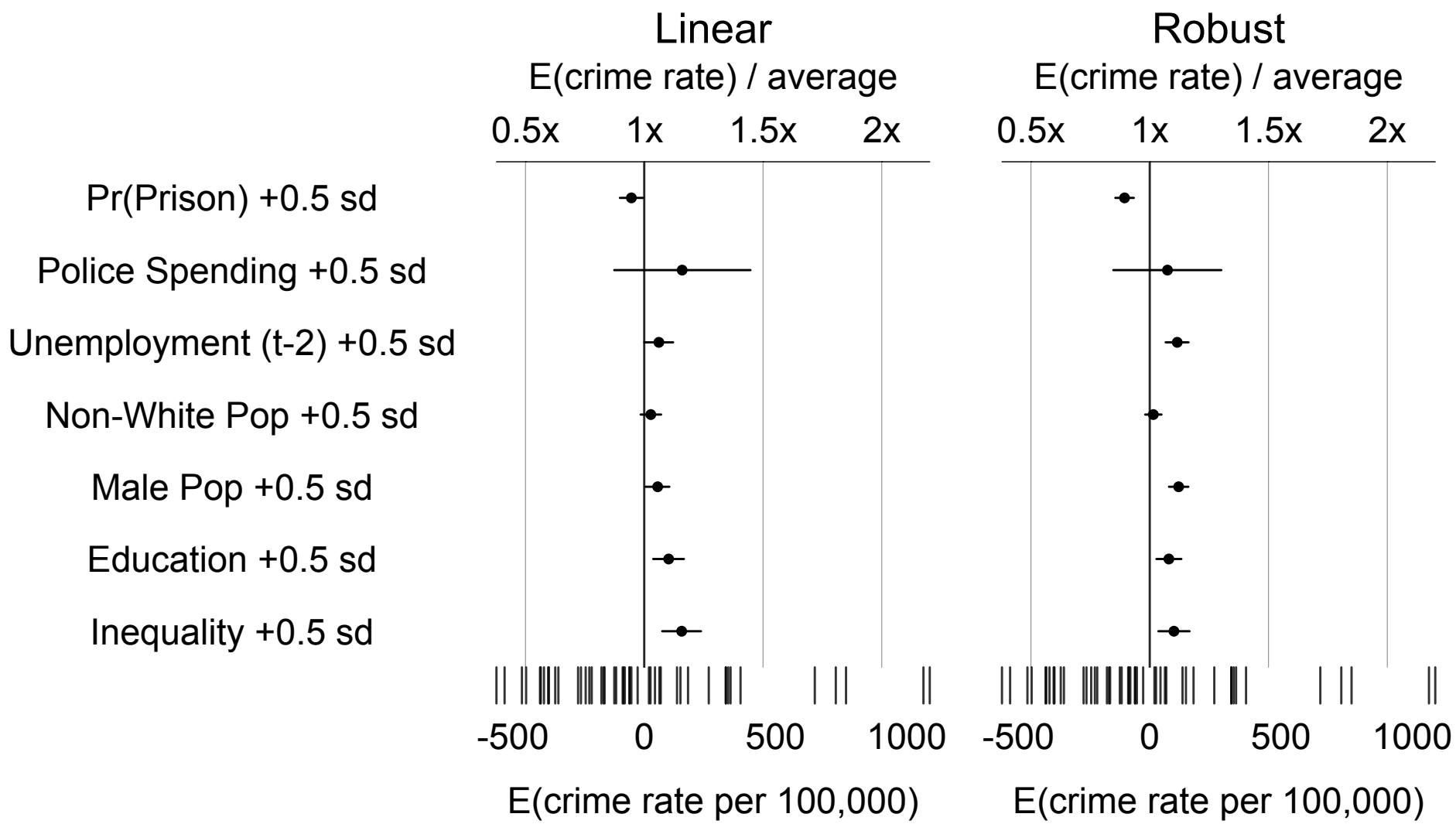
	Linear	Robust	Poisson	Neg Bin
Constant	−28820.91	−17784.56	−19.08	−15.43
	(10199.82)	(8158.71)	(1.77)	(7.81)
% males aged 14–24	1156.49	2480.55	1.1	1.53
	(522.98)	(418.32)	(0.1)	(0.4)
Southern state	0.97	138.11	0.06	0.06
	(141.49)	(113.18)	(0.02)	(0.11)
Mean education (yrs)	1802.64	1413.62	1.84	2.11
	(590.84)	(472.61)	(0.11)	(0.45)
Police spending 1960	897.54	422.45	0.81	0.76
	(813.8)	(650.95)	(0.15)	(0.62)
Police spending 1959	6.66	651.14	0.01	0.01
	(823.35)	(658.59)	(0.15)	(0.63)
Labor participation	143.91	2235.29	0.63	0.62
	(727.79)	(582.15)	(0.13)	(0.56)
Males per 1000	94.71	−3469.7	−1.46	−2.3
	(1943.8)	(1554.82)	(0.36)	(1.49)
State population	−79.39	−138.58	−0.08	−0.07
	(51.4)	(41.12)	(0.01)	(0.04)

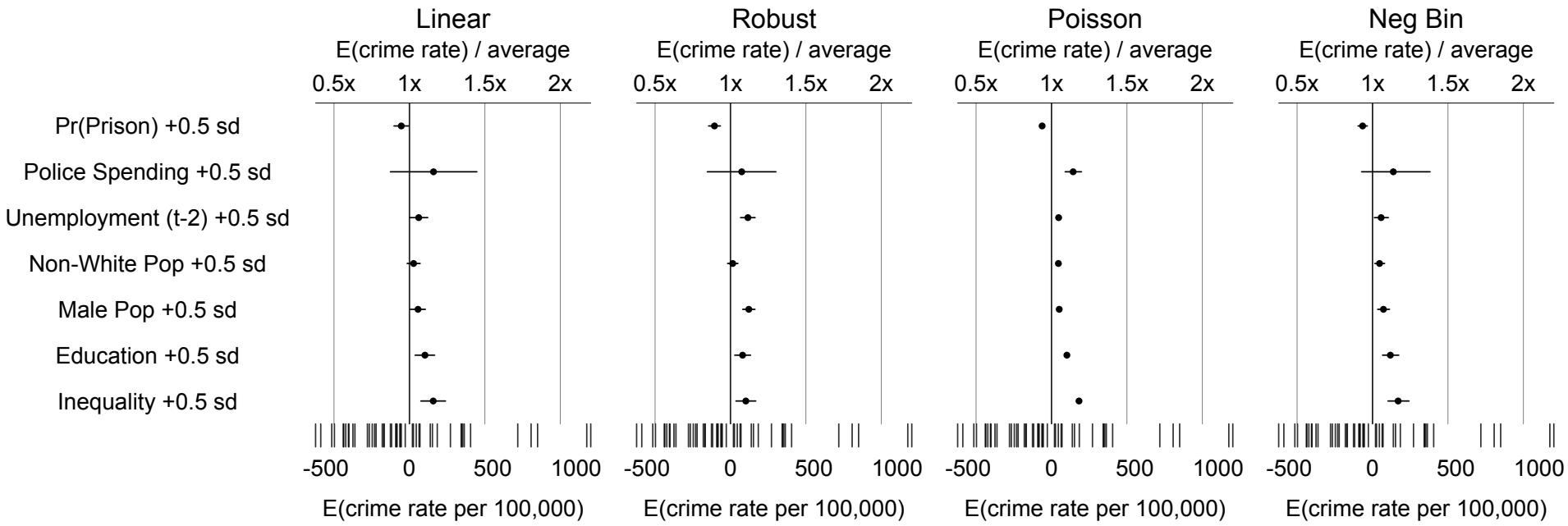
Kitchen sink models of 1960 US crime rates

continued

	Linear	Robust	Poisson	Neg Bin
Nonwhites per 1000	61.25	32.47	0.11	0.11
	(47.85)	(38.28)	(0.01)	(0.04)
Unem, males 14–24	−325.65	−444.95	−0.18	−0.18
	(336.46)	(269.13)	(0.06)	(0.26)
Unem, males 35–39	475.14	895.28	0.39	0.46
	(239.62)	(191.67)	(0.04)	(0.18)
Gross state product, pc	282.31	−196.44	0.69	0.64
	(420.2)	(336.11)	(0.08)	(0.32)
Income inequality	1461.68	943.27	1.68	1.56
	(386.64)	(309.27)	(0.07)	(0.3)
Pr(imprisonment)	−226.39	−443.28	−0.29	−0.31
	(103.39)	(82.7)	(0.02)	(0.08)
E(time in prison)	−69.91	−294.41	−0.16	−0.27
	(184.13)	(147.29)	(0.03)	(0.14)


Linear
E(crime rate) / average
0.5x
1x
1.5x
2x
Pr(Prison) +0.5 sd
Police Spending +0.5 sd
Unemployment (t-2) +0.5 sd
Non-White Pop +0.5 sd
Male Pop +0.5 sd
Education +0.5 sd
Inequality +0.5 sd
-500
0
500
1000
E(crime rate per 100,000)


Linear
E(crime rate) / average
0.5x
1x
1.5x
2x
Pr(Prison) +0.5 sd
Police Spending +0.5 sd
Unemployment (t-2) +0.5 sd
Non-White Pop +0.5 sd
Male Pop +0.5 sd
Education +0.5 sd
Inequality +0.5 sd
-500
0
500
1000
E(crime rate per 100,000)
Robust
E(crime rate) / average
0.5x
1x
1.5x
2x
-500
0
500
1000
E(crime rate per 100,000)


Linear
Robust
Poisson
Neg Bin
E(crime rate) / average
0.5x
1x
1.5x
2x
Pr(Prison) +0.5 sd
Police Spending +0.5 sd
Unemployment (t-2) +0.5 sd
Non-White Pop +0.5 sd
Male Pop +0.5 sd
Education +0.5 sd
Inequality +0.5 sd
-500
0
500
1000
E(crime rate per 100,000)

Ropeladder example

```
# Linear, Poisson, and Negative Binomial regression using UScrime data

# Uses ropeladders to show how the expected crime rate varies in
# response to changes in 7 covariates under each of four estimation
# methods.

# Plot 1 shows a four plot set up, one plot per method.
# This approach highlights differences in effects across covariates

# Plot 2 squeezes all four ropeladders into a single plot.
# This approach gives equal attention to differences
# across covariates and models

# Plot 3 creates a plot for each covariate.
# This approach highlights differences across models.
```

Ropeladder example

```
# Load data and libraries; set up specification
library(tile)
library(simcf)
library(MASS)
data(UScrime)
model <- (y ~ log(M) + So + log(Ed) + log(Po1) + log(Po2)
              + log(LF) + log(M.F) + log(Pop) + log(NW) +log(U1)
              + log(U2) + log(GDP) + log(Ineq) + log(Prob) +
              log(Time))
```

Ropeladder example

```
# Estimate Linear regression model
lm1.res <- lm(model, data = UScrime)
lm1.pe <- lm1.res$coefficients        # point estimates
lm1.vc <- vcov(lm1.res)               # var-cov matrix

# Estimate Robust and resistant regression model
mm1.res <- rlm(model, data = UScrime, method="MM")
mm1.pe <- mm1.res$coefficients        # point estimates
mm1.vc <- vcov(mm1.res)               # var-cov matrix

# Estimate Poisson model
po1.res <- glm(model, family=poisson, data = UScrime)
po1.pe <- po1.res$coefficients         # point estimates
po1.vc <- vcov(po1.res)                # var-cov matrix

# Estimate Negative Binomial model
nb1.res <- glm.nb(model, data = UScrime)
nb1.pe <- nb1.res$coefficients         # point estimates
nb1.vc <- vcov(nb1.res)                # var-cov matrix
```

Ropeladder example

```
# Initialize 7 different scenarios to mean values of covariates
xscen <- cfMake(model, data=UScrime, nscen=7)

# Configure scenario 1:  Raise Probability of Imprisonment by 1/2 sd
xscen <- cfName(xscen, "Pr(Prison) +0.5 sd", scen=1)
xscen <- cfChange(xscen, "Prob",
                  x = mean(UScrime$Prob) + 0.5*sd(UScrime$Prob),
                  scen=1)
```

Notes:

Unlike the lineplot example, no longer looping over a continuous covariate

Now considering discrete changes in a series of covariates, each time holding others constant at their means

Result similar to table of linear regression coefficients (partial derivaratives)

`cfName()` lets us name each scenario; we'll use these later to label the plot

Ropeladder example

```
# Configure scenario 2:  Raise Police Spending by 1/2 sd
xscen <- cfName(xscen, "Police Spending +0.5 sd", scen=2)
xscen <- cfChange(xscen, "Po1",
                  x = mean(UScrime$Po1) + 0.5*sd(UScrime$Po1),
                  scen=2)

# Configure scenario 3:  Raise Unemployment (Age 35-39) by 1/2 sd
xscen <- cfName(xscen, "Unemployment (t-2) +0.5 sd", scen=3)
xscen <- cfChange(xscen, "U2",
                  x = mean(UScrime$U2) + 0.5*sd(UScrime$U2),
                  scen=3)

# Configure scenario 4:  Raise Non-white population by 1/2 sd
xscen <- cfName(xscen, "Non-White Pop +0.5 sd", scen=4)
xscen <- cfChange(xscen, "NW",
                  x = mean(UScrime$NW) + 0.5*sd(UScrime$NW),
                  scen=4)
```

Ropeladder example

```
# Configure scenario 5:  Raise Male Pop by 1/2 sd
xscen <- cfName(xscen, "Male Pop +0.5 sd", scen=5)
xscen <- cfChange(xscen, "M",
                  x = mean(UScrime$M) + 0.5*sd(UScrime$M),
                  scen=5)

# Configure scenario 6:  Raise Education by 1/2 sd
xscen <- cfName(xscen, "Education +0.5 sd", scen=6)
xscen <- cfChange(xscen, "Ed",
                  x = mean(UScrime$Ed) + 0.5*sd(UScrime$Ed),
                  scen=6)

# Configure scenario 7:  Raise Inequality by 1/2 sd
xscen <- cfName(xscen, "Inequality +0.5 sd", scen=7)
xscen <- cfChange(xscen, "Ineq",
                  x = mean(UScrime$Ineq) + 0.5*sd(UScrime$Ineq),
                  scen=7)
```

Ropeladder example

```
# Simulate conditional expectations for these counterfactuals
sims <- 10000

# Linear regression simulations
simbetas.lm <- mvrnorm(sims, lm1.pe, lm1.vc)
lm1.qoi <- linearsimfd(xscen, simbetas.lm, ci=0.95)

# Robust regression simulations
simbetas.mm <- mvrnorm(sims, mm1.pe, mm1.vc)
mm1.qoi <- linearsimfd(xscen, simbetas.mm, ci=0.95)

# Poisson simulations
simbetas.po <- mvrnorm(sims, po1.pe, po1.vc)
po1.qoi <- loglinsimfd(xscen, simbetas.po, ci=0.95)

# Negative Binomial simulations
simbetas.nb <- mvrnorm(sims, nb1.pe, nb1.vc)
nb1.qoi <- loglinsimfd(xscen, simbetas.nb, ci=0.95)
```

Ropeladder example

```
# Create ropeladder traces of first differences from each model
trace1 <- ropeladder(x=lm1.qoi$pe,
                     lower=lm1.qoi$lower,
                     upper=lm1.qoi$upper,
                     labels=row.names(xscen$x),
                     plot=1
                     )
```

Notes:

Ropeladder trace needs 4 inputs:

The points themselves, the lower bound, the upper bound, and a label for each point

Often best to sort these based on the size of the point estimate (clearest comparisons)

Ropeladder example

```
trace2 <- ropeladder(x=mm1.qoi$pe,
                     lower=mm1.qoi$lower,
                     upper=mm1.qoi$upper,
                     plot=2
                     )

trace3 <- ropeladder(x=po1.qoi$pe,
                     lower=po1.qoi$lower,
                     upper=po1.qoi$upper,
                     plot=3
                     )

trace4 <- ropeladder(x=nb1.qoi$pe,
                     lower=nb1.qoi$lower,
                     upper=nb1.qoi$upper,
                     plot=4
                     )
```

Note: Can leave out labels argument to save space in adjacent ropeladders

Make sure they line up with the same scenarios though!

Ropeladder example

```
rug1 <- rugTile(x = UScrime$y - mean(UScrime$y),
                plot = 1:4
                )

vertmark <- linesTile(x = c(0,0),
                      y = c(0,1),
                      lty = "solid",
                      plot = 1:4
                      )
```

Note:

We don't really need a rug here. Might be best to not show it.

A vertical reference line, though, is usually very helpful.

For first diffs, a line at 0 is best.

For expected values, a line at the mean is best.

Ropeladder example

```
# Create Plot 1 (focus on covariates) and save to pdf
tc <- tile(trace1, trace2, trace3, trace4,
           rug1, vertmark,
           #output = list(file = "ropeladderEx1"),
           xaxistitle = list(labels="E(crime rate per 100,000)"),
           topaxis= list(at = mean(UScrime$y)*c(0.5, 1, 1.5, 2)
                           - mean(UScrime$y),
                         labels = c("0.5x","1x","1.5x","2x"),
                         add = rep(TRUE,4)
                         ),
           topaxistitle = list(labels="E(crime rate) / average"),
           plottitle = list(labels1 = "Linear",
                            labels2 = "Robust",
                            labels3 = "Poisson",
                            labels4 = "Neg Bin"),
           gridlines=list(type="t")
           )
```

Ropeladder example

What's going on with this?

```
topaxis= list(at = mean(UScrime$y)*c(0.5, 1, 1.5, 2)
                    - mean(UScrime$y),
              labels = c("0.5x","1x","1.5x","2x"),
              add = rep(TRUE,4)
              ),
```

We plot the absolute change in crime rates on the x-axis

Want the *relative* change on the top-axis

We want to mark where our FDs have halved the crime rate, increased it by 50%, or doubled it.

Ropeladder example

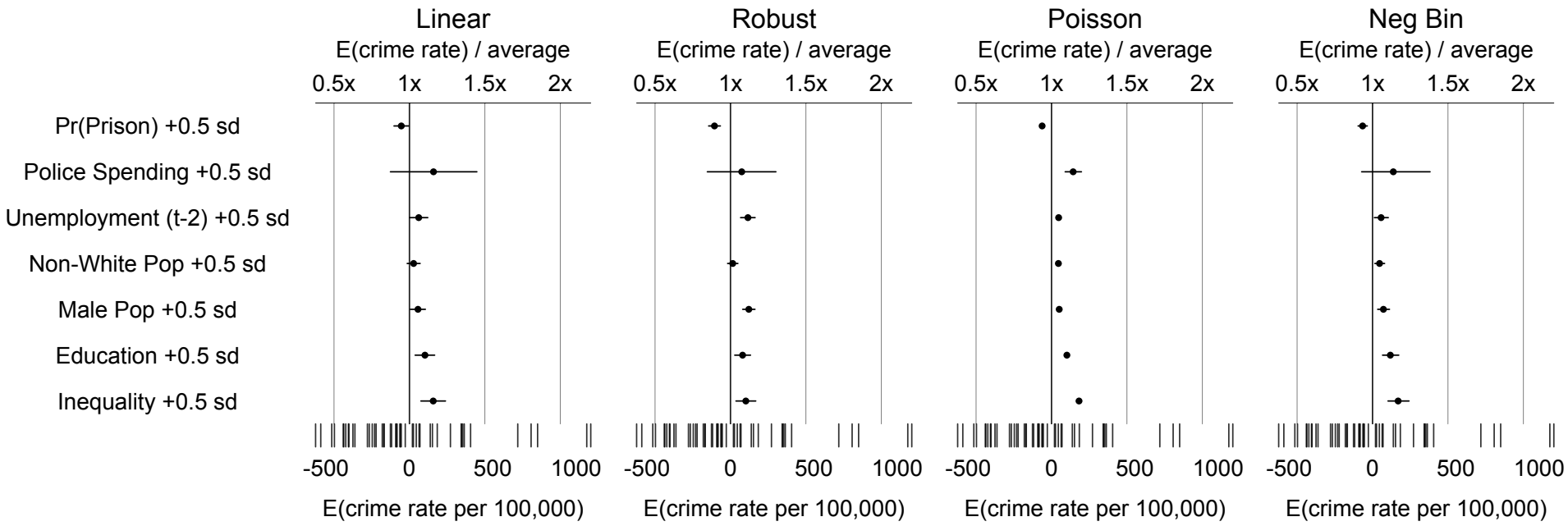
What's going on with this?

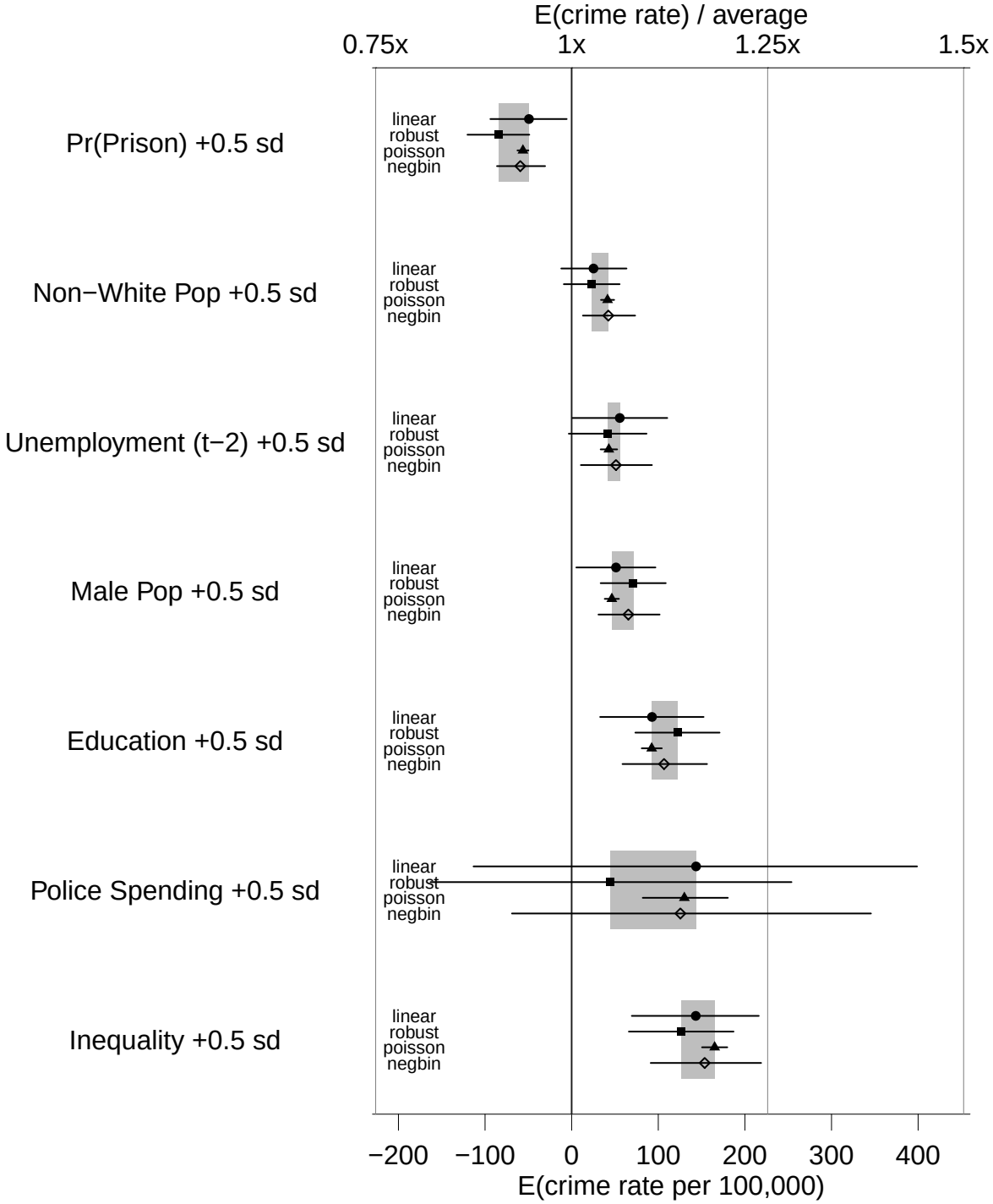
```
topaxis= list(at = mean(UScrime$y)*c(0.5, 1, 1.5, 2)
                    - mean(UScrime$y),
              labels = c("0.5x","1x","1.5x","2x"),
              add = rep(TRUE,4)
              ),
```

We make labels for these points, but where to put them?

The levels of crime corresponding to half, the same, +50%, and double the mean are: `mean(UScrime$y)*c(0.5, 1, 1.5, 2)`

But our x-axis is on a first differenced scale, so we need to subtract off the expected value of crime for the average case: `-mean(UScrime$y)`


Linear
Robust
Poisson
Neg Bin
E(crime rate) / average
0.5x
1x
1.5x
2x
Pr(Prison) +0.5 sd
Police Spending +0.5 sd
Unemployment (t-2) +0.5 sd
Non-White Pop +0.5 sd
Male Pop +0.5 sd
Education +0.5 sd
Inequality +0.5 sd
-500
0
500
1000
E(crime rate per 100,000)


E(crime rate) / average
0.75x
1x
1.25x
1.5x
Pr(Prison) +0.5 sd
Non−White Pop +0.5 sd
Unemployment (t−2) +0.5 sd
Male Pop +0.5 sd
Education +0.5 sd
Police Spending +0.5 sd
Inequality +0.5 sd
linear
robust
poisson
negbin
−200
−100
0
100
200
300
400
E(crime rate per 100,000)

Ropeladder example

```
# Plot all four models together:
# equal focus on covariates and models

# Revise traces to place on same plot
trace1$plot <- trace2$plot <- trace3$plot <- trace4$plot <- 1
vertmark$plot <- 1

# Revise traces to make symbols different
trace1$pch <- 19
trace2$pch <- 15
trace3$pch <- 17
trace4$pch <- 23
```

Note: Tricky re-use and modification of traces. Just changing a few elements of each trace

Ropeladder example

```
# Add sublabels to each trace
trace1$sublabels <- "linear"
trace2$sublabels <- "robust"
trace3$sublabels <- "poisson"
trace4$sublabels <- "negbin"

# Widen space between entries to make labels visible
trace1$entryheight <- 0.25

# Shift sublabels to left side of plot to avoid overlap
trace1$sublabelsX <- 0.07
trace2$sublabelsX <- 0.07
trace3$sublabelsX <- 0.07
trace4$sublabelsX <- 0.07
```

Note: Tricky re-use and modification of traces. Just changing a few elements of each trace

Ropeladder example

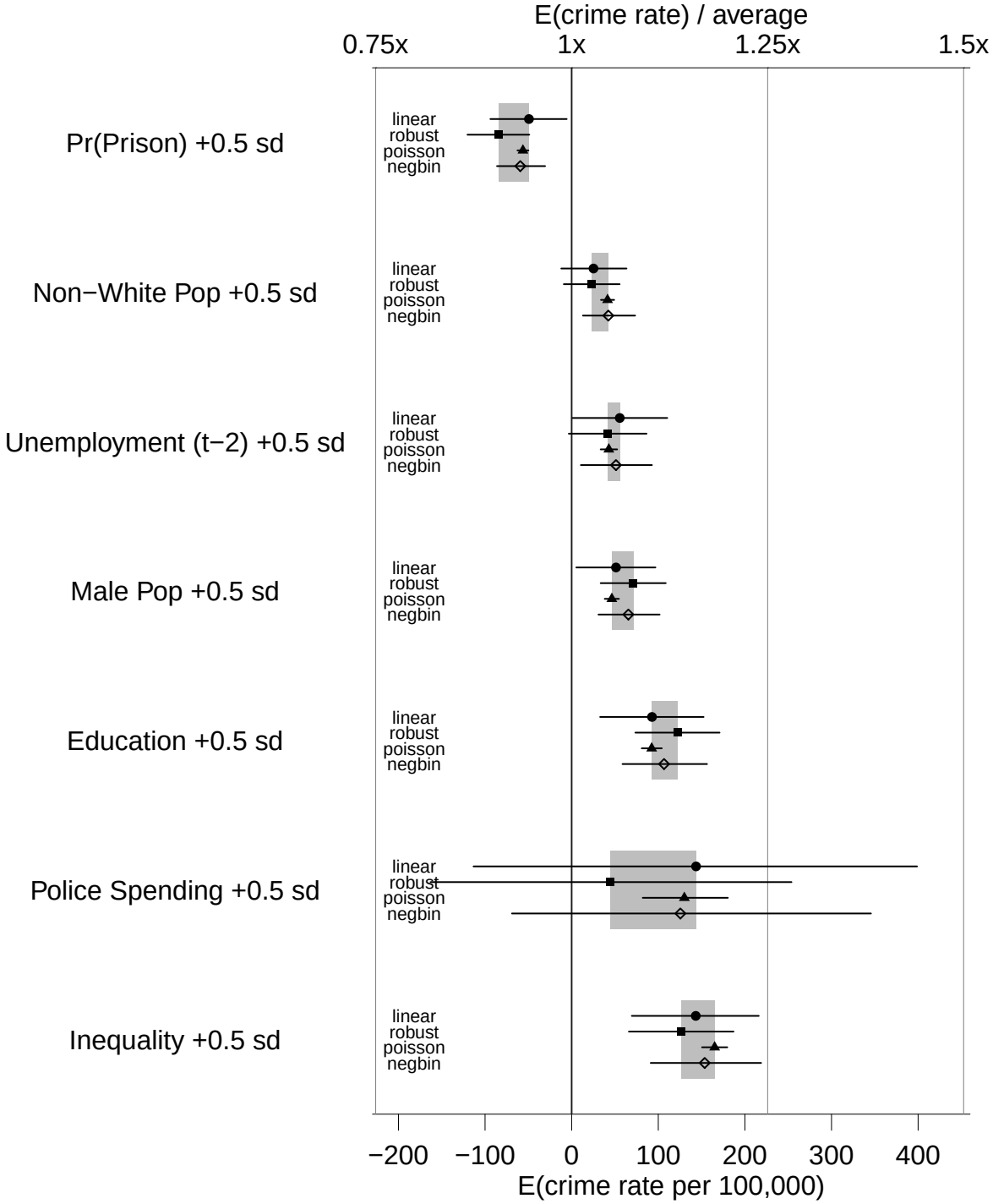
```
# Add boxes around the results for each covariate
# when traces are plotted to the same graph
# (could add to any of the traces)
trace1$shadowrow <- TRUE
```

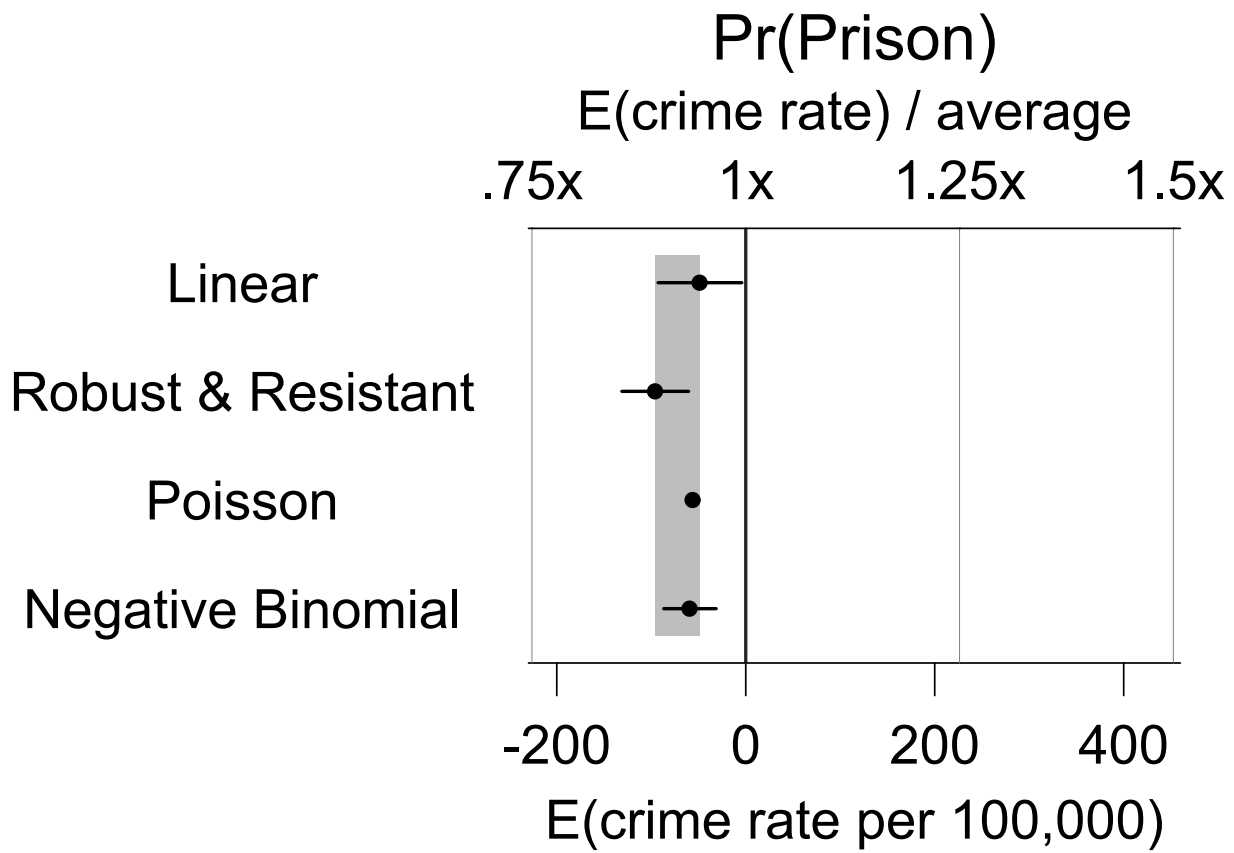
Note: Tricky re-use and modification of traces. Just changing a few elements of each trace

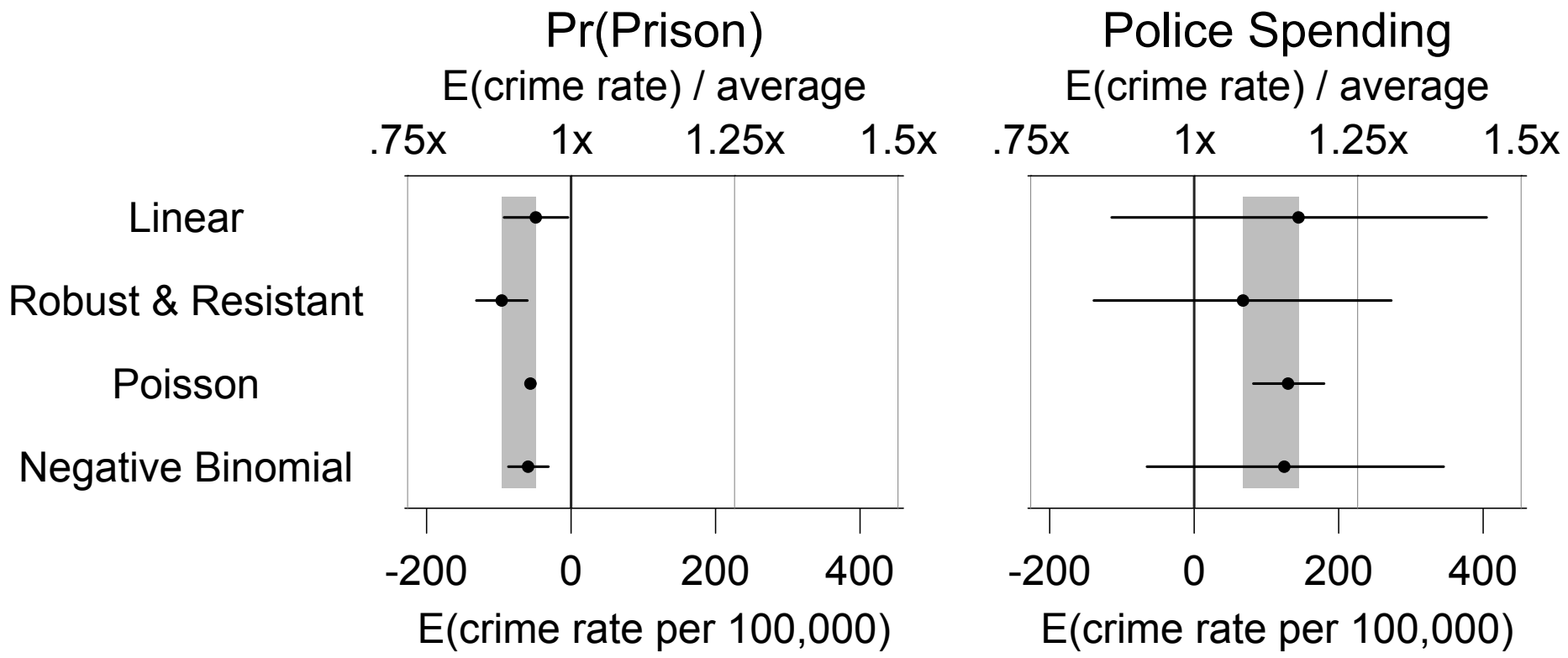
Ropeladder example

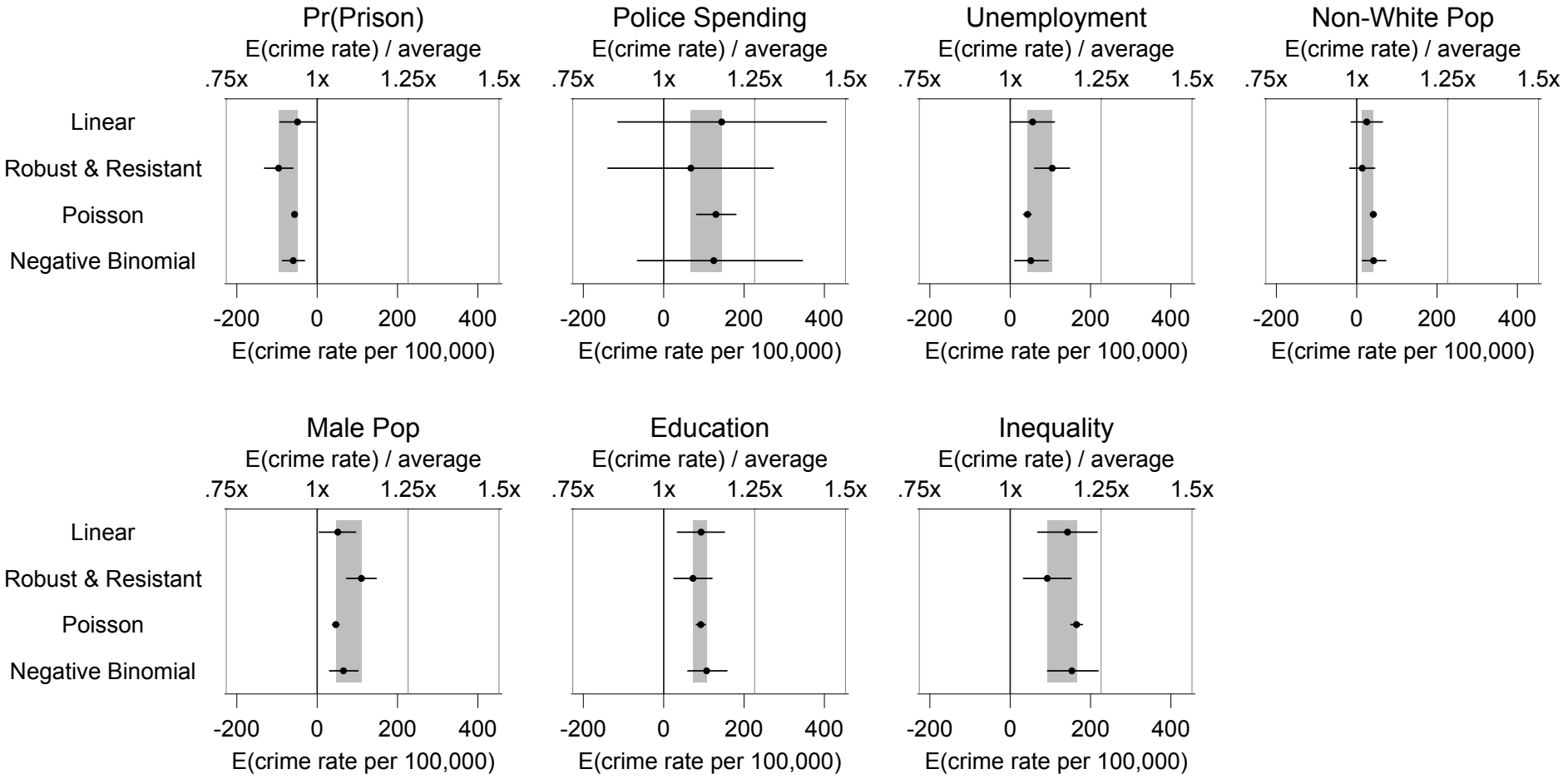
```
# Create Plot 2 and save to pdf
tc <- tile(trace1, trace2, trace3, trace4,
           vertmark,
           limits = c(-230, 460),
           width=list(null=4),
           #output = list(file="ropeladderEx2"),
           xaxistitle = list(labels="E(crime rate per 100,000)"),
           topaxis= list(at = mean(UScrime$y)*c(0.75, 1, 1.25, 1.5)
                                - mean(UScrime$y),
                         labels = c("0.75x","1x","1.25x","1.5x"),
                         add = TRUE),
           topaxistitle = list(labels="E(crime rate) / average"),
           gridlines=list(type="t")
           )
```

Note new `limits` argument. Got rid of the rugs, so need to set (common) limits on plot(s) now.


E(crime rate) / average
0.75x
1x
1.25x
1.5x
Pr(Prison) +0.5 sd
Non−White Pop +0.5 sd
Unemployment (t−2) +0.5 sd
Male Pop +0.5 sd
Education +0.5 sd
Police Spending +0.5 sd
Inequality +0.5 sd
linear
robust
poisson
negbin
−200
−100
0
100
200
300
400
E(crime rate per 100,000)


Pr(Prison)
E(crime rate) / average
.75x
1x
1.25x
1.5x
Linear
Robust & Resistant
Poisson
Negative Binomial
-200
0
200
400
E(crime rate per 100,000)


Pr(Prison)
E(crime rate) / average
.75x
1x
1.25x
1.5x
Linear
Robust & Resistant
Poisson
Negative Binomial
-200
0
200
400
E(crime rate per 100,000)
Police Spending
E(crime rate) / average
.75x
1x
1.25x
1.5x
-200
0
200
400
E(crime rate per 100,000)


Pr(Prison)
Police Spending
Unemployment
Non-White Pop
Male Pop
Education
Inequality
E(crime rate) / average
.75x
1x
1.25x
1.5x
Linear
Robust & Resistant
Poisson
Negative Binomial
-200
0
200
400
E(crime rate per 100,000)

Anatomy of a ropeladder plot

I call this a **ropeladder** plot.

The column of dots shows the relationship between Y and a specific X under different model assumptions

Each entry corresponds to a different assumption about the specification, or the measures, or the estimation method, etc.

If all the dots line up, with narrow, similar CIs, we say the finding is robust, and reflects the data under a range of reasonable assumptions

If the ropeladder is "blowing in the wind", we may be skeptical of the finding. It depends on model assumptions that may be controversial

The shaded gray box shows the full range of the point estimates for the QoI.

Narrow is better.

Why ropeladders?

1. Anticipate objections on model assumptions, and have concrete answers.

   Avoid: "I ran it that other way, and it came out the 'same'."

   Instead: "I ran it that other way, and look
   —it made no *substantive* or *statistical* difference worth speaking of."

   Or: ". . . it makes *this* much difference."

2. Investigate robustness more thoroughly.

   Traditional tabular presentation can run to many pages, making comparison hard.

   Without a compact graphical tool in mind, one might stop specification searches too early

Ropeladder example

```
# Plot by coefficient to show variation across models

# Collect in matrix form all first differences and confidence
# intervals across models (columns) and covariates (rows)
allPE <- cbind(lm1.qoi$pe, mm1.qoi$pe, po1.qoi$pe, nb1.qoi$pe)
allLOWER <- cbind(lm1.qoi$lower,
                  mm1.qoi$lower,
                  po1.qoi$lower,
                  nb1.qoi$lower)
allUPPER <- cbind(lm1.qoi$upper,
                  mm1.qoi$upper,
                  po1.qoi$upper,
                  nb1.qoi$upper)
```

Ropeladder example

```
# Create a trace for each covariate of the
# different models' estimates
# (Save these traces in a vector of traces;
#  note double bracket indexing)

collectedtraces <- vector("list", nrow(allPE))

for (i in 1: nrow(allPE)) {
  collectedtraces[[i]] <- ropeladder(x = allPE[i,],
                                     lower = allLOWER[i,],
                                     upper = allUPPER[i,],
                                     shadowbox = TRUE,
                                     plot = i
                                     )
  }
```

Pay close attention to this code: Making a *vector* of lists.

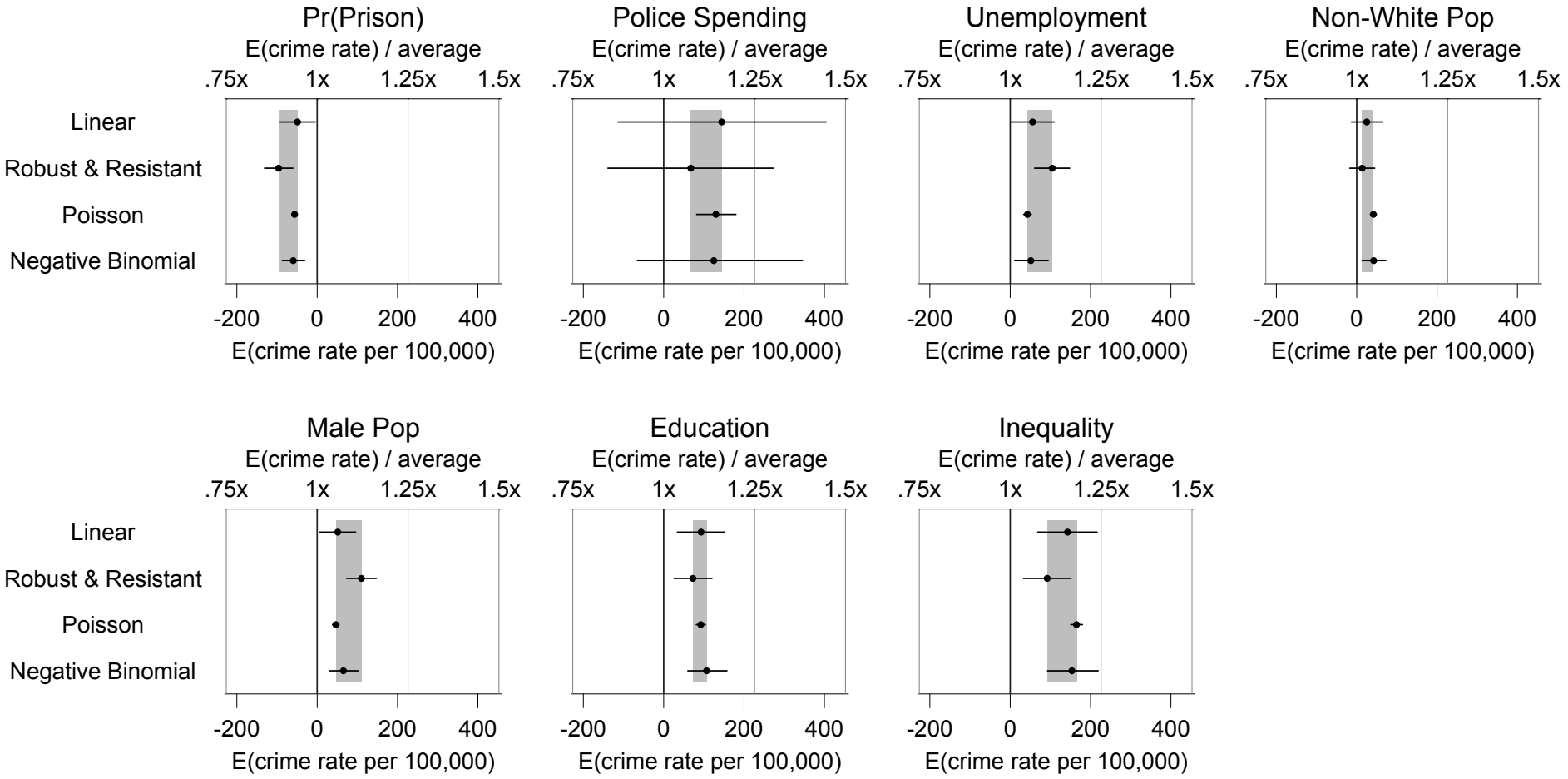
Ropeladder example

```
# Add ropeladder labels to first and fifth plots
# (The first ropeladder in each row of plots;
#  note double bracket indexing)
collectedtraces[[1]]$labels <-
  collectedtraces[[5]]$labels <-
    c("Linear", "Robust & Resistant",
      "Poisson", "Negative Binomial")

# Revise vertical mark to plot on all seven plots
vertmark$plot <- 1:7
```

Ropeladder example

```
tc <- tile(collectedtraces, vertmark,
           RxC = c(2,4),
           limits = c(-230, 460),
           width = list(spacer=3),
           #output = list(file="ropeladderEx3"),
           xaxis = list(at = c(-200, 0, 200, 400)),
           xaxistitle = list(labels="E(crime rate per 100,000)"),
           topaxis= list(at = mean(UScrime$y)*c(0.75, 1, 1.25, 1.5)
                             - mean(UScrime$y),
                         labels = c(".75x","1x","1.25x","1.5x"),
                         add = rep(TRUE,4)),
           topaxistitle = list(labels="E(crime rate) / average"),
           plottitle = list(labels1 = "Pr(Prison)",
                            labels2 = "Police Spending",
                            labels3 = "Unemployment",
                            labels4 = "Non-White Pop",
                            labels5 = "Male Pop",
                            labels6 = "Education",
                            labels7 = "Inequality"),
           gridlines=list(type="top"))
```


Pr(Prison)
Police Spending
Unemployment
Non-White Pop
Male Pop
Education
Inequality
E(crime rate) / average
.75x
1x
1.25x
1.5x
Linear
Robust & Resistant
Poisson
Negative Binomial
-200
0
200
400
E(crime rate per 100,000)