

CSSS/POLS 510 MLE Lab

Lab 5. Binary Model, tile, and goodness of fit

Ramses Llobet

Agenda

1. Binary model
2. QIs with logit model
3. Overview of `tile`

0. MLE general notation

$$Y_i \sim f(\theta_i, \alpha) \quad (\text{stochastic})$$

$$\theta_i = g(\mathbf{x}_i\boldsymbol{\beta}) \quad (\text{systematic})$$

where

- ▶ Y_i is a random outcome variable.
- ▶ $f(\cdot)$ is a probability density function.
- ▶ θ_i is a systematic feature of the PDF that varies over i .
- ▶ α is an ancillary parameter (feature of f that we treat as constant).
- ▶ $g(\cdot)$ functional form for reparametrization of the data model.
- ▶ \mathbf{x}_i explanatory variables vector.
- ▶ $\boldsymbol{\beta}$ vector of effect parameters.

1. Binary models

$$Y_i \sim f_{Bernoulli}(\pi_i) \quad (\text{stochastic})$$

$$\pi_i = g(\mathbf{x}_i\boldsymbol{\beta}) \quad (\text{systematic})$$

where

$$\pi_i = \text{logit}^{-1}(\mathbf{x}_i\boldsymbol{\beta}) \quad (\text{systematic for logistic model})$$

$$\pi_i = \text{probit}^{-1}(\mathbf{x}_i\boldsymbol{\beta}) \quad (\text{systematic for probit model})$$

1. Binary model: Overview

1. Obtain data
2. Think model (distribution/covariates)
3. Fit model
4. Obtain ML estimates of it
5. Interpret those estimates (Estimate QOI)
6. Test goodness of fit
7. Present your results to a broad audience

2. Simulating QoI

In order to use `simcf` to generate quantities of interest, the following steps are needed:

1. Estimate: MLE $\hat{\beta}$ and its variance $\hat{V}(\hat{\beta})$
2. Simulate estimation uncertainty from a multivariate normal distribution:
Draw $\tilde{\beta} \sim MVN[\hat{\beta}, \hat{V}(\hat{\beta})]$
3. Create hypothetical scenarios of your substantive interest:
Choose values of X: X_c

2.1 Simulating QoI

4. Calculate expected values:

$$\tilde{\mu}_c = g(X_c, \tilde{\beta})$$

5. Simulate fundamental uncertainty:

$$\tilde{y}_c \sim f(\tilde{\mu}_c, \tilde{\alpha})$$

or

6. Compute EVs, First Differences or Relative Risks

$$\text{EV: } \mathbb{E}(y|X_{c1})$$

$$\text{FD: } \mathbb{E}(y|X_{c2}) - \mathbb{E}(y|X_{c1})$$

$$\text{RR: } \frac{\mathbb{E}(y|X_{c2})}{\mathbb{E}(y|X_{c1})}$$

2. Simulating QoI

In order to use `simcf` to generate quantities of interest, the following steps are needed:

1. Estimate: MLE $\hat{\beta}$ and its variance $\hat{V}(\hat{\beta})$
→ `optim()`, `glm()`
2. Simulate estimation uncertainty from a multivariate normal distribution:
Draw $\tilde{\beta} \sim MVN[\hat{\beta}, \hat{V}(\hat{\beta})]$
→ `MASS::mvrnorm()`
3. Create hypothetical scenarios of your substantive interest:
Choose values of X: X_c
→ `simcf::cfmake()`, `cfchange()` ...

2. Simulating QoI

4. Calculate expected values:

$$\tilde{\mu}_c = g(X_c, \tilde{\beta})$$

5. Simulate fundamental uncertainty:

$$\tilde{y}_c \sim f(\tilde{\mu}_c, \tilde{\alpha})$$

→ `simcf::hetnormsimpv()` ...

or

6. Compute EVs, First Differences or Relative Risks

$$\text{EV: } \mathbb{E}(y|X_{c1})$$

→ `simcf::logitsimev()` ...

$$\text{FD: } \mathbb{E}(y|X_{c2}) - \mathbb{E}(y|X_{c1})$$

→ `simcf::logitsimfd()` ...

$$\text{RR: } \frac{\mathbb{E}(y|X_{c2})}{\mathbb{E}(y|X_{c1})}$$

→ `simcf::logitsimrr()` ...

3. 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
 - ▶ Among others

3. 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()`, `pointstile()`, `polygonTile()`, `polylinesTile()`, and `textTile()`
 - ▶ Complex traces: `lineplot()`, `scatter()`, `ropeladder()`, and `rugTile()`

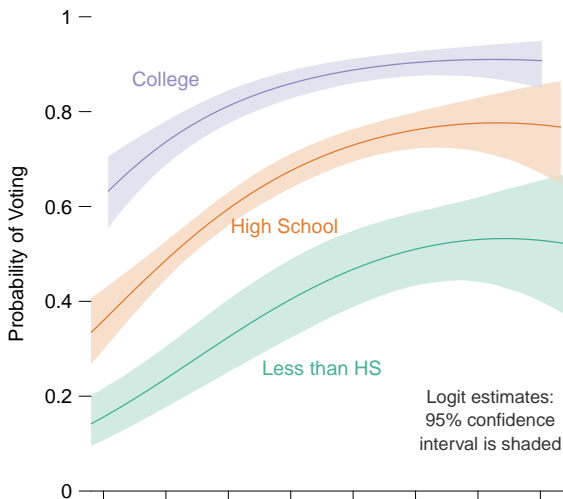
3. Overview of tile

- ▶ 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

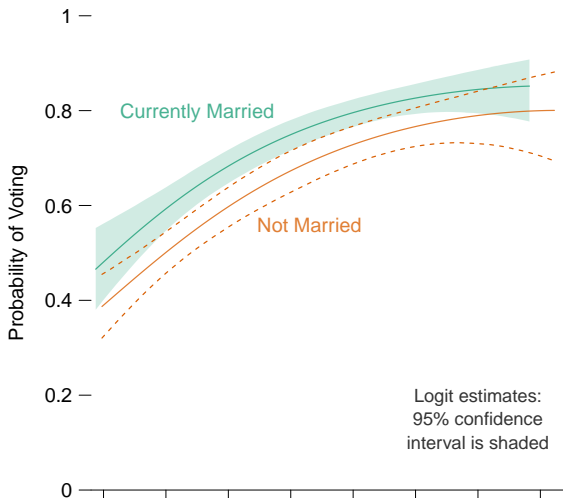
3. Overview of tile

- ▶ 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

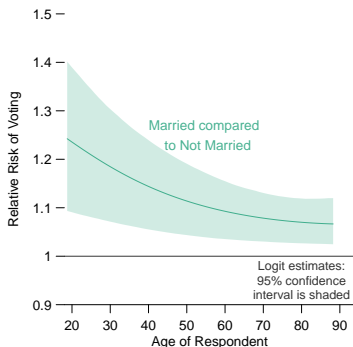
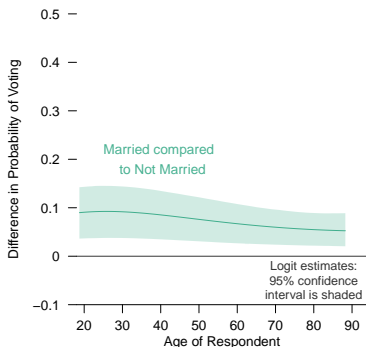
3. Expected probabilities and first differences: Voting example



3. Expected probabilities and first differences: Voting example



3. Expected probabilities and first differences: Voting example



FIN