This module will discuss how to implement two statistical tests in R: linear regression and logistic regression.

**Lines 7 to 24** provide a reminder about how to set your working directory and read in data. Variable and value labels for the items that are used in the demonstration follow on lines 28 to 58. A dataframe that only includes the variables of interest is generated at **line 68**. Data cleaning procedures follow on **lines 70 to 115**.

**Lines 119 to 163** demonstrate how to conduct linear regression in R with the lm() function to examine the influence of self-reported health and health insurance on the time since last doctor appointment. **Lines 132 to 135** implement simple linear regression with one predictor, while **lines 140 to 143** implement multiple linear regression with both predictors.

[SCROLL TO 148]

**Lines 148 to 159** extract the results from the lm output object. Note that there is more output stored in the object than what we extract here in this demonstration; these other elements can also be extracted with the dollar sign operator.

**Lines 167 to 227** demonstrate how to conduct logistic regression in R with the glm() function to examine the influence of self-reported health and health insurance on the odds of having delayed getting medical care in the past. GLM stands for generalized linear model and represents a larger class of analyses than just logistic regression. **Lines 180 to 183** implement simple logistic regression with one predictor. Logistic regression coefficient estimates are reported on the log-odds scale by default, which is not easily interpreted. **At line 190**, we exponentiate a log odds B to produce an odds ratio, which has a more intuitive interpretation.

[SCROLL TO 193]

**Lines 195-199** implement multiple logistic regression with both predictors. Odds ratios for the log-odds scale coefficients are calculated at **lines 206 and 209**.

[SCROLL TO 213]

**Lines 216 to 221** extract the results from the glm output object.

**Lines 231 to 358** generate tables for the linear regression and logistic regression models. Two approaches are implemented for each analysis: manual calculation of the relevant statistics and export as a custom table using stargazer and automatic table generation with sjPlot. The results from the linear regression model are extracted at **line 239** and labeled at **line 245**. 95% confidence intervals are calculated and added the table at **lines 250-252**.

[SCROLL TO 254]

Variable names are added to the table at **lines 255-264** and the table is exported at **lines 271-274.**

[SCROLL TO 276]

A table can instead be generated using specialized functions from the sjPlot package.

[SCROLL TO 297]

We repeat this process for the logistic regression results, with a few key differences. At **line 311**, the log odds B is converted to an odds ratio with the exp() function. The confidence intervals that are calculated at **lines 314 to 315** also use the exp() function to rescale the intervals to reflect a range around the odds ratio.

[SCROLL TO 334]

Again, the custom table is exported using stargazer at **lines 335-338**. sjPlot can also be used to generate tables for the logistic regression, as shown on **lines 340-358**.

**Lines 361 to 604** demonstrate how to generate plots for the linear regression and logistic regression models using ggplot2. Forest plots are used to display coefficient estimates and predicted values are displayed on either a bar plot or line plot. Note that sjPlot also has some functionality for generating plots, which can be explored further by entering the command at **line 366.**

[SCROLL TO 381]

A forest plot is generated at **lines 383 to 399.**

[SCROLL TO 403]

Plotting predicted values of a regression equation can be useful to give a visual summary of the model as a whole. This is accomplished by supplying sample values for the X variables in the regression equation and weighting them by the corresponding B (or slope) coefficient.

[SCROLL TO 431]

**Lines 432 to 435** generate the sample values of X for a subject with health insurance and mean self-reported health and a subject with no health insurance and mean self-reported health. Next, these sample values are supplied to the predict.lm() function at **line 438** to generate predicted values.

[SCROLL TO 448]

L**ines 449 to 460** generate a bar plot to display the predicted values, indicating the predicted difference in years since last doctor appointment between subjects with health insurance and without health insurance, at the mean level of self-reported health.

[SCROLL TO 463]

**Lines 467 to 475** repeat the procedure of generating predictions for all possible combinations of self-reported health and health insurance status.

[SCROLL TO 482]

**Lines 483 to 496** generate a line plot to display the predicted values, indicating the predicted change in years since last doctor appoint as a function of self-reported health among subjects with health insurance and subjects without health insurance.

[SCROLL TO 499]

These general steps are repeated for the logistic regression model. A forest plot is generated at **lines 508-524.**

[SCROLL TO 527]

Note that the predicted values that are generated by predict.glm() for the logistic regression model are on the log odds scale by default. At **lines 543-544**, these predictions are converted to predicted probabilities.

[SCROLL TO 552]

L**ines 553 to 564**  generate a bar plot to display the predicted probabilities, indicating the predicted difference in probability of delaying medical care between subjects with health insurance and without health insurance, at the mean level of self-reported health.

[SCROLL TO 589]

**Lines 590 to 603** generate a line plot to display the predicted probabilities, indicating the predicted change in probability of delaying medical care as a function of self-reported health among subjects with health insurance and subjects without health insurance.

[SCROLL TO 624]

**Lines 624 to 653** demonstrate how to export all tables and plots generated in this demonstration.