CSSS 569 Visualizing Data and Models

Lab 5: Intro to `tile`

Ramses Llobet

Department of Political Science, UW

February 4, 2022

Introduction

- Overview of `tile`

Introduction

- Overview of `tile`
- Preview of three examples

Introduction

- Overview of `tile`
- Preview of three examples
  - Scatterplot: HW1 example

Introduction

- Overview of `tile`
- Preview of three examples
  - Scatterplot: HW1 example
  - Expected probabilities and first differences: Voting example

Introduction

- Overview of `tile`
- Preview of three examples
  - Scatterplot: HW1 example
  - Expected probabilities and first differences: Voting example
  - Ropeladder: Crime example

Introduction

- Overview of `tile`
- Preview of three examples
  - Scatterplot: HW1 example
  - Expected probabilities and first differences: Voting example
  - Ropeladder: Crime example
- Installing `tile` and `simcf`

Introduction

- Overview of `tile`
- Preview of three examples
  - Scatterplot: HW1 example
  - Expected probabilities and first differences: Voting example
  - Ropeladder: Crime example
- Installing `tile` and `simcf`
- Walking through examples

Overview of `tile`

- A fully featured R graphics package built on the `grid` graphics environment

Overview of `tile`

- A fully featured R graphics package built on the `grid` graphics environment
- Features:

Overview of `tile`

- A fully featured R graphics package built on the `grid` graphics environment
- Features:
  - Make standard displays like scatterplots, lineplots, and dotplots

Overview of `tile`

- A fully featured R graphics package built on the `grid` graphics environment
- Features:
  - Make standard displays like scatterplots, lineplots, and dotplots
  - Create more experimental formats like ropeladders

Overview of `tile`

- A fully featured R graphics package built on the `grid` graphics environment
- Features:
  - Make standard displays like scatterplots, lineplots, and dotplots
  - Create more experimental formats like ropeladders
  - Summarize uncertainty in inferences from model

Overview of `tile`

- A fully featured R graphics package built on the `grid` graphics environment
- Features:
  - Make standard displays like scatterplots, lineplots, and dotplots
  - Create more experimental formats like ropeladders
  - Summarize uncertainty in inferences from model
  - Avoid extrapolation from the original data underlying your model

Overview of `tile`

- A fully featured R graphics package built on the `grid` graphics environment
- Features:
  - Make standard displays like scatterplots, lineplots, and dotplots
  - Create more experimental formats like ropeladders
  - Summarize uncertainty in inferences from model
  - Avoid extrapolation from the original data underlying your model
  - Fully control titles, annotation, and layering of graphical elements

Overview of `tile`

- A fully featured R graphics package built on the `grid` graphics environment
- Features:
  - Make standard displays like scatterplots, lineplots, and dotplots
  - Create more experimental formats like ropeladders
  - Summarize uncertainty in inferences from model
  - Avoid extrapolation from the original data underlying your model
  - Fully control titles, annotation, and layering of graphical elements
  - Build your own tiled graphics from primitives

Overview of `tile`

- A fully featured R graphics package built on the `grid` graphics environment
- Features:
  - Make standard displays like scatterplots, lineplots, and dotplots
  - Create more experimental formats like ropeladders
  - Summarize uncertainty in inferences from model
  - Avoid extrapolation from the original data underlying your model
  - Fully control titles, annotation, and layering of graphical elements
  - Build your own tiled graphics from primitives
- Work well in combination with `simcf` package

Overview of `tile`

- A fully featured R graphics package built on the `grid` graphics environment
- Features:
  - Make standard displays like scatterplots, lineplots, and dotplots
  - Create more experimental formats like ropeladders
  - Summarize uncertainty in inferences from model
  - Avoid extrapolation from the original data underlying your model
  - Fully control titles, annotation, and layering of graphical elements
  - Build your own tiled graphics from primitives
- Work well in combination with `simcf` package
  - Calculate counterfactual expected values, first differences, and relative risks, and their confidence intervals

Overview of `tile`

- A fully featured R graphics package built on the `grid` graphics environment
- Features:
  - Make standard displays like scatterplots, lineplots, and dotplots
  - Create more experimental formats like ropeladders
  - Summarize uncertainty in inferences from model
  - Avoid extrapolation from the original data underlying your model
  - Fully control titles, annotation, and layering of graphical elements
  - Build your own tiled graphics from primitives
- Work well in combination with `simcf` package
  - Calculate counterfactual expected values, first differences, and relative risks, and their confidence intervals
  - More later

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data traces**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data traces**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
     - Could be a set of points, or text labels, or lines, or a polygon

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data traces**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
     - Could be a set of points, or text labels, or lines, or a polygon
     - Could be a set of points and symbols, colors, labels, fit line, CIs, and/or extrapolation limits

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data traces**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
     - Could be a set of points, or text labels, or lines, or a polygon
     - Could be a set of points and symbols, colors, labels, fit line, CIs, and/or extrapolation limits
     - Could be the data for a dotchart, with labels for each line

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data traces**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
     - Could be a set of points, or text labels, or lines, or a polygon
     - Could be a set of points and symbols, colors, labels, fit line, CIs, and/or extrapolation limits
     - Could be the data for a dotchart, with labels for each line
     - Could be the marginal data for a rug

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data traces**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
     - Could be a set of points, or text labels, or lines, or a polygon
     - Could be a set of points and symbols, colors, labels, fit line, CIs, and/or extrapolation limits
     - Could be the data for a dotchart, with labels for each line
     - Could be the marginal data for a rug
     - All annotation must happen in this step

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data traces**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
     - Could be a set of points, or text labels, or lines, or a polygon
     - Could be a set of points and symbols, colors, labels, fit line, CIs, and/or extrapolation limits
     - Could be the data for a dotchart, with labels for each line
     - Could be the marginal data for a rug
     - All annotation must happen in this step
     - Basic traces: `linesTile()`, `pointsile()`, `polygonTile()`, `polylinesTile()`, and `textTile()`

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data traces**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
     - Could be a set of points, or text labels, or lines, or a polygon
     - Could be a set of points and symbols, colors, labels, fit line, CIs, and/or extrapolation limits
     - Could be the data for a dotchart, with labels for each line
     - Could be the marginal data for a rug
     - All annotation must happen in this step
     - Basic traces: `linesTile()`, `pointsile()`, `polygonTile()`, `polylinesTile()`, and `textTile()`
     - Complex traces: `lineplot()`, `scatter()`, `ropeladder()`, and `rugTile()`

Overview of `tile`

- Primitive trace functions:
  - `linesTile()`: Plot a set of connected line segments
  - `pointsTile()`: Plot a set of points
  - `polygonTile()`: Plot a shaded region
  - `polylinesTile()`: Plot a set of unconnected line segments
  - `textTile()`: Plot text labels
- Complex traces for model or data exploration:
  - `lineplot()`: Plot lines with confidence intervals, extrapolation warnings
  - `ropeladder()`: Plot dotplots with confidence intervals, extrapolation warnings, and shaded ranges
  - `rugTile()`: Plot marginal data rugs to axes of plots
  - `scatter()`: Plot scatterplots with text and symbol markers, fit lines, and confidence intervals

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data trace**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data trace**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
  2. **Plot the data traces**: Using the `tile()` function, simultaneously plot all traces to all plots

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data trace**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
  2. **Plot the data traces**: Using the `tile()` function, simultaneously plot all traces to all plots
     - This is the step where the scaffolding gets made: axes and titles

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data trace**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
  2. **Plot the data traces**: Using the `tile()` function, simultaneously plot all traces to all plots
     - This is the step where the scaffolding gets made: axes and titles
     - Set up the rows and columns of plots

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data trace**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
  2. **Plot the data traces**: Using the `tile()` function, simultaneously plot all traces to all plots
     - This is the step where the scaffolding gets made: axes and titles
     - Set up the rows and columns of plots
     - Titles of plots, axes, rows of plots, columns of plots, etc.

Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data trace**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
  2. **Plot the data traces**: Using the `tile()` function, simultaneously plot all traces to all plots
     - This is the step where the scaffolding gets made: axes and titles
     - Set up the rows and columns of plots
     - Titles of plots, axes, rows of plots, columns of plots, etc.
     - Set up axis limits, ticks, tick labels, logging of axes

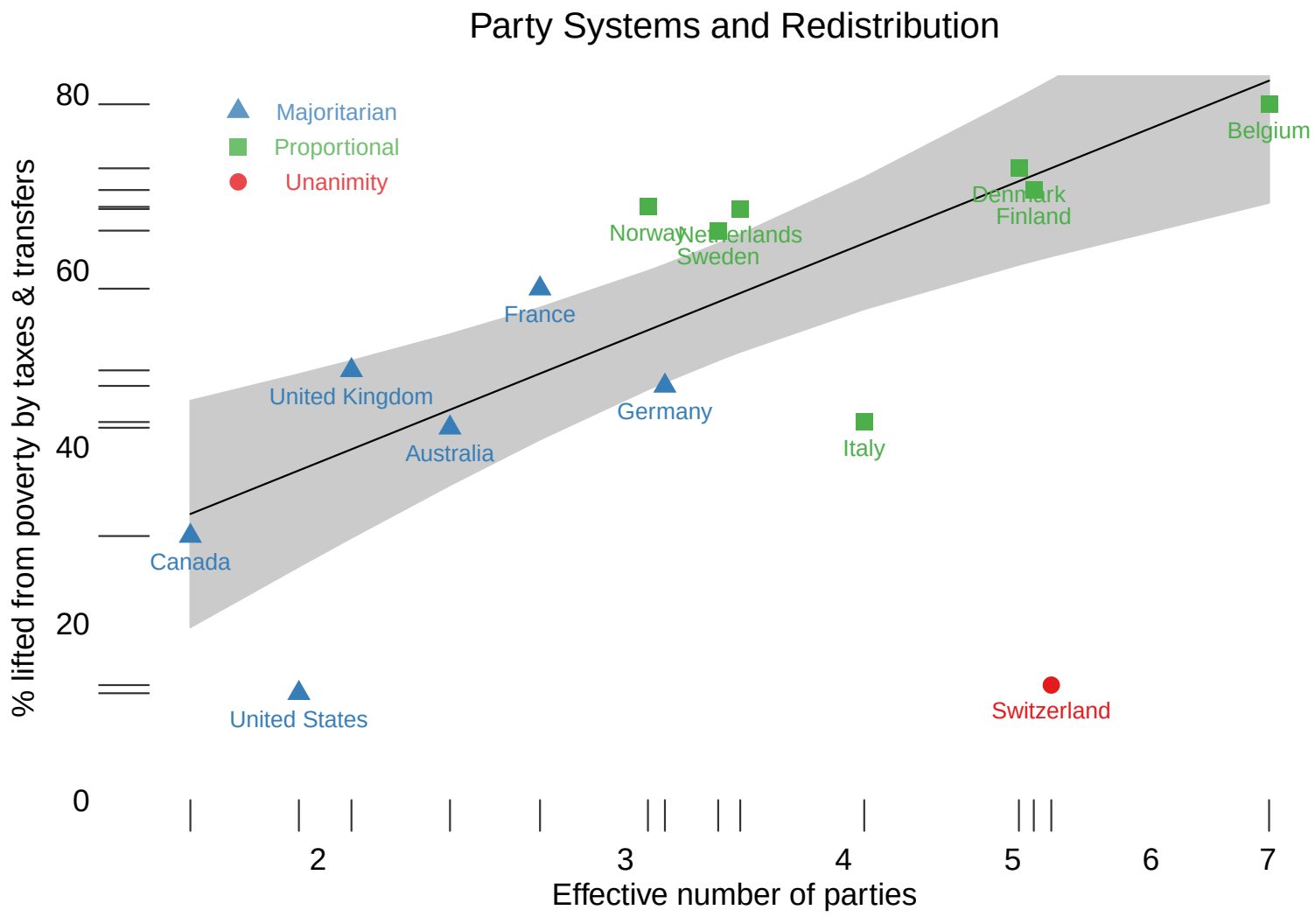
Overview of `tile`

- Three steps to make `tile` plots (from Chris's "Tufte Without Tears")
  1. **Create data trace**: Each trace contains the data and graphical parameters needed to plot a single set of graphical elements to one or more plots
  2. **Plot the data traces**: Using the `tile()` function, simultaneously plot all traces to all plots
     - This is the step where the scaffolding gets made: axes and titles
     - Set up the rows and columns of plots
     - Titles of plots, axes, rows of plots, columns of plots, etc.
     - Set up axis limits, ticks, tick labels, logging of axes
  3. **Examine output and revise**: Look at the graph made in step 2, and tweak the input parameters for steps 1 and 2 to make a better graph

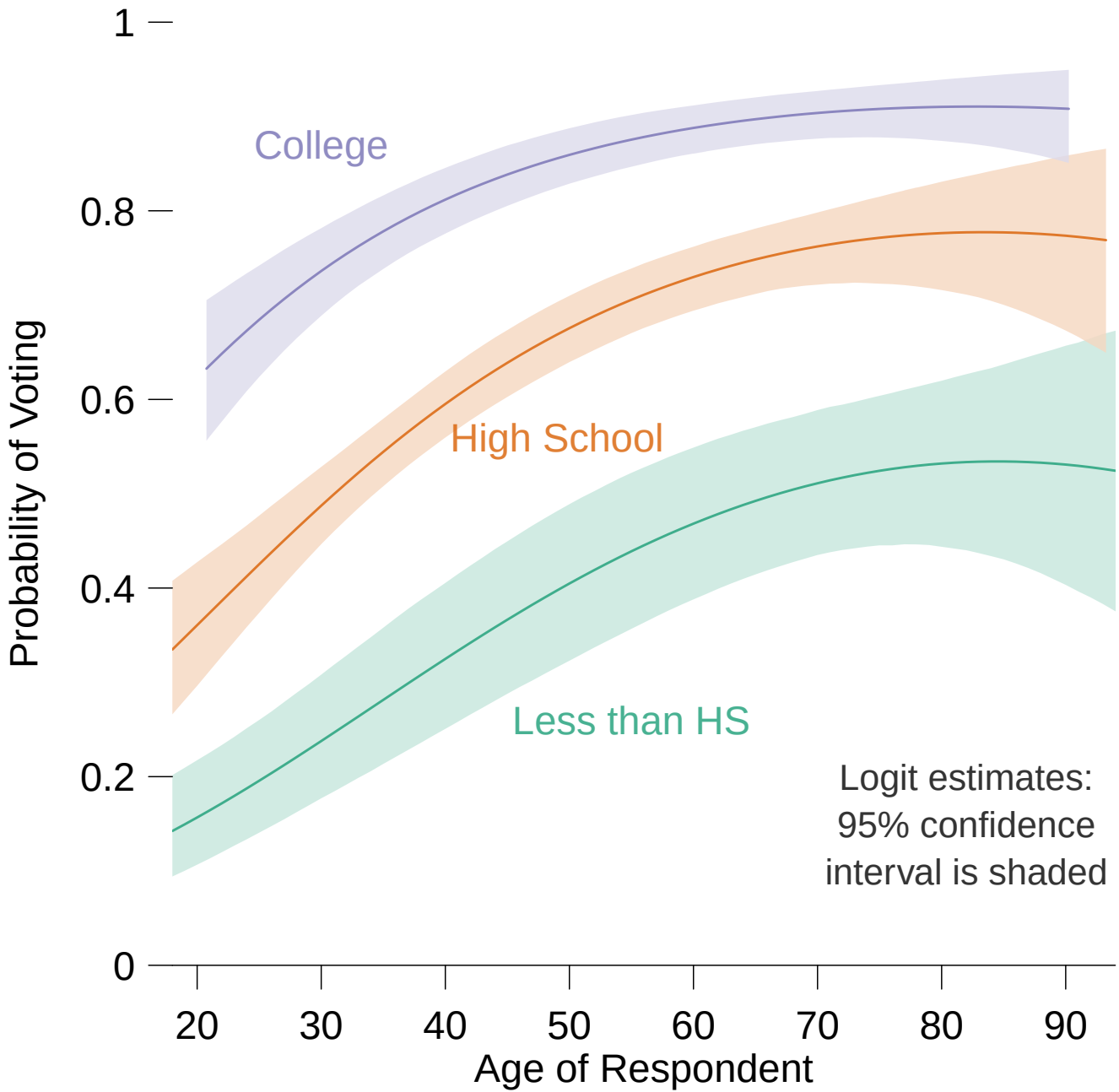
Three examples

- Scatterplot: HW1 example
- Expected probabilities and first differences: Voting example
- Ropeladder: Crime examples (if time permits)

Scatterplot: HW 1 example



Expected probabilities and first differences: Voting example



Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)

Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)

Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)
     - `lm()`, `glm()...`; `coef()`, `vcov()...`

Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)
     - `lm()`, `glm()...`; `coef()`, `vcov()...`
     - What you see in usual regression tables

Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)
     - `lm()`, `glm()...`; `coef()`, `vcov()...`
     - What you see in usual regression tables
  2. Capture our uncertainty around $\hat{\beta}_k$ by drawing, say, 10,000 $\tilde{\beta}_k$ from a multivariate normal distribution

Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)
     - `lm()`, `glm()...`; `coef()`, `vcov()...`
     - What you see in usual regression tables
  2. Capture our uncertainty around $\hat{\beta}_k$ by drawing, say, 10,000 $\tilde{\beta}_k$ from a multivariate normal distribution
     - `MASS::mvrnorm()`

Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)
     - `lm()`, `glm()...`; `coef()`, `vcov()...`
     - What you see in usual regression tables
  2. Capture our uncertainty around $\hat{\beta}_k$ by drawing, say, 10,000 $\tilde{\beta}_k$ from a multivariate normal distribution
     - `MASS::mvrnorm()`
  3. Specify counterfactual scenarios (hypothetical values for all relevant covariates x_k)

Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)
     - `lm()`, `glm()...`; `coef()`, `vcov()...`
     - What you see in usual regression tables
  2. Capture our uncertainty around $\hat{\beta}_k$ by drawing, say, 10,000 $\tilde{\beta}_k$ from a multivariate normal distribution
     - `MASS::mvrnorm()`
  3. Specify counterfactual scenarios (hypothetical values for all relevant covariates x_k)
     - `simcf::cfMake`, `cfChange`...

Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)
     - `lm()`, `glm()...`; `coef()`, `vcov()...`
     - What you see in usual regression tables
  2. Capture our uncertainty around $\hat{\beta}_k$ by drawing, say, 10,000 $\tilde{\beta}_k$ from a multivariate normal distribution
     - `MASS::mvrnorm()`
  3. Specify counterfactual scenarios (hypothetical values for all relevant covariates x_k)
     - `simcf::cfMake`, `cfChange`...
  4. Simulate quantities of interest by compounding those 10,000 $\tilde{\beta}_k$ with counterfactual scenarios

Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)
     - `lm()`, `glm()...`; `coef()`, `vcov()...`
     - What you see in usual regression tables
  2. Capture our uncertainty around $\hat{\beta}_k$ by drawing, say, 10,000 $\tilde{\beta}_k$ from a multivariate normal distribution
     - `MASS::mvrnorm()`
  3. Specify counterfactual scenarios (hypothetical values for all relevant covariates x_k)
     - `simcf::cfMake`, `cfChange`...
  4. Simulate quantities of interest by compounding those 10,000 $\tilde{\beta}_k$ with counterfactual scenarios
     - Then compute average (point estimate) and appropriate percentiles (confidence intervals)

Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)
     - `lm()`, `glm()...`; `coef()`, `vcov()...`
     - What you see in usual regression tables
  2. Capture our uncertainty around $\hat{\beta}_k$ by drawing, say, 10,000 $\tilde{\beta}_k$ from a multivariate normal distribution
     - `MASS::mvrnorm()`
  3. Specify counterfactual scenarios (hypothetical values for all relevant covariates x_k)
     - `simcf::cfMake`, `cfChange`...
  4. Simulate quantities of interest by compounding those 10,000 $\tilde{\beta}_k$ with counterfactual scenarios
     - Then compute average (point estimate) and appropriate percentiles (confidence intervals)
     - `simcf::logitsimev()` for expected values for logit models

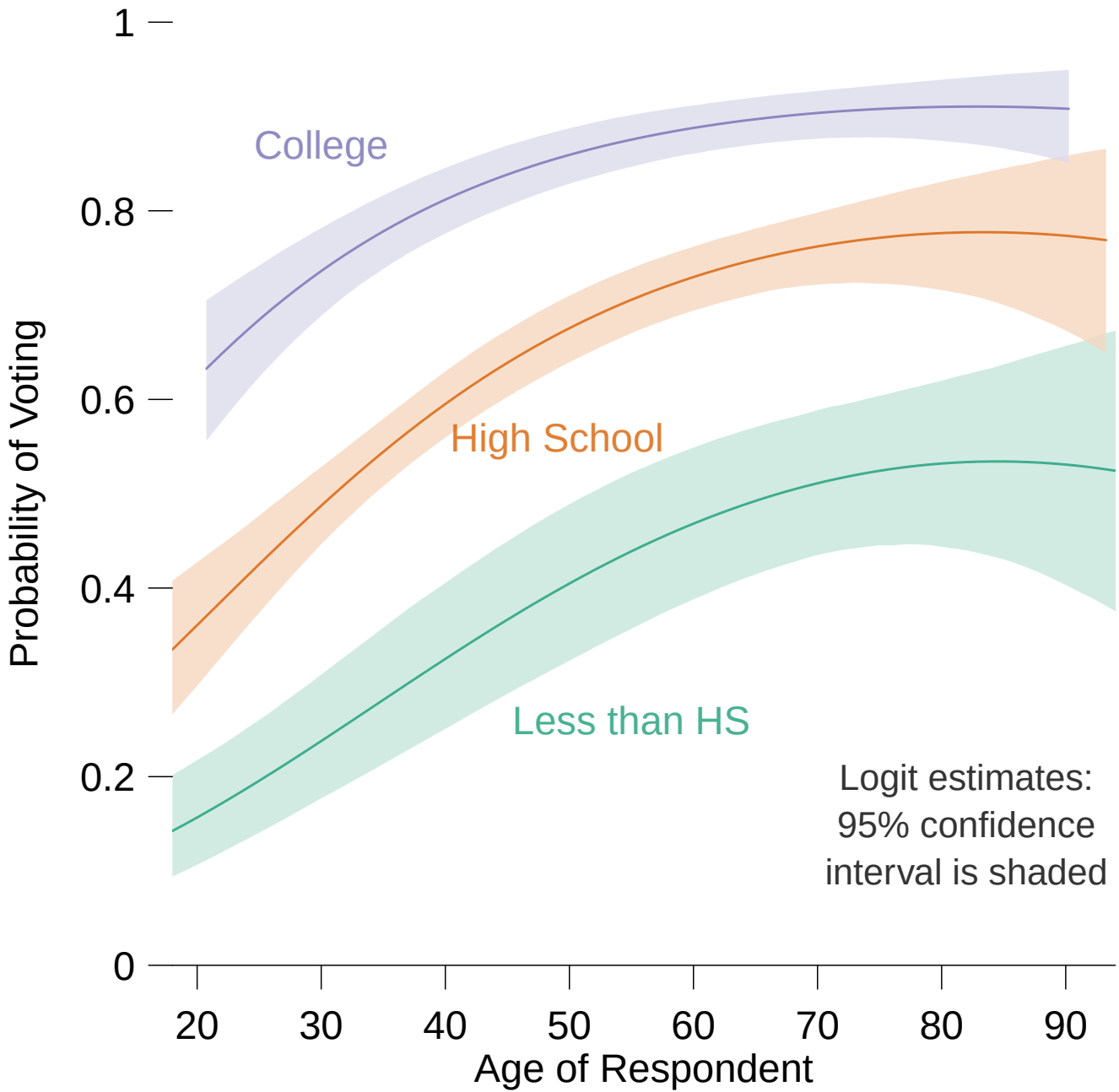
Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)
     - `lm()`, `glm()...`; `coef()`, `vcov()...`
     - What you see in usual regression tables
  2. Capture our uncertainty around $\hat{\beta}_k$ by drawing, say, 10,000 $\tilde{\beta}_k$ from a multivariate normal distribution
     - `MASS::mvrnorm()`
  3. Specify counterfactual scenarios (hypothetical values for all relevant covariates x_k)
     - `simcf::cfMake`, `cfChange`...
  4. Simulate quantities of interest by compounding those 10,000 $\tilde{\beta}_k$ with counterfactual scenarios
     - Then compute average (point estimate) and appropriate percentiles (confidence intervals)
     - `simcf::logitsimev()` for expected values for logit models
     - `logitsimfd` for first differences

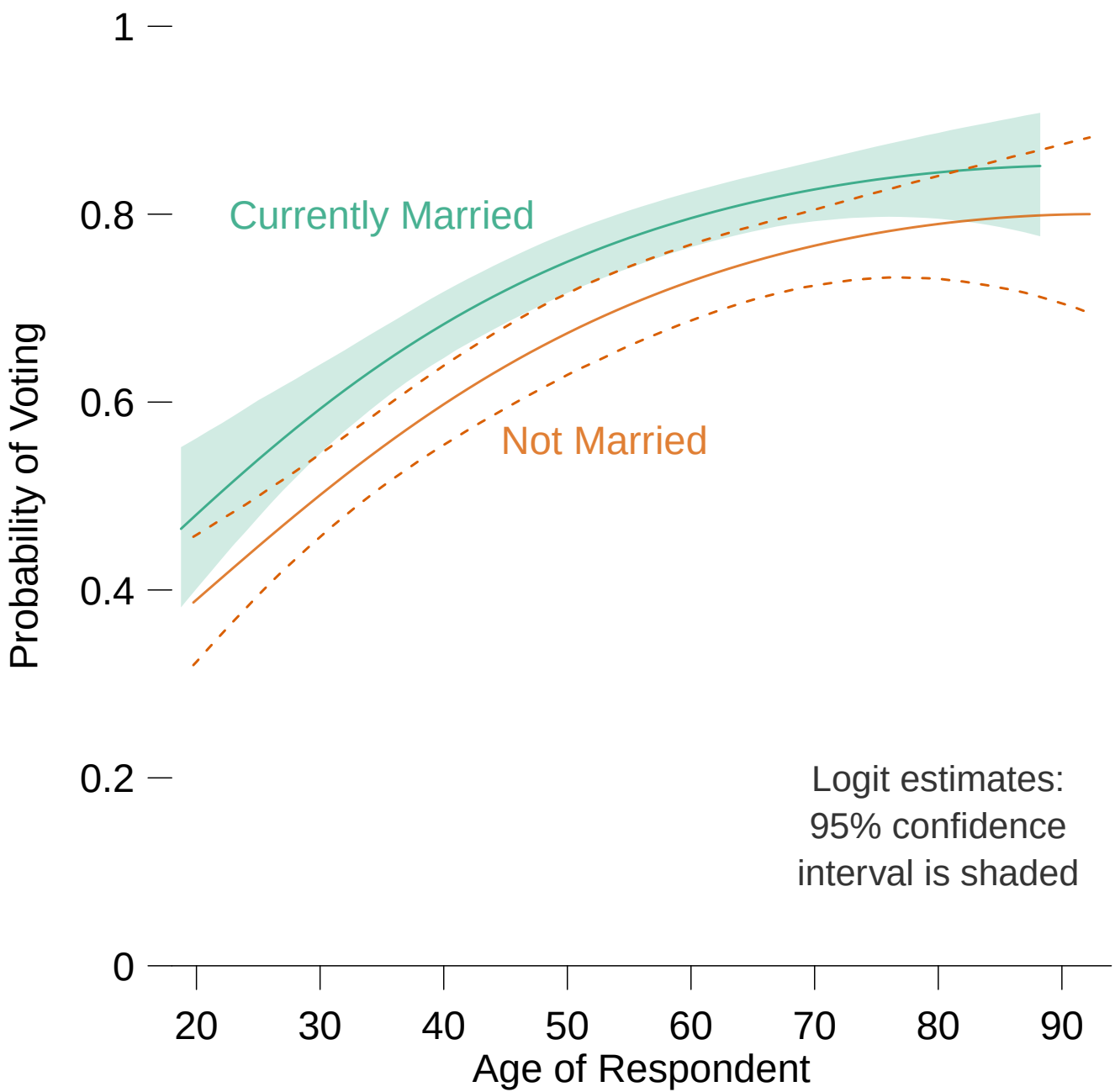
Voting example

- A quick detour to model results presentation and the logic of simulation (consult POLS/CSSS 510:MLE::Topic 3)
  1. Obtain estimated parameters ($\hat{\beta}_k$) and standard errors (more precisely, the variance-covariance matrix)
     - `lm()`, `glm()...`; `coef()`, `vcov()...`
     - What you see in usual regression tables
  2. Capture our uncertainty around $\hat{\beta}_k$ by drawing, say, 10,000 $\tilde{\beta}_k$ from a multivariate normal distribution
     - `MASS::mvrnorm()`
  3. Specify counterfactual scenarios (hypothetical values for all relevant covariates x_k)
     - `simcf::cfMake`, `cfChange`...
  4. Simulate quantities of interest by compounding those 10,000 $\tilde{\beta}_k$ with counterfactual scenarios
     - Then compute average (point estimate) and appropriate percentiles (confidence intervals)
     - `simcf::logitsimev()` for expected values for logit models
     - `logitsimfd` for first differences
     - `logitsimrr` for relative risks

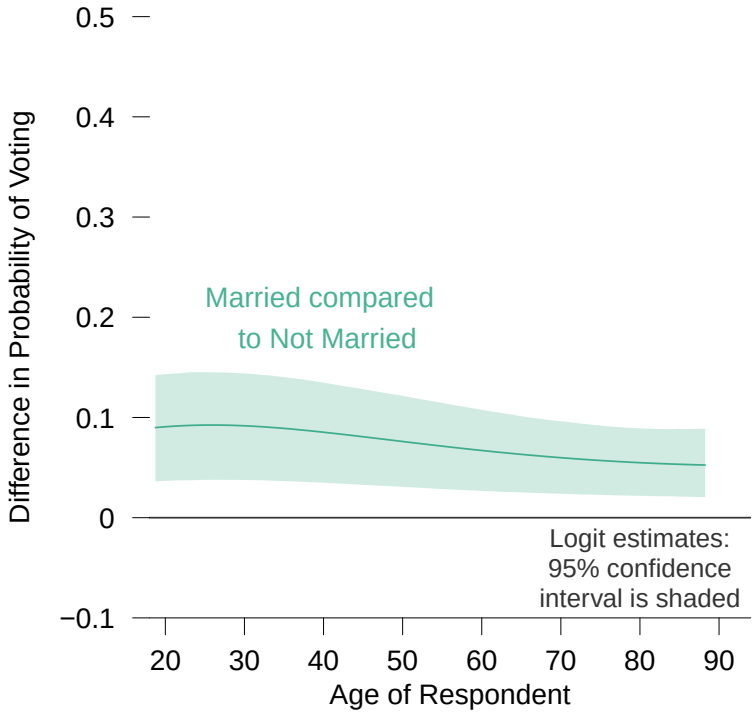
Expected probabilities and first differences: Voting example

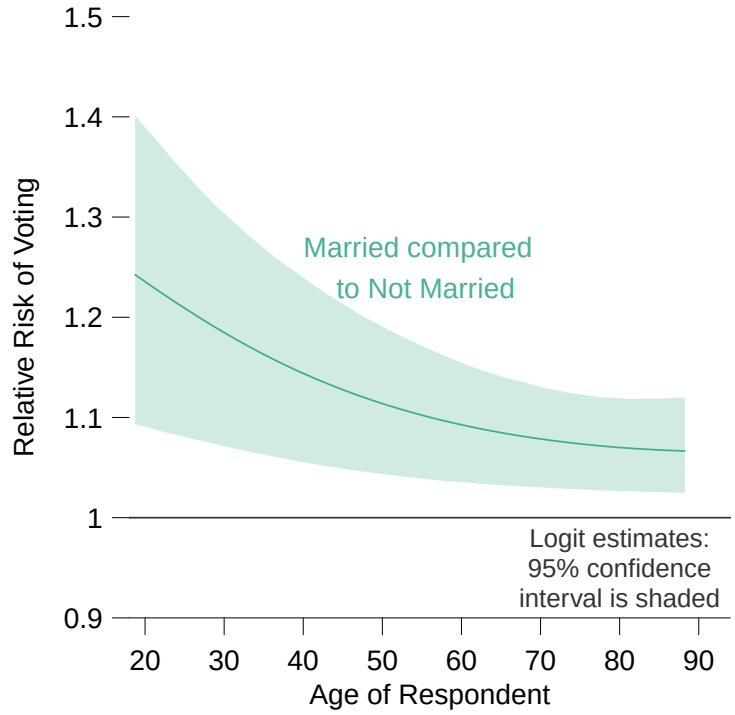


Expected probabilities and first differences: Voting example

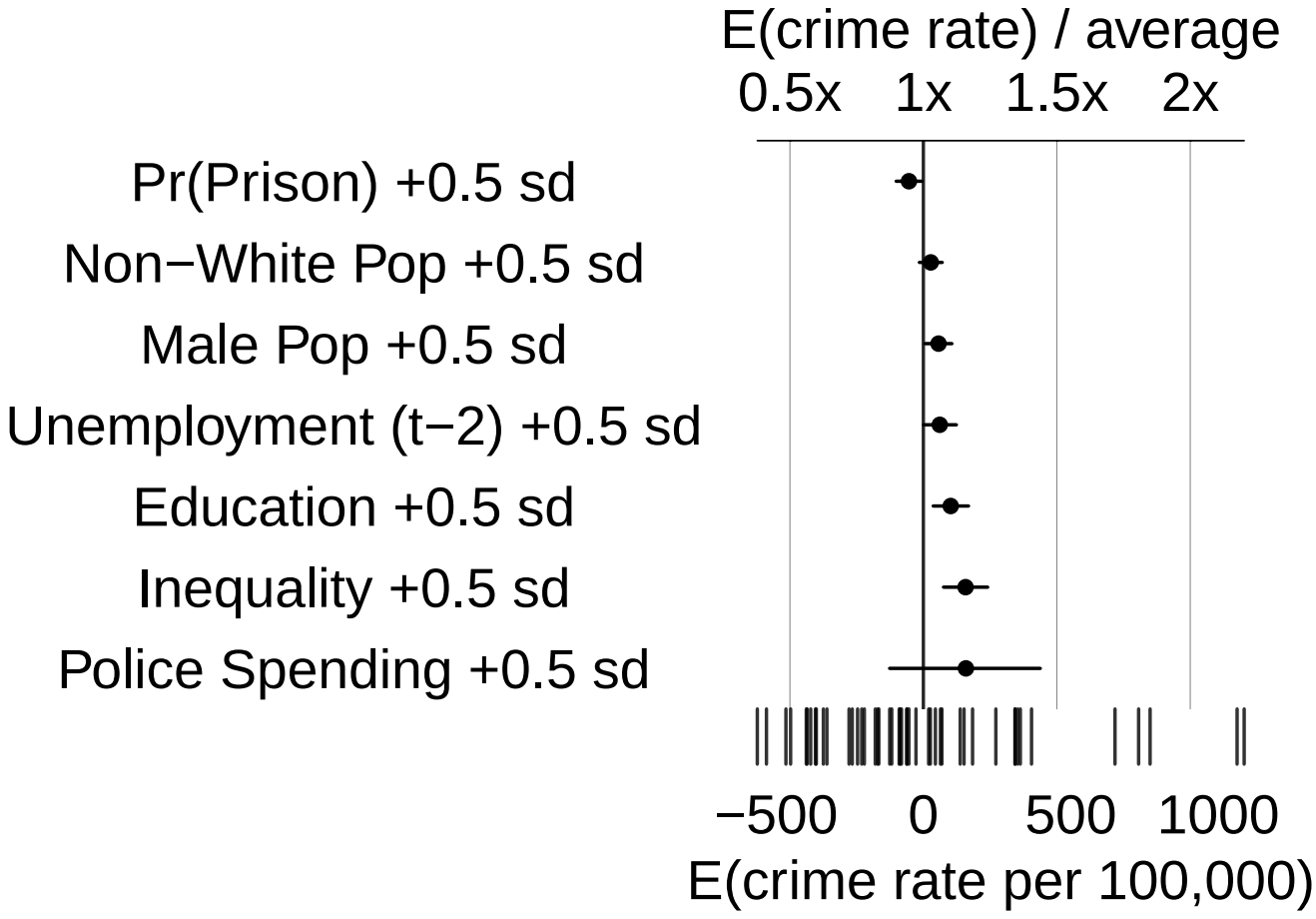


Expected probabilities and first differences: Voting example

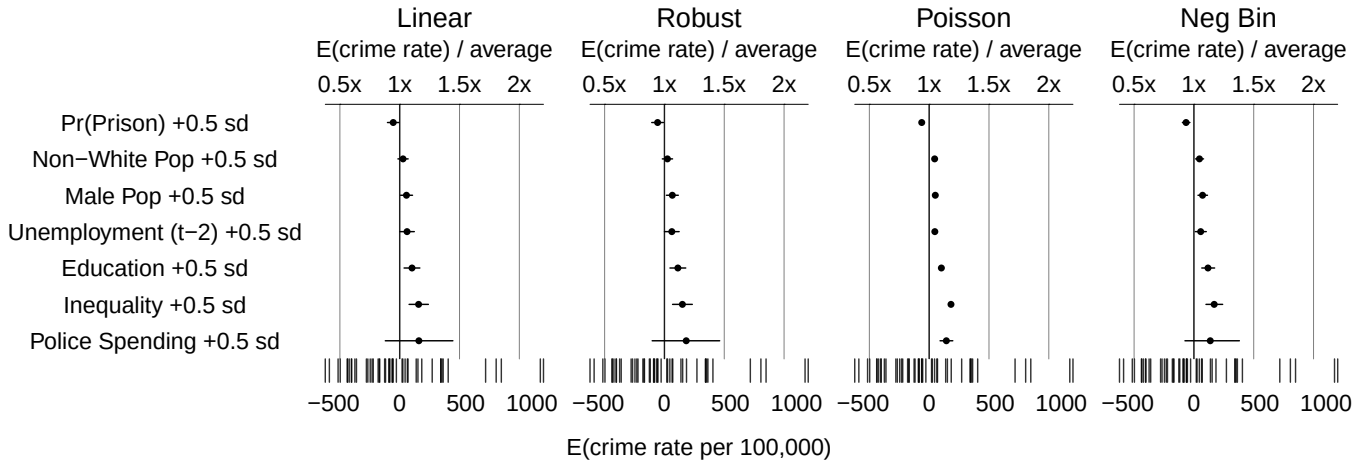


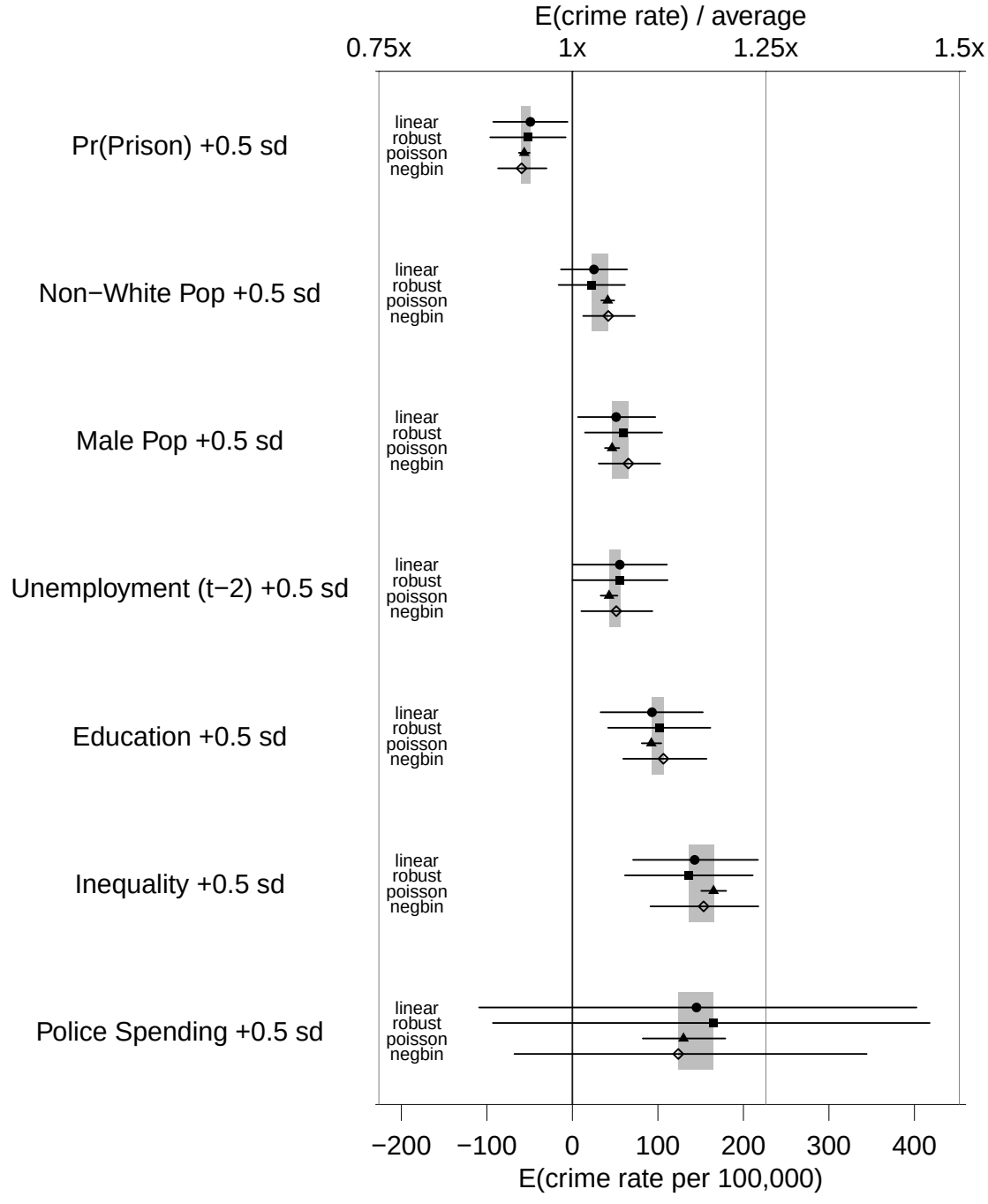


Ropeladder: Crime example (if time permits)

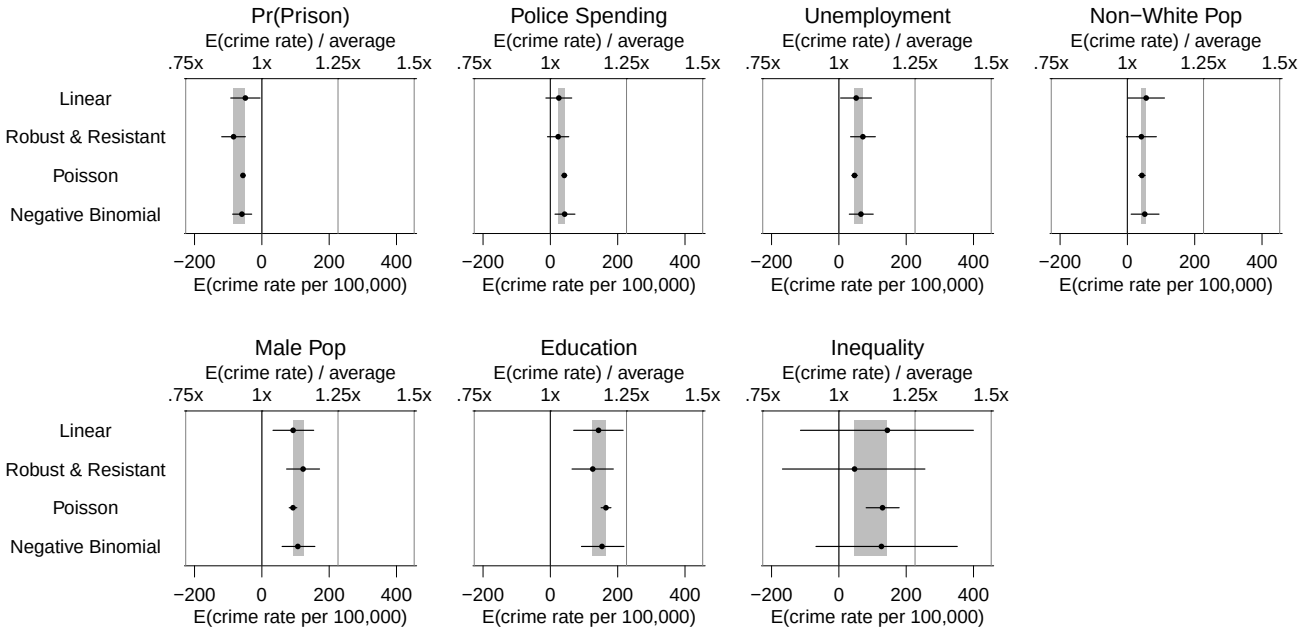


Ropeladder: Crime example (if time permits)




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

Ropeladder: Crime example (if time permits)



Installing `tile` and `simcf`

- Go to Chris's website, Software section

Installing `tile` and `simcf`

- Go to Chris's website, Software section
- Also download all `R` scripts and data under today's Lab section