

19-704: APPLIED DATA ANALYSIS

Alex Davis
Carnegie Mellon University
Pittsburgh, Pennsylvania

April 13, 2015

Foreward

This class presents a lot of ideas and applications, covering much of the ground you will need to complete an applied data analysis project. Covering this much ground makes it necessary to sacrifice some material that is traditionally covered in statistics courses. I chose to sacrifice two things: 1) hypothesis testing, and 2) proofs. My reasoning is as follows. Hypothesis tests are automatically generated by computer programs and are overused, with researchers tending to just look at a printout of p-values and stopping the analysis there. I see no need to further reinforce this. Second, I find students don't get much intuition for modeling when writing a bunch of proofs, and the proofs focus on ideal cases, making it difficult to step back and figure out how to conduct data analyses in the real world.

I would also like to thank John Sperger, Dena Asta, Chris Yang, Patrick Sheehan, and Jeff Anderson for their helpful comments on these notes. All errors are the sole responsibility of the instructor.

Contents

I Ethics and Reproducibility	20
1 Research Ethics	22
1.1 Feynman’s Rule	22
1.2 The Nine Circles of Scientific Hell	23
1.2.1 Peer-Review	23
1.2.2 Overselling	25
1.2.3 Post-hoc Storytelling	25
1.2.4 p-value Fishing	25
1.2.5 Creative Outliers	26
1.2.6 Non and Partial Publication	26
1.3 How to Get $p < .05$: Monte Carlo Examples	27
1.3.1 Combining Dependent Variables	27
1.3.2 Using “controls”	31
1.3.3 Why hypothesis tests measure sample size	34
1.4 Reproducibility	35
1.5 Technical Problems with Applied Data Analysis	36
1.6 Homework 1	38
1.6.1 Optional	39
1.7 Mathematical Appendix	40
1.7.1 Unbiased Sample Variance Estimator	40
1.7.2 Variance of the Sample Mean	41
1.7.3 Student’s Two-Sample t-test	42
1.8 Homework 1 Solution	44
1.8.1 Situation B	44
1.8.2 Does anyone ever believe the null hypothesis?	44
1.8.3 When are the results of a t-test meaningless?	45
1.8.4 Weakly or strongly correlated dependent variables	45
II Data Analysis	46
2 Getting Started and the Five Stories	47
2.1 Getting Started	47

2.1.1	First Looks at the Data	48
2.2	Five Stories of Data Analysis	51
2.2.1	Model Checking	51
2.3	The Data Summary Story	52
2.3.1	Histograms	53
2.3.2	Cumulative Distributions	65
2.3.3	Scatterplots	68
2.4	The Conditional Distribution Story	70
2.4.1	Discrete Independent Variables	70
2.4.2	Continous Independent Variables	93
2.4.3	Outliers Mess Up the Conditional Distribution Story .	109
2.4.4	Why Use Graphics For The Conditional Distribution Story?	124
2.5	The Forecasting Story	136
2.5.1	Backcasting	137
2.5.2	Cross-Validation	139
2.6	The Statistical Inference Story	147
2.6.1	Four Responses to Non-random Samples	148
2.6.2	Statistical Inference for a Truly Random Sample	150
2.7	The Causal Inference Story	160
2.7.1	Counterfactuals	161
2.7.2	Random Assignment	161
2.7.3	Quasi-Experiments	163
2.8	Mathematical Appendix	165
2.8.1	Why confidence intervals for the conditional mean fan out	165
2.9	Homework 2	166
2.10	Homework 2 Solution	167
2.10.1	Replication	167
2.10.2	The Data Summary Story	168
2.10.3	The Conditional Distribution Story	180
2.10.4	The Forecasting Story	188
2.10.5	The Statistical Inference Story	191
2.10.6	The Causality Story	192
3	Multiple Linear Regression and Extensions	193
3.1	Gauss-Markov Assumptions	193
3.1.1	Linear in Parameters	194
3.1.2	Random Sampling	195
3.1.3	Zero Conditional Mean	195
3.1.4	No Perfect Collinearity	196
3.1.5	Unbiased Estimator	197
3.2	Examples and Interpretation	198

3.2.1	Dummy Coded Variables, Factors, and Intercepts	198
3.2.2	Intercept and Continuous Predictor	200
3.2.3	Interaction Between a Factor and Continuous Predictor	205
3.2.4	Standardizing Predictors for Interpretability	211
3.3	Multiple Regression and Partial Residuals	212
3.3.1	Apartment Building Example	213
3.4	Discovering Non-Linear Transformations	218
3.4.1	Component Plus Residual Plots	219
3.4.2	Box-Tidwell Power Transformations	221
3.4.3	Semi/Non Parametric Approaches	227
3.4.4	Interpreting Coefficients with Non-linear Transforms .	235
3.5	Multicollinearity	240
3.5.1	Variance Inflation	243
3.5.2	Dealing with Multicollinearity	243
3.6	Heteroskedasticity	244
3.6.1	Baseball Example	244
3.6.2	Natural Logarithm and Box-Cox Transforms	255
3.6.3	Robust Standard Errors	274
3.6.4	Weighted Least Squares	282
3.6.5	Heteroskedasticity and Interactions	283
3.7	Summary	291
3.8	Mathematical Appendix	292
3.8.1	Proof that OLS estimators are unbiased	292
3.8.2	Percent Change Interpretation of the Log Transform .	293
3.8.3	Robust Standard Errors	294
3.8.4	Robust Standard Errors in Matrix Form	295
3.8.5	Box-Tidwell Transformation	296
3.8.6	Semi/Nonparametric Models	297
3.8.7	Box-Cox Transformation	302
3.9	Homework 3	305
4	Discrete Outcomes and The Generalized Linear Model	308
4.1	The Data Summary Story	308
4.1.1	Absolute Risk, Relative Risk, and The Odds Ratio .	311
4.1.2	The Odds Ratio	311
4.1.3	The Odds Ratio	312
4.2	The Conditional Distribution Story	322
4.2.1	Linear Probability Model	322
4.2.2	Non-Parametric Comparison	327
4.2.3	Incorrectly Using the Logit Transform	328
4.3	Logistic Regression	332
4.3.1	Bernoulli Stochastic Component	332

4.3.2	Bernoulli Likelihood	334
4.3.3	The Logit Link Function	339
4.3.4	The Logistic (Inverse Logit) Function	343
4.4	Interpreting Logistic Regression	352
4.4.1	Odds Ratio in Logistic Regression	356
4.4.2	Average Marginal Effect	356
4.4.3	Arsenic in Bangladesh Example	361
4.5	Model Checking for Logistic Regression	378
4.5.1	Calibration	379
4.5.2	Alternative Link Functions	382
4.5.3	Partial Residual Plots and Non-Linear Transformations	390
4.5.4	Complete and Quasi-Complete Separation	400
4.6	The Forecasting Story	402
4.6.1	Confusion Matrix	403
4.6.2	ROC Curve	404
4.6.3	Cross-Validated ROC Curve	407
4.7	Statistical Inference Story	411
4.7.1	Confidence Intervals For Coefficients and Odds Ratios .	413
4.7.2	Confidence Interval for the Conditional Mean	414
4.8	Homework 4	417
5	Multi-level Models	419
5.1	Introduction to Multi-Level Models	419
5.1.1	Varying Intercepts	420
5.1.2	Varying Slopes	422
5.1.3	Varying Intercepts and Slopes	425
5.2	Radon Example	427
5.3	Three Types of Pooling	440
5.3.1	Fixed and Random Effects	441
5.3.2	Completely Pooled Intercept	442
5.3.3	No Pooled Intercept	447
5.3.4	Partially Pooled Intercept	450
5.4	Pooling with Predictors	455
5.5	Group-Level Predictors	466
5.6	Varying Intercepts and Varying Slopes	475
5.7	Ordinary Least Squares Regression with Correlated Errors .	482
5.7.1	Estimation Using Feasible Generalized Least Squares .	483
5.8	Residual Diagnostics for Multi-Level Models	493
5.8.1	Level 1 Least Squares Diagnostics	493
5.8.2	Level 2 Empirical Bayes Diagnostics	501
5.9	An Iterative Approximate MLM Procedure	508
5.10	The Forecasting Story	508

5.10.1	Predictions for Existing and New Groups	508
5.10.2	Cross-Validation	510
5.11	Statistical Inference Story	513
5.11.1	Intraclass Correlation	513
5.11.2	The Moulton Factor	515
5.11.3	The Moulton Factor for Individual Level Regressors . .	516
5.11.4	Cluster-Robust Standard Errors	517
5.12	Homework 5	524

List of Tables

3.1	Summary table of three models of Child's Test Score based on mother's characteristics.	211
4.1	Calibration Table. Observed (Obs) and estimated expected (Exp) frequencies within each decile of risk for Switch = 1 and Switch = 0 using the fitted logistic regression model.	381
5.1	Degree of pooling as a function of sample size, within-group variance σ_y^2 and between group variance σ_α^2	454
5.2	Standard errors for the intercept, floor, and loguranium regressors using homoskedastic standard errors (OLS), partial pooling (Partial Pool), heteroskedastic standard errors (Het), cluster-robust standard errors (Cluster), and block bootstrap (Bootstrap).	523

List of Figures

1.1	The Nine Circles of Scientific Hell	24
2.1	Histogram of average happiness z-scores for each treatment group.	55
2.2	Histograms with bin width chosen using the Freedman-Diaconis rule (top), a tiny bin width ($h = 0.001$), or a large bin with ($h = 0.5$)	58
2.3	Scatterplots of mean happiness z-scores by treatment group.	60
2.4	Scatterplots of mean happiness z-scores by treatment group, with mean line (red) and median line (blue).	62
2.5	Box plot of mean happiness z-scores by treatment group.	63
2.6	Box plot of mean happiness z-scores by treatment group.	65
2.7	Plots of empirical cumulative distribution functions of mean happiness z-scores for each treatment group.	67
2.8	Scatterplot with mean happiness z-scores (y-axis) and mean economic self-sufficiency z-scores (x-axis) for each cell in the study.	69
2.9	Plot of sorted data from the normal distribution and the experimental group. The normal distribution has the same number of observations (68), mean, and standard deviation as the experimental group sample.	72
2.10	Plot of quantiles from a simulated normal distribution and the experimental group. There are 100 quantiles, but only 68 observations in the experimental group. This causes a problem which makes it look like the experimental group data is more compressed than the normal distribution, when this is an artifact of the truncation.	75
2.11	Plot of quantiles from a simulated normal distribution and the experimental group using the rule $(k - .5)/n$ to select probabilities for the quantiles.	77
2.12	Plot of quantiles from a simulated normal distribution and simulated normal distribution with twice the standard deviation.	79

2.13 Plot of quantiles from a simulated normal distribution and simulated normal distribution with twice the standard deviation on the x axis.	81
2.14 Plot of quantiles from a simulated normal distribution (x axis) and simulated normal distribution with mean shift of +1 (y axis).	83
2.15 Probability density function of the log-normal distribution, showing positive skew, with high density to the left of the graph (toward zero) and a long right tail.	85
2.16 Plot of quantiles from a simulated normal distribution, simulated log-normal distribution (top), and log-transformed simulated log-normal distribution (bottom).	87
2.17 Plot of quantiles from a simulated normal distribution and t-distribution with 10 degrees of freedom.	89
2.18 Plot of the empirical cumulative distribution for the three treatment groups (red) versus the normal distribution with equivalent mean and standard deviation.	92
2.19 Scatterplot of mean happiness z-scores versus mean economic self-sufficiency z-scores.	95
2.20 Scatterplot of mean happiness z-scores versus mean economic self-sufficiency z-scores with least squares regression line added.	98
2.21 Plot of mean economic self-sufficiency z-scores (x-axis) against mean happiness z-scores (y-axis) with observed data (black) and data simulated from the linear model (red).	101
2.22 Plot of mean economic self-sufficiency z-scores (x-axis) against mean happiness z-scores (y-axis) with observed data (black) and data simulated from the linear model (red).	103
2.23 Plot of mean economic self-sufficiency z-scores (x-axis) against mean happiness z-scores (y-axis) with observed data (black) and data simulated from three simulated linear models (red, blue, and green).	106
2.24 Q-Q plots of the studentized regression residuals (y-axis) plotted against quantiles of the t-distribution (x-axis) for the actual (left) and simulated (right) regressions. Red dotted lines show 95% confidence envelopes for simulations from the t distribution with $n - k - 1$ degrees of freedom.	108
2.25 Leverage demonstration. The data are on a horizontal line except for a single observation, with the same y value. As we go from the top left to bottom right, the x value of that observation changes, showing that the same y value has a larger effect on the regression line when it is farther away from the mean of x	111

2.26 Source: Judd, C. M., McClelland, G. H. (1989). Data analysis: A model comparison approach. New York: Harcourt Brace Jovanovich.	112
2.27 Influence index plot showing Cook's D values (top), jackknife residuals, bonferonni adjusted p-values, and hat values (bottom). To some degree the Cook's distance can be seen as the product of the hat value (leverage) and jackknife residual (discrepancy).	120
2.28 Quantile (median) regression (red) and linear least squares (mean) regression (blue).	123
2.29 Histograms of y for the four Anscombe datasets.	126
2.30 Histograms of x for the four Anscombe datasets.	128
2.31 Scatterplot and regression lines for the four Anscombe datasets.	131
2.32 Regression diagnostics for Anscombe dataset 2, that has a quadratic form but the regression only has a linear term.	133
2.33 Regression diagnostics for Anscombe dataset 3, with massive Case A outlier.	135
2.34 Regression diagnostics for Anscombe dataset 4, with massive Case C outlier.	136
2.35 In-sample predictions from linear regression. The actual happiness scores are on the y axis, with predicted happiness scores on the x axis.	138
2.36 Variation in cross-validated rMSE of the simple and complex models as a function of the number of folds.	146
2.37 Confidence interval for the mean of happiness scores when economic self-sufficiency is 0.5.	158
2.38 Confidence bands for the conditional mean happiness z-scores as a function of mean cconomic self-sufficiency z-scores.	160
2.39 Histogram of weekly wages from the 1988 CPS data. Histogram bandwidth is chosen using the Freedman-Diaconis rule.	170
2.40 Histogram of log weekly wages from the 1988 CPS data. Histogram bandwidth is chosen using the Scott rule.	171
2.41 Histogram of years of education from the 1988 CPS data. Histogram bandwidth is chosen using the Freedman-Diaconis rule.	173
2.42 Histogram of years of potential experience from the 1988 CPS data. Histogram bandwidth is chosen using the Freedman- Diaconis rule.	174
2.43 Histogram of years of potential experience from the 1988 CPS data, where observations with negative experience are recoded as zero. Histogram bandwidth is chosen using the Freedman- Diaconis rule.	176

2.44	Scatterplot with wages (top plot) or log wages (bottom plot) and years of education (x axis) with median regression line (red) and least-squares regression line (blue).	178
2.45	Scatterplot with wages (top plot) or log wages (bottom plot) and potential years of experience (x axis) with median regression line (red) and least-squares regression line (blue).	180
2.46	Plot of quantiles from a simulated normal distribution and weekly wages.	182
2.47	Plot of quantiles from a simulated normal distribution and log weekly wages.	184
2.48	Q-Q plots of the jackknife regression residuals against the <i>t</i> -distribution for the Table 1.A regression (left) and Table 2.A regression (right).	186
2.49	Plots of Cook's D (top row), jackknife residuals, Bonferroni p-values on the jackknife residuals, and hat values (bottom row) for the Mincer Table 1.A model.	187
2.50	Actual versus predicted log wages from the Mincer type model.	189
3.1	Child's test score by mother's high school completion.	200
3.2	QQ plot of the jackknife regression residuals. Dashed red lines show 95% confidence bands for quantiles of the <i>t</i> distribution with $n - k - 1$ degrees of freedom.	201
3.3	Child's test score by mother's IQ score, with different intercepts based on mother's high school graduation.	203
3.4	Child's test score by mother's IQ score, with different intercepts based on whether the mother did not graduate high school (top), or did graduate high school (bottom).	205
3.5	Child's test score by mother's IQ score, with different intercepts and slopes based on mother's high school graduation.	208
3.6	Child's test score by mother's IQ score, with different intercepts and slopes based on whether the mother did not graduate high school (top), or did graduate high school (bottom).	210
3.7	Relationship between building area and lot size.	215
3.8	Relationship between building area and building assessed value.	216
3.9	Partial regression residuals of building's assesed value and lot size.	218
3.10	Toy component-residual plot example. The top left plot shows the marginal relationship between x and y , which is $y = x^2 + \epsilon$. The top right plot shows that this relationship is recovered by plotting the component plus residuals against x . The bottom left plot shows the residuals plotted against x , failing to show the correct transformation. The bottom right plot shows the component plotted against x	221

3.11	Toy component-residual plot example. The true relationship is $y = x_1 + x_2^2$, meaning x_2 needs to be transformed $g(x_2) = x_2^2$. The estimated regression is $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$. x_1 and x_2 are linearly related such that $x_2 = 0.5x_1 + u$ where u is a uniform random variable.	224
3.12	Toy component-residual plot example, using the correct power transformation $g(x_2) = x_2^2$. The true relationship is $y = x_1 + x_2^2$, meaning x_2 needs to be transformed $g(x_2) = x_2^2$. The estimated regression is $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$. x_1 and x_2 are linearly related such that $x_2 = 0.5x_1 + u$ where u is a uniform random variable.	226
3.13	“Unknown” fourth-degree polynomial, possibly hidden in a multiple regression function with other independent variables.	228
3.14	Example of cubic spline function with knots circled and first and second derivatives around the knots showing continuity. There are 7 sections of cubic polynomial spliced together to construct the spline. Image taken from Wood, S. (2006). Generalized additive models: An introduction with R. CRC press.	229
3.15	Cross-validation to select λ for cubic regression spline. As can be seen λ is too large on the far left plot (too much penalization for wigginess) and too small on the far right plot (not enough penalty). The plot in the middle shows the optimal value of λ selected by cross-validation. Image taken from Wood, S. (2006). Generalized additive models: An introduction with R. CRC press.	230
3.16	Unknown fourth-degree polynomial with GAM overlaid.	232
3.17	Toy partial residual plot example, fit using a generalized additive model. The true population regression function is $y = 11x_1 - 6x_1^2 + x_1^3 + x_2^2$. The plot on the left shows that the GAM discovered the cubic polynomial relationship between x_1 and y , and the right plot shows the GAM discovered the power relationship between x_2 and y . Shaded areas show 95% confidence bands around the conditional mean.	234
3.18	Scatterplot of time to finish a 5k race against runner’s age (top), and runner’s average number of miles run in training (bottom).	237
3.19	Scatterplot of time to finish a 5K race against runner’s average number of miles run in training. Curve from regression with quadratic fit is shown.	239
3.20	Scatterplot matrix of building value, lot size, building area, number of apartments, and age.	242
3.21	Histogram of homeruns in the major league (Majors).	245
3.22	Quantile-Quantile plot of homeruns in the majors against normal distribution with equal mean and standard deviation.	247
3.23	Histogram of homeruns in the minor league (Minors).	248

3.24	Scatterplot of major and minor homeruns.	249
3.25	Quantile-Quantile plot of homeruns in the minors against homeruns in the majors.	251
3.26	Jackknife residuals for the regression of homeruns in the majors on homeruns in the minors.	253
3.27	Fitted versus Jackknife residual plot of regression of homeruns in the majors on homeruns in the minors.	254
3.28	Plot of Box-Cox estimate of the parameter λ for the Box-Cox transformation. The x axis shows the λ values that were tested. The y axis shows the log-likelihood.	258
3.29	Histogram of the log-transformed major homeruns (top), and power transformed homeruns in the majors (bottom) with $\lambda = 0.3$	259
3.30	Plot of log major homeruns versus minor homeruns (top), and power transformed major homeruns versus minor homeruns (bottom) with $\lambda = 0.3$ for the Box-Cox power transformation.	261
3.31	Jackknife residuals versus fitted values from the regression of log-transformed major homeruns on minor homeruns.	263
3.32	Relationship between major homeruns and minor homeruns for the logged (top) and unlogged (bottom) regressions.	267
3.33	Relationship between major homeruns and minor homeruns for the logged (top) and unlogged (bottom) regressions.	269
3.34	Relationship between major homeruns and minor homeruns for the logged (top) and unlogged (bottom) regressions.	271
3.35	Fitted versus residual plot of logged (left) and unlogged (right) regressions.	273
3.36	Simulated linear regression with heteroskedasticity proportional to $(x_i - \bar{x})^2$ (left), and no heteroskedasticity (right). The average variance of the errors is the same for the two datasets.	276
3.37	Simulated linear regression with heteroskedasticity proportional to $(x_i - \bar{x})^2$ (left) and homoskedasticity (right). Small sample ($n = 50$) regression lines are plotted in blue.	279
3.38	Heteroskedasticity resulting from an omitted interaction.	285
3.39	Heteroskedasticity resulting from an omitted interaction.	286
3.40	The histogram of y , qqplot of y against the normal distribution, the jackknife residuals versus fitted values, and qqplot of the jackknife regression residuals versus the quantiles of the t distribution, all show potential model misspecification.	288
3.41	The histogram of \sqrt{y} , qqplot of \sqrt{y} against the normal distribution, the jackknife residuals versus fitted values, and qqplot of the jackknife regression residuals versus the quantiles of the t distribution.	290

3.42	Cubic regression spline fit to engine size and wear data with 4 knots.	300
3.43	Cross-validation to select λ for cubic regression spline. As can be seen λ is too large on the far left plot (too much penalization for wigginess) and too small on the far right plot (not enough penalty). The plot in the middle shows the optimal value of λ selected by cross-validation. Image taken from Wood, S. (2006). Generalized additive models: An introduction with R. CRC press.	302
4.1	Spineplot showing the proportion of patients with no (dark grey), some (middle grey), or marked (light grey) improvement by treatment condition.	310
4.2	Plot of odds ratios by probability difference. The same data are used in each plot, with the y axis increasing from 1 to unconstrained, going from the top left to bottom right.	314
4.3	Plot of log odds ratios by probability difference (left), with linear and GAM fit (right).	316
4.4	Plot of deviation of the log odds ratio from linearity depending on the proportion p_1 and the difference in proportions $p_2 - p_1$ by size of the difference (0.1, 0.2, 0.3, 0.4), from top left to bottom right respectively.	319
4.5	Plot of odds ratios by risk ratio.	321
4.6	Conservativism as a function of the person's vote for governor.	324
4.7	Probability of voting republican for different levels of reported conservativism using a linear probability model.	326
4.8	Probability of voting republican for different levels of reported conservativism. The black dashed line shows the linear fit, whereas the red line shows the non-parametric kernel regression.	328
4.9	Plot of a proportion (p) and the logit transformation of the proportion (Logit (p)).	330
4.10	Plot of Logit(p) and $p = \text{Logistic}(\text{Logit}(p))$	331
4.11	Plots of 100 draws from a Bernoulli random variable with $P(Y = 1) = p$ varying from 0.1 (top left), to 0.9 (bottom right).	333
4.12	Plot of the Bernoulli likelihood function $L(p y_1) = p^{y_1}(1-p)^{1-y_1}$ for $y_1 = 1$ versus the hypothesized parameters p	336
4.13	Plot of the Bernoulli likelihood function $L(p y_1) \times L(p y_2) = p^{y_1}(1-p)^{1-y_1} \times p^{y_2}(1-p)^{1-y_2}$ for $y_1 = 1$ and $y_2 = 0$ versus the hypothesized parameters p	338
4.14	Plot of a proportion ($p(x)$) and linear predictors $\eta(x)$ related through the logistic function.	344
4.15	Plot of draws from a Bernoulli($\frac{1}{1+e^{-\eta(x)}}$) distribution and logistic regression curve for $\eta(x) = x\beta$ with $\beta \in \{0.5, 1, 2\}$ (from top to bottom).	347

4.16 Plot of draws from a Bernoulli($\frac{1}{1+e^{-\eta(x)}}$) distribution and logistic regression curve for $\eta(x) = x\beta$ with $\beta \in \{-0.5, -1, -2\}$ (from top to bottom).	349
4.17 Plot of draws from a Bernoulli($\frac{1}{1+e^{-\eta}}$) distribution and logistic regression curve for $\eta(x) = \beta_0 + \beta x$ with $\beta_0 \in \{-5, 0, 5\}$ and $\beta = 1$ (from top to bottom).	351
4.18 Predicted probability of voting for a republican given different levels of conservatism using logistic regression.	353
4.19 Marginal effect of conservatism on the predicted probability of voting for a republican given different levels of predicted probability.	358
4.20 Discrete marginal effect of conservatism on the predicted probability of voting for a republican given different levels of predicted probability.	361
4.21 Plot of the predicted probability of switching wells from the logistic regression given the distance from the nearest safe well (in meters).	364
4.22 Histogram of distances to the nearest safe well, measured in meters.	365
4.23 Histogram of arsenic levels (micrograms/L) measured in the household's well.	367
4.24 Plot of the predicted probability of switching wells from the logistic regression given the fitted linear predictors $\hat{\eta}(x)$, where $\hat{\eta}(x)$ includes arsenic levels (in micrograms/L) and distance from the nearest safe well (in meters).	369
4.25 Plot of the predicted probability of switching wells from the logistic regression given the distance from the nearest safe well (in meters). The black curve shows the relationship between distance and probability of switching wells at the mean of arsenic levels (1.66 micrograms/L). Dashed red curves show the distance and probability relationship for increasing levels of arsenic (0.5, 3.5, 6.5, and 9.5 micrograms/L), with darker red lines indicating greater levels of arsenic.	371
4.26 Plot of the predicted probability of switching wells from the logistic regression given the distance from the nearest safe well (in meters). Black curve shows the relationship between distance and probability of switching wells at the mean of arsenic levels (1.66 micrograms/L). Dashed red curves show the distance and probability relationship for increasing levels of arsenic, evaluated at the deciles of the arsenic levels.	373

4.27 Plot of the predicted probability of switching wells from the logistic regression given the distance from the nearest safe well (in meters). Black curve shows the relationship between distance and probability of switching wells at the mean of arsenic levels (1.66 micrograms/L). Dashed red curves show the distance and probability relationship for increasing levels of arsenic, evaluated at the deciles of the arsenic levels, including an interaction between arsenic and distance.	376
4.28 Plot of the predicted probability of switching wells from the logistic regression given arsenic levels (micrograms/L), with distance from the nearest safe well, education, and membership in community associations held at their means.	378
4.29 Calibration plot of empirical relative frequencies of switching and predicted probability of switching for 10 deciles of predicted probabilities.	382
4.30 Logistic (red), probit (blue), and Stukel's approximation to the probit (dashed blue) ($\alpha_1 = \alpha_2 = 0.165$).	385
4.31 Logistic (red), complementary log-log (blue), and Stukel's approximation to the complementary log-log (dashed blue) with $\alpha_1 = .62, \alpha_2 = -.037$	387
4.32 Component plus residual plots for arsenic levels from the logistic regression model predicting decision to switch wells.	391
4.33 Component plus residual plot with Pearson residuals for arsenic levels from the logistic regression model predicting decision to switch wells.	393
4.34 Partial residual plots for distance from the nearest safe well and arsenic levels from the generalized additive model predicting decision to switch wells.	395
4.35 Partial residual plots for distance from the nearest safe well and arsenic levels from the generalized additive model predicting decision to switch wells.	397
4.36 Partial residual plots for distance from the nearest safe well and arsenic levels from the generalized additive model predicting decision to switch wells.	399
4.37 (a) Complete separation, (b) quasi-complete separation, (b') quasi-complete separation, (c) overlap. Figure taken from Albert, A., & Anderson, J. A. (1984). On the existence of maximum likelihood estimates in logistic regression models. <i>Biometrika</i> , 71(1), 1-10.	402

4.38	Receiver Operating Characteristic (ROC) curve showing true positive rates against false positive rates for several cutoff points from the logistic regression predicting voting for the republican candidate for governor depending on conservatism score.	405
4.39	Plots of the estimated probabilities for each observation, estimated probabilities by outcome, and ROC curve. From Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). <i>Applied Logistic Regression</i> . John Wiley Sons.	407
4.40	Cross-validated Receiver Operating Characteristic (ROC) curve showing true positive rates against false positive rates for several cutoff points from the logistic regression.	410
4.41	Predicted probabilities from logistic regression with 95% confidence bands as dotted lines.	416
5.1	1000 observations clustered within 10 observational units (groups). Each group has a random intercept. Observations within each group vary randomly around that intercept.	422
5.2	1000 observations clustered within 5 observational units (groups). Each group has a random slope, but same intercept. Observations within each group vary randomly around that slope.	425
5.3	1000 observations clustered within 5 observational units (groups). Each group has both a random intercept and random slope. Observations within each group vary randomly around that group-level intercept and slope.	427
5.4	Radon levels (in picoCuries/L) in sampled Minnesota homes.	429
5.5	Log likelihood of the λ parameter for the Box-Cox transformation of the measured home radon levels.	430
5.6	Histogram of log transformed radon activity and radon activity with $\lambda = -0.2$ Yeo-Johnson Power transformation.	432
5.7	Quantile-Quantile plot of log-radon activity levels and $\lambda = -0.2$ transformed radon activitiy levels versus the normal distribution.	434
5.8	Boxplots of log radon activity in each county. Horizontal line is the overall mean log radon activity, ignoring county information.	437
5.9	Boxplots of log radon activity in each county. Horizontal line is the overall mean log radon activity, ignoring county information.	439
5.10	100 observations clustered within 10 observational units (groups). Each group has a random intercept. Observations within each group vary randomly around that intercept. The plot on the right shows the complete pooled intercept and 95% confidence intervals for the intercept are shown in red. The actual population intercept is in blue, and in this sample lies outside the confidence interval.	445

5.11	Boxplots of log radon activity in each county. Horizontal line is the complete pooling overall mean log radon activity, ignoring county information.	446
5.12	Plot of mean log radon activity in each county (y-axis) against the sample size in each county (x-axis), with ± 1 standard error bars shown. Lac Qui Parle county is circled.	449
5.13	Top plots have low intercept variance $\sigma_\alpha^2 = 1$, while bottom plots have high intercept variance $\sigma_\alpha^2 = 4$. Plots on the left have low within-cluster variance $\sigma_y^2 = 1$, while plots on the right have high within-cluster variance $\sigma_y^2 = 4$	453
5.14	Plot of log radon activity in six counties by whether the measurements were taken in the basement or first floor of the house (x-axis). Complete pooling regression is shown on the dashed line, whereas the no pooling regression line, with different intercepts for each county, is shown in the solid line.	458
5.15	Plot of log radon activity in six counties by whether the measurements were taken in the basement or first floor of the house (x-axis). Complete pooling regression is shown on the dashed line. No pooling regression line, with different intercepts for each county, is shown in the solid line. Partial pooling line, with varying intercepts by county, is shown in the red dotted line.	461
5.16	Plot of partially pooled intercept estimate of log radon activity in each county (y-axis) against the sample size in each county (x-axis), with ± 1 standard error bars shown. Lac Qui Parle county is circled.	465
5.17	County-level log radon measurements plotted against county-level log uranium measurements.	468
5.18	Partially pooled log radon intercept for each county from the multi-level model plotted against county-level log uranium in the soil.	471
5.19	Two levels of a multi-level model. The level 1 regression (top) makes predictions about log radon levels at the level of individual observations. The level 2 regression (bottom) makes predictions about county-level log radon levels from county-level predictors.	474
5.20	Level 1 model of log radon activity in six counties. Complete pooling regression is shown on the dashed line. No pooling regression line is shown in the solid line. Partial pooling varying-slope varying intercept is shown in red, with county-level uranium as a predictor of the intercept and slope.	479
5.21	Level 2 model of partially pooled county-level intercepts (top) and slopes (bottom) plotted against county-level log uranium in the soil.	481

5.22 Level 1 Least Squares (LS) jackknife residuals from the complete pooling regression. The top left plot shows the jackknife residuals versus fitted values. The top right plot shows the jackknife residuals versus floor of radon level measurement (0 = basement, 1 = first floor). The bottom left plot shows the jackknife residuals versus log uranium levels. The bottom right plot shows the jackknife residuals against the quantiles of the t distribution.	495
5.23 Level 1 Least Squares (LS) jackknife residuals from the complete pooling regression. The top left plot shows the jackknife residuals versus fitted values. The top right plot shows the jackknife residuals versus floor of radon level measurement (0 = basement, 1 = first floor). The bottom left plot shows the jackknife residuals versus log uranium levels. The bottom right plot shows the jackknife residuals against the quantiles of the t distribution.	497
5.24 Level 1 Least Squares (LS) jackknife residuals from the complete pooling regression using the $\lambda = -0.2$ Box-Cox transformation on a house's measured radon activity. The top left plot shows the jackknife residuals versus fitted values. The top right plot shows the jackknife residuals versus floor of radon level measurement (0 = basement, 1 = first floor). The bottom left plot shows the jackknife residuals versus log uranium levels. The bottom right plot shows the jackknife residuals against the quantiles of the t distribution.	499
5.25 Level 1 Least Squares influence index plots. The top plot shows Cook's Distances, followed by the jackknife residuals, then the Bonferroni adjusted p-values, then the hat values for each observation.	501
5.26 Level 2 Empirical Bayes (EB) residual histograms.	503
5.27 Level 2 Empirical Bayes (EB) residual plots. The plots on the left show the relationship between the model's residuals and fitted values (top) and normal distribution (bottom) for county-level intercepts. The plots on the right show the relationship between the model's residuals and fitted values (top) and normal distribution (bottom) for county-level slopes.	505
5.28 Cook's Distance as a function of deleted counties.	507

Part I

Ethics and Reproducibility

Ethical Problems in Data Analysis

I'd like you to get two things out of this class: 1) An understanding of how to critically evaluate the data analyses of others, and 2) an understanding of how to do this for your own work.

For the first goal, as you will see, reported data analyses often do not match what was actually done. This is rarely discussed in statistics classes, that focus on mathematics and technical details, but ignore the common problems researchers face when conducting and communicating their analyses.

Unfortunately, serious errors are often overlooked. Because of this I find it difficult to believe published results that I cannot check and reproduce. Being able to do this depends on authors providing data and comprehensible code.

For the second part, I hope that you learn how to be clear and transparent in your work, and make best efforts to minimize errors. Others should be able to reproduce what you've done. You should understand your models, know how to check them, and know their limitations. Above all, your analyses should be clearly reported to those reading your papers.

Chapter 1

Research Ethics

Researchers don't always conduct and report data analyses in an ethical manner. For the purpose of this class, "ethical data analysis" broadly means *not deceiving one's audience with statistics*. Such deception can occur unintentionally in at least two ways: 1) Lack of awareness about bad practices, and 2) unknowingly convincing ourselves that bad behavior is good because the bad behavior yields a reward. Intentional deception also occurs, usually because someone wants to get ahead, or is afraid of falling too far behind.¹ Scientific organizations only handle clear cut cases of intentional deception, called *scientific misconduct*: Someone misrepresented their data (falsification) or made up data out of thin air (fabrication).² They leave the grey area to teachers and mentors. Intentional ethical violations are very serious, but do not represent the challenges researchers usually face.

1.1 Feynman's Rule

If there were one guiding principle to help researchers navigate the challenges of ethical data analysis, it was probably best stated by the well-known physicist Richard Feynman. Simply put, when conducting and communicating data analyses, scientific integrity entails ([Feynman, 1974](#)):

"bending over backwards to show how you're maybe wrong."

The general principle is to work very hard to show how we could be wrong when presenting our data analyses to ourselves and others. Although such a general principle is nice, it is also helpful to think about some specific,

¹This is called falsification, fabrication, and plagiarism: http://www.nap.edu/openbook.php?record_id=12192&page=15.

²If you want to see what some of these clear cut cases are like, check out the Office of Research Integrity, which regularly releases public reports: <http://ori.hhs.gov/>, here's the most recent one <https://ori.hhs.gov/content/case-summary-xiao-dong>.

recurring cases where one might fail to “bend over backwards.” This applies to both science and data analysis. Your advisors will help you make sure the science is right. In the first part of this class you will learn how to make sure your data analyses are right; that you aren’t fooling yourself, or anyone else, with bad statistics.

1.2 The Nine Circles of Scientific Hell

An article by the Neuroskeptic called “The Nine Circles of Scientific Hell” presents a humorous perspective on the different ways researchers can fail ethically when handling and analyzing data (Neuroskeptic, 2012). Figure 1.1 shows these nine circles.

1.2.1 Peer-Review

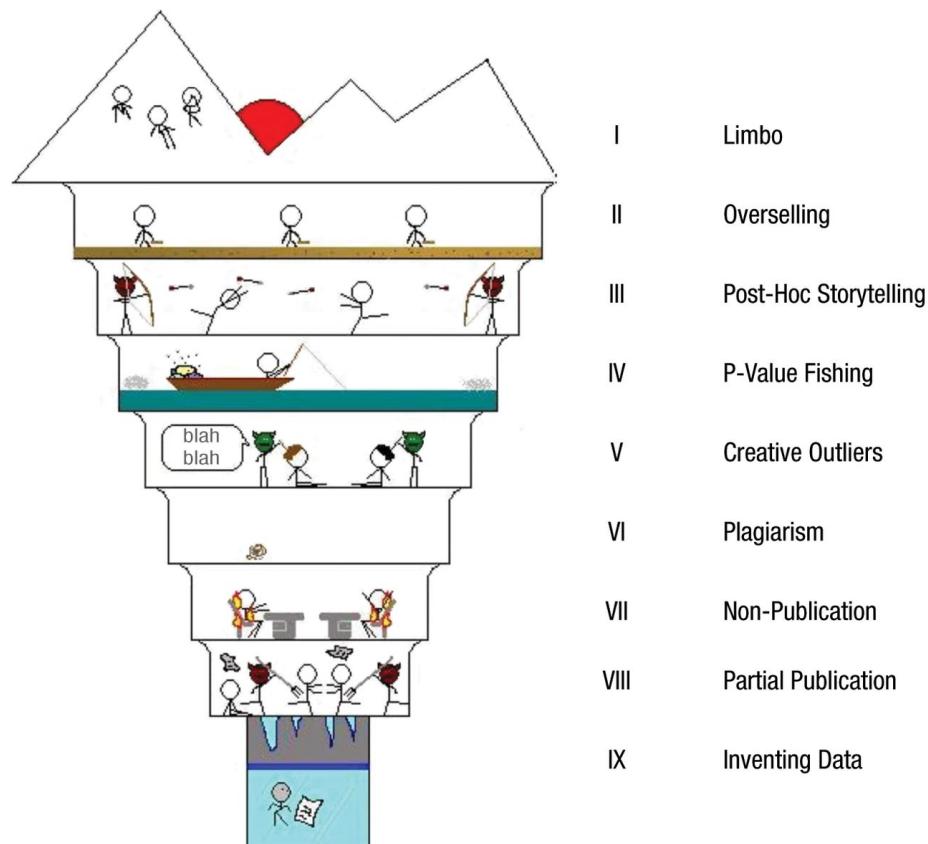
The first circle of scientific hell is one familiar to most researchers. In addition to being a venue for trashing other peoples’ results based on our own theoretical perspective (Mahoney, 1977), peer-review offers researchers the opportunity to overlook serious flaws in methods, especially abuses of statistics.

While turning a blind eye to statistical malfeasance likely has a large *quid pro quo* element, many oversights probably occur because pointing out sloppy statistics incurs retaliation. For example, Andrew Gelman received some unfriendly emails after criticizing the statistics in two articles published in preeminent psychology journals (Gelman, 2013). The unfriendly professor asked Gelman why he hadn’t first contacted the authors to discuss his critiques, to which Gelman replied that published reports, as part of the public record, are meant to be scrutinized publicly.

Here’s my answer to the problem. As a peer-reviewer it is possible to ask authors to present *all* the raw data they collected for a project, as well as provide a reproducible description of their statistical approach. If the authors provide such information (which many journals are now requiring pre-publication) then it is easy to replicate and check their analyses. If not, additional tests, such as Uri Simonsohn’s p-curve,³ can test whether the statistical evidence presented in a report is too good to be true (i.e., whether the authors are hiding data).

³<http://www.p-curve.com/>

Fig. 1. The nine circles of scientific hell (with apologies to Dante and xkcd).



Neuroskeptic Perspectives on Psychological Science
2012;7:643-644

Perspectives on
Psychological
SCIENCE

1.2.2 Overselling

To get grants and papers, researchers sometimes feel pressure to make their results seem more important than they really are. An experiment produces a small effect, but through lack of transparency and hand-waving, one can pretend that the result, however tiny or ephemeral, is worthy of millions of dollars of funding.

Putting results in the proper context should help avoid landing us in this circle. Measures of effect-size and model checks help us evaluate the magnitude and limitations of our findings. Attention to internal and external validity help with causality and scope. We will cover these topics in Chapters 2 and 3.

1.2.3 Post-hoc Storytelling

Post-hoc stories are constructed to explain unexpected data after the fact. While this isn't problematic itself, pretending that the results were expected all along is misleading. This is a communication problem: The author deceives readers by making them believe that surprising results were expected even when they weren't.

Unexpected results can easily be recognized as such, and clearly separated in a report from pre-planned analyses. There are more extreme approaches. For example, it is possible to write code before seeing the data, then pass the code to a 3rd party to execute once the data are obtained. A second approach, used by biomedical researchers, is to require all data and protocols that will be used in published papers to be submitted to an online repository *prior* to analysis. This is what the FDA does with ClinicalTrials.gov ([Tse et al., 2009](#)). *Blind data analysis*, where the data analyst does not know who the treatment and control groups are, can also help, as having two different people analyze the same dataset can lead to different conclusions ([Gøtzsche, 1996](#)). I believe some economics and political science journals are now trying to encourage people to publish their data analysis plans ahead of time and conduct blind data analyses.

1.2.4 p-value Fishing

P-value fishing (sometimes called p-hacking), is dropping data, running many experiments, or running many different statistical models until a particular result is obtained in the form of $p < .05$. Because of this, p-values are invalid in all but the most constrained, pre-planned analyses. As put by Ed Leamer ([Leamer, 1983](#)):

“The econometric art as it is practiced at the computer terminal involves fitting many, perhaps thousands, of statistical models.

One or several that the researcher finds pleasing are selected for reporting purposes. This searching for a model is often well intentioned, but there can be no doubt that such a specification search invalidates the traditional theories of inference.”

P-hacking misrepresents our statistical analyses by taking advantage of random fluctuations in the data that yield spurious correlations. The problem has plagued researchers in economics, psychology, and biomedical research for decades. The use of the p-value as *the* criterion for scientific significance has been the source of controversy since its inception, and as we will soon see, it is possible to get a statistically significant p-value to support pretty much any hypothesis if we are creative enough ([Simmons et al., 2011](#)).

One way to protect against this is to separate planned from exploratory analyses. This means having a data analysis plan set out ahead of time. A second method is to use cross-validation to evaluate models that have been extensively revised based on the data we have at hand.

1.2.5 Creative Outliers

Outliers are serious problems for many models, especially the most common: Ordinary Least Squares (OLS). Because outliers cause problems, we often want to get rid of them, or minimize their impact. However, whether something is an outlier is not clearly defined, and in fact there are multiple possible definitions and ways of handling them. Once we start dredging through data to separate “good” from “bad” observations, we’ve lost our way to valid statistical inferences.

It is important to separate tests and diagnostics for outliers with procedures for handling them. Tests and diagnostics all have assumptions and limitations, and there is no single algorithm for how to handle outliers appropriately. Different types of outliers have different implications for our analyses, and can be handled in different ways. The most important feature of handling outliers well is to honestly report how they were identified, exactly what was done to address the issue, and the limitations of that approach. I personally prefer retaining outliers to convey the actual variability in the data, then compare usual methods (e.g., least squares regression) to robust alternatives (e.g., median regression).

1.2.6 Non and Partial Publication

The seventh and eighth circles apply to research that has multiple outcomes, experiments, and treatment variables within experiments. It is often too tempting for researchers to exclude variables, experimental treatments, or only reporting some outcome variables that are statistically significant. The

more outcome measures we take, the more variables we look at, and the more experiments we conduct, the greater chance we have for creating a rosy but false story. Some of the most robust evidence comes from comparing antidepressant clinical trials that were submitted to the FDA with those that were published (Turner et al., 2008).

In Gelman's paper (Gelman, 2013) there is a nice example of researchers (Nosek et al., 2012) who initially found a result that washed away when they replicated it, and the dilemma they faced about deciding whether to report the failure to replicate (thus scuttling the junior author's job market paper). Usually what happens is that such failures to replicate are not reported, making the reported statistical evidence look better than reality. Such reporting failures are often called the *file-drawer problem*, *publication bias*, or *selective outcome reporting*, although the three terms have slightly different meanings.

1.3 How to Get $p < .05$: Monte Carlo Examples

As mentioned, one of the biggest statistical controversies across the sciences has to do with the use and abuse of p-values. Specifically, researchers are very concerned about rejecting a “null hypothesis” (e.g., that the difference in methane leakage rates between natural gas power plants in Pennsylvania and Illinois is zero) when it is actually true (a *false positive*). That is, a false positive would be concluding that methane leakage rates are higher in Pennsylvania than Illinois when the rates are actually the same.

In this section we introduce the issue and R using some simple simulations. Simmons et al. (2011) provide a number of examples of ways to make results look statistically significant (i.e., $p < .05$) even when there is no real difference between the populations we are comparing.

1.3.1 Combining Dependent Variables

In their first example (“Situation A”), Simmons et al. (2011) conduct three t-tests but present the results as if they conducted only one. In their story the two dependent variables are a “liking” judgment of a product and a willingness to pay (WTP). These two dependent variables are correlated, with a Pearson’s r of 0.5 (also called the Pearson Product-Moment Correlation). Recall that:

$$\text{Pearson's } r = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}} \quad (1.1)$$

Where the sample covariance estimator is:

$$\text{Cov}(X, Y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{n - 1} \quad (1.2)$$

which measures to what degree x and y are simultaneously above or below their means. The sample variance estimator for X is:

$$\text{Cov}(X, X) = \text{Var}(X) = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1} \quad (1.3)$$

which measures how far away x is on average from its mean \bar{x} . The three t-tests compare the two dependent variables and their average, then only report whether at least one test is significant. The authors show that this inflates the number of times we erroneously conclude that the results are statistically significant even when there is no difference between populations. Let's work through the example in R. First, let's create the dependent variables for the first group, Group 1:

```
#Group 1
Liking1 <- rnorm(20, 0, 1) # Generate normal RV with mean = 0, sd = 1
WTP1 <- rnorm(20, 0.5*Liking1, 1) # Generate normal RV correlated with Liking
AVG1 <- (Liking1 + WTP1)/2 # Average them
```

Note: You can see what these objects look like by typing their name into the console. You should do this often to make sure you haven't messed something up.

```
Liking1 # See that Liking1 is a standard normal vector of length 20
```

As you can see we've randomly generated 20 observations for the Group 1 "Liking" ratings from a normal distribution with mean zero and standard deviation 1.⁴ We've done essentially the same thing for the WTP judgments, but have added a correlation of 0.5 with the Liking ratings. Finally, we compute the third dependent variable, which is the average of Liking and WTP. We then repeat this for Group 2:

```
# Group 2
Liking2 <- rnorm(20, 0, 1)
WTP2 <- rnorm(20, 0.5*Liking2, 1)
AVG2 <- (Liking2 + WTP2)/2
```

⁴Note that we usually think of normal random variables in terms of mean and variance, so be careful to put the standard deviation, not variance into the simulation.

The only thing that is different here are the labels (Liking1 goes to Liking2, etc). Notice that Liking1 and Liking2 are drawn from the *exact same distribution*, that is $N(0, 1)$, so t-tests should only be significant 5% of the time. Next, we conduct t-tests comparing the three dependent variable combinations:

```
#Two-Sample t-tests
t.test1 <- t.test(Liking1, Liking2, var.equal = TRUE) # Assumes equal var
t.test2 <- t.test(WTP1, WTP2, var.equal = TRUE)
t.test3 <- t.test(AVG1, AVG2, var.equal = TRUE)
```

Note that we are conducting two-sample t-tests which assume that the dependent variables from the two different groups (e.g., Liking1 and Liking2) are independent (which is the case for our simulation) and identically distributed (with equal variances), and a whole bunch of other stuff (e.g., randomly sampled from a well-defined population) that, as we will see throughout the course, are almost never met in applied data problems. For now, suspend disbelief. As [Draper et al. \(1966\)](#) put it, a t-test is:

$$t = \frac{(\text{Estimate of thing}) - (\text{Postulated test value of thing})}{(\text{Standard error of estimate of thing})} \quad (1.4)$$

Intuitively, the t-test formula is telling us how big our estimate of the *difference* in population means is relative to the *uncertainty* in our estimate of this difference (where the denominator represents the uncertainty about this difference, aka the standard error of the estimate of the difference in means). Keep in mind that the sample estimate of the mean is a random variable, in that it will take on different values depending on which elements of the population are sampled. Thus, for a fixed sample size n , the sample mean could take on a number of different values. The variation in the sample mean of size n that would occur in repeated sampling is variance of the estimate, and the square root of that is the standard error of the estimate. The formula for the two-sample t-test is (see [Wackerly et al. \(2008\)](#) Ch. 7.1, 7.2, 8.8, and 10.8 for a review):

$$T = \frac{\bar{Y}_1 - \bar{Y}_2}{S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \quad (1.5)$$

where S_p is:

$$S_p = \sqrt{\frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}} \quad (1.6)$$

and S_1, S_2 are the sample standard deviations for groups 1 and 2 respectively,⁵ and \bar{Y}_1, \bar{Y}_2 are the sample means. The t-test has a Student's t

⁵ $S^2 = \frac{1}{n-1} \sum_{i=1}^n (Y_i - \bar{Y})^2$. The reason we divide by $n - 1$ is shown in the mathematical appendix.

distribution with $n_1 + n_2 - 2$ degrees of freedom. The pooled sample standard deviation is just the average of the sample standard deviations for each group, weighting each one by their sample size. This tends to give more precise estimates of the population standard error than using the sample standard deviation from a single group.

Finally, we create an indicator variable⁶ that tells us if *at least one* of the t-tests is statistically significant at the .05 level (i.e., $p < .05$):

```
#signif = 1 if at least one test gives p < .05, 0 otherwise
signif <- ifelse(t.test1$p.value < .05 | #Vertical bar is logical "or"
                  t.test2$p.value < .05 |
                  t.test3$p.value < .05,
                  1, # value of signif if condition is met
                  0) # value of signif if condition not met
```

Now we want to repeat this 15,000 times. We can put it all together in a function that takes the p-value (e.g., .05) as a cutoff and outputs *signif*:

```
SituationA <- function(pval){
  #Group 1
  Lik1 <- rnorm(20, 0, 1) # Generate normal RV with mean = 0, var = 1
  WTP1 <- rnorm(20, 0.5*Lik1, 1) # Generate normal RV correlated with Lik1
  AVG1 <- (Lik1 + WTP1)/2 # Average them

  # Group 2
  Lik2 <- rnorm(20, 0, 1)
  WTP2 <- rnorm(20, 0.5*Lik2, 1)
  AVG2 <- (Lik2 + WTP2)/2

  #Two-Sample t-tests
  t.test1 <- t.test(Lik1, Lik2, var.equal = TRUE) # Two-sample t-test
  t.test2 <- t.test(WTP1, WTP2, var.equal = TRUE)
  t.test3 <- t.test(AVG1, AVG2, var.equal = TRUE)

  #signif = 1 if at least one test gives p < pval, 0 otherwise
  signif <- ifelse(t.test1$p.value < pval | #Vertical bar is logical "or"
                  t.test2$p.value < pval |
                  t.test3$p.value < pval,
                  1, 0)
  return(signif) #value returned by the function
}
```

⁶An indicator variable is a numeric variable that is equal to 1 if some condition is met, zero otherwise

We can then replicate this function 15,000 times and calculate the mean to tell us the proportion of times the null hypothesis was rejected ($p < .05$):

```
replicates <- replicate(15000, SituationA(.05))
SitASim <- mean(replicates)
```

Because the process uses random number generation, we won't get the same result each time. However, this time I get about 9.9% rejection of the null hypothesis when we should only get 5%. The 9.9% seems reasonably close to the 9.5% reported in the paper.

1.3.2 Using “controls”

The third example in [Simmons et al. \(2011\)](#) involves “controlling” for different covariates, such as gender or an interaction with gender, in a regression. We repeat a similar process as before, but now include two additional elements: 1) A binary regressor for gender, and 2) an indicator variable for treatment group:

```
Liking1 <- rnorm(20, 0, 1)
Gender1 <- rbinom(20, 1, .5) # Flip a coin with 50% chance of being female
Treat1 <- rep(0, 20) # Repeat "0" 20 times; indicates the control group
```

We can now create a data frame with the 20 observations as different rows and their values on Liking, Gender, and Treat as columns. A data frame is like a matrix that can handle mixed data types (e.g., numeric, factor) and meta-data such as the names of rows and columns:

```
# Data frame with observations as rows
# Liking, gender, and treatment are columns
Group1 <- data.frame("Liking" = Liking1, "Gender" = Gender1, "Group" = Treat1)
```

We can do the same for the second group, the “treatment” group:

```
Liking2 <- rnorm(20, 0, 1)
Gender2 <- rbinom(20, 1, .5)
Treat2 <- rep(1, 20)
Group2 <- data.frame("Liking" = Liking2, "Gender" = Gender2, "Group" = Treat2)
```

We can now merge the two data frames by stacking Group 1 on top of Group 2:

```
#Merge the two data frames by stacking Group 1 on top of group 2
#"r" in rbind is for "row"
all.data <- rbind(Group1, Group2)
```

The result is a single data frame with all 40 observations as rows and their values on the three variables as columns. The three analyses were a simple t-test between conditions, a regression with a gender main effect, and a regression with a gender by group interaction:

```
t.test1 <- t.test(Liking1, Liking2, var.equal = TRUE)
# A regression with Group and Gender main effects
reg1 <- lm(Liking ~ Group + Gender, data = all.data)
# A regression with both main effects and interaction
reg2 <- lm(Liking ~ Group + Gender + Group*Gender, data = all.data)
```

As you can see, the estimate of the group effect changes in each of the three analyses. In particular, the third analysis (main effects and interaction regression) has *two* chances for the Group variable to be significant.

Again, we can see whether at least one of these analyses rejects the null hypothesis and gives us a p-value less than .05 for Group. To do this we need to pull out the p-value for the regressions, which is a bit complicated, but don't worry too much about it now:

```
reg1.p <- summary(reg1)$coefficients[2, 4]
reg2.p1 <- summary(reg2)$coefficients[2, 4]
reg2.p2 <- summary(reg2)$coefficients[4, 4]
```

The summary function provides us information about the regression. *Coefficients* is a matrix of information about each regression including the point estimate, standard error, t-value, and p-value for each regressor. We can pull the p-value out of this matrix, then test whether at least one is significant:

```
signif <- ifelse(t.test1$p.value < .05 |  
                  reg1.p < .05 |  
                  reg2.p1 < .05 |  
                  reg2.p2 < .05,  
                  1, 0)
```

Again, we need to put it all together in a function that can be repeated:

```
SituationC <- function(pval){  
  
  Liking1 <- rnorm(20, 0, 1)  
  Gender1 <- rbinom(20, 1, .5)  
  Treat1 <- rep(0, 20)  
  Group1 <- data.frame("Liking" = Liking1, "Gender" = Gender1, "Group" = Treat1)  
  
  Liking2 <- rnorm(20, 0, 1)  
  Gender2 <- rbinom(20, 1, .5)
```

```

Treat2 <- rep(1, 20)
Group2 <- data.frame("Liking" = Liking2, "Gender" = Gender2, "Group" = Treat2)

all.data <- rbind(Group1, Group2)

t.test1 <- t.test(Liking1, Liking2, var.equal = TRUE)
reg1 <- lm(Liking ~ Group + Gender, data = all.data)
reg2 <- lm(Liking ~ Group + Gender + Group*Gender, data = all.data)

reg1.p <- summary(reg1)$coefficients[2, 4]
reg2.p1 <- summary(reg2)$coefficients[2, 4]
reg2.p2 <- summary(reg2)$coefficients[4, 4]

signif <- ifelse(t.test1$p.value < pval |  

                    reg1.p < pval |  

                    reg2.p1 < pval |  

                    reg2.p2 < pval,  

                    1, 0)

return(signif)
}

```

Finally, we replicate this 15,000 times (this takes a minute):

```

replicates <- replicate(15000, SituationC(.05))
SitCSim <- mean(replicates)

```

I get about 12% rejections of the null hypothesis, close to the 11.7% reported in [Simmons et al. \(2011\)](#). So what's the upshot? “Controlling” for things provides an immense amount of flexibility to get statistically significant results when no effect exists (the null hypothesis is true). This problem dramatically increases in real-world problems where there are far more things to “control” for than a single regressor (in our case gender).

How to Choose Controls

A common question that I get from students is “what controls [or covariates] should I include in my data analysis?” This question usually comes up *after* the student has tried a number of different approaches, or even used algorithms for covariate “selection,” and confusing results came up. This is *very bad!*

Here’s my solution. *Before* doing any regressions, first come up with reasons based on theory, prior evidence, or anything substantial about why a variable should be included in a regression. There should actually be a good

reason to include the variable. Then do the following: 1) Conduct a regression with the variable of interest only, 2) look at the correlation matrix between the theoretically relevant variables and your variable of interest, 3) conduct a regression with only these theoretically relevant variables. As those two regressions are pre-planned, the p-values are valid. Report both regressions in a table (economists do this well), and check them graphically (I'll show you how to do this in the next two chapters). If the results are unexpected conduct follow-up regressions trying to figure out what is going on. *These must be clearly communicated as exploratory regressions*, and both you and the reader should be aware that the p-values are invalid (as we just demonstrated).

1.3.3 Why hypothesis tests measure sample size

A student asked a great question when I said that hypothesis tests are roughly a measure of sample size. The intuitive explanation that I gave in class was that the t-test is the estimate of the population mean difference (i.e., $\bar{Y}_1 - \bar{Y}_2$) divided by the standard error of the estimate ($S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$). The standard error just tells us how uncertain we are about the population mean difference that we are trying to estimate. In finite samples, as we get closer to sampling the entire population, the uncertainty about the population means decreases to 0 (i.e., when we sample the entire population, we *know* the population means, so there is no uncertainty about their values), so as the sample size n approaches the population size N , the t-test will approach infinity. This also holds for populations of infinite size, except that the t-test never degenerates by dividing by zero (just approaches in the limit as n goes to infinity).

The mathematical explanation is that unless $\mu_1 = \mu_2$ exactly, then the T statistic is increasing without bound as the sample size increases. Just consider taking the limit as n goes to infinity of the t-test statistic:

$$T = \frac{\bar{Y}_1 - \bar{Y}_2}{S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \quad (1.7)$$

The n_1 and n_2 end up making the denominator really small as the total sample size n increases. You can also see this with simulation. All you need to do is make the mean of the normal distributions trivially different between Liking1 and Liking2, and there is always some sample size where you will reject the null hypothesis with high probability. The following function takes the total sample size n as an input and outputs whether the result is significant. We can replicate this a number of times with different levels of n to show that the probability of rejecting the null hypothesis increases to 1 as n increases:

```
samplesize <- function(n){
```

```

Liking1 <- rnorm(n/2, 0, 1)
Liking2 <- rnorm(n/2, .01, 1) # the mean difference is 1/100 of the variance
t.test1 <- t.test(Liking1, Liking2, var.equal = TRUE)
signif <- ifelse(t.test1$p.value < .05,
                  1, 0)
return(signif)
}
replicates <- replicate(100, samplesize(200000))
sampsizeSim <- mean(replicates)

```

As you increase the sample size, the rejection of the null hypothesis increases in probability, even with a trivially small effect. Thus, in some sense “big data” is another way of saying “small effects.” The converse is also true: We won’t be able to detect huge effects with very small sample sizes. As a result, thinking about the size and uncertainty of the effect give us much more information than the p-value, which does not disentangle the two concepts.

To disentangle these ideas, you need three pieces of information, not one (the p-value): 1) the estimated size of the population mean difference ($\bar{Y}_1 - \bar{Y}_2$), 2) the pooled sample standard deviation $S_p = \sqrt{\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2}}$, and 3) the sample size $n = n_1 + n_2$. With this information, you can calculate an estimate of the *effect size*, which is a unitless measure of the standardized mean difference:

$$d = \frac{\bar{Y}_1 - \bar{Y}_2}{\sqrt{\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2}}} \quad (1.8)$$

This is sometimes called Cohen’s d , named after the psychologist Jacob Cohen, who, to demonstrate the folly of neglecting effect size, humorously wrote a paper entitled “The earth is round: $p < .05$ ” (Cohen, 1994). As n increases d approaches $\frac{\mu_1 - \mu_2}{\sigma}$, the population standardized mean difference. Unlike the t-statistic (and p-values), it does not increase (or decrease) as a function of the sample size.

1.4 Reproducibility

Ultimately, you, and you alone, are responsible for the accuracy of your statistical work. Thus, it is your responsibility to bend over backwards in the way Feynman discusses. This applies to data collection, pre-processing, analysis, and reporting. Unless each element is documented, other people won’t be able to reproduce what you’ve done. Unless you start documentation from the beginning, it’s unlikely that you’ll be able to reconstruct what you’ve done.

There are definitely a lot of ways to go wrong. Awareness and moral commitment help, but we need some tools. Our first tool to defend against bad data analysis is *reproducibility* (Stodden, 2009, 2011). Reproducible results show the reader how we went from raw data to the final reported result.

To make sure you learn reproducibility, all of your assignments must include reproducible code. I encourage you to make your work available through a link at Harvard's Dataverse (King, 2007), or any other public data/code repository.⁷ For assignments I will give you the raw data, and you should turn in a well-written report including clear instructions on how to get from the raw data to a specified result (e.g., calculating the mean of a column). The reproducibility element of the class should help you get into good data analysis habits, as well as ensure a minimal level of data ethics. All work should follow the best practices for publishing computational results⁸ and the Google style guide for R.⁹ *You will lose points if you do not adhere to these practices.*

1.5 Technical Problems with Applied Data Analysis

The Nine Circles don't really cover the technical issues that arise in applied data analysis, instead focus on ethics and temptation. But technical issues do arise, so frequently that data analytics may be considered a profession more scandalous than politics. As Richard Berk puts it (Berk, 2004):

“There will often be no clear understandings about how the data came to be. A common practice then is to construct some convenient account giving the investigator a free hand. But if the inputs are a matter of convenience, so are the outputs. Another approach is to note those features of the data generation process that are important and then simply list the assumptions that will allow the analysis to proceed. Such “assume-and-proceed” statistics allow the investigator to meet the obligation of full disclosure as if that were sufficient to justify completely all that follows. Finally, there are a great many regression analyses that reflect only the dimmest understandings about what can be

⁷Gary King has a bunch of additional information about replication on his website:
<http://gking.harvard.edu/replication.shtml>

⁸http://wiki.stodden.net/Best_Practices_for_Researchers_Publishing_Computational_Results

⁹<http://google-styleguide.googlecode.com/svn/trunk/Rguide.xml#todo>

learned from the data alone. Anything goes as long as SPSS or SAS can be made to run.”

Part II of this class, which focuses on the technical aspect of data analysis, aims to help you avoid these bad habits. Unfortunately, we do not have the time to go into the depth necessary to cover all of the details for all of the methods we cover. I strongly suggest that if you decide to use a method discussed in this class, first pick up a book (listed in the bibliography) and read some of the relevant papers (referenced in these notes and optional readings on Blackboard). Be sure you know exactly what you are doing before proceeding with an analysis, and if you have questions there are plenty of graduate students and faculty (including me) who can help.

1.6 Homework 1

The first homework helps you get set up with R so that you can start making reproducible code. There are a number of sources to help you get started in R. Here's a few:

- <http://cran.r-project.org/doc/manuals/R-intro.html>
- <http://www.statmethods.net/>
- <http://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf>
- http://www.burns-stat.com/pages/Tutor/R_inferno.pdf
- <http://cran.r-project.org/doc/contrib/usingR.pdf>
- Other students

Thanks to the TA Jeff Anderson for providing links on CMU courses related to R and statistical computing:

- <http://www.andrew.cmu.edu/user/achoulde/94842/>
- <http://www.stat.cmu.edu/~cshalizi/statcomp/14/>

The first assignment is to go through one of the R tutorials and then write a simulation in R that replicates Situation B from [Simmons et al. \(2011\)](#). After writing up the simulation, answer these questions in a brief paragraph each:

- Do you agree with Feynman's Rule? Is it practical?
- Which of the Neuroskeptic's 9 circles are most relevant for your field? Why, and what corrective policies do you think are necessary (if any)?
- Does anyone ever believe the null hypothesis?
- When are the results of a t-test meaningless? In other words, what additional information do you need to make sense of the results of a t-test?
- Without doing any math, which would yield more bias in [Simmons et al. \(2011\)](#) Situation A: Two dependent variables that are highly correlated with each other or weakly correlated with each other?

Note that all homework assignments must be completed **through Blackboard**. Please include a link or other document including the code you used to produce your work. Remember, all work should follow the best practices for publishing computational results¹⁰ and the Google style guide for R.¹¹ *You will lose points if you do not adhere to these practices.*

Take the extra time that you have from the light homework this week to familiarize yourself with R, using the introductory resources above. Next week's lecture notes and homework are the longest in the class, so you should be ready to move quickly in R.

1.6.1 Optional

Create an account at Harvard's Dataverse: <http://thedata.org/>. Submit your code by linking to a folder in Harvard's Dataverse.

¹⁰http://wiki.stodden.net/Best_Practices_for_Researchers_Publishing_Computational_Results

¹¹<http://google-styleguide.googlecode.com/svn/trunk/Rguide.xml#todo>

1.7 Mathematical Appendix

1.7.1 Unbiased Sample Variance Estimator

This follows Wackerly et al. (2008) Chapter 8.3. Call the population variance of Y_i , $\text{Var}(Y) = \sigma^2 = \sum_{i=1}^N \frac{(y_i - \mu)^2}{N}$ and population mean of Y is $E(Y_i) = \mu$. If we randomly sample Y_i from the population, the sample variance of the Y_i is defined as:

$$\text{Var}(Y_1, Y_2 \dots Y_n) = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 \quad (1.9)$$

We want to see whether this is biased, in that $E(\text{Var}(Y_1, Y_2 \dots Y_n)) \neq \sigma^2$. Note that:

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n y_i^2 - 2 \sum_{i=1}^n y_i \bar{y} + \sum_{i=1}^n \bar{y}^2 \quad (1.10)$$

Using the fact that $\sum_{i=1}^n c = nc$ and $\sum_{i=1}^n y_i = n\bar{y}$, it follows that:

$$\sum_{i=1}^n y_i^2 - 2 \sum_{i=1}^n y_i \bar{y} + \sum_{i=1}^n \bar{y}^2 = \sum_{i=1}^n y_i^2 - 2n\bar{y}^2 + n\bar{y}^2 = \sum_{i=1}^n y_i^2 - n\bar{y}^2 \quad (1.11)$$

Taking the expected value:

$$E(\text{Var}(Y_1, Y_2 \dots Y_n)) = \frac{1}{n} E\left(\sum_{i=1}^n y_i^2 - n\bar{y}^2\right) \quad (1.12)$$

We now use the definition of variance:

$$\text{Var}(Y_i) = E(y_i^2) - E(y_i)^2 \quad (1.13)$$

Which means that:

$$E(y_i^2) = \text{Var}(Y_i) + E(y_i)^2 = \sigma^2 + \mu^2 \quad (1.14)$$

Similarly we use the fact that:

$$\text{Var}(\bar{Y}) = E(\bar{Y}^2) - E(\bar{Y})^2 \quad (1.15)$$

Which means that:

$$E(\bar{Y}^2) = \text{Var}(\bar{Y}) + E(\bar{Y})^2 = \frac{\sigma^2}{n} + \mu^2 \quad (1.16)$$

Thus,

$$E(\text{Var}(Y_1, Y_2 \dots Y_n)) = \frac{1}{n} E\left(\sum_{i=1}^n y_i^2 - n\bar{y}^2\right) = \frac{1}{n} \left[\sum_{i=1}^n E(y_i^2) - nE(\bar{y}^2) \right] \quad (1.17)$$

Substituting $E(y_i^2) = \sigma^2 + \mu^2$ and $E(\bar{Y}^2) = \frac{\sigma^2}{n} + \mu^2$ gives us:

$$\frac{1}{n} \left[\sum_{i=1}^n E(y_i^2) - nE(\bar{Y}^2) \right] = \frac{1}{n} \left[\sum_{i=1}^n (\sigma^2 + \mu^2) - n\left(\frac{\sigma^2}{n} + \mu^2\right) \right] \quad (1.18)$$

It follows that

$$\frac{1}{n} \left[\sum_{i=1}^n (\sigma^2 + \mu^2) - n\left(\frac{\sigma^2}{n} + \mu^2\right) \right] = \frac{1}{n} [n(\sigma^2 + \mu^2) - n\left(\frac{\sigma^2}{n} + \mu^2\right)] \quad (1.19)$$

After pulling out the n this leaves us with:

$$(\sigma^2 + \mu^2 - \frac{\sigma^2}{n} - \mu^2) = \sigma^2 \left(1 - \frac{1}{n}\right) = \frac{n-1}{n} \sigma^2 \quad (1.20)$$

Thus, we should divide by $n-1$ rather than n to get an unbiased estimate of the population variance.

1.7.2 Variance of the Sample Mean

Suppose Y_1, Y_2, \dots, Y_n are a random sample drawn from a population where $Y_i \sim N(\mu, \sigma^2)$. Let $\bar{Y} = \sum_{i=1}^n \frac{Y_i}{n}$ be the sample mean estimate. Then:

$$E[\bar{Y}] = E\left[\frac{1}{n} \sum_{i=1}^n Y_i\right] = \frac{1}{n} E\left[\sum_{i=1}^n Y_i\right] \quad (1.21)$$

Because $E[\sum_{i=1}^n Y_i] = \sum_{i=1}^n E[Y_i]$, and $E[Y_i] = \mu$, we have:

$$E[\bar{Y}] = \frac{n}{n} E[Y_i] = E[Y_i] = \mu \quad (1.22)$$

Next,

$$Var[\bar{Y}] = Var\left[\frac{1}{n} \sum_{i=1}^n Y_i\right] = \frac{1}{n^2} Var\left[\sum_{i=1}^n Y_i\right] \quad (1.23)$$

Because $Var[\sum_{i=1}^n Y_i] = \sum_{i=1}^n Var[Y_i]$ when Y_i are independent of each other for $i \neq j$. Thus:

$$Var[\bar{Y}] = \frac{1}{n^2} \sum_{i=1}^n Var[Y_i] = \frac{n}{n^2} \sigma^2 = \frac{\sigma^2}{n} \quad (1.24)$$

Thus, $\bar{Y} \sim N(\mu, \frac{\sigma^2}{n})$.

1.7.3 Student's Two-Sample t-test

This review follows Wackerly Ch. 7.1, 7.2, 8.8, and 10.8. We have Y_1, Y_2, \dots, Y_n as a random sample of size n from a population with a normal distribution with mean μ and variance σ^2 . It follows that:

$$\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i \quad (1.25)$$

is normally distributed with expected value μ and variance $\frac{\sigma^2}{n}$. Now suppose we have the same setup but want to test whether $\mu_1 = \mu_2$ where $\bar{Y}_1 \sim N(\mu_1, \frac{\sigma^2}{n_1})$ and $\bar{Y}_2 \sim N(\mu_2, \frac{\sigma^2}{n_2})$. We can calculate a Z-test under the null hypothesis that $\mu_1 = \mu_2$:

$$Z = \frac{(\bar{Y}_1 - \bar{Y}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma^2}{n_1} + \frac{\sigma^2}{n_2}}} \quad (1.26)$$

Here we use the fact that $\text{Var}(\bar{Y}_1 - \bar{Y}_2) = \frac{\sigma^2}{n_1} + \frac{\sigma^2}{n_2}$. Under the null hypothesis, Z has a standard normal distribution (mean zero and variance 1). Recall that our sample estimator for the population variance σ^2 is:

$$S^2 = \frac{1}{n-1} \sum_{i=1}^n (Y_i - \bar{Y})^2 \quad (1.27)$$

Then:

$$\frac{(n-1)S^2}{\sigma^2} = \frac{1}{\sigma^2} \sum_{i=1}^n (Y_i - \bar{Y})^2 \quad (1.28)$$

Has a χ^2 distribution with $n-1$ degrees of freedom, which follows from considering that the difference of Y_i 's are the difference of standard normal random variables, which are squared. Furthermore:

$$W = \frac{(n_1-1)S_1^2}{\sigma^2} + \frac{(n_2-1)S_2^2}{\sigma^2} \quad (1.29)$$

Has a χ^2 distribution with n_1+n_2-2 degrees of freedom because the sum of two independent χ^2 random variables is also χ^2 . Finally, if Z is a standard normal random variable and W is a χ^2 random variable with v degrees of freedom. Then:

$$T = \frac{Z}{\sqrt{W/v}} \quad (1.30)$$

Has a Student's t distribution with v degrees of freedom, where:

$$W = \frac{(n_1-1)S_1^2}{\sigma^2} + \frac{(n_2-1)S_2^2}{\sigma^2} \quad (1.31)$$

and $v = n_1 + n_2 - 2$. Substituting $Z = \frac{\bar{Y}_1 - \bar{Y}_2}{\sqrt{\frac{\sigma^2}{n_1} + \frac{\sigma^2}{n_2}}}$ gives us:

$$T = \frac{\frac{\bar{Y}_1 - \bar{Y}_2}{\sqrt{\frac{\sigma^2}{n_1} + \frac{\sigma^2}{n_2}}}}{\sqrt{\frac{\frac{(n_1-1)S_1^2}{\sigma^2} + \frac{(n_2-1)S_2^2}{\sigma^2}}{n_1+n_2-2}}} \quad (1.32)$$

We can factor out σ^2 from the numerator and denominator, then simplify this to:

$$T = \frac{\bar{Y}_1 - \bar{Y}_2}{\sqrt{\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2}} \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} = \frac{\bar{Y}_1 - \bar{Y}_2}{S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \quad (1.33)$$

where:

$$S_p = \sqrt{\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2}} \quad (1.34)$$

1.8 Homework 1 Solution

1.8.1 Situation B

For the first homework you were asked to replicate Situation B from Simmons *et al.* This code should do that in the simplest manner:

```
SituationB <- function(pval){  
  #Group 1: Generate all 30 observations at once  
  Liking1 <- rnorm(30, 0, 1)  
  
  # Group 2  
  Liking2 <- rnorm(30, 0, 1)  
  
  # For the first t-test, use the first 20 observations  
  t.test1 <- t.test(Liking1[1:20], Liking2[1:20], var.equal = TRUE)  
  # For the second t-test, use all 30  
  t.test2 <- t.test(Liking1[1:30], Liking2[1:30], var.equal = TRUE)  
  
  #signif = 1 if at least one test gives p < pval, 0 otherwise  
  signif <- ifelse(t.test1$p.value < pval | #Vertical bar is logical "or"  
    t.test2$p.value < pval,  
    1, 0)  
  return(signif) #value returned by the function  
}
```

We can then replicate this function 15,000 times and calculate the mean to tell us the proportion of times the null hypothesis was rejected ($p < .05$):

```
replicates <- replicate(15000, SituationB(.05))  
SitBSim <- mean(replicates)
```

I get 7.7%, same as Simmons et al.

1.8.2 Does anyone ever believe the null hypothesis?

First, in Frequentist statistics (which we are using), nobody ever *believes* anything; inferences are just the natural result of well-defined sampling processes. If one were to have an opinion about the null hypothesis, it would probably be that since it is an infinitely small set (i.e., a point in a continuous space), that we would assign zero probability to the event “null hypothesis is true.” However, this is very much a Bayesian way of thinking (i.e., in terms of probabilities of hypotheses), and much of the confusion probably centers around confusing the null hypothesis with prior beliefs.

1.8.3 When are the results of a t-test meaningless?

As discussed in class, all hypothesis tests are meaningless without information on 1) the estimated mean difference, 2) the estimated population variance, and 3) the sample size. With this information we can evaluate whether the results of a t-test are informative or trivial. We can also calculate the effect size (e.g., Cohen's d), to get a sense of whether the results are high or low impact.

1.8.4 Weakly or strongly correlated dependent variables

Consider an extreme case: Two dependent variables are *perfectly* correlated. Then a test on the first dependent variable will produce the same result as a test on the second dependent variable. Thus, with two perfectly correlated dependent variables we have only one chance to reject the null hypothesis. With two uncorrelated dependent variables we have two chances. As a result, the situation is worse when we do multiple independent tests and don't report them than when we do multiple correlated tests.

Part II

Data Analysis

Chapter 2

Getting Started and the Five Stories

2.1 Getting Started

In Part I we touched on some problems with modern data analysis, and one of the solutions: Reproducibility. Now we'll focus on how to understand and effectively communicate our data analyses to others. We will begin by obtaining data, recording what we've done, summarizing the data, making plots, and providing some descriptive statistics. Let's start by getting some actual data.

Economists are all about data sharing these days, so a good bet for a place to find some data is one of the premier economics journals, the *American Economic Review*. Poking around there for a bit leads us to a comprehensible article.¹ From a first look, it seems as though the article provides some data.

Follow the link, scroll down and you can find a link called “Download Data Set.” Click on the link and a zip file comes up.² Open the zip file and you find a .docx file, a quite unconventional format for data (usually .csv is used). Open up the .docx file and . . . no data. They won't share the long-term data due to a confidentiality issue, but they give us some aggregate data in a public use file.³ This is a common problem, where we get proprietary or confidential data with limited ability to share. To handle this, the authors make anonymized and aggregate data available, but not micro data.

Follow the link to the PUF, scroll down a bit and there seems to be another interesting study, published in *Science* (Ludwig et al., 2012). Scroll down to the *Science* PUF and download the “Cell-Level PUF” (original

¹<https://www.aeaweb.org/articles.php?doi=10.1257/aer.103.3.226>

²http://www.aeaweb.org/aer/data/may2013/P2013_4358_data.zip

³www.nber.org/mtopuf

version data 9/21/12).⁴ This should download a .dta file, Stata's format. If we want to get the full data, we have to put in a request. Let's work with what they've made publicly available.

Save the data in a folder and open R in that folder. Alternatively, you can open R and set the working directory to the folder with the .dta file:

```
setwd("working directory with .dta file")
```

As you read, type the code into a .R file. When you want to see what the code does, execute the file in R to see what happens by typing:

```
source("name of text file")
```

As you follow along, *do not cut and paste*. Type in all commands by hand. Think about what is being done at each step, and how that relates to the R language. To read Stata's .dta format we need to install and load an R package, called *foreign*:

```
install.packages("foreign", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(foreign)
```

The first line grabs the *foreign* package for us, and the second loads it for our current R session. Now we can load our data into a data frame called “mto.data”:

```
mto.data <- read.dta("mto_sci_puf_cells_20130206.dta")
```

2.1.1 First Looks at the Data

Before we continue, let's learn about the data we've obtained. The title of the article is “Neighborhood Effects on the Long-Term Well-Being of Low-Income Adults.” The study moves some low-income families to low poverty neighborhoods, then measures their subsequent physical and mental health and economic self-sufficiency.

The study randomly assigned participants to three groups. Those in the *low-poverty voucher* group had to use their voucher to move to a census tract with a poverty rate of less than 10%. There was also a *traditional voucher* group who could use their voucher to go anywhere, and a *control* group who received no voucher. Of those assigned a voucher, 48% in the low poverty group and 63% in the traditional group used it.

⁴The 100 page main appendix is available here: www.nber.org/mtopuf/mto_sci_puf_docu_memo_20131008.pdf There are 10 additional appendices. Who says papers can't be 300 pages!

The experiment enrolled 4,604 families who lived in subsidized public housing in Baltimore, Boston, Chicago, Los Angeles, and New York from 1994–1998. The data we have are from a 10–15 year follow-up of adults who participated in the study. About 90% of those from each group responded to the long-term follow-up survey.⁵ Observations from these individuals are collapsed into 158 cells, each with 14–41 people. Each cell is from a single study site and treatment group.

The study found no effect of the program on three measures of “objective well-being.”⁶ The first measure was an *economic self-sufficiency index*, which was whether they were currently employed, whether they were on Temporary Assistance to Needy Families (TANF), their total annual earned income, and their total annual income from government programs. The second was a *physical health index*, which was self-reported health (fair or poor), asthma attacks, obesity, hypertension, and trouble carrying groceries or climbing stairs. The *mental health index* was a psychological distress score in the last month, depression in the last year, generalized anxiety disorder in the last year, whether they felt calm and peaceful in the last month, and whether they slept normal last night. The study did find an effect on “subjective well-being,” which we focus on here.

Recall that the data are now stored in a data frame object called “mto.data.” We can manipulate this object to get information about the data. If you want to get a typical graphical interface with the data, use:

```
fix(mto.data)
```

A separate window should show up, allowing you to do some simple editing, although I highly recommend against doing this, as any edits you make are not reproducible because they are not coded in R.

Looking at the data we can see the first column is an identifier, and that there are a large number of columns. Quit out of the window and type:

```
names(mto.data)
```

This shows us the names of the variables (columns) of the data frame. We can see that there are 251 variables in the data. If we do:

⁵The appendix says only 3,273 were included in the analysis. Why the difference? $3,273 \neq 0.9 \times 4,604$.

⁶Note the following statements: “have long-term beneficial effects on a narrow but important set of physical health measures” (pg. 2). Which of the nine circles does this sound like? What about this one “We focus on adults in part because of our interest in well-being over the long term, which may not yet be evident for the MTO children” (pg. 4). What researcher degree of freedom does this sound like: “Aggregating outcomes improves statistical power to detect impacts and reduces risk of “false positives” from examining numerous outcomes (7).”

```
objects(mto.data)
```

It orders the variables alphabetically, rather than by their position in the spreadsheet. Alternatively, we can see the first few rows and all columns of the data with:

```
head(mto.data)
```

Let's take a look at the first variable:

```
# Subset the data frame to only the "cell_id" column  
mto.data$cell_id
```

We can see that this is the variable that indexes the “observational units” in the study, in this case, groups of participants (cells) in the randomized experiment. The cells are numbered from 1-158.

What about the second variable? We can access the variable either by its name (as we did with the cell id), or by its column number:

```
# Subset the data frame by column number  
mto.data[, 2]
```

This is the main outcome measure for the subjective well-being score, their response to the question: “Taken all together, how would you say things are these days—would you say that you are very happy, pretty happy, or not too happy?” (1 = not too happy, 2 = pretty happy, 3 = very happy).⁷ These responses are then z-scored by subtracting the control group mean and dividing by the control group standard deviation:

$$z_i = \frac{y_i - \bar{y}_c}{\sigma_c} \quad (2.1)$$

Where z_i is the z-score for cell i , y_i is the raw happiness score for cell i , \bar{y}_c is the average happiness score for the control group, and σ_c is the standard deviation of the raw happiness scores for the control group.

After looking at the main appendix, we can see that *ra_group* corresponds to the treatment group in the study:

```
mto.data$ra_group
```

From the output we can see that this is a factor with three levels: “1 = Low-Poverty Voucher/Experimental,” “2 = Traditional Voucher/Section 8,” and “3 = Control.”

⁷It’s weird to have an unbalanced scale like this. Usually you’d have something like 1 = unhappy, 2 = neither happy nor unhappy, 3 = happy, so that the scale is symmetric around the neutral point.

2.2 Five Stories of Data Analysis

Now that we have access to the data, some sense of what happened in the study, and a sense of the data structure, let's take a moment to think about the stories we can tell with the data. These stories fall roughly into five categories ([Berk, 2004, 2008](#)):

1. A *data summary* story, where we merely try to summarize the data without any particular model.
2. A *conditional distribution* story, where we identify how the distribution of a variable of interest is related to other variables, using some models of the possible distributions.
3. A *forecasting* story, where we evaluate how well our conditional distribution story is able to predict future values of the variable of interest.
4. A *statistical inference* story, where we try to generalize from a sample to its relevant population.
5. A *causal inference* story, where we guess at whether the independent variables directly cause changes in the dependent variable.

The causal and statistical inference stories are the most difficult. Their strength lies in what is often unobservable: How variables were measured, how observations were sampled, intuitive and scientific theories about what possible causes are correlated with each other. These are the most challenging because their justification requires information that is not contained in the data.

2.2.1 Model Checking

Why the heck do we care about these five stories? The basic idea is that they are complementary and constrain each other: Getting the data summary story right helps us understand what we are modeling; getting the conditional distribution story right makes sure our forecasts are based on our best models; getting the forecasting story right tests whether our conditional distribution story was too complex, etc.

Another way of thinking about the five stories is that they are ways of understanding and checking what we are doing with our data ([Gelman and Shalizi, 2013](#)). They correspond to: 1) Checking our models against the data themselves (data summary story), 2) checking our models against the assumptions we've made about the data (conditional distribution story), 3)

checking our models against data we haven't seen yet (forecasting story), 4) checking our models against possible population parameters (statistical inference story), and 5) checking our models against our causal theories (causal story).

It is usually much easier to do this type of model checking visually than through tables of coefficients or other readouts (Gelman, 2011). In fact, as we will soon see, it is almost impossible to detect some problems with a model without looking at how it fits the data graphically (Anscombe, 1973). Looking at the data, the distributions, the residuals, the forecasts, etc., help inform our intuitions about what is right and wrong about our models. Very few, if any, discoveries have been made by looking at a column of p-values.

2.3 The Data Summary Story

According to Berk (2004), accurate data descriptions are often enough to settle policy matters of any importance. We can see data directly and examine their properties and distribution, without assuming anything about them. Doing data summary well before doing any modeling allows us to ground our conditional distribution story. Failing to do this permits serious errors, for example drawing conclusions based on an outlier or measurement error.

Let's return to our example dataset and try to construct a data summary story. Let's see how the primary outcome measure, happiness, is related to the treatment, the person's housing voucher. In this approach we aren't using any particular model to talk about the data. Instead, we are looking at the data as directly as possible, using plots and simple calculations.

First, let's check the minimum and maximum mean happiness z-score:

```
min(mto.data$mn_happy_scale123_z_ad) # minimum z-score  
max(mto.data$mn_happy_scale123_z_ad) # maximum z-score
```

This shows us that the z-scores fall nicely between -1 and 1. Keep in mind that if you want more information about any R function, just type:

```
?name.of.function
```

For example, to find out more about the *min* function, type:

```
?min
```

Next, we'll use a histogram to show the data, so let's set the x-axis of the histogram:

```
# Assign a sequence of numbers from -1 to 1 by increments of 0.1 to "bin"  
bins <- seq(-1, 1, by = 0.1)
```

This should generate a sequence of numbers from -1 to 1 in increments of 0.1. Then, let's assign a variable name to each group of data:

```
# Subset the data to the experimental group
experimental <- mto.data$mn_happy_scale123_z_ad[mto.data$ra_group
    == "1=Low-Poverty Voucher/Experimental"]

# Subset the data to the section 8 group
section8 <- mto.data$mn_happy_scale123_z_ad[mto.data$ra_group
    == "2=Traditional Voucher/Section 8"]

# Subset the data to the control roup
control <- mto.data$mn_happy_scale123_z_ad[mto.data$ra_group
    == "3=Control"]
```

Notice that

```
mto.data$mn_happy_scale123_z_ad
```

gives us the correct column, and the subsetting

```
[mto.data$ra_group == "1=Low-Poverty Voucher/Experimental"]
```

says, within that column, find the rows such that *ra_group* is equal to the string “1=Low-Poverty Voucher/Experimental.”

2.3.1 Histograms

We now have three variables, *experimental*, *section8*, and *control*, that contain the data for the averaged z-scored happiness ratings for each cell in the study. Let's plot the z-scores to see what they look like:⁸

```
# Create a .png file that will contain everything between png() and dev.off()
png(filename = "happyhist.png",
    width = 3000, # adjust width of .png file, in pixels
    height = 3000, # adjust height of .png file, in pixels
    res     = 300) # set resolution in pixels per inch

# Parameters for the plots
par(mfrow = c(3, 1), # mfrow sets the number of rows and columns
    cex = 1.3, # cex scales text and symbols
    mar = c(4, 3, 1, 0), # mar and oma set the inner and outer margins
```

⁸Some useful links on histograms in R: <http://www.r-bloggers.com/basics-of-histograms/>

```

oma = c(0, 0, 0, 0)

# Plot a histogram
hist(experimental,
      xlim = c(-1, 1), # the x axis ranges from -1 to 1
      ylim = c(0, 15), # the y axis ranges from 0 to 15
      main = "Experimental Group", # The main title is "Experimental Group"
      breaks = bins, # Use the "bins" vector to determin the x axis bins
      xlab = "")

hist(section8, xlim = c(-1, 1), ylim = c(0, 15),
      main = "Section 8 Group", breaks = bins, xlab = "")

hist(control, xlim = c(-1, 1), ylim = c(0, 15),
      main = "Control Group", breaks = bins, xlab = "Mean Happiness Z-Score")

dev.off()

```

This should output a .png file to your current working directory with histograms of mean z-scores for each of the three groups.

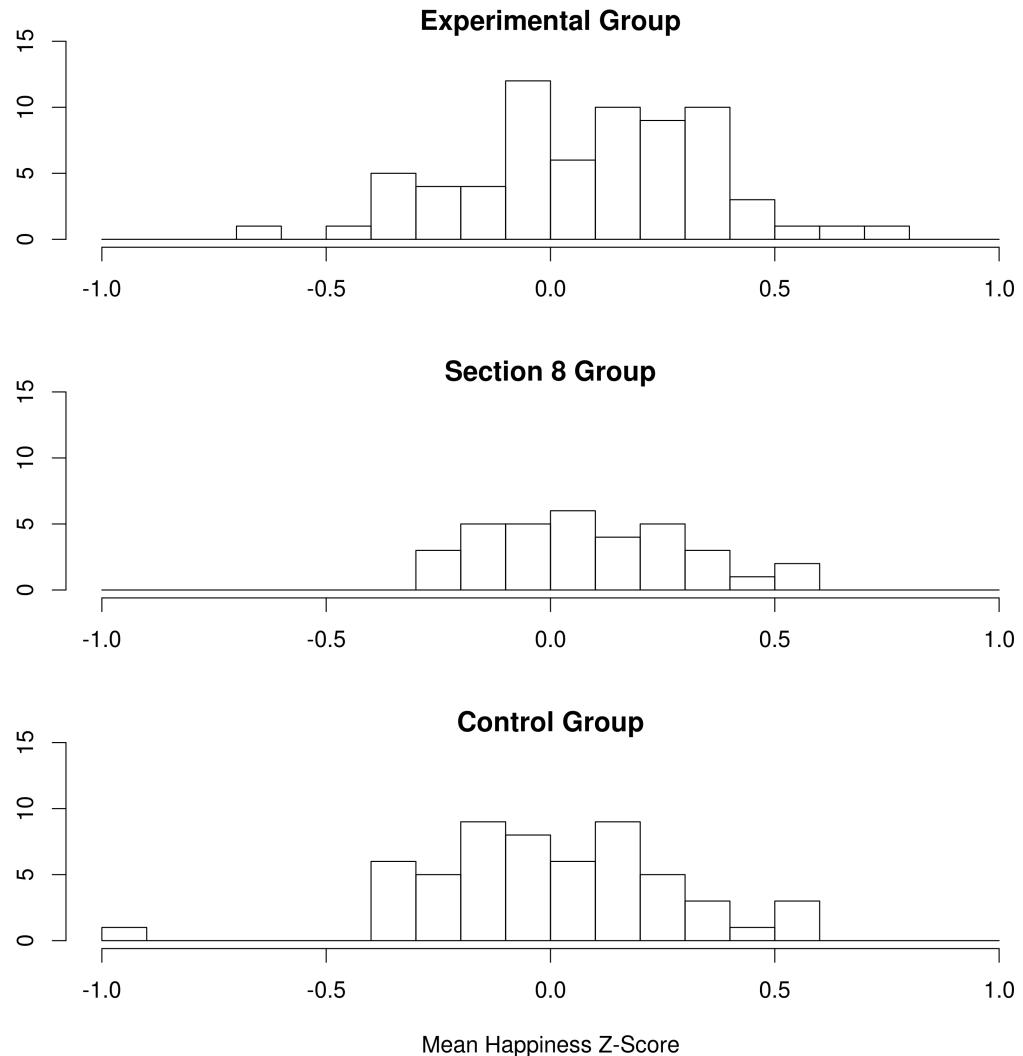


Figure 2.1: Histogram of average happiness z-scores for each treatment group.

What's important about the histogram is that very little information about the data is lost or interpolated. This is about as direct a view of the data that we can have. We can visually inspect the minimum value, maximum value, mean, standard deviation, and whether the distribution is skewed.

Histograms do suffer some data loss, however. The problem is that we have to arbitrarily choose bin widths (width of bars on the x axis). The default rule for choosing histogram bin width (h) in R is Sturges' Rule:

$$\text{Sturges' Rule, } h = \frac{R}{1 + 3.322 \log_{10}(n)} \quad (2.2)$$

where R is the range of the data. There are some other options such as Scott's rule:

$$\text{Scott's Rule, } h = \frac{3.5\hat{\sigma}}{n^{1/3}} \quad (2.3)$$

where $\hat{\sigma}$ is the sample standard deviation, and Freedman-Diaconis:

$$\text{Freedman-Diaconis, } h = 2 \times \frac{IQR(x)}{n^{1/3}} \quad (2.4)$$

where $IQR(x)$ is the interquartile range of x . The basic idea behind these rules is that as we get more data (n increases) our bin width h decreases, giving us more bins. This allows the histogram to approximate a continuous distribution as the sample size increases. In general, Sturges's rule creates too many bins (oversmoothing) (Wand, 1997), but is the default rule in R. The optimal rate of decay of the bin width is proportional to $n^{-1/3}$, so the essential difference between Scott's rule and Freedman-Diaconis is sensitivity to outliers, where Scott's rule uses a sample estimate of the standard deviation $\hat{\sigma}$ (making it sensitive to outliers), while Freedman-Diaconis is less sensitive to outliers as it uses the interquartile range. Scott's rule is better for normal data, Freedman-Diaconis is better for data with potential outliers.

If we have too many bins (h too small), then the histogram will look like a rug plot, with one observation per bin. If we have too few bins (h too large), then there will only be one box:

```

png(filename = "happyhistbins.png",
    width = 3000,
    height = 3000,
    res     = 300)

par(mfrow = c(3, 1),
    cex = 1.3, # cex scales text and symbols
    mar = c(4, 3, 1, 0), # mar and oma set the inner and outer margins
    oma = c(0, 0, 0, 0))

hist(experimental,
      xlim = c(-1, 1),
      ylim = c(0, 40),
      breaks = "FD",
      xlab = "",
      main = "Freedman-Diaconis")
rug(experimental, col = rgb(1, 0, 0, .9), ticksize = -0.1)

hist(experimental, xlim = c(-1, 1), ylim = c(0, 40),
      breaks = seq(-1, 1, by = 0.001), xlab = "", main = "h = 0.001")
rug(experimental, col = rgb(1, 0, 0, .9), ticksize = -0.1)

hist(experimental, xlim = c(-1, 1), ylim = c(0, 40),
      breaks = seq(-1, 1, by = 0.5),

```

```
xlab = "Mean Happiness Z-Score", main = "h = 0.5")
rug(experimental, col = rgb(1, 0, 0, .9), ticksize = -.1)

dev.off()
```

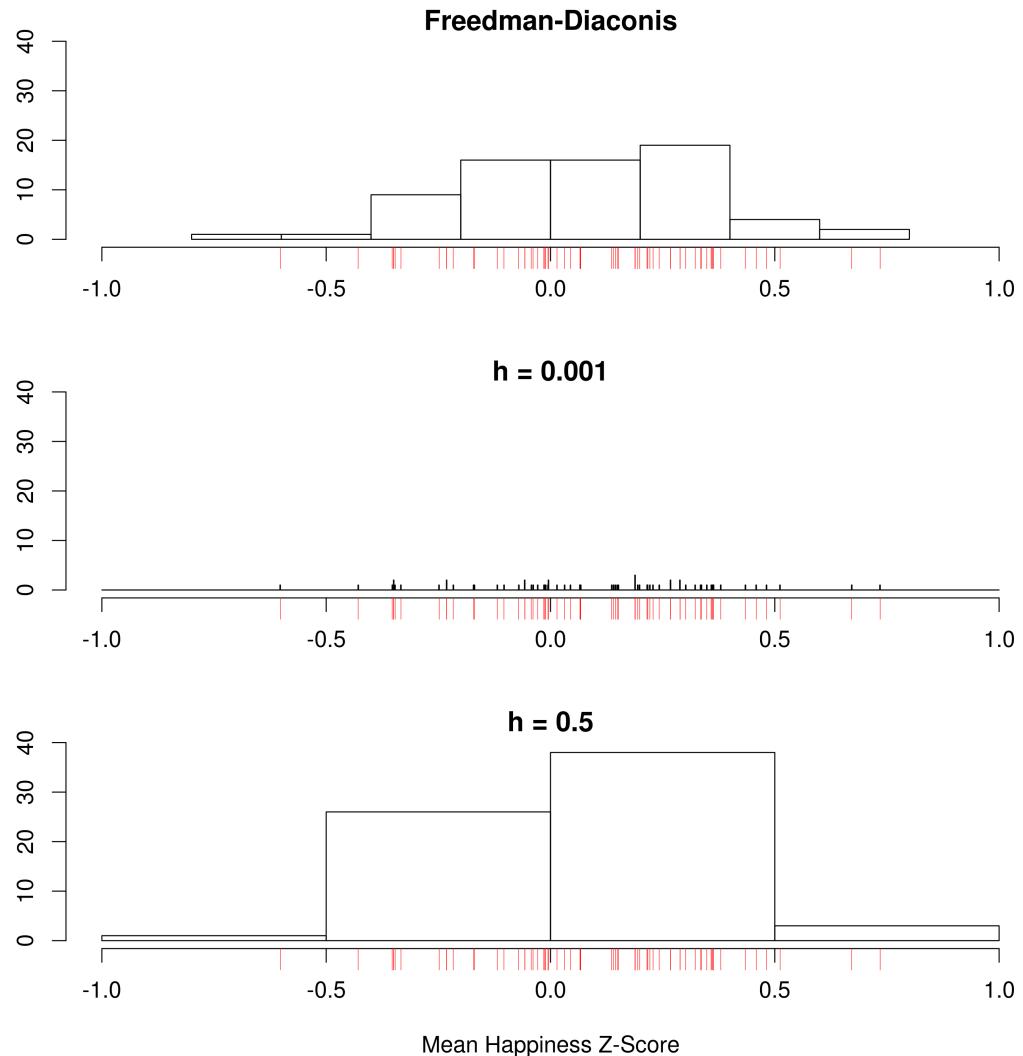


Figure 2.2: Histograms with bin width chosen using the Freedman-Diaconis rule (top), a tiny bin width ($h = 0.001$), or a large bin with ($h = 0.5$)

We can also use a scatterplot of the data to get another perspective of the three distributions, side-by-side. The rug plot on the right side of the plot shows the marginal distribution of the dependent variable, whereas the interior points show the three conditional distributions:

```
png(filename = "happyscatter.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(-100, -100, xlim = c(1, 3),
      ylim = c(-1, 1), # set y axis limits
      ylab = "Mean Happiness Z-score",
      xlab = "Treatment Group",
      xaxt = "n") # suppress default x axis
```

```

# Create x axis for plot
axis(side = 1, # side = 1 corresponds to the bottom of the plot
     at    = c(1, 2, 3), # at determines where to put the tick marks
     labels = c("Experimental", "Section 8", "Control")) # the labels

# Plot data with jitter
# jitter adds some noise into the data to avoid points plotted over each other
points(jitter(mn_happy_scale123_z_ad)~factor(ra_group),
       data = mto.data,
       pch = 19, # pch chooses the point character, here a solid circle
       col = rgb(0, 0, 0, .2))

rug(side = 4, mto.data$mn_happy_scale123_z_ad, # side = 4 is the right
     ticksize = -.025, # negative ticksize flips ticks around the axis
     col      = "red")

dev.off()

```

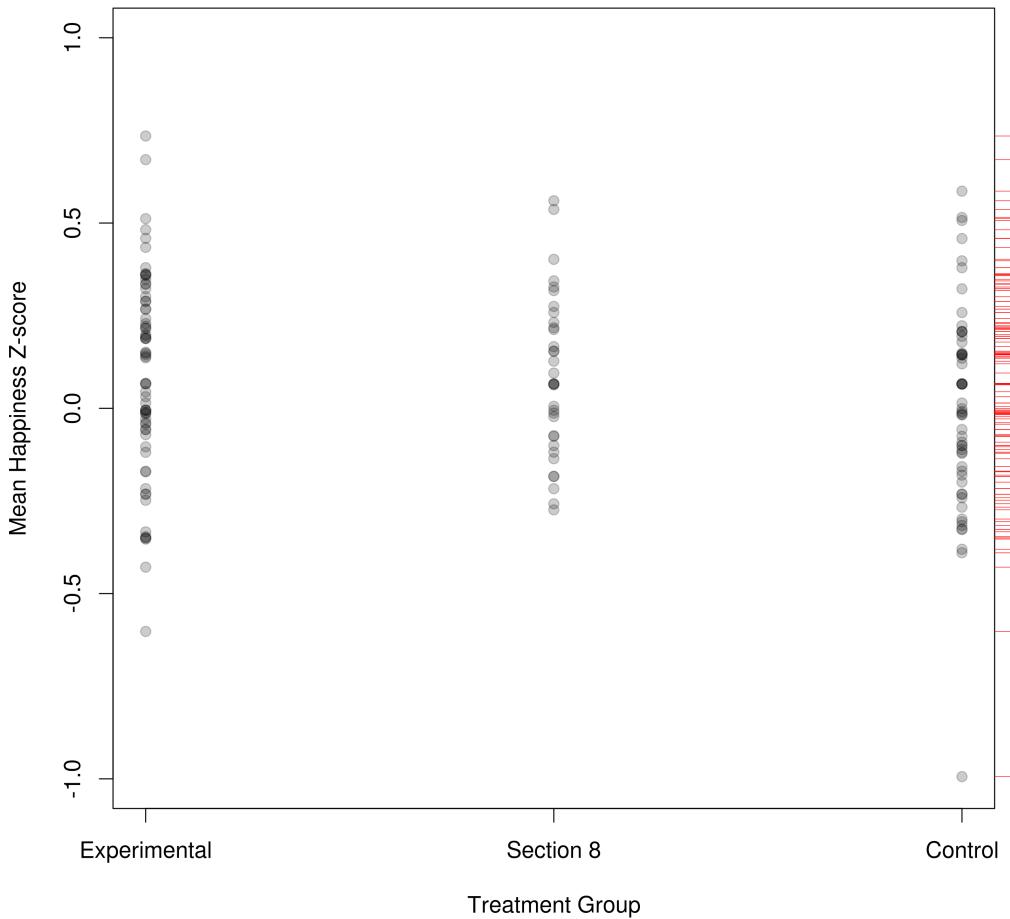


Figure 2.3: Scatterplots of mean happiness z-scores by treatment group.

It looks like the spread on the experimental group is larger than the other two, with what look like two outliers at the top. The standard deviation is largest in the control group (0.28), followed by experimental (0.27) followed by the section 8 group (0.22):

```
sd(experimental) # sd() gives us the standard deviation
sd(section8)
sd(control)
```

Part of this could be the difference in sample sizes: The spread of the data in small samples tends to underestimate the population spread. Looking at the sample sizes:

```

length(experimental) # length gives us the length of an object
length(section8)
length(control)

```

The experimental group has 68 cells, the section 8 group has 34 cells, and the control group 56 cells. So it could be that the data actually come from distributions with similar spreads, but the experimental group, with larger sample size, is better able to estimate a distribution with greater spread, or heavy tails.

Also notice that the control group might have an outlier, with mean happiness z-score well below that of any other observation for any group.⁹ If we drop the outlier:

```

# subset the control data such that values are greater than - 0.8
sd(control[control > -.8])

```

The control standard deviation drops down to 0.24, larger than the section 8 group, but smaller than the experimental group.

From the plots it is hard to tell, but the means decrease as we go from left to right: 0.094 (experimental), 0.088 (section8), and 0.002 (control), as do the medians (0.138, 0.066, and -0.005, respectively). The median of the experimental group is higher than the mean, indicating some skew toward lower values. The other two groups, on the other hand, have medians that are lower than the means. We can add means and medians to the scatterplot:

```

png(filename = "happyscatter2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(-100, -100, xlim = c(1, 3),
      ylim = c(-1, 1),
      ylab = "Mean Happiness Z-score",
      xlab = "Treatment Group",
      xaxt = "n")
axis(side = 1, at = c(1, 2, 3),
      labels = c("Experimental", "Section 8", "Control"))
points(jitter(mn_happy_scale123_z_ad)~factor(ra_group),
       data = mto.data, pch = 19, col = rgb(0, 0, 0, .2))
rug(side = 4, mto.data$mn_happy_scale123_z_ad, ticksize = -.025)

# Add line that connects points (1, .094), (2, .088) and (3, .002)
lines(c(1, 2, 3), c(.094, .088, .002), col = "red")

```

⁹It worries me that their results may have been driven by this outlier. They don't mention anything about outliers in the paper. I picked this paper without having any suspicion of outliers in the control group, which might give you an idea of the prevalence of data analysts failing to look at their data.

```

lines(c(1, 2, 3), c(.138, .066, -.005), col = "blue")
dev.off()

```

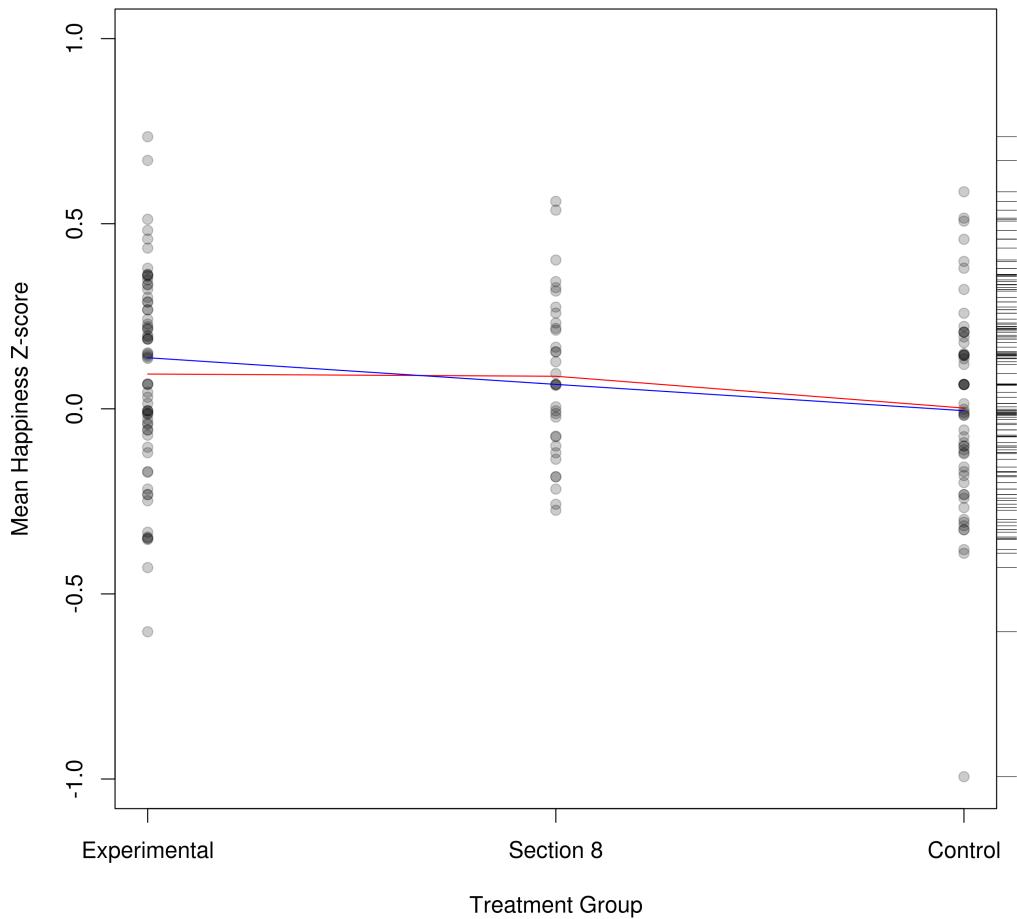


Figure 2.4: Scatterplots of mean happiness z-scores by treatment group, with mean line (red) and median line (blue).

Another alternative is to use box plots, which are the R default plot for grouped (factor) data:

```

png(filename = "happyboxplot.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(factor(mto.data$ra_group), jitter(mto.data$mn_happy_scale123_z_ad),
     xlim = c(0, 4),
     ylim = c(-1, 1),

```

```

ylab = "Mean Happiness Z-score",
xlab = "Treatment Group",
xaxt = "n",
pch = 19)
axis(side = 1, at = c(1, 2, 3),
     labels = c("Experimental", "Section 8", "Control"))
dev.off()

```

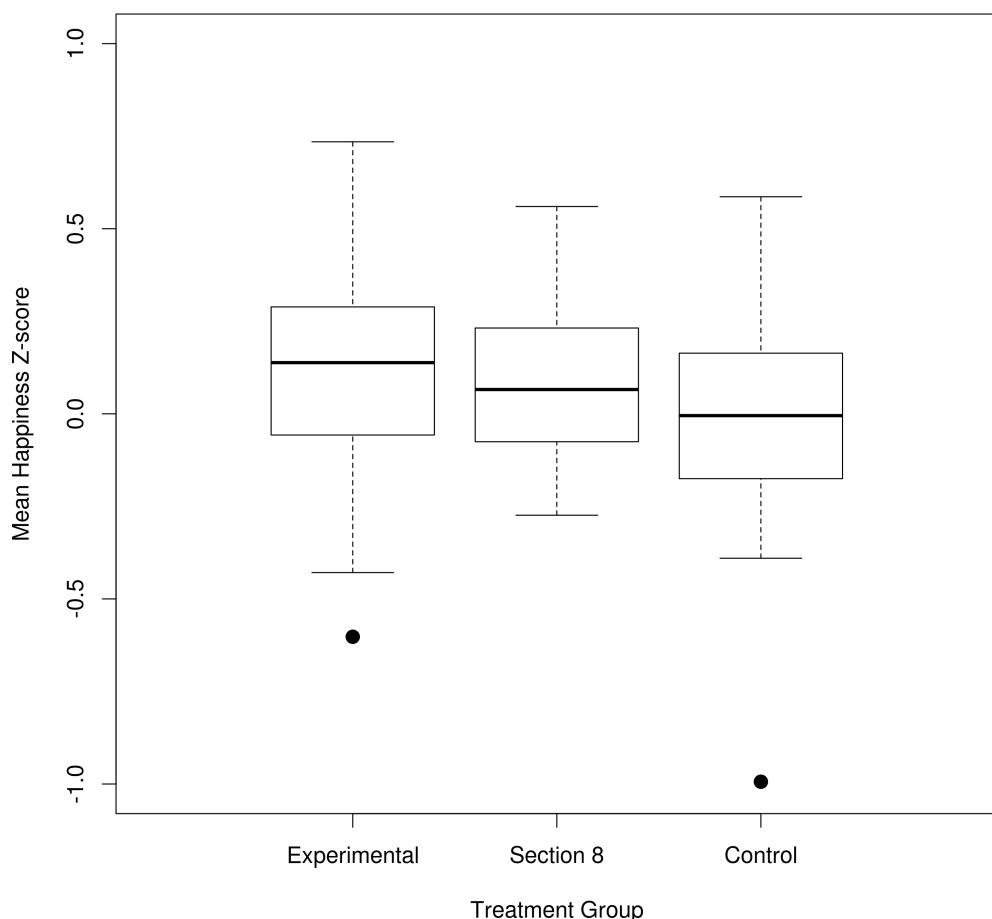


Figure 2.5: Box plot of mean happiness z-scores by treatment group.

The horizontal bar in the middle of each box shows the median for each group. The lower hinge shows the location of the 25th percentile or first quartile. The upper hinge shows the location of the 75th percentile or third quartile. The whiskers extend to the most extreme observation that is at most 1.5 times the interquartile range, which is the length of the box.

One reason I don't like box plots is because you can't see the sample size. You don't know if there are 100 or 1000 observations in each group. To fix this we can plot the data over the boxplots:

```
png(filename = "happyboxplot2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(factor(mto.data$ra_group), jitter(mto.data$mn_happy_scale123_z_ad),
      xlim = c(0, 4),
      ylim = c(-1, 1),
      ylab = "Mean Happiness Z-score",
      xlab = "Treatment Group",
      xaxt = "n",
      pch = 19)
axis(side = 1, at = c(1, 2, 3),
      labels = c("Experimental", "Section 8", "Control"))
points(jitter(mn_happy_scale123_z_ad)~factor(ra_group),
       data = mto.data, pch = 19, col = rgb(0, 0, 0, .15))
dev.off()
```

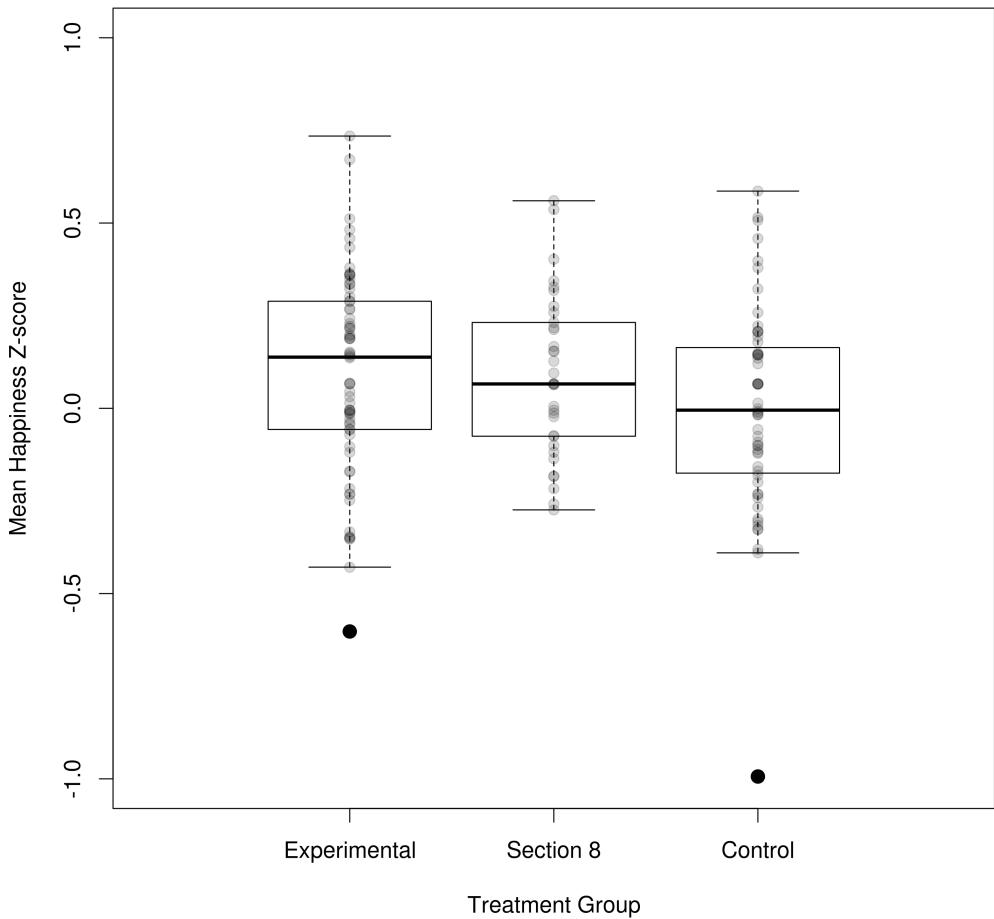


Figure 2.6: Box plot of mean happiness z-scores by treatment group.

This tells us what we want, conveying information about the distribution of the data and the sample size.

2.3.2 Cumulative Distributions

Alternatively, we can plot the empirical cumulative distributions (eCDFs) of the data:

```
png(filename = "happyecdf.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(ecdf(experimental),
      xlab = "Mean Happiness Z-Score",
      ylab = "Cumulative Probability",
```

```

xlim = c(-1, 1),
col = "red",
pch = 21,
main = "Experimental, Section 8, and Control Group eCDFs")
plot(ecdf(section8),
      main = "Section 8",
      ylab = "",
      xlab = "Mean Happiness Z-Score",
      add = TRUE, # add to same plot as previous data
      col = "blue",
      pch = 22)
plot(ecdf(control),
      main = "Control Group",
      ylab = "",
      xlab = "Mean Happiness Z-Score",
      add = TRUE,
      col = "green",
      pch = 23)

# Add legend to plot
legend(-.75, 1, # legend location in x-y coordinates
       c("Experimental", "Section 8", "Control"), # legend labels in order
       col = c("red", "blue", "green"), # colors for legend labels in order
       pch = c(21, 22, 23)) # symbols for legend labels in order
dev.off()

```

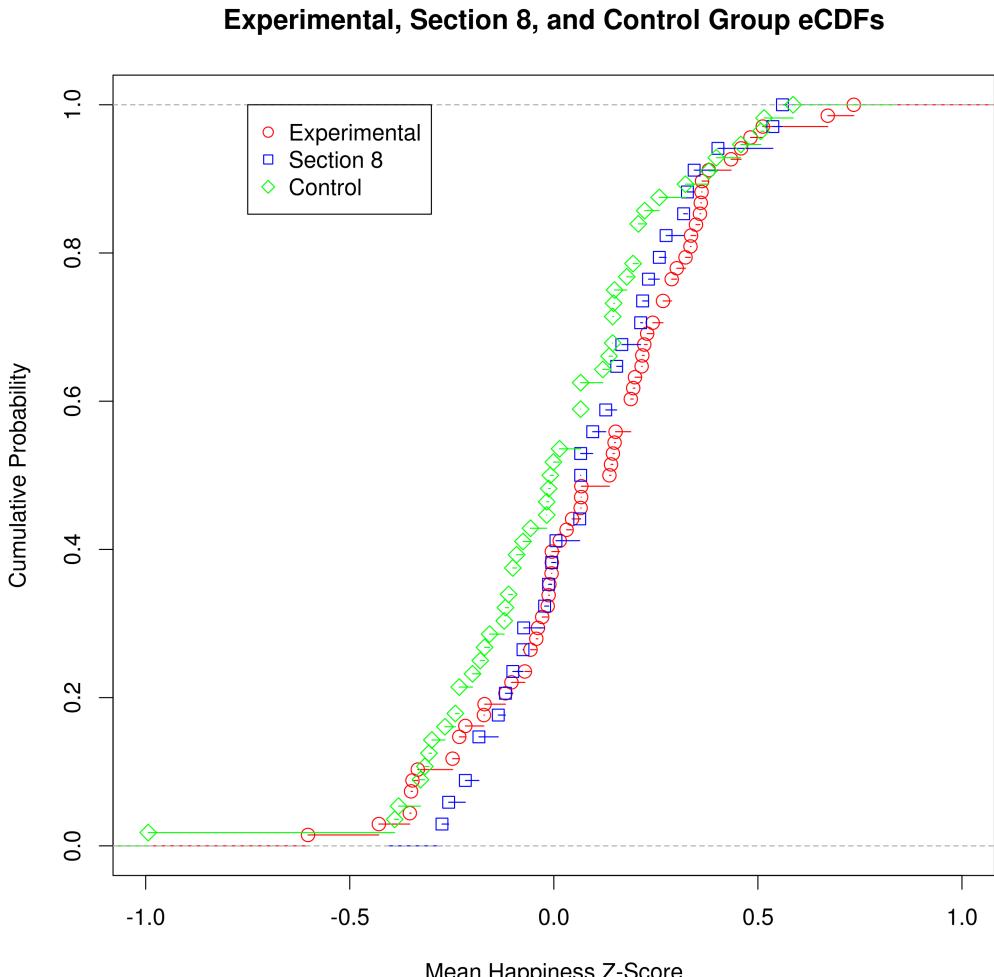


Figure 2.7: Plots of empirical cumulative distribution functions of mean happiness z-scores for each treatment group.

Looking at the histograms and CDFs, it seems like the experimental and Section 8 groups have nearly identical distributions of mean happiness z-scores, while the control group fares worse, but not by a lot. Again, we see the potential outlier in the control group as the long left tail.

Our first pass at the data summary story looked at the distribution of the dependent variable across three discrete experimental groups. We've looked at the data as closely as we can without imposing data loss. Here's what we've learned:

- All the data fall between -1 and $+1$ on the mean happiness z-score scale.

- The experimental and control groups seem to be more spread out than the section 8 group.
- Both the control and experimental groups have observations that don't fit a simple distribution well (potential outliers).

We could ask for a larger sample to more carefully understand how often these “rare” events occur. We could also think about how to capture these rare events with distributions that have heavy-tails. We'll think about this more in the conditional distribution story. However, we should consider that the data from the different treatment groups might not come from independent normal distributions.

2.3.3 Scatterplots

Let's continue our data summary story for two continuous variables, happiness and the economic self sufficiency index. First, to make things more manageable, let's create variables in our dataset with slightly shorter names:

```
# Create a new column in data frame called "happy"
# Assign the mn_happy_scale_123_z_ad mean happiness z-scores to this column
mto.data$happy <- mto.data$mn_happy_scale123_z_ad
mto.data$selfsuff <- mto.data$mn_f_ec_idx_z_ad
```

Now, let's make a scatterplot. Scatterplots are a very powerful tool that rarely involve any loss of data when creating our summary, and are essential for examining the relationship between two variables:

```
png(filename = "scatter1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, oma = c(0, 0, 0, 0))
plot(mto.data$selfsuff,mto.data$happy,
      ylab = "Mean Happiness Z-Score",
      xlab = "Mean Economic Self-Sufficiency Z-Score",
      xlim = c(-1, 1),
      ylim = c(-1, 1),
      pch = 19,
      cex = 1, # set size for points in plot
      col = rgb(0, 0, 0, 0.7))
dev.off()
```

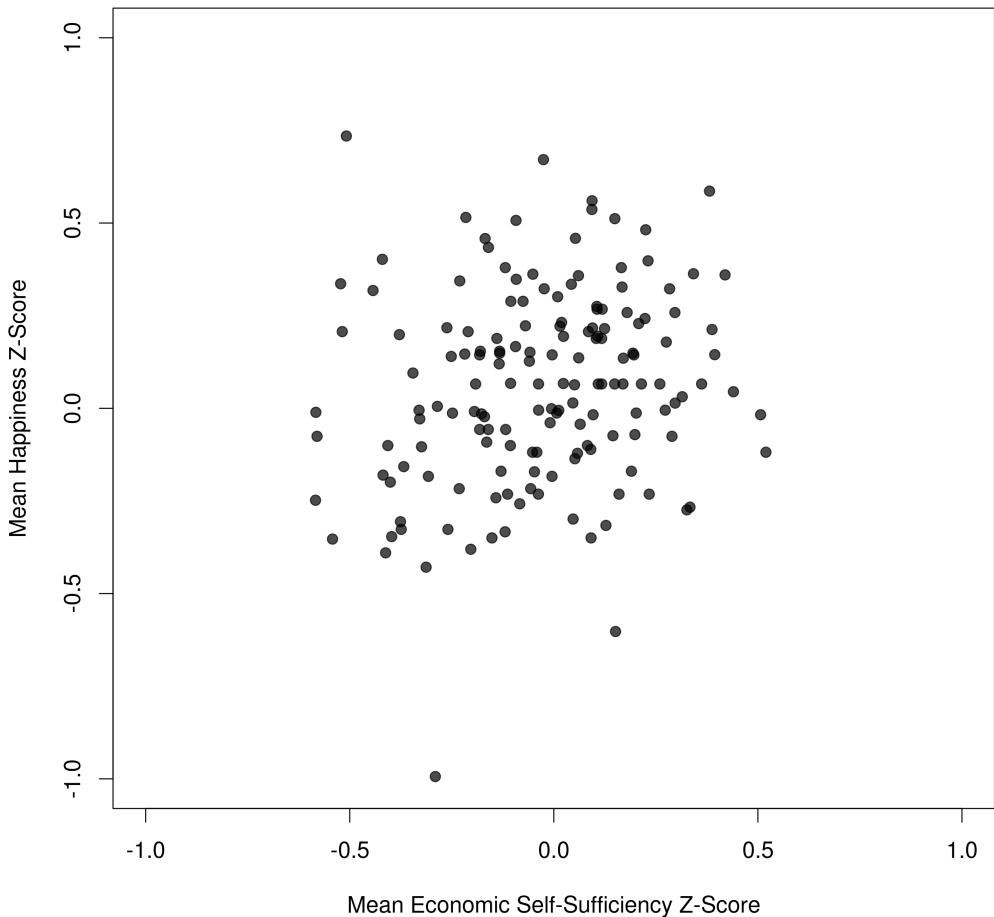


Figure 2.8: Scatterplot with mean happiness z-scores (y-axis) and mean economic self-sufficiency z-scores (x-axis) for each cell in the study.

It looks like a giant blob. We can see a few points that are outside the blob. For example, there is one observation with a very low mean happiness z-score (-1), which would be outside the blob no matter what the self-sufficiency score.

The dispersion (spread) of the two variables looks similar, although there is one very low mean happiness z-score. Looking at the standard deviations:

```
sd(mto.data$happy)
sd(mto.data$selfsuff)
```

The standard deviations of the two variables are similar (0.26 for happiness and 0.24 for self sufficiency).

There doesn't seem to be much of a discernable pattern visually. The relationship between mean economic self-sufficiency z-scores and mean happiness z-scores in this data is not obvious. To the unaided eye, the relationship could be positive or negative. However, the range of mean economic self-sufficiency z-scores (-0.5 to 0.5) is about half the size of the range of the mean happiness z-scores (-1, 0.6), partly due to a really low mean happiness z-score for one cell.

2.4 The Conditional Distribution Story

So far we've mostly avoided saying anything about a model of the underlying process that generated the data. The conditional distribution story goes further than the data summary story by talking about how the data may be realizations of a probabilistic process with a well-specified distribution, such as the normal distribution. Describing the data in terms of models of conditional distributions loses some fidelity to the data, as we cease to talk about the data as directly as we were with histograms and scatterplots. Instead, we are abstracting away a bit and trying to say something more general about the data generating process.

One might think that this process necessarily involves hypothesis tests, inferences, or causal statements. While hypothesis tests and statistical inferences often rely on distributional assumptions, the converse is not true, and one can build a model of the conditional distribution of the data without making any such inferences.

2.4.1 Discrete Independent Variables

Our dependent variable, mean happiness z-score, was generated by participants responding to a survey, with three possible responses. These responses were z-scored, then averaged for each cell in the study. We previously plotted the CDFs of the three groups.

Let's plot the data from the experimental group against simulations of the normal distribution directly, to see how they compare. First, we generate random draws from the normal distribution with sample size, mean, and standard deviation equal to the data from the experimental group:

```
# Simulate a normal random variable (RV) using rnorm(x, y, z)
# The number of observations to draw is x
# The mean of the normal random variable is y
# The standard deviation of the normal random variable is z
x <- length(experimental)
y <- mean(experimental)
```

```

z <- sd(experimental)
set.seed(1) # Set seed for the random number generator, to reproduce results
norm <- rnorm(x, y, z) # rnorm is a function that generates the normal RV

```

Sorted Data Plot

Let's plot the normal data against the experimental group data, sorting the data from least to greatest value:

```

png(filename = "sortplot.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
# The floor and ceiling functions find the integer that is not greater
# (or less than) any element of a vector
min.d <- floor(min(experimental, norm)) # Find the smallest value
max.d <- ceiling(max(experimental, norm)) # Find the largest value
plot(sort(norm),
      sort(experimental),
      xlim = c(min.d, max.d),
      ylim = c(min.d, max.d),
      ylab = "Sorted Experimental Group Average Happiness Z-Scores",
      xlab = "Sorted Draws from a Normal Distribution",
      pch = 19,
      col = rgb(0, 0, 0, .7))
abline(0, 1) # Draw a line on the plot with intercept 0 and slope 1
dev.off()

```

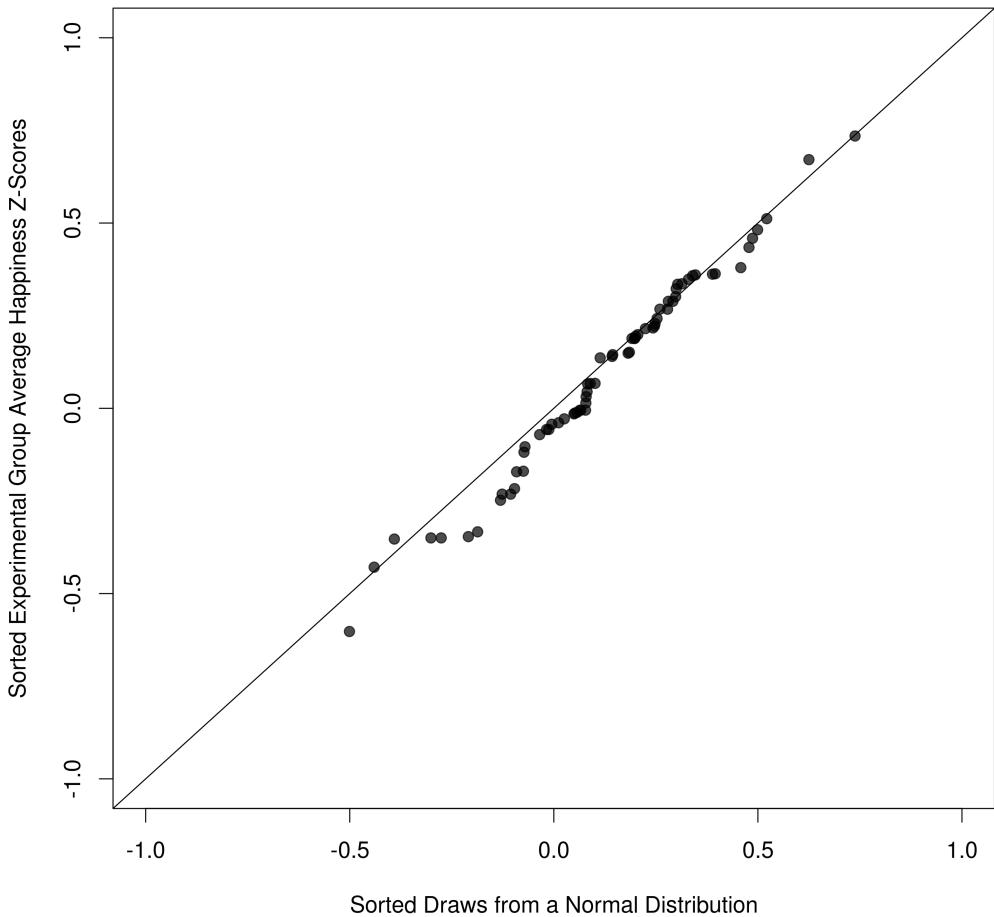


Figure 2.9: Plot of sorted data from the normal distribution and the experimental group. The normal distribution has the same number of observations (68), mean, and standard deviation as the experimental group sample.

Let's interpret what the plot means. The lowest value from each data set is at the bottom left. If all the values are equal from the two distributions, all points will fall along the diagonal line. We can see that this is not the case. At the lower end, there is a curve where observations fall below the diagonal line, indicating that the experimental group data are more extreme (lower) than the data drawn from a normal distribution.

So far we've learned a bit about how our experimental group data compare to a normal distribution with the same mean and standard deviation. The problem with the sort plot is that the two samples being compared must have the same sample size. If one of the samples has a small sample size, then

so must the other. However, when sample sizes are small, the variation from sample to sample will be large. If this is the case, then large deviations will occur in the sort plot fairly often, even when the samples are drawn from the same distribution.

Q-Q Plot

Next, let's plot the quantiles. The quantiles of a distribution are the set of values such that $x\%$ of the observations fall below that value. The plot of the quantiles of two distributions against each other is called a Q-Q plot (for quantile-quantile plot). We will call it qqplot for short. The advantage of the qqplot over the sorted data plot is that we do not need equal sample sizes. In fact we can simulate as many draws from the normal distribution as we want to make sure we accurately represent its distribution:

```
# 10,000 draws from the normal distribution
set.seed(2)
norm <- rnorm(10000, mean(experimental), sd(experimental))
```

To find the first decile for the smaller experimental dataset, we need to sort the data, then find the number such that 10% of the observations fall below that number. Because there are 68 observations in the experimental group, 10% of the data would be 6.8 observations, so we need to round up to 7:

```
sort(experimental)[7]
```

The value for the 0.1 quantile (the seventh observation) is -0.33 . But if we look at the quantile, it is not the same:

```
# Find the quantile of the experimental data
# probs is a vector that specifies the quantiles to be returned
quantile(experimental, probs = 0.1)
```

This is because we are using the 7th observation, rather than 6.8. R's quantile function does some interpolation, trying to figure out what the 0.1 quantile should be when the 0.1 quantile (the 6.8th observation) does not actually exist. To get R to use a simple linear interpolation between the 6th and 7th observations, we can set the quantile "type":

```
# Use linear interpolation of the empirical CDF
quantile(experimental, probs = 0.1, type = 4)
```

Which is closer to observation 7 than 6. We can do the same for the simulated data. The sample size was 10,000, so the 0.1 quantile is at the 1,000th observation:

```
sort(norm)[1000] # Select the 1000th ordered observation
```

The value for the 0.1 quantile in the simulated data is -0.25. We can now plot these two sets of quantiles against each other. We have to be careful in the number of quantiles we choose. If we choose too many quantiles, then the real data will look artificially compressed, because the 0.01 quantile will likely be the same as the 0.001 quantile, which is the same as the 0.0001 quantile. For example, the 0.01 quantile for the experimental data is the 0.68th observation, if we round up to 1, that is -0.60. The 0.001 quantile is still -0.60. On the other hand, the 0.01 quantile for the simulated data is observation 100, which is -0.52, but the 0.001 quantile is -0.76, and the 0.0001 quantile is -1. So, we need to make sure that the quantiles are not truncated for the smaller sample. To do this, we can have no more quantiles than the number of observations of the *smaller* sample.

```
png(filename = "qqplot1a.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
experimental.q <- quantile(experimental,
  probs = seq(0, 1, 0.01)) # create a sequence of quantiles
normal.q <- quantile(norm,
  probs = seq(0, 1, 0.01))
min.q <- floor(min(experimental.q, normal.q))
max.q <- ceiling(max(experimental.q, normal.q))
plot(normal.q, experimental.q,
  xlim = c(min.q, max.q),
  ylim = c(min.q, max.q),
  ylab = "Quantiles of Experimental Group Average Happiness Z-Scores",
  xlab = "Quantiles of Draws from a Normal Distribution",
  pch = 19,
  col = rgb(0, 0, 0, 0.4))
abline(0, 1)
dev.off()
```

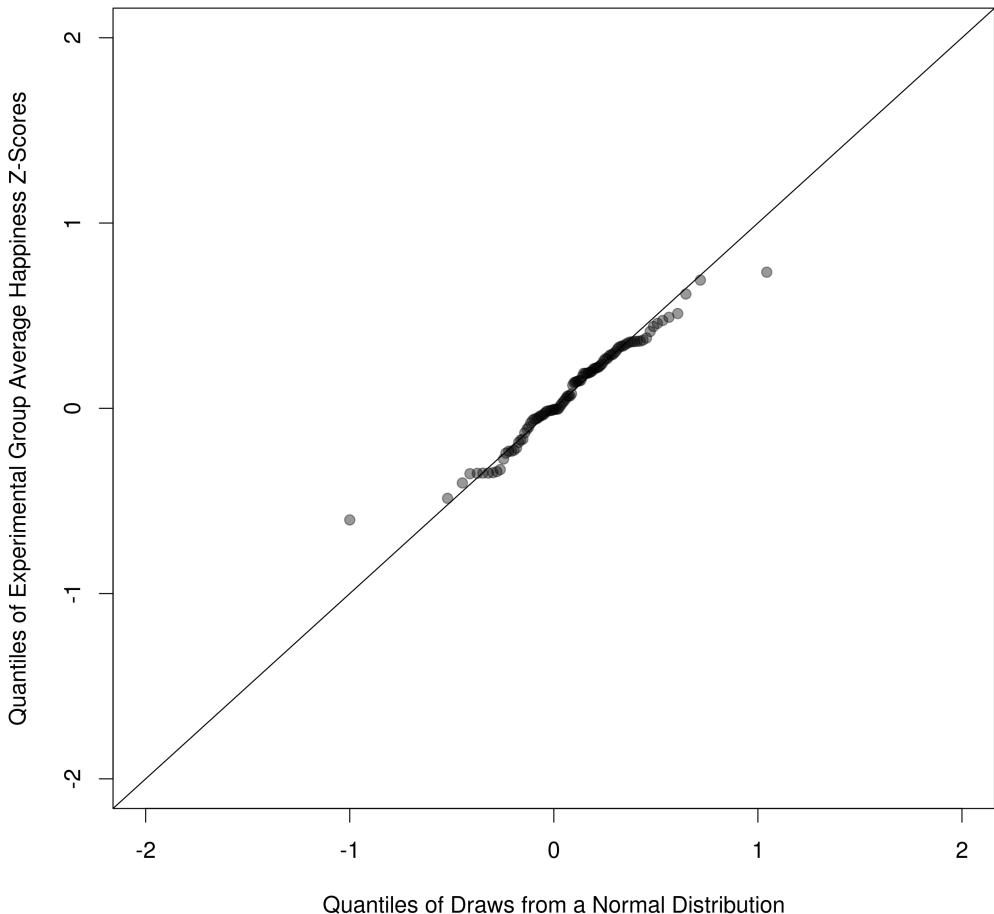


Figure 2.10: Plot of quantiles from a simulated normal distribution and the experimental group. There are 100 quantiles, but only 68 observations in the experimental group. This causes a problem which makes it look like the experimental group data is more compressed than the normal distribution, when this is an artifact of the truncation.

There's a lot of debate about the correct rule for positioning quantiles. The two most common rules for selecting the probabilities of the i th quantile are:

$$P(X < q_i) = \frac{k_i - 0.5}{n} \quad (2.5)$$

and

$$P(X < q_i) = \frac{k_i}{n + 1} \quad (2.6)$$

where k_i ranges from 1 to n , and n is the sample size of the variable with *fewer* observations. For our experimental group, there are 68 observations, so using the first rule the quantiles should be associated with probabilities $.5/68$, $1.5/68, \dots, 67.5/68$:

```
png(filename = "qqplot1b.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
experimental.q <- quantile(experimental,
  probs = (seq(1, 68, 1) - .5)/68, type = 4) # create a sequence of quantiles
normal.q <- quantile(norm,
  probs = (seq(1, 68, 1) - .5)/68, type = 4)
min.q <- floor(min(experimental.q, normal.q))
max.q <- ceiling(max(experimental.q, normal.q))
plot(normal.q, experimental.q,
  xlim = c(min.q, max.q),
  ylim = c(min.q, max.q),
  ylab = "Quantiles of Experimental Group Average Happiness Z-Scores",
  xlab = "Quantiles of Draws from a Normal Distribution",
  pch = 19,
  col = rgb(0, 0, 0, 0.4))
abline(0, 1)
dev.off()
```

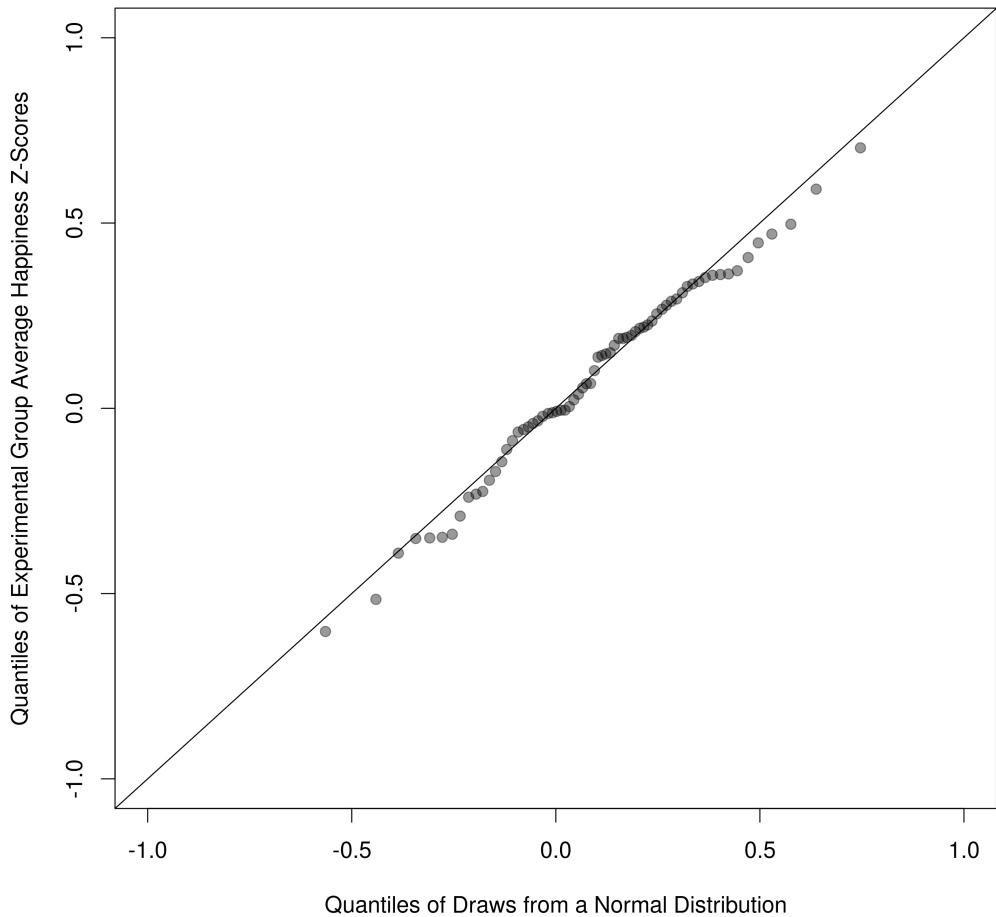


Figure 2.11: Plot of quantiles from a simulated normal distribution and the experimental group using the rule $(k - .5)/n$ to select probabilities for the quantiles.

Looking at the qqplot, we see that most of the quantiles of the experimental and normal distribution are very similar. Now, what if the data had a uniformly larger variance than its equivalent normal distribution? Let's simulate this:

```
set.seed(3)
norm1 <- rnorm(10000, mean(experimental), sd(experimental))
norm2 <- rnorm(10000, mean(experimental), 2*sd(experimental))
png(filename = "qqplot2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
norm1.q <- quantile(norm1, probs = seq(.05, .95, .01))
```

```
norm2.q <- quantile(norm2, probs = seq(.05, .95 , .01))
min <- floor(min(norm1.q, norm2.q))
max <- ceiling(max(norm1.q, norm2.q))
plot(norm1.q, norm2.q,
      xlim = c(min, max),
      ylim = c(min, max),
      ylab = "Quantiles of Normal Distribution with Twice the SD",
      xlab = "Quantiles of Normal Distribution",
      pch  = 19,
      col  = rgb(0,0,0,0.5))
abline(0,1)
dev.off()
```

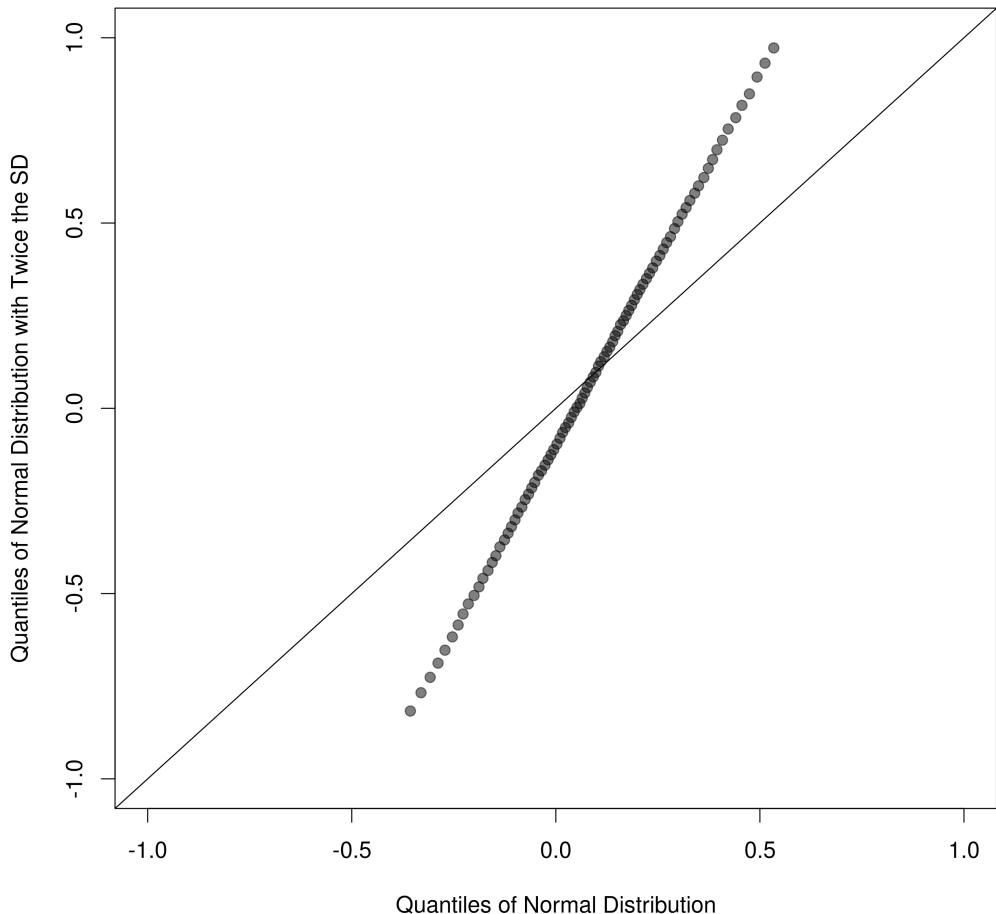


Figure 2.12: Plot of quantiles from a simulated normal distribution and simulated normal distribution with twice the standard deviation.

As we can see, the line is steeper than the 45 degree diagonal line. Looking at the lowest quantile (0.05), our distribution with the greater standard deviation has a lower value (-0.82) than the distribution with the smaller standard deviation (-0.36). Likewise, the highest value is higher for the normal distribution with greater standard deviation (0.97), compared to the normal distribution with smaller standard deviation (0.53). So, a distribution that is uniformly more “spread out” than another will show up as a steep slope on the qq-plot. If we reverse the axes the opposite happens, and the graph flattens out:

```
png(filename = "qqplot3.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
```

```
min <- floor(min(norm1.q, norm2.q))
max <- ceiling(max(norm1.q, norm2.q))
plot(norm2.q, norm1.q,
      xlim = c(min, max),
      ylim = c(min, max),
      ylab = "Quantiles of Normal Distribution",
      xlab = "Quantiles of Normal Distribution with Twice the SD",
      pch  = 18,
      col  = rgb(0, 0, 0, .5))
abline(0, 1)
dev.off()
```

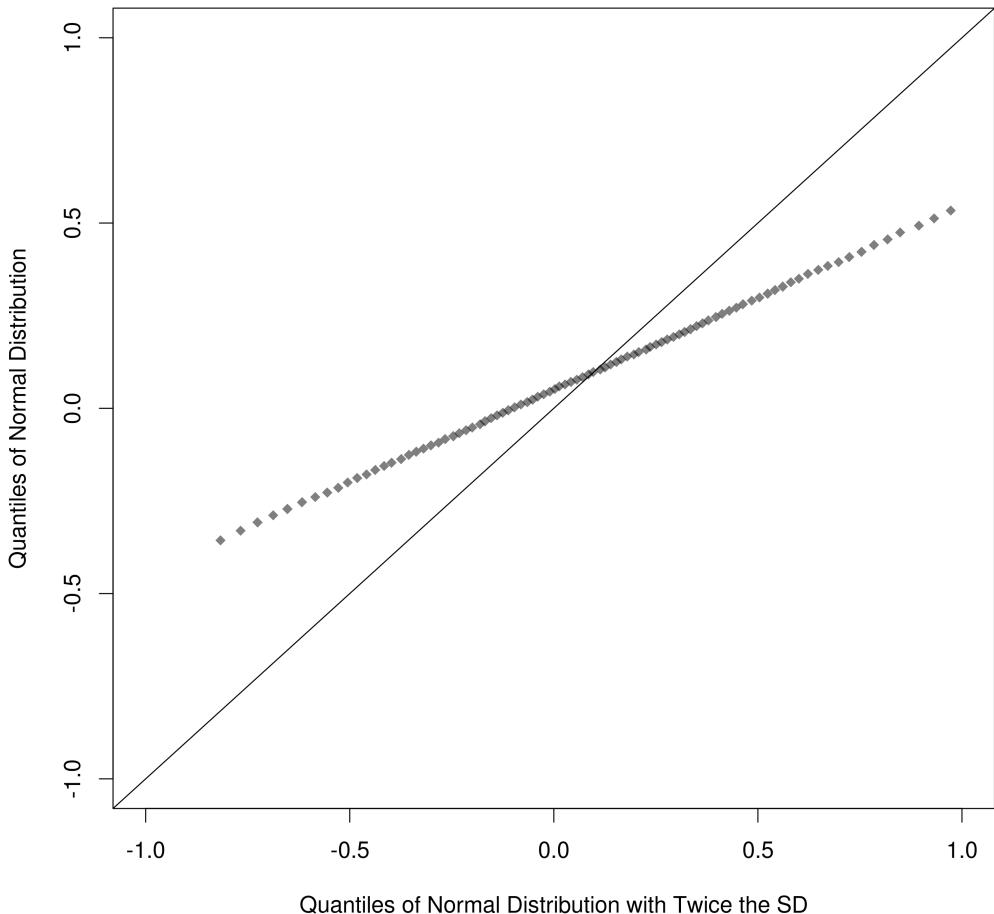


Figure 2.13: Plot of quantiles from a simulated normal distribution and simulated normal distribution with twice the standard deviation on the x axis.

Thus, a uniformly less variable, or more “compact” distribution will show up as a line with a smaller slope than the diagonal on the qqplot. As you should expect, mean shifts in distributions show up as parallel lines above or below the diagonal:

```

norm1 <- rnorm(10000, mean(experimental), sd(experimental))
norm2 <- rnorm(10000, mean(experimental) + 1, sd(experimental))

png(filename = "qqplot4.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
norm1.q <- quantile(norm1, probs = seq(.05, .95, .01))
norm2.q <- quantile(norm2, probs = seq(.05, .95, .01))

```

```
min<-floor(min(norm1.q, norm2.q))
max<-ceiling(max(norm1.q, norm2.q))
plot(norm1.q, norm2.q,
      xlim = c(min, max),
      ylim = c(min, max),
      ylab ="Quantiles of Normal Distribution with Mean Shift",
      xlab = "Quantiles of Normal Distribution",
      pch  = 17,
      col  = rgb(0, 0, 0, .5))
abline(0, 1)
dev.off()
```

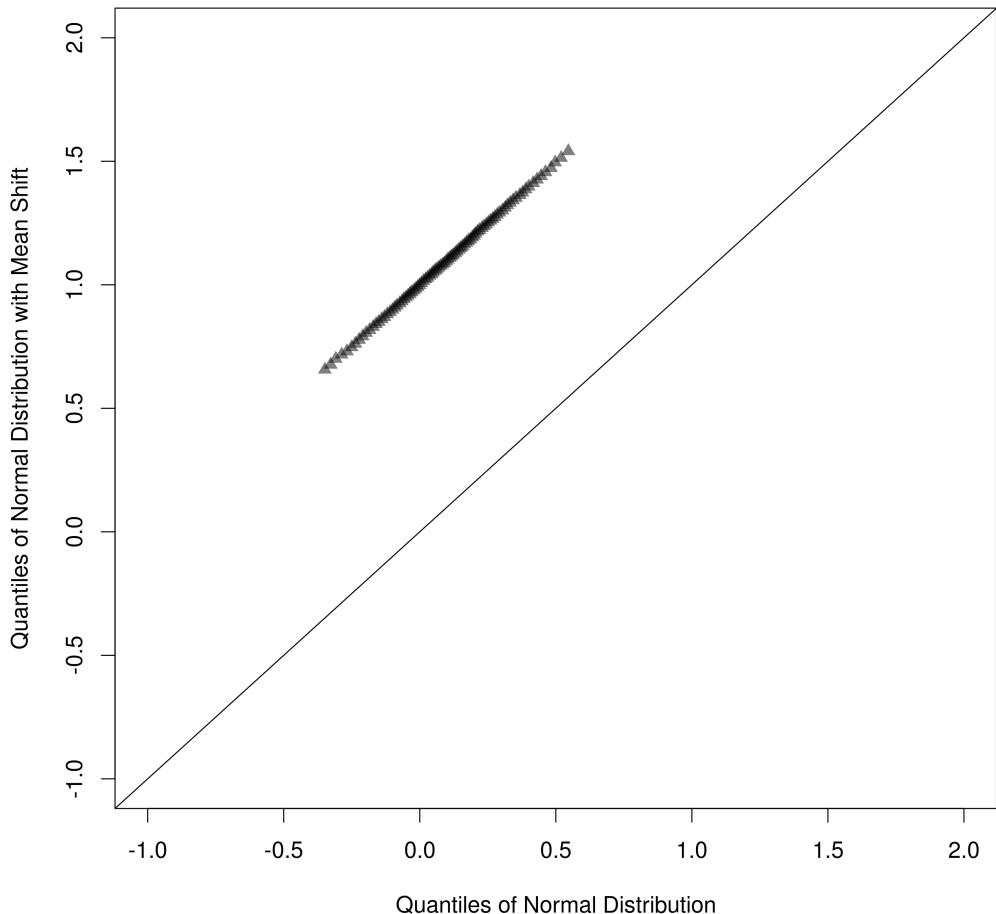


Figure 2.14: Plot of quantiles from a simulated normal distribution (x axis) and simulated normal distribution with mean shift of +1 (y axis).

Using our powers of induction, we can conclude that *any straight lines, when compared to an alternative distribution in a qqplot, are not much of a problem*. The reason is that a simple linear transformation can change one distribution into another, if the qqplot is linear. As we've seen so far, the qqplot can give us an idea about difference in means and differences in spread. In statistics differences in means are sometimes called differences in *location*, while differences in spread are called differences in *scale*. A transformation that corrects for these location and scale differences shows that the two distributions are from the same *location-scale* family, and can move any straight line in a qqplot onto the diagonal.

Location-scale differences in distribution show up as shifted straight lines on the qqplot. *Non-linearities* in qq-plots can signal serious differences in distribution, including *skewness*, *multi-modality*, or *heavy tails*.

Skewness means that a distribution is clumped to the left or right of a histogram and will show up as a non-linearity on a qqplot. Consider the log-normal distribution, which has a very condensed, short left tail, and a long right tail, and thus is positively skewed:

```
x <- seq(0, 10, length = 100)
hx <- dlnorm(x, 0, 1) # transform x using the lognormal pdf
png(filename = "lognorm1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(x, hx,
      xlab = "x",
      ylab = "lognorm(x)",
      pch = 16,
      col = rgb(0, 0, 0, .5))
lines(x, hx) # connect the dots
dev.off()
```

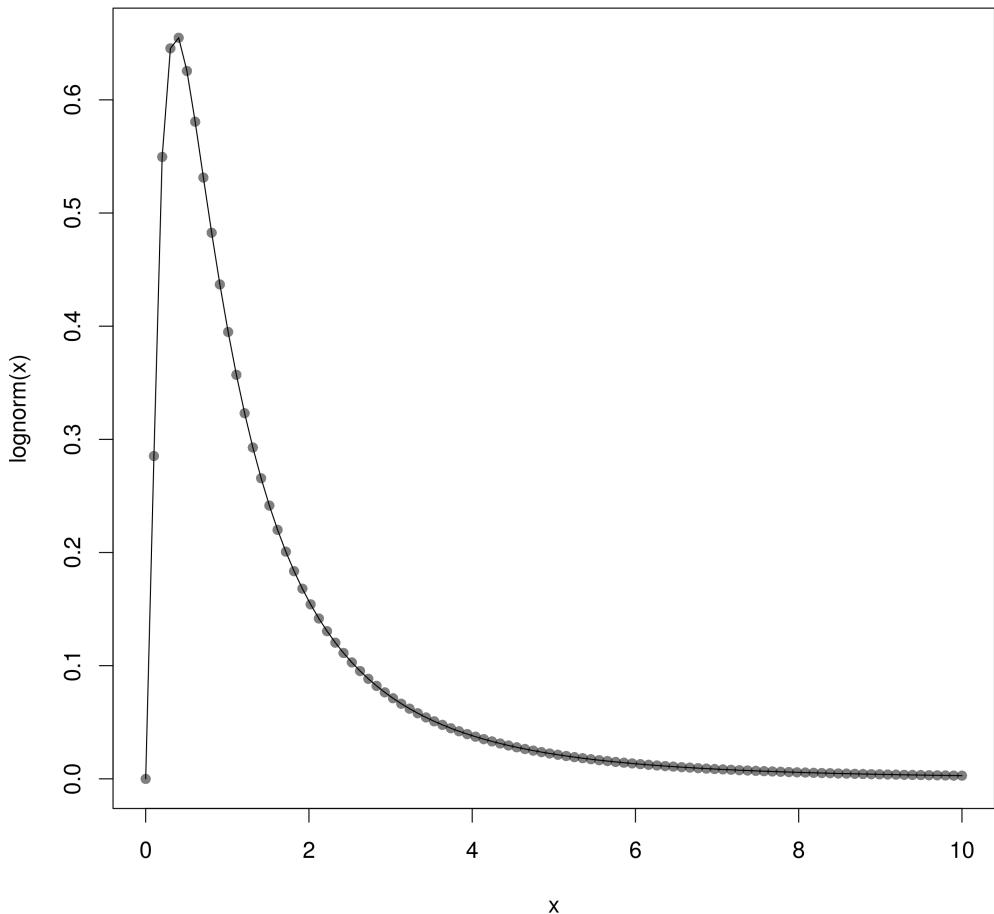


Figure 2.15: Probability density function of the log-normal distribution, showing positive skew, with high density to the left of the graph (toward zero) and a long right tail.

Because the left side of the distribution is compressed, it should be above the diagonal on the qqplot, meaning the first quantile is not extreme enough. On the other hand, because the right side of the distribution is spread out, the highest quantile should also be above the diagonal:

```
# generate 10000 draws from a lognormal distribution
# with mean(log(x)) = 0 and sd(log(x)) = 1
set.seed(4)
lognorm1 <- rlnorm(10000, 0, 1)
norm1 <- rnorm(10000, 0, 1)
```

```

png(filename = "qqplot5.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 1), mar = c(4, 4, 2, 0))
par(cex = 1.3)
norm1.q <- quantile(norm1, probs = seq(.05, .95, .01))
lognorm1.q <- quantile(lognorm1, probs = seq(.05, .95, .01))
min <- floor(min(norm1.q, lognorm1.q))
max <- ceiling(max(norm1.q, lognorm1.q))
plot(norm1.q, lognorm1.q,
      xlim = c(min, max),
      ylim = c(min, max),
      ylab = "Quantiles of Log-Normal",
      xlab = "Quantiles of Normal Distribution",
      pch = 16,
      col = rgb(0, 0, 0, .5))
abline(0, 1)

# Log transform the log-normal observations
loglognorm1.q <- quantile(log(lognorm1), probs = seq(.05, .95, .01))
min <- floor(min(norm1.q, loglognorm1.q))
max <- ceiling(max(norm1.q, loglognorm1.q))
plot(norm1.q, loglognorm1.q,
      xlim = c(min, max),
      ylim = c(min, max),
      ylab = "Quantiles of Logged Log-Normal",
      xlab = "Quantiles of Normal Distribution",
      pch = 16,
      col = rgb(0, 0, 0, .5))
abline(0, 1)
dev.off()

```

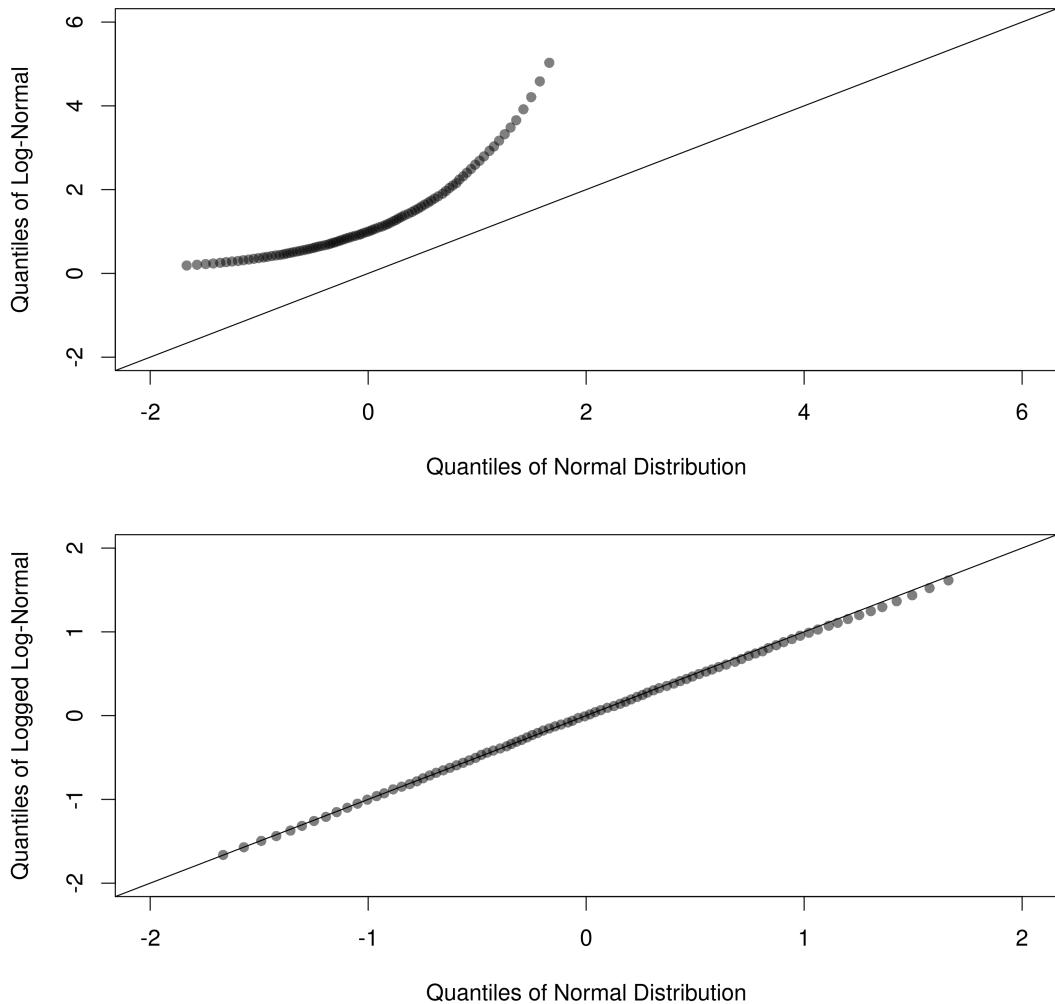


Figure 2.16: Plot of quantiles from a simulated normal distribution, simulated log-normal distribution (top), and log-transformed simulated log-normal distribution (bottom).

Often a non-linear transformation can make the non-linear qq plot linear. Consider the case of the normal and log-normal distribution. The log-normal distribution is normal in natural logs, and thus is a non-linear transformation of the normal distribution (causing skew). When we take the log of the lognormal, the qqplot versus the normal distribution looks linear, meaning the log of the lognormal and the normal are a location scale family.

Heavy Tails

Lastly, let's compare a heavy-tailed distribution, the t-distribution, with the normal. Keep in mind, that the heavy-tails are symmetric here (we can also

have asymmetric heavy tails, which should look similar to a skewed distribution):

```
# draw from t distribution with 10 degrees of freedom
set.seed(5)
tdist <- rt(10000, df = 10)
norm1 <- rnorm(10000, 0, 1)

png(filename = "qqplot8.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
norm1.q <- quantile(norm1, probs = seq(0, 1, .001))
tdist.q <- quantile(tdist, probs = seq(0, 1, .001))
min <- floor(min(norm1.q, tdist.q))
max <- ceiling(max(norm1.q, tdist.q))
plot(norm1.q, tdist.q,
      xlim = c(min, max),
      ylim = c(min, max),
      ylab = "Quantiles of t-distribution with 10 df",
      xlab = "Quantiles of Normal Distribution",
      pch = 15,
      col = rgb(0, 0, 0, .5))
abline(0, 1)
lines(norm1.q, tdist.q)
dev.off()
```

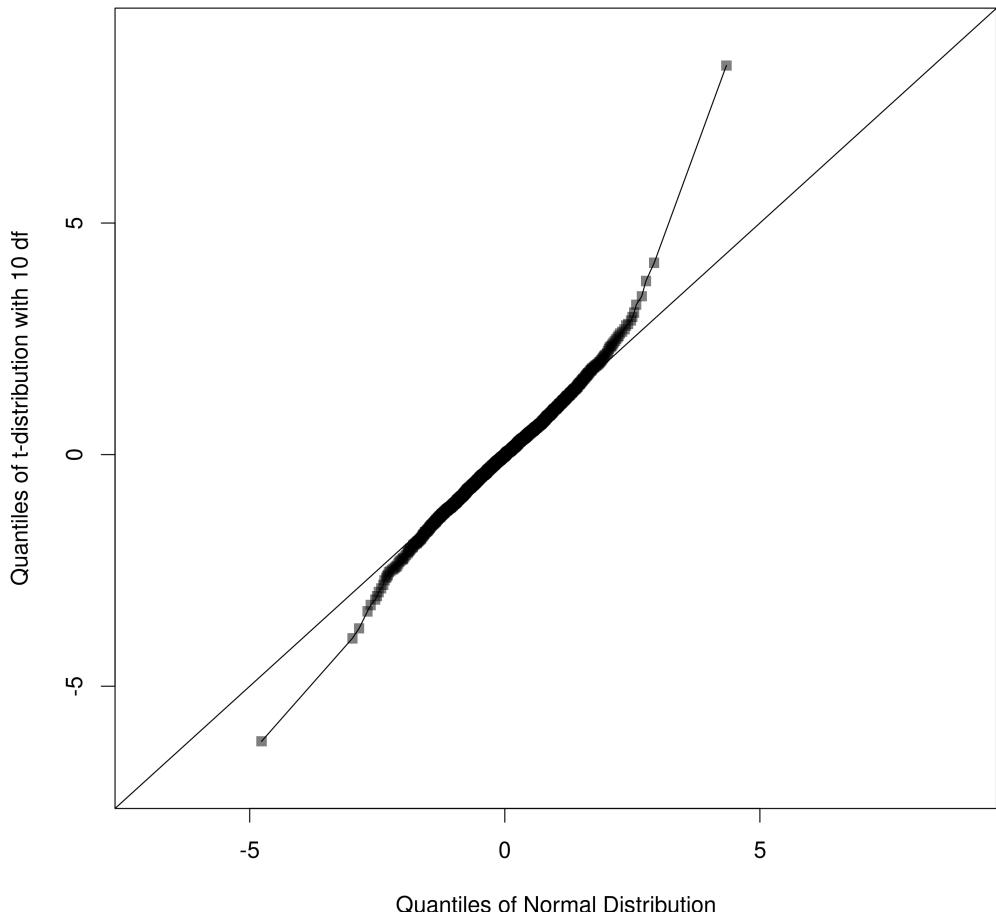


Figure 2.17: Plot of quantiles from a simulated normal distribution and t-distribution with 10 degrees of freedom.

The signature of a symmetric heavy-tailed distribution is more extreme quantiles on both ends of the plot. Most of the quantiles are along the straight line. However, the lowest quantile is below the diagonal, and the highest quantile above. The reverse pattern, with lowest quantile above and highest quantile below the diagonal, would be a short-tailed distribution (one that is too compressed). The quantile-quantile comparisons can be extended to compare any exponential family distribution against sample data.¹⁰

¹⁰Here's some info on different distributions we might be interested in: <http://www.statmethods.net/advgraphs/probability.html>, <http://rwiki.sciviews.org/doku.php?id=tips:stats-distri:Overview&rev=1183296386>

Another way to compare empirical distributions is to plot their cumulative distributions against each other. It is generally more difficult to spot patterns in plots of cumulative distributions than the qqplot, but is important to look at to understand the Kolmogorov-Smirnov test, probably the only test we'll cover in this course:

```
n.experimental <- rnorm(10000, mean(experimental), sd(experimental))
n.section8 <- rnorm(10000, mean(section8), sd(section8))
n.control <- rnorm(10000, mean(control), sd(control))

png(filename = "normalcdf.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(1, 3), cex = 1.3, oma = c(5, 4, 3, 0), mar = c(0, 1, 2, 0))
plot(ecdf(experimental),
      xlab = "Mean Happiness Z-Score",
      col = "red",
      xlim = c(-1, 1),
      pch = 21,
      main = "Experimental")

plot(ecdf(n.experimental),
      xlab = "Mean Happiness Z-Score",
      col = "blue",
      pch = 21,
      add = TRUE)

plot(ecdf(section8),
      xlab = "Mean Happiness Z-Score",
      ylab = "",
      col = "red",
      xlim = c(-1, 1),
      pch = 21,
      main = "Section 8",
      yaxt = 'n')

plot(ecdf(n.section8),
      xlab = "Mean Happiness Z-Score",
      col = "blue",
      pch = 21,
      add = TRUE)

plot(ecdf(control),
      xlab = "Mean Happiness Z-Score",
      col = "red",
```

```
ylab = "",  
xlim = c(-1, 1),  
pch = 21,  
main = "Control Group",  
yaxt = 'n')  
  
plot(ecdf(n.control),  
      xlab = "Mean Happiness Z-Score",  
      col = "blue",  
      pch = 21,  
      add = TRUE)  
  
title(main = "Empirical (Red) vs. Normal (Blue) Cumulative Distributions",  
      sub = "Mean Happiness Z-Score",outer=TRUE,  
      ylab = "Cumulative Probability")  
  
dev.off()
```

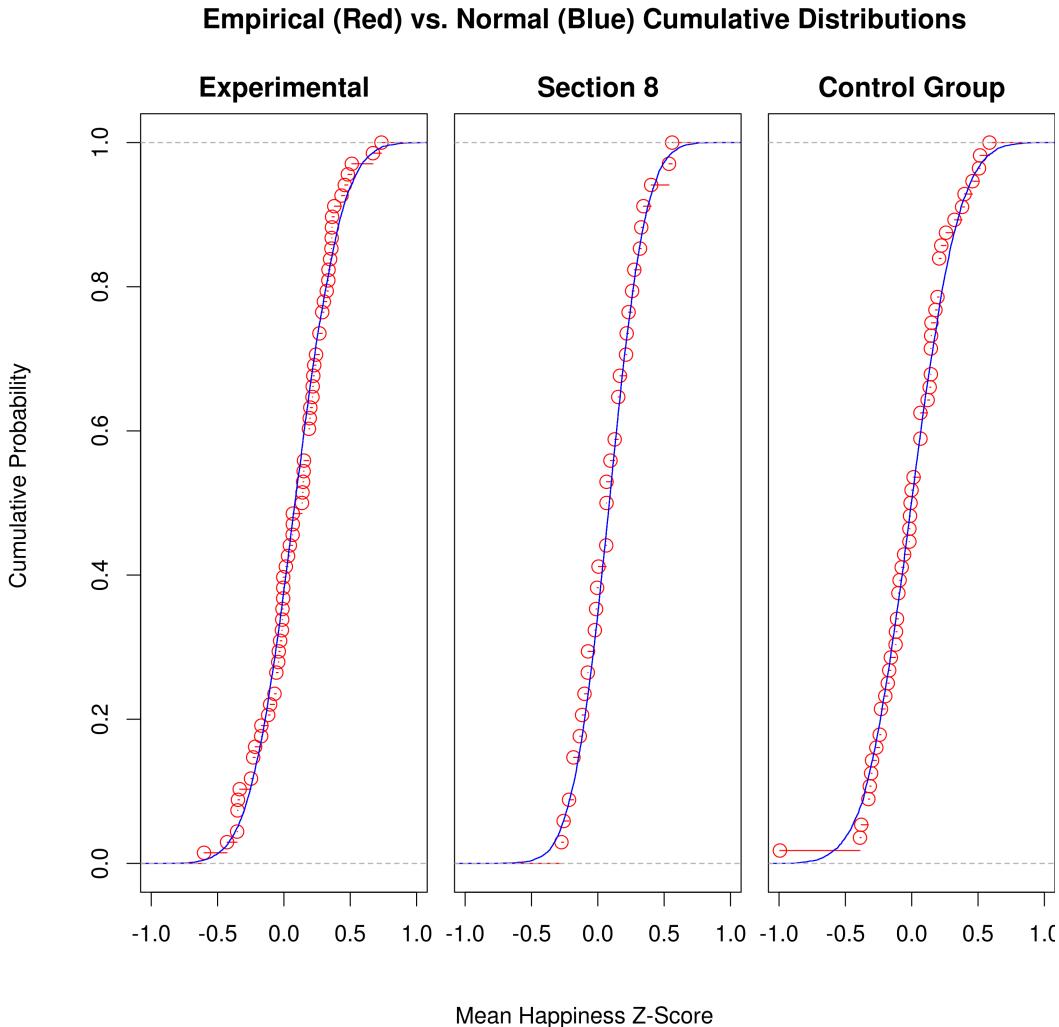


Figure 2.18: Plot of the empirical cumulative distribution for the three treatment groups (red) versus the normal distribution with equivalent mean and standard deviation.

From these plots there looks like a very close correspondence between the cumulative distribution of the data simulated from the normal distribution and the observed data. We can also do more formal tests, using the Kolmogorov-Smirnov test. This test looks at the maximum vertical difference between the cdfs. From the plots, it is hard to tell where the maximum vertical distance is for the experimental group, probably at the upper end, around 0.3 where the red dots are above the blue line. For the section 8 group, the maximum vertical difference seems to be at the lowest scores: The blue line is above the red dots with some very small separation. For the control group the maximum vertical difference could be at the lowest values,

where there is higher density in the lower values than the normal distribution, possibly indicating a heavy tail.

```
# Compare empirical cdfs against the normal distribution with same mean and sd
ks.test(experimental, "pnorm", mean(experimental), sd(experimental))
ks.test(section8, "pnorm", mean(section8), sd(section8))
ks.test(control, "pnorm", mean(control), sd(control))
```

So what we can see is that the data that our groups fall roughly into quantiles that we would expect from the normal distribution. The Kolmogorov-Smirnov test (K-S test for short) indicates that the differences between the empirical cumulative distribution and the normal distribution is not significant.

In general, such significance tests have serious problems: If we have a small sample, they will rarely tell us that our distribution is different from normal, and if we have a large sample, they will always tell us this. That is, *hypothesis tests are roughly a measure of sample size. With a small sample they will never reject the null hypothesis no matter how large the effect. With a large one, they always will, no matter how trivial the effect.*

2.4.2 Continous Independent Variables

So far we've looked at the distribution of a single dependent variable, mean happiness z-score, across discrete groups (experimental, section 8, control). We also often have data from an independent variable that varies continuously rather than discretely. Additional tools are available here, focusing on how to fit some sort of line or curve to the scatterplot. This line helps us understand the conditional distribution of the dependent variable at different levels of the independent variable, and is usually called *regression analysis*. This can be done for the entire conditional distribution, or other important statistics, such as the conditional mean, median, variance, etc. We usually use the conditional mean, although this might not be the most relevant part of the conditional distribution to focus on, depending on the application.

Back to our example data. We have an independent variable, economic self-sufficiency. Are economic self-sufficiency and happiness related? Let's look at the scatterplot:

```
png(filename = "scatterline.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, oma = c(0, 0, 0, 0))
plot(mto.data$selfsuff, mto.data$happy,
      ylab = "Mean Happiness Z-Score",
      xlab = "Mean Economic Self-Sufficiency Z-Score",
      xlim = c(-1, 1),
```

```
ylim = c(-1, 1),  
pch = 19,  
col = rgb(0, 0, 0, .5))  
dev.off()
```

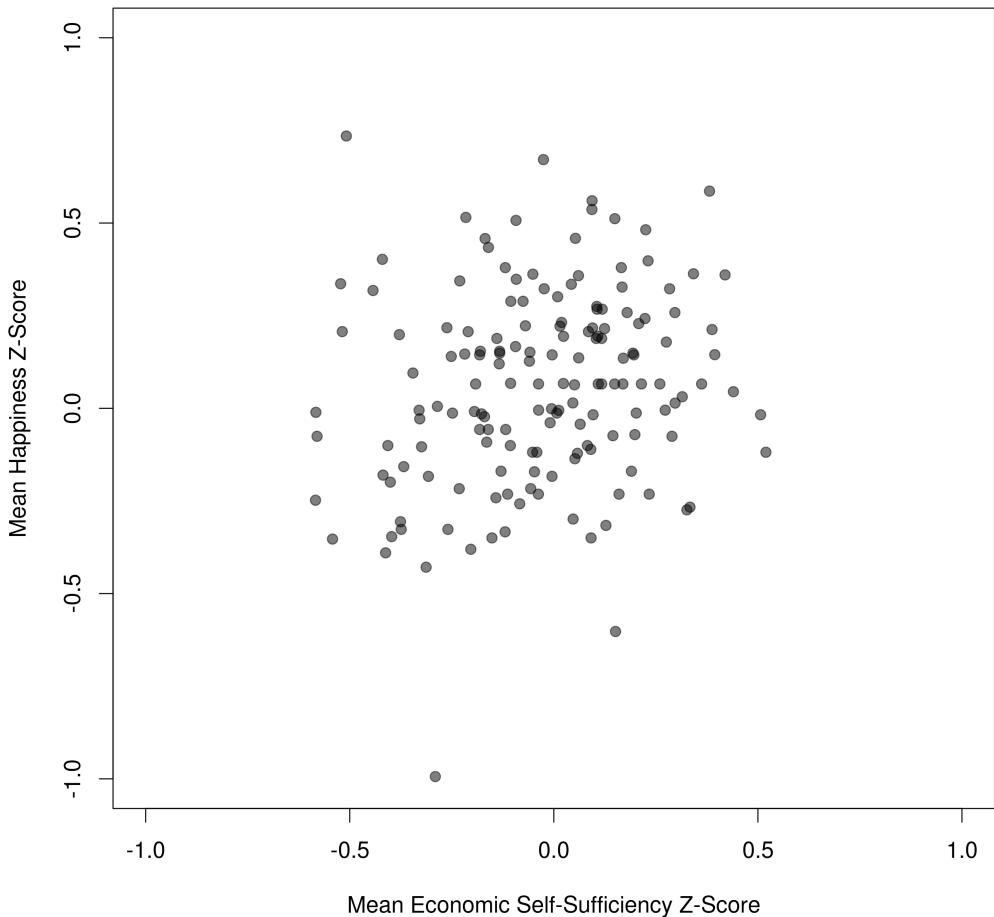


Figure 2.19: Scatterplot of mean happiness z-scores versus mean economic self-sufficiency z-scores.

From the graph, it seems like the answer is no; it's hard to discern a pattern.

Least Squares Regression Line

To help our eyes, we can fit a regression line to estimates the mean happiness z-score for each level of economic self-sufficiency. The constraint is that the line must be globally linear, and that it minimizes the *squared* deviations from the line (i.e., the line is fit according to a squared loss function, hence linear regression is often called *ordinary least squares* or OLS). More

specifically, suppose there is actually a population regression function:

$$y_i = \beta_0 + \beta_1 x_i + e_i \quad (2.7)$$

where e_i is uncorrelated with x_i . We want to estimate this population regression function with a sample counterpart:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \quad (2.8)$$

The simple least squares regression finds the values $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimizes the sum of squared residuals, where the residual is the difference between the actual value y_i and the fitted value \hat{y}_i :

$$\operatorname{argmin}_{\hat{\beta}_0, \hat{\beta}_1} (y_i - \hat{y}_i)^2 \quad (2.9)$$

Because $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$, this is the same as solving:

$$\operatorname{argmin}_{\hat{\beta}_0, \hat{\beta}_1} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \quad (2.10)$$

Lucky for us there is a unique solution. In the case of simple regression:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \quad (2.11)$$

and

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (2.12)$$

Notice the similarity between $\hat{\beta}_1$ and Pearson's r . Looking at the numerator, both are measuring the degree to which x and y are simultaneously above or below their mean (the covariance between x and y). If we have multiple regressors as columns of a matrix X (with the first column as a vector of 1's to estimate the intercept) then we can write the solution for the vector $\hat{\beta}$ in matrix form:

$$\hat{\beta} = (X' X)^{-1} X' y \quad (2.13)$$

We can easily calculate this in R:

```
# Response vector Y as the dependent variable
Y <- mto.data$happy
# create a vector of ones to estimate the intercept beta0
ones <- rep(1, nrow(mto.data))
# bind that vector to self-sufficiency by column
X <- as.matrix(cbind(ones, mto.data$selfsuff))

# solve(X) gives you matrix inverse of X
# t(X) gives you matrix transpose of X
# X%*%Y gives you matrix multiplication of X and Y
beta.hat <- solve(t(X) %*% X) %*% t(X) %*% Y
```

We can verify that this gives us the correct solution by comparing it to the lm function in R which runs a linear regression:

```
mto.reg <- lm(happy ~ selfsuff, data = mto.data)
```

The solution for $\hat{\beta}$ is identical. We can add the least squares regression line to our plot:

```
png(filename = "scatterline2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, oma = c(0, 0, 0, 0))
plot(mto.data$selfsuff,mto.data$happy,
      ylab = "Mean Happiness Z-Score",
      xlab = "Mean Economic Self-Sufficiency Z-Score",
      xlim = c(-1, 1),
      ylim = c(-1, 1),
      pch = 19,
      col = rgb(0, 0, 0, .5))
# Run linear regression of happiness on self-sufficiency
mto.reg <- lm(happy ~ selfsuff, data = mto.data)
abline(mto.reg) # Plot fitted regression line
dev.off()
```

There seems to be a small, but positive slope to the line. Looking at the summary of the regression line:

```
# Print summary of the regression
summary(mto.reg)
```

This shows us that the equation for our regression line has an intercept of 0.065, with a slope of 0.20:

$$\text{Happiness} = 0.065 + 0.20 \times \text{Economic Self-Sufficiency}$$

Let's interpret the intercept. For a cell with an average z-score of zero economic self-sufficiency (that is, they are exactly at the mean of the control group), the average happiness of people in that cell is 0.065 (in z-score units), that is, 0.065 standard deviations higher than the control group average. However, we can see from the scatterplot that cells with approximately zero mean self-sufficiency range in happiness from about -0.3 to 0.6. Thus, knowing the average economic self-sufficiency in a cell doesn't tell us a lot about the average happiness of those in the cell, as there is wide variability in happiness among those with equal levels of self-sufficiency.

Next, let's interpret the slope. The slope is 0.20, meaning that, for any level of mean economic self sufficiency, cells with a mean economic self-sufficiency that is 1 unit higher than another cell also has, on average, an average happiness of 0.20 units higher.

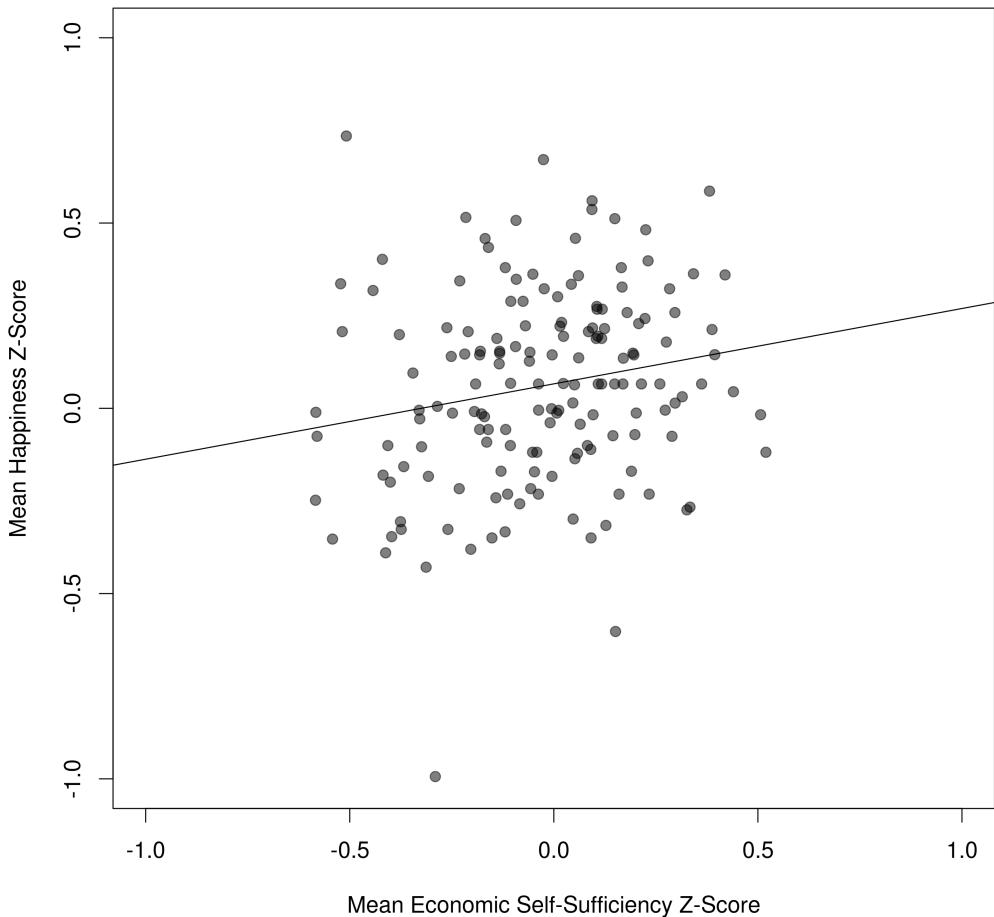


Figure 2.20: Scatterplot of mean happiness z-scores versus mean economic self-sufficiency z-scores with least squares regression line added.

Root Mean Squared Error

R's summary function also gives us information on how variable happiness scores are for those with equal economic self-sufficiency. Let's call the mean squared error MSE, where:

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n - k} \quad (2.14)$$

and k is the number of estimated parameters in our regression model (in our case $k = 2$ for the slope and intercept). The "residual standard error," sometimes also called the standard error of the model, standard error of the estimate, or root-mean-squared error just tells us the average standard

deviation of the points above and below the regression line, assuming the standard deviation is homogenous across levels of the independent variable. The equation is:

$$\sqrt{MSE} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n - k}} \quad (2.15)$$

Where y_i is the observed value, \hat{y}_i is the predicted value of the line (also called fitted value), n is the total number of observations, and k is the number of parameters (predictors plus intercept) in the model. You can calculate this number pretty easily. First, make a prediction from the model:

```
# Make predictions for all the mto.data from the mto.reg regression
predictions <- predict(mto.reg, mto.data)
```

Then calculate the residual, which is the difference between the actual values and the model's predictions:

```
residuals <- mto.data$happy - predictions
```

Then, the mean squared error (MSE) is the sum of the squared the residuals divided by $n - k$:

```
# nrow(x) tells us the number of rows in vector, array or data frame x
MSE <- sum(residuals^2)/(nrow(mto.data) - 2)
```

Then root-MSE is the square-root of the MSE:

```
rMSE <- sqrt(MSE)
```

Giving us a root-MSE of 0.2589, the same as the “residual standard error” from the output of the `summary()` function, to rounding error.¹¹ What does this mean? The number 0.259 indicates that, on average, 66% of observations fall between ± 0.259 of the regression line. This comes from the fact that 66% of samples drawn from a normal distribution are within ± 1 standard deviation of the mean, making it clear that this interpretation of the root-MSE depends on an assumption of normally distributed errors.

This is quite a wide margin of error, given that the total range of the happiness scores is 1.73, and the standard deviation of the happiness scores is 0.26. Thus, happiness scores are about as widely dispersed if we know economic self-sufficiency as if we don't know it.

¹¹There's also other information provided by our `summary()`, including p-values, degrees of freedom, R-squared, multiple R-squared, adjusted R-squared, the F-statistic, etc. For the purpose of understanding the conditional distribution story, this stuff is distraction.

It's important to note that linear regression, when fit this way, assumes the conditional distributions are the same except for their means. In other words, linear regression is a model of conditional distributions with a *single* error that is independent of the levels of the independent variable. This is called *homoskedasticity*, or constant spread. In our case, the residual standard error is 0.259, meaning our model of the conditional distribution of mean happiness z-scores as a function of mean economic self-sufficiency z-scores has a constant spread (standard deviation) of 0.259 for all levels of mean economic self-sufficiency z-scores.

Simulating a Linear Conditional Distribution

A useful tool for checking our conditional distribution story (i.e., linear model) is to compare simulations from the model against the data. We are looking for problems with our conditional distribution story in the form of patterns in the data that do not fit with our simulations. We can simulate our conditional distribution story in the following way:

```
set.seed(7)
# Use seq to draw a sequence of values from -0.6 to 0.52 in increments of 0.05
# use rep to repeat this process 4 times
selfsuff <- rep(seq(-0.6, 0.52, by = 0.05), 4)
error <- rnorm(length(selfsuff), 0, 0.259)
happy <- 0.065 + 0.20*selfsuff + error
```

We've used a span of our self-sufficiency variable from -0.6 to 0.52 and drawn an error term for each observation. We then create our happiness variable as the linear combination of the intercept, self-sufficiency, and residual error. Plotting this against the actual data:

```
png(filename = "scattersim.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(mto.data$selfsuff, mto.data$happy,
     ylab = "Mean Happiness Z-Score",
     xlab = "Mean Economic Self-Sufficiency Z-Score",
     xlim = c(-1, 1),
     ylim = c(-1, 1),
     pch = 19,
     col = rgb(0, 0, 0, .5))
points(jitter(selfsuff), jitter(happy),
       pch = 19,
       col = rgb(1, 0, 0, 0.5))
dev.off()
```

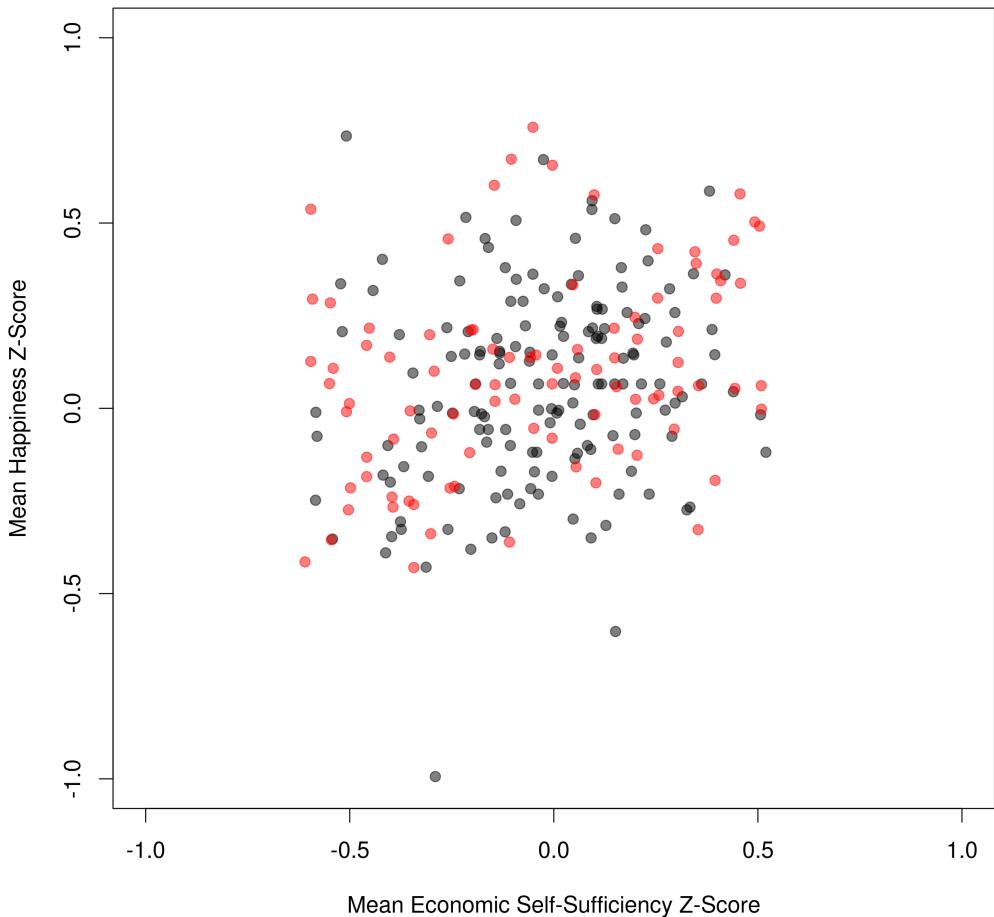


Figure 2.21: Plot of mean economic self-sufficiency z-scores (x-axis) against mean happiness z-scores (y-axis) with observed data (black) and data simulated from the linear model (red).

The distributions look very similar, meaning our conditional distribution story is not too bad. However, there are two points, at the upper left and lower left, that seem to not be modeled well. Let's draw many more points from our conditional distribution model and see if these points pop out:

```
set.seed(8)
selfsuff <- rep(seq(-0.6, 0.52, by = 0.05), 20)
errors <- rnorm(length(selfsuff), 0, 0.259)
happy <- 0.065 + 0.20*selfsuff + errors

png(filename = "scattersim2.png", width = 3000, height = 3000, res = 300)
```

```
par(cex = 1.3)
plot(mto.data$selfsuff, mto.data$happy,
      ylab = "Mean Happiness Z-Score",
      xlab = "Mean Economic Self-Sufficiency Z-Score",
      xlim = c(-1, 1),
      ylim = c(-1, 1),
      pch = 19,
      col = rgb(0, 0, 0, .5))
points(jitter(selfsuff), jitter(happy),
      pch = 19,
      col = rgb(1, 0, 0, .5))
dev.off()
```

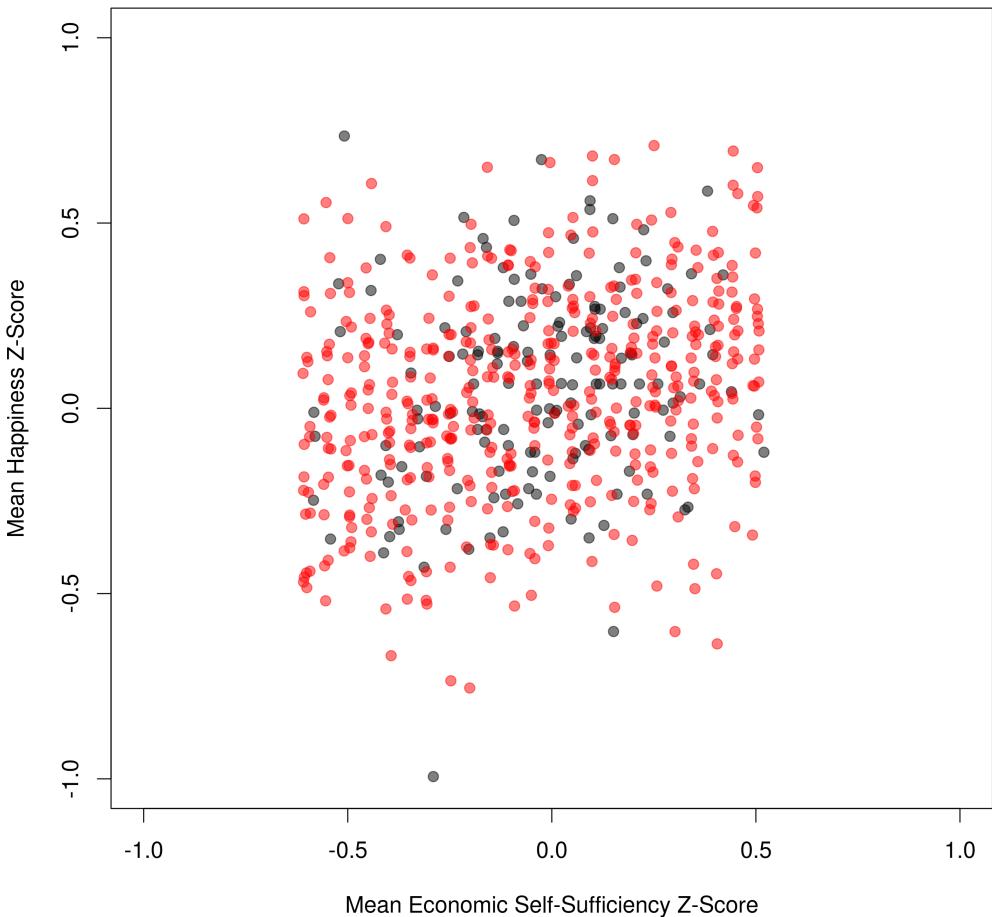


Figure 2.22: Plot of mean economic self-sufficiency z-scores (x-axis) against mean happiness z-scores (y-axis) with observed data (black) and data simulated from the linear model (red).

The observation at the top doesn't seem too implausible, but still doesn't seem to be captured well. However, the observation at the bottom is not captured well at all, even at 20 sequences.

Here, we've taken our intercept and self-sufficiency regression coefficients to be deterministic. If we wanted to include uncertainty more comprehensively, we could also draw the regression parameters from their distributions.¹² We can start by assuming that the regression parameters (the intercept and slope) are drawn from a multivariate normal distribution with mean equal to the parameter estimates and variance-covariance matrix equal

¹²Check out [Gelman and Hill \(2007\)](#) chapters 7 and 8 for more on this.

to the estimated variance-covariance matrix:

```
install.packages("MASS", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(MASS)

set.seed(9)
reg.sim <- mvrnorm(n = 3,
                     mto.reg$coefficients, # pulls out vector of betas as the means
                     vcov(mto.reg), # The variance covariance matrix of betas
                     empirical = TRUE)

error1 <- rnorm(length(selfsuff), 0, 0.259)
happy1 <- reg.sim[1, 1] + reg.sim[1, 2]*selfsuff + error1

error2 <- rnorm(length(selfsuff), 0, 0.259)
happy2 <- reg.sim[2, 1] + reg.sim[2, 2]*selfsuff + error2

error3 <- rnorm(length(selfsuff), 0, 0.259)
happy3 <- reg.sim[3, 1] + reg.sim[3, 2]*selfsuff + error3
```

What we've done here is drawn the regression parameters from their distributions, then constructed several simulations based on those parameters:

```
png(filename = "scattersim3.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(mto.data$selfsuff, mto.data$happy,
     ylab = "Mean Happiness Z-Score",
     xlab = "Mean Economic Self-Sufficiency Z-Score",
     xlim = c(-1, 1),
     ylim = c(-1, 1),
     pch = 19,
     col = rgb(0, 0, 0, .5))

points(jitter(selfsuff), jitter(happy1),
       pch = 19,
       col = rgb(1, 0, 0, .5))

points(jitter(selfsuff), jitter(happy2),
       pch = 19,
       col = rgb(0, 1, 0, .5))

points(jitter(selfsuff), jitter(happy3),
       pch = 19,
       col = rgb(0, 0, 1, .5))
```

```
dev.off()
```

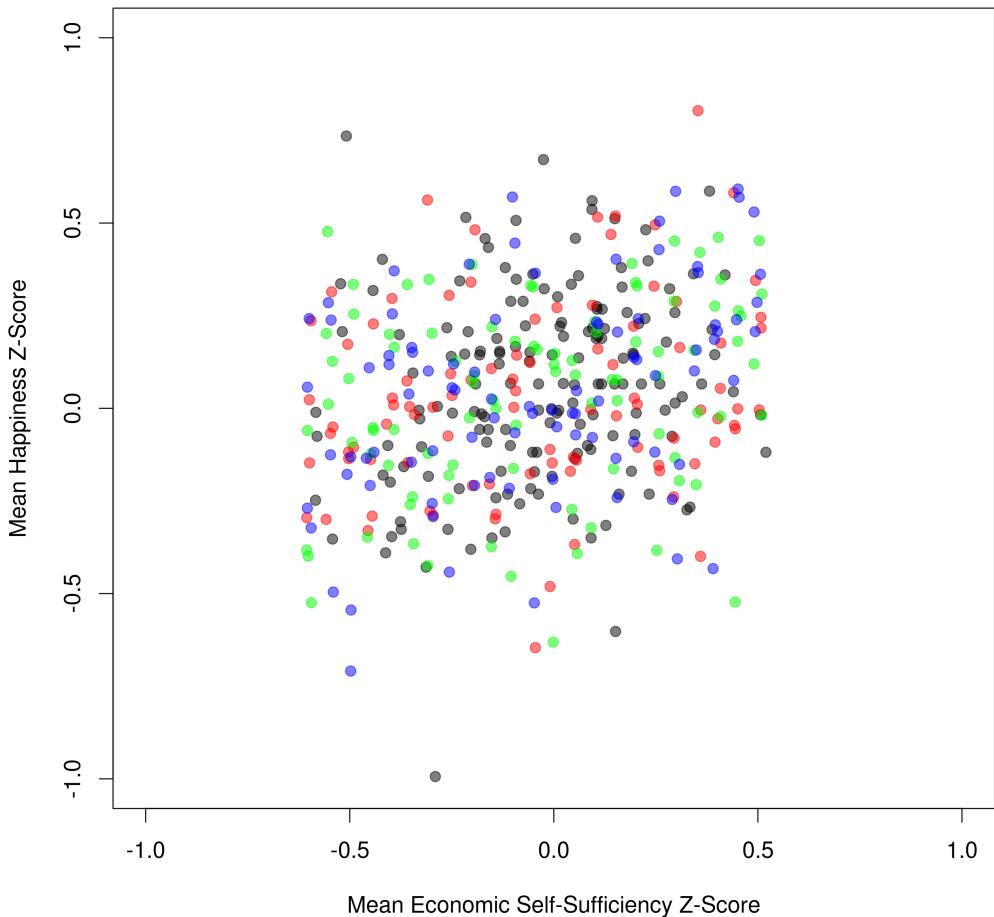


Figure 2.23: Plot of mean economic self-sufficiency z-scores (x-axis) against mean happiness z-scores (y-axis) with observed data (black) and data simulated from three simulated linear models (red, blue, and green).

Even after allowing the model parameters to vary according to their distributions, there's little evidence that the outlying case can be accounted for by the regression model. However, we would need to run more simulations to see how infrequently the observation occurs (e.g., does it occur in 1 out of 100 simulations?).

Q-Q Plot of the Regression Residuals

Another way to look at whether our conditional distribution story works well is to look at the distribution of the regression residuals. Let's look at the qqplots of the residuals for the simulated and real data. First, let's make a

dataframe for our simulated data:

```
simdata <- data.frame(selfsuff, happy)
mto.reg <- lm(happy ~ selfsuff, data = mto.data)
sim.reg <- lm(happy ~ selfsuff, data = simdata)
```

Now, let's qqplot the residuals of a regression on the original data and simulated data. Here we use a function from the *car* package that is specifically made for looking at qqplots of regression residuals. This package uses the assumption that the studentized regression residuals (we'll cover this shortly) have a student's t-distribution with $n - k - 1$ degrees of freedom, where n is the sample size, and k is the number of model parameters (including the intercept):

```
install.packages("car", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(car)

png(filename = "qqreg.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(1, 2), cex = 1.3, mar = c(5, 4, 4, 1))

# qqplot of the regression residuals from the real regression
# id.n = 3 identifies the three largest residuals
qqPlot(mto.reg,
       main = "Observed Data",
       id.n = 3,
       pch = 19,
       col = rgb(0, 0, 0, .5))
abline(0, 1)
qqPlot(sim.reg, main = "Simulated Data", id.n = 3,
       pch = 19,
       col = rgb(0, 0, 0, .5))
abline(0, 1)
dev.off()
```

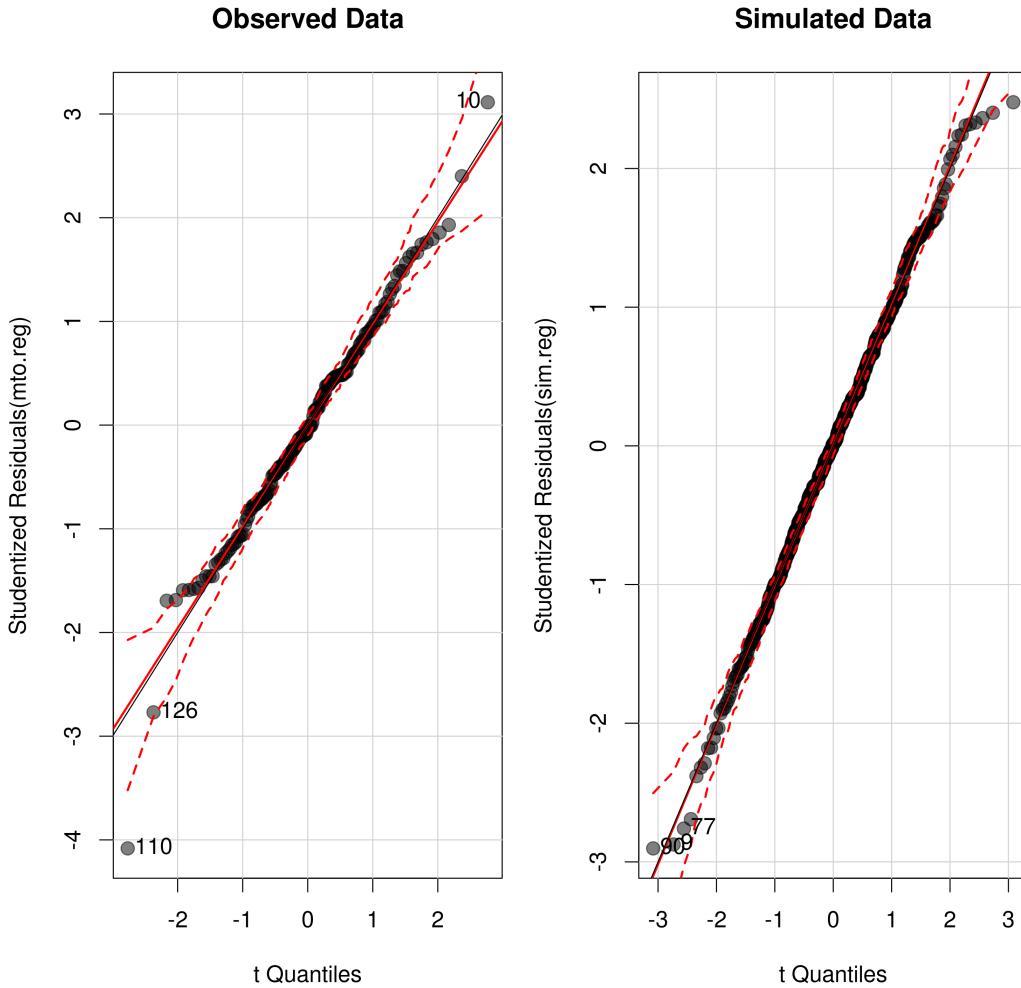


Figure 2.24: Q-Q plots of the studentized regression residuals (y-axis) plotted against quantiles of the t -distribution (x-axis) for the actual (left) and simulated (right) regressions. Red dotted lines show 95% confidence envelopes for simulations from the t distribution with $n - k - 1$ degrees of freedom.

The confidence envelopes can be understood using the following thought experiment: Simulate draws from a t distribution with $n - k - 1$ degrees of freedom repeatedly, then calculate each of the quantiles. The location of the quantiles will vary from sample to sample, and that variability is captured in the confidence envelopes.

These plots are kind of annoying because the axes are not equivalent, so it is harder to see the diagonal. However, we know that if the qqplot is linear, then the data are a location-scale family of the t -distribution, so a transformation could make them identical.

There is one observation, however, that seems to be a problem. The pattern of studentized residuals looks like a symmetric heavy-tailed distribution, which fits a t-distribution. However, the lowest quantile is very low compared to what we would expect from the t-distribution, indicating a lower (left) tail that is heavier than the upper (right) tail. We'll return to this shortly.

Before continuing, let's remember what linear regression does as a conditional distribution story ([Berk, 2004](#)). In our case, linear regression makes a commitment to modeling conditional means in the following way:

1. That the relationship between conditional means is linear,
2. That we are happy with a symmetric, quadratic loss function, rather than one that is asymmetric or linear.

Understanding the data in terms of conditional distributions using linear regression means we are making these commitments, and we should be careful to ask ourselves whether they are the ones we want to make. For example, we may care more about overprediction than underprediction, and should thus prefer something other than least squares, as least squares gives equal weight to all prediction errors, regardless of direction.

2.4.3 Outliers Mess Up the Conditional Distribution Story

Outliers seriously mess up our conditional distribution story, so the natural reaction is to try to destroy them at all costs. There are often no distributions that allow for all the kinds of crazy outliers that we see in applied data analysis projects, meaning real outliers make our conditional distribution story fail.

Recall that there seemed to be two observations that were not fit well by our simulated regression model. Each observation has two distinct characteristics with respect to the fitted linear regression line: *discrepancy* and *leverage*. The discrepancy is the model's residual, or some scaled version of the model's residual, and reflects how good a job we did at predicting that observation's value:

$$\text{Discrepancy}_i \propto y_i - \hat{y}_i$$

Where y_i is an observed value and \hat{y}_i is the predicted or fitted value. A larger discrepancy means a worse prediction, and larger residual.

Leverage

Leverage has to do with the distance between the observation's predictor value and the mean of the predictor values. With one independent variable, the leverage of an observation i is:

$$\text{Leverage}_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (2.16)$$

Here's a quick demonstration of leverage. We have a fixed vector of x values, and a vector of y values, where one y value changes for each vector for different levels of x :

```
x1 <- c(1, 2, 3, 4, 5)
y0 <- c(3, 3, 3, 3, 3)
y1 <- c(6, 3, 3, 3, 3)
y2 <- c(3, 6, 3, 3, 3)
y3 <- c(3, 3, 6, 3, 3)
y4 <- c(3, 3, 3, 6, 3)
y5 <- c(3, 3, 3, 3, 6)
```

```
png(filename = "leverage1.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(3, 2), cex = 1.3, mar = c(4, 4, 1, 1))
plot(x1, y0, ylim = c(0, 8), xlim = c(0, 6),)
abline(lm(y0 ~ x1), col = "blue")
abline(h = 3, col = "red") # add horizontal line at y = 3
plot(x1, y1, ylim = c(0, 8), xlim = c(0, 6),)
abline(lm(y1 ~ x1), col = "blue")
abline(h = 3, col = "red") # add horizontal line at y = 3
plot(x1, y2, ylim = c(0, 8), xlim = c(0, 6),)
abline(lm(y2 ~ x1), col = "blue")
abline(h = 3, col = "red") # add horizontal line at y = 3
plot(x1, y3, ylim = c(0, 8), xlim = c(0, 6),)
abline(lm(y3 ~ x1), col = "blue")
abline(h = 3, col = "red") # add horizontal line at y = 3
plot(x1, y4, ylim = c(0, 8), xlim = c(0, 6),)
abline(lm(y4 ~ x1), col = "blue")
abline(h = 3, col = "red") # add horizontal line at y = 3
plot(x1, y5, ylim = c(0, 8), xlim = c(0, 6),)
abline(lm(y5 ~ x1), col = "blue")
abline(h = 3, col = "red") # add horizontal line at y = 3
dev.off()
```

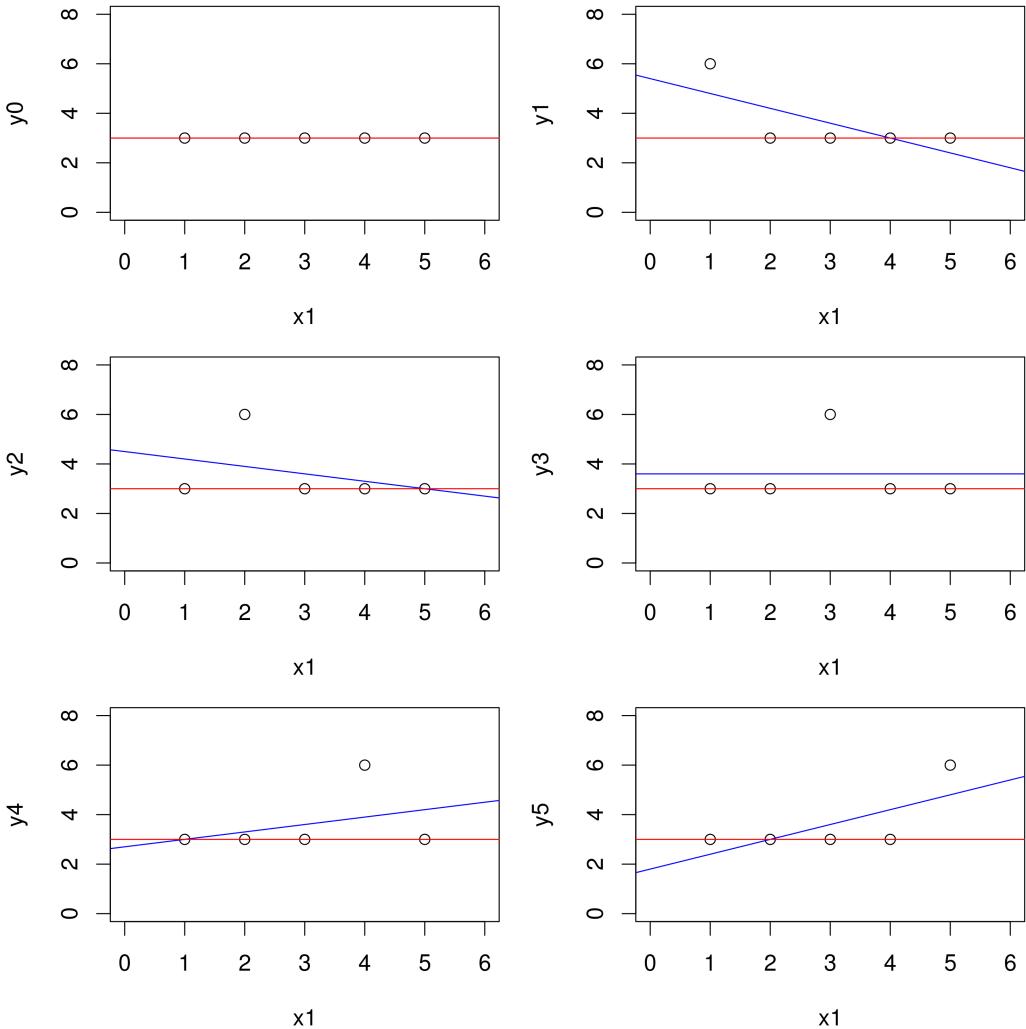


Figure 2.25: Leverage demonstration. The data are on a horizontal line except for a single observation, with the same y value. As we go from the top left to bottom right, the x value of that observation changes, showing that the same y value has a larger effect on the regression line when it is farther away from the mean of x .

Three Types of Outliers

When an observation has both high leverage and high discrepancy, it is *influential*, and will tend to have large effects on our conditional distribution story, as the observation draws the regression line toward it. The following figure shows three different types of outliers in terms of discrepancy and leverage:

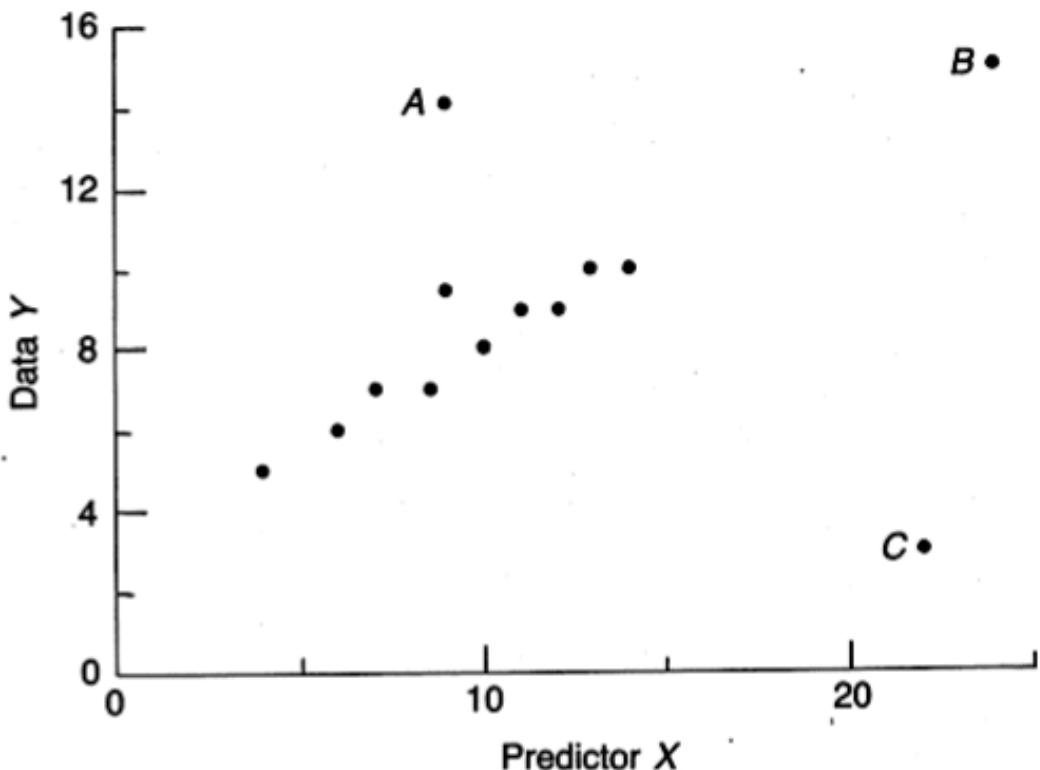


Figure 2.26: Source: Judd, C. M., McClelland, G. H. (1989). Data analysis: A model comparison approach. New York: Harcourt Brace Jovanovich.

Case A has an unusual value of Y, but a usual value of X. Case A has high discrepancy and low leverage. Linear regression will generally make a poor prediction for this case, and we should expect that the line is not drawn toward the observation much. Estimates of the conditional means are okay, except for the intercept, which will be shifted upward. This type of outlier will pretty much have the effect of increasing the model's MSE. If we want to identify Case A observations, then using the *Studentized deleted residual (aka Jackknife Residual)* will work.

Case B has an unusual value of X, and a plausible value of Y. Observation B has an unusual value of the predictor variable, but has low discrepancy from the model's predictions. Thus, Case B has high leverage and low discrepancy. It will not affect the regression line at all because of the low discrepancy. If we want to identify Case B observations, we should use a leverage statistic.

Case C has unusual values of both X and Y. Case C is poorly predicted by a linear regression using just the other observations, and has a unique value of the predictor variable that is far away from the mean. Because it has both high discrepancy and high leverage, this observation is likely to be influential. The regression line may or may not make an accurate prediction for this case,

because if the observation is influential enough, it will pull the regression line toward it, reducing its discrepancy. This is the worst of the three cases, as this type of outlier has a large effect on our conditional distribution story. *Cook's D* is a measure of an observation's influence, that measures how much our predictions change when this observation is removed.

Our happiness outlier seems to be a case A outlier: We have an unusually small value of our dependent variable that comes along with values of our independent variable that we've seen quite frequently. To calculate the jackknife residual outlier statistic, also called studentized residual or studentized deleted residual or externally studentized residual (ugh),¹³ we need to first consider a few other types of residual statistics.

Standardized Residuals

Why should we not just look at the residual for the Case A outlier we've identified and try to determine whether this residual is "large"? The problem is that the size of the residual depends on the scale of our dependent variable, so it's hard to have a consistent measure of size. The *standardized residual* basically does this by normalizing the residuals by the mean-squared error of the regression line:

$$r_i^{Std} = \frac{y_i - \hat{y}_i}{\sqrt{MSE}} \quad (2.17)$$

Dividing by the MSE allows us to tell how many error standard deviations the observation is away from the regression line, where the error standard deviation is estimated by the MSE of the residuals. We can think of r_i^{Std} as being like a typical z-statistic from the normal distribution. This makes some sense if our objective is to determine the *size* of an outlier, and we can see that people want something like this if they are p-value crazy (i.e., they want to say something like "this observation is an outlier $p < .05$ ").

To calculate this, remember our root-MSE from earlier is 0.259. Thus, to find the standardized residuals we compute:

```
mto.data$residuals <- mto.data$happy - predict(mto.reg)
mto.data$std.residuals <- mto.data$residuals/0.259
mto.data[, c("std.residuals", "selfsuff", "happy")]
```

We can see that our evil observation has a standardized residual of 3.86, quite a "large" value. Unfortunately, using some arbitrary "large" categorization creates the opportunity for creative dropping of outliers,

¹³Note that different books, regression packages, and software may call a standardized residual a studentized residual, or vice versa, so be careful to check which one it is actually using.

especially if our regression was almost statistically significant with the outlier, and statistically significant with the outlier removed. If we sort our standardized residuals:

```
sort(abs(mto.data$std.residuals))
```

We can see that we have four observations with absolute standardized residuals greater than 2. Should we drop all four, or some combination of them? There are many different combinations of dropping these four observations that could possibly push our regression to statistical significance. Thus, dropping outliers is a shady practice, regardless of the “size” of the residual.

Studentized Residuals

The standardized residuals are a good start, but don’t work for the following reason. Recall that the residual is:

$$r_i = y_i - \hat{y}_i$$

Now, for the sake of simplicity, assume that $\text{Var}(y_i|x) = \sigma^2$ for all x , meaning there’s just one error variance (i.e., homoskedasticity). However, we are doing our analysis on *residuals*, not the actual errors. Thus, $\text{Var}(r_i) \neq \sigma^2$, because $\text{Var}(r_i) = \text{Var}(y_i - \hat{y}_i)$. Instead ([Behnken and Draper, 1972](#)):

$$\begin{aligned}\text{Var}(r_i) &= \text{Var}(y_i - \hat{y}_i) \\ &= \sigma^2(1 - \text{Leverage}_i)\end{aligned}$$

This shows us that the variance of the residuals depends on the leverage. Thus, the farther away an observation is from the average of the predictor value, the higher the leverage, and the *lower* the variance of the residuals, *even if the true errors are homoskedastic*.

Why is this the case? Well, observations farther away from the mean of the predictors have greater leverage, meaning a unit change in their value affects the slope of the regression line more than observations closer to the mean of the predictor variable. This will tend to pull the regression line toward the observation, reducing the residual.

Thus, if we want to use the “standardization” approach as we did in the case of standardized residuals, we must take into account the fact that observations with high leverage should have a smaller variance in their residuals from sample to sample.

In the multivariate case, leverage is determined by the *hat matrix* H :

$$H = X(X'X)^{-1}X' \tag{2.18}$$

This is called the hat matrix because it puts the “hat” on y , in the following sense:

$$\hat{y} = Hy \quad (2.19)$$

It is easy to see why H is the hat matrix. Recall our estimator of $\hat{\beta}$:

$$\hat{\beta} = (X'X)^{-1}X'y \quad (2.20)$$

For our model $\hat{y} = X\hat{\beta}$, thus:

$$\hat{y} = X\hat{\beta} = X(X'X)^{-1}X'y = Hy$$

The main diagonal of this matrix gives h_{ii} = Leverage _{i} , the leverage for each observation i . The hat matrix is useful for understanding the following relationship between the variance of the fitted values and variance of the residuals:

$$\text{Var}(r_i) = \text{Var}(y_i - \hat{y}_i) = \text{Var}(y_i - Hy_i) = \sigma^2(I - H) \quad (2.22)$$

and

$$\text{Var}(\hat{y}_i) = \text{Var}(Hy_i) = \sigma^2 H \quad (2.23)$$

Both results depend critically on the fact that H is idempotent (i.e., $HH = H$). As can be seen, the variance of the fitted values and the variance of the residuals are inversely related to each other: The variance of the fitted values increase with greater leverage, while the variance of the residuals decrease with increasing leverage.

Studentized residuals (also called the internally studentized residuals or standardized residuals) account for the fact that observations with higher leverage should also be expected to have lower residuals, as they tend to draw the regression line toward them. The studentized residual divides the residuals by the standard error of the residuals, thus fitting the Draper et al. (1966) definition of a t statistic:

$$r_i^{Student} = \frac{y_i - \hat{y}_i}{\sqrt{MSE \times (1 - h_{ii})}} \quad (2.24)$$

We can grab the hat matrix from the handy R function, influence:

```
hat <- influence(mto.reg)$hat
mto.data <- cbind(mto.data, hat)
```

The hat value for observation i is along the main diagonal of H (denoted h_{ii}), and is the leverage of that observation. We can now calculate the studentized residuals as:

```
mto.data$r.studentized <- mto.data$residuals/sqrt(MSE * (1 - mto.data$hat))
sort(abs(mto.data$r.studentized))
```

The residuals get a bit larger when studentized compared to standardized. This will always happen, as $1 - h_{ii}$ is always less than or equal to 1, making the denominator smaller. As h_{ii} gets larger, indicating greater leverage, the denominator gets smaller, meaning the studentized residual is larger than the standardized residual.

Notice, that if the studentized residuals are very different from the standardized residuals, that suggests that there are some peculiarities with the data. In particular, it suggests that some observations have x values (regressor values) that are far away from the rest of the observations (giving them high leverage). Thus, one way to pick out a strange design is to compare the standardized and studentized residuals. [Behnken and Draper \(1972\)](#) provide a stunning example of the importance of leverage in determining the variance of residuals. This example comes from considering a specific degenerate case where all of the observations except one have the same x value. Recall the univariate leverage equation:

$$\text{Leverage}_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (2.25)$$

Suppose we have the following (y, x) pairs as observations: $(1, 3), (2, 3), (3, 3), (4, 3), (5, 5)$. All the x values are the same (3), except for the last observation which has an x value of 5. First, let's calculate the leverage for each observation. We know $\bar{x} = 17/5$. The leverage is the same for the first four observations, because they have the same x value:

$$\text{Leverage}_{1:4} = \frac{1}{5} + \frac{(3 - \frac{17}{5})^2}{4 \times (3 - \frac{17}{5})^2 + (5 - \frac{17}{5})^2} = 0.25 \quad (2.26)$$

On the other hand, the fifth observation has a unique x value:

$$\text{Leverage}_5 = \frac{1}{5} + \frac{(5 - \frac{17}{5})^2}{4 \times (3 - \frac{17}{5})^2 + (5 - \frac{17}{5})^2} = 1 \quad (2.27)$$

Thus, the element $\text{Leverage}_5 = H_{5,5}$ of the hat matrix is 1. Plugging this into $\text{Var}(r_i) = \sigma^2(1 - \text{Leverage}_i)$ we get:

$$\text{Var}(r_5) = \sigma^2(1 - \text{Leverage}_5) = \sigma^2(1 - 1) = 0 \quad (2.28)$$

and

$$\text{Var}(\hat{y}_5) = \text{Var}(\text{Leverage}_5 \times y_i) = \sigma^2 \quad (2.29)$$

Thus, in this degenerate case, the variance of the residuals is always zero, meaning that the regression line always passes through the fifth observation exactly. Another way of saying this is that the residual for the fifth observation is a constant (zero) that does not vary from sample to sample. On the other hand, the fitted value \hat{y}_5 has maximal variance, demonstrating that high leverage points pull the regression line toward them, reducing the residual variance at that point, while increasing the variance of the fitted regression line at that point. Importantly, if we had used the standardized residuals (which doesn't divide by the leverage), we wouldn't have noticed that the residual for one observation is undefined (which would signal a serious problem in our regression).

Jackknife Residual (aka R-Studentized, Deleted Studentized)

There's one more issue with the studentized residual. It uses the MSE of the regression including all of the data. However, the MSE is just an estimate, and observations with high leverage can pull the regression line toward themselves, thus reducing their residual and reducing the estimated MSE. We might want to know what the residual would have been had the potential outlier not influenced the regression line.

We can do this with something called the *Jackknife Residual* or *Externally Studentized Residual*. However, this is usually called the studentized residual, so be careful when reporting and reading papers, as it is likely that these two will be confused. The Jackknife Residual is:

$$r_i^{Jkni} = \frac{y_i - \hat{y}_i}{\sqrt{MSE_{(-i)} \times (1 - h_{ii})}} \quad (2.30)$$

This is the same as the studentized residual, except we use the MSE of the regression *without* the offending observation i . Let's drop the observation with the absolute standardized residual greater than 3:

```
mto.data.drop <- mto.data[abs(mto.data$std.residuals) < 3, ]
```

Calculate the regression without the observation:

```
mto.reg.drop <- lm(happy ~ selfsuff, data = mto.data.drop)
```

Generate our predictions again:

```
predictions <- predict(mto.reg, mto.data)
predictions.drop <- predict(mto.reg.drop, mto.data.drop)
```

Then calculate the residuals:

```
residuals.drop <- mto.data.drop$happy - predictions.drop
```

Then, the mean squared error is the sum of the squared the residuals divided by $n - k$:

```
MSE.drop <- sum(residuals.drop^2) / (nrow(mto.data.drop) - 2)
```

We can see that the MSE without the observation is smaller than the MSE with it, verifying that the observation has a high discrepancy. With this information, we can calculate the jackknife residual:

```
# Grab the residual for observation 110, the potential outlier
resid <- resid(mto.reg)[110]

# Grab the hat value for observation 110
hat.val <- influence(mto.reg)$hat[110]
jackresid <- resid / sqrt(MSE.drop * (1 - hat.val))
```

Verifying this, we can use the R command *rstudent* to automatically calculate the jackknife residual (I recommend you do it by hand a few times to make sure you understand):

```
rstudent(mto.reg)[110] # Get the jackknife residual for observation 110
```

Again, the outlier statistic increases. The MSE with the outlier will almost always be larger than the MSE without it, meaning, the jackknife residuals will always be larger than the studentized, which in turn are necessarily larger than the standardized.

Cook's D

So far we've looked at case A, where we have an unusual value of the dependent variable occurring at a previously seen value of the independent variable. What if we have an unusual value on both variables? That is, what if an observation has both high discrepancy and high leverage? This observation tends to strongly affect the regression line by exerting *influence* on the regression line, pulling the line toward the observation, and for this reason is considered the worst case outlier.

Cook's Distance (or Cook's D) is similar to the Jackknife Residual, in that it recalculates the regression after dropping an observation. Here we are interested in how the predicted values of all observations change when we remove an observation that is suspected of being an outlier. The statistic looks at how much excluding the observation changes the predictions for all observations:

$$D_k^{Cook} = \frac{\sum_{i=1}^n (\hat{y}_i - \hat{y}_{i(k)})^2}{pMSE} \quad (2.31)$$

Here, \hat{y}_i is the prediction for observation i based on all the data, $\hat{y}_{i(k)}$ is the prediction for observation i based on a regression where observation k is removed, p is the number of parameters in the model, and MSE is the mean squared error from the regression with all the data. Cook's D is basically a standardized measure of the change in predictions when an observation is removed.

Let's calculate it. We begin by generating our predictions for each observation based on the full or jackknifed regression model,¹⁴ then calculate the sum of squared differences:

```
predictions <- predict(mto.reg, mto.data)
predictions.drop <- predict(mto.reg.drop, mto.data)
predictions.ssqdiff <- sum((predictions - predictions.drop) ^ 2)
```

The model has two parameters, and we use our MSE from before:

```
D = predictions.ssqdiff/(2 * MSE)
```

We can verify this is correct using the *cooks.distance* function in R, again doing this by hand several times first to make sure we understand:

```
# Get Cook's D for
cooks.distance(mto.reg)[110]
```

The Cook's D is very small. Thus, although the observation is an outlier, it does not change our predictions very much. Instead, it increases the MSE of our regression by about 10% (MSE/MSE.drop).

Here's a handy plot from the car package that shows us information about potential outliers all on a single plot, including Cook's Distance, the jackknife residuals (called studentized residuals in the plot), hat values, and a p-value:

```
png(filename = "influenceindex.png", width = 3000, height = 3000, res = 300)
# Influence index plot and identify the largest three observations
influenceIndexPlot(mto.reg, id.n = 3)
dev.off()
```

¹⁴Jackknifing generally means that we remove a single observation.

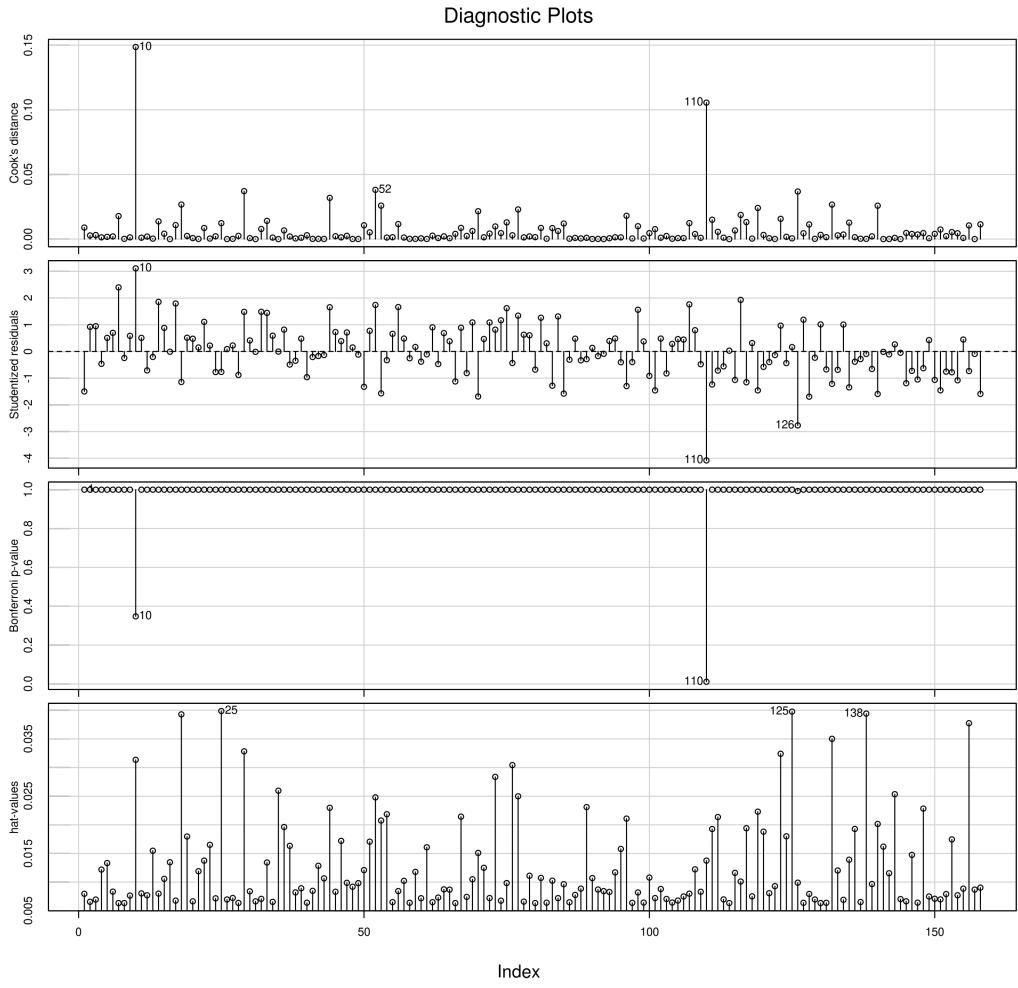


Figure 2.27: Influence index plot showing Cook's D values (top), jackknife residuals, bonferonni adjusted p-values, and hat values (bottom). To some degree the Cook's distance can be seen as the product of the hat value (leverage) and jackknife residual (discrepancy).

Notice that observations with high influence (large Cook's D) must have both high discrepancy and high leverage, showing us that:

$$\text{Influence} \propto \text{Discrepancy} \times \text{Leverage}$$

What to do with outliers

As we remember from our fifth circle of scientific hell, there is a lot of creative opportunities for us to include or exclude outliers based on different heuristic criteria. We could used a standardized residual > 3 , a studentized residual

> 3.7 , a Cook's $D > 4p/n$. I really don't see how one can make a compelling argument for dropping outliers using these rules, especially given the temptation to not report it.

A common approach is to present results both with and without the outlier, showing the audience how much the results depend on a single observation. Another creative approach would be to summarize how much the estimated model parameters change after deleting each observation individually (like Cook's D for parameters).

Here are a few additional thoughts for coping with outliers. As general good practice, the first thing to do is check the outlier against other data sources to see if it is a data entry error. Be careful to *document this checking procedure*. If you think there is a data entry error, and can justify that with some form of evidence, then provide the justification, document the evidence, and fix the error with reproducible code. Be sure to create a *new* variable with the adjusted data. Do not delete or overwrite the observation that is being fixed, as this can eliminate the paper trail for others who want to reproduce what you've done.

A second great option is to provide a better model of the data, for example using a conditional distribution with heavy tails (e.g., the t distribution) that allows for such observations. This could be a distributional model, or a data transformation. Often, however, there are no suitable distributions that can capture outliers.

A third option is to think carefully about the outlier and see if it can be modeled, not as a different distribution of the dependent variable, but as a function of an independent variable. Finding an explanation for the outlying observation by adding a regressor to the model can be a valuable thought experiment, but this runs the risk of building a model that is too complex. We can trivially just create an indicator variable for the outlying observation and consider it modeled, but obviously that indicator variable would have no meaning if we were trying to predict future data.

A fourth option is to use an alternative regression method that is less sensitive to outliers. Quantile regression and median regression are two popular alternatives, and are often called "robust" regression. This will be particularly valuable for outliers with large Cook's D (case C outliers), but less so with only large jackknife residuals but small Cook's D (case A outliers; what we have).

```
install.packages("quantreg")
library(quantreg)
qreg1 <- rq(happy ~ selfsuff, data = mto.data, tau = 0.5)
png(filename = "medianreg.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(mto.data$selfsuff, mto.data$happy,
```

```
ylab = "Mean Happiness Z-Score",
xlab = "Mean Economic Self-Sufficiency Z-Score",
xlim = c(-1, 1),
ylim = c(-1, 1),
pch  = 19,
col  = rgb(0, 0, 0, .5))
abline(qreg1, col = "red")
abline(mto.reg, col = "blue")
dev.off()
```

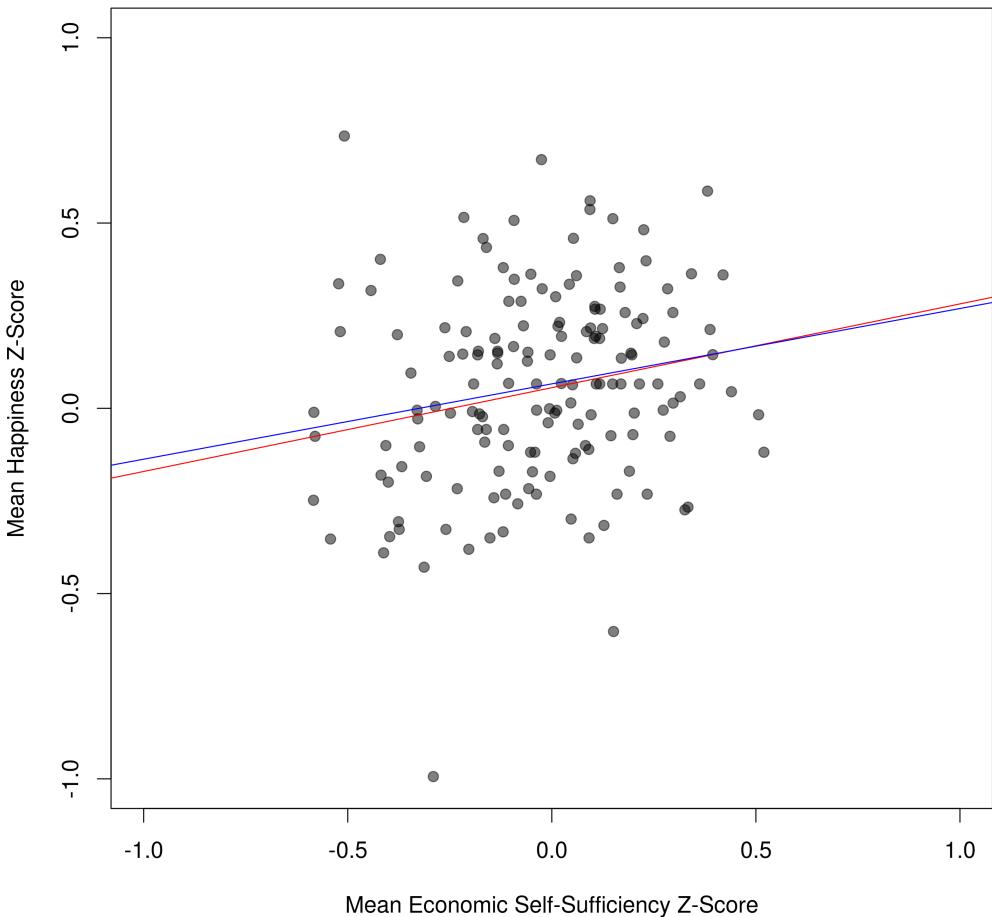


Figure 2.28: Quantile (median) regression (red) and linear least squares (mean) regression (blue).

Here `rq` does a quantile regression estimating the quantiles in τ . Since τ is 0.5, the quantile regression just estimates the conditional median rather than conditional mean. As expected, there is very little difference between the median regression and linear least squares (mean regression), as the outlier as case A. If the outlier was case C, we should expect a substantial difference between the two regressions.

Keep in mind that just because something is “robust” to outliers doesn’t mean that using the method improves our substantive understanding of the topic. Robust methods just circumvent the problem by caring less about it.

A final alternative is to use a bootstrap to see how sensitive the results of the new model are. Likewise, bootstrap might be good for just summarizing

the uncertainty without making any specific assumptions about the distribution. We'll return to bootstrapping later.

Overall, if we can't come up with a good conditional distribution story because of an outlier, we need to admit that. We just don't understand the distribution from which the data are drawn.

2.4.4 Why Use Graphics For The Conditional Distribution Story?

Here's a classic fake data example that demonstrates why the data summary and conditional distribution stories are so important, and why you cannot adequately create these stories without looking at the data. This example is sometimes called Anscombe's Quartet ([Anscombe, 1973](#)). Here's the data file:

```
anscombe <- read.table("AnscombeData.txt",
                        col.names = c("set", "y", "x"))
```

Looking at the data summaries:

```
set1 <- anscombe[anscombe$set == 1,]
set2 <- anscombe[anscombe$set == 2,]
set3 <- anscombe[anscombe$set == 3,]
set4 <- anscombe[anscombe$set == 4,]
summary(set1)
summary(set2)
summary(set3)
summary(set4)
```

The y averages for each of the four datasets are exactly equal at 7.50, but they have different medians and distributions. A similar story holds for the x values. It's hard to understand how they differ from the descriptives alone, so let's plot them:

```
png(filename = "yhist.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 2), cex = 1.3, mar = c(4, 3, 1, 0), oma = c(0, 0, 0, 0))
hist(set1$y,
     xlim = c(2, 14),
     ylim = c(0, 10),
     main = "Set 1",
     breaks = "FD",
     xlab = "")
hist(set2$y,
     xlim = c(2, 14),
```

```
    ylim   = c(0, 10),
    main   = "Set 2",
    breaks = "FD",
    xlab   = "")  
hist(set3$y,
    xlim   = c(2, 14),
    ylim   = c(0, 10),
    main   = "Set 3",
    breaks = "FD",
    xlab   = "")  
hist(set4$y,
    xlim   = c(2, 14),
    ylim   = c(0, 10),
    main   = "Set 4",
    breaks = "FD",
    xlab   = "")  
dev.off()
```

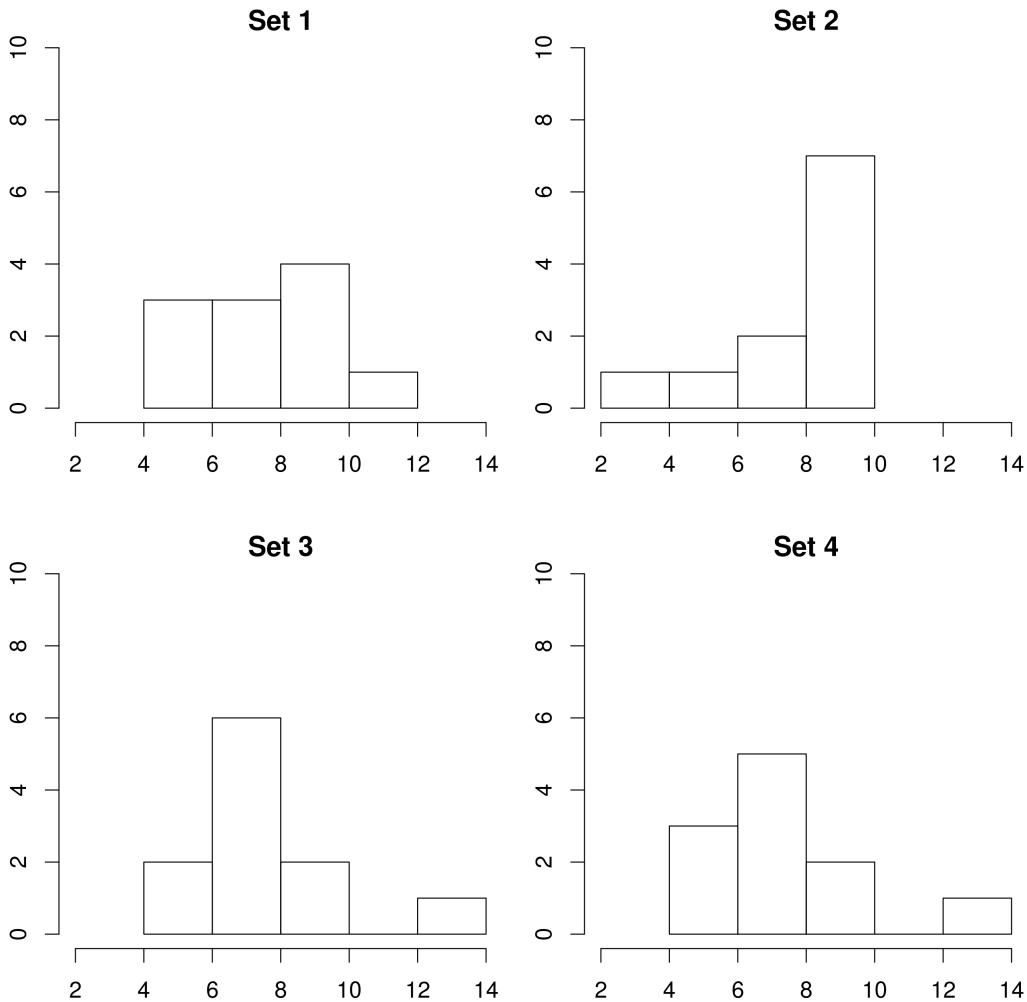


Figure 2.29: Histograms of y for the four Anscombe datasets.

Looking at the histograms, the y values for Set 1 seem to be relatively compact, with bumps around 7 and 8. Set 2 is more skewed, with higher density between 8 and 10. Set 3 has many observations around its mean, then a single observation at 13, with a similar pattern for Set 4. The different sets have essentially the same mean and standard deviations but very different distributions. Let's do the same thing for the x-values:

```
png(filename = "xhist.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 2), cex = 1.3, mar = c(4, 3, 1, 0), oma = c(0, 0, 0, 0))
hist(set1$x,
      xlim = c(0, 20),
      ylim = c(0, 10),
      main = "Set 1",
```

```
breaks = "FD",
xlab   = "")  
hist(set2$x,
    xlim  = c(0, 20),
    ylim  = c(0, 10),
    main   = "Set 2",
    breaks = "FD",
    xlab   = "")  
hist(set3$x,
    xlim  = c(0, 20),
    ylim  = c(0, 10),
    main   = "Set 3",
    breaks = "FD",
    xlab   = "")  
hist(set4$x,
    xlim  = c(0, 20),
    ylim  = c(0, 10),
    main   = "Set 4",
    breaks = "Sturges",
    xlab   = "")  
dev.off()
```

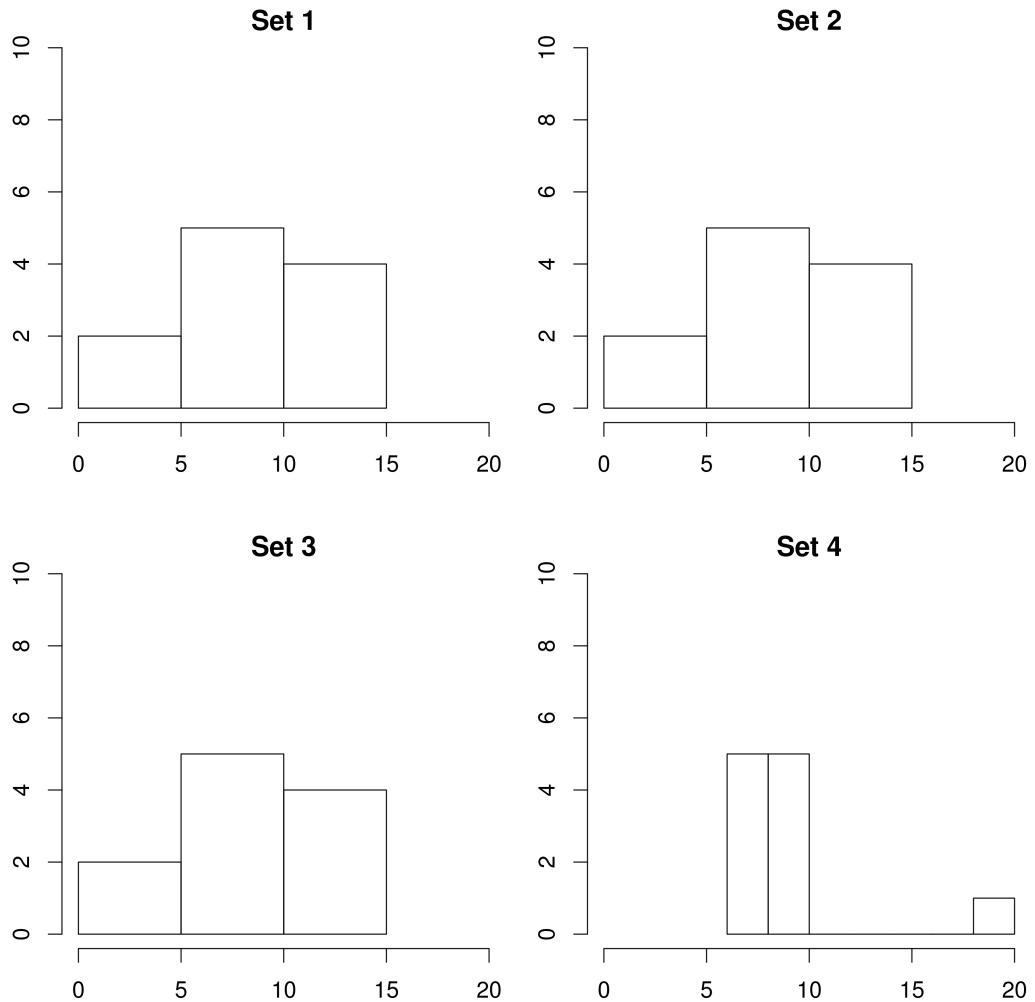


Figure 2.30: Histograms of x for the four Anscombe datasets.

The x -values are roughly uniformly distributed across the range of observed values, except for set 4, that is heavily concentrated on one value, with one observation far away from the others. We can expect that this observation will have high leverage. Now lets compare the regression results of the four different datasets:

```
reg1 <- lm(y ~ x, data = set1)
reg2 <- lm(y ~ x, data = set2)
reg3 <- lm(y ~ x, data = set3)
reg4 <- lm(y ~ x, data = set4)
summary(reg1)
summary(reg2)
summary(reg3)
```

```
summary(reg4)
```

The intercept, slope, standard errors, t-values, and p-values are *identical* for the four regressions. If we had the blinders on and just looked at the coefficients and p-values or a regression table, these regressions would appear identical. Let's look at scatterplots for each dataset:

```
png(filename = "setscatters.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 2), cex = 1.3, oma = c(0, 0, 0, 0), mar = c(4, 4, 2, 1))
plot(set1$x, set1$y,
     ylab = "Y",
     xlab = "X",
     xlim = c(4, 20),
     ylim = c(2, 14),
     pch = 19,
     col = rgb(0, 0, 0, .5),
     main = "Set 1")
abline(reg1)
plot(set2$x, set2$y,
     ylab = "Y",
     xlab = "X",
     xlim = c(4, 20),
     ylim = c(2, 14),
     pch = 19,
     col = rgb(0, 0, 0, .5),
     main = "Set 2")
abline(reg2)
plot(set3$x, set3$y,
     ylab = "Y",
     xlab = "X",
     xlim = c(4, 20),
     ylim = c(2, 14),
     pch = 19,
     col = rgb(0, 0, 0, .5),
     main = "Set 3")
abline(reg3)
plot(set4$x, set4$y,
     ylab = "Y",
     xlab = "X",
     xlim = c(4, 20),
     ylim = c(2, 14),
     pch = 19,
     col = rgb(0, 0, 0, .5),
```

```
main = "Set 4")
abline(reg4)
dev.off()
```

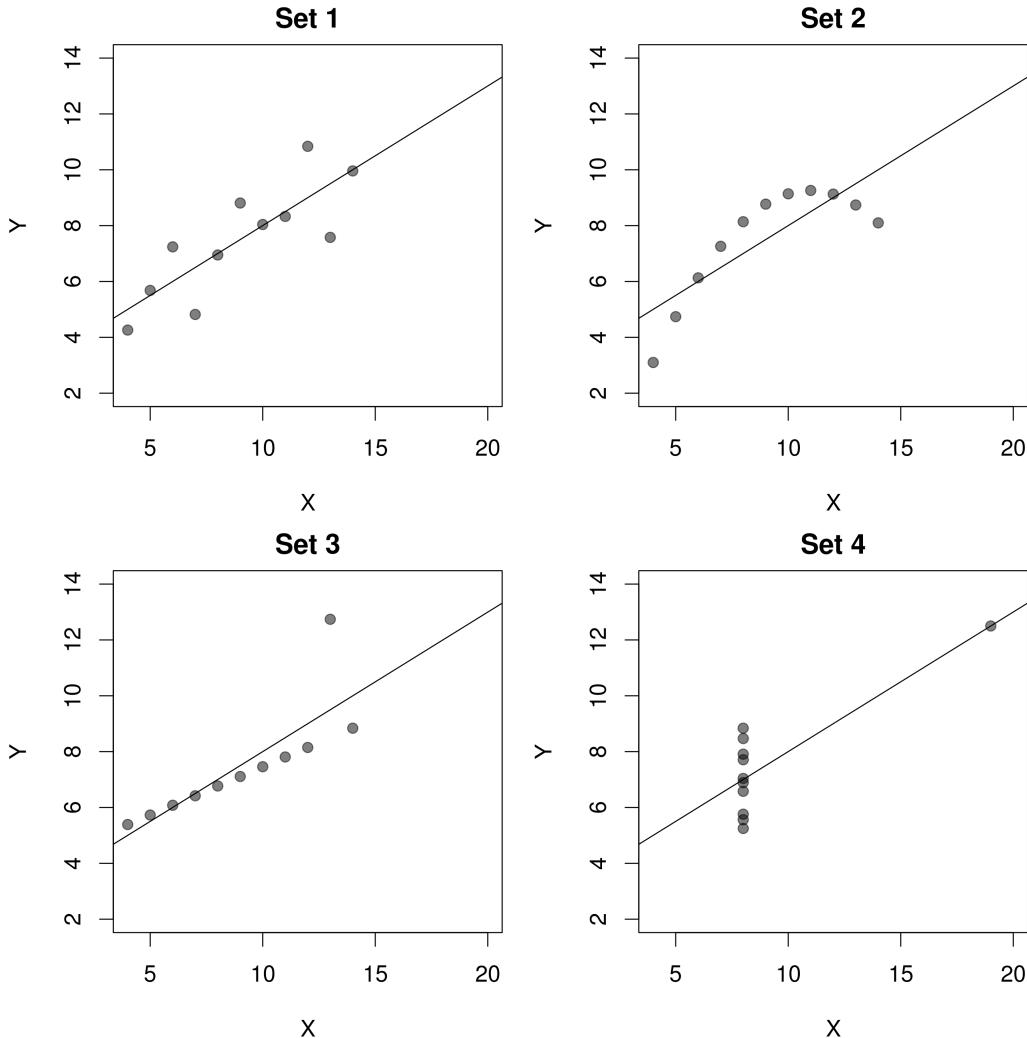


Figure 2.31: Scatterplot and regression lines for the four Anscombe datasets.

From the scatterplots, we can see that only in Set 1 does the linear regression seem to make sense. Set 2 looks quadratic. Set 3 seems to have an outlier that is a unique y value, with high discrepancy but low leverage (Case A from before). Set 4 has an outlier that is both a new x and y value (Case C from before), indicating the observation likely has both high leverage and discrepancy, thus high influence.

Let's look at Jackknife residuals and Cook's D for the four datasets. We could write a function to calculate the Jackknife residuals, but now that we understand what is being done, we can use some functions for regression diagnostics:

```
rstudent(reg1)
rstudent(reg2)
```

```

rstudent(reg3)
rstudent(reg4)
cooks.distance(reg1)
cooks.distance(reg2)
cooks.distance(reg3)
cooks.distance(reg4)

```

For our regression on Set 1, the jackknife residuals are relatively small (all less than 1.84), as well as Cook's D (all less than 0.49). For our regression on Set 2, the two observations have slightly larger jackknife residuals (observations 17 and 19). The same observations have large(ish) Cook's D. Without the scatterplot showing the quadratic relationship it would be hard to make sense of this, although they seem to be Case C outliers. We know from the scatterplot the problem is that the functional form of the relationship between X and Y is not correctly specified. One way to look for functional form misspecification more carefully is to look at a plot of the residuals versus the fitted (predicted) values:

```

png(filename = "reg2diag.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 1), cex = 1.3, mar = c(4, 4, 2, 0))
plot(fitted(reg2), resid(reg2),
      ylab = "Y",
      xlab = "X",
      ylim = c(-5, 10),
      xlim = c(-5, 10),
      pch = 19,
      col = rgb(0, 0, 0, .5),
      main = "Set 2")
# Plot the fitted values on the x axis using the fitted function
# Plot the jackknife residuals on the y axis
plot(fitted(reg2), rstudent(reg2),
      ylab = "Jackknife Residuals",
      xlab = "Fitted Values",
      pch = 19,
      col = rgb(0, 0, 0, .5))
abline(h = 0)
dev.off()

```

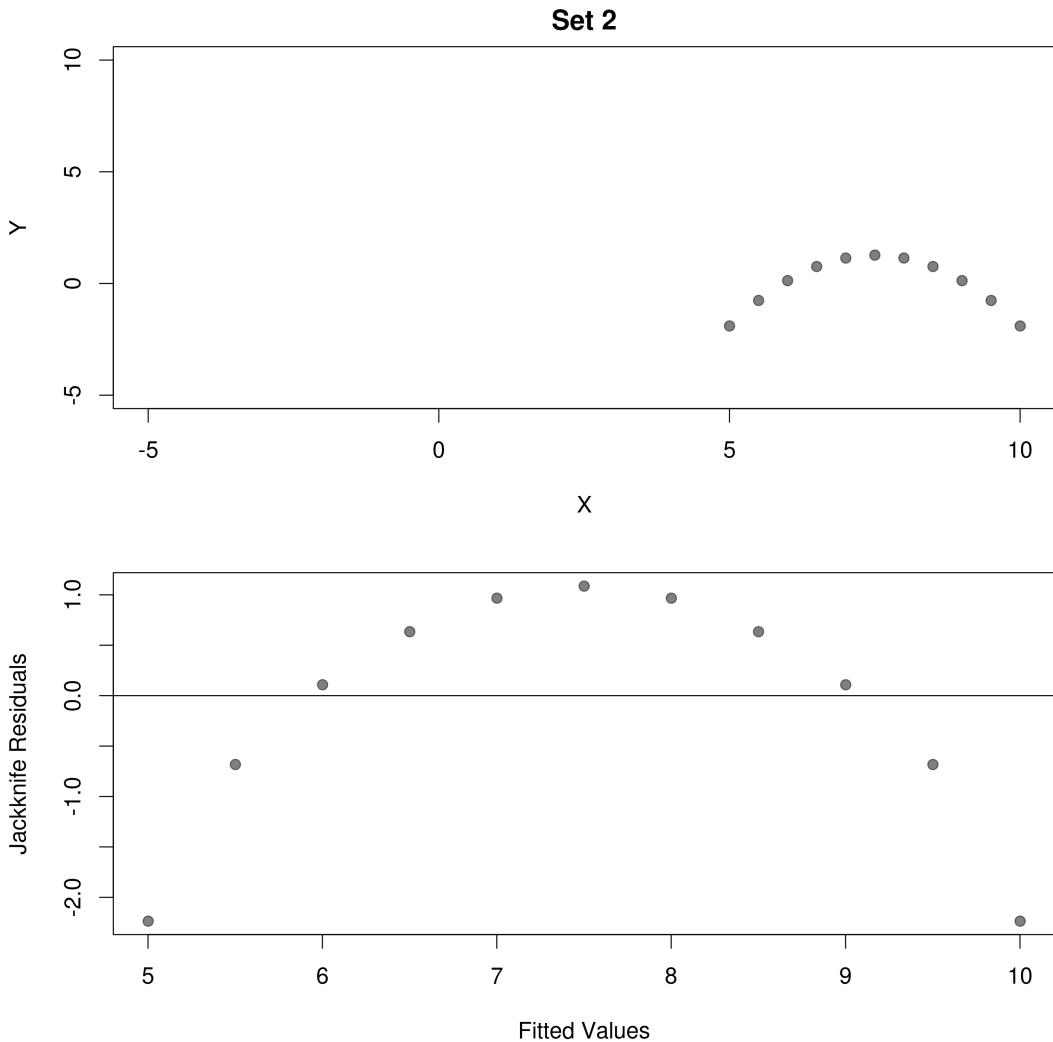


Figure 2.32: Regression diagnostics for Anscombe dataset 2, that has a quadratic form but the regression only has a linear term.

If our regression is correct, then the residuals should be uncorrelated with the fitted values, meaning the average jackknife residual should be zero for all fitted (predicted) values of y . The top plot shows that our fitted values, or predictions based on the regression line, are first too high, then too low, then too high again. The bottom plot shows the residuals, which are first below zero, then above, then below. Thus, if we get the functional form of our regression wrong, the residuals will show a pattern when plotted against the fitted values. The fitted values should also have greater variability than the residuals, indicating that the explained variation is more than the error variation.

Next, our regression on Set 3, the data look good except observation 25 has an enormous jackknife residual of 1203, with a large Cook's distance of 1.39. Plotting the diagnostics again:

```
png(filename = "reg3diagnostics.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 1), cex = 1.3, mar = c(4, 4, 2, 0))
plot(fitted(reg3), rstudent(reg3),
      pch = 19,
      col = rgb(0, 0, 0, .5),
      xlab = "Fitted Values",
      ylab = "Jackknife Residuals")
plot(reg3, which = 4) # Plot 4 shows Cook's D by index
dev.off()
```

The plots show that the residuals are tiny except for a huge miss on observation 25.

Lastly, for our regression on Set 4, the jackknife residuals seem relatively small, and are zero on average, indicating we haven't misspecified the functional form. On the other hand observation 41 has a huge Cook's D. From the plot we know that we would essentially have a flat line if it weren't for observation 41, so the sum of squared changes in predictions (Cook's D) will be large.

```
png(filename = "reg4diagnostics.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 1), cex = 1.3, mar = c(4, 4, 2, 0))
plot(fitted(reg4), rstudent(reg4),
      xlab = "Fitted Values",
      ylab = "Jackknife Residuals",
      pch = 19,
      col = rgb(0, 0, 0, .5))
abline(h = 0)
plot(reg4, which = 4)
dev.off()
```

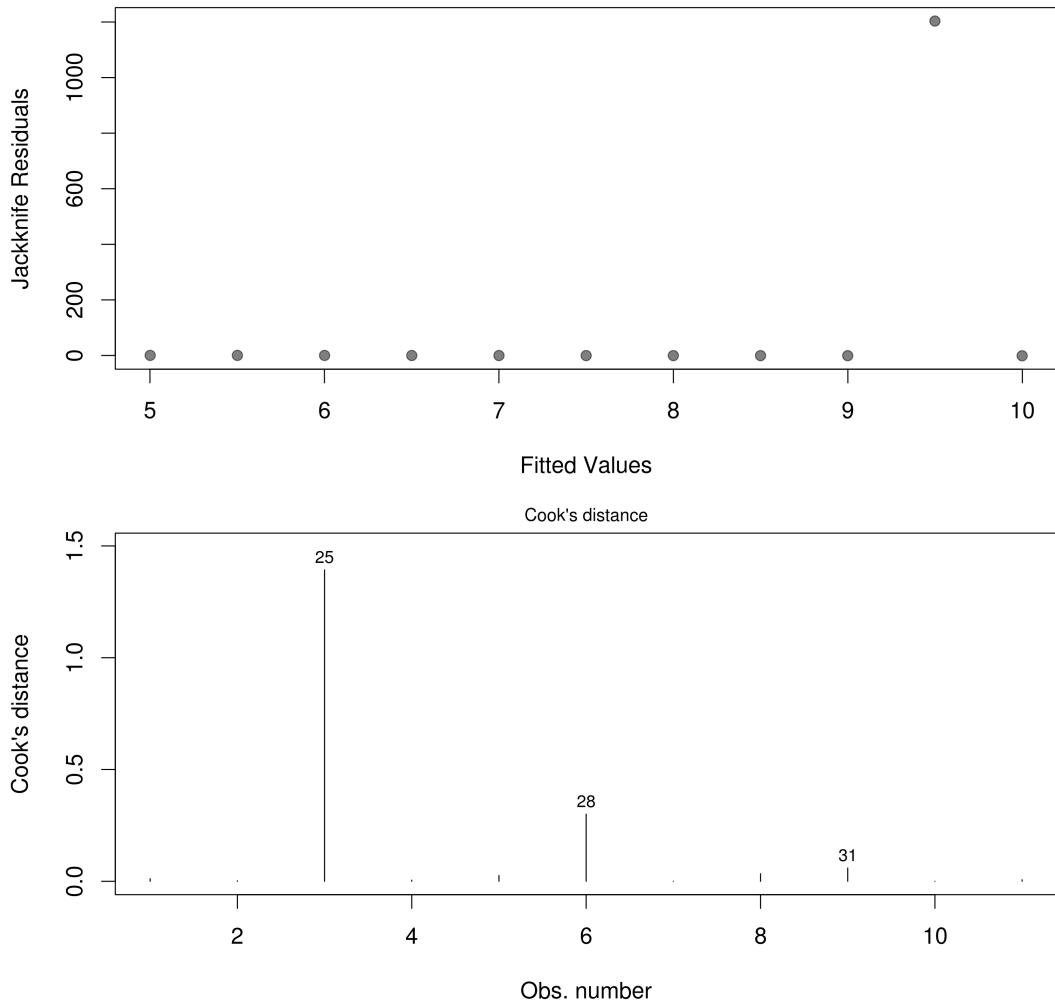


Figure 2.33: Regression diagnostics for Anscombe dataset 3, with massive Case A outlier.

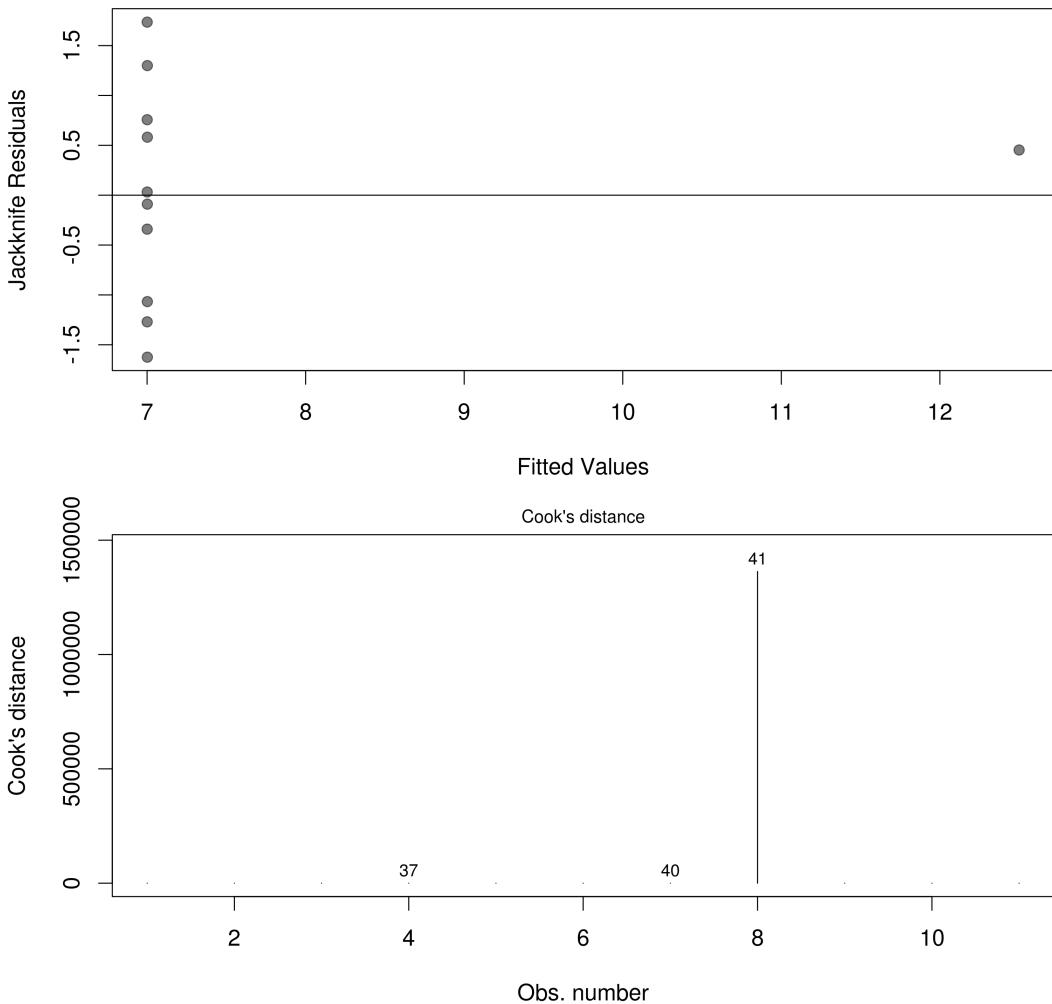


Figure 2.34: Regression diagnostics for Anscombe dataset 4, with massive Case C outlier.

So we can see how misleading simple summary statistics, such as the mean and standard deviation, and regression outputs can be. Looking at the data directly, through histograms, scatterplots, and regression diagnostic plots (fitted vs. residuals, Cook's D), can help us spot really bad problems with the regression model.

2.5 The Forecasting Story

After thinking carefully about the conditional distributions we think generate the data, we often have different models. We may add a predictor, do a transformation, etc. Modeling conditional distributions this way is good

because it gets us to think about the data we actually have, rather than the data that fit a particular statistical approach we want to use. However, this approach tends to produce models that are progressively more complex, and the more complex they are, the more we risk explaining randomness.

After finishing creating the conditional distribution story, the best method for testing the validity of our model is to make predictions about future data from the same process that generated our old data. That is, we make a forecast. The key element of forecasts that “test” the validity of our model is that we cannot adjust our model to explain the future data, because we don’t have the data yet. Thus, forecasting is in all senses a scientific test of our model.

Unfortunately, we often do not have the time, patience, and money to collect new data to test our model against. Fortunately, there is another option, called *cross-validation*.¹⁵ The basic idea behind cross-validation is that our best guess at what new data would look like is the data we have on hand. This logically follows if the data we have are a random sample from the population of data we are trying to understand. If this is the case, then splitting our data into two parts, *training data* and *test data*, can allow us to simulate the forecasting validation test. The training data are used to fit the model, then the model is tested on the test data. By repeating this process a bunch of times, we can get an idea about the size of the model’s error on new data. We can then compare a simpler model to our more complex one to see if we are capturing useful features of the conditional distributions, or just creating complex models of noise.

2.5.1 Backcasting

Before we forecast, we should first take a look at how well we are able to predict the data we have, using all the data. We are looking backward at how our model predicts data it has already seen. As we’ve done previously, we use the entire data to train the model, with the entire data being the test set:

```
png(filename = "backcast.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
# Plot the predicted happiness scores versus the observed happiness scores
plot(predict(mto.reg,mto.data), mto.data$happy,
      xlim = c(-1, 1),
      ylim = c(-1, 1),
      ylab = "Actual Happiness z-score",
      xlab = "Predicted Happiness z-score",
```

¹⁵Shalizi’s Chapter 3 notes provides a more formal treatment: <http://www.stat.cmu.edu/~cshalizi/uADA/13/lectures/ch03.pdf>

```

pch  = 19,
col  = rgb(0, 0, 0, .3))
abline(0, 1)
dev.off()

```

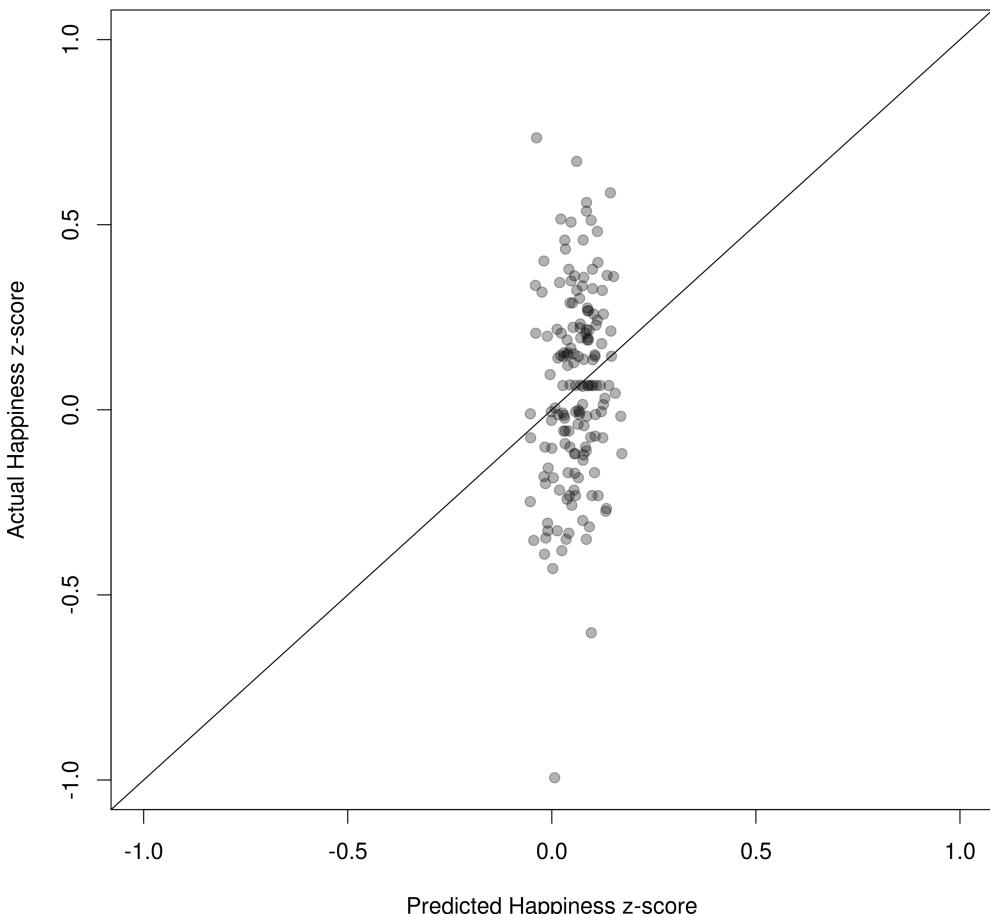


Figure 2.35: In-sample predictions from linear regression. The actual happiness scores are on the y axis, with predicted happiness scores on the x axis.

We can see that the actual dependent variable is distributed almost all the way from -1 to 1. The predictions, on the other hand, are closely packed around the mean. If the predictions were perfectly correlated with the actual data, all points would fall along the diagonal. Thus, the variation in the predictions is much lower than the total variation in the dependent variable (another way of saying this is that the model has a low R^2). The reason for

such tightly packed predictions is that economic self-sufficiency only ranges from -0.584 to 0.520 and is not strongly correlated with happiness. Thus, the most extreme predictions will be from -0.053 to 0.172. This checks out, as the range of predictions don't vary much from the average of the dependent variable:

```
range(predict(mto.reg, mto.data)) # examine the range of predictions
```

Thus, even though economic self-sufficiency captures some information about the conditional mean of happiness scores, looking at the predictions we know that we can't really learn much from such a model, which doesn't make predictions that are much different from the overall mean happiness z-score.

2.5.2 Cross-Validation

The previous analyses raise the question of whether economic self-sufficiency tells us anything at all about happiness. As mentioned before, the best way to do this is to test models with and without economic self-sufficiency against each other, using cross-validation.¹⁶ Cross-validation is the industry standard for model testing and selection, and there is evidence that other measures (e.g., Mallow's C_p , adjused R^2 , etc.) are highly biased (Breiman and Spector, 1992; Breiman, 1992).

First, let's split our data into training and test sets. Here we will use two-thirds training, one-third test for ease of coding, but 5 or 10 splits are the usual number. To do this, we randomly select from the cell ids of our data frame:

```
# Sample 2/3 of the cell ids from the mto.data without replacement
# This is our "training" data
set.seed(10)
train <- sample(mto.data$cell_id, # Sample the cell ids
                2*length(mto.data$cell_id)/3, # Sample 2/3 of the ids
                replace = FALSE) # Sample without replacement
# Create "test" data that are the cell ids not selected
test <- mto.data$cell_id[ - train]
```

Next, we run two regressions on the training data, one with and one without economic self-sufficiency:

```
# Run linear regression.
# Subset the data to only select rows where the cell ids
```

¹⁶For a dissenting viewpoint: <http://andrewgelman.com/2012/06/27/cross-validation-what-is-it-good-for/>.

```

# are "in" (%in%) the training data
train1 <- lm(happy ~ selfsuff,
             data = mto.data[mto.data$cell_id %in% train, ])
train2 <- lm(happy ~ 1, # Include only an intercept
             data = mto.data[mto.data$cell_id %in% train, ])

```

Calculate the squared residuals for the two regressions on the test data:

```

# The actual outcome is on the first line
# The regression predictions are on the second line
# Calculate the squared difference for all predictions on the third line
test1 <- (mto.data$happy[mto.data$cell_id %in% test] -
           predict(train1, mto.data[mto.data$cell_id %in% test, ]))^2

test2 <- (mto.data$happy[mto.data$cell_id %in% test] -
           predict(train2, mto.data[mto.data$cell_id %in% test, ]))^2

```

Then, we calculate the rMSE for the two regressions on the *test* data:

```

rMSEtest1 <- sqrt(sum(test1) / length(test1))
rMSEtest2 <- sqrt(sum(test2) / length(test2))

```

It is important to notice the following. The rMSE of the two regressions on the training data (the backcast) shows that the more complex model, with self-sufficiency as a predictor, is a better model with rMSE of 0.259, whereas the rMSE of the simpler model (with only an intercept) is 0.270:

```

summary(train1)$sigma # Get rMSE from the backcast
summary(train2)$sigma

```

But, when we test our models on their ability to forecast new data, the reverse is true; the rMSE of the simpler model, without economic self-sufficiency, is 0.248, whereas the rMSE of the more complex model is 0.269. In fact, the rMSE of the more complex model is greater than the standard deviation of the dependent variable (0.263), meaning including economic self-sufficiency in our model has made predictions *worse* than merely predicting the average happiness z-score:

```
sd(mto.data$happy)
```

Recall that we made a random split of our data into training and test sets. This random split could have merely been a bad split for our more complex model. To make sure this isn't an anomaly, we can repeat the cross-validation procedure a bunch of times (1000), selecting random splits of the data each time:

```

complex <- c() # Create an empty vector
simple <- c()
set.seed(11)

for(i in 1:1000){ # Loop i from 1 to 1000

  train <- sample(mto.data$cell_id,
    2*length(mto.data$cell_id)/3,
    replace = FALSE)
  test <- mto.data$cell_id[ - train]
  train1 <- lm(happy ~ selfsuff,
    data = mto.data[mto.data$cell_id %in% train, ])
  train2 <- lm(happy ~ 1,
    data = mto.data[mto.data$cell_id %in% train, ])

  test1 <- (mto.data$happy[mto.data$cell_id %in% test] -
  predict(train1, mto.data[mto.data$cell_id %in% test, ]))^2

  test2 <- (mto.data$happy[mto.data$cell_id %in% test] -
  predict(train2,mto.data[mto.data$cell_id %in% test, ]))^2

  rMSEtest1 <- sqrt(sum(test1)/length(test1))
  rMSEtest2 <- sqrt(sum(test2)/length(test2))
  # Append the rMSE from this iteration to vectors
  complex <- append(complex, rMSEtest1)
  simple <- append(simple, rMSEtest2)
}

```

The code takes a while to run, and can be optimized to increase the speed, but we aren't really concerned about optimal coding right now. Looking at the results:

```

summary(complex)
summary(simple)

```

The median test rMSE is almost identical for the two models, with a lower mean for the more complex model that includes economic self-sufficiency. Thus, even though there seems to be some (although weak) relationship between economic self-sufficiency and happiness in these data, including this variable in the model isn't hurting our predictive power much, but also isn't helping. With few observations this will often be the case.

Let's try this one more time for a model that we are pretty sure will be bad: Include a fourth-degree polynomial function of economic self-sufficiency:

```

complex <- c()
simple <- c()
set.seed(12)

for(i in 1:1000){

  train<-sample(mto.data$cell_id,
  2*length(mto.data$cell_id)/3,
  replace = FALSE)
  test <- mto.data$cell_id[ - train]
  train1 <- lm(happy ~ selfsuff
  # We need to include I() for R to correctly interpret the powers
  + I(selfsuff^2)
  + I(selfsuff^3)
  + I(selfsuff^4),
  data = mto.data[mto.data$cell_id %in% train, ])
  train2 <- lm(happy ~ 1,
  data = mto.data[mto.data$cell_id %in% train, ])

  test1 <- (mto.data$happy[mto.data$cell_id %in% test] -
  predict(train1, mto.data[mto.data$cell_id %in% test, ]))^2

  test2 <- (mto.data$happy[mto.data$cell_id %in% test] -
  predict(train2, mto.data[mto.data$cell_id %in% test, ]))^2

  rMSEtest1 <- sqrt(sum(test1)/length(test1))
  rMSEtest2 <- sqrt(sum(test2)/length(test2))
  complex <- append(complex, rMSEtest1)
  simple <- append(simple, rMSEtest2)
}

```

Now the complex model is worse than the simple one, with rMSE of 0.268 (complex model with fourth degree polynomial) compared to 0.264 (simple model with only an intercept). Thus, using the more complex model makes little sense, unless we had some very strong prior theory that a fourth degree polynomial was likely. This demonstrates that cross-validation can help us understand the relative predictive performance of different models, and is a widely accepted and general method of model comparison.

Unfortunately, the choice of number of splits of the data into training and test sets (e.g., splitting the data into 2 parts, 3 parts, or 10 parts) is somewhat arbitrary. As you would expect, splitting a small dataset in half will yield an overestimate of a model's prediction error (see [Hastie et al. \(2004\)](#) chapter 7.10), suggesting the number of splits should be larger for

datasets with fewer observations. Leave-one-out cross-validation (dropping each observation individually and forecasting it from the rest of the data) can provide an unbiased estimate of prediction error but tends to be noisy (has high variance across different samples). Thus, with small datasets, it is important to examine the sensitivity of cross-validated error estimates parametrically (e.g., varying the number of splits from 2 to 20) to see if the conclusions change. Five or 10 splits are conventionally seen as an appropriate starting place. Here's an example using 5-fold cross-validation:¹⁷

```
# Compare predictive ability using five-fold CV
nfolds <- 5
# divide the cases as evenly as possible
case.folds <- rep(1:nfolds, length.out = nrow(mto.data))
# randomly permute the order
set.seed(42)
case.folds <- sample(case.folds)

complex <- c() # Create empty vectors
simple <- c()
for (fold in 1:nfolds) {
  # Create training and test cases
  train <- mto.data[case.folds != fold, ]
  test <- mto.data[case.folds == fold, ]
  # Run the simple and complex models
  simple.train <- lm(happy ~ 1, data = train)
  complex.train <- lm(happy ~ selfstuff, data = train)
  # Generate test MSEs
  simple.test <- (test$happy - predict(simple.train, test))^2
  complex.test <- (test$happy - predict(complex.train, test))^2
  # Calculate rMSEs
  rMSEtest1 <- sqrt(sum(simple.test)/length(simple.test))
  rMSEtest2 <- sqrt(sum(complex.test)/length(complex.test))
  # Append the rMSE from this iteration to vectors
  simple <- append(simple, rMSEtest1)
  complex <- append(complex, rMSEtest2)
}
# Average the MSEs
simple.avg <- mean(simple)
complex.avg <- mean(complex)
```

Here the 5-fold cross-validation errors favor the more complex model, so

¹⁷Code is based on this assignment <http://www.stat.cmu.edu/~cshalizi/402/hw/02/hw-02.R>

the conclusions seem sensitive to the number of folds. It would be wise to repeat the process varying the number of folds parametrically to test the sensitivity of the conclusions to the choice of fold number. We will cheat a bit and use a package to do this with just 1 replicate per k-fold, but if it were a serious project you'd want to do multiple replicates:

```
#Use the cvTools package
install.packages("cvTools", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(cvTools)

# Conduct base regressions
simple.reg <- lm(happy ~ 1, data = mto.data)
complex.reg <- lm(happy ~ selfsuff, data = mto.data)

# Set the sequence of folds to try
k <- c(seq(from = 2, to = 20, by = 2), nrow(mto.data))

# Allocate empty vectors to store results
simple.result <- c()
complex.result <- c()

for(folds in k){
  #Cross-validate simple model
  simple.cv <- cvFit(simple.reg, data = mto.data, y = mto.data$happy,
                      K = folds, # vary the number of folds
                      R = 1) # number of replicates per k-fold
  #Cross-validate complex model
  complex.cv <- cvFit(complex.reg, data = mto.data, y = mto.data$happy,
                       K = folds,
                       R = 1)

  simple.result <- append(simple.result, simple.cv$cv)
  complex.result <- append(complex.result, complex.cv$cv)
}

png(filename = "foldsensitivity.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(k, simple.result, col = "blue", type = "b",
      ylim = c(min(complex.result, simple.result),
              max(complex.result, simple.result)),
      ylab = "rMSE",
      xlab = "Folds (k)")
points(k, complex.result, col = "red")
```

```
lines(k, complex.result, col = "red")

# Add legend to plot
legend(100, .265, # legend location in x-y coordinates
       c("Simple Model", "Complex Model"), # legend labels in order
       col = c("blue", "red"), # colors for legend labels in order
       pch = 19)
dev.off()
```

As you can see, the complex model performs better but the results are noisy, so it would be wise to include additional replicates. However, overall it seems like the forecasting quality for the complex model is better than the simple model.

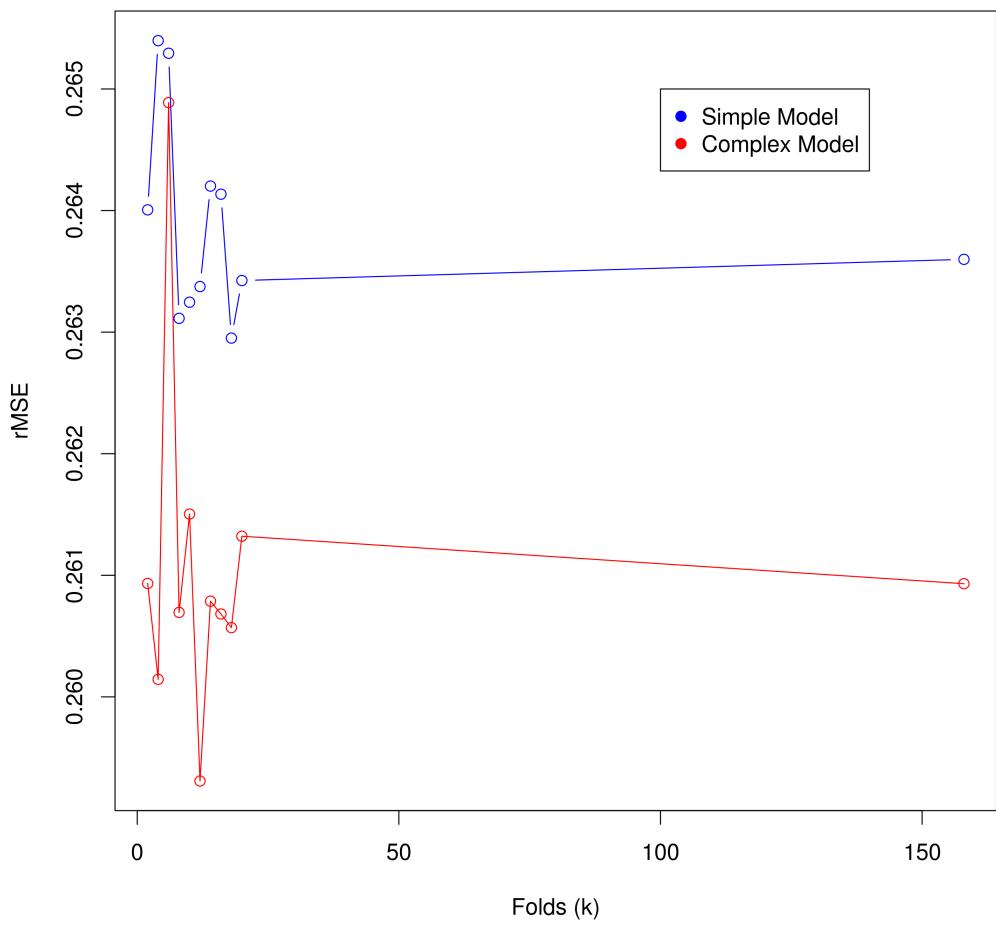


Figure 2.36: Variation in cross-validated rMSE of the simple and complex models as a function of the number of folds.

2.6 The Statistical Inference Story

Statistical inference means going beyond the data we have and saying something about how the sample, and statistics calculated from our sample, relate to the population and population statistics. For example, we might take a sample of people who respond to a craigslist advertisement about a survey on climate change, then record the percent that agree with the statement “climate change is real,” then try to make a statement about the percent of people in the U.S. population who agree that climate change is real. Sometimes if $p < .05$ on some hypothesis test it is concluded that a relationship is real, or that we can rule out some value of a population parameter.

In fact, generalizations from sample to population are almost completely unrelated to p-values, or any feature of the data collected. Instead, these generalizations rely on information external to the data, most importantly, information about *how* the data were collected. This information comes in the form of statements about a data generating process that operates in the real world. Some important statements are answers to the following questions ([Berk and Freedman, 2003](#)):

1. How did we get the data? Is it a random sample?
2. Does a population exist that we could plausibly sample from?

The answers to each question are statements about the real world, and thus are “subject to the conventional standards of scientific evidence” ([Berk, 2004](#)). A case needs to be presented and analyzed supporting or refuting them.

The only way we can make a case for random sampling is to create a random sample *by design*, where we have complete control of who (or what) enters the study, and who (or what) does not. This information is completely external to any data collected.

Random sampling is very difficult (and costly) to implement. First, you must have a well-defined population, for example, adult homeowners in Ithaca, New York. Then you must make sure every homeowner is identified. You must then assign each homeowner a random number, and admit them into a study if their number meets some criterion (e.g., sort their numbers by size, then admit residents with the lowest 1,000 numbers). They must then *agree to participate*.

This process can fail in a number of ways. There may be no population. If the population exists, we may not be able to identify every member of the population. If we can exhaustively identify every member, we may not be able to get them to agree to participate. When we cannot do all of these things, we do not have a random sample, but instead a *convenience sample*, with essentially no model for statistical inference.

With a convenience sample there is no basis for making statements about a population with respect to any statistic. There is no guarantee that the sample mean is an unbiased estimate of the population mean. If we take a convenience sample the actual sample selection mechanism is unknown.

The following scenario will hopefully illustrate the problem. Although there could be any sampling mechanism with a convenience sample, let's just suppose that there are two: 1) the mechanism recruits people who are friends with each other, or 2) the sample is random. Suppose there are 4 people in the population, with scores on a dependent variable of $y_1 = 1$, $y_2 = 2$, $y_3 = 3$, $y_4 = 7,000$, with a population mean of 1752. Also suppose people 1, 2, and 3, are friends.

If we take a sample of size 2 from this population and sampling mechanism 1 is working (friend sampling), then the possible samples are persons (1, 2), (1, 3), and (2, 3) with sample means 1.5, 2, and 2.5. The expected value of the sample mean is just the average of these three means (as they are equally likely under the sampling mechanism), giving us an expected value of the sample mean of 2. Clearly this is very different from the actual population mean of 1752. A similar argument holds for p-values and confidence intervals.

One might argue that because there is some chance that the sampling mechanism is random we don't have to worry too much. Maybe the convenience sample is random by chance! The point is that *we have no idea about the relative probabilities of the different sampling mechanisms*. With a convenience sample, we don't know if we are recruiting friends, randomly, or by some other mechanism (pasta preferences). We can't even assign a probability to each of the different sampling mechanisms, hoping to average across them by their probabilities. This means we can't make any statements about the value of our estimators or their uncertainty.

2.6.1 Four Responses to Non-random Samples

Because random samples are costly, if not impossible, to collect, they are a rare occurrence, so researchers often make arguments that, even though they don't have a random sample, it doesn't matter. They can still calculate a standard error and p-value, and rule out some possible population parameter value with a test. Usually one of four arguments can be made ([Berk and Freedman, 2003](#); [Berk, 2004](#)):

1. The data *are* the population,
2. We can act "as if" the data come from a random sample from a real population.
3. We can act "as if" the data are a random sample from an imaginary population.

4. We have a model of the sampling process that corrects for non-randomness.

The Data are the Population

If we are only interested in the data at hand, and not generalizing to other times, places, people, etc., then we can just treat this data as the population and ignore inference. There simply is nothing to make inference to. Instead, we just need to describe the data we have, using our data summary and conditional distribution stories. But then we need to be careful to not think about inferring things about the population. We must limit our conclusions purely to the data we've collected.

Act “as if” the data come from a random sample from a real population

When we pretend the data come from a random sample, we basically say that, although we know that the data were not *formally* drawn from a truly random sampling mechanism, we cannot think of a reason why the sampling mechanism used would be non-random. Taking samples of water from a local beach to measure concentrations of toxic chemicals at noon every Wednesday would be such an example. We can argue that there is enough time between samples to make them independent, and that time of day does not affect the water's properties. The population we would make an inference to would be the toxic chemical concentrations on the beach for all times and all days, rather than just Wednesdays at noon.

The argument that we can't think of a reason why a non-random sample wouldn't be random is not a scientific argument, it's an argument from ignorance. If any data can be brought to bear to show that the sampling is random, it can be used to support the “act as if” random sampling assumption. Simply stating that we are assuming a non-random sample is random, or that we can't think why it is not random, is not enough. The burden of proof lies on the researcher to show why a non-random sample is random, and this will be difficult, if not impossible, to do.

Act “as if” the data are a random sample from an imaginary population

A third way to handle non-random samples is to say that our sample is actually generated from a population that is not itself real, but part of an *imaginary superpopulation*. Storms in the Pacific is an example: We do not have a population of all possible storms in the Pacific which we can exhaustively list, then sample from, but we can try to say that some set of

storms in the Pacific that we have recordings of are a random realization from the population of all Pacific storms. We think that there is some process that is stable over time generating these storms, so the storms we see are random realizations of this process.

If there is good scientific theory as to why these storms would be random draws from the hypothetical superpopulation, then the theory and evidence supporting it could be presented to make the case for random sampling. The circumstances that define the population must clearly be laid out, the random process that generates the sample must be discussed, and hopefully data from the process are available.

Model of the sampling process that corrects for non-randomness

The last defense is to say that we've modeled the sampling process sufficiently so that we can account for any failures of random sampling. Sample selection corrections, or adaptive sampling schemes, are examples of such model-based sampling approaches. The problem is that these models are difficult to verify. It is my guess that it would be easier to collect a simple random sample than verify these models.

2.6.2 Statistical Inference for a Truly Random Sample

Suppose we've (in a very rare case) successfully presented evidence, both internal and external to the data we've collected, to support the conclusion that we have a random sample, or that we've conducted a study where the sample is random by design. Then, we can say something about how the estimates in our sample regression relate to the population. The standard error of our estimate tells us how far each sample estimate is likely to be from the population estimate, and is a measure of *uncertainty* about the population estimate. For a simple regression slope, the variance of $\hat{\beta}_1$ is:

$$\text{Var}(\hat{\beta}_1) = \frac{MSE}{\sum_{i=1}^n (x_i - \bar{x})^2} = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n - k) \sum_{i=1}^n (x_i - \bar{x})^2} \quad (2.32)$$

where k is the number of model parameters (including intercept). Thus, the standard error of $\hat{\beta}$ is the square root of $\text{Var}(\hat{\beta})$, which is the rMSE divided by the square root of the total variation in the predictor variable:

$$SE_{selfsuff} = \frac{rMSE}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n - k) \sum_{i=1}^n (x_i - \bar{x})^2}} \quad (2.33)$$

Note that:

$$\sum_{i=1}^n (x_i - \bar{x})^2 = (n - 1)\sigma_{selfsuff}^2 \quad (2.34)$$

Where $\sigma_{selfsuff}^2$ is the unbiased sample variance estimate of the regressor $selfsuff$. Thus, we can rewrite $SE_{selfsuff}$ as:

$$SE_{selfsuff} = \frac{rMSE}{\sqrt{n - 1}\sigma_{selfsuff}} \quad (2.35)$$

Calculated in R and compared to the summary from our regression:

```
rMSE <- summary(mto.reg)$sigma
# Calculate the standard error by dividing the regression rMSE
# by the standard deviation of the predictor times the sqrt(n-1)
SE <- rMSE / (sd(mto.data$selfsuff) * sqrt(length(mto.data$selfsuff)-1))
```

Compare this to the output of the R `summary()` function:

```
summary(mto.reg)
```

Notice that the standard error depends on three factors. If the rMSE is larger, that is, we have a worse model of the data, then the standard error is larger. This is intuitive: If we are not very good at predicting the data in general, we should expect that we won't be very good at predicting a population statistic, such as the slope of a linear regression line.

Also notice that the standard error depends on the variation in the predictor variable, in our case, the standard deviation of economic self-sufficiency. All else equal, *greater variation in the independent variable will be associated with smaller standard errors*. Having more observations at different levels of our independent variable help us better understand what the slope looks like. If we have only one level of the independent variable (i.e., no variation), the slope is undefined: We can fit any linear slope to that single point. The more variation in the independent variable, the more constraint there is on the slope of the linear regression, thus reducing the standard error. Note that observations with high leverage increase the spread of the independent variable, thus decreasing the standard error.

Lastly, the standard error depends on the square-root of the sample size, $n - 1$. Again, intuitively this makes sense: If we collect data on the entire population, then the error in generalizing from sample to population should be zero.

95% confidence intervals

Another procedure for generalizing from sample to population is to use confidence intervals. Suppose we want to calculate the confidence interval around the mean of each of our three treatment groups. Our primary assumption is the distribution from which the data for each group was drawn. If we don't know the distribution, that is, we've failed to construct the

conditional distribution story (e.g., we can't explain some outliers), then we can't construct the interval. Thus, the conditional distribution story is necessary to construct a confidence interval (in addition to random sampling).

Suppose we're lucky enough to have a random sample and succeed in showing that our conditional distributions are normally distributed. We can create a 95% confidence interval around our estimate of the population mean for the three groups, roughly, as the mean \pm the 2 times the standard error. Thus the 95% confidence interval for the experimental group is [0.028, 0.0159], for the section 8 group is [0.014, 0.162], and for the control group is [-0.071, 0.076].

When we construct a 95% confidence interval, we've used a decision procedure. We've decided that the confidence interval covers the population parameter. If random sampling and our conditional distribution story are correct, then 95% of the confidence intervals we construct will actually include the population parameter. Essentially, we accept a 5% error rate when using 95% confidence intervals.

There are two common misconceptions about the meaning of a confidence interval. First, a confidence interval is not the same as saying there is a 95% chance the population parameter falls within the interval. Such a statement is Bayesian, as it expresses a belief (probability) about an unobservable quantity (a hidden population parameter). Such an interval is called a *credible interval*, and must be calculated and interpreted differently.

The second common misconception is subtle. 95% confidence intervals are based on sample statistics that will vary from sample to sample. Thus, when we make a 95% confidence interval, it is not *the* confidence interval, but one of many. Thus, it is not the case that if we were to keep sampling from the population that the true population value of interest would fall within the *first* 95% confidence interval we construct, only that 5% of our 95% confidence intervals will fail to cover the population value.

Null Hypothesis Significance Testing

Once we've constructed our confidence intervals, people usually want to say something about "statistical significance." This usually entails calculating a p-value, then rejecting a null hypothesis that there is no statistically significant difference between the means of the treatment and control groups. Ronald Fisher came up with this p-value approach, and it is called *null hypothesis significance testing*. There are, of course, a bunch of problems with this approach, the biggest is that failing to reject the null hypothesis when our sample size is small doesn't tell us much, because, from our standard error equation, our standard error is large. Thus, null hypothesis significance testing generally confounds effect size (the difference in means of the different groups), with sample size, which determines the standard error.

If we want to do such a null hypothesis significance test, we pretty much already know the answer: Our standard errors are relatively large compared to the difference in means between the groups, in part because the differences are small, and also because our sample size is small. If we want to get more precise results, we can either reduce the standard deviation for each group (for example by developing measures with lower measurement error) or increase the sample size. Let's do some t-tests anyway:

```
# Two sample t-test comparing experimental to control group
t.test(experimental, control)
t.test(experimental, section8)
t.test(section8, control)
```

The t-test comparing the experimental and control group gives us a t-value of 1.86, with 116.2 degrees of freedom, and a p-value of 0.065. This coincides with the fact that the bounds of the confidence interval for each group are close to the means of the other groups. In the case of these two groups, the upper bound on the 95% confidence interval for the control group (0.076) is slightly less than the mean of the experimental group (0.094). The lower bound of the 95% confidence interval for the experimental group is 0.028, which is slightly greater than the mean of the treatment group (0.002). The output also mentions that we are using the Welch Two Sample t-test, which basically allows for the two different groups to have different variances, rather than assuming they have the same variances. You can find this out by typing:

```
>?t.test
```

This gives us information about the t.test function. The t-test output also tells us the 95% confidence interval for the difference in means, which is [-0.006, 0.1889]. Because the 95% confidence interval for the difference in means includes zero, the p-value is greater than 0.05. If we wanted to get that p-value below 0.05, to publish a paper, we could try setting the variances to be equal using:

```
t.test(experimental, control, var.equal=TRUE)
```

But, unfortunately, that still gives us $p = 0.065$, so we can't publish and give up on our careers. Another way to game the p-value is to use a one-tailed test:

```
t.test(experimental, control, alternative = "greater")
```

Success! Now the p-value is 0.03! Now, since we were only looking to see whether the experimental group was greater than the control, I'm sure we can justify using the one-tailed test to ourselves. But it seems pretty weird for a paper to be publishable, and conclusions to be different, with the same confidence intervals, and thus same uncertainty, even though we have different p-values. Thus, we can see how p-value fishing is deceptive, violating Feynman's rule, and summarizing the uncertainty using confidence intervals is a better way to go (if confidence intervals make sense at all).

As can be seen, p-values don't give us nearly enough information about our statistic as we want. We can have a low p-value due to a large sample size, large effect, or low error variance.

Resampling: Bootstrap, Jackknife

So far, the way we've constructed the t-test and confidence interval depends on a distributional assumption. Maybe the sampling distribution of the means are not normally distributed. Do we need to make such an assumption?

The answer turns out to be that the assumption gets us something: A more powerful test, meaning lower p-values. However, the assumption sacrifices robustness: If the sampling distribution of the mean is not normal, then our inferences are wrong. Thus, there tends to be a tradeoff in statistical methods, where those with more assumptions tend to be more powerful, and are able to estimate results more precisely, whereas those that are more robust tend to require larger sample sizes.

There are some non-parametric tests out there that can be useful in some cases. However, there are ways of calculating these statistics in a very general manner, called *bootstrapping* ([Efron, 1979](#); [Efron and Tibshirani, 1993](#)). Bootstrapping uses the following logic: Frequentist inference requires that we imagine repeatedly sampling from a population, constructing a statistic (e.g., the sample mean), then seeing how that statistic varies over the samples (the sampling distribution of the statistic). Using that sampling distribution we can make statements about the probability of our particular sample occurring. However, this requires us to assume some sampling distribution of the statistic, such as the normal distribution. It also requires that we can derive a closed form solution for the variance of an estimator, which is often very difficult, if not impossible for more complicated statistics (e.g., the variance of the median of an unknown distribution).

Instead of assuming the sampling distribution or spending all of our time deriving variances of estimators, resampling techniques, such as the bootstrap, simulate the sampling distribution by treating the sample data that we observe as the population, then sampling from this sample. If the original sample is a random sample from the actual population, then the random subsample of the sample is also a random sample from the actual population.

The approach is very similar to cross-validation. We take a random sub-sample from our sample, but this time *with replacement*, calculate the statistic we are interested in, then repeat this a bunch of times:

```
slope <- c()

for(i in 1:100){
  # Take a random sample of cell ids with replacement
  bootids <- sample(mto.data$cell_id,
    length(mto.data$cell_id),
    replace = TRUE)

  newdata <- data.frame()

  for(i in 1:length(bootids)){
    # create a new data frame with the cell ids selected to be in the sample
    newdata <- rbind(newdata,
      mto.data[mto.data$cell_id == bootids[i], ])
  }

  # Run the regression on the sampled data and store the parameter estimate
  reg <- lm(happy ~ selfsuff, data = newdata)

  # coef(reg)[2] gets the second parameter from the "reg" regression
  # If we did coef(reg)[1] we would get the intercept estimate
  slope <- append(slope, coef(reg)[2])
}
```

This is super slow, however, looking at the mean and standard deviation of the estimated slopes from the bootstrap and comparing to the original regression:

```
mean(slope)
sd(slope)
summary(mto.reg) # compare to regression
```

The mean of the bootstrap estimates of the slope is the same as that in the original regression, but the standard deviation is larger than the standard error from the original regression. Because the bootstrap distribution of parameter estimates may be skewed, it is better to use the 2.5 and 97.5 quantiles rather than 2 times the standard deviation to form confidence intervals:

```
# Bootstrapped confidence interval for the slope
slope.interval <- quantile(slope, probs = c(.025, .975))
```

The confidence interval from the bootstrap and original regression are very similar.

Confidence Intervals for the Conditional Mean

We can also make predictions about the conditional mean with a confidence interval. Confidence intervals constructed in this way should cover the population *conditional mean* 95% of the time. Recall from our discussion of leverage that:

$$\text{Var}(\hat{y}_k) = \text{Var}(h_{kk}y_k) = \sigma^2 h_{kk} \quad (2.36)$$

where h_{kk} is the k th row and column of the hat matrix H . So a confidence interval around the predicted value (or conditional mean) \hat{y}_k is just $\hat{y}_k \pm 2\sqrt{\hat{\sigma}^2 h_{kk}}$, where $\hat{\sigma}^2$ is the MSE of the regression, and h_{kk} is just the leverage for observation k . Thus the equation for the confidence interval in simple linear regression with one regressor is:

$$CI_k = \hat{y}_k \pm 2 \times rMSE \sqrt{\frac{1}{n} + \frac{(x_k - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}} \quad (2.37)$$

Where y_k is the fitted value of the conditional mean (i.e., a value on the regression line), $rMSE$ is the root-mean-squared error of the regression, and the term in the square root is the leverage of the predicted value y_k .

Notice that the confidence interval for our predictions grows as we move away from the mean of x . Why is this the case? As we previously found, observations farther away from the mean of a predictor variable have a smaller residual variance because they have more leverage. On the other hand, the regression line tends to be pulled toward these observations, giving the regression line higher variance. Let's calculate the confidence interval for the conditional mean at $x_k = 0.5$:

```
x.bar <- mean(mto.data$selfsuff) # Calculate the mean of x
x.k <- 0.5 # set the x value of our prediction
se.y <- summary(mto.reg)$sigma # get rMSE
ss.x <- sum((mto.data$selfsuff - x.bar)^2) # sum of squared deviations
# Make a prediction from our regression when selfsufficiency is 0.5
yk.hat <- predict(mto.reg, newdata = data.frame(selfsuff = x.k))
CI.upper <- yk.hat + 2*se.y*sqrt(1/nrow(mto.data) + (x.k - x.bar)^2/(ss.x))
CI.lower <- yk.hat - 2*se.y*sqrt(1/nrow(mto.data) + (x.k - x.bar)^2/(ss.x))
png(filename = "Cintervalxk.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
```

```
plot(mto.data$selfsuff, mto.data$happy,
      xlim = c(-1, 1),
      ylim = c(-1, 1),
      pch  = 19,
      col  = rgb(0, 0, 0, .2),
      xlab = "Mean Economic Self-Sufficiency z-scores",
      ylab = "Mean Happiness z-scores")
abline(mto.reg, lwd = 2)
lines(c(x.k, x.k), c(CI.lower, CI.upper),
      col = rgb(1, 0, 0, 1),
      lwd = 2)
dev.off()
```

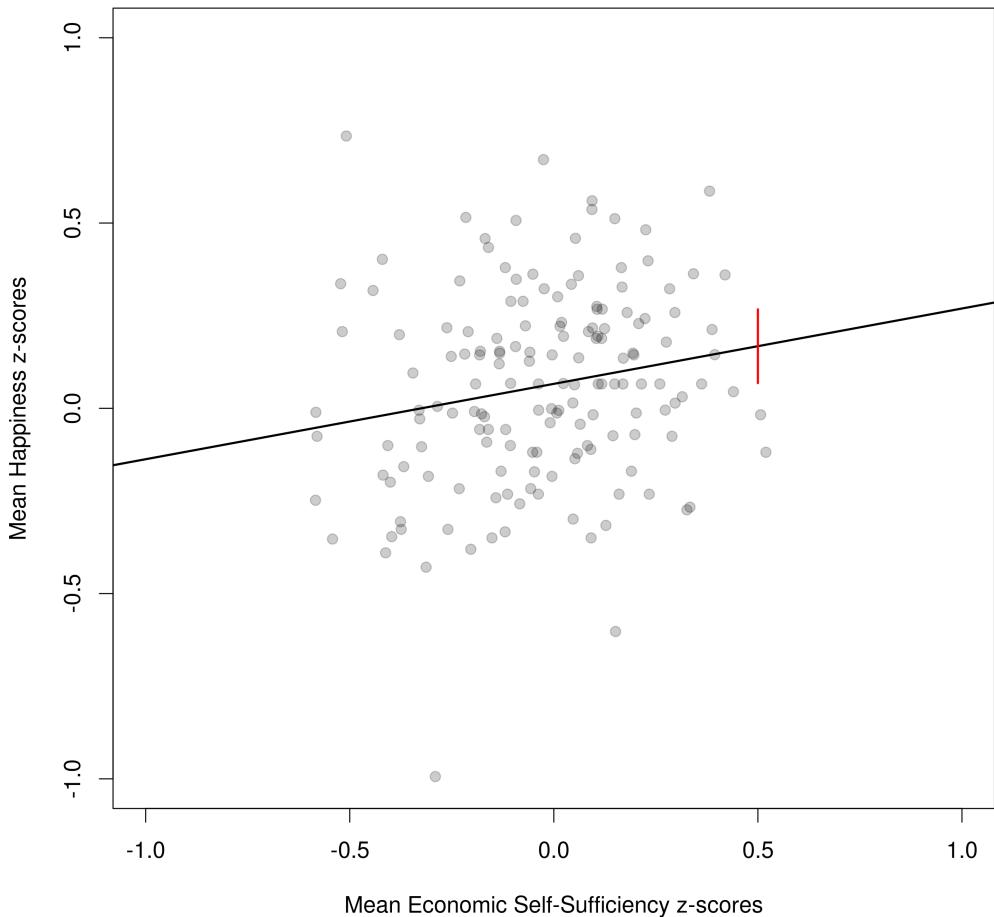


Figure 2.37: Confidence interval for the mean of happiness scores when economic self-sufficiency is 0.5.

We can repeat this and make predictions for each value of economic self-sufficiency in the data, to get confidence “bands” for the conditional mean:

```
# Make confidence intervals for each value of self sufficiency
conf.pred1 <- predict(mto.reg, mto.data, interval = "confidence")

png(filename = "Cinterval.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(mto.data$selfsuff, mto.data$happy,
     xlim = c(-1, 1),
     ylim = c(-1, 1),
     pch = 19,
```

```
col = rgb(0, 0, 0, .2),
xlab = "Mean Economic Self-Sufficiency z-scores",
ylab = "Mean Happiness z-scores")
abline(mto.reg, lwd = 2)
# Plot lower bound of the confidence interval
lines(sort(mto.data$selfsuff),
      sort(conf.pred1[, c("lwr")]),
      lwd = 2)
# Plot upper bound of the confidence interval
lines(sort(mto.data$selfsuff),
      sort(conf.pred1[, c("upr")]),
      lwd = 2)
dev.off()
```

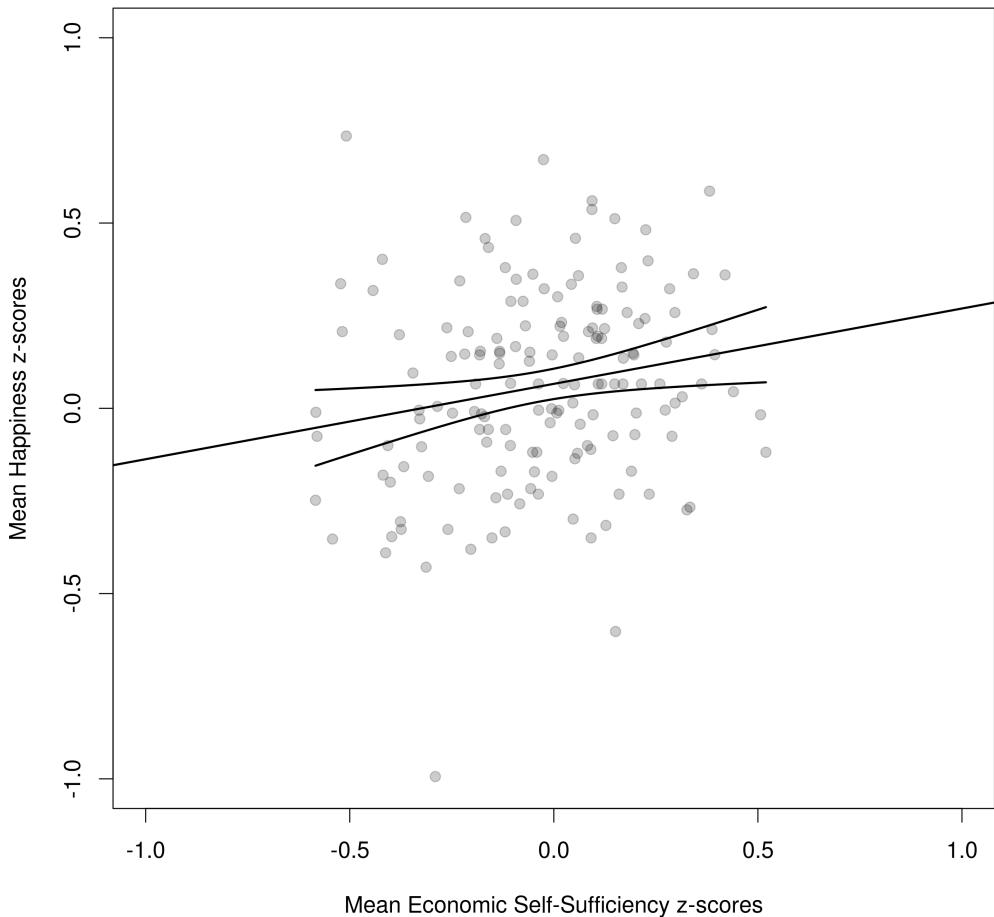


Figure 2.38: Confidence bands for the conditional mean happiness z-scores as a function of mean economic self-sufficiency z-scores.

2.7 The Causal Inference Story

Our last story is about causal inference. Philosophers have long known that people think in causal terms, and psychologists have had some luck verifying this (Tversky and Kahneman, 1977; Gopnik et al., 2004). As a result, suspending our belief that our models are causal, and instead thinking in terms of conditional distributions and data summaries takes some work. Once we've done that, we can start thinking about the requirements for making a causal argument from our data.

The key concept of causality is *manipulation*: We change a variable and

see how the dependent variable changes, rather than observing the variables as they covary together naturally. If that change is made in a way that is uncorrelated with everything else in the world (i.e., it is random), then we've identified a causal effect.

2.7.1 Counterfactuals

Manipulating a variable is important because we want to see how the output changes as a result of a change in that variable. In ideal circumstances, we can see this by changing the level of an input for a single unit, and observing the output *for that same unit*. This is compared to another group set at a different level of that variable. These are *counterfactuals*.

For example, if someone gave me a blood pressure drug and didn't give me a blood pressure drug at the same time, then comparing my blood pressure and side effects in the two different worlds would measure the causal effect of the drug. These parallel worlds have everything exactly the same, the only difference is the drug. However, we can't both give me the drug and not give me the drug; that is a contradiction. We can, however, give me the drug and give someone else *like me* no drug as the counterfactual comparison, but we are not exactly the same. We can only guarantee that this comparison will be accurate, in the long run, if whoever gets the drug and doesn't is random.

2.7.2 Random Assignment

From a statistical point of view, the “gold standard” for making a causal argument is a randomized controlled trial. You use a randomizing device to give some people or groups the “treatment”, whatever you are interested in, and others are given the control (usually nothing). Because factors that cause people to be in the treatment and control groups are explicitly random, members of the treatment and control groups serve as valid counterfactuals for each other. A study that uses an experimental design with such randomization is called a randomized controlled trial.

Such random assignment works perfectly in many cases, but when it comes to people, a bunch of problems arise. These problems come from one of three sources: 1) failure to implement random assignment correctly, 2) knowledge of giving or receiving a “treatment” or “control” affecting the behavior of participants in a study or researchers, and, 3) participants non-randomly choosing to participate or drop out from the study. A fourth problem is even more insidious, where the treatment itself may be poorly designed or implemented. These problems are associated with study outcomes that look more favorable than they should ([Schulz et al., 1995](#)).

Failure to Implement Random Assignment: Sequence Generation and Allocation Concealment

Medical researchers have done a lot of randomized controlled trials and have found, surprisingly, that randomization in theory is not the same as randomization in practice. First, someone has to actually create a sequence of assignments to the treatment or control group that approximate randomness. However, people intuitively think non-random processes are random, and tend to skimp on creating actually random sequences ([Schulz and Grimes, 2002b](#)).

There is also evidence of researchers failing to implement a random sequence correctly because, for whatever reason, they think their choice is better than randomness ([Schulz, 1995a,b, 1996](#)). This might occur because a doctor has a patient who is very ill and believes the patient really needs a treatment drug rather than a placebo, even though randomness assigned the patient to receive the placebo. Allocation concealment methods are procedures for making sure researchers cannot subvert random assignment prior to its implementation, and focus on coordination and verification between those implementing the randomization and those at a central location generating and verifying the random assignment ([Schulz and Grimes, 2002a](#)).

Participant and Researcher Knowledge: Blinding, Placebo Effect, Hawthorne Effect

After random assignment occurs, the counterfactual comparison that we set up can still be undermined. Randomized controlled trials are usually “double blind,” meaning neither the boots on the ground researchers implementing the study nor participants know who received the treatment and control conditions ([Altman et al., 2004; Schulz et al., 1996](#)). Other researchers can also be blinded, and the meaning of blinding is generally not well understood ([Haahr and Hróbjartsson, 2006](#)).

Researchers are “blinded” from knowing which condition received the treatment because if they did know they may treat these groups of participants differently. For example, knowing someone is in a control group may make researchers feel like these participants haven’t received enough care, resulting in researchers giving more attention to the control group. Similarly, if participants know they received a placebo, they may report more symptoms because they expect no improvement in their health, and they may seek other treatments. Thus, to maintain our valid counterfactual comparisons, participants in our study not only need to randomly receive a different intervention, but also believe that they’ve received the same intervention or that they are equally likely to receive the treatment or control group as anyone else.

Placebo effects and Hawthorne effects are perfect examples of why the

beliefs of those in the control and treatment groups matter. A placebo perfectly mimics the treatment except for the active ingredient. It has the same taste, smell, touch, and look. Researchers have found that merely giving people a sugar pill can improve their health outcomes, showing an astonishing effect of beliefs on human behavior. Even worse, merely even knowing one is in a study, rather than believing one is receiving a treatment or placebo, can affect behavior and outcomes. One paper found about a 3% reduction in overall electricity use for people sent a postcard telling them their electricity use was being monitored, compared to a control group who was not contacted (Schwartz et al., 2013).

In short, beliefs matter a lot when it comes to people. The statistician's random assignment works well for non-human experiments, but equating the beliefs of researchers and participants in the treatment and control group is necessary for experiments involving humans.

Non-random Unassignment: Volunteerism and Attrition

In a study involving people, those who are interested in receiving the treatment will probably be more likely to volunteer for the experiment than someone randomly selected from the population. For example those who are sick or believe in the effectiveness of a new medical technology will be more likely to volunteer, and because of their beliefs, behavior, or other factors, more likely to benefit from random assignment to the treatment group than someone randomly selected from the population.

On the other hand, those who are not benefiting may drop out of a study. Thus, those left in the study who we have outcome data on will be those who benefitted relatively more than the general population. It's not hard to see how this attrition will bias the statistical effectiveness of the treatment.

2.7.3 Quasi-Experiments

There are a bunch of approaches that try to proxy for random assignment or otherwise handle data that are non-experimental. These are sometimes called *quasi-experimental* approaches. They have generally been a disaster. As (Berk, 2004) wrote:

“I recall a conversation with Don Campbell in which he openly wished that he had never written “Campbell and Stanley” (1963). The intent of the justly famous book, *Experimental and Quasi-Experimental Designs for Research*, was to contrast randomized experiments to quasi-experimental approximations and to strongly discourage the latter. Yet the apparent impact of

the book was to legitimize a host of quasi-experimental designs for a wide variety of applied social science research.”

2.8 Mathematical Appendix

2.8.1 Why confidence intervals for the conditional mean fan out

It is easy to show why the variance of the conditional mean depends on the leverage using the hat matrix. Here is a mathematical note on the topic without matrices, following Wackerly et al. (2008) Chapter 11. Consider the estimated linear regression $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$. We are interested in predicting the mean of y at some specific value of x , call it x_k . The variance of our estimate \hat{y} is:

$$\text{Var}(\hat{y}) = \text{Var}(\hat{\beta}_0 + \hat{\beta}_1 x_k) \quad (2.38)$$

Using the variance property:

$$\text{Var}(\hat{\beta}_0 + \hat{\beta}_1 x_k) = \text{Var}(\hat{\beta}_0) + x_k^2 \text{Var}(\hat{\beta}_1) + 2x_k \text{Cov}(\hat{\beta}_0, \hat{\beta}_1) \quad (2.39)$$

which treats x_k as a constant rather than a random variable. In the next chapter we'll show that:

$$\text{Var}(\hat{\beta}_0) = \frac{\sigma^2 \sum_{i=1}^n x_i^2}{n \sum_{i=1}^n (x_i - \bar{x})^2} \quad (2.40)$$

and we've already noted that:

$$\text{Var}(\hat{\beta}_1) = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2} = \frac{\sigma^2}{\sigma_x^2(n-1)} \quad (2.41)$$

where σ^2 is the MSE from the regression. The estimated value of the intercept and slope are correlated, giving us:

$$\text{Cov}(\hat{\beta}_0, \hat{\beta}_1) = \frac{-\bar{x}\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (2.42)$$

These all have σ^2 , so pulling that out, we get

$$\text{Var}(\hat{y}) = \text{Var}(\hat{\beta}_0 + \hat{\beta}_1 x_k) = \sigma^2 \left(\frac{\frac{\sum_{i=1}^n x_i^2}{n} + x_k^2 - 2\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right) \quad (2.43)$$

Using the identity $\frac{\sum_{i=1}^n x_i^2}{n} = \frac{\sum_{i=1}^n (x_i - \bar{x})^2 + n\bar{x}^2}{n}$, we can rearrange and show that:

$$\frac{\frac{\sum_{i=1}^n x_i^2}{n} + x_k^2 - 2\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2} = \frac{1}{n} + \frac{(x_k - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (2.44)$$

2.9 Homework 2

All homework will have a replication plus extension format. There's a very important reason for this: **If you're going to base your results on a paper, you'd better be sure the results are right.** Thus, you should get in the habit of replicating the results of papers that you want to trust. The extension part takes things further, seeing how reported results hold up to exploration the original authors may have omitted, as well as looking around for new discoveries.

In this homework assignment we will first replicate the results of a previous paper, then extend those results using the five stories discussed in this chapter. The homework is worth 10 points in total, with up to two bonus points.

First, read sections 6.1, 6.2, and 6.4 from [Bierens and Ginther \(2001\)](#). Next, write a short report that replicates Tables 1.A and 2.A on page 317. The data can be obtained from the package AER:

```
install.packages("AER")
library(AER)
data(CPS1988)
```

Here is what I expect you to do in this replication report:

1. Conduct the median (LAD) regressions. (1 point)
2. Compare your results to those reported in the paper. (1 point)
3. Briefly explain what you think the regression summaries mean (give it your best). (1 point for effort)

Note that if we want to conduct a regression of y on $x + x^2$, we use the following R command:

```
lm(y ~ x + I(x^2))
```

After attempting to reproduce Tables 1.A and 2.A, write a short report (in the same document) that extends the replication, telling each story of the data. Make sure your report includes reproducible R code, either as an appendix or through Harvard's Dataverse (or some other site of your choice). Here's what I expect to be included in the extension part of the report:

1. Create histograms of the wages, log wages, education, and experience variables, with a summary of what you're seeing. (1 point)
2. Create scatterplots of wages and log wages against education and experience, with fitted linear or median regression lines, again with a summary. (1 point)

3. Compare wages and log wages to the normal distribution, with a summary of what you think is going on. (1 point)
4. Plot the regression residuals for the regressions in Tables 1.A and 2.A, using ordinary linear regression rather than median regression. Are there outliers? If so, what type are they? Would median regression or ordinary linear regression be more appropriate for these data? (1 point)
5. Using median or ordinary linear regression, evaluate the backcasting quality of the Mincer type model (Table 1.A). Use 5-fold cross-validation to test whether this model is too complex. (1 point)
6. Discuss the statistical inference story. Explain why using confidence intervals would be valid or invalid for these data. (1 point)
7. Discuss the causality story. To what degree do you think we can make causal claims about the regression models used? Explain your reasoning. (1 point)
8. (Optional) Extend the data summary and/or conditional distribution stories in novel ways, either using methods covered in the lecture notes, or methods that you've developed yourself. (1-2 bonus points)

You may have to do some light reading about CPS dataset to tell some of the stories (just extract the key features).

2.10 Homework 2 Solution

First, read sections 6.1, 6.2, and 6.4 from [Bierens and Ginther \(2001\)](#). Next, write a short report that replicates Tables 1.A and 2.A on page 317. The data can be obtained from the package AER:

```
install.packages("AER")
library(AER)
data(CPS1988)
```

2.10.1 Replication

Table 1.A

We can reproduce Table 1.A using the following code:

```
install.packages("AER", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(AER)
data(CPS1988)
```

```

library(quantreg)
reg1.A <- rq(log(wage)
~ethnicity + education + experience + I(experience^2),
data = CPS1988)

```

We get the same point estimates as reported in the paper, however the t-values are different. The reason for this difference is that in the original paper they use kernel density estimation of the error distribution, whereas the quantile regression from the quantreg package uses some other procedure. Noting the replication of the point estimates but not standard errors is sufficient.

Table 2.A

We can reproduce Table 2.A using the following code:

```

reg2.A <- rq(log(wage)
~ethnicity + education + experience + I(experience^2)
+I(experience^3) + I(experience^4) , data = CPS1988)

```

Again, the point estimates are the same, and t-values different.

Brief Explanation of Table 1.A and 2.A

Because the dependent variable is in log wages, I don't expect a correct answer for the explanation. With a log transformed dependent variable, a one unit change in the regressor can be thought of as having an x% change in the conditional *median* of wages (because we are using median regression), where x is the coefficient in the regression times 100. So, for example, a person with one additional year of education has a median wage that is 9% greater than a person without that additional year of education. It's also fine to say something like blacks have a median log wage that is 0.25 points lower than whites, or something like that. Anything similar to that is acceptable for Table 2.A.

2.10.2 The Data Summary Story

Weekly Wages

Let's first take a look at the distribution of weekly wages. We'll use the Freedman-Diaconis rule to select bin width for a histogram, as it is likely that there are wage outliers:

```

png(filename = "wages.png", width = 3000, height = 3000, res = 300)
par(cex = 1.5, mar = c(5, 4, 1, 1), oma = c(0, 0, 0, 0))

```

```
hist(CPS1988$wage,
     prob = FALSE,
# There should be some rule for selecting bins other than the default (Sturges)
# Freedman-Diaconis ("FD"), Scott, or kernel bw selection are all fine
     breaks = "FD",
     ylim = c(0, 2000),
     xlim = c(0, 20000),
     xlab = "Weekly Wage",
     main = "Histogram of Weekly Wages")
rug(CPS1988$wage, col = rgb(1, 0, 0, .3))
rug(CPS1988$wage[CPS1988$wage > 2400], col = rgb(0, 0, 1, 1))
dev.off()
```

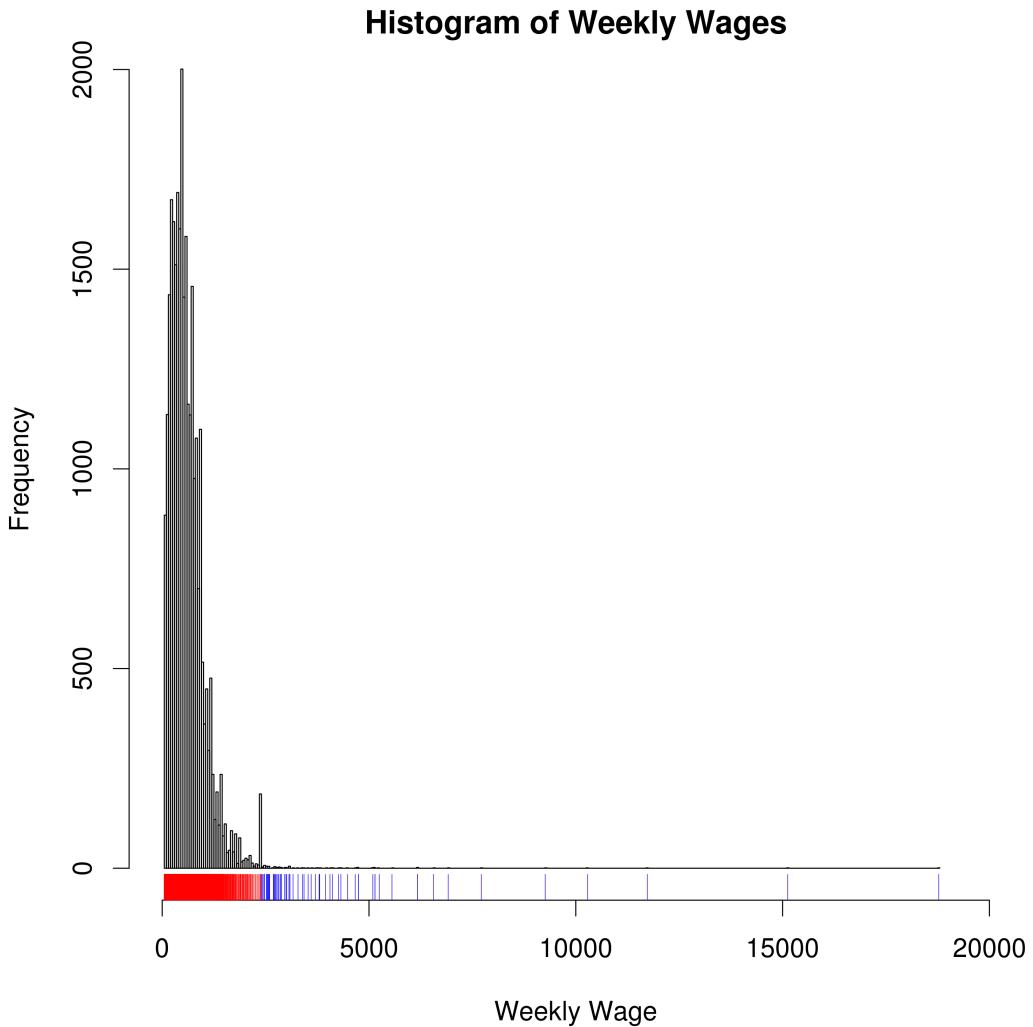


Figure 2.39: Histogram of weekly wages from the 1988 CPS data. Histogram bandwidth is chosen using the Freedman-Diaconis rule.

We can see that most weekly wages are below \$2,000, with a spike around \$2,400. There's also a few outliers, with weekly wages over \$5,000. We can see the reason why labor economists are interested in median regression, given the possibility of outliers.

Next, let's see whether taking the log transform helps. Here I'm using the Scott rule, as it is geared for normally distributed data:

```
png(filename = "logwages.png", width = 3000, height = 3000, res = 300)
par(cex = 1.5, mar = c(5, 4, 1, 1), oma = c(0, 0, 0, 0))
hist(log(CPS1988$wage),
prob = FALSE,
breaks = "Scott",
```

```

ylim = c(0, 2000),
xlab = "Log Weekly Wage",
main = "Histogram of Log Weekly Wages")
rug(log(CPS1988$wage), col = rgb(1, 0, 0, .3))
rug(log(CPS1988$wage[CPS1988$wage > 2400]), col = rgb(0, 0, 1, 1))
dev.off()

```

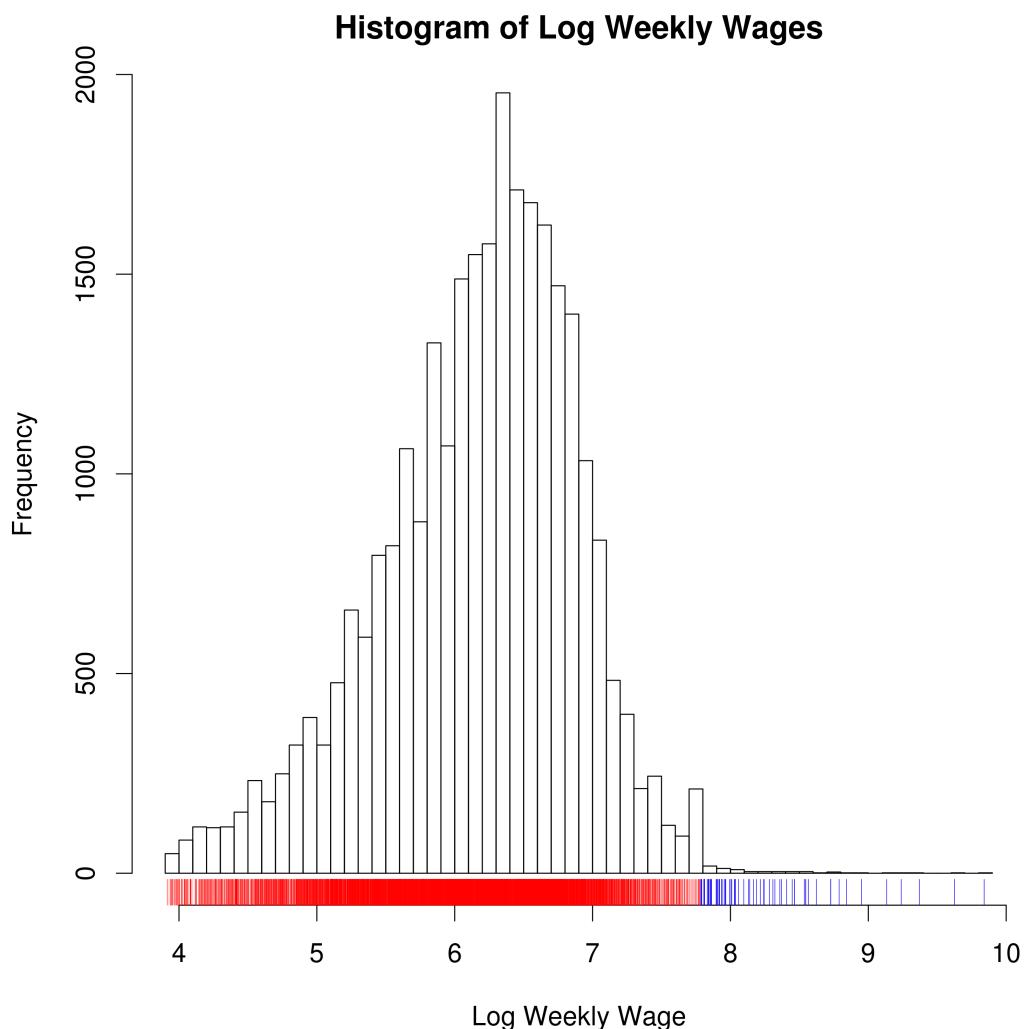


Figure 2.40: Histogram of log weekly wages from the 1988 CPS data. Histogram bandwidth is chosen using the Scott rule.

The log-transformed weekly wages look much more like a normal distribution than the un-transformed weekly wages. However, there's still that bump and the long right tail.

Education

Next, let's look at education:

```
png(filename = "educationhist.png", width = 3000, height = 3000, res = 300)
par(cex = 1.5, mar = c(5, 4, 1, 1), oma = c(0, 0, 0, 0))
hist(CPS1988$education,
     prob = FALSE,
# There should be some rule for selecting bins other than the default (Sturges)
# Freedman-Diaconis ("FD"), Scott, or kernel bw selection are all fine
     breaks = "FD",
     xlab = "Years of Education",
     main = "Histogram of Years of Education",
     xlim = c(0, 20))
dev.off()
```

The number of years of education among those in the sample ranges from zero to 18 years. There is a large spike for those with 12 years of education, which must correspond to a high school diploma. The frequencies above high school diploma are greater than below, indicating the sample is relatively well educated.

Experience

Finally, let's look at experience:

```
png(filename = "experiencehist.png", width = 3000, height = 3000, res = 300)
par(cex = 1.5, mar = c(5, 4, 1, 1), oma = c(0, 0, 0, 0))
hist(CPS1988$experience,
     prob = FALSE,
# There should be some rule for selecting bins other than the default (Sturges)
# Freedman-Diaconis ("FD"), Scott, or kernel bw selection are all fine
     breaks = "FD",
     xlab = "Years of Potential Experience",
     ylim = c(0, 1000),
     xlim = c(-10, 70),
     main = "Histogram of Years of Potential Experience")
axis(side = 1, at = seq(-10, 70, by = 10))
dev.off()
```

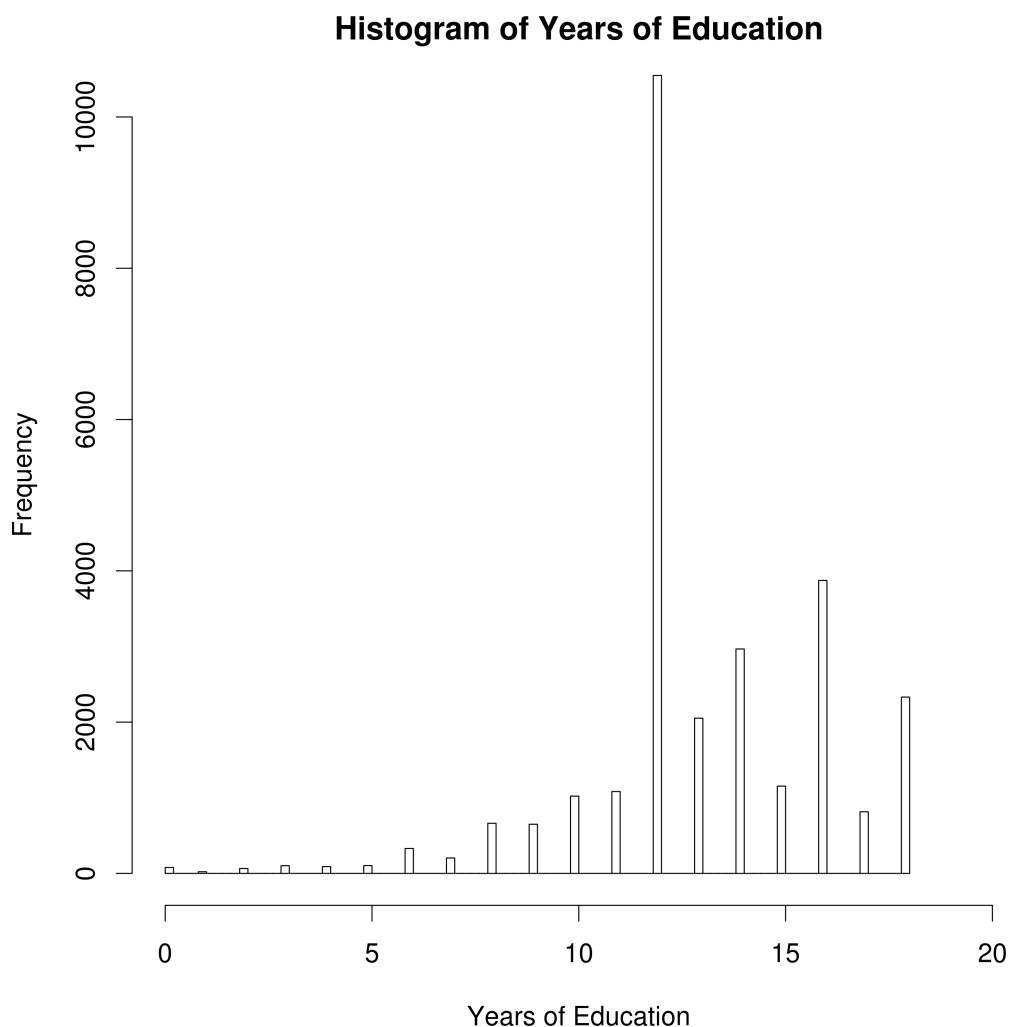


Figure 2.41: Histogram of years of education from the 1988 CPS data. Histogram bandwidth is chosen using the Freedman-Diaconis rule.

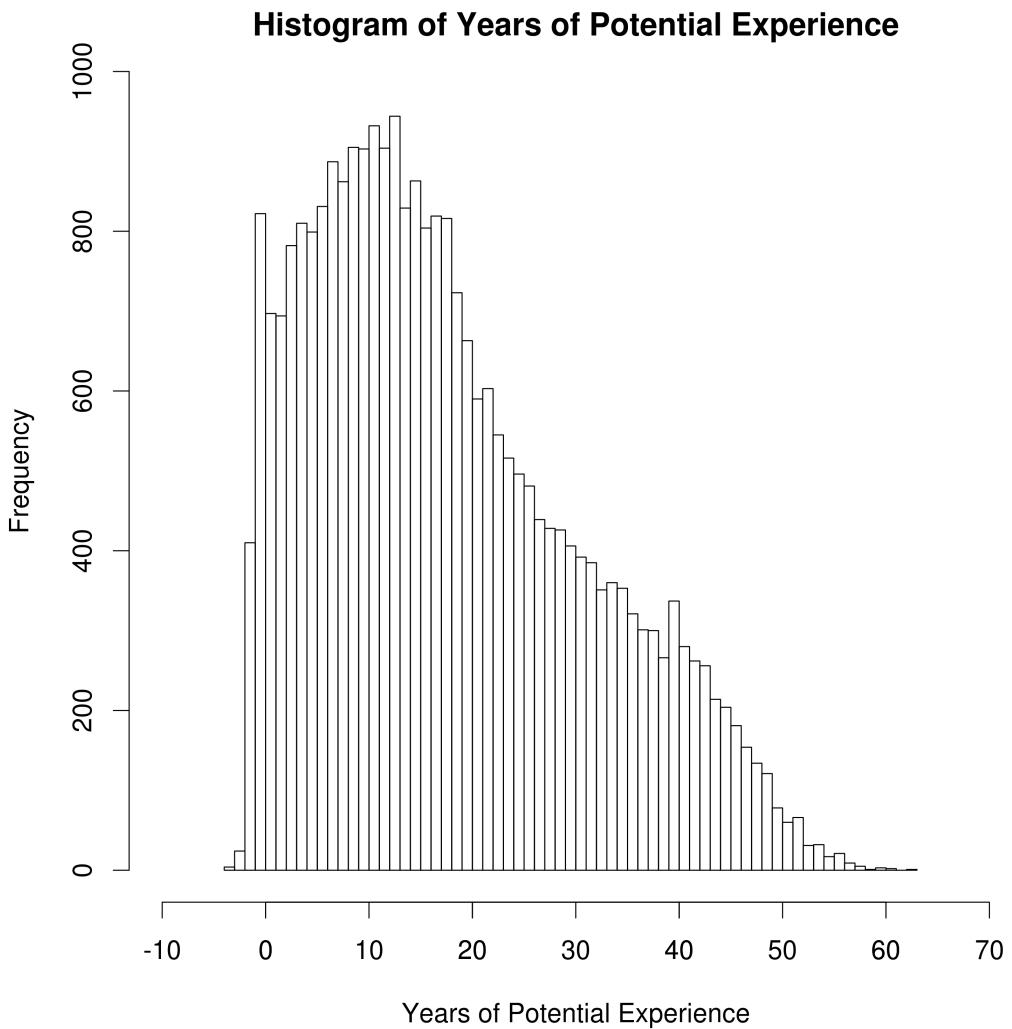


Figure 2.42: Histogram of years of potential experience from the 1988 CPS data. Histogram bandwidth is chosen using the Freedman-Diaconis rule.

The number of years of experience peaks between 10 and 20 years, with the frequency fairly smoothly dropping after 20 years. However, there's a quite bizarre anomaly in the data, where over 400 people have *negative* years of experience. This does make sense given that they use *potential* years of experience, which is age – years of schooling – 6. Basically, you can't have any work experience while in school or under the age of six (when you enter first grade). The negative numbers probably come from 18 year olds who skipped grades.

It seems pretty weird to have potential years of experience, and *squared* potential years of experience, when potential experience can take on negative values. Squaring those with negative years of experience will make them

positive, as though they had more potential experience than someone with zero years of experience. In fact, some people have -3 or even -4 potential years of experience, meaning they would be treated as having as many potential years of experience as someone with 9 or 16 years! Granted, there are only a few of these observations, but the conditional distribution story won't make sense unless that problem is fixed.

A simple approach is to recode the experience variable as zero for all negative numbers:

```
# Recodes observations less than zero as zero,
# Don't change the other observations
CPS1988$exp.zero <- ifelse(CPS1988$experience < 0, 0, CPS1988$experience)
```

If we do this, then those with zero potential experience are the most frequent category:

```
png(filename = "exphistzero.png", width = 3000, height = 3000, res = 300)
par(cex = 1.5, mar = c(5, 4, 1, 1), oma = c(0, 0, 0, 0))
hist(CPS1988$exp.zero,
     prob = FALSE,
# There should be some rule for selecting bins other than the default (Sturges)
# Freedman-Diaconis ("FD"), Scott, or kernel bw selection are all fine
     breaks = "FD",
     xlab = "Years of Potential Experience",
     xlim = c(-10, 70),
     main = "Histogram of Years of Potential Experience")
axis(side = 1, at = seq(-10, 70, by = 10))
dev.off()
```



Figure 2.43: Histogram of years of potential experience from the 1988 CPS data, where observations with negative experience are recoded as zero. Histogram bandwidth is chosen using the Freedman-Diaconis rule.

Scatterplots

Let's look at the bivariate scatterplots of weekly wages on education:

```

rq1 <- rq(wage ~ education, data = CPS1988)
lm1 <- lm(wage ~ education, data = CPS1988)
rq2 <- rq(log(wage) ~ education, data = CPS1988)
lm2 <- lm(log(wage) ~ education, data = CPS1988)
png(filename = "scattereducation.png", width = 3000, height = 3000, res =300)
par(cex = 1.5, mar = c(5, 4, 2, 1), mfrow = c(2,1))

```

```
plot(jitter(CPS1988$education),
      jitter(CPS1988$wage),
      ylab = "Weekly Wages",
      xlab = "Years of Education",
      pch = 19,
      cex = .5,
      col = rgb(0, 0, 0, .1),
      xlim = c(0, 20),
      ylim = c(0, 20000))
abline(rq1, col = "red", lwd = 2)
abline(lm1, col = "blue", lwd = 2)
plot(jitter(CPS1988$education),
      jitter(log(CPS1988$wage)),
      ylab = "Log Weekly Wages",
      xlab = "Years of Education",
      cex = .5,
      pch = 19,
      col = rgb(0, 0, 0, .1),
      xlim = c(0, 20))
abline(rq2, col = "red", lwd = 2)
abline(lm2, col = "blue", lwd = 2)
dev.off()
```

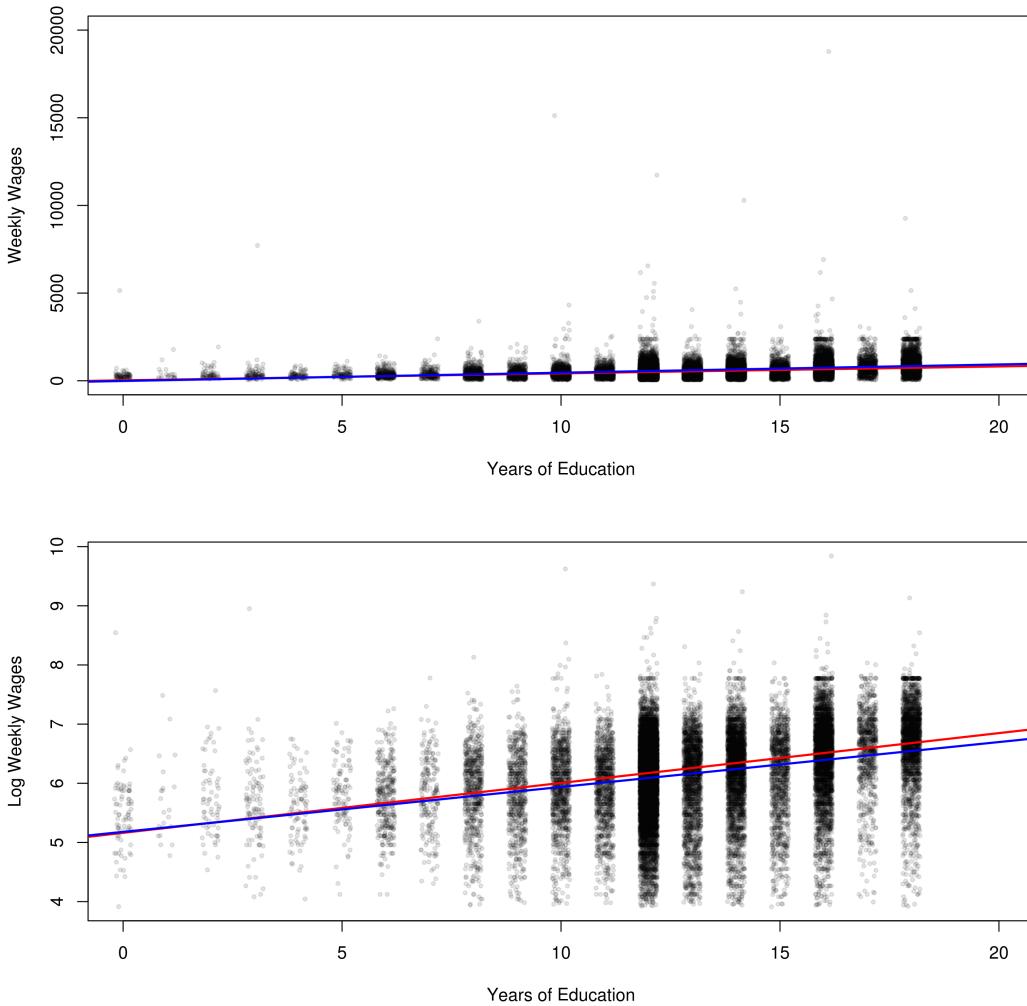


Figure 2.44: Scatterplot with wages (top plot) or log wages (bottom plot) and years of education (x axis) with median regression line (red) and least-squares regression line (blue).

Wages seem to be increasing with years of education, but the variability also increases, as does the presence of outliers. There are many people with about 12 years of education, the most common category, with a wide range of wages.

Let's do the same for experience:

```
rq1 <- rq(wage ~ exp.zero, data = CPS1988)
lm1 <- lm(wage ~ exp.zero, data = CPS1988)
rq2 <- rq(log(wage) ~ exp.zero, data = CPS1988)
lm2 <- lm(log(wage) ~ exp.zero, data = CPS1988)
png(filename = "scatterexperience.png", width = 3000, height = 3000, res =300)
```

```

par(cex = 1.5, mar = c(5, 4, 2, 1), mfrow = c(2,1))
plot(jitter(CPS1988$exp.zero),
     jitter(CPS1988$wage),
     ylab = "Weekly Wages",
     xlab = "Potential Years of Experience",
     pch = 19,
     cex = .5,
     col = rgb(0, 0, 0, .1),
     ylim = c(0, 20000))
abline(rq1, col = "red", lwd = 2)
abline(lm1, col = "blue", lwd = 2)
plot(jitter(CPS1988$exp.zero),
     jitter(log(CPS1988$wage)),
     ylab = "Log Weekly Wages",
     xlab = "Potential Years of Experience",
     cex = .5,
     pch = 19,
     col = rgb(0, 0, 0, .1))
abline(rq2, col = "red", lwd = 2)
abline(lm2, col = "blue", lwd = 2)
dev.off()

```

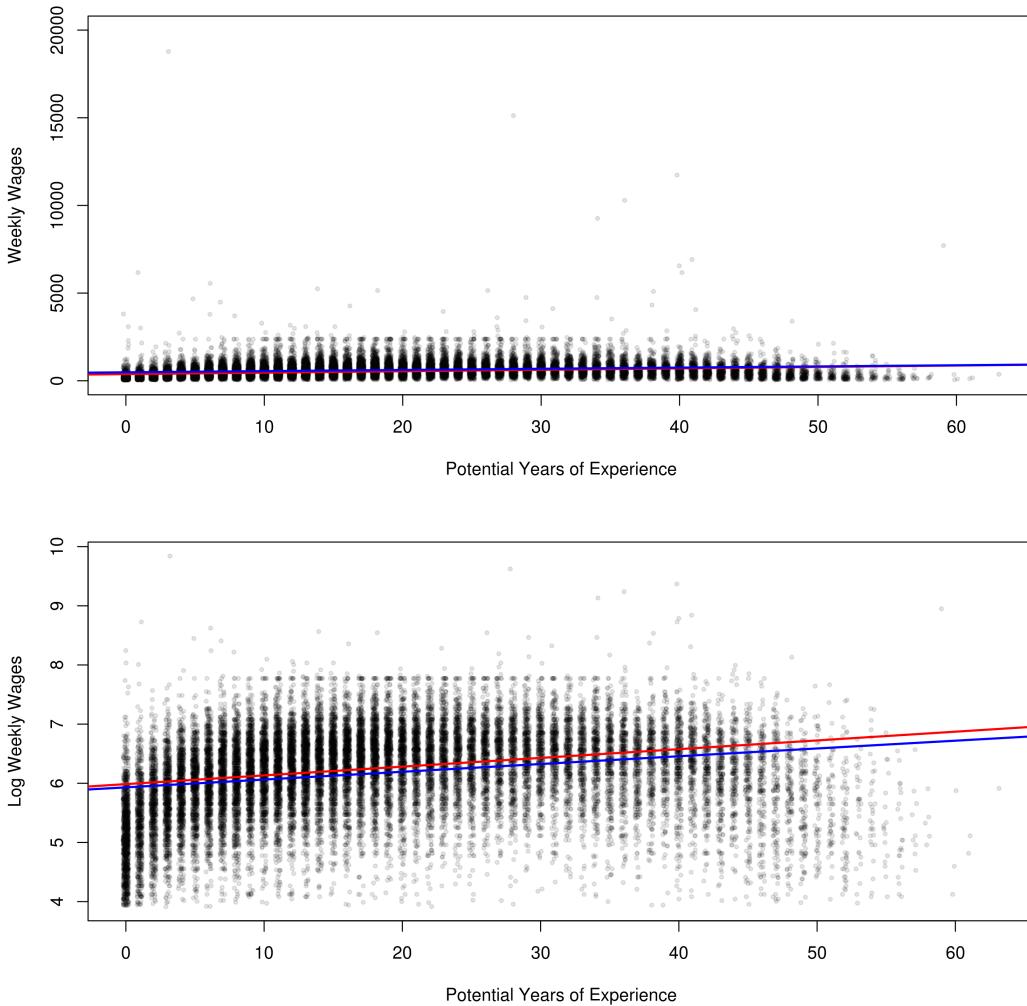


Figure 2.45: Scatterplot with wages (top plot) or log wages (bottom plot) and potential years of experience (x axis) with median regression line (red) and least-squares regression line (blue).

There seems to be a curvilinear relationship between potential years of experience and wages, which is only perceptible in the log wages plot. A linear fit is clearly inappropriate.

2.10.3 The Conditional Distribution Story

Unconditional Distribution of Wages and Log Wages

Let's compare the weekly wages to a normal distribution with equal mean and standard deviation:

```
set.seed(23)
```

```

norm <- rnorm(100000, mean(CPS1988$wage), sd(CPS1988$wage))
png(filename = "wageqq.png", width = 3000, height = 3000, res = 300)
par(cex = 1.5)
wages.q <- quantile(CPS1988$wage, probs = seq(.01,.99, .01))
normal.q <- quantile(norm, probs = seq(.01, .99, .01))
min <- min(floor(min(wages.q, normal.q)))
max <- max(ceiling(max(wages.q, normal.q)))
plot(normal.q, wages.q,
      xlim = c(min, max),
      ylim = c(min, max),
      ylab = "Quantiles of Weekly Wages",
      xlab = "Quantiles of Draws from a Normal Distribution",
      pch = 19)
abline(0, 1)
dev.off()

```

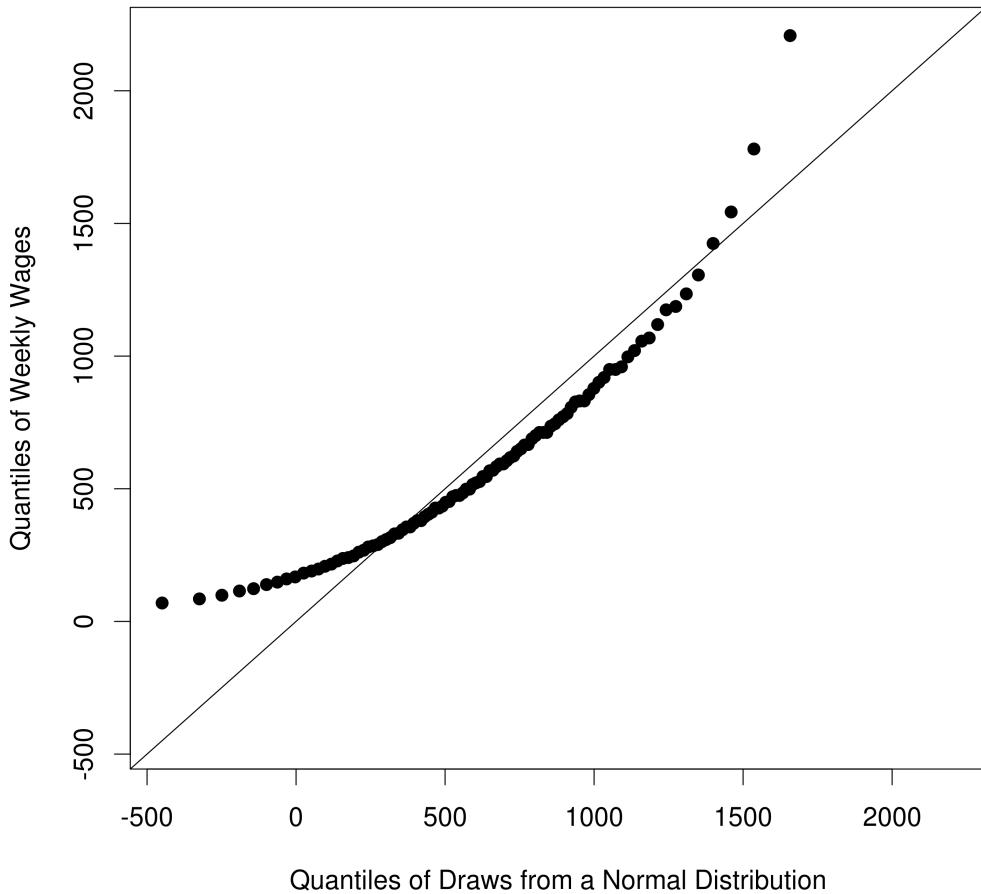


Figure 2.46: Plot of quantiles from a simulated normal distribution and weekly wages.

The qqplot shows a classic positively skewed distribution. Because wages cannot be negative, but the normal distribution can, the lowest quantiles of the wage distribution are much more compressed than the normal distribution. We also see a long right tail, where the highest quantiles of wages are much higher than what we'd expect from the normal distribution. Thus, we have a compressed left tail and heavy right tail.

Next, let's compare the log-transformed weekly wages to a normal distribution with equal mean and standard deviation:

```
set.seed(24)
norm <- rnorm(100000, mean(log(CPS1988$wage)), sd(log(CPS1988$wage)))
png(filename = "logwageqq.png", width = 3000, height = 3000, res = 300)
```

```
par(cex = 1.5)
wages.q <- quantile(log(CPS1988$wage), probs = seq(.01,.99, .01))
normal.q <- quantile(norm, probs = seq(.01, .99, .01))
min <- min(floor(min(wages.q, normal.q)))
max <- max(ceiling(max(wages.q, normal.q)))
plot(normal.q, wages.q,
      xlim = c(min, max),
      ylim = c(min, max),
      ylab = "Quantiles of Log Weekly Wages",
      xlab = "Quantiles of Draws from a Normal Distribution",
      pch = 19)
abline(0, 1)
dev.off()
```

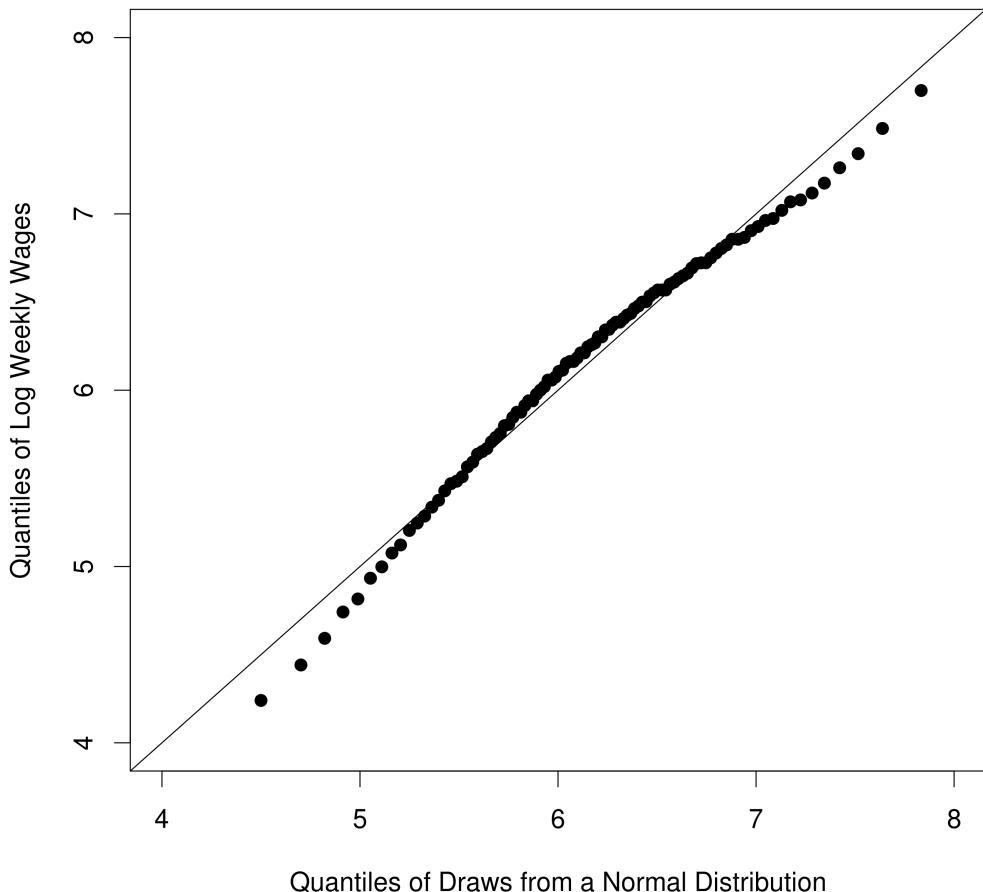


Figure 2.47: Plot of quantiles from a simulated normal distribution and log weekly wages.

The log transformed wages are much closer to the normal distribution, indicating that an unconditional distribution story might be that the distribution of wages is lognormally distributed. The lowest quantiles of the log wage distribution are more extreme than the normal distribution, and the highest quantiles are less extreme. There's also “bowing” in the middle, where the middle quantile of log weekly wages is larger than expected by the normal distribution.

Regression residuals

Next, let's conduct the linear regressions and plot the regression residuals:

```
reg1.A.lm <- lm(log(wage)
```

```

~ethnicity + education + exp.zero + I(exp.zero^2),
data = CPS1988)

reg2.A.lm <- lm(log(wage)
~ ethnicity + education + exp.zero + I(exp.zero^2)
+I(exp.zero^3) + I(exp.zero^4) , data = CPS1988)

```

We can use the car package to plot the jackknife regression residuals against the $t n - k - 1$ df distribution (here k is the number of model parameters, including the intercept):

```

library(car)
png(filename = "qqreg1.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(1, 2), cex = 1.3, mar = c(5, 4, 2, 1))
qqPlot(reg1.A.lm,
       main = "Table 1.A Regression Residuals",
       id.n = 3,
       pch = 19,
       cex = .75,
       col = rgb(0, 0, 0, .1),
       ylab = "Jackknife Residuals")
abline(0, 1)
qqPlot(reg2.A.lm,
       main = "Table 2.A Regression Residuals",
       id.n = 3,
       pch = 19,
       cex = .75,
       col = rgb(0, 0, 0, .1),
       ylab = "Jackknife Residuals")
abline(0, 1)
dev.off()

```

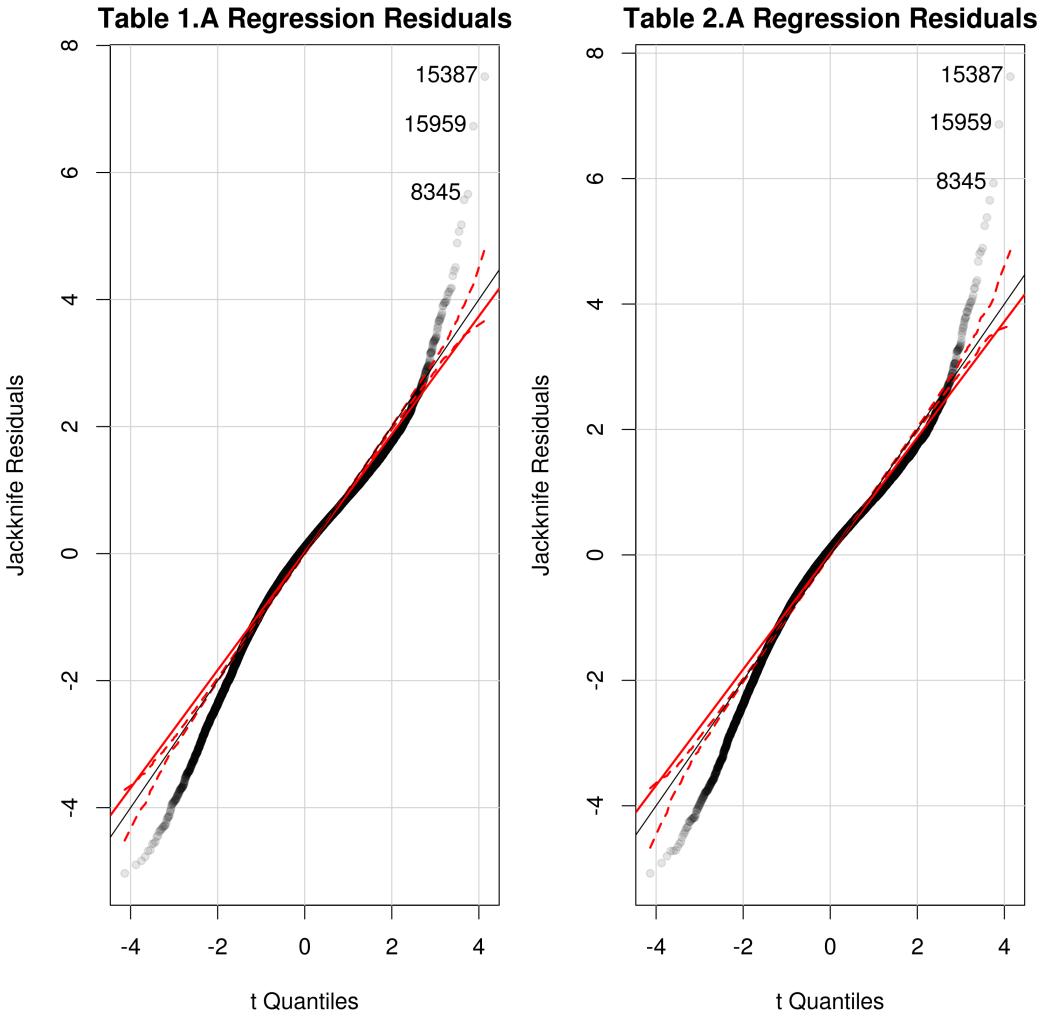


Figure 2.48: Q-Q plots of the jackknife regression residuals against the t-distribution for the Table 1.A regression (left) and Table 2.A regression (right).

We have a number of observations with large jackknife residuals greater than 4 in absolute value, and they fall outside the confidence bands of the t distribution. With almost 30,000 observations, the t distribution is essentially normal. The bowing that we observed in the unconditional distribution of log wages seems to be worse in the regression residuals, and adding in additional polynomial terms for experience doesn't seem to help.

To get an idea whether these are Case A or Case C (influential) outliers, let's take a look at the influence index plot

```
png(filename = "inffindexreg1A.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
influenceIndexPlot(reg1.A.lm, id.n = 10)
```

```
dev.off()
```

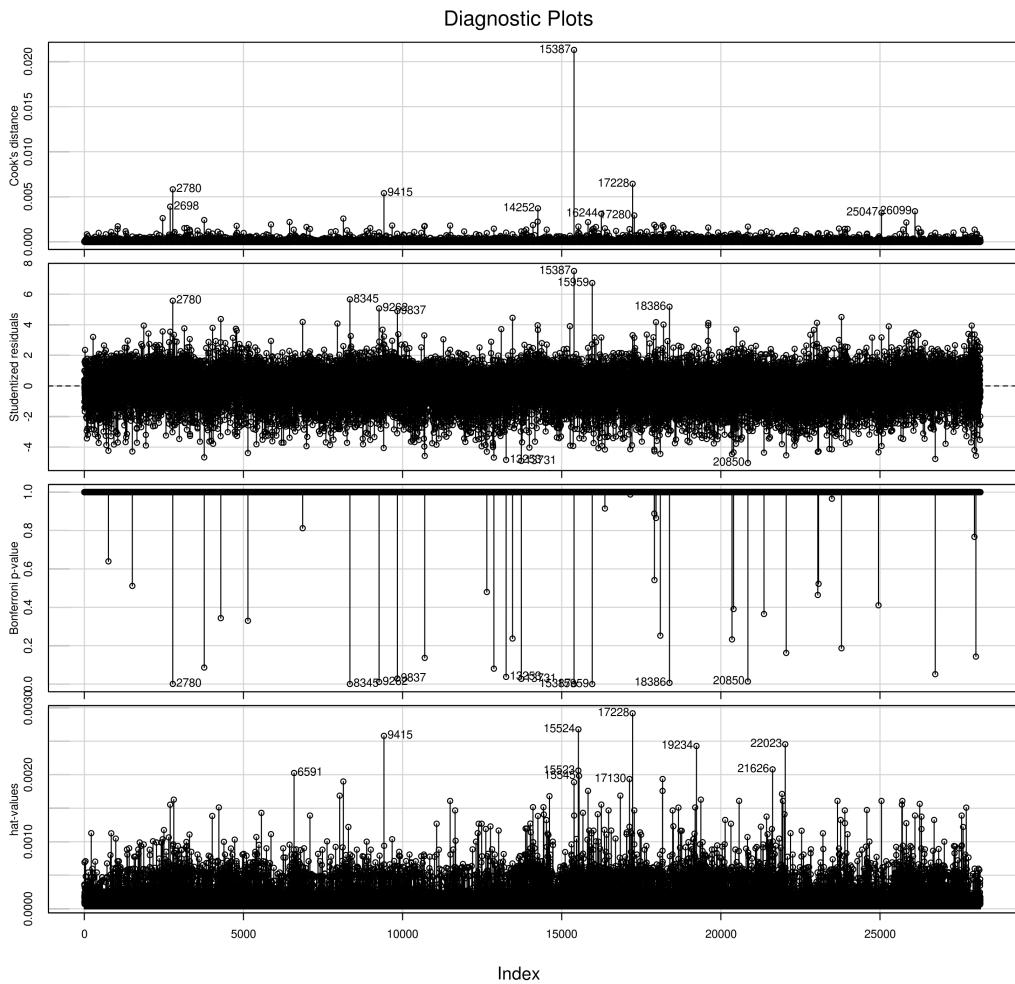


Figure 2.49: Plots of Cook's D (top row), jackknife residuals, Bonferroni p-values on the jackknife residuals, and hat values (bottom row) for the Mincer Table 1.A model.

The median Cook's D is 0.0000089:

```
median(cooks.distance(reg1.A.lm))
```

The maximum is 0.021, indicating that this one observation has an influence that is over 2000 times the median influence!

```
max(cooks.distance(reg1.A.lm))
```

```
max(cooks.distance(reg1.A.lm))/median(cooks.distance(reg1.A.lm))
```

From the influence index plots, we can see that this is observation 15387. Let's look into this observation more closely:

```
influence(reg1.A.lm)$hat[15387]
cooks.distance(reg1.A.lm)[15387]
rstudent(reg1.A.lm)[15387]
```

The influence is greater than the median, but not that much higher. Observation 17228 has the largest leverage, and second largest influence, but its Cook's D is still less than half that of 15387. The real issue is that this observation is so poorly fit by the model. Let's look at the observation's information:

```
CPS1988[15387, ]
```

This person has very high potential experience (59 years), extremely low education (3 years), and a *weekly* wage of almost \$8,000! Being from the south, this must be some oil baron or something. We have a similar problem with the Table 2.A regression, but the problem seems to be even worse, and 17228 also seems to be a hugely influential observation:

```
CPS1988[17228, ]
```

This observation has high leverage because the person has the minimum education (none), and the maximum potential experience (63 years). In contrast to 15387, this person only makes \$370 per week! I find these two people really interesting, and I'm glad we had a chance to tell their story (whatever we know about it). Median regression might be a better choice than mean regression here, lest we want one or two observations counting for more than thousands of others. Both seem to be Case C outliers, the kind we worry the most about, although there are a bunch of Case A outliers, with very high jackknife residuals (> 6).

2.10.4 The Forecasting Story

Backcasting

First, let's backcast the Mincer model:

```
png(filename = "mincerback.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(predict(reg1.A.lm), log(CPS1988$wage),
      xlab = "Predicted Log Wages",
      ylab = "Actual Log Wages",
      xlim = c(3, 10),
```

```

ylim = c(3, 10),
pch  = 19,
col  = rgb(0, 0, 0, .1),
cex  = 0.5
abline(0, 1)
dev.off()

```

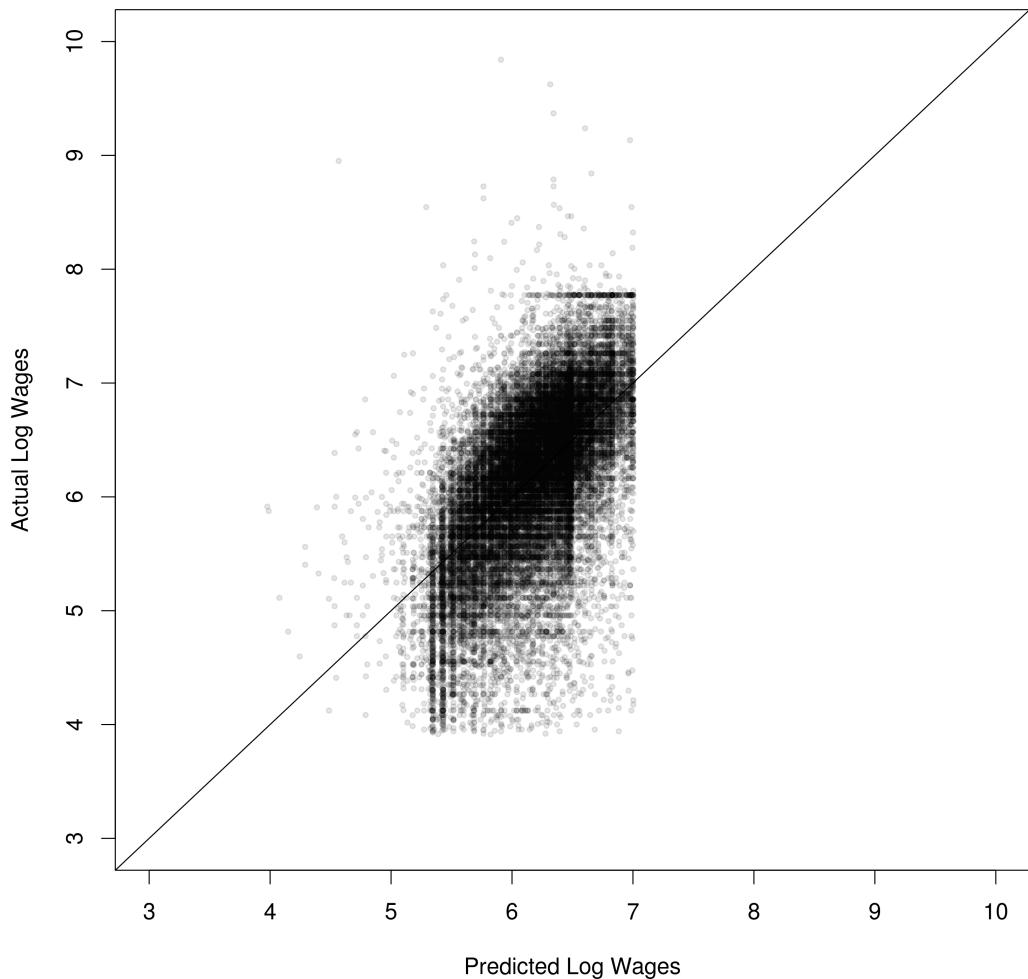


Figure 2.50: Actual versus predicted log wages from the Mincer type model.

The model does a fair job of backcasting, but also exhibits some interesting patterns. The actual log wages frequently exceed values of seven, but the model never makes predictions greater than seven. We can see that “bump” of wages from the histogram of the unconditional distribution of log wages, and this “bump” is totally unpredicted by our model, indicating, as far as our

current stories go, we have no idea what's going on there. The actual log wages also frequently take values well below five, but again our model is being conservative and only feels comfortable making predictions in the 5 to 7 range.

Cross-Validation

We've already done a couple models where it looks like adding a squared term helps. Let's use our forecasting story to see whether this is the case:

```
complex<-c()
simple<-c()

for(i in 1:100){
  train <- sample(rownames(CPS1988),
    2*length(rownames(CPS1988))/3,
    replace = FALSE)
  test <- rownames(CPS1988[!(rownames(CPS1988) %in% train), ])

  train1 <- rq(log(wage) ~ education + ethnicity + exp.zero + I(exp.zero^2),
    data = CPS1988[rownames(CPS1988) %in% train, ])
  # Remove the squared term
  train2 <- rq(log(wage) ~ education + ethnicity + exp.zero,
    data = CPS1988[rownames(CPS1988) %in% train, ])

  test1 <- (log(CPS1988$wage[rownames(CPS1988) %in% test])
    - predict(train1, CPS1988[rownames(CPS1988) %in% test, ]))^2

  test2 <- (log(CPS1988$wage[rownames(CPS1988) %in% test])
    - predict(train2, CPS1988[rownames(CPS1988) %in% test, ]))^2

  rMSEtest1 <- sqrt(sum(test1)/length(test1))
  rMSEtest2 <- sqrt(sum(test2)/length(test2))
  complex <- append(complex, rMSEtest1)
  simple <- append(simple, rMSEtest2)
}
```

Summarizing the results, the more complex model with the additional quadratic term performs better in terms of cross-validation:

```
summary(complex)
summary(simple)
```

2.10.5 The Statistical Inference Story

The most important factor to consider when formulating our statistical inference story is how the 1988 CPS data were collected. Were they a random sample? How was the sample collected? As mentioned in the Bierens article, only men ages 18-70 with annual income greater than \$50 who are not self-employed or working without pay are included.

As mentioned in the Buchinsky paper ([Buchinsky, 1994](#)), the sample frame includes the entire civilian population in the US in housing. That's pretty good coverage. As mentioned all black and white males were sampled in that age range, accounting for 96% of the sample. They must have worked at least one week in the year prior the March CPS year. There were also some other "noted" exclusion restrictions.

There's also some issues with measurement error in the actual calculation of the variables used. It seems like the actual number of weeks worked in the last year is not always known. In his paper Buchinsky models weeks worked based on race age and education. Measurement error will usually reduce the slope of regression lines. There is also the "top coding" issue, where the earnings of a person are reported as some maximum value even if their true earnings exceed that value to preserve the "confidentiality" of respondents (i.e., everyone knows who the billionaires are). Buchinsky just drops the observations that are at this maximum value.

There's no mention of non-response (i.e., those people who refused to take the survey), nor is there a mention of whether the survey was a random sample. Looking at the CPS wiki, it uses stratified random sampling.

Most people in this sample are white, so we should probably stick to white people when generalizing any of our sample estimates to the population, as with generalizing to part-time people. The results also only apply to males.

We also have one cross-sectional time period. Do these parameter estimates from the 1988 CPS apply to any other time period? It seems like the returns to things like education and experience would change over time, subject to changes in labor market characteristics etc. Thus, as far as statistical inference goes, where we are trying to infer the parameter value of a population from a sample, these results probably only apply to cross-sections around the 1988 time period. However, qualitatively, it's hard to imagine that there's no returns to education, but this is not a statistical argument, instead a causal/theoretical one.

In terms of the specific parameter estimates, for most of them we are saying that we judge the population parameter to be within the confidence interval, accepting a 5% error rate. Anything outside that confidence interval we reject as a plausible value of the population parameter. We could look at the coverage probabilities of our confidence intervals too, using bootstrap methods.

2.10.6 The Causality Story

Recall from our discussion of the causality story that the only way we can assure that our counterfactuals are correct is to randomly assign, taking into account issues like blinding, sample selection, attrition, etc. Was there random assignment? No. Was there blinding? No. Thus, we can't make any causal claims based on these data, because there's a host of omitted variables that could account for the observed relationships. For example, unmeasured ability or intelligence could drive education and experience effects: Those who are smarter get more education and get more work experience. Because proper random assignment wasn't used, we should give up on saying that these data support causal claims. We can recognize, however, that they are consistent with our causal theories. It's important to separate in our minds consistency with causal theories from evidence of causality.

Chapter 3

Multiple Linear Regression and Extensions

As seen in the previous chapter, the most popular way to model the conditional distribution of some variable of interest is to use a line, or linear regression. The defining feature of linear regression is the restriction that we must fit a single line (or hyperplane) that is a linear function of input variables, choosing the parameters of the linear function that minimize the sum of squared residuals. [Wooldridge \(2009\)](#) gives the following example of a linear population regression (i.e., a “true” linear function in the population):

$$wage = \beta_0 + \beta_1 educ + \beta_2 exper + u \quad (3.1)$$

So the conditional distribution of someone’s wages are a linear function of education and experience, with unobserved factors captured in the error term u .

We’ve already explored some features of linear regression in the previous chapter, including the concepts of leverage, discrepancy, and influence. In this chapter we’ll do more to link theoretical concepts with actual data analysis within the linear regression framework and some useful extensions.

3.1 Gauss-Markov Assumptions

Linear least squares regression has some very desirable properties if some assumptions are met, assumptions that probably never hold in the real world. If they did, linear least squares is basically the best you could do to describe the conditional mean of the dependent variable as a function of a set of inputs. Most importantly, linear least squares is *unbiased* under these assumptions, meaning it correctly specifies the relationship between the dependent and independent variables in terms of their conditional means.

These assumptions, called the *Gauss-Markov Assumptions* are:

1. Linearity in Parameters
2. Random Sampling
3. Zero Conditional Mean
4. No Perfect Collinearity

3.1.1 Linear in Parameters

The first assumption is that the population regression function is actually linear in parameters. That is, in the real world there is some mechanism that generates an outcome variable that is a linear function of the independent variables. Here, linear means that the parameters of the model are added together, rather than multiplied or divided, or some other operation. For example, the following regression would be non-linear in parameters:

$$wage = \beta_0 + \frac{\beta_1 educ}{1 + \beta_2 exper^{\beta_1}} + u \quad (3.2)$$

We can make any non-linear transformations we want to the input variables and the regression will still be linear in *parameters*. The following is still a linear regression model:

$$wage = \beta_0 + \beta_1 educ + \beta_2 exper + \beta_3 exper^2 + u \quad (3.3)$$

Why would such a function ever exist in the real world? It's hard to believe that nature would ever provide us with a nice linear function to discover, but this isn't as big a practical problem as it seems. Because we can transform the predictors and outcome variable, the linear function is very flexible. Meeting the linear in parameters assumption then depends on our ability to figure out which of the many predictor variables should be non-linearly transformed.

Sometimes we have a strong theoretical model to limit the possible transformations. For example, in psychophysics Fechner's Law relates the energy of a physical stimulus X to the perceived magnitude of the stimulus Y (Cohen et al., 2002):

$$c^Y = dX \quad (3.4)$$

Take a moment to think about whether this is linear in parameters. Can we solve for c and d using a linear function? Similarly, Steven's Law provides a different functional form for the same relationship:

$$Y = cX^d \quad (3.5)$$

Again, take a moment to think about whether this is linear in parameters. Is it possible to solve for c and d ? A third example is an exponential

relationship, for example relating number of statistics facts retained as a function of time, where:

$$Y = ce^{dX} \quad (3.6)$$

Another model is a hyperbolic relationship, or a diminishing returns model:

$$Y = \frac{X}{c + dX} \quad (3.7)$$

3.1.2 Random Sampling

The second assumption is that we have a random sample. This is a critical assumption because anything we want to say about the *population* regression function from our *sample* regression function depends entirely on how the sample was collected.

How often do we actually get truly random samples? Unless the sample is random by design, or some very compelling evidence can be put forth proving that a sample collected in a non-random way is actually random, linear least squares is hopeless. However, this isn't really a knock against least squares: Pretty much *any* estimation approach will fail if random sampling fails.

3.1.3 Zero Conditional Mean

Probably the second most important Gauss-Markov assumption (aside from random sampling) is the zero conditional mean of the errors assumption:

$$E(u|x) = 0, \forall x \quad (3.8)$$

This says that anything that is not included in our regression model that is associated with the outcome variable is also uncorrelated with the predictor variables (x 's) we've included in the model. Because humans naturally want to interpret relationships in causal terms, it's not surprising that statisticians and econometricians have spent a lot of time thinking about how linear regression fails to provide this interpretation, why, and to what degree. Consider the previous wage equation:

$$wage = \beta_0 + \beta_1 educ + \beta_2 exper + u \quad (3.9)$$

Suppose our natural inclinations kick in, and we want to interpret β_1 in causal terms. However, we've unfortunately omitted experience from our regression model. If experience and education are correlated at all, then failing to include experience in the regression will bias the estimated effect of education on wages. In other words, those with higher education also have more work experience, meaning much of the effect of work experience on wages is misattributed to education.

This is the *omitted variables problem*, where anything that is not included in our model that is a common cause of both the dependent variable and an input variable will bias our estimate of the input variable. The magnitude of this bias is the product of the correlation between the omitted variable and the input variable we are interested in δ , and the correlation between the omitted variable and the dependent variable γ :

$$\text{Omitted Variable Bias} = \delta \times \gamma \quad (3.10)$$

Omitted variables are factors contained in the error term, u . As discussed previously, there is every chance that *some* unobserved factors will have non-zero δ and γ , unless we explicitly make them zero. Random assignment makes $\delta = 0$: If people are randomly assigned different education levels (and assuming no placebo effects, hawthorne effects, volunteer bias, attrition, etc), then having more or less education must be uncorrelated with the person's experience (at least in the short run). This guarantees $\delta = 0$, and is the only case where this is guaranteed. More generally, random assignment means that anything in u is uncorrelated with the variable that is being randomly assigned, or:

$$E(u|education) = 0 \quad (3.11)$$

Here, the expected value of the error term is the same for all levels of education, meaning any other factor that might affect the person's wages are randomly distributed across levels of education. It is easy to think of a counterexample, such as those with higher intelligence getting more education and also earning higher wages.

As with random sampling, omitted variables will affect any estimation method, so linear least squares isn't much at fault here. However, $E(u|\text{all regressors}) \neq 0$ if we get the functional form of the regression wrong. For example, if we include income when we really needed to include squared income, $E(u|\text{all regressors}) \neq 0$. Thus, an approach that is able to specify the functional forms will be better than linear regression.

3.1.4 No Perfect Collinearity

If two of any variables are exact linear functions of each other, then our regression is poorly specified. We have to drop one of them in our regression. If you ever run a regression and the software package drops a variable, or outputs NA for its estimate, you're doing it wrong, and need to rethink your model. You've basically included the same predictor twice (e.g., $y = \beta_1x + \beta_2x$).

3.1.5 Unbiased Estimator

If the Gauss-Markov assumptions hold, we can prove that ordinary linear least squares gives unbiased estimates, meaning the estimators from the regression $\hat{\beta}$ match the population parameters *on average*. That is, $E(\hat{\beta}) = \beta$. In any specific sample, however, $\hat{\beta}$ will differ from β , due to sampling variability. Consider a population linear regression with one independent variable:

$$y_i = \beta_0 + \beta_1 x_{i1} + u_i \quad (3.12)$$

And our sample regression:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} \quad (3.13)$$

To derive our estimators for $\hat{\beta}$, we will use the fact that linear least squares regression minimizes the sum of squared residuals:

$$\operatorname{argmin}_{\hat{\beta}_0, \hat{\beta}_1} \left(\sum_{i=1}^n \hat{u}_i^2 \right) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1})^2 \quad (3.14)$$

To find the minimum, take the derivatives with respect to $\hat{\beta}_0$ and $\hat{\beta}_1$, set them equal to zero, and solve:

$$\frac{\partial \sum_{i=1}^n \hat{u}_i^2}{\partial \hat{\beta}_0} = 2 \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1}) = 0 \quad (3.15)$$

$$\frac{\partial \sum_{i=1}^n \hat{u}_i^2}{\partial \hat{\beta}_1} = 2 \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1}) x_{i1} = 0 \quad (3.16)$$

These are called the *normal equations*. If we solve them (proof is in the mathematical appendix at the end of the chapter, for those interested), we get our unbiased sample estimates of the intercept and slope of the population regression:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}_1 \quad (3.17)$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1}) y_i}{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2} \quad (3.18)$$

We've shown that if we have a linear function in the real world, and estimate it with a sample version of the same exact linear function, we get the real population linear function back. So, then all we need to do is have someone tell us what the real linear function in the world is, and we can just repeat back the correct answer using ordinary least squares! I suppose what is novel is that the least squares minimization gives you the correct answer for the conditional mean, rather than using some other optimization procedure (e.g., minimizing the *absolute* deviations, which we saw in the last chapter gives us the conditional *median*).

3.2 Examples and Interpretation

Let's go through some linear least squares regression examples to focus on how to interpret the estimates and plot the results. We'll begin with data from [Gelman and Hill \(2007\)](#) Chapter 3 looking at cognitive test scores of three and four year old children. Here we look at the child's test score and its relationship to the mother's high school education, IQ, labor force participation, and age:

```
# Install package from CMU repository
install.packages("arm", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(arm)
library(foreign)
child <- read.dta(
  "http://www.stat.columbia.edu/~gelman/arm/examples/child.iq/kidiq.dta")
```

3.2.1 Dummy Coded Variables, Factors, and Intercepts

First, let's look at the relationship between the child's IQ scores and whether the mother finished high school. Mother's high school completion is "dummy coded" as 1 if she graduated from high school, and 0 if not. We can run the regression with this dummy coded variable, or convert it to a factor:

```
# Convert mom_hs into a factor with appropriate labels
child$mom_hs_fac <- factor(child$mom_hs,
  levels = c(0, 1),
  labels = c("Did Not Graduate HS", "Graduated HS"))
```

Make sure you've got the labels and levels aligned correctly, otherwise the factor will be the reverse of the dummy code (unless you want that for some reason). The dummy coded regression and factor regression will give the same results:

```
childreg1 <- lm(kid_score ~ mom_hs, data = child)
childreg2 <- lm(kid_score ~ mom_hs_fac, data = child)
summary(childreg1)
summary(childreg2)
```

Let's interpret the model. The estimated equation is:

$$\text{Score}_i = 77.55 + 11.77 \times \text{mom.hs}_i \quad (3.19)$$

So, if the mother graduated from high school, then $\text{mom.hs} = 1$ and the average (conditional mean) cognitive test score is 89.32 ($= 77.55 + 11.77 \times 1$).

If the mother did not graduate from high school, then mom_hs = 0 and the average (conditional mean) cognitive test score is 77.55 ($= 77.55 + 11.77 \times 0$). Thus, the coefficient on mom_hs is the *difference* between the average test scores of children whose mother graduated from high school and those whose mother did not. The regression line isn't that interesting, but we can show the scatterplots and regression line:

```
png(filename = "momhs.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 1, 1))
plot(jitter(child$mom_hs, amount = .05),
     jitter(child$kid_score),
     pch = 19,
     col = rgb(0, 0, 0, .5),
     xlab = "Mother's High School (HS) Completion",
     ylab = "Child's Test Score",
     xaxt = "n", # Turn off default x axis
     yaxt = "n", # Turn off default y axis
     xlim = c(-.3, 1.3),
     ylim = c(0, 160))
axis(side = 1,
     at = c(0, 1),
     labels = c("Did Not Graduate HS", "Graduated HS"))
axis(side = 2, at = seq(0, 160, by = 20))
abline(childreg1)
dev.off()
```

We can see that there are fewer children with mothers who did not graduate high school, and their test scores seem to have two potential outliers. We can look at the jackknife residuals of the regression to check this further:

```
library(car)
png(filename = "momhsqq.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
qqPlot(childreg1,
       pch = 19,
       col = rgb(0, 0, 0, .5),
       ylab = "Jackknife Residuals")
dev.off()
```

We could investigate the residuals more, but let's leave it for now, and press on with interpretation issues.

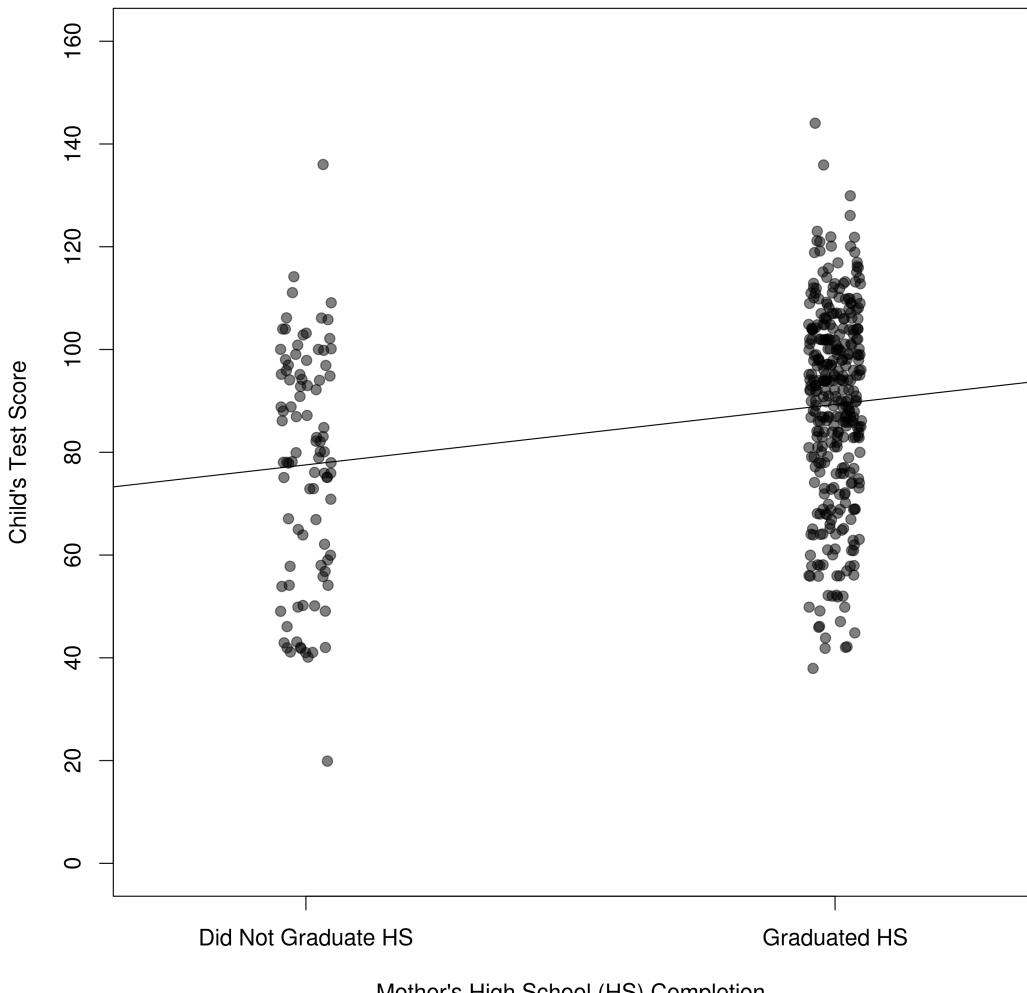


Figure 3.1: Child's test score by mother's high school completion.

3.2.2 Intercept and Continuous Predictor

We've already covered regression with a continuous predictor before, but let's reiterate. We'll add in the mother's IQ score to the previous regression:

```
childreg3 <- lm(kid_score ~ mom_hs + mom_iq, data = child)
summary(childreg3)
```

Let's interpret the model. The estimated equation is:

$$\text{Score}_i = 25.73 + 5.95 \times \text{mom_hs}_i + 0.56 \times \text{mom_iq}_i \quad (3.20)$$

The intercept of 25.73 has a nonsense interpretation now. It would be the average child's test score for a mother with an IQ of zero ($\text{mom_iq}_i = 0$) who

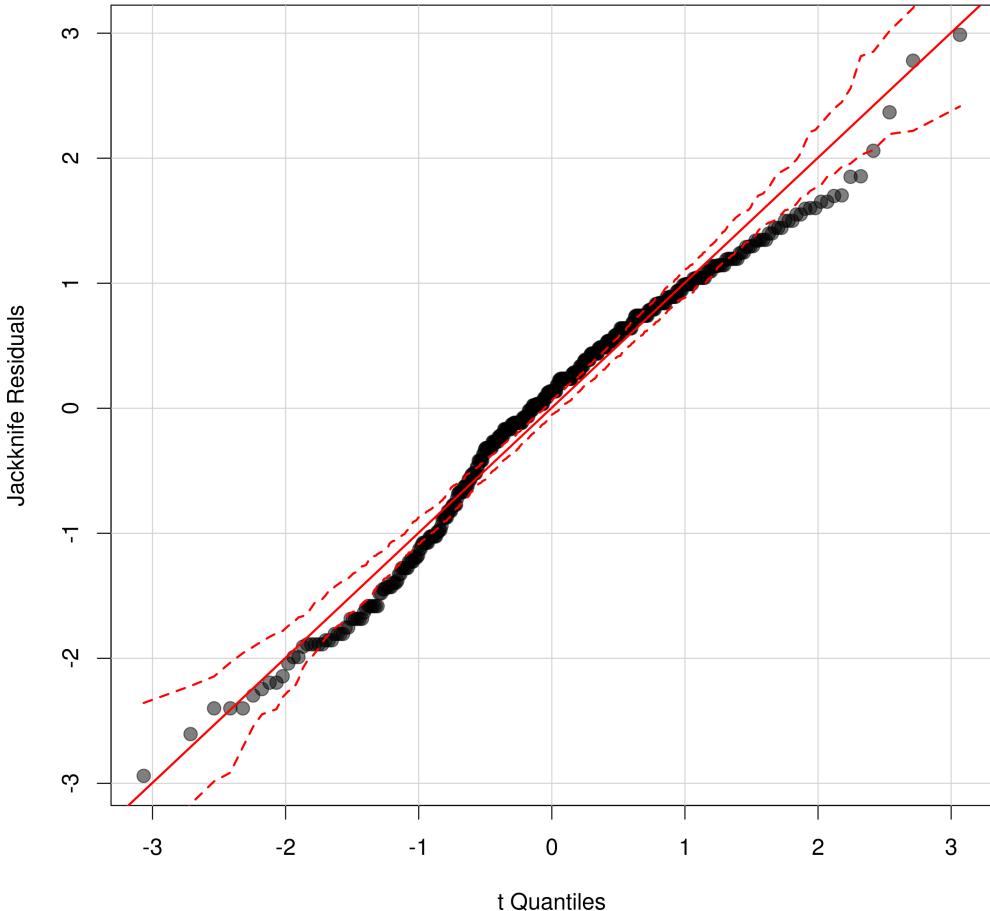


Figure 3.2: QQ plot of the jackknife regression residuals. Dashed red lines show 95% confidence bands for quantiles of the t distribution with $n - k - 1$ degrees of freedom.

did not graduate high school ($\text{mom_hs}_i = 0$). Obviously, it is not possible to have an IQ of zero, so the intercept is meaningless in this context.

The coefficient on mom_hs_i has the same interpretation as in the previous model although the magnitude of the difference between the averages is smaller: Children whose mothers graduate high school ($\text{mom_hs}_i = 1$) have a test score that is about 6 points higher ($= 5.95 \times 1$) than those whose mothers did not graduate high school ($\text{mom_hs}_i = 0$).

The coefficient on mom_iq_i tells us that a child whose mother has one additional IQ point ($\text{mom_iq}_i + 1$) has a test score that is 0.56 units higher than a child whose mother has one less IQ point. Let's plot the results:

```

# Subset to kids with mothers who did or did not graduate high school
no.hs <- child[child$mom_hs == 0, ]
hs <- child[child$mom_hs == 1, ]

png(filename = "momhsiq.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 1, 1))
# Make an empty plot with appropriate axes
plot(-100, -100, # Hack to create axes
      type = "n", # Suppress plotting any points
      ylab = "Child's Test Score",
      xlab = "Mother's IQ Score",
      xlim = c(60, 160),
      ylim = c(0, 160),
      yaxt = "n") # suppress y axis
axis(side = 2, at = seq(0, 160, by = 20))

# Plot the scores for kids with mothers who didn't graduate
points(no.hs$mom_iq,
       no.hs$kid_score,
       pch = 20,
       col = rgb(1, 0, 0, .5))

# Add regression line without the graduation intercept
abline(coef(childreg3)[1], # The first coefficient is the overall intercept
       coef(childreg3)[3], # The third is the mom_iq coefficient
       col = rgb(1, 0, 0, .5),
       lty = 2)

# Plot the scores for kids with mothers who graduated
points(hs$mom_iq, hs$kid_score,
       pch = 18,
       col = rgb(0, 0, 1, .2))
# Add in the intercept for mom_hs, which is coef(childreg3)[2]
abline(coef(childreg3)[1] + coef(childreg3)[2],
       coef(childreg3)[3],
       col = rgb(0, 0, 1, .5),
       lty = 2)
legend(130, 20,
       c("Graduated HS", "Did not graduate HS"),
       col = c("blue", "red"),
       pch = c(18, 20))
dev.off()

```

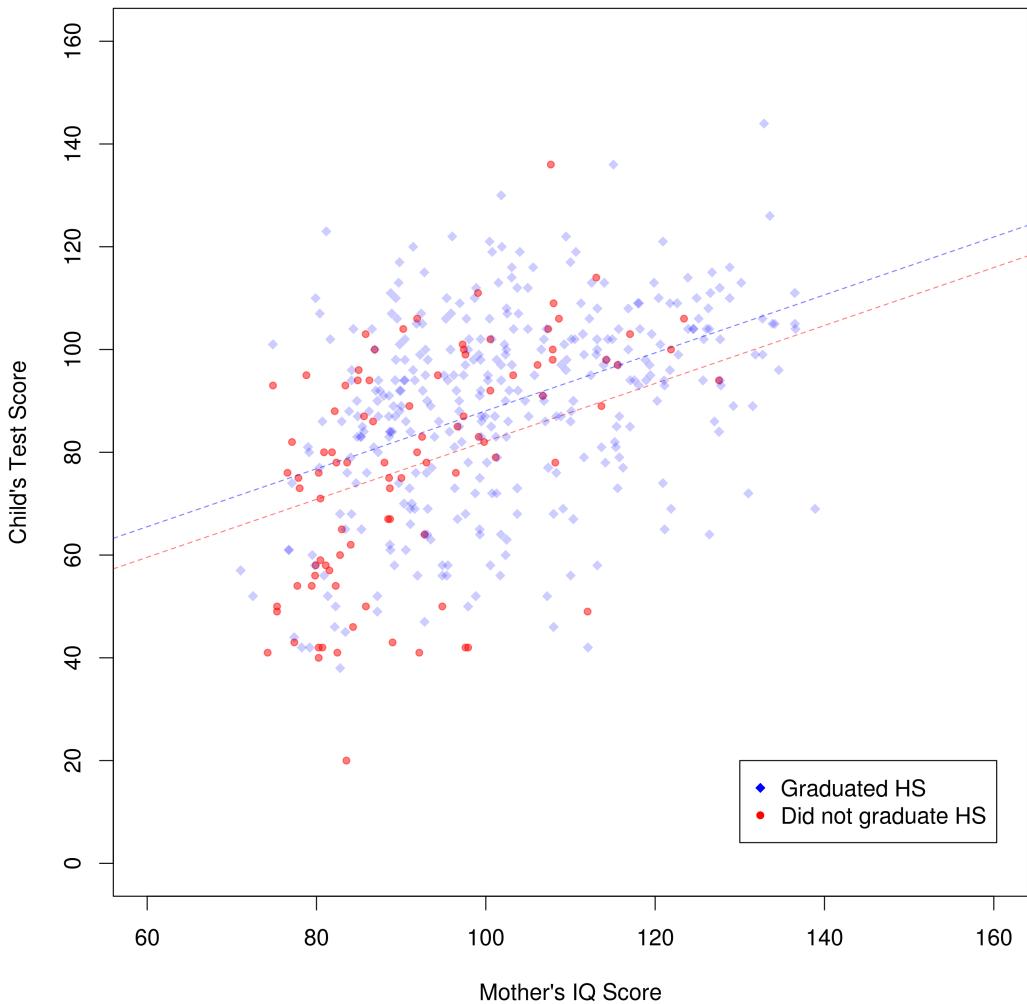


Figure 3.3: Child's test score by mother's IQ score, with different intercepts based on mother's high school graduation.

I did my best to keep the red points visible, showing the observations for those whose mother did not graduate from high school, and minimizing the visual impact of the regression lines by using a dashed line with some transparency. Still, it's really hard to see the difference between the observations for those who graduated versus did not graduate high school. Separate plots would probably work better:

```
png(filename = "momhsiq2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 1, 1), mfrow = c(2, 1))
plot(-100, -100,
      type = "n",
      ylab = "Child's Test Score",
```

```

xlab = "Mother's IQ Score",
xlim = c(60, 160),
ylim = c(0, 160),
yaxt = "n") # suppress y axis
axis(side = 2, at = seq(0, 160, by = 20))
points(no.hs$mom_iq,
       no.hs$kid_score,
       pch = 20,
       col = rgb(1, 0, 0, .5))
abline(coef(childreg3)[1], # The first coefficient is the overall intercept
       coef(childreg3)[3],
       col = "red") # The third is the mom_iq coefficient
# Add the line for the other group for comparison
abline(coef(childreg3)[1] + coef(childreg3)[2],
       coef(childreg3)[3],
       col = rgb(0, 0, 1, .7),
       lty = 2)
plot(-100, -100,
      type = "n",
      ylab = "Child's Test Score",
      xlab = "Mother's IQ Score",
      xlim = c(60, 160),
      ylim = c(0, 160),
      yaxt = "n") # suppress y axis
axis(side = 2, at = seq(0, 160, by = 20))
points(hs$mom_iq, hs$kid_score,
       pch = 18,
       col = rgb(0, 0, 1, .5))
abline(coef(childreg3)[1] + coef(childreg3)[2],
       coef(childreg3)[3],
       col = "blue")
abline(coef(childreg3)[1],
       coef(childreg3)[3],
       col = rgb(1, 0, 0, .7),
       lty = 2)
dev.off()

```

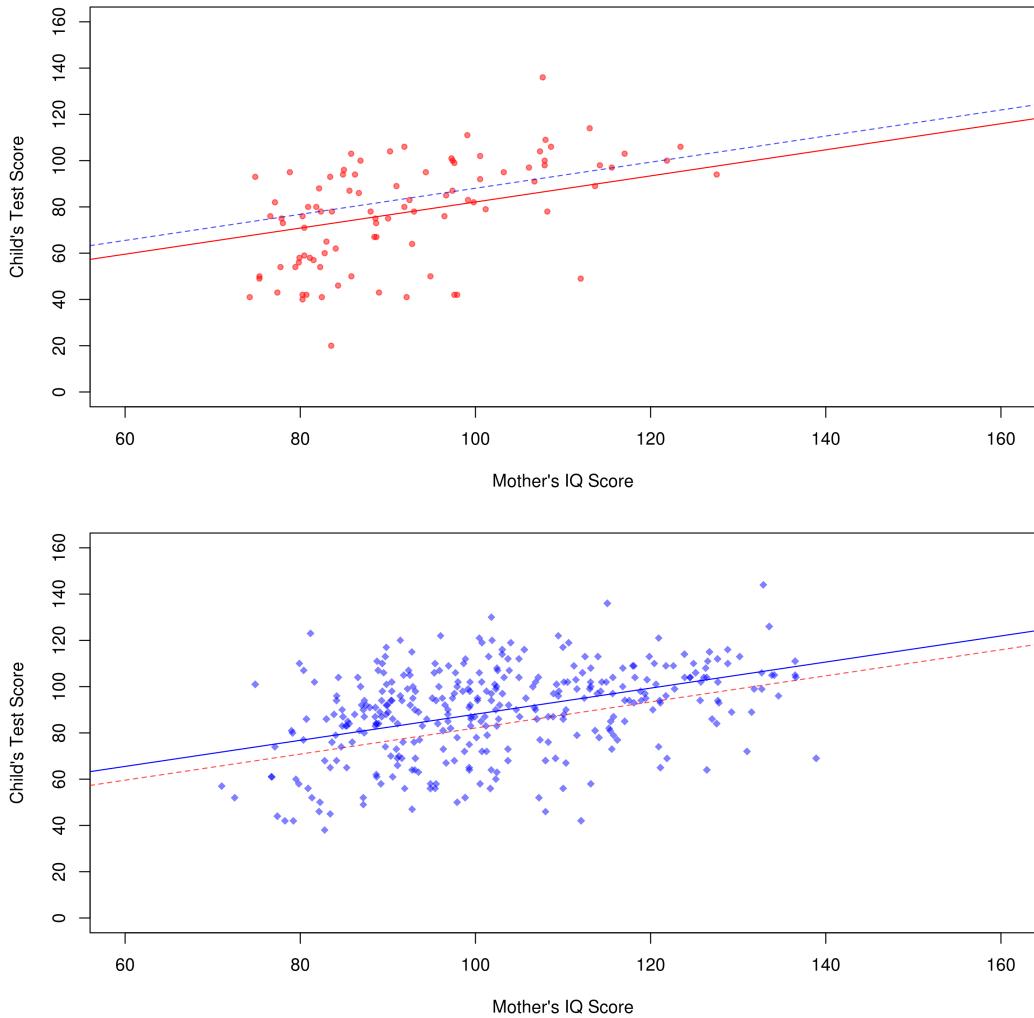


Figure 3.4: Child’s test score by mother’s IQ score, with different intercepts based on whether the mother did not graduate high school (top), or did graduate high school (bottom).

Now we can see the data and fitted lines better, and it’s easier to compare intercepts, although the plots are a bit confusing.

3.2.3 Interaction Between a Factor and Continuous Predictor

Lastly, we can allow for the slope of the childrens’ scores on their mothers’ IQs to be different, depending on whether the mother completed high school or not. This “different slopes for different folks” is an interaction between the dummy/factor variable and the continuous predictor:

```

childreg4 <- lm(kid_score ~
                  mom_hs + mom_iq + mom_hs * mom_iq, data = child)
summary(childreg4)

```

Let's interpret the model. The estimated equation is:

$$\text{Score}_i = -11.48 + 51.27 \times \text{mom_hs}_i + 0.97 \times \text{mom_iq}_i - 0.48 \times \text{mom_iq}_i \times \text{mom_hs}_i \quad (3.21)$$

The model allows different intercepts and slopes for children with mothers who completed high school and children whose mothers who did not complete high school. To find the slope and intercepts for the children whose mothers did not complete high school, we set $\text{mom_hs} = 0$ and get the following reduced equation:

$$\text{Score}_i = -11.48 + 0.97 \times \text{mom_iq}_i \quad (3.22)$$

The intercept is again nonsense, with a score of -11.48 on the test (probably impossible) for children whose mother's IQ is zero (definitely impossible). We can see that the slope is almost twice as high as in the previous regression, where children whose mothers have an IQ that is 1 point higher than another child score almost 1 point higher on the test (0.97×1).

To find the slope and intercepts for the children whose mothers *did* complete high school, we set $\text{mom_hs} = 1$ and get the following equation:

$$\text{Score}_i = -11.48 + 51.27 \times 1 + 0.97 \times \text{mom_iq}_i - 0.48 \times \text{mom_iq}_i \times 1 \quad (3.23)$$

We can collect the common terms, calculating the intercept for those whose mother's completed high school of 39.79 ($= 51.27 - 11.48$) and new slope for those whose mother's completed high school 0.49 ($= 0.97 - 0.48$). This gives us the following reduced equation:

$$\text{Score}_i = 39.79 + 0.49 \times \text{mom_iq}_i \quad (3.24)$$

The intercept says that a child whose mother has zero IQ would score a 39.79 on the test, which is again nonsense, but is much higher than the intercept for the regression for children whose mother did not complete high school. However, the relationship between mother's IQ and score is now much weaker, where children whose mothers have an IQ that is 1 point higher than another child score 0.49 points higher on the test, compared to almost twice that amount for children whose mothers did not complete high school.

Looking at the plot:

```

# Subset to kids with mothers who did or did not graduate high school
no.hs <- child[child$mom_hs == 0, ]
hs <- child[child$mom_hs == 1, ]

```

```
png(filename = "momhsiq3.png", width = 3000, height = 3000, res = 300)  
dev.off()
```

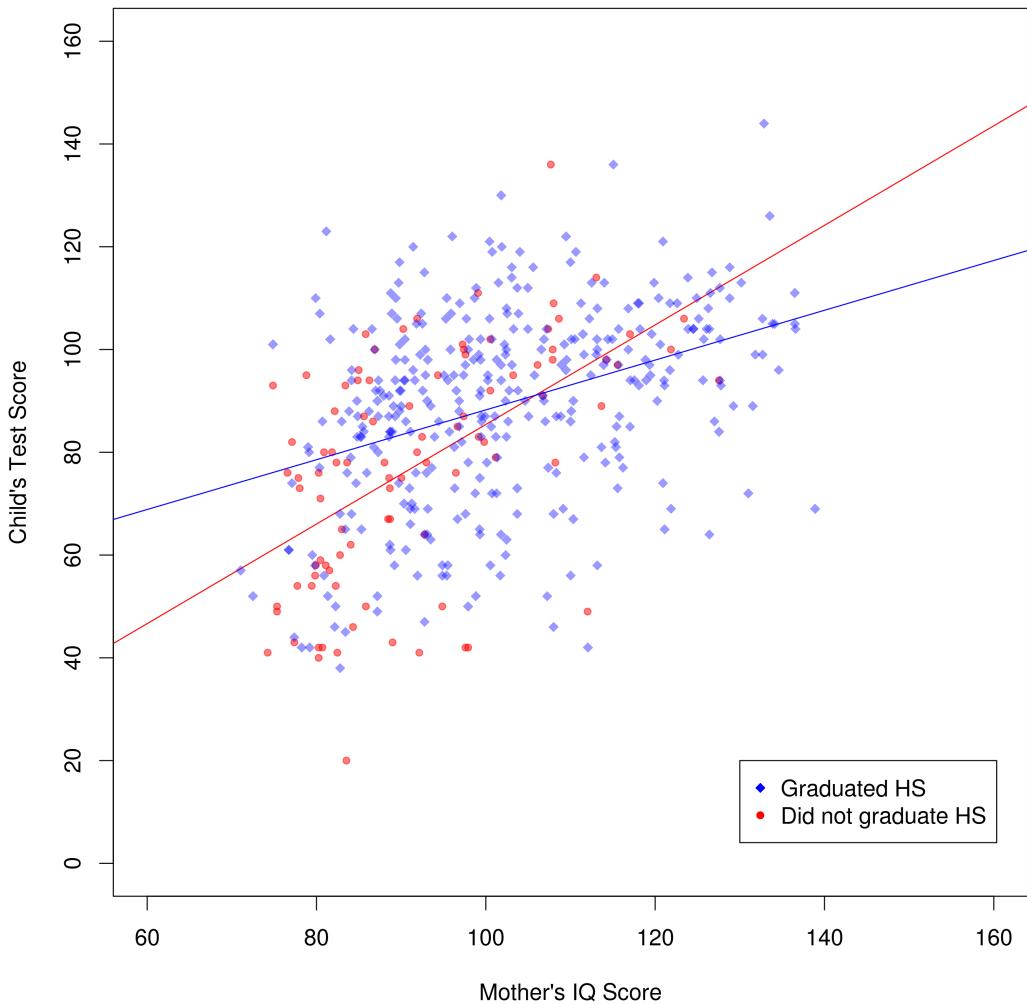


Figure 3.5: Child's test score by mother's IQ score, with different intercepts and slopes based on mother's high school graduation.

Again, checking out the separate plots:

```
png(filename = "momhsiq4.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 1, 1), mfrow = c(2, 1))
plot(-100, -100,
      type = "n",
      ylab = "Child's Test Score",
      xlab = "Mother's IQ Score",
      xlim = c(60, 160),
      ylim = c(0, 160),
      yaxt = "n") # suppress y axis
axis(side = 2, at = seq(0, 160, by = 20))
```

```

points(no.hs$mom_iq,
      no.hs$kid_score,
      pch = 20,
      col = rgb(1, 0, 0, .5))
abline(coef(childreg4)[1],
       coef(childreg4)[3],
       col = "red")
plot(-100, -100,
      type = "n",
      ylab = "Child's Test Score",
      xlab = "Mother's IQ Score",
      xlim = c(60, 160),
      ylim = c(0, 160),
      yaxt = "n") # suppress y axis
axis(side = 2, at = seq(0, 160, by = 20))
points(hs$mom_iq, hs$kid_score,
       pch = 18,
       col = rgb(0, 0, 1, .5))
abline(coef(childreg4)[1] + coef(childreg4)[2],
       coef(childreg4)[3] + coef(childreg4)[4],
       col = "blue")
dev.off()

```

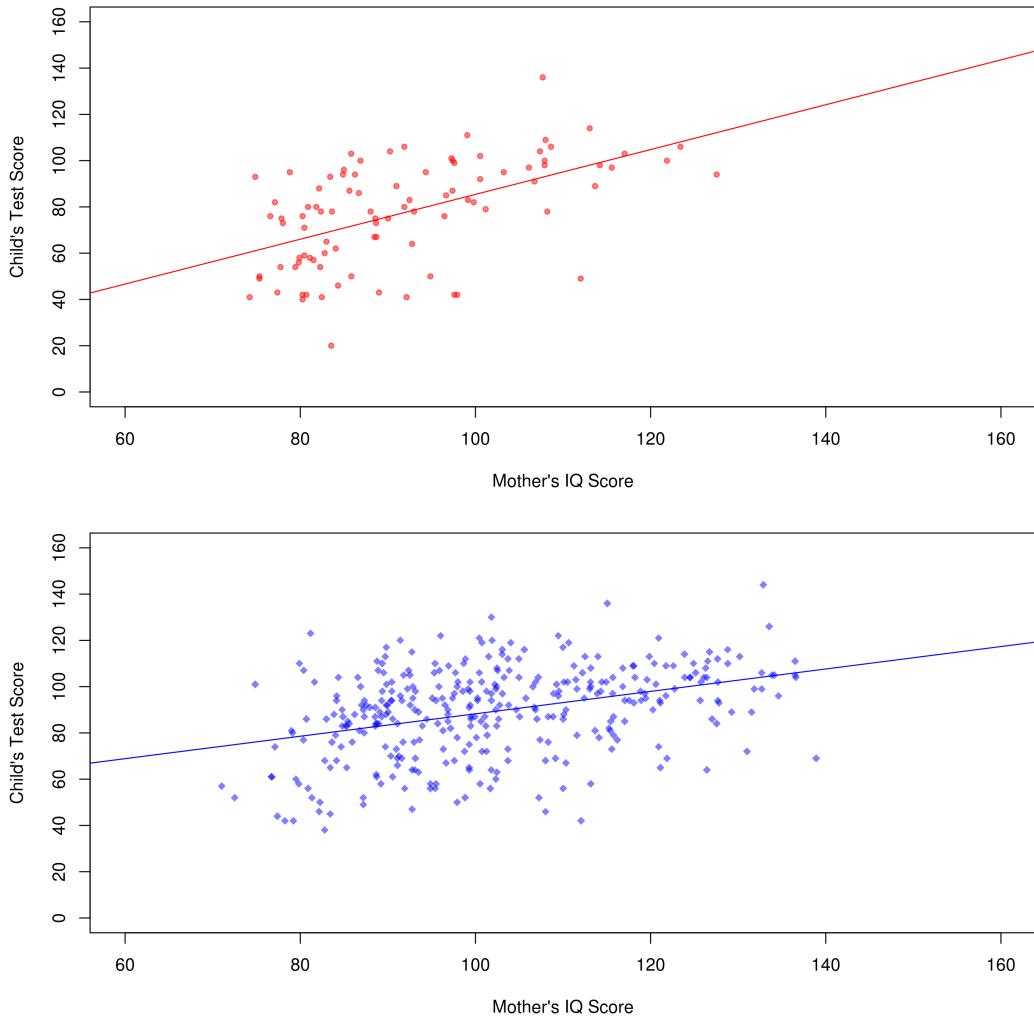


Figure 3.6: Child's test score by mother's IQ score, with different intercepts and slopes based on whether the mother did not graduate high school (top), or did graduate high school (bottom).

We can summarize these models in table form to give the reader a simple way of seeing the regression parameters, their standard errors, and other relevant information about the regression (e.g., sample size). It is important to include such a table in a published report, but this should be the *last* step in the model building process, after the models have been checked using each of the stories (and importantly, checked using graphical comparisons). It is also important that you know how to interpret each of the elements of the model, especially the coefficients.

As we can see, the table contains the independent variables in each row with the coefficients for each independent variable in the three regression

Dependent Variable: Child's Test Score			
Independent Variables	Model 1	Model 2	Model 3
Mother's HS Graduation	11.77 (2.32)	5.95 (2.21)	51.27 (15.34)
Mother's IQ		0.56 (0.06)	0.97 (0.15)
Mother's HS X IQ			-0.48 (0.16)
Intercept	77.55 (2.06)	25.73 (5.87)	-11.48 (13.76)
Observations	434	434	434
5-fold rMSE	19.91	18.19	18.20

Table 3.1: Summary table of three models of Child's Test Score based on mother's characteristics.

models in the columns. Below each estimated coefficient is the standard error in parentheses. At the bottom of the table we have summary characteristics of each model, including the sample size and 5-fold CV rMSE. Although the results strongly suggest adding Mother's IQ to the model, adding an interaction is questionable, with a significant p-value on the interaction, but greater cross-validated rMSE for the more complex model.

3.2.4 Standardizing Predictors for Interpretability

If we want a model with interpretable coefficients (e.g., without having to think about mothers with an IQ of zero) we can standardize the predictors by subtracting the mean of the predictor and dividing by the standard deviation. We saw this in the previous chapter, where the standard score (or z score) for mother's IQ is:

```
# To get the standard score, subtract the mean and divide by the sd
child$z.mom.iq <- (child$mom_iq - mean(child$mom_iq)) / sd(child$mom_iq)
childreg5 <- lm(kid_score ~ z.mom.iq, data = child)
summary(childreg5)
```

Let's interpret the model. The estimated equation is:

$$\text{Score}_i = 86.80 + 9.15 \times z.\text{mom}.iq \quad (3.25)$$

The intercept of 86.80 now has a meaningful interpretation as the child's test score when the mother's *standardized* score is zero, which is when the mother's IQ is at the mean of all the IQ scores in the sample. Thus, the intercept is the child's test score for a child whose mother has an average IQ.

The coefficient on `z.mom.iq` is a little more complex to understand. In this case, children whose mothers have an IQ score *1 standard deviation* above another child have an average test score that is 9.15 points higher. Thus, the test score is about 96 ($= 86.80 + 9.15$) for those children with mothers who have an IQ score that is one standard deviation above the mean. We get a similar effect by just subtracting the mean, and possibly an easier time at interpretation. For most communications it's probably better to just subtract the mean to avoid confusion, especially for laypeople who may have not heard of standardized scores.

3.3 Multiple Regression and Partial Residuals

As we're considering regression with multiple independent variables in this chapter, it's important to think about how these variables work simultaneously in a multiple regression model. Adding additional independent variables to a regression is often called *controlling* for other variables, for example examining how the conditional distribution of earned wages depends on education after controlling for work experience. Thinking of regression in this way is very misleading: We have *no* control over any of the variables in our regression without random assignment or some other real manipulation.

A much better way of thinking about multiple regression is by understanding one method of estimating a multiple regression function. This is the *partial residual* approach, and works in the following way. First, we take the variable we are interested in using as an independent variable, say education, and compute the regression of this variable on *all the other independent variables*:

$$\hat{educ} = \hat{\gamma}_0 + \hat{\gamma}_1 exper + \hat{v} \quad (3.26)$$

In our example, \hat{v} is the sample residuals of the regression of education on experience, or the variation in education that is *not* explained by experience. Thus, \hat{v} is the variability in education that is “left over” that could possibly be associated with wages but cannot be associated with experience.¹

Next, we do the same for wages, our dependent variable:

$$\hat{wages} = \hat{\delta}_0 + \hat{\delta}_1 exper + \hat{d} \quad (3.27)$$

In our example, \hat{d} is the sample residuals of the regression of wages on experience, or the variation in wages that is *not* explained by experience.

¹This assumes that our model of education is correctly specified (e.g., that we haven't omitted `exper`²), a point that we will return to.

Again, \hat{d} is the variability in wages that is “left over” that could possibly be associated with education but not experience.

Finally, we take these residuals and then compute the following simple regression:

$$\hat{d} = \hat{\beta}_0 + \hat{\beta}_1 \hat{v} \quad (3.28)$$

Here, $\hat{\beta}_1$ in this regression will be the same as $\hat{\beta}_1$ had we done multiple regression of wages on education and experience simultaneously. Thus, rather than thinking of multiple regression as “controlling” for other variables, it makes more sense to think of the regression coefficient on a variable of interest as the regression of the residuals of the dependent variable and that variable after regressing on all other variables. We can extend this to the case of more than two independent variables, by iteratively regressing the dependent variable and each independent variable on all other independent variables, then finding the parameter for the variables we are interested in.

3.3.1 Apartment Building Example

Let’s see the partial residual interpretation with some example data. This dataset includes the assessed value of an apartment building (in dollars) with various characteristics of the building (size of the lot in square feet, total area of the building in square feet, the number of apartments, the age of the building in years, etc.):

```
aptdata <- read.table("AptsData.txt",
  col.names = c("case", "value", "lotsize", "bldgarea",
  "numapts", "age", "cond"))
```

Let’s calculate the conditional mean apartment building value given lotsize and building area using linear regression:

```
valuereg <- lm(value ~ lotsize + bldgarea, data = aptdata)
options(scipen = 99, digits = 2) # suppress scientific notation
summary(-valuereg)
```

The estimated coefficient on lotsize is 4.97, and 18.27 for bldg area. If we believe the regression, this means that a building with one additional square foot of lot area is expected to have an assessed value that is \$5 greater than a building without that additional lot area. Similarly, buildings with one additional square foot of building area have an assessed value that is about \$18 greater than a building without that additional building area.

Now, let’s look at the conditional mean of value only as a function of lotsize alone:

```

valuereg2 <- lm(value ~ lotsize, data = aptdata)
summary(valuereg2)

```

The estimated coefficient on lotsize now corresponds to a \$32.57 increase in assessed value with an additional one square foot of lot area. Recall that this inflation reflects omitted variable bias; buildings built on larger lots are also larger themselves, so much of the effect of building *size* is now incorrectly captured in the lotsize estimate, because building size is omitted.

Let's use the partial residual approach to calculate the "correct" parameter. First, we calculate the regression of lotsize (the independent variable of interest) on building size (the other independent variable):

```

lotreg <- lm(lotsize ~ bldgarea, data = aptdata)
png(filename = "lotreg.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 1), cex = 1.3, mar = c(5, 4, 1, 1))
plot(aptdata$bldgarea, aptdata$lotsize,
      pch = 19,
      xlab = "Building Area (square feet)",
      ylab = "Lot Size (square feet)",
      ylim = c(0, 25000),
      xlim = c(0, 50000))
abline(lotreg)
hist(rstudent(lotreg),
      main = "",
      breaks = "FD", # Freedman-Diaconis binwidth selection
      xlim = c(-3, 3),
      xlab = "Jackknife Regression Residuals")
dev.off()

```

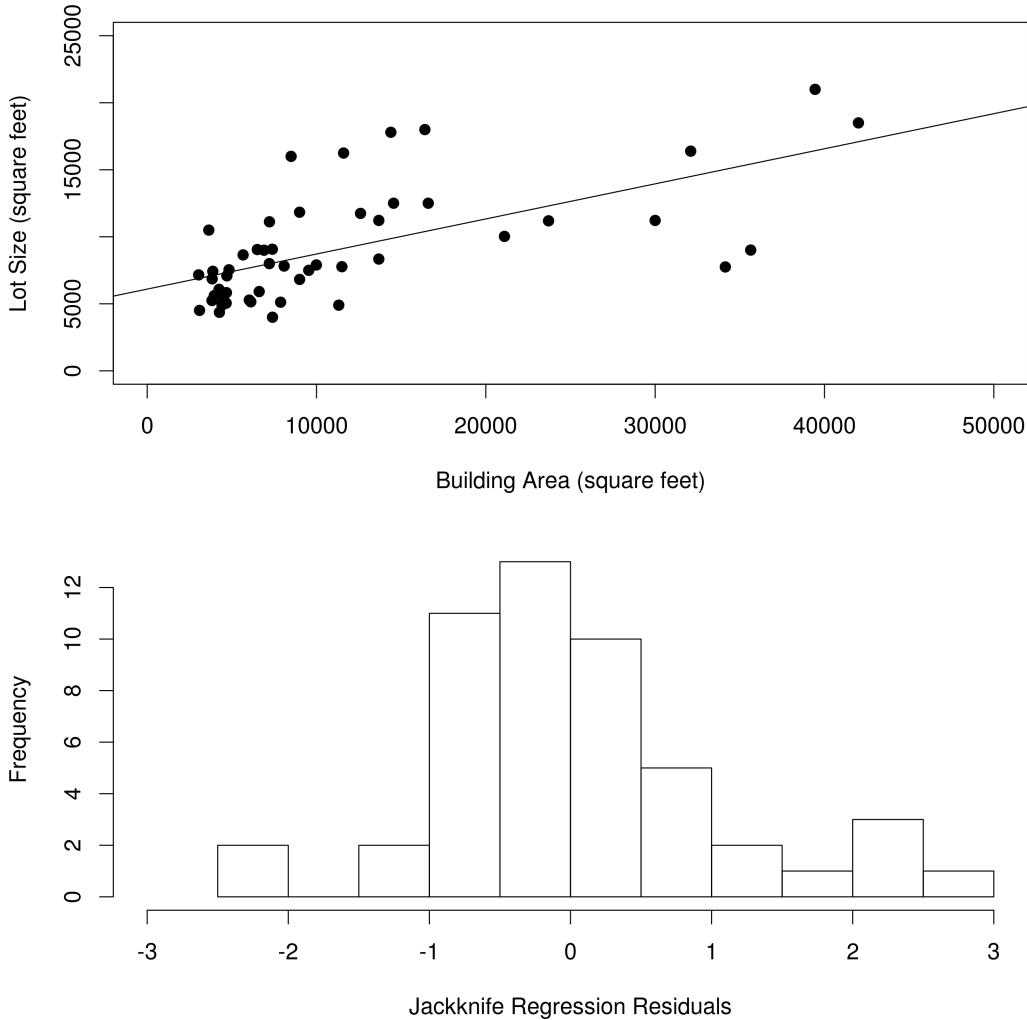


Figure 3.7: Relationship between building area and lot size.

As expected, building area and lotsize are strongly correlated. The scatterplot and residuals look a bit lumpy, which should probably cause us to think a bit more carefully about this model, but for now we'll continue with the partial residual demonstration. Next, let's calculate the regression of building value on building area:

```
valuereg3 <- lm(value ~ bldgarea, data = aptdata)
png(filename = "valuereg.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 1), cex = 1.3, mar = c(5, 4, 1, 1))
plot(aptdata$bldgarea, aptdata$value,
      pch = 19,
      xlab = "Building Area (square feet)",
      ylab = "Building Assessed Value ($)",
```

```

xlim = c(0, 50000),
ylim = c(0, 1000000))
abline(valuereg3)
hist(rstudent(valuereg3),
main = "",
xlab = "Jackknife Regression Residuals",
breaks = "FD")
dev.off()

```

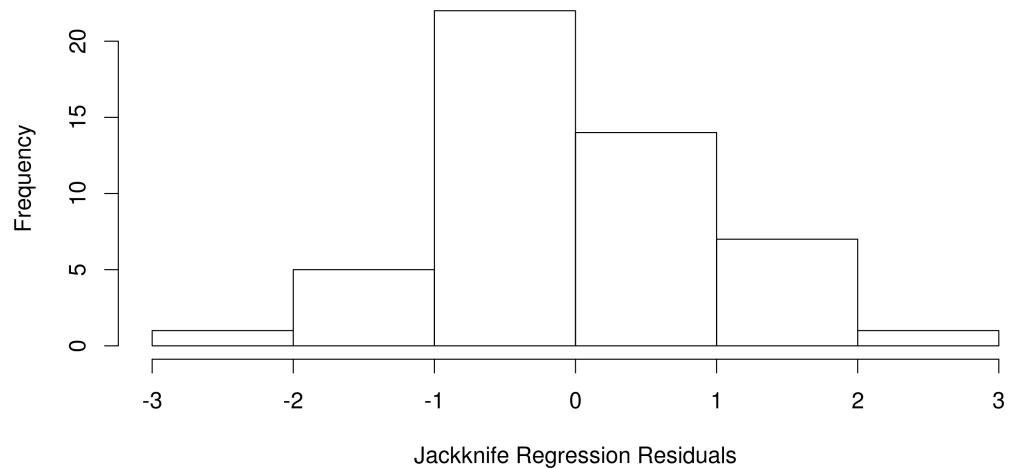
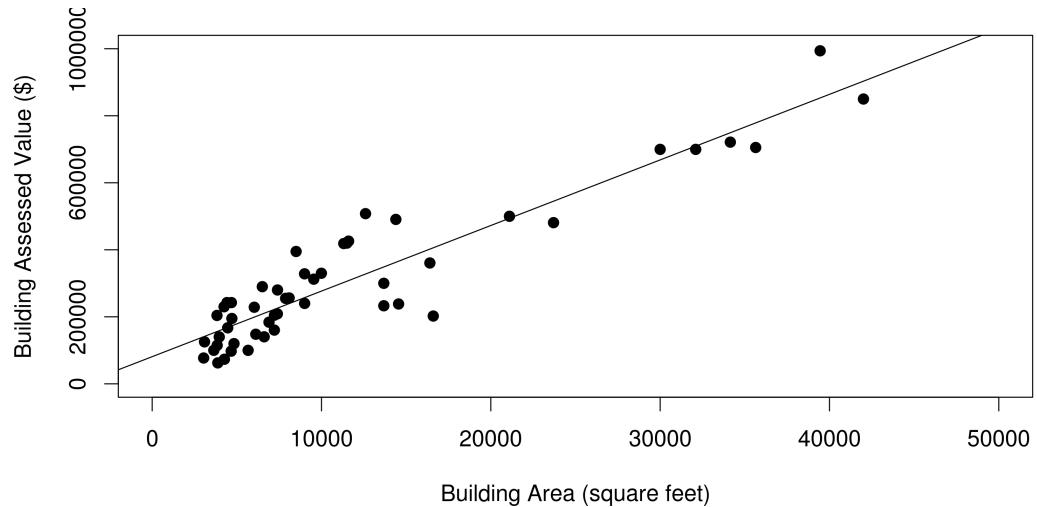


Figure 3.8: Relationship between building area and building assessed value.

The scatterplot indicates that building area is very strongly associated with building value. Now, let's calculate the regression of the residuals of the building's value on the residuals of the building's lotsize, after the regression of both on building area:

```
# resid(x) gives us the raw residuals from the regression object x
partialreg <- lm(resid(valuereg3) ~ resid(lotreg))
png(filename = "partialreg.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 1), cex = 1.3, mar = c(5, 4, 1, 1))
plot(resid(lotreg), resid(valuereg3),
     pch = 19,
     xlab = "Partial Lot Size Residuals",
     ylab = "Partial Building Value Residuals",
     ylim = c(-200000, 200000),
     xlim = c(-10000, 10000))
abline(partialreg)
hist(rstudent(partialreg),
      main = "",
      breaks = "FD",
      xlab = "Jackknife Regression Residuals")
dev.off()
```

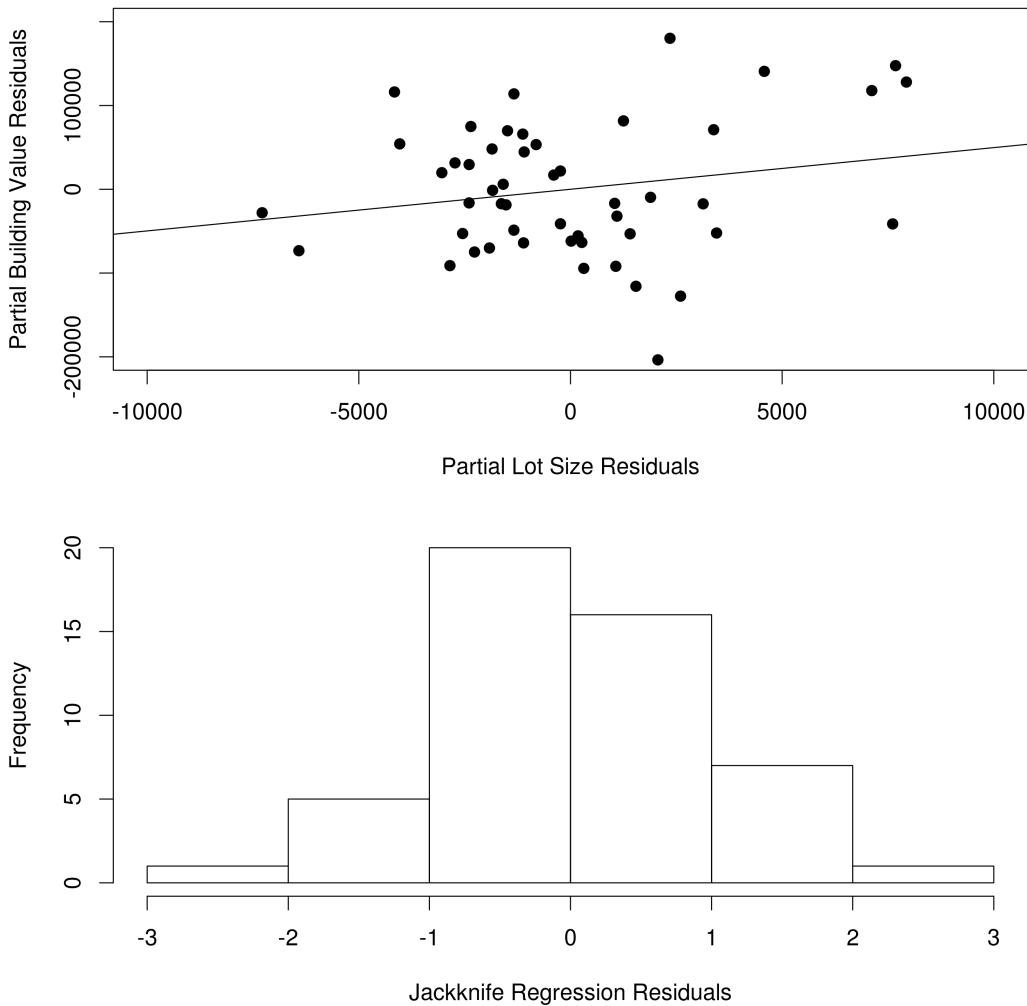


Figure 3.9: Partial regression residuals of building's assesed value and lot size.

As expected, the regression of the partial building value residuals on the partial lotsize residuals is 4.97, the same parameter estimate as we obtained when running the regression of building value on lotsize and building area simultaneously.

```
summary(partialreg)
```

3.4 Discovering Non-Linear Transformations

In this section we will discuss a number of tools that can be used to discover non-linear transformations of the independent variables. Most importantly, they help us continue our conditional distribution story graphically when

there are multiple independent variables that may be non-linearly related to each other and the dependent variable.

3.4.1 Component Plus Residual Plots

Our first approach to continuing the conditional distribution story graphically with multiple regressors is the *Component Plus Residual Plot* (Cook, 1993). Suppose we have a true regression function that has some unknown non-linear function $g()$ that we want to be able to detect from inspecting our regression residuals:

$$y_i = \alpha_0 + \alpha_1 x_{i1} + g(x_{i2}) + \epsilon_i \quad (3.29)$$

In this model, $g()$ could be any non-linear function of x_2 that is linear in parameters (e.g., $g(x_2) = \gamma_1 x_2 + \gamma_2 x_2^2$). This is called a *semi-parametric* model, because we specify a fixed set of parameters for some part of the model (i.e., α_0, α_1), but not others ($g()$). We want to find $g()$, so we try estimating the following equation then look at the residuals:

$$\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} \quad (3.30)$$

But what is the best way to look at the residuals? Note that the residuals from our incorrect regression are:

$$y_i - \hat{y}_i = y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}) \quad (3.31)$$

Substituting y_i with its true population regression gives:

$$y_i - \hat{y}_i = (\alpha_0 + \alpha_1 x_{i1} + g(x_{i2}) + \epsilon_i) - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}) \quad (3.32)$$

Rearranging, we get:

$$y_i - \hat{y}_i = (\alpha_0 - \beta_0) + (\alpha_1 - \beta_1)x_{i1} + (g(x_{i2}) - \beta_2 x_{i2}) + \epsilon_i \quad (3.33)$$

So, if β_0 and β_1 are correctly specified (i.e., $\beta_0 = \alpha_0$ and $\beta_1 = \alpha_1$), then the regression residuals will reflect the discrepancy between $g(x_{i2})$, some unknown function, and $\beta_2 x_{i2}$, a line on the partial residuals. The trick to visualizing this relationship with respect to x_2 is adding $\beta_2 x_{i2}$, called the *component*, to each side:

$$y_i - \hat{y}_i + \beta_2 x_{i2} = (\alpha_0 - \beta_0) + (\alpha_1 - \beta_1)x_{i1} + g(x_{i2}) + \epsilon_i \quad (3.34)$$

Thus, with correct specification of the other parameters, adding the component to the residuals gives us the plot of $g()$:

$$y_i - \hat{y}_i + \beta_2 x_{i2} = g(x_{i2}) + \epsilon_i \quad (3.35)$$

Plotting $y_i - \hat{y}_i + \beta_2 x_{i2}$ against x_2 shows us the shape of the function $g()$, and allows us to identify non-linearities. This is called a *component plus residual plot*, as the y axis of the plot includes a residual $y_i - \hat{y}_i$, and regression component $\beta_2 x_{i2}$. That is, the component-residual plot shows us how the residuals differ from what the residuals should look like if $\beta_2 x_2$ was the correct model of $g(x_2)$. Here's a toy component-residual plot example:

```

set.seed(14)
# sample from the uniform distribution from 1 to 10
x <- runif(200, 1, 10)
error <- rnorm(200, 0, 1.5)
y <- x^2 + error # y is a power function of x
lm1 <- lm(y ~ x) # estimate linear regression with incorrect g(x)

png(filename = "toycr1.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 2), cex = 1.3, mar = c(4, 4, 1, 1))
plot(x, y,
      pch = 19,
      col = rgb(0, 0, 0, .5),
      ylim = c(0, 120))
abline(lm1, col = "red",
       lwd = 4, # set line width to 4
       lty = 2) # make it a dashed line

# Plot the residual plus component against x
plot(x, resid(lm1) + coef(lm1)[2]*x,
      pch = 19,
      col = rgb(0, 0, 0, .5),
      ylab = "Component + Residual",
      ylim = c(0, 120))
# Plot the model residual versus x
plot(x, resid(lm1),
      pch = 19,
      col = rgb(0, 0, 0, .5),
      ylab = "Residual")
# Plot the component versus x
plot(x, coef(lm1)[2]*x,
      pch = 19,
      col = rgb(0, 0, 0, .5),
      ylab = "Component",
      ylim = c(0, 120))
dev.off()

```

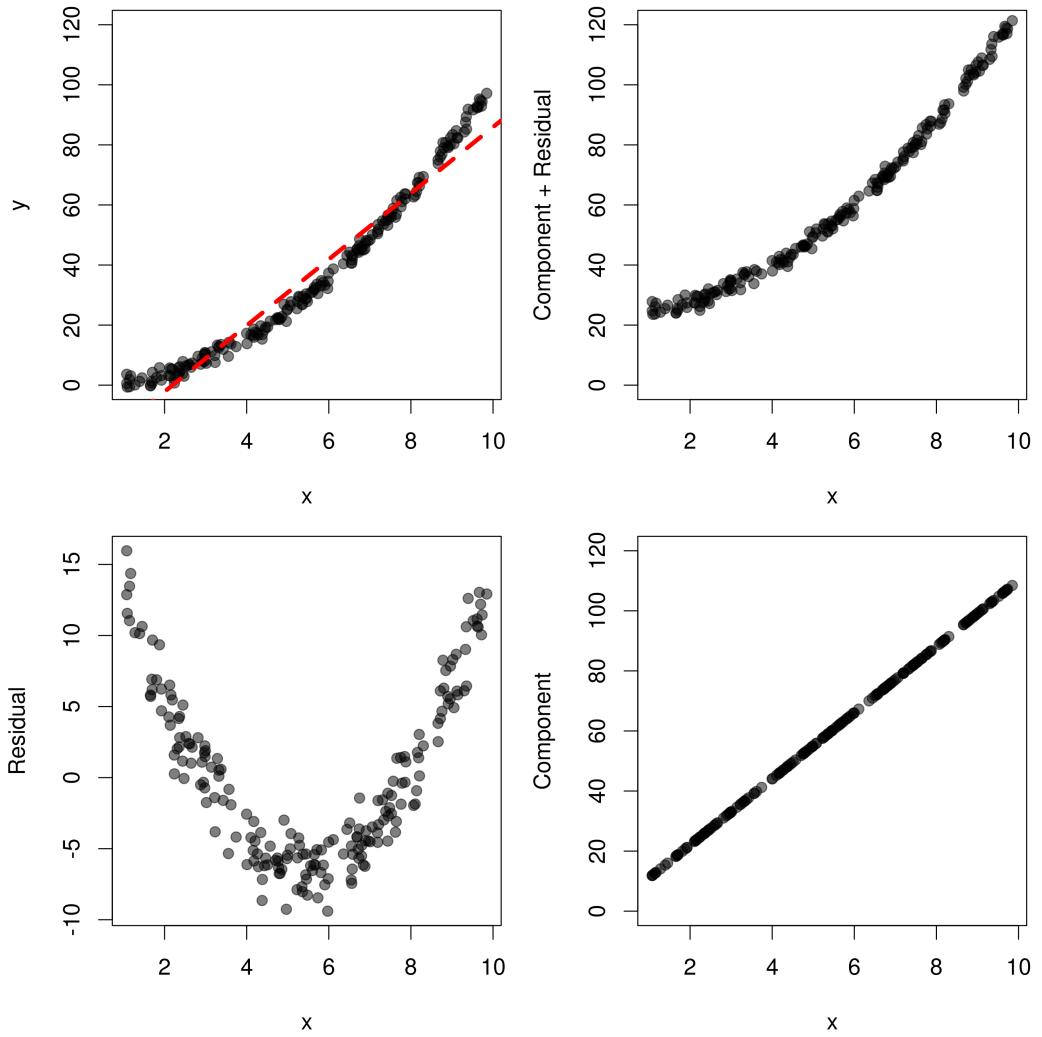


Figure 3.10: Toy component-residual plot example. The top left plot shows the marginal relationship between x and y , which is $y = x^2 + \epsilon$. The top right plot shows that this relationship is recovered by plotting the component plus residuals against x . The bottom left plot shows the residuals plotted against x , failing to show the correct transformation. The bottom right plot shows the component plotted against x .

3.4.2 Box-Tidwell Power Transformations

We can see that the component-residual plot often recovers the true relationship between x and y . If we can't eyeball that relationship, we can use a handy Box-Tidwell function to find a transformation (Box and Tidwell, 1962), if that transformation is a *power function* (i.e., find λ such that $g(x) = x^\lambda$)

```
library(car)
boxTidwell(y ~ x)
```

The BoxTidwell function suggests $\lambda = 2$, which corresponds to the correct power transformation. Box-Tidwell works by first guessing at a value of λ for each regressor in our model, then uses the fact that if there is any difference in our guess and the true value of λ this will show up in the residuals. If we use a Taylor approximation argument, the difference between our hypothesized λ^0 and the actual λ can be estimated using regression (see mathematical appendix for more). We can use this to estimate the difference between our hypothesized value of λ and its true value, then iterate the process until they converge. Specifically, we can find the maximum likelihood estimate of λ using the following approach:

1. Estimate $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1$
2. Estimate $y^* = \beta_0^* + \beta_1^* x_1 + \phi x_1 \ln(x_1)$
3. Calculate $\hat{\lambda} = \frac{\phi}{\hat{\beta}_1} + 1$
4. Repeat 1-3 with $x_1^{\hat{\lambda}}$ rather than x_1 until convergence

Here's an example finding $\lambda = 0.5$ (the square root function):

```
# Generate x uniform 1 to 1000
x <- runif(50, 1, 1000)
# Y is the square root of x plus noise
y <- sqrt(x) + rnorm(50, 0, 1)

# Initial values for lambda and delta lambda
lambda.old <- 1
delta.lambda <- 1

while(delta.lambda > .01){ #loop until the error is small
  # x is raised to the lambda power
  x.lambda <- x^(lambda.old)
  ln.x <- log(x.lambda)
  xln.x <- x.lambda*ln.x

  # Conduct regression with old value of lambda
  hat.reg <- lm(y ~ x.lambda)
  beta.hat <- coef(summary(hat.reg))["x.lambda", "Estimate"]

  # Conduct regression to estimate phi
  star.reg <- lm(y ~ x.lambda + xln.x)
```

```

phi <- coef(summary(star.reg))["xln.x", "Estimate"]

# Calculate new value of lambda
lambda.new <- lambda.old*(phi/beta.hat + 1)

# Measure change in lambda
delta.lambda <- abs(lambda.old - lambda.new)

# Set new value of lambda
lambda.old <- lambda.new
}

# Final value for lambda
lambda.old

```

Keep in mind, that if the true transformation is not a power function (e.g., $g(x)$ is a polynomial function: $g(x) = x + x^2 + x^3$), Box-Tidwell will not work, and that x must be positive and unbounded on the upper end.

We can extend this to partial relationships between some regressor x_j and y , and use the component-residual plot in multiple regression. Let's try with two independent variables:

```

set.seed(15)
# x2 is a uniform random variable from 1 to 100
x2 <- runif(200, 1, 100)
# x1 is a function of x2 plus some error
x1 <- 0.5 * x2 + runif(200, 0, 5)
error <- rnorm(200, 0, 3)
y <- x1 + x2^2 + error # y has g(x) = x^2
lm2 <- lm(y ~ x1 + x2) # fit incorrect linear model

png(filename = "toy2.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 2), cex = 1.3, mar = c(4, 4, 1, 1))
plot(x1, y, pch = 19, ylim = c(0, 10000), col = rgb(0, 0, 0, .5))
plot(x2, y, pch = 19, ylim = c(0, 10000), col = rgb(0, 0, 0, .5))
# Use the component plus residual plot from the car package
crPlots(lm2, terms = "x1", col = rgb(0, 0, 0, .5), pch = 19)
crPlots(lm2, terms = "x2", col = rgb(0, 0, 0, .5), pch = 19)
dev.off()

```

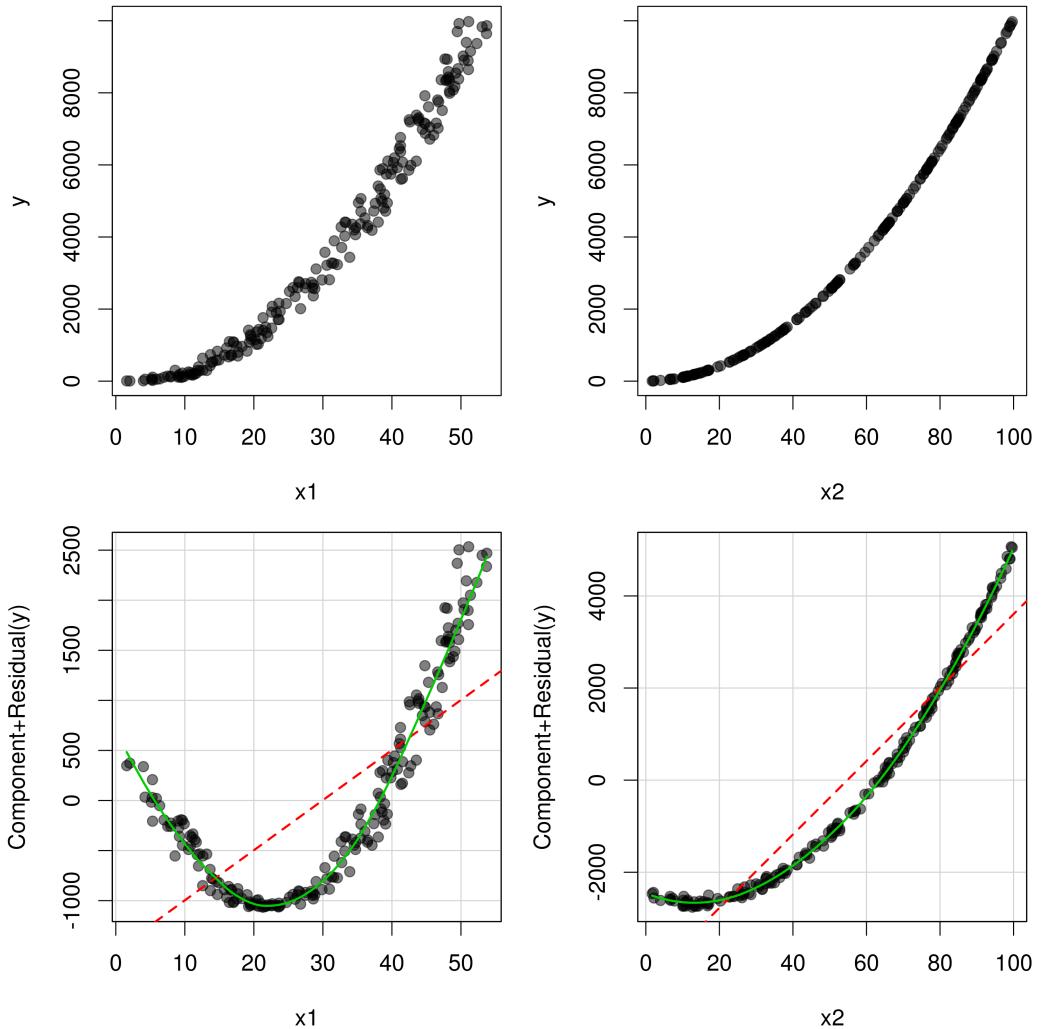


Figure 3.11: Toy component-residual plot example. The true relationship is $y = x_1 + x_2^2$, meaning x_2 needs to be transformed $g(x_2) = x_2^2$. The estimated regression is $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$. x_1 and x_2 are linearly related such that $x_2 = 0.5x_1 + u$ where u is a uniform random variable.

Because x_1 and x_2 are correlated, and the functional form of x_2 is misspecified, the component-residual plot shows misspecification for *both* x_1 and x_2 . The component-residual plot for x_2 shows its correct functional form. However, the component-residual plot for x_1 does not show the correct functional form. Why is that? Well, because x_2 is misspecified, there is an omitted variable, and because x_2 is correlated with x_1 , this leads to omitted variable bias. Thus, whenever a functional form is misspecified, the component-residual plots on linearly related variables will be biased. Because of this, we don't know which variable to transform.

Let's see if Box-Tidwell helps:

```
boxTidwell(y ~ x1 + x2)
```

Box-Tidwell guesses right, again suggesting using $\lambda = 2$ for $g(x_2) = x_2^\lambda$. Alternatively, we might get lucky and try a power transformation $g()$ of x_2 . Either way, $g(x_2) = x^2$ gives us a perfectly linear component-residual plot:

```
lm3 <- lm(y ~ x1 + I(x2^2))
png(filename = "toy3.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 2), cex = 1.3, mar = c(4, 4, 1, 1))
plot(x1, y, pch = 19, col = rgb(0, 0, 0, .5))
plot(x2, y, pch = 19, col = rgb(0, 0, 0, .5))
# We need to tell the crPlots function the correct "terms"
# The terms need to be the exact string used in the regression
crPlots(lm3, terms = "x1", pch = 19, col = rgb(0,0,0, .5))
crPlots(lm3, terms = "I(x2^2)", pch = 19, col = rgb(0,0,0, .5))
dev.off()
```

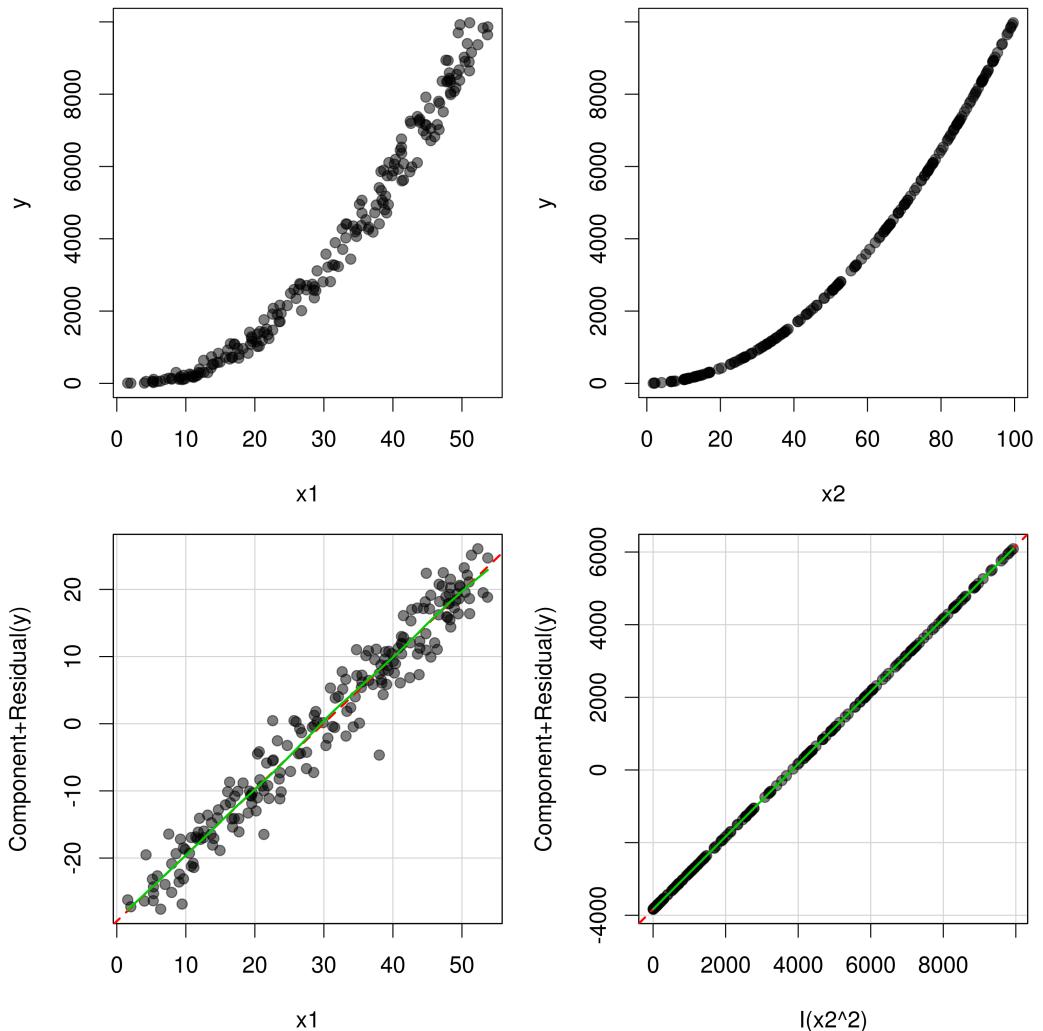


Figure 3.12: Toy component-residual plot example, using the correct power transformation $g(x_2) = x_2^2$. The true relationship is $y = x_1 + x_2^2$, meaning x_2 needs to be transformed $g(x_2) = x_2^2$. The estimated regression is $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$. x_1 and x_2 are linearly related such that $x_2 = 0.5x_1 + u$ where u is a uniform random variable.

Transformations of the regressors are particularly useful when there is some non-linearity in the residuals but the residuals are otherwise homoskedastic, as transforming the dependent variable when residuals are homoskedastic and normal may make the residuals non-normal or heteroskedastic.

3.4.3 Semi/Non Parametric Approaches

Looking at component plus residual plot for potential transformations is an important part of understanding our data and conditional distribution story. Box-Tidwell is also very helpful, but requires knowing that our transformation is monotonic (i.e., a power function). To help us tell whether Box-Tidwell is applicable we can use semi-parametric or non-parametric regression (Wood, 2006).² By “parametric” we mean whether the number of parameters are specified ahead of time. In the usual linear regression, we have a fixed set of parameters, one for each regressor. In semi-parametric or non-parametric regression, we have some regressors that have one parameter, but others are allowed to have more (i.e., combinations of cubic polynomials).

Here’s an example. Suppose the true regression function is a fourth degree polynomial:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 \quad (3.36)$$

Let’s simulate this regression function:

```
# Set polynomial parameters
beta0 <- 24
beta1 <- -50
beta2 <- 35
beta3 <- -10
beta4 <- 1
set.seed(17)
error <- rnorm(100, 0, .5)
x <- runif(500, 0, 5)
# Generate polynomial function
y <- beta0 + beta1*x + beta2*x^2 + beta3*x^3 + beta4*x^4 + error

png(filename = "polynomial1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(x, y, pch = 19, col = rgb(0, 0, 0, .3))
dev.off()
```

²This is covered in detail in the textbook by Simon Wood who wrote the mgcv package
<http://www.maths.bath.ac.uk/~sw283/igam/index.html>

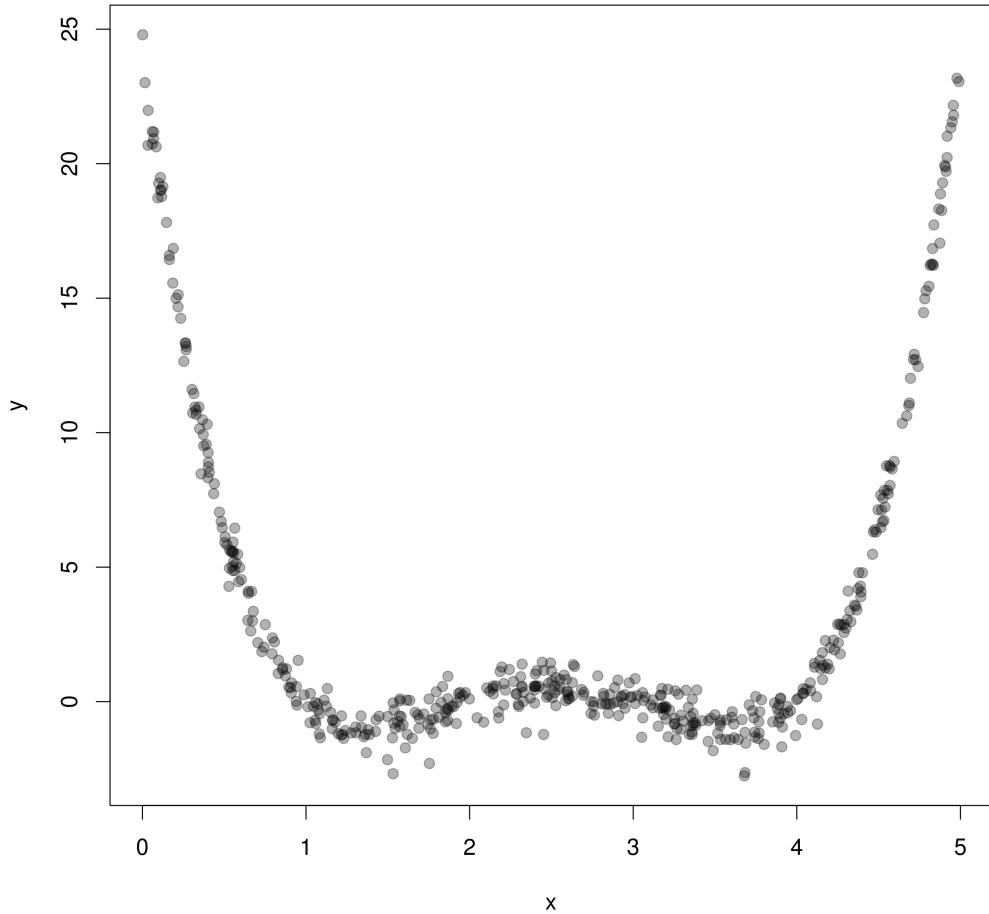


Figure 3.13: “Unknown” fourth-degree polynomial, possibly hidden in a multiple regression function with other independent variables.

We could try to discover the fourth-degree polynomial by trying successively more complex models, or eyeballing the correct function from a component plus residual plot, but there’s no guarantee that we’ll hit the right degree, that we won’t propose a model that is too complex, or that we’ll be able to figure out what the complex model should be when the regression includes multiple linearly and non-linearly related variables.

What if we were to fit this model using a more flexible form where all we assumed about the relationship between y and x was that it included some smooth function $f(x)$:

$$y_i = f(x_i) + e_i \quad (3.37)$$

The question is then how do we choose that smooth function without

assuming it is a power function, polynomial, or trying to eyeball it, all of which become more difficult with more regressors. One way to do this is to choose a space of functions B that $f(x)$ is some subspace of, meaning $f(x)$ is some combination of functions in our basis B . Suppose $b_i(x)$ is the i th basis function, then:

$$f(x) = \sum_{i=1}^q b_i(x)\beta_i \quad (3.38)$$

where the parameters β_i are unknown. For example, a fourth order polynomial basis would be:

$$f(x) = \beta_1 + \beta_2 x + \beta_3 x^2 + \beta_4 x^3 + \beta_5 x^4 \quad (3.39)$$

As can be seen, $b_1 = 1$, $b_2 = x$, $b_3 = x^2$, $b_4 = x^3$ and $b_5 = x^4$. Obviously this basis would easily fit the example population regression function. A more general basis function is called the *cubic spline*, which is just a bunch of cubic polynomials “spliced” together. Where the functions are spliced together are called the *knots* of the spline, and the spline must be smooth around the knots (with continuous first and second derivatives). Knots are usually evenly spaced across the regressor values.

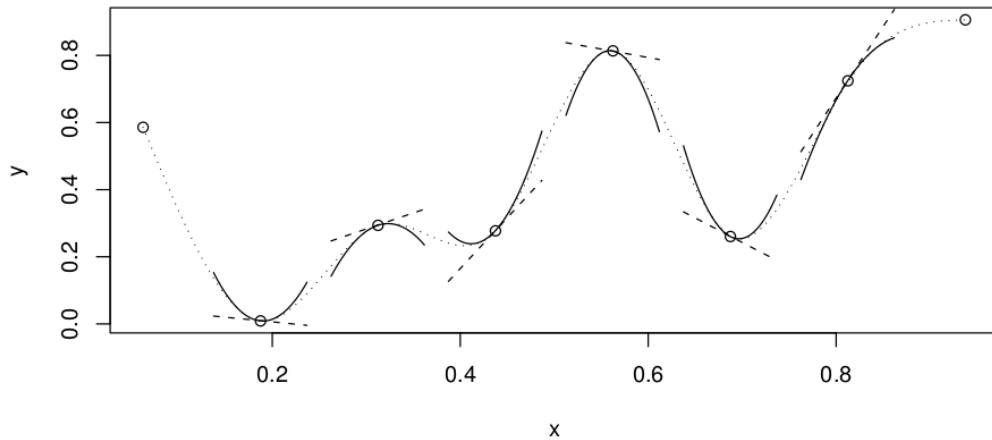


Figure 3.14: Example of cubic spline function with knots circled and first and second derivatives around the knots showing continuity. There are 7 sections of cubic polynomial spliced together to construct the spline. Image taken from Wood, S. (2006). Generalized additive models: An introduction with R. CRC press.

Unfortunately, the number of knots was chosen arbitrarily, and the performance of the regression spline will depend strongly on the knot locations.

We would like to have a more robust approach that doesn't involve such arbitrary decisions. To do this, consider the following penalized regression:

$$\operatorname{argmin}_{\hat{\beta}} [(y - X\hat{\beta})^2 + \lambda \int_x [f''(x)]^2 dx] \quad (3.40)$$

Here $f''(x)$ defines how “wiggly” the model’s smoothing functions are, where the wigglier the functions, the more penalty in the minimization. For example, if $f(x) = x$, then there is no penalty. The degree of penalization is determined by λ , called the *smoothing parameter*. As you would expect, large λ means a very high penalty, favoring linearity, while small $\lambda = 0$ allows for a model with arbitrary flexibility. We can rewrite the objective function as (see [Wood \(2006\)](#) and appendix for more details):

$$\operatorname{argmin}_{\hat{\beta}} [(y - X\hat{\beta})^2 + \lambda \hat{\beta}' S \hat{\beta}] \quad (3.41)$$

The estimator of $\hat{\beta}$ is then:

$$\hat{\beta} = (X' X + \lambda S)^{-1} X' y \quad (3.42)$$

It turns out that we can select λ using cross-validation by leaving a single observation out, fitting the model given a specific λ , then calculating the mean squared forecasting error over all observations, then repeating for different values of λ until the MSE is minimized.

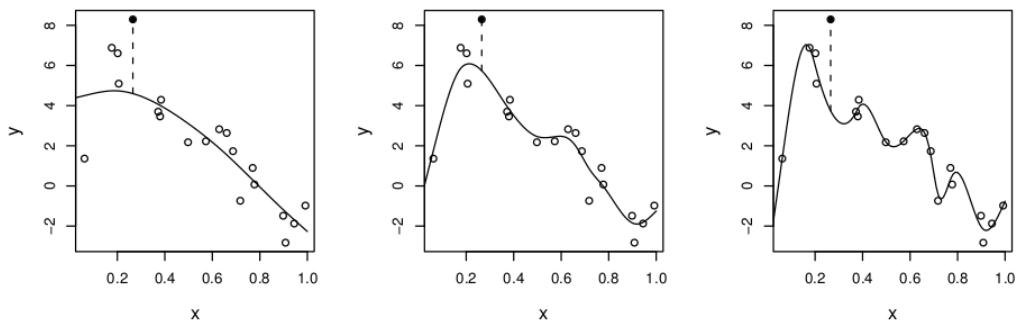


Figure 3.15: Cross-validation to select λ for cubic regression spline. As can be seen λ is too large on the far left plot (too much penalization for wiggliness) and too small on the far right plot (not enough penalty). The plot in the middle shows the optimal value of λ selected by cross-validation. Image taken from [Wood, S. \(2006\)](#). Generalized additive models: An introduction with R. CRC press.

In our example, let's use a GAM to discover the fourth-degree polynomial. Terms with $s()$ are fit using splines (or smoothers), whereas terms without $s()$ are fit using normal linear regression:

```

install.packages("mgcv", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(mgcv)
# Set polynomial parameters
beta0 <- 24
beta1 <- -50
beta2 <- 35
beta3 <- -10
beta4 <- 1
set.seed(17)
error <- rnorm(100, 0, .5)
x <- runif(500, 0, 5)
# Generate polynomial function
y <- beta0 + beta1*x + beta2*x^2 + beta3*x^3 + beta4*x^4 + error
gam1 <- gam(y ~ s(x))

png(filename = "gam1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(x,y, pch = 19, col = rgb(0, 0, 0, .3))
lines(sort(x),
      predict(gam1, # make predictions from the GAM
# The predictions should be onto the sorted x values
      newdata = data.frame(x = sort(x))),
      col      = "green",
      lwd      = 4)
dev.off()

```

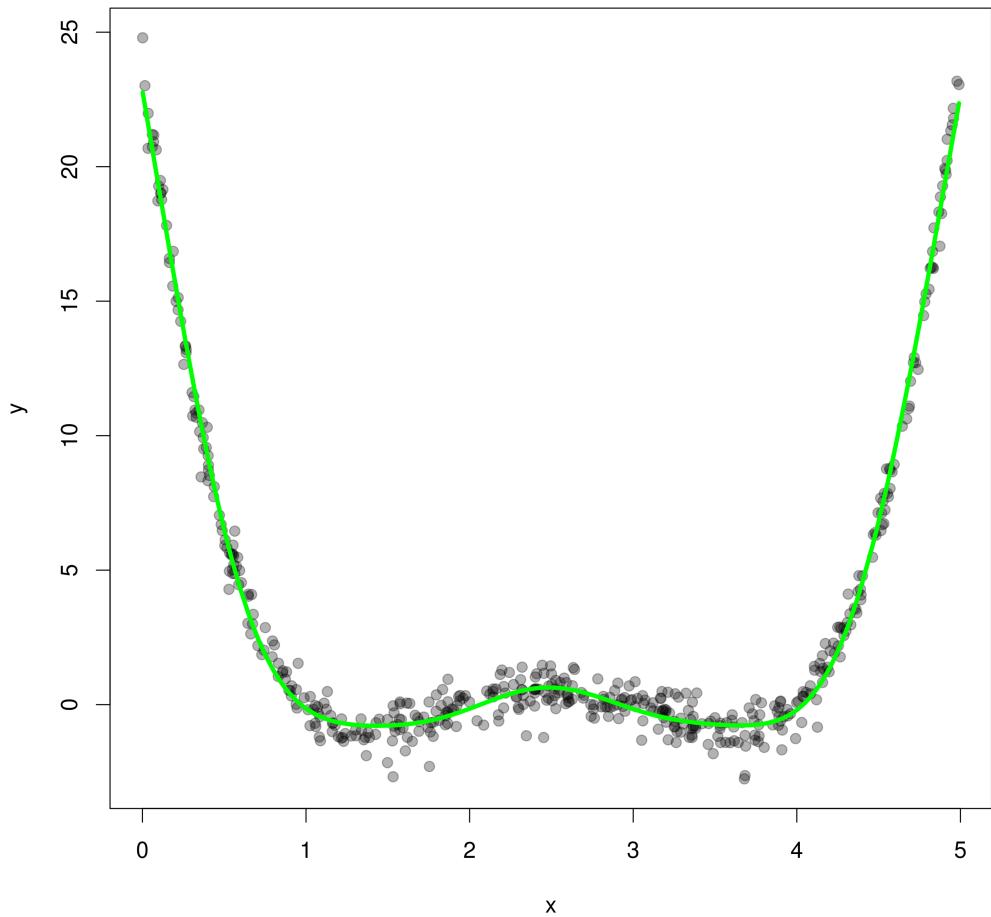


Figure 3.16: Unknown fourth-degree polynomial with GAM overlaid.

Looks pretty good. The GAM discovered the polynomial function perfectly, but there was only one variable. Will our success extend to a more complex example?

```
set.seed(18)
x2 <- runif(200, 1, 5)
x1 <- log(x2) + runif(200, 0, 2) # x2 is the log of x1, plus noise
x1.sq <- x1^2 # Raise to the second power
x1.cu <- x1^3 # Raise to the third power
error <- rnorm(200, 0, 1)
# Generate outcome
y <- 11*x1 - 6*x1.sq + x1.cu + x2^2 + error
```

```
# Use GAM to smooth both variables
gam2<-gam(y~s(x1) + s(x2))

png(filename = "toygam.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(gam2,
      residuals = TRUE, # Include the partial residuals
      shade     = TRUE, # Include shaded confidence bands
      pages    = 1,
      scale     = 0,
      cex       = 3)
dev.off()
```

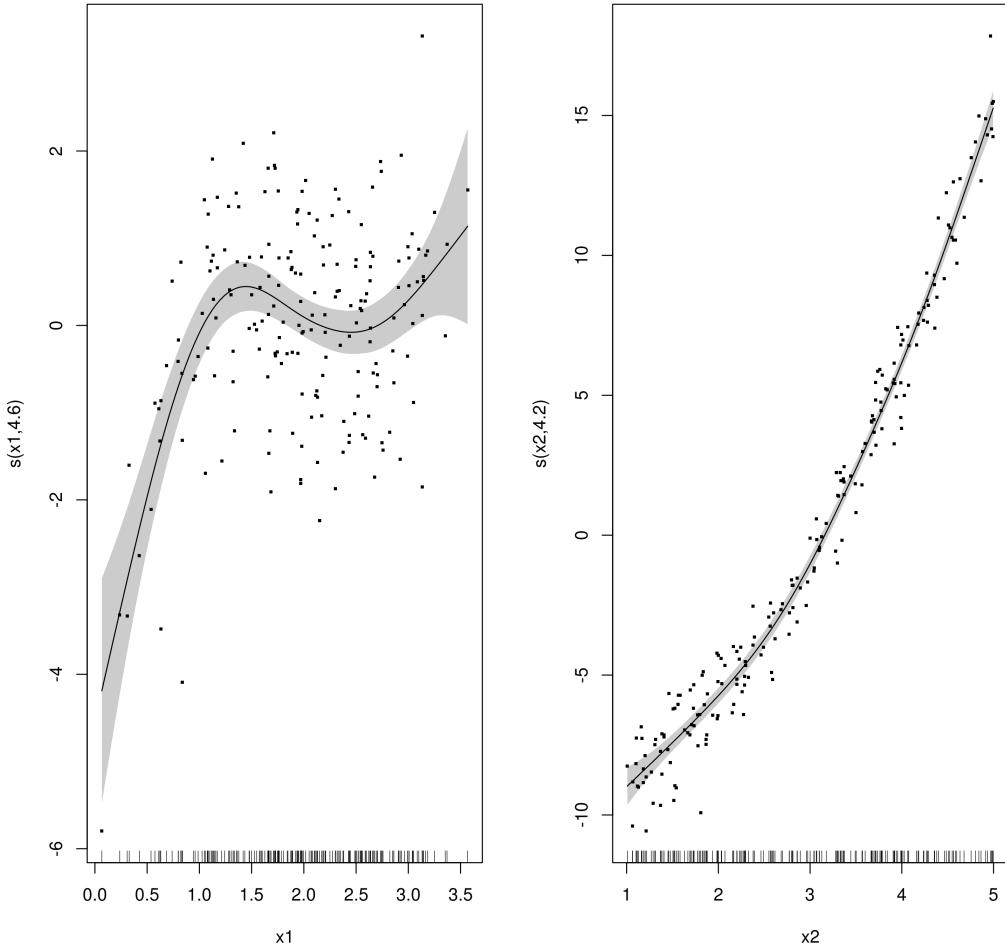


Figure 3.17: Toy partial residual plot example, fit using a generalized additive model. The true population regression function is $y = 11x_1 - 6x_1^2 + x_1^3 + x_2^2$. The plot on the left shows that the GAM discovered the cubic polynomial relationship between x_1 and y , and the right plot shows the GAM discovered the power relationship between x_2 and y . Shaded areas show 95% confidence bands around the conditional mean.

The most important thing I want you to learn from the GAM is that it allows us to continue our conditional distribution story graphically with multiple regressors using *partial residual plots*. The partial residual plots from the GAM show the correct partial relationships between y and x_1 and x_2 . In this plot, the y axis shows the partial residuals:

$$\hat{\epsilon}_{i1}^{partial} = s_1(x_{i1}) + \hat{\epsilon}_i \quad (3.43)$$

where $\hat{\epsilon}_i = y_i - \hat{y}_i$. The logic behind the partial residuals is the same as

the component plus residual plot. The population regression function is:

$$y = 11x_1 - 6x_1^2 + x_1^3 + x_2^2 \quad (3.44)$$

The fitted model was:

$$\hat{y} = s(x_1) + s(x_2) \quad (3.45)$$

Thus, the partial residuals for x_1 are:

$$y_i - \hat{y}_i + s(x_1) = (11x_1 - 6x_1^2 + x_1^3 + x_2^2) - (s(x_1) + s(x_2)) + s(x_1) \quad (3.46)$$

Rearranging, this gives us:

$$y_i - \hat{y}_i + s(x_1) = (11x_1 - 6x_1^2 + x_1^3) + (x_2^2 - s(x_2)) \quad (3.47)$$

So, if $x_2^2 = s(x_2)$, then the partial residual plot shows us

$$y_i - \hat{y}_i + s(x_1) = 11x_1 - 6x_1^2 + x_1^3 \quad (3.48)$$

which is what we want to see. However, we should be more certain about a pattern in the partial residual plot than the component plus residual plot because we have a good chance that $x_2^2 = s(x_2)$, as this was estimated using penalized splines.

You might be wondering why not just use GAM all the time. The answer is that the results are useful for prediction and capturing patterns in the data, but it is difficult to use the GAM to give a simple functional form for our results. Thus, if we want to produce a parametric model that we believe in, we can use GAM to check whether that model needs regressors to be transformed, especially for non-monotonic transformations. Most importantly, partial residual plots allow us to continue our conditional distribution story graphically for regressions with multiple independent variables, giving us the opportunity to stay as close to the data as possible.

3.4.4 Interpreting Coefficients with Non-linear Transforms

Once we've discovered and decided on any non-linear transformations we'd like to do to our predictors, we then need to interpret the results. As an example, we'll look at a person's time to finish a 5K footrace based on the person's age and the average distance run each week in training. The example is taken from [Judd and McClelland \(1989\)](#). Let's begin by loading the data:

```
runners <- read.table("RunnersData.txt",
                      col.names = c("time", "age", "miles"))
```

Let's look at some descriptive statistics using a the *describe* function:³

```
install.packages("psych", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(psych)
describe(runners)
```

Next, let's look at the scatterplots of time and age as well as time and miles:

```
png(filename = "timescatters.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 1), cex = 1.3, mar = c(5, 4, 1, 1))
plot(jitter(runners$age),
     jitter(runners$time),
     ylab = "Race Finish Time",
     xlab = "Runner's Age",
     xlim = c(20, 60),
     ylim = c(14, 40),
     pch = 19,
     col = rgb(0, 0, 0, .5),
     main = "")
plot(jitter(runners$miles),
     jitter(runners$time, amount = .00001),
     ylab = "Race Finish Time",
     xlab = "Average Miles Run in Training",
     xlim = c(0, 60),
     ylim = c(14, 40),
     pch = 19,
     col = rgb(0, 0, 0, .5),
     main = "")
dev.off()
```

³More on descriptives here:<http://www.statmethods.net/stats/descriptives.html>

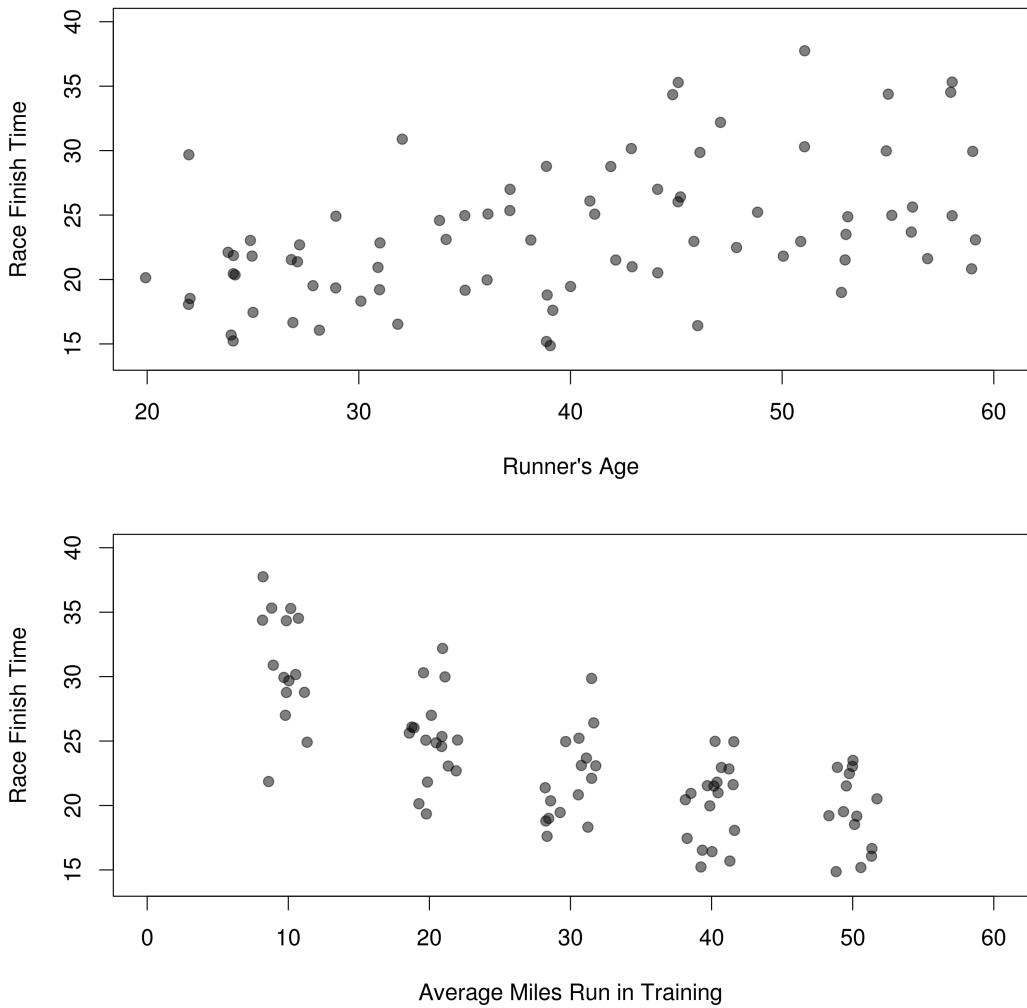


Figure 3.18: Scatterplot of time to finish a 5k race against runner's age (top), and runner's average number of miles run in training (bottom).

From the plots it looks like older runners have longer times, and runners who trained more by running more miles have shorter times. Seems pretty intuitive. It also looks like there might be a quadratic relationship between average miles run in training and race finish time:

```
# Subtract the mean to "center" the miles run variable
runners$miles.c <- runners$miles - mean(runners$miles)
run.quad <- lm(time ~ miles.c + I(miles.c^2), data = runners)
summary(run.quad)
png(filename = "miles_scatters.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 1, 1))
plot(jitter(runners$miles),
```

```
jitter(runners$time, amount = .00001),  
ylab = "Race Finish Time",  
xlab = "Average Miles Run in Training",  
xlim = c(0, 60),  
ylim = c(14, 40),  
pch = 19,  
col = rgb(0, 0, 0, .5),  
main = "")  
# Add quadratic curve fit  
curve(coef(run.quad)[1] + # Intercept  
      coef(run.quad)[2]*(x - mean(x)) + # Coefficient on x  
      coef(run.quad)[3]*(x - mean(x))^2, # Coefficient on x^2  
      add = TRUE)  
dev.off()
```

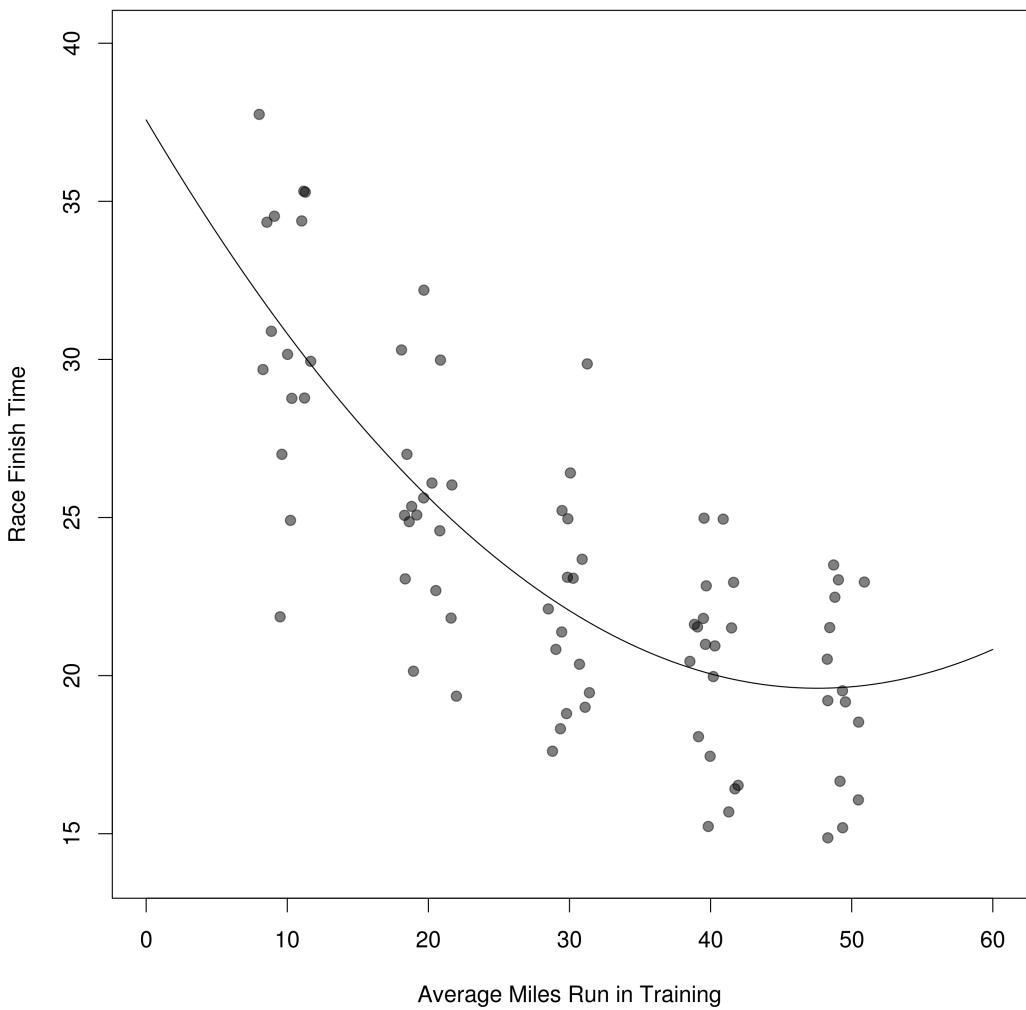


Figure 3.19: Scatterplot of time to finish a 5K race against runner's average number of miles run in training. Curve from regression with quadratic fit is shown.

The quadratic effect of miles in the regression, along with the plot of predicted run times from the regression, can be interpreted in the following way: There are bigger gains, in the form of shorter run times, from increasing training from very little training to some training, but smaller gains from increasing training from a lot to a very lot. The regression equation is:

$$\text{Time} = 22.05 - 0.28 \times \text{miles.c} + 0.008 \times \text{miles.c}^2 \quad (3.49)$$

The intercept tells us that it takes a runner who ran an average number of miles in training (30 miles) about 22 minutes to complete the race on average. To interpret the other two coefficients, it is important to think in terms of the

marginal effect of a unit change in miles run in training, captured by the partial derivative of the regression equation with respect to miles.c. In our example this is:

$$\frac{\partial \text{Time}}{\partial \text{miles.c}} = -0.28 + 2 \times 0.008 \times \text{miles.c} \quad (3.50)$$

When miles is at its mean (i.e., miles.c = 0), the simple slope of miles on time is $-0.28 (= -0.28 + 2 \times 0.008 \times 0)$, meaning those who ran one more mile in training than the average (30 miles) are expected to have a shorter run time by 0.28 minutes, from 22.05 minutes to 21.77 minutes.

We can also find the slope at other quantities of miles trained. For example, at 10 miles of training *below the mean* (i.e., 20 miles of training), the simple slope is $-0.44 (= -0.28 + 2 \times 0.008 \times -10)$, meaning those who ran one more mile in training than someone 10 miles below the average (20 miles) are expected to reduce their run time by 0.44 minutes, from 25.65 minutes to 25.21 minutes.

Thus, the coefficient on the quadratic term indicates the degree of change of the slope of miles on time as miles changes. Because the partial derivative is used, the coefficient is *half* the change in the slope of run times for 1 additional mile of training. If we were to use a cubic term, things would get more complicated. The overall point is that in regression with non-linear predictors, the change in the dependent variable with respect to a one unit change in an independent variable is not constant.

3.5 Multicollinearity

Discovering non-linearities from component plus residual plots and generalized additive models depends in part on the independent variables in our models being distinguishable from each other. *Multicollinearity* happens when predictors in a regression are highly correlated with each other, and will make it difficult to use these tools.

Multicollinearity will also make it hard to precisely estimate parameters for the independent variables that are highly correlated. In multiple ordinary least squares regression, the variance of an estimated parameter is:

$$\text{Var}(\hat{\beta}_{x_1}) = \frac{MSE}{\sum_{i=1}^n (x_i - \bar{x})^2 (1 - R_{x_1}^2)} \quad (3.51)$$

The problematic factor is $R_{x_1}^2$, which is the R^2 value of the regression of x_1 on the remaining predictor variables. A high R^2 means x_1 is highly linearly related to the remaining variables. This will decrease the value of the denominator, thus increasing $\text{Var}(\hat{\beta}_{x_1})$.

Let's see what happens with the building value data. First, let's create dummy codes for the building condition:

```
# Create a variable that is equal to 1 if condition is fair and zero otherwise
aptdata$fair <- ifelse(aptdata$cond == "F", 1, 0)
# Variable is equal to 1 if condition is excellent and zero otherwise
aptdata$excel <- ifelse(aptdata$cond == "E", 1, 0)
```

The first dummy variable is 1 if the apartment's condition is assessed at fair quality ("F"), and 0 if it is assessed at good or excellent quality. The second dummy variable is 1 if the apartment's condition is assessed at excellent quality ("E") and 0 if it is assessed at fair or good quality. When we include them both in a regression, then the coefficient on fair tells us the assessment value of buildings in fair condition compared to those that are in good condition, and the coefficient on excellent tells us the assessment value of buildings in excellent condition compared to those in good condition:

```
valuereg <- lm(value ~ lotsize + bldgarea + numapts
+ age + fair + excel, data = aptdata)
```

The signs of the estimated parameters all seem to make sense (i.e., larger lot size is associated with higher assessments).

Students often feel confused when adding multiple variables into a linear regression then finding very unexpected coefficients or high p-values. Very often this is due to multicollinearity. Thus, before dumping everything into a regression, it's very valuable to first examine the correlation matrix of the regressors:

```
# Correlation matrix
cor(aptdata[,c("value","lotsize","bldgarea","numapts","age","fair","excel")])

#Scatterplot matrix
library(car)
png(filename = "scattermatrix.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 1, 1))
scatterplotMatrix(~value + lotsize + bldgarea + numapts + age,
                 data = aptdata)
dev.off()
```

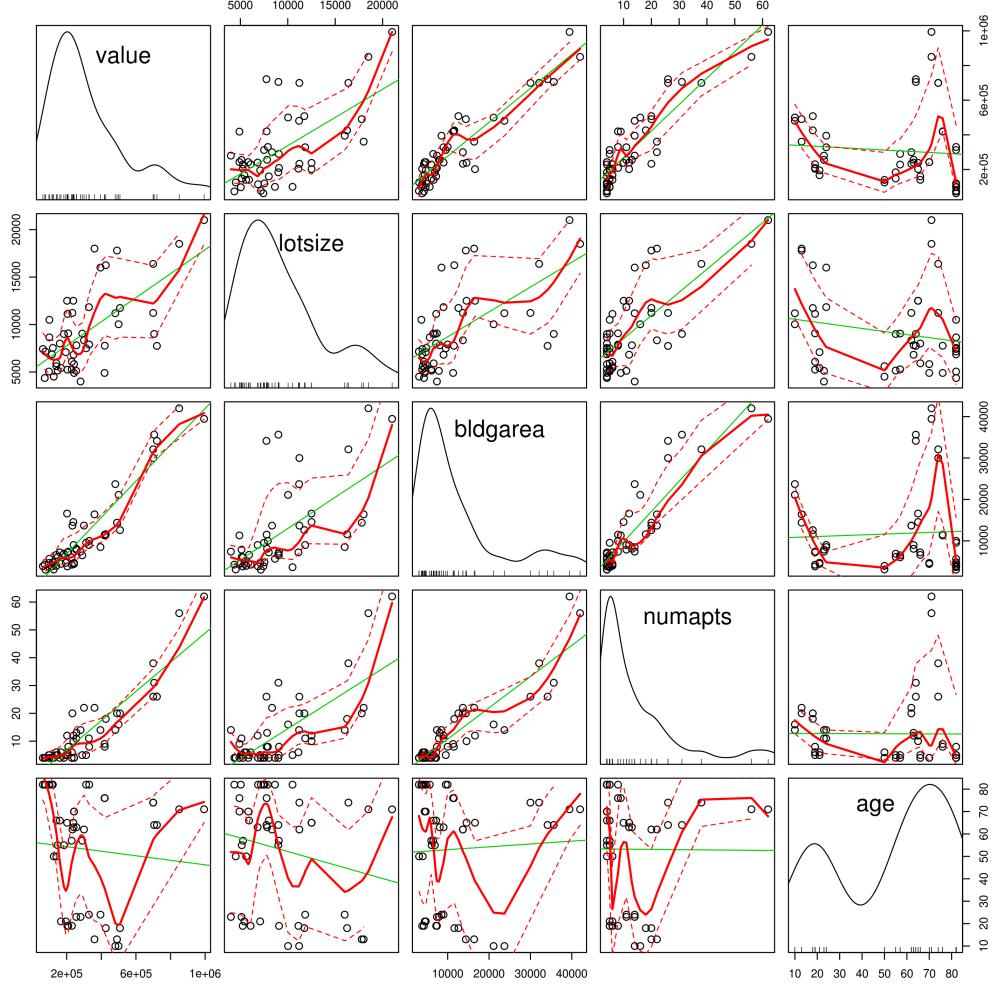


Figure 3.20: Scatterplot matrix of building value, lot size, building area, number of apartments, and age.

We can see that the size of the lot is highly correlated with the building's assessed value, but also highly correlated with the building area and number of apartments in the building. The fact that these variables are also correlated with the building's assessed value explains why lot size is not a significant predictor. Once we've accounted for the effect of building area and number of apartments, lot size does not predict the building's assessed value.

Another way of saying this is that if we regress the building's value on building area and number of apartments, the residuals of that regression are uncorrelated with the variation in lot size that is not accounted for by building area and number of apartments (i.e., the residual of the regression of lot size on building area and number of apartments). Thus, if we were

particularly interested in the effect of lot size, we would say it is difficult to estimate the relationship of lot size to the building's assessed value because lot size is multicollinear with the building's area and number of apartments. We really cannot disentangle the separate effect of lot size above these two variables (if there is any).

3.5.1 Variance Inflation

From the $\text{Var}(\hat{\beta}_{x_1})$ equation above it makes sense that lot size has a large standard error, as $R_{x_1}^2$ is high. Consider the regression of lot size on the other independent variables:

```
lotreg <- lm(lotsize ~ bldgarea + numapts + age
              + fair + excel, data = aptdata)
```

The multiple R^2 of this regression is 0.6, increasing $\text{Var}(\hat{\beta}_{x_1})$ by a factor of about 2.5 ($= \frac{1}{1-0.6}$) (and thus the standard error of the estimated parameter in the regression). This increase in variance is sometimes called the *Variance Inflation Factor* (VIF), which is equal to:

$$\text{VIF} = \frac{1}{1 - R_{x_1}^2} \quad (3.52)$$

We can automatically compute this:

```
vif(valuereg)
```

We see that the variance inflations are quite large for lotsize, building area, and number of apartments.

3.5.2 Dealing with Multicollinearity

Handling multicollinearity depends on what we want to know. If we think we can make a causal story, then omitting variables that are correlated with our independent variable *and* the dependent variable leads to omitted variable bias. Removing building area to more precisely estimate the effect of lot size confounds the two in the regression, making us think that improving lot size improves value, while it also improves building area which is doing a lot of work.

If it makes sense to do so, we can combine the independent variables together to create an index variable, using something like averaging or factor analysis, if that index would be meaningful.

We can also get rid of or control for factors that may be a common cause of the variables in the regression, thus causing them to be highly correlated. For example, numer of crimes, prison cells, and budget for law-enforcement

are highly correlated across cities due to the common factor of population size. Looking at a per-capita basis would reduce such associations.

The latter approach can be done in our case. Let's create a general size variable, which is the building area per apartment:

```
aptdata$aptsize <- aptdata$bldgarea/aptdata$numapts
```

This gives a rough measure of the size of each apartment. Let's drop number of apartments and building area and use our apartment size variable in the regression:

```
valuereg2 <- lm(value ~  
    lotsize + aptsize + age + fair + excel, data = aptdata)
```

The multicollinearity problem is now much less severe:

```
vif(valuereg2)
```

3.6 Heteroskedasticity

So far we've looked at how linear least squares behaves when there are outliers, non-linearities, multicollinearity, and omitted variables. These failures, if unidentified, can make linear least squares biased (except multicollinearity). If identified, they can signal ways to improve our conditional distribution story, and hopefully extend our thinking about the subject matter.

In this section, we will cover *heteroskedasticity*, where the error variances change as a function of the regressors. Equal variances of the errors are not a requirement for linear least squares to be unbiased. However, heteroskedasticity can signal bias, even though it is not a necessary or sufficient condition for bias. There are two reasons to take heteroskedasticity seriously: 1) Heteroskedasticity may suggest an omitted variable or other way of improving our conditional distribution story (e.g., transformations, hidden interactions), 2) Heteroskedasticity messes up standard errors.

3.6.1 Baseball Example

To think about heteroskedasticity, we will use a dataset where a major league coach wants to know the relationship between a baseball player's number of home runs in the minor leagues and major leagues. Suspend your disbelief and imagine that a random sample of minor league players was collected, and followed through the major leagues. Minor is the number of home runs in the last full season in the minors, and Major is the number of home runs in the first full season of the majors.

```
homeruns <- read.table("HomeRunsData.txt", col.names = c("Major", "Minor"))
```

Let's start by looking at a histogram of the number of major homeruns:

```
png(filename = "major.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
hist(homeruns$Major,
      main = "Homeruns in the Majors",
      xlab = "Number of Homeruns",
      breaks = "FD")
dev.off()
```

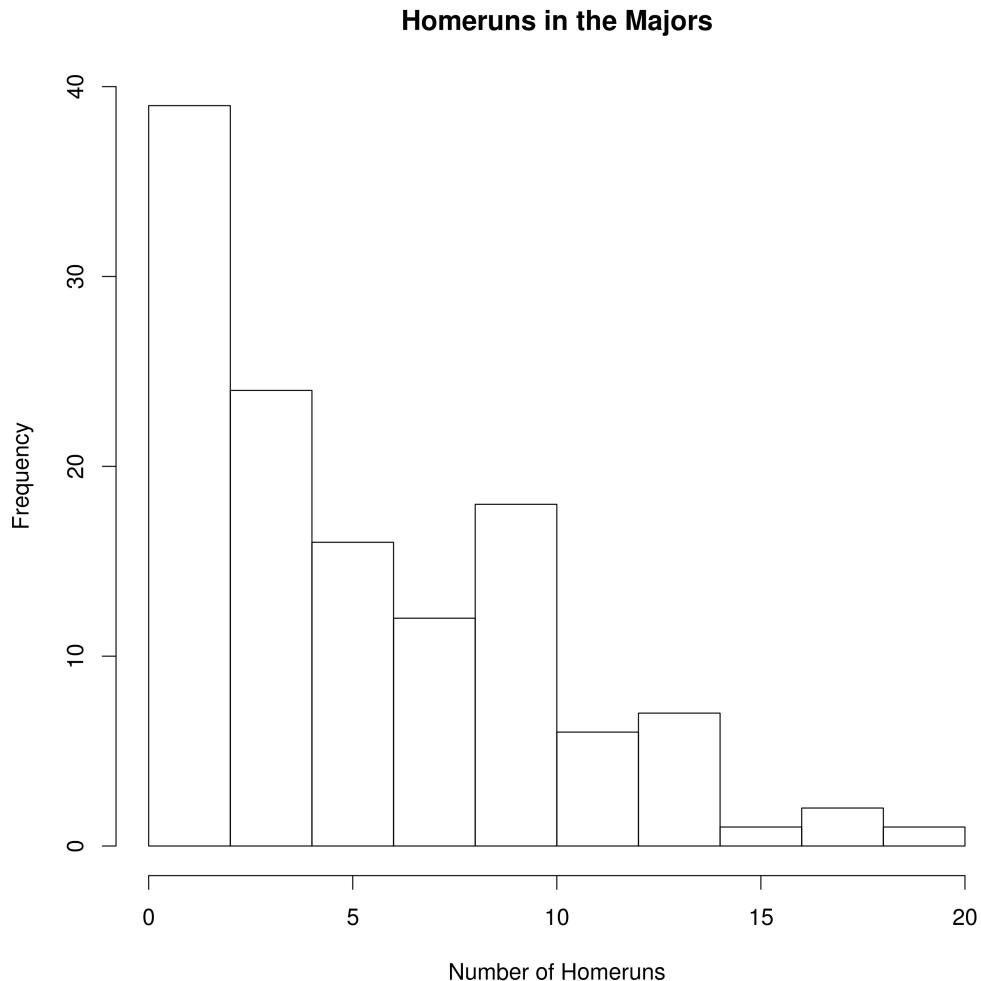


Figure 3.21: Histogram of homeruns in the major league (Majors).

We can see that the distribution of homeruns in the majors is highly skewed. Most players hit zero or one homerun in the majors, with the number of players hitting more homeruns quickly tapering off. Let's look at the qqplot to check out the skew more carefully:

```
norm <- rnorm(10000, mean(homeruns$Major), sd(homeruns$Major))
png(filename = "qqmajor.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
major.q <- quantile(homeruns$Major, probs = seq(.05, .95, .05))
normal.q <- quantile(norm, probs = seq(.05, .95, .05))
min <- min(floor(min(major.q, normal.q)))
max <- max(ceiling(max(major.q, normal.q)))
plot(normal.q, major.q,
      xlim = c(min, 15),
      ylim = c(min, 15),
      ylab = "Quantiles of Major Homeruns",
      xlab = "Quantiles of Normal Distribution",
      pch = 19,
      col = rgb(0, 0, 0, .5))
abline(0,1)
dev.off()
```

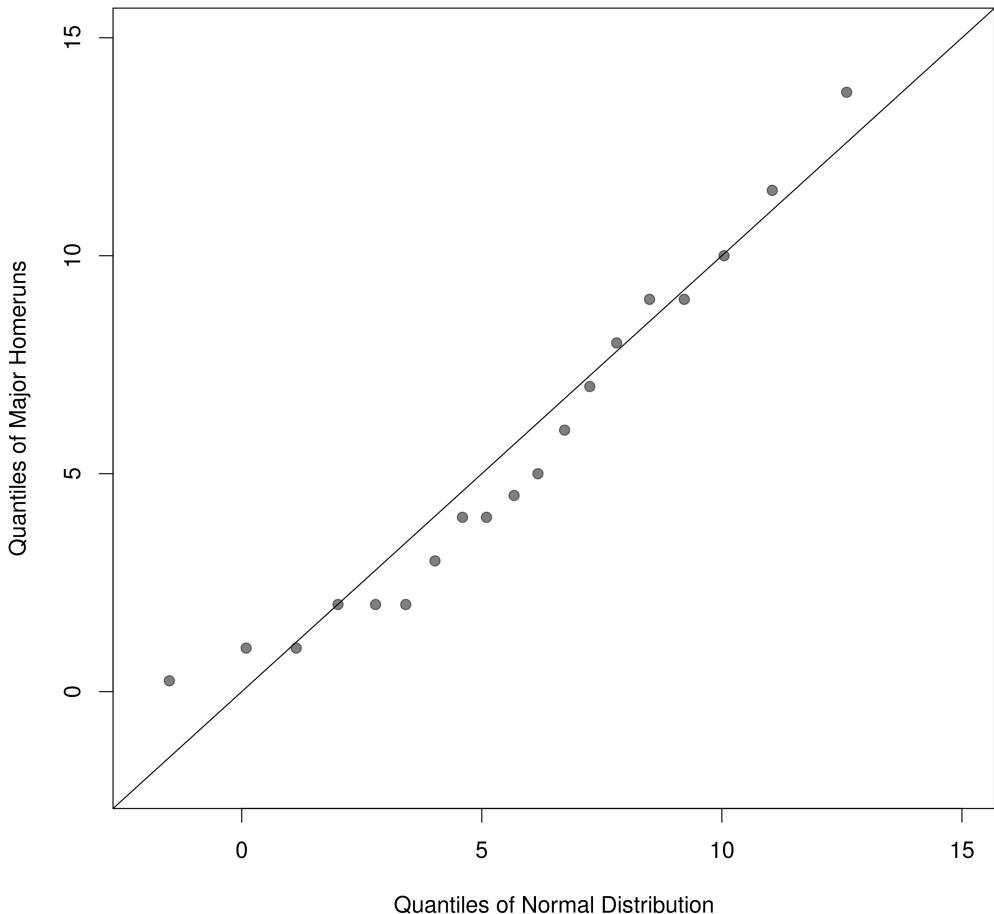


Figure 3.22: Quantile-Quantile plot of homeruns in the majors against normal distribution with equal mean and standard deviation.

It looks like the number of major homeruns are compressed relative to the normal distribution, most likely because number of homeruns cannot be negative, while the normal distribution can generate negative observations. Let's do the same for the minor homeruns:

```
png(filename = "minor.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
hist(homeruns$Minor,
      ylim = c(0, 50), # Set the limit manually
      main = "Homeruns in the Minors",
      xlab = "Number of Homeruns",
      breaks = "FD")
```

```
dev.off()
```

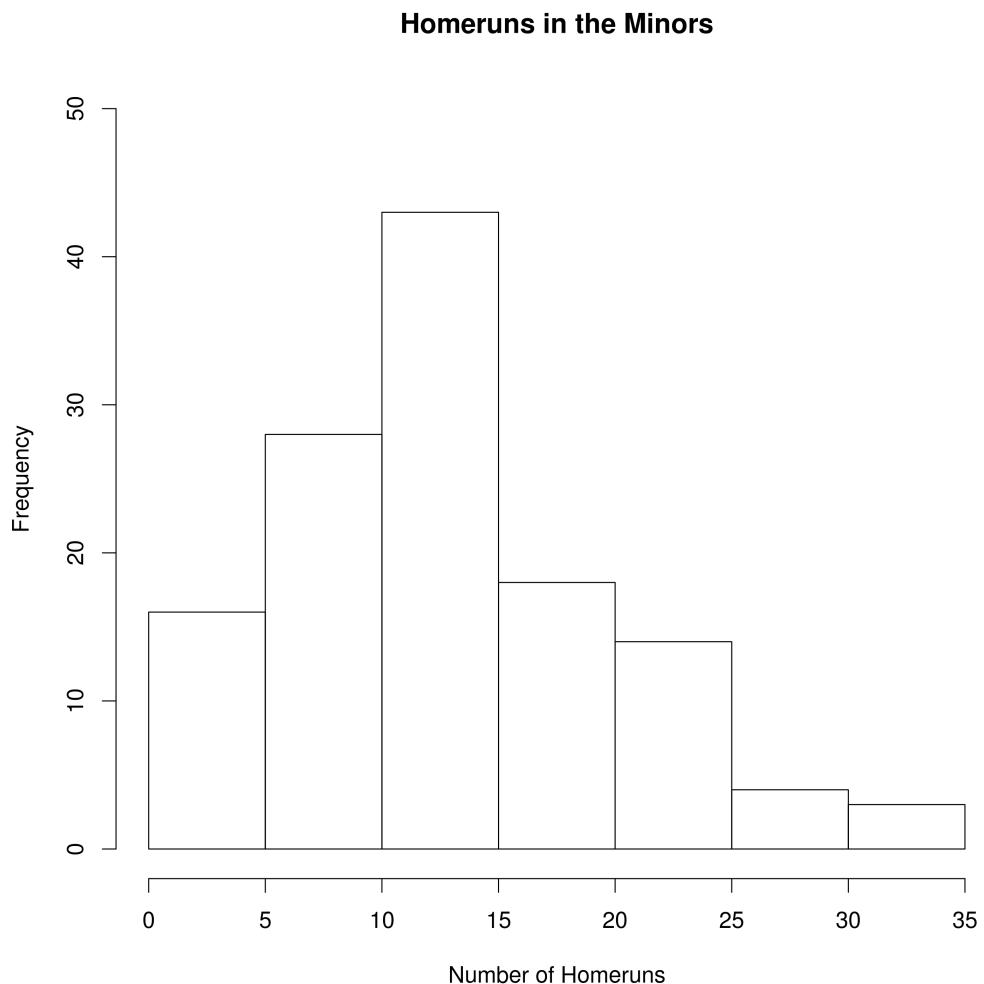


Figure 3.23: Histogram of homeruns in the minor league (Minors).

This distribution looks quite different from the first. Many players hit more than zero homeruns in the minors, with most hitting between 12 and 15 homeruns. The distribution also has a few players with a huge number of homeruns (more than 25).

Next, let's look at the conditional distribution of major homeruns by minor homeruns using a scatterplot:

```
png(filename = "minormajorsscatter.png", width = 3000, height = 3000, res = 300)
par(cex = 1.5)
plot(jitter(homeruns$Minor),
```

```

jitter(homeruns$Major),
ylab = "Major Homeruns",
xlab = "Minor Homeruns",
xlim = c(0, 35),
ylim = c(0, 35),
pch  = 19,
col  = rgb(0, 0, 0, .3),
main = "Homeruns in the Minors and Majors")
abline(0, 1)
dev.off()

```

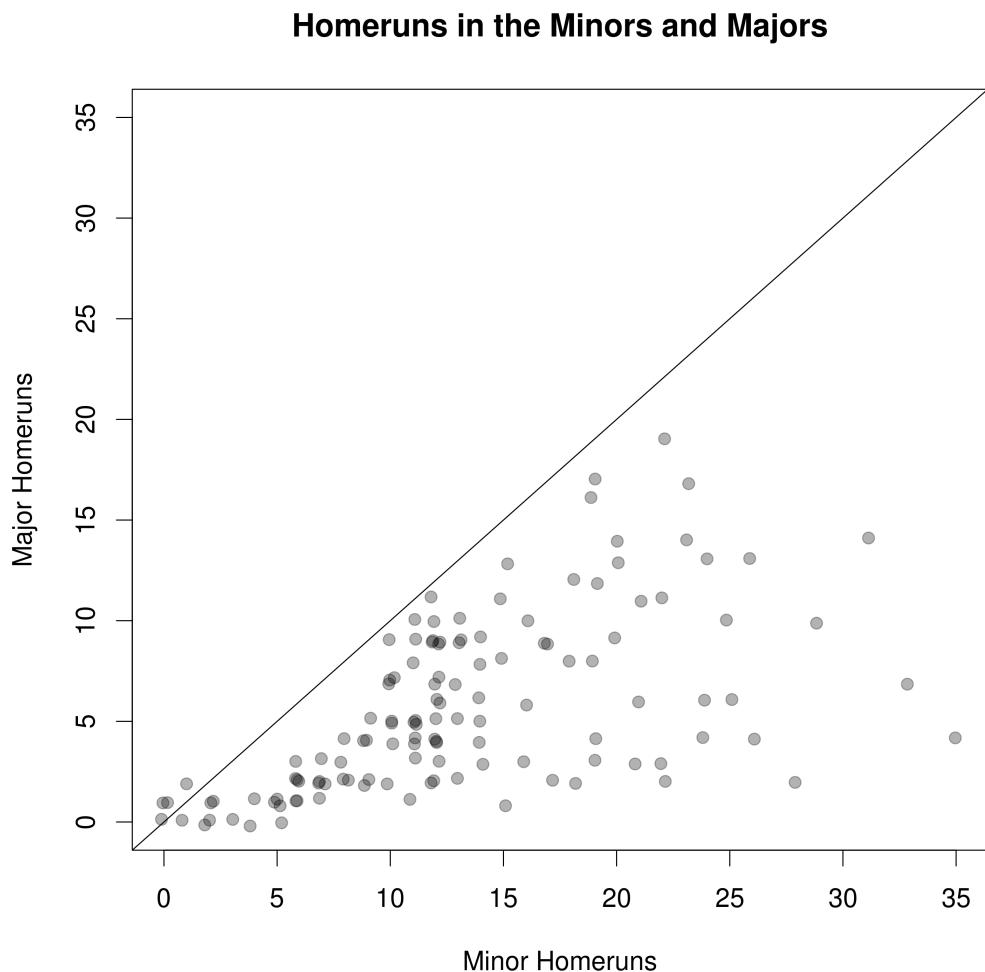


Figure 3.24: Scatterplot of major and minor homeruns.

Almost every point on the graph is below the $y = x$ line, meaning players hit fewer homeruns in the majors than the minors. The only exceptions are at

the low end, where a few players who hit zero or one homerun in the minors hit one or two in the majors.

We can also see that the spread of major homeruns increases as minor homeruns increases. The performance of those who hit many homeruns in the minors is more difficult to predict than those who hit fewer homeruns, as their performance is highly variable in the majors. Let's look at the q-q plots of major and minor homeruns. We can use a sort plot because they have the same sample size:

```
png(filename = "qqminormajor.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(sort(homeruns$Minor), sort(homeruns$Major),
      xlim = c(0, 35),
      ylim = c(0, 35),
      xlab = "Quantiles of Minor Homeruns",
      ylab = "Quantiles of Major Homeruns",
      pch = 19,
      col = rgb(0, 0, 0, .7))
abline(0, 1)
dev.off()
```

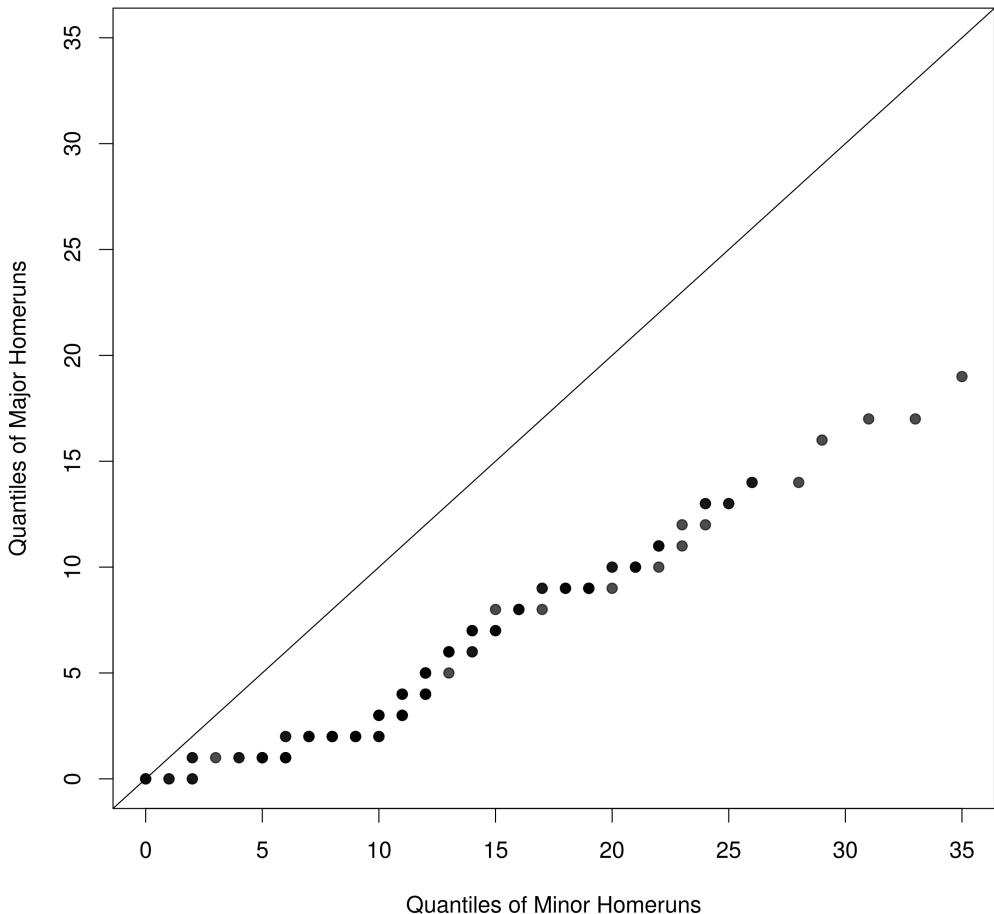


Figure 3.25: Quantile-Quantile plot of homeruns in the minors against homeruns in the majors.

The qq-plot looks almost linear, meaning the minor and major homeruns may be drawn from the same location-scale family. The fact that the slope is flatter than 1 means the spread of minor homeruns is greater than the major homeruns. If minor and major homeruns are a location-scale family, then a regression of major homeruns on minor homeruns should have very well-behaved residuals:

```
homereg <- lm(Major ~ Minor, data = homeruns)
summary(homereg)
```

Obviously, the number of homeruns in the minors is strongly related to the number of homeruns in the majors. The root MSE is 3.53, so on average

66% of observations are within ± 3.53 homeruns of the predicted value. The intercept indicates that a person with zero homeruns in the minors will hit slightly more than one homerun in the majors. The slope indicates that for each additional homerun hit in the minors, the a player is expected to hit 0.34 more homeruns in the majors. Checking out the qqplot of the residuals:

```
png(filename = "homeregqq.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
qqPlot(homereg,
       pch = 19,
       col = rgb(0, 0, 0, .3),
       ylab = "Jackknife Residuals")
dev.off()
```

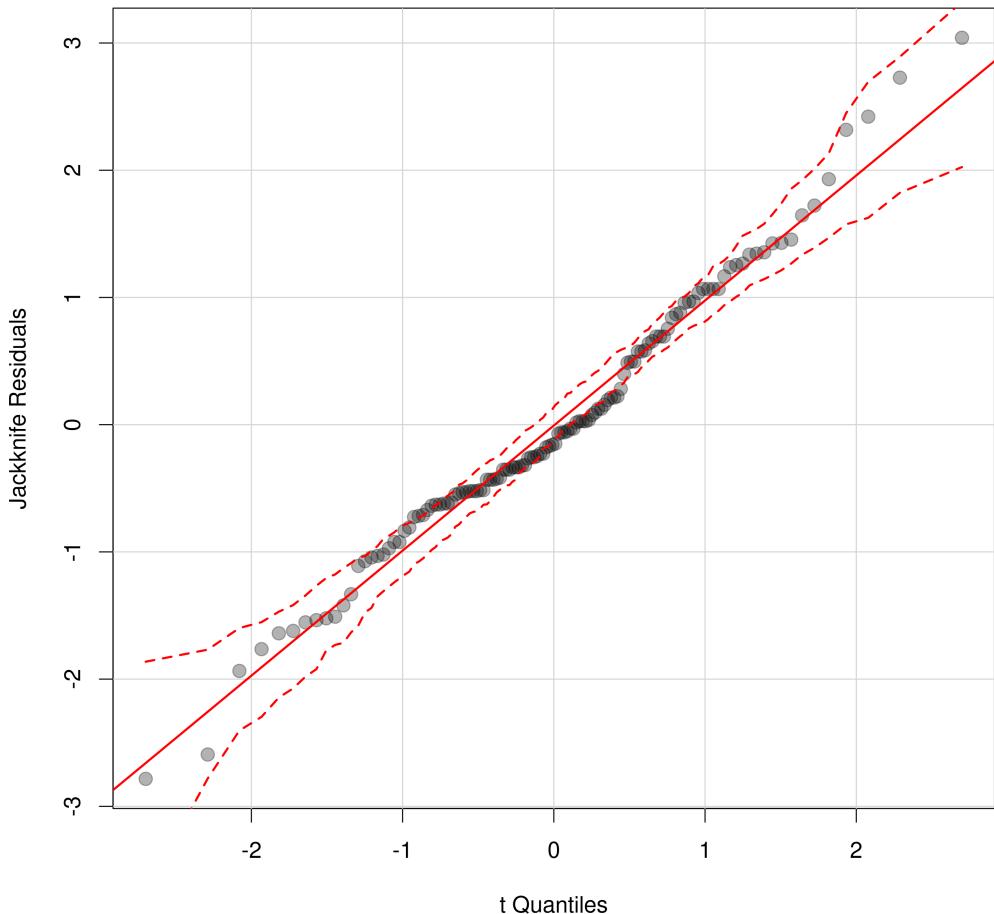


Figure 3.26: Jackknife residuals for the regression of homeruns in the majors on homeruns in the minors.

We see a bit of curvature in the middle, but doesn't look too bad. It seems fair to say that the Jackknife regression residuals follow a distribution that is roughly t with $n - k - 1$ degrees of freedom.

The unconditional distribution of the residuals seems fairly tame. However, looking at a plot of the residuals vs. fitted (predicted) values, a pattern emerges:

```
png(filename = "homeregrefitted.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(jitter(fitted(homereg)), # Plot the fitted/predicted values
     jitter(rstudent(homereg)), # Plot the jackknife residuals
     xlab = "Fitted Values",
```

```

ylab = "Jackknife Residuals",
pch  = 19,
col  = rgb(0, 0, 0, .3),
ylim = c(-4, 4),
xlim = c(0, 14))
abline(h = 0)
dev.off()

```

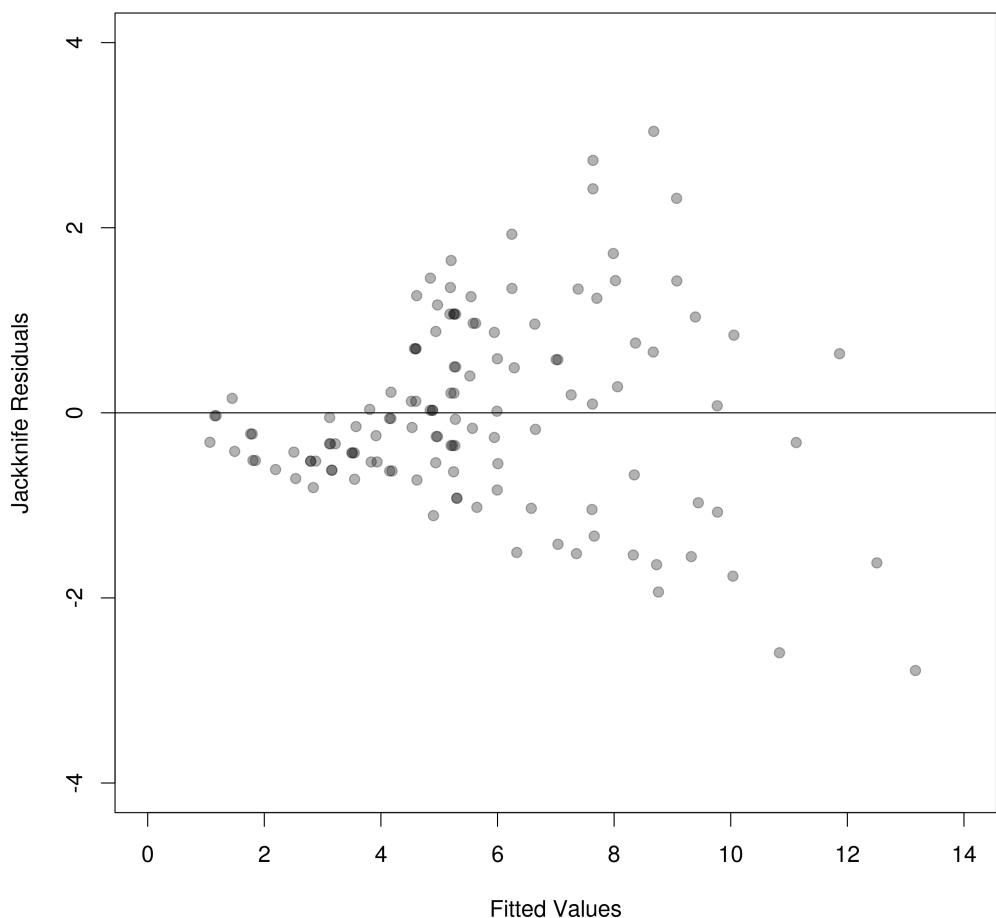


Figure 3.27: Fitted versus Jackknife residual plot of regression of homeruns in the majors on homeruns in the minors.

The zero conditional mean of the errors Gauss-Markov assumption seems to be met, as, on average, the residuals seem to be zero for all fitted values. We also know that the variance of the residuals should get smaller as we move

away from the mean of the predictor variable, due to leverage, but we don't see any such "ballooning."

Instead, the residuals get larger, and more spread out, as the fitted values increase. So, the more homeruns predicted by our model, the greater the variability and error in our predictions. In other words, our predictions only fail about the same on average across the levels of independent variables, but the variability in our prediction errors is not constant. This is a clear case of *heteroskedasticity*, and is common with a skewed dependent variable.

3.6.2 Natural Logarithm and Box-Cox Transforms

Rather than looking to develop a deeper conditional distribution story, we might instead try transforming our dependent variable, hoping the heteroskedasticity goes away. Generally speaking, transforming the dependent variable rather than independent variables is a bad idea for two reasons: 1) It makes *everything* harder to interpret (rather than just the coefficient on a single non-linear predictor), and 2) transformations of predictor variables may still be needed. The natural logarithm transform is so widely used that we can't ignore it. The log transform is part of a set of power transformations called the *Box-Cox Transformations* (Box and Cox, 1964). The goal of these transformations is usually one of three purposes (Cohen et al., 2002):

- *Simplification.* To make the relationship between independent and dependent variables simpler or easier to interpret. For example, economists usually consider the utility of wealth as a concave function, so the natural logarithm is the usual choice.
- *Eliminate Heteroskedasticity.* Transformations may compress or expand regression residuals differently across the independent variables, potentially removing heteroskedasticity.
- *Normalize Residuals.* Transformations can reduce patterns in the residuals, making the regression residuals more well-behaved.

Transformations simultaneously change the conditional mean function, the pattern in the variance function, and the distribution of the residuals, so it is possible to solve problems in all three with a single transformation, or resolve one problem but cause another. A very useful family of transformations of the dependent variable is the Box-Cox family:

$$g(y) = \begin{cases} \frac{y^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(y) & \text{if } \lambda = 0 \end{cases}$$

One key feature of these transformations is that the dependent variable must be strictly positive. To make this true we can add a constant (e.g., 1) to

the number of homeruns in the Majors, then use the Box-Cox transformation from the car package. Alternatively, we can use the Yeo-Johnson power transformations, which work the same as Box-Cox, but for our purposes, use ([Weisberg, 2001](#)):

$$g(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(y + 1) & \text{if } \lambda = 0 \text{ and } y \geq 0 \end{cases}$$

```
library(car)
bc <- boxCox(Major ~ Minor,
               data = homeruns,
               family = "yjPower") # Use the Yeo-Johnson power family
```

You can see that the Box-Cox transformation includes any power transformation. For example, if $\lambda = 2$ then $g(y) = \frac{y^2 - 1}{2}$ giving us a quadratic transformation, and if $\lambda = -1$ then $g(y) = \frac{y^{-1} - 1}{-1} = \frac{-1}{y} + 1$ giving us a reciprocal transformation. Values of $\lambda > 1$ compress the lower tail of a distribution and stretch out the upper tail, which would make a negatively skewed variable (long left tail) more symmetric. Values of $\lambda < 1$ stretch the lower tail and compress the upper tail, which would work to make positively skewed distributions (long right tail) more symmetric. The further from $\lambda = 1$ the stronger the transformation, meaning the log-transformation ($\lambda = 0$) does more to stretch the left tail and compress the right tail than the square root ($\lambda = .5$).

The Box-Cox function outputs a set of possible λ values with a log-likelihood score for each potential λ . We want to select the λ that maximizes the log likelihood. This is equivalent to calculating the transformed dependent variable y^λ for a number of different λ , normalizing y^λ by the scaled geometric mean of y (so we can compare across different dependent variables), then finding the value of λ that minimizes the sum of squared residuals from a regression of $y^\lambda = X\beta$ (more details in the appendix to this chapter). More precisely, we find the value of λ that minimizes the sum of squared residuals from the regression of $Z^{(\lambda)}$ on βX , where:

$$Z^{(\lambda)} = \frac{Y^\lambda - 1}{\lambda(Y_G)^{\lambda-1}} \quad (3.53)$$

for $\lambda \neq 0$ and

$$Z^{(\lambda)} = Y_G \ln(Y) \quad (3.54)$$

for $\lambda = 0$. In each expression, Y_G is the geometric mean of the Y scores on the untransformed scale:

$$Y_G = (Y_1 \times Y_2 \times \cdots \times Y_n)^{1/n} \quad (3.55)$$

This normalization is very important, as it tells us that *we cannot compare models with different dependent variables* unless we make an appropriate adjustment that accounts for the scale of those independent variables. If we accidentally compare two models with different transformations of the same dependent variable, the comparisons will be meaningless. However, we won't make such a blunder because we know that although Box-Cox lets us compare models with different dependent variables (due to the λ transformation), we are comparing *normalized* versions of the transformed dependent variables. Continuing, let's look at the plot of the log likelihood function:

```
png(filename = "bccox11.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
boxCox(Major ~ Minor,
       data = homeruns,
       family = "yjPower",
       plotit = TRUE)
dev.off()
```

We can see that there is a flat maximum (Log-Likelihood = -441) between $\lambda = 0.141$ and $\lambda = 0.303$. So, technically the log transform $\lambda = 0$ is not the best in this case, but it's close enough. So, let's use the log transform and add a constant of 1, as well as a Box-Cox transform with $\lambda = 0.3$:

```
homeruns$lnMajor <- log(homeruns$Major + 1)
homeruns$lambda.0.3 <- ((homeruns$Major + 1)^0.3 - 1)/0.3
```

Notice that we added an arbitrary constant to avoid $\ln(0)$, although it's hard to make general statements about the effect of the choice of constant, it definitely has an effect on the pattern of residuals from the regression. Then plot the log homeruns and $\lambda = 0.3$ transforms:

```
png(filename = "lnmajor.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(2,1))
hist(homeruns$lnMajor,
      ylim = c(0, 40),
      breaks = "FD",
      xlab = "Log Homeruns in the Majors",
      main = "")
hist(homeruns$lambda.0.3,
      ylim = c(0, 40),
      breaks = "FD",
      xlab = "Lambda = 0.3 Transformed Homeruns in the Majors",
      main = "")
dev.off()
```

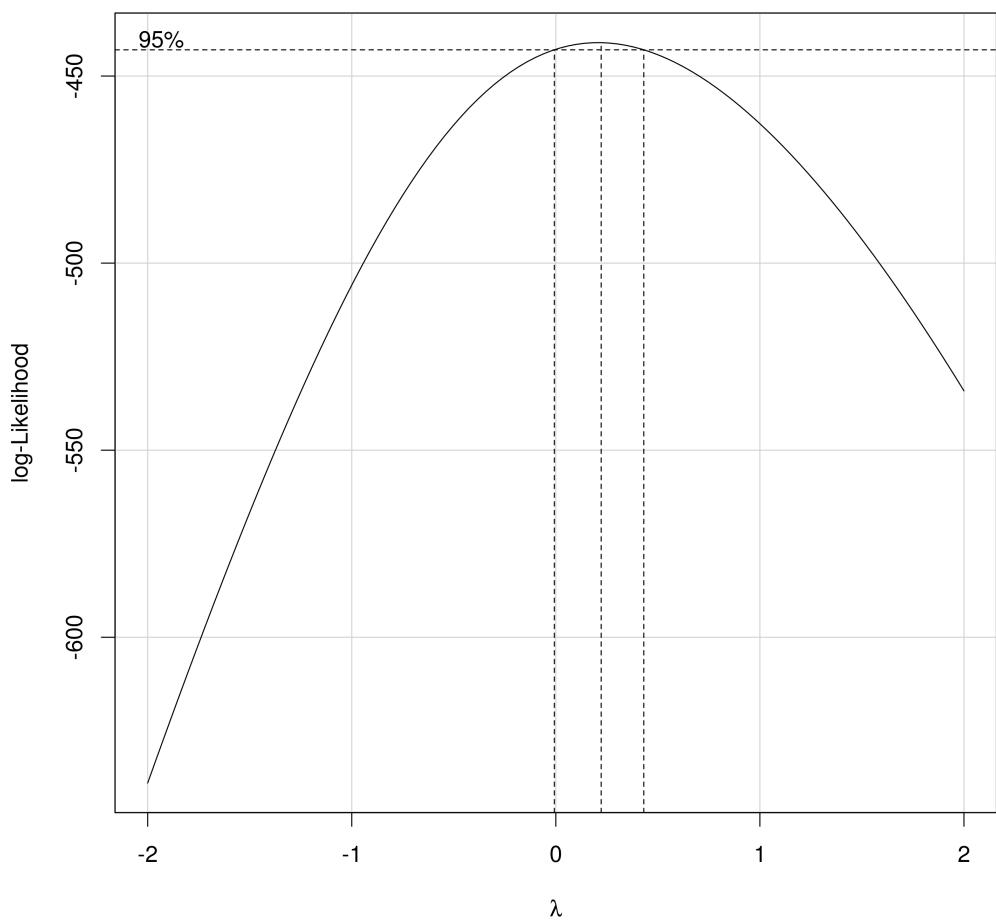


Figure 3.28: Plot of Box-Cox estimate of the parameter λ for the Box-Cox transformation. The x axis shows the λ values that were tested. The y axis shows the log-likelihood.

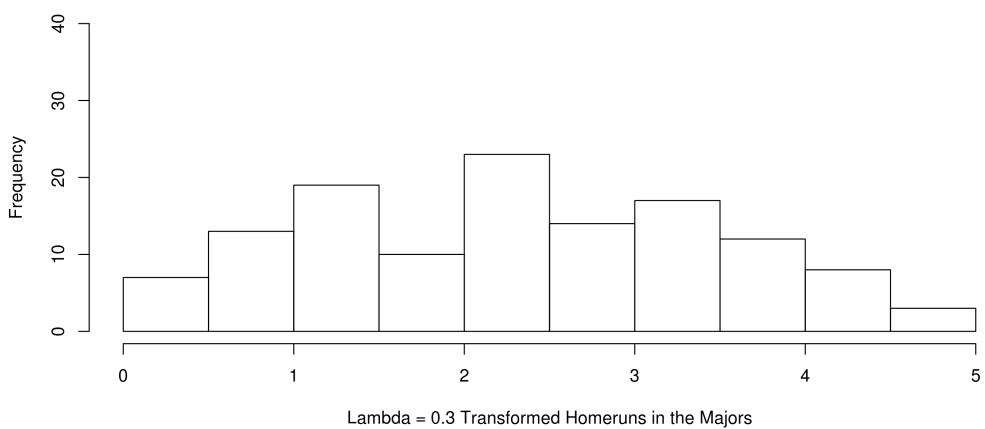
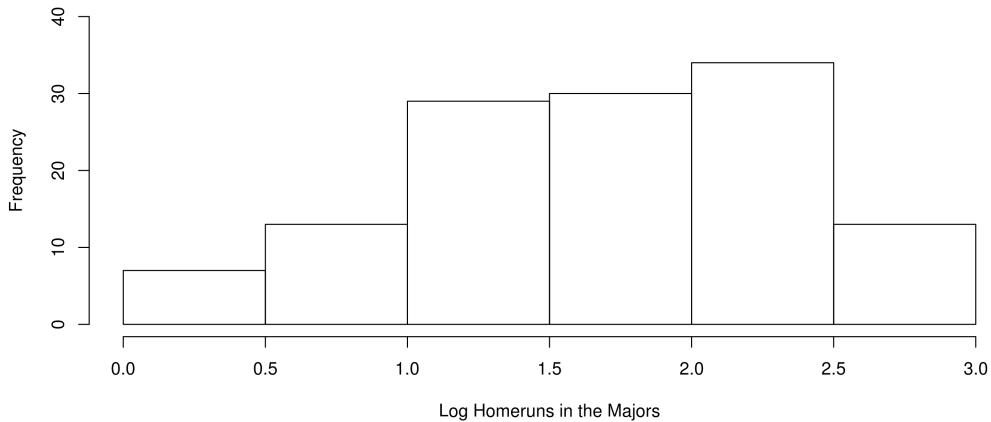


Figure 3.29: Histogram of the log-transformed major homeruns (top), and power transformed homeruns in the majors (bottom) with $\lambda = 0.3$.

I guess we can say the log-transformed distribution is less skewed, although it doesn't look that much better. The $\lambda = 0.3$ transform seems to do a better job, although we'll leave this for now, as the log transform is a popular convention. Next, let's look at the linear regression of the transformed major homeruns on the minor homeruns:

```
png(filename = "minorlnmajorsscatter.png",
     width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(2, 1))
plot(jitter(homeruns$Minor),
     jitter(homeruns$lnMajor),
     ylab = "Log Major Homeruns",
```

```

xlab = "Minor Homeruns",
xlim = c(0, 35),
ylim = c(0, 3),
pch = 19,
col = rgb(0, 0, 0, .5),
main = "Homeruns in the Minors and Majors")
abline(lm(lnMajor ~ Minor, data = homeruns))
plot(jitter(homeruns$Minor),
     jitter(homeruns$lambda.0.3),
     ylab = "Lambda 0.3 Major Homeruns",
     xlab = "Minor Homeruns",
     xlim = c(0, 35),
     ylim = c(0, 5),
     pch = 19,
     col = rgb(0, 0, 0, .5),
     main = "Homeruns in the Minors and Majors")
abline(lm(lambda.0.3 ~ Minor, data = homeruns))
dev.off()

```

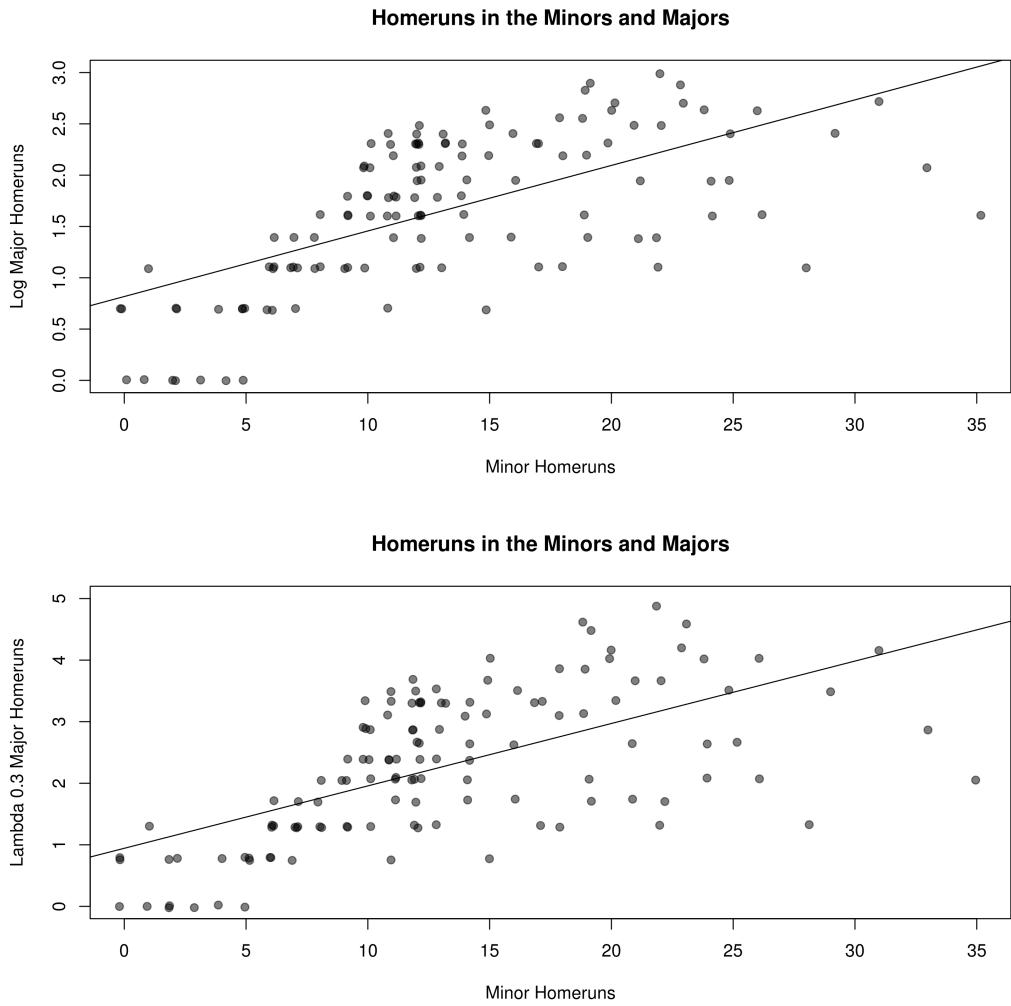


Figure 3.30: Plot of log major homeruns versus minor homeruns (top), and power transformed major homeruns versus minor homeruns (bottom) with $\lambda = 0.3$ for the Box-Cox power transformation.

There's no discernable difference between the two regressions, so we'll focus on the log. Looking at the residuals vs. fitted values:

```
lnreg <- lm(lnMajor ~ Minor, data = homeruns)
png(filename = "lnhomeresfitted.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(jitter(fitted(lnreg)),
     jitter(rstudent(lnreg)),
     xlab = "Fitted Values",
     ylab = "Jackknife Residuals",
     pch = 19,
```

```
col = rgb(0, 0, 0, .3),
ylim = c(-3, 3),
xlim = c(0.5, 3))
abline(h = 0)
dev.off()
```

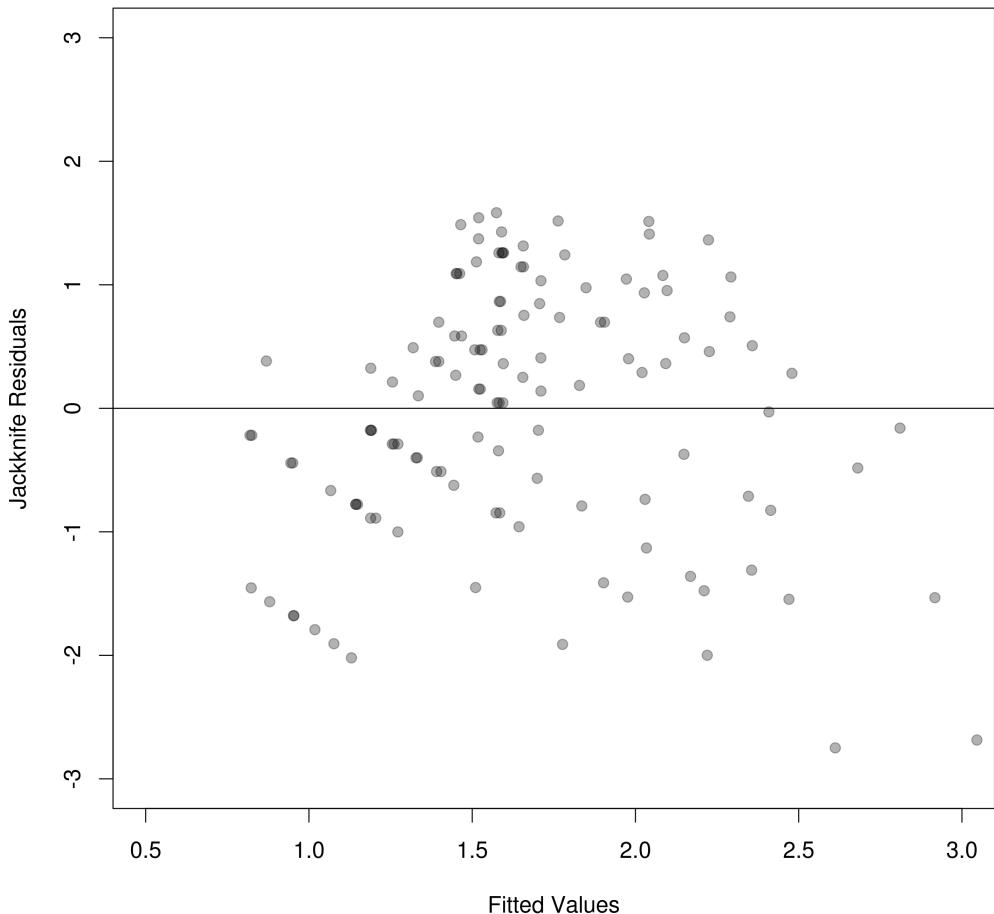


Figure 3.31: Jackknife residuals versus fitted values from the regression of log-transformed major homeruns on minor homeruns.

The residuals vs. fitted values look better. However, the residuals are low, meaning the line is overpredicting the actual values. Notice a few things about the regression output of our original regression versus the regression using the natural logarithm of homeruns in the majors:

```
summary(homereg)
summary(lnreg)
```

The interpretation of the parameters in the two different models are different. For the natural transformed log dependent variable, the estimated regression equation is:

$$\ln(\text{Major Homeruns} + 1) = 0.82 + 0.06 \times \text{Minor Homeruns} \quad (3.56)$$

So, with 0 minor homeruns, the predicted major homeruns is $e^{0.82} - 1 = 1.30$, higher than our un-logged regression, with 1.11 homeruns for zero homeruns in the minors.

The coefficient on Minor Homeruns is now, however, difficult to interpret. The easiest way to interpret the coefficient is to say that one additional homerun in the minors is associated with a 6% (100×0.06) increase in homeruns in the majors. More generally, there are four different ways of interpreting coefficients when the dependent and/or independent variables are log transformed (Wooldridge, 2009):⁴

Model	Dependent Variable	Independent Variable	Interpretation of β_1
level-level	y	x	$\Delta y = \beta_1 \Delta x$
level-log	y	$\log(x)$	$\Delta y = \frac{\beta_1}{100} \% \Delta x$
log-level	$\log(y)$	x	$\% \Delta y = 100 \beta_1 \Delta x$
log-log	$\log(y)$	$\log(x)$	$\% \Delta y = \beta_1 \% \Delta x$

The level-level model, with no transformations, is interpreted as a change in x by 1 ($\Delta x = 1$) is associated with a change of β_1 in y . For the level-log model we have a log transformed independent variable, and our interpretation is that a 1% change in x ($\Delta x = 1\%$) is associated with a change in y of $\frac{\beta_1}{100}$. For the log-level model, the dependent variable is log transformed, and we interpret this as a change in x by 1 ($\Delta x = 1$) is associated with a $100 \times \beta_1$ percent change in y . This is where we got the interpretation that “one additional homerun in the minors is associated with a 6% (100×0.06) increase in homeruns in the majors.” Finally, in the log-log model, the model where both independent and dependent variables are log transformed, we interpret a 1% change in x ($\Delta x = 1\%$) as being associated with a β_1 percent change in y .

If our regressor is a factor or dummy variable, this is not true. It doesn’t make sense to log transform a dummy variable, so we only need to think about the log-level case. Here, if the coefficient on the dummy variable is β_1 then if the dummy switches from 0 to 1, the % change of y is $[100(e^{\beta_1} - 1)]$ (Halvorsen and Palmquist, 1980). If we want to go in the other direction, from 1 to 0, the % change of y is $[100(e^{-\beta_1} - 1)]$.

But this doesn’t tell us how many more homeruns in the majors we should expect if minor homeruns increases by one unit. To get this, we differentiate

⁴More detail on this is provided in the mathematical appendix at the end of the chapter.

the model with respect to Minor Homeruns after inverting the log:

$$\text{Major Homeruns} = e^{0.82+0.06 \times \text{Minor Homeruns}} - 1 \quad (3.57)$$

$$\frac{\partial}{\partial \text{Minor Homeruns}} = 0.06 \times e^{0.82+0.06 \times \text{Minor Homeruns}} \quad (3.58)$$

So, for each additional homerun in the minors, we should expect $0.06 \times e^{0.82+0.06 \times \text{Minor Homeruns}}$ additional homeruns in the majors. Thus, there is a greater effect of getting an additional homerun in the minors for those with more homeruns in the minors than those with fewer homeruns. To see this, let's compare the predictions of our natural log model to the un-logged model:

```
lnpredictions <- predict(lnreg)
predictions <- predict(homereg)
```

We need to transform our natural logged predictions back to their original scale:

$$\lnMajor = \ln(\text{Major} + 1) \quad (3.59)$$

$$\hat{y} = \text{Major} = e^{\lnMajor} - 1 \quad (3.60)$$

Here's the transformation:

```
exp.lnpredictions <- exp(lnpredictions) - 1
```

Now plot the predicted values:

```
png(filename = "tworegscatter1.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 1), cex = 1.3)
plot(jitter(homeruns$Minor),
     jitter(homeruns$Major),
     ylab = "Major Homeruns",
     xlab = "Minor Homeruns",
     xlim = c(0, 35),
     ylim = c(0, 20),
     pch = 19,
     col = rgb(0, 0, 0, .5),
     main = "Log Transformed Model")
# Draw a line showing the model's predictions
lines(homeruns$Minor, exp.lnpredictions, col = "red", lwd = 2)

plot(jitter(homeruns$Minor),
     jitter(homeruns$Major),
```

```
ylab = "Major Homeruns",
xlab = "Minor Homeruns",
xlim = c(0, 35),
ylim = c(0, 20),
pch  = 19,
col  = rgb(0, 0, 0, .5),
main = "Untransformed Model")
abline(lm(Major ~ Minor, data = homeruns), col = "blue", lwd = 2)
dev.off()
```

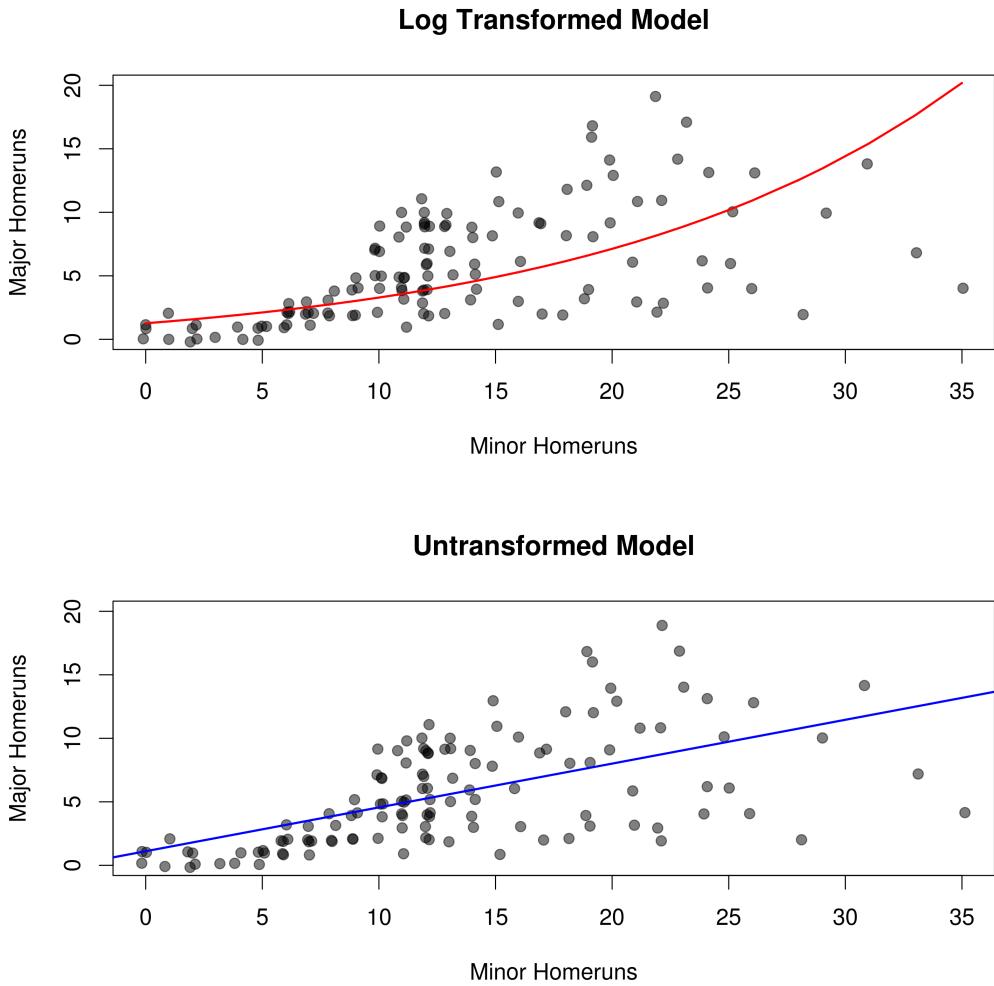


Figure 3.32: Relationship between major homeruns and minor homeruns for the logged (top) and unlogged (bottom) regressions.

As mentioned, there is a greater partial effect of additional minor homeruns the more minor homeruns that were hit. Unfortunately this is not exactly right. Because we are predicting the conditional mean (expected value) after taking a non-linear transformation, the expected value of the transformed equation does not equal the transformed equation of the expected value (this is Jensen's Inequality). Instead, we are predicting the geometric mean, not the arithmetic mean. If we use a logged dependent variable, we have estimated:

$$\ln(\hat{y}) = \hat{\beta}_0 + \hat{\beta}_1 x_1 \quad (3.61)$$

If this follows a normal distribution, then it is true that the log of y is normally distributed with mean $X\beta$ and variance σ^2 . This means that y is

log-normally distributed with mean $e^{X\beta + \frac{\sigma^2}{2}}$. Thus, if we think the errors of the regression of $\ln(y)$ on $X\beta$ are normally distributed, then we can predict the mean for y assuming it has a lognormal distribution, using \hat{MSE} as our estimator of σ^2 (see Wooldridge (2009) Chapter 6):

$$\hat{y} = e^{MSE/2} e^{\ln(\hat{y})} \quad (3.62)$$

Where MSE is the mean squared error from the regression with the log dependent variable:

```
MSE <- summary(lnreg)$sigma^2
exp.lnpredictions.normal <- exp(MSE/2)*exp.lnpredictions

png(filename = "tworegscatter2.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 1), cex = 1.3)
plot(jitter(homeruns$Minor),
      jitter(homeruns$Major),
      ylab = "Major Homeruns",
      xlab = "Minor Homeruns",
      xlim = c(0, 35),
      ylim = c(0, 20),
      pch = 19,
      col = rgb(0, 0, 0, .5),
      main = "Log Transformed Model")
# Draw a line showing the model's predictions
lines(homeruns$Minor, exp.lnpredictions, col =" red", lwd = 2)
lines(homeruns$Minor, exp.lnpredictions.normal,
      col = rgb(.5, 0, 0, 1), lwd = 2, lty = 2)

plot(jitter(homeruns$Minor),
      jitter(homeruns$Major),
      ylab = "Major Homeruns",
      xlab = "Minor Homeruns",
      xlim = c(0, 35),
      ylim = c(0, 20),
      pch = 19,
      col = rgb(0, 0, 0, .5),
      main = "Untransformed Model")
abline(lm(Major ~ Minor, data = homeruns), col = "blue",lwd = 2)
dev.off()
```

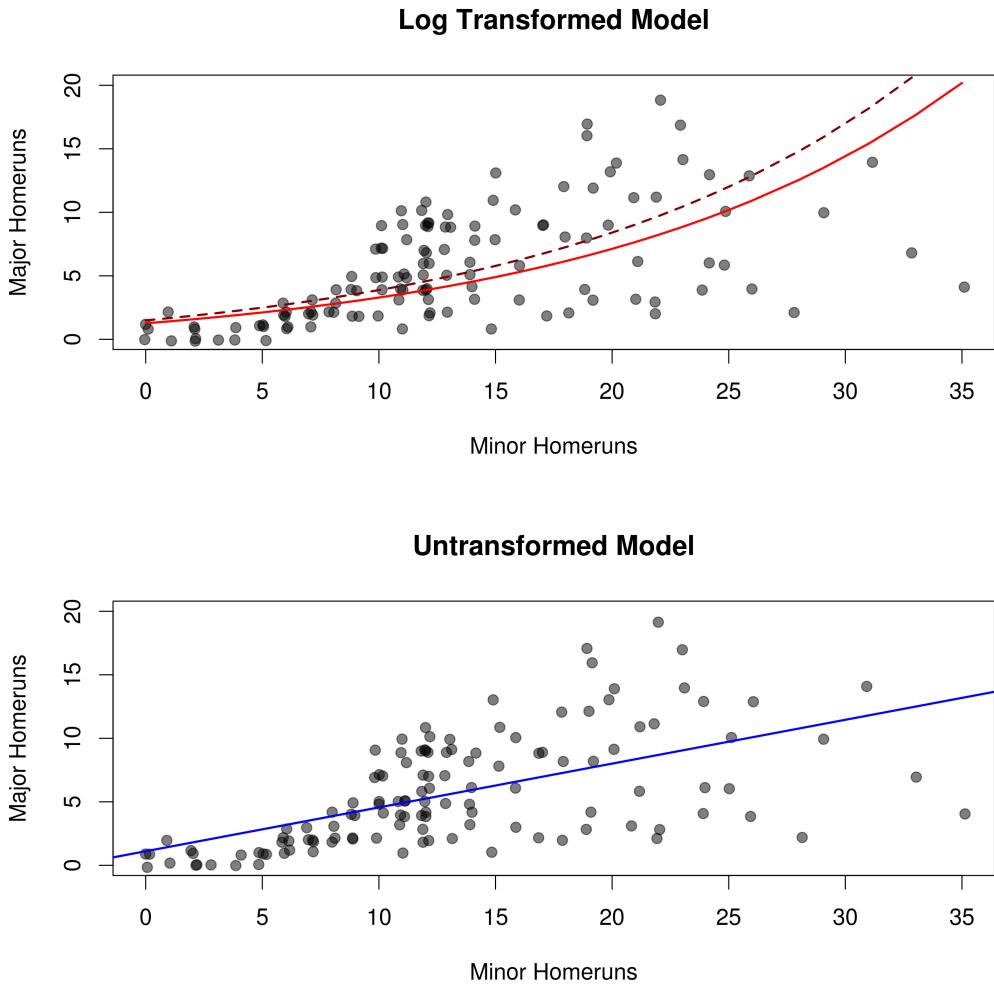


Figure 3.33: Relationship between major homeruns and minor homeruns for the logged (top) and unlogged (bottom) regressions.

As expected, the conditional arithmetic mean is greater than the conditional geometric mean. If we don't want to assume normality, we can use $\hat{\alpha}$, the expected value of e^u , where u is the error term. To obtain $\hat{\alpha}$, we regress our untransformed variable y on the exponentiated fitted values, without an intercept:

```
# Using -1 in the regression suppresses the intercept
alpha.reg <- lm(Major ~ exp.lnpredictions - 1, data = homeruns)
```

The coefficient in the regression is the $\hat{\alpha}$ term:

```
alpha.hat <- coef(alpha.reg)[1]
```

Our prediction is then:

$$\hat{y} = \hat{\alpha} e^{\log \hat{y}} \quad (3.63)$$

```
adj.preds <- alpha.hat*exp.lnpredictions
png(filename = "tworegscatter3.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 1), cex = 1.3)
plot(jitter(homeruns$Minor),
     jitter(homeruns$Major),
     ylab = "Major Homeruns",
     xlab = "Minor Homeruns",
     xlim = c(0, 35),
     ylim = c(0, 20),
     pch = 19,
     col = rgb(0, 0, 0, .5),
     main = "Log Transformed Model")
# Draw a line showing the model's predictions
lines(homeruns$Minor, exp.lnpredictions, col =" red", lwd = 2)
lines(homeruns$Minor, adj.preds, col = "green", lwd = 2, lty = 2)

plot(jitter(homeruns$Minor),
     jitter(homeruns$Major),
     ylab = "Major Homeruns",
     xlab = "Minor Homeruns",
     xlim = c(0, 35),
     ylim = c(0, 20),
     pch = 19,
     col = rgb(0, 0, 0, .5),
     main = "Untransformed Model")
abline(lm(Major ~ Minor, data = homeruns), col = "blue",lwd = 2)
dev.off()
```

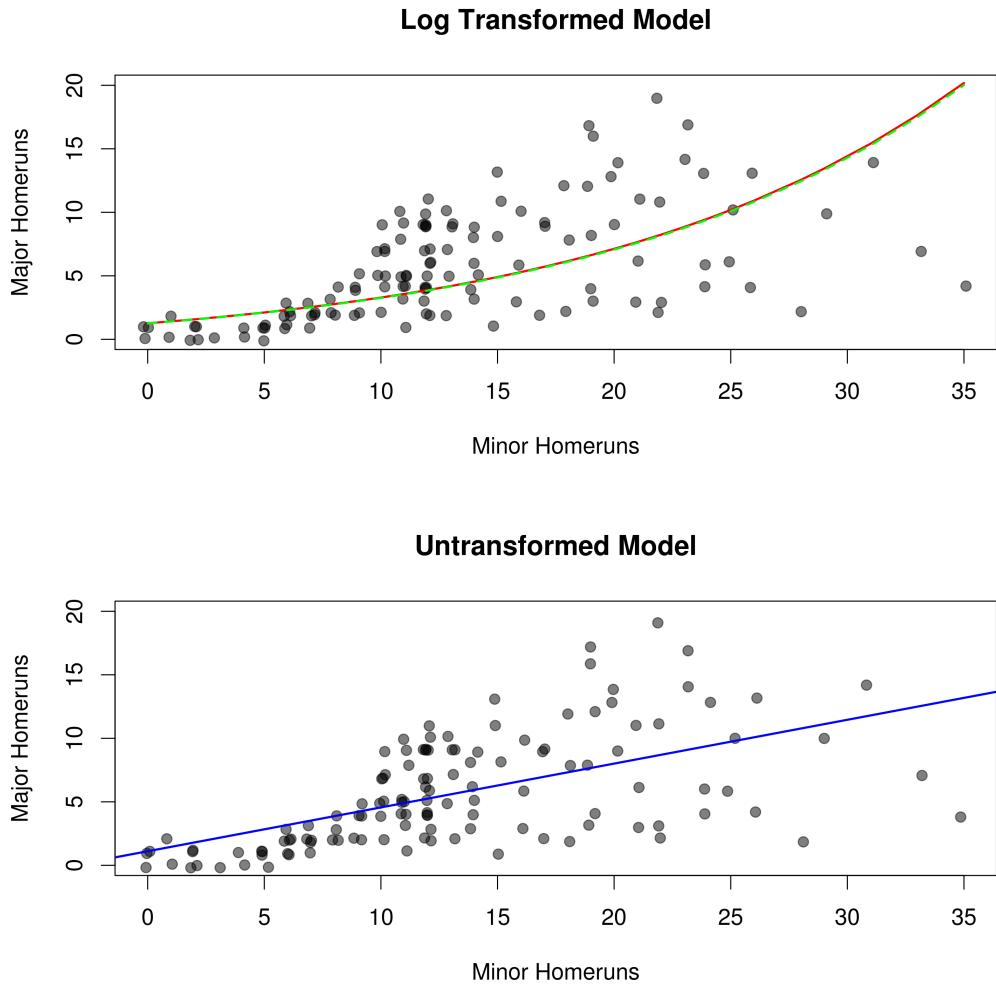


Figure 3.34: Relationship between major homeruns and minor homeruns for the logged (top) and unlogged (bottom) regressions.

Without assuming normality, our predictions for the geometric mean are basically the same as the arithmetic mean. Overall, the added difficulty of interpretation and prediction might be worth it when taking the reduction of heteroskedasticty:

```
png(filename = "tworegsspreadlevel.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(1, 2), cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(fitted(lnreg)),
     jitter(rstudent(lnreg)),
     xlab = "Fitted Values",
     ylab = "Jackknife Residuals",
     pch = 19,
```

```
col  = rgb(0, 0, 0, .3),
ylim = c(-4, 4),
xlim = c(0, 4),
main = "Log Transformed Model")
abline(h = 0)
plot(jitter(fitted(homereg)),
      jitter(rstudent(homereg)),
      xlab = "Fitted Values",
      ylab = "Jackknife Residuals",
      pch  = 19,
      col  = rgb(0, 0, 0, .3),
      ylim = c(-4, 4),
      xlim = c(0, 14),
      main = "Untransformed Model")
abline(h = 0)
dev.off()
```

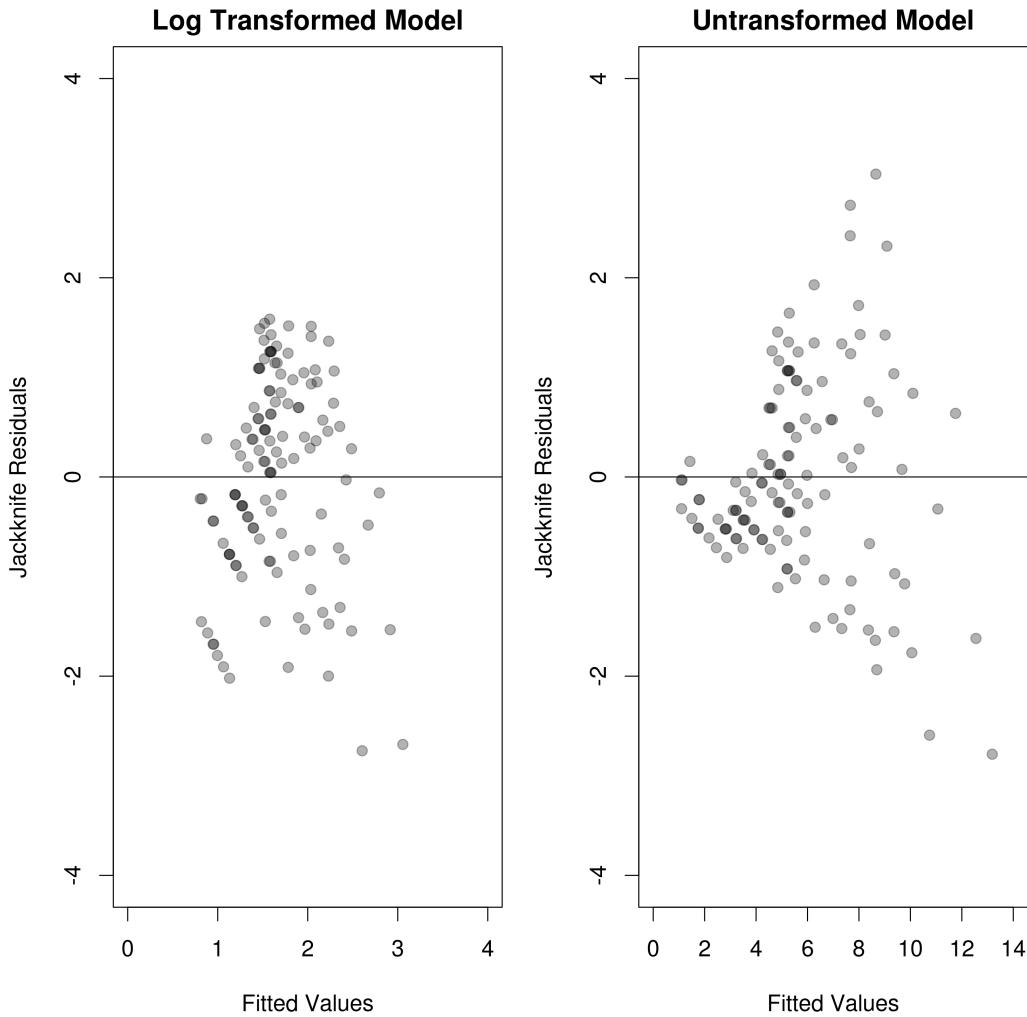


Figure 3.35: Fitted versus residual plot of logged (left) and unlogged (right) regressions.

Elasticity

One common way of interpreting coefficients, often used by economists, is the elasticity, which is the % change in the dependent variable with an equivalent % change in the independent variable. If both the dependent and independent variable are log transformed, then the coefficient on the independent variable can be interpreted as an elasticity. This follows from the line of reasoning discussed earlier, where the log of the difference between two variables is approximately the proportionate change, as long as the difference is small.

3.6.3 Robust Standard Errors

If we are only concerned about statistical inference and causality, then we'll probably see heteroskedasticity as a nuisance that messes up our standard error estimates, rather than an interesting part of our conditional distribution story. This isn't seen as much of a problem by many applied data analysts because, like in the case of outliers, there are "robust" options that allow us to ignore heteroskedasticity. These are called *robust standard errors*. In the case of homoskedastic errors, we have:

$$\text{Var}(u_i|x_i) = \text{Var}(u_i) = \sigma^2 \quad (3.64)$$

In other words, the variance of the errors given different values of x are exactly the same, and equal to a constant σ^2 . On the other hand, if we have heteroskedasticity then:

$$\text{Var}(u_i|x_i) = \sigma_i^2 \quad (3.65)$$

That is, the variance of the errors changes as a function of the regressors, as we saw in the case of the homeruns example, with the variance of the regression residuals increasing for more homeruns hit in the minors. Robust standard errors are used to get correct estimates of the variance of a parameter in the presence of heteroskedasticity. Detail on derivations is included in the mathematical appendix.

We can use the squared regression residuals \hat{u}_i^2 as an estimator of σ_i^2 . The "robust" variance estimator of $\hat{\beta}_1$ from a sample regression $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1$ is then:

$$\text{Var}(\hat{\beta}_1) = \frac{\sum_{i=1}^n (x_i - \bar{x})^2 \hat{u}_i^2}{(\sum_{i=1}^n (x_i - \bar{x})^2)^2} \quad (3.66)$$

This basically tells us that the variance of $\hat{\beta}_1$ is a function, not just of the mean squared error (σ^2) (i.e., the average variability in the residuals), which gives equal weight to residuals regardless of where they are. Instead, it is a weighted average of the squared residuals, where weights are determined by the squared difference between the value of an observation with respect to the regressor and the mean of the regressor.⁵

Why should the residuals be weighted by $(x_i - \bar{x})^2$? The reason is that the regression line must pass through \bar{x} . This is a defining property of linear regression, and can be seen from recalling the equation for the intercept in linear regression:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}_1 \quad (3.67)$$

$$\bar{y} = \hat{\beta}_0 + \hat{\beta}_1 \bar{x}_1 \quad (3.68)$$

⁵What if \hat{u}_i^2 are positively or negatively correlated with $(x_i - \bar{x})^2$?

Thus, the point (\bar{y}, \bar{x}_1) is always on the regression line. Points farther away affect the slope more than those close to \bar{x} (recall leverage). This is pretty esoteric, so let's take a look with some simulated data. Suppose the errors are positively correlated with $(x_i - \bar{x})^2$:

```

set.seed(22)
x <- runif(1000, 0, 100)
# The function takes in a vector x and spits out
# 1 + (x - mean(x))^2. This is our "heteroskedasticity" function
h = function(x) 1 + (x - mean(x))^2
error1 <- rnorm(1000, 0, h(x)) # Generate heteroskedastic errors
error2 <- rnorm(1000, 0, sd(error1)) # Generate errors with the same sd
y1 <- 20*x + error1
y2 <- 20*x + error2
lm1 <- lm(y1 ~ x)
lm2 <- lm(y2 ~ x)
png(filename = "hetsim1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(1, 2))
plot(x, y1,
      xlab = "x",
      ylab = "y",
      pch = 19,
      col = rgb(0, 0, 0, .3),
      main = "",
      ylim = c(-6000, 6000))
abline(lm1, col = "red", lwd = 4, lty = 2)
plot(x,y2,
      xlab = "x",
      ylab = "y",
      pch = 19,
      col = rgb(0, 0, 0, .3),
      main = "",
      ylim = c(-6000, 6000))
abline(lm2, col = "red", lwd = 4 , lty = 2)
dev.off()

```

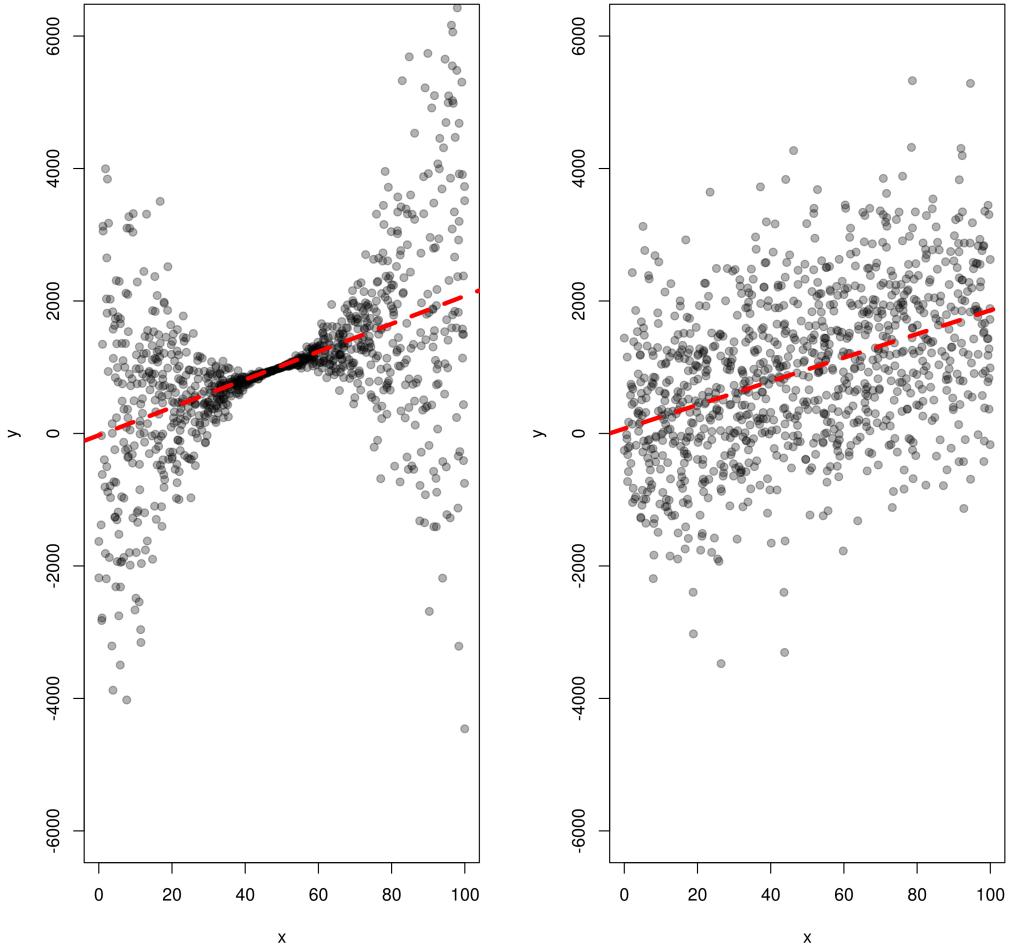


Figure 3.36: Simulated linear regression with heteroskedasticity proportional to $(x_i - \bar{x})^2$ (left), and no heteroskedasticity (right). The average variance of the errors is the same for the two datasets.

We can see that the estimated slope is exactly the same with and without heteroskedasticity and a large enough sample, showing that linear least squares is “unbiased” in the presence of heteroskedasticity.

To see why heteroskedasticity affects the variance of the slope, let’s take smaller samples and plot the regression line estimated from each sample against the large sample regression lines:

```

h = function(x) 1 + (x - mean(x))^2
int.het <- c() # Vector for heteroskedastic intercepts
int.hom <- c() # Vector for homoskedastic intercepts
slope.het <- c() # Vector for heteroskedastic slopes

```

```

slope.hom <- c() # Vector for homoskedastic slopes

for(i in 1:100){
  # For each iteration simulate data with a sample size of 50
  x <- runif(50, 0, 100)
  # Generate heteroskedastic or homoskedastic errors
  error1 <- rnorm(50, 0, h(x))
  error2 <- rnorm(50, 0, sd(error1))

  y1 <- 20*x + error1
  y2 <- 20*x + error2

  # Fit a linear model
  lm1 <- lm(y1 ~ x)
  lm2 <- lm(y2 ~ x)

  # Save the fitted parameters for each iteration
  int.het <- append(int.het,coef(lm1)[1]) # het intercept
  int.hom <- append(int.hom,coef(lm2)[1]) # hom intercept
  slope.het <- append(slope.het,coef(lm1)[2]) # het slope
  slope.hom <- append(slope.hom,coef(lm2)[2]) # hom slope
}

# Run the "population" regression with 1000 observations
x <- runif(1000, 0, 100)
h = function(x) 1 + (x - mean(x))^2
error1 <- rnorm(1000, 0, h(x))
error2 <- rnorm(1000, 0, sd(error1))
y1 <- 20*x + error1
y2 <- 20*x + error2
lm1 <- lm(y1 ~ x)
lm2 <- lm(y2 ~ x)

png(filename = "hetsim2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(1, 2))
plot(x, y1,
      xlab ="x",
      ylab ="y",
      pch  = 19,
      col  = rgb(0, 0, 0, .3),
      main = "",
      ylim = c(-6000, 6000))

```

```

# Generate a line for each "sample" regression estimate
for(i in 1:100){
  abline(int.het[i], slope.het[i], col = rgb(0, 0, 1, .2), lwd = 2)
}
abline(lm1, col = "red", lwd = 4, lty = 2)

plot(x, y2,
      xlab = "x",
      ylab = "y",
      pch = 19,
      col = rgb(0, 0, 0, .3),
      main = "",
      ylim = c(-6000, 6000))

for(i in 1:100){
  abline(int.hom[i], slope.hom[i], col = rgb(0, 0, 1, .2), lwd = 2)
}
abline(lm2, col = "red", lwd = 4, lty = 2)
dev.off()

```

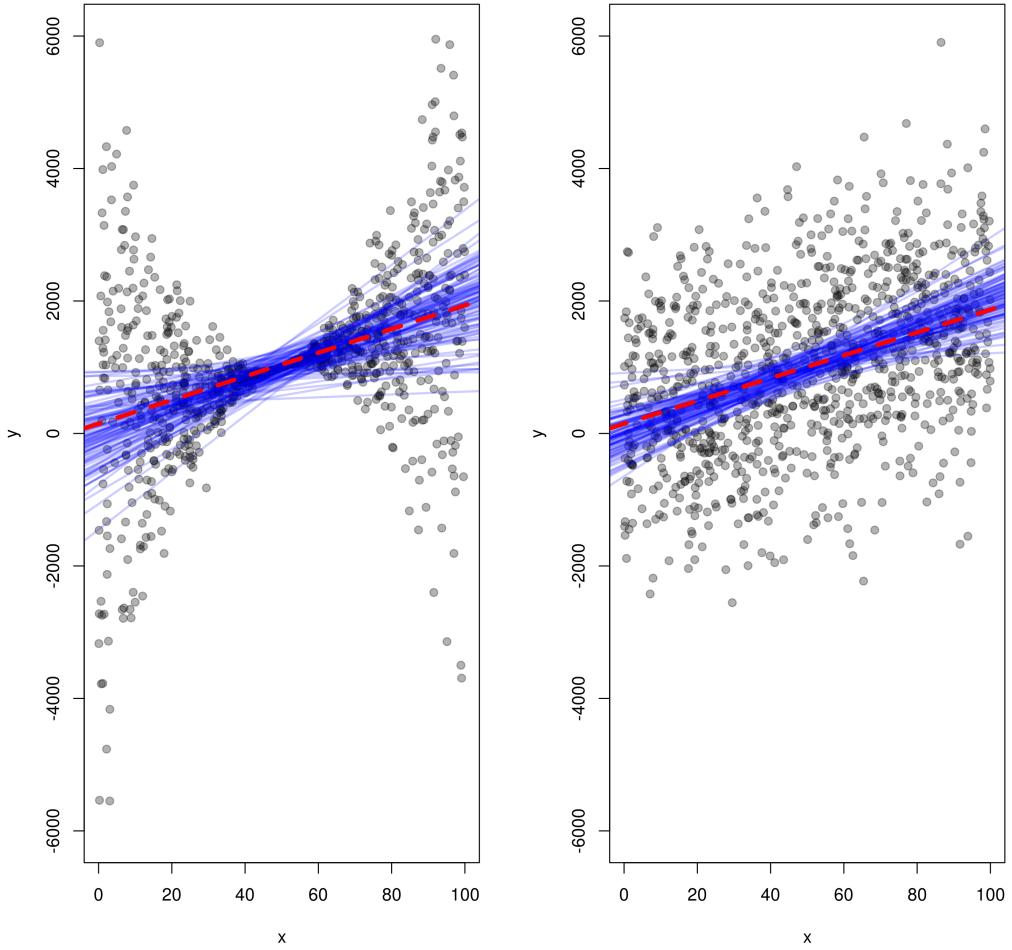


Figure 3.37: Simulated linear regression with heteroskedasticity proportional to $(x_i - \bar{x})^2$ (left) and homoskedasticity (right). Small sample ($n = 50$) regression lines are plotted in blue.

We can see that in the heteroskedastic case sometimes we get values to the left of \bar{x} that are very small and to the right of \bar{x} that are very large. A regression on these data will have a steep positive slope. Conversely, we may also get values to the left of \bar{x} that are very large and to the right of \bar{x} that are very small, leading to a regression with a steep *negative* slope.

The points in the middle near \bar{x} do little to help pin down the slope of the line, consistent with our previous results that leverage increases with an observation's distance from \bar{x} . Thus, in the homoskedastic case, points further from \bar{x} provide more information about the slope than those close to \bar{x} , but in the heteroskedastic case, this greater information is damped by the greater

errors as $(x_i - \bar{x})^2$ increases.

We can calculate the robust standard errors using the regression residuals as weights. First, notice that the “classical” or usual standard errors are about the same, about 1.28:

```
lm1 <- lm(y1 ~ x)
lm2 <- lm(y2 ~ x)
summary(lm1)
summary(lm2)
```

Now, calculate the robust standard errors using $\sigma_{robust} = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2 \hat{u}_i^2}{(\sum_{i=1}^n (x_i - \bar{x})^2)^2}}$:

```
lm1.resid.sq <- resid(lm1)^2 # Get the squared residual
lm2.resid.sq <- resid(lm2)^2
x.sq.dist <- (x - mean(x))^2 # Find each squared distance from x.bar
# Calculate the robust standard error
lm1.robust <- sqrt((sum(lm1.resid.sq*x.sq.dist))/sum(x.sq.dist)^2)
lm2.robust <- sqrt((sum(lm2.resid.sq*x.sq.dist))/sum(x.sq.dist)^2)
```

The robust standard error (1.9) is larger than the classical standard error (1.3) in the heteroskedastic case, and about the same as the classical standard error in the homoskedastic case (1.3). We can also use the sandwich and lmtest packages to calculate the robust standard errors for us, but I advise against doing this in the near future to instead get a grasp of the computations ([Zeileis, 2006](#)):

```
install.packages("sandwich", repos = "http://lib.stat.cmu.edu/R/CRAN/")
install.packages("lmtest", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(sandwich)
library(lmtest)
# There's a bunch of different heteroskedasticity robust estimators
# Here we use HC0, the others have different corrections
coeftest(lm1,
          vcov = vcovHC(lm1, type = "HC0"),
          df = df.residual(lm1)) # residual df is n - number of parameters
coeftest(lm2,
          vcov = vcovHC(lm2, type = "HC0"),
          df = df.residual(lm2))
```

The robust standard errors from the sandwich package match our estimate fairly closely. Recall that the effect of heteroskedasticity depends on the correlation between $(x_i - \bar{x})^2$ and σ_i^2 . If the correlation is positive, the robust standard errors are larger than the classical ones, as we just saw. What if the correlation is negative?

With multiple regressors we have a matrix of variances and covariances between the different parameters:

$$\text{Var}(\hat{\beta}) = E((\hat{\beta} - E(\hat{\beta}))(\hat{\beta} - E(\hat{\beta}))^t) \quad (3.69)$$

Some algebra (in mathematical appendix) shows that the robust variance covariance matrix is:

$$\text{Var}(\hat{\beta}) = (X'X)^{-1} X' \hat{e} \hat{e}' X (X'X)^{-1} \quad (3.70)$$

The matrix $\hat{e} \hat{e}'$ comes from taking the outer product of the regression residual \hat{e} for each observation with itself, then setting all the off-diagonal elements to zero (this is equivalent to assuming no autocorrelation between different observations). Using our previous example, we can calculate the robust variance-covariance matrix and compare to the output of coeftest:

```
x <- runif(1000, 0, 100)
h = function(x) 1 + (x - mean(x))^2
error1 <- rnorm(1000, 0, h(x))
error2 <- rnorm(1000, 0, sd(error1))
y1 <- 20*x + error1
y2 <- 20*x + error2
lm1 <- lm(y1 ~ x)
lm2 <- lm(y2 ~ x)

# Create residual
e1 <- y1 - predict(lm1)

# Create outer product 1000 x 1000 matrix
ee.t1 <- e1 %*% t(e1)

# Set off diagonal to zero
ee.t1 <- diag(diag(ee.t1))

# Calculate the robust variance covariance matrix
x <- cbind(rep(1, length(x)), x) # add an intercept to x
rob.var1 <- solve(t(x) %*% x) %*% ee.t1 %*% x %*% solve(t(x) %*% x)
```

If we take the square root of the diagonal of the robust variance-covariance matrix we should get the same result as coeftest for the standard errors of the parameter estimates:

```
rob.se <- sqrt(diag(rob.var1))
coeftest(lm1,
        vcov = vcovHC(lm1, type = "HC0"),
        df = df.residual(lm1)) # residual df is n - number of parameters
```

3.6.4 Weighted Least Squares

Robust standard errors make it so that we don't have to worry about understanding the functional form of the heteroskedasticity. As long as we have enough data, robust standard errors will approximate the true variance function of the errors, no matter the form of heteroskedasticity. However, looking at our data and understanding the functional form of the heteroskedasticity, and thinking about why it is occurring, might give us some insights into the actual data generating process. Weighted least squares is an old school way of doing this, where we take a guess at the functional form of the heteroskedasticity, then adjust our modeling to account for this. For example, suppose the heteroskedasticity has the following form:

$$\text{Var}(u_i|x_i) = \sigma^2 h(x_i) \quad (3.71)$$

Where $h(x)$ is some function we think might be the form of the heteroskedasticity. A simple example would be $h(x) = x$, meaning the variance of the errors increases linearly with x . If we understand the errors well, then we might be able to guess $h(x)$. If this is true, then all we have to do is divide the dependent and independent variables by $\sqrt{h(x)}$. This gives us:

$$\text{Var}\left(\frac{u_i}{\sqrt{h(x_i)}}|x_i\right) = E\left(\left(\frac{u_i}{\sqrt{h(x_i)}}\right)^2\right) = \frac{1}{h(x_i)}E(u_i^2) = \frac{\sigma^2 h(x_i)}{h(x_i)} = \sigma^2 \quad (3.72)$$

Thus, if we divide our regression function by $\sqrt{h(x)}$, the variance of the errors will be constant σ^2 . This is called the weighted least squares estimator (see Wooldridge (2009) chapter 8 for more). The estimator for $\hat{\beta}$ in matrix notation is:

$$\hat{\beta} = (X'W^{-1}X)^{-1}X'W^{-1}Y \quad (3.73)$$

where W is a matrix with $h(x)$ along the diagonal. In the homeruns example, we can either divide everything by the square root of the minor home runs or use the matrix form directly:

```
# Calculate WLS by matrices
x <- cbind(rep(1, nrow(homeruns)), homeruns$Minor)
y <- homeruns$Major
w <- diag(homeruns$Minor + 1)
beta.hat <- solve(t(x) %*% solve(w) %*% x) %*% t(x) %*% solve(w) %*% y

# Transform regressors directly
homeruns$Major.div <- homeruns$Major/sqrt(homeruns$Minor + 1)
homeruns$Minor.div <- homeruns$Minor/sqrt(homeruns$Minor + 1)
homeruns$intercept <- 1/sqrt(homeruns$Minor + 1)
```

```
homereg.wls <- lm(Major.div ~ intercept + Minor.div - 1, data = homeruns)
summary(homereg.wls)
```

Although the approach is good for trying to generate hypotheses about the causes of heteroskedasticity, there is a drawback: Because the regression function is divided by $\sqrt{h(x)}$, the interpretation of each coefficient becomes very difficult. However, we can rely on an unadjusted OLS to interpret parameters, but use the weighted least squares and associates standard errors for statistical inference.

3.6.5 Heteroskedasticity and Interactions

Now that we have some intuition about both interactions and heteroskedasticity, we can see how heteroskedasticity can result from hidden interactions. Here's one stylized example, where the true regression function includes an interaction that is omitted:

```
x <- runif(100, 0, 10)
z <- rbinom(100, 1, 0.5) # Binomial random variable (coin flip)
e <- rnorm(100, 0, 3)
y1 <- x + z + 2*x*z + e # Notice there is an interaction between x and z
reg1 <- lm(y1 ~ x) # Interaction is omitted
reg1.i <- lm(y1 ~ x + z + x*z) # Interaction is included
```

This leads to apparent (but not real) heteroskedasticity, which is diagnosable from a plot of the jackknife residuals versus fitted values. Looking at the plot of y_1 on x :

```
png(filename = "hidden1.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 2), mar = c(5, 4, 1, 1), cex = 1.3)
plot(x,y1,
      main = "",
      col = rgb(0, 0, 0, .5),
      pch = 19)
abline(reg1,
       col = "blue")
plot(x, y1,
      col = rgb(0, 0, 0, .5),
      pch = 19)
abline(coef(reg1.i)[1],
       coef(reg1.i)[2],
       col = "red",
       pch = 19)
```

```

abline(coef(reg1.i)[1] + coef(reg1.i)[3],
       coef(reg1.i)[2] + coef(reg1.i)[4],
       col = "red")
plot(fitted(reg1), rstudent(reg1),
      pch = 19,
      col = rgb(0, 0, 0, .5),
      ylim = c(-3, 3),
      ylab = "Jackknife Residuals",
      xlab = "Fitted Values")
plot(fitted(reg1.i), rstudent(reg1.i),
      pch = 19,
      col = rgb(0, 0, 0, .5),
      ylim = c(-3, 3),
      ylab = "Jackknife Residuals",
      xlab = "Fitted Values")
dev.off()

```

In bivariate regression we could have spotted the interaction from the plot, but in multivariate regression we would have needed to rely on the spread-level plot or other diagnostics. Here's another example:

```

x <- runif(100, -10, 10)
z <- rbinom(100, 1, 0.5)
e <- rnorm(100, 0, 3.5)
y2 <- x + z - 2*x*z + e
reg2 <- lm(y2 ~ x)
reg2.i <- lm(y2 ~ x + z + x*z)

```

Looking at the plot of y_1 on x :

```

png(filename = "hidden2.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 2), mar = c(5, 4, 1, 1), cex = 1.3)
plot(x, y2,
      pch = 19,
      col = rgb(0, 0, 0, .5))
abline(reg2, col = "blue")
plot(x, y2,
      pch = 19,
      col = rgb(0, 0, 0, .5))
abline(coef(reg2.i)[1], coef(reg2.i)[2], col="red")
abline(coef(reg2.i)[1] + coef(reg2.i)[3],
       coef(reg2.i)[2] + coef(reg2.i)[4], col="red")
plot(fitted(reg2), rstudent(reg2),
      pch = 19,

```

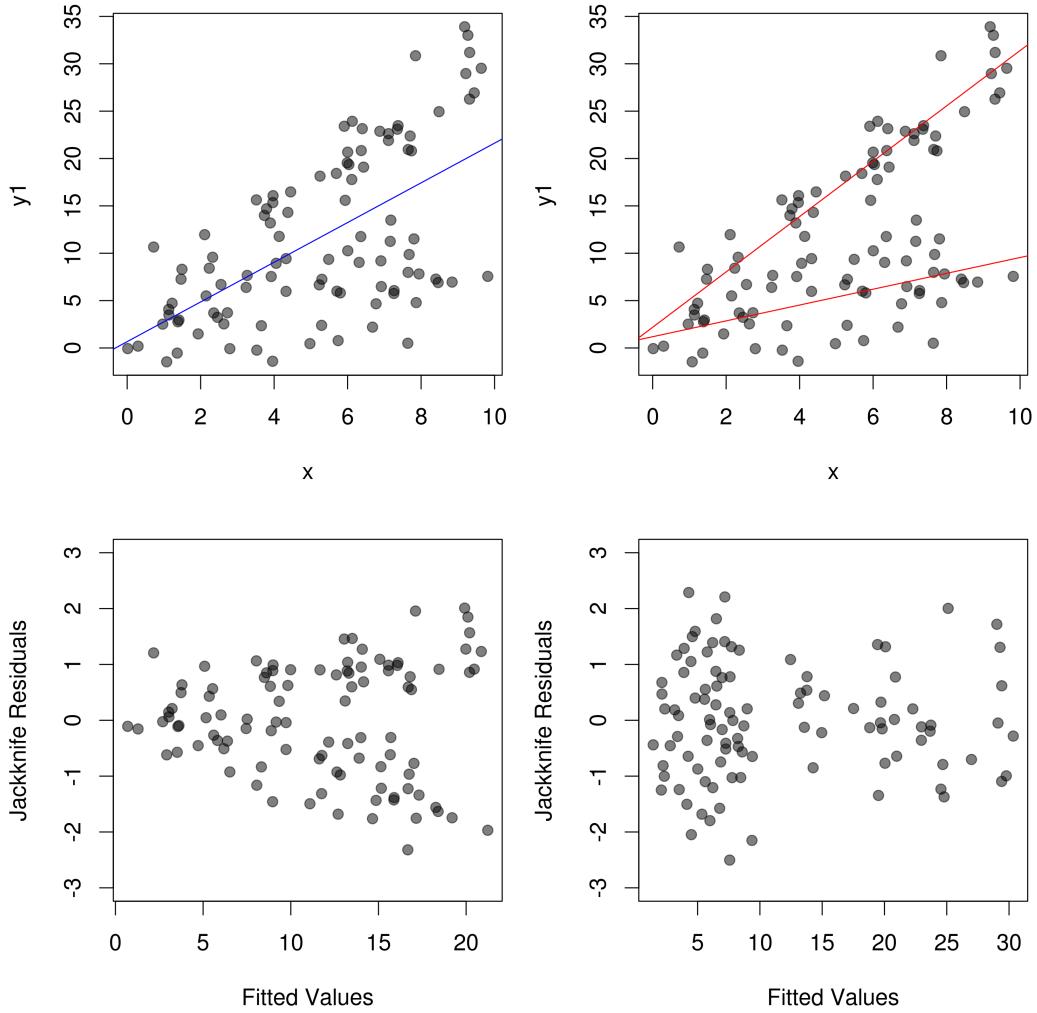


Figure 3.38: Heteroskedasticity resulting from an omitted interaction.

```

col  = rgb(0, 0, 0, .5),
ylim = c(-3, 3),
ylab = "Jackknife Residuals", xlab = "Fitted Values")
plot(fitted(reg2.i), rstudent(reg2.i),
      pch  = 19,
      col  = rgb(0, 0, 0, .5),
      ylim = c(-3, 3),
      ylab = "Jackknife Residuals",
      xlab = "Fitted Values")
dev.off()

```

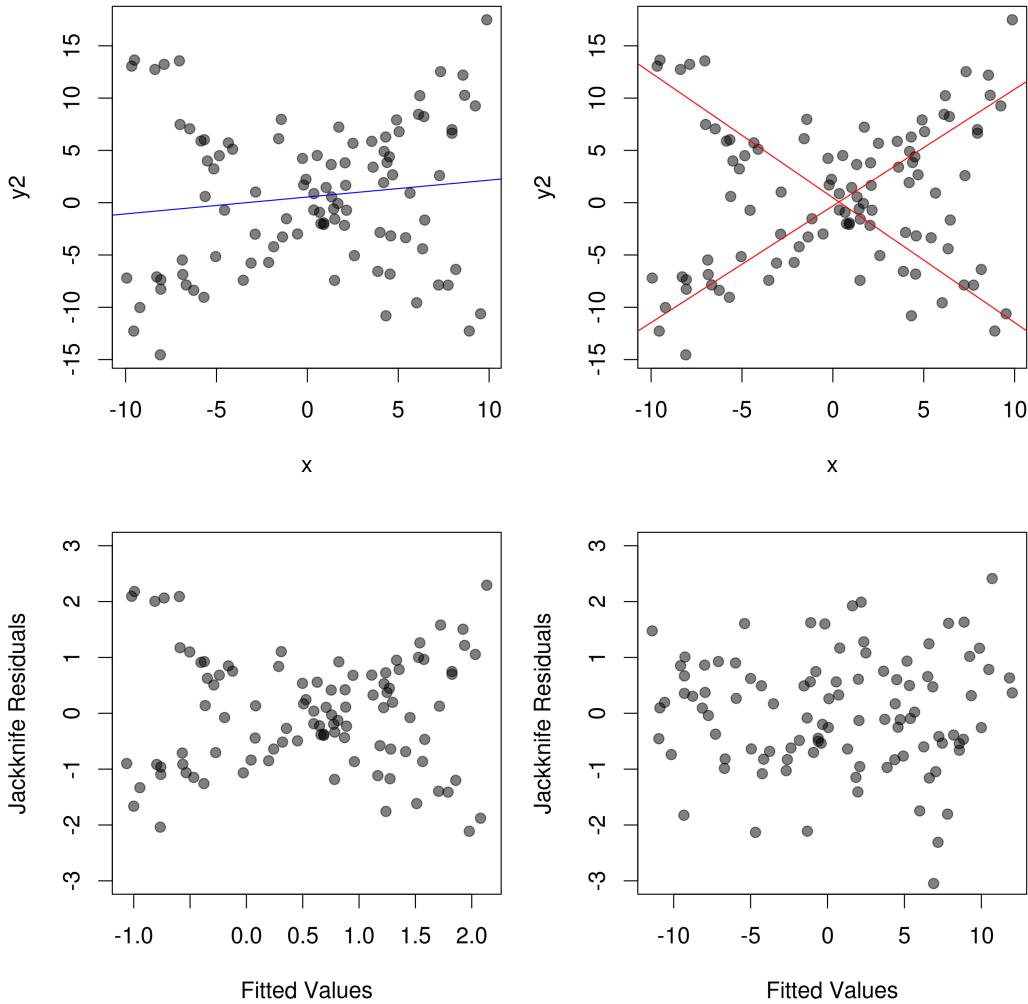


Figure 3.39: Heteroskedasticity resulting from an omitted interaction.

In general, the presence of heteroskedasticity, as indicated by a difference between robust and regular standard errors, should indicate that something is wrong with the model, and may help us diagnose the problem. If robust standard errors are different from classical ones, then this suggests model misspecification, and looking deeper into improving our model. [King and Roberts \(2013\)](#) give several examples of how to do this. For example, if we have a skewed error distribution, this will also show up as a difference between robust and classical standard errors. Suppose we have the following population regression function:

```
set.seed(36)
x <- rnorm(1000, 8, 1)
z <- rnorm(1000, 5, 1) + 0.5*x
```

```

error <- rnorm(1000, 0, 1)
m <- x + x^2 + 5*z + error
y <- m^2
lm1 <- lm(y ~ x + I(x^2) + z)

```

Using our data summary and conditional distribution stories, we should easily detect the problematic model:

```

png(filename = "incdistribution.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 2), mar = c(5, 4, 1, 1), cex = 1.3)
hist(y, breaks = "FD", ylim = c(0, 100), xlim = c(0, 40000))
qqnorm(y, pch = 19, col = rgb(0, 0, 0, .4))
qqline(y)
plot(fitted(lm1), rstudent(lm1),
      ylab = "Jackknife Residuals",
      xlab = "Fitted Values",
      pch = 19,
      col = rgb(0, 0, 0, .4))
qqPlot(lm1,
       ylab = "Jackknife Residuals",
       pch = 19,
       col = rgb(0, 0, 0, .4))
dev.off()

```

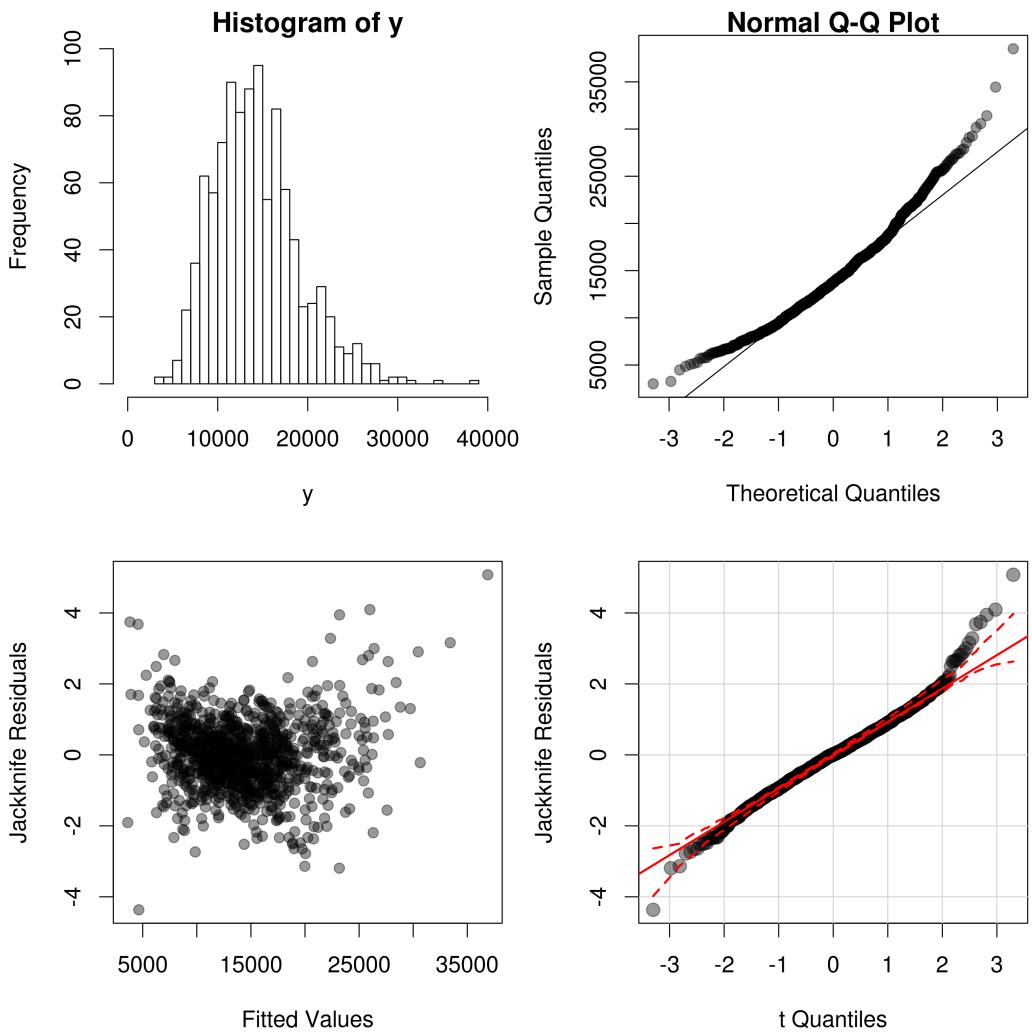


Figure 3.40: The histogram of y , qqplot of y against the normal distribution, the jackknife residuals versus fitted values, and qqplot of the jackknife regression residuals versus the quantiles of the t distribution, all show potential model misspecification.

However, what if we didn't do this, and instead ran our regression and used "robust" standard errors?

```
coeftest(lm1)
coeftest(lm1,
  vcov = vcovHC(lm1, type = "HC0"),
  df = df.residual(lm1)) # residual df is n - number of parameters
```

The robust standard error estimate for the intercept, x , and x^2 are about twice as large as the classical standard errors. This should tell us that our

model is misspecified. Only under limited circumstances where the model is correctly specified will differences between classical and robust standard errors only be the result of heteroskedasticity.

If we try the Box-Cox transformation on y we get:

```
boxCox(y ~ x + I(x^2) + z, plotit = TRUE)
bc <- boxCox(y ~ x + I(x^2) + z, plotit = TRUE)
bc$x[bc$y == max(bc$y)]
```

Which reaches a maximum around $\lambda = 0.5$, the correct square root transformation. If we respecify our model:

```
y.lambda <- sqrt(y)
lm2 <- lm(y.lambda ~ x + I(x^2) + z)
png(filename = "incdistribution2.png", width = 3000, height = 3000, res = 300)
par(mfrow = c(2, 2), mar = c(5, 4, 1, 1), cex = 1.3)
hist(y.lambda, breaks = "FD")
qqnorm(y.lambda, pch = 19, col = rgb(0, 0, 0, .4))
qqline(y.lambda)
plot(fitted(lm2), rstudent(lm2),
      ylab = "Jackknife Residuals",
      xlab = "Fitted Values",
      pch = 19,
      col = rgb(0, 0, 0, .4))
qqPlot(lm2,
       ylab = "Jackknife Residuals",
       pch = 19,
       col = rgb(0, 0, 0, .4))
dev.off()
```

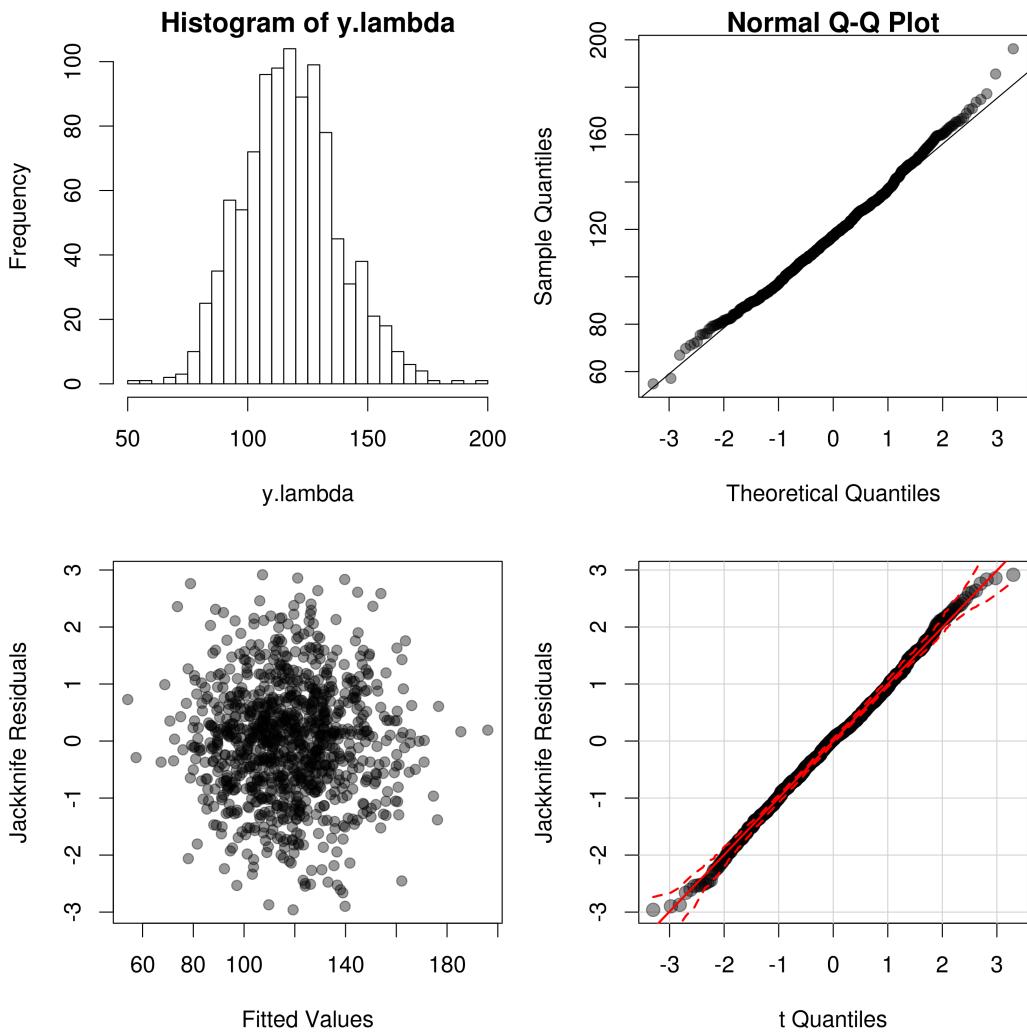


Figure 3.41: The histogram of \sqrt{y} , qqplot of \sqrt{y} against the normal distribution, the jackknife residuals versus fitted values, and qqplot of the jackknife regression residuals versus the quantiles of the t distribution.

Now, there's only a small difference between our robust and classical standard errors:

```
coeftest(lm2)
coeftest(lm2,
         vcov = vcovHC(lm2, type = "HC0"),
         df = df.residual(lm2)) # residual df is n - number of parameters
```

Robust Standard Errors Summary

Using robust standard errors to not have to worry about model misspecification, and failing to look at the variance of residuals by fitted

values means we don't really care about our data. This problem gets worse when we consider models that are non-linear in parameters. As Halbert White (the creator of robust standard errors) himself stated ([White, 1980](#)):

“Until now, one had either to model heteroskedasticity correctly or suffer the consequences. The fact that the covariance matrix estimator and heteroskedasticity test given here do not require formal modeling of the heteroskedastic structure is a great convenience, but it does not relieve the investigator of the burden of carefully specifying his models. Instead, it is hoped that the statistics presented here will enable researchers to be even more careful in specifying and estimating econometric models.”

3.7 Summary

This chapter covered the Gauss-Markov assumptions to give you an idea of the ideal situation under which linear regression is interpreted, which probably never happens. We've seen the difficulty with the zero conditional mean assumption because of issues like omitted variables, difficult to detect non-linearities, and hidden interactions. We've looked at how different diagnostic plots, like the component plus residual plot and residual vs. fitted plots, can help us get insight into when and why a linear model fails, and how to improve it. We've seen how robust standard errors can cloak good information about a model's misspecification. This chapter aimed to give you the basic tools to take the “all models are wrong” idea seriously. Hopefully you take these tools seriously by caring about regression residuals.

3.8 Mathematical Appendix

3.8.1 Proof that OLS estimators are unbiased

Here's the standard proof that ordinary least squares (OLS) is unbiased. To find the minimum, take the derivative with respect to $\hat{\beta}_0$ and $\hat{\beta}_1$, set it equal to zero, and solve:

$$\frac{\partial \sum_{i=1}^n \hat{u}_i^2}{\partial \hat{\beta}_0} = 2 \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1}) = 0 \quad (3.74)$$

$$\iff \sum_{i=1}^n y_i - n\hat{\beta}_0 - \sum_{i=1}^n \hat{\beta}_1 x_{i1} = 0 \quad (3.75)$$

$$\iff \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}_1 \quad (3.76)$$

Which uses “ybar” as the mean of y : $\frac{\sum_{i=1}^n y_i}{n} = \bar{y}_i$. Using this, let's find $\hat{\beta}_1$:

$$\frac{\partial \sum_{i=1}^n \hat{u}_i^2}{\partial \hat{\beta}_1} = 2 \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1}) x_{i1} = 0 \quad (3.77)$$

$$\iff \sum_{i=1}^n (y_i - \bar{y}_i + \hat{\beta}_1 \bar{x}_{i1} - \hat{\beta}_1 x_{i1}) x_{i1} = 0 \quad (3.78)$$

$$\iff \sum_{i=1}^n x_{i1} (y_i - \bar{y}_i) = \hat{\beta}_1 \sum_{i=1}^n x_{i1} (x_{i1} - \bar{x}_{i1}) \quad (3.79)$$

$$\iff \hat{\beta}_1 = \frac{\sum_{i=1}^n x_{i1} (y_i - \bar{y}_i)}{\sum_{i=1}^n x_{i1} (x_{i1} - \bar{x}_{i1})} \quad (3.80)$$

We can write this more compactly using the following fact:

$$\sum_{i=1}^n x_{i1} (x_{i1} - \bar{x}_{i1}) = \sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2 \quad (3.81)$$

Giving us:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1}) y_i}{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2} \quad (3.82)$$

To show unbiasedness, we must show $E(\hat{\beta}_1) = \beta_1$. Notice the following:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})y_i}{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2} \quad (3.83)$$

$$= \frac{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})(\beta_0 + \beta_1 x_{i1} + u_i)}{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2} \quad (3.84)$$

$$= \beta_0 \frac{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})}{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2} + \beta_1 \frac{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})x_{i1}}{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2} + \frac{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})u_i}{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2} \quad (3.85)$$

$$= \beta_1 + \frac{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})u_i}{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2} \quad (3.86)$$

(3.87)

Which uses the fact that $\sum_{i=1}^n (x_{i1} - \bar{x}_{i1}) = 0$ and, again, $\sum_{i=1}^n x_{i1}(x_{i1} - \bar{x}_{i1}) = \sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2$. Taking the expectation:

$$E(\hat{\beta}_1) = E(\beta_1 + \frac{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})u_i}{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2}) \quad (3.88)$$

$$= \beta_1 + E(\frac{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})u_i}{\sum_{i=1}^n (x_{i1} - \bar{x}_{i1})^2}) \quad (3.89)$$

Because of the zero conditional mean assumption, $E(u_i|x_i) = 0$. Thus, $E(\hat{\beta}_1) = \beta_1$ and $\hat{\beta}_1$ is unbiased.

3.8.2 Percent Change Interpretation of the Log Transform

Log-Level

Why does this percent change interpretation make sense? Consider the following fact:

$$\log(v_i) - \log(v_j) = \log(v_i/v_j) \simeq \frac{v_i - v_j}{v_j} \quad (3.90)$$

For example, $\log(8) - \log(7) \simeq 0.13 \simeq (8 - 7)/7 \simeq 0.13$. This is not true for large differences in v_i and v_j . $\frac{v_i - v_j}{v_j}$ gives the proportionate change, and $100 \times \frac{v_i - v_j}{v_j}$ gives the % change. To understand how this works in regression, consider two equations:

$$\log(y_1) = \beta_0 + \beta_1 x_1 \quad (3.91)$$

$$\log(y_2) = \beta_0 + \beta_1(x_1 + 1) \quad (3.92)$$

The second equation increases x_1 by one unit. So, the proportionate change in y with a 1 unit increase in x_1 is:

$$\frac{y_2 - y_1}{y_1} \simeq \log(y_2) - \log(y_1) = \beta_0 + \beta_1 x_1 - (\beta_0 + \beta_1 x_1 + \beta_1) = \beta_1 \quad (3.93)$$

The proportionate change in y is approximately β_1 , and the % change y is approximately $100 \times \beta_1$. In terms of our homerun example, the proportionate change in homeruns is approximately 0.056 for each additional minor homerun, and the percent change is 5.6%.

Level-Log

In the case where y is in levels and x is in logs we have:

$$y^0 = \beta_0 + \beta_1 \log x \quad (3.94)$$

If we increase x by 100% (multiply by 2), we get:

$$y^1 = \beta_0 + \beta_1 \log 2x \quad (3.95)$$

Thus, the change in y is:

$$y^1 - y^0 = \beta_0 + \beta_1 \log 2x - (\beta_0 + \beta_1 \log x) = \beta_1 \log 2 \quad (3.96)$$

3.8.3 Robust Standard Errors

Recall that the estimator of $\hat{\beta}$ is:

$$\hat{\beta}_1 = \beta_1 + \frac{\sum_{i=1}^n (x_i - \bar{x}) u_i}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (3.97)$$

If the Gauss-Markov assumptions hold, then:

$$E(\hat{\beta}_1) = \beta_1 + E\left(\frac{\sum_{i=1}^n (x_i - \bar{x}) u_i}{\sum_{i=1}^n (x_i - \bar{x})^2}\right) = \beta_1 + E(u_i) \frac{\sum_{i=1}^n (x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (3.98)$$

Because $E(u_i|x) = E(u_i) = 0$ under the zero conditional mean assumption, our estimator $\hat{\beta}_1$ is unbiased (i.e., $E(\hat{\beta}_1) = \beta_1$).

We can then find the variance of $\hat{\beta}_1$:

$$\text{Var}(\hat{\beta}_1) = \text{Var}\left(\beta_1 + \frac{\sum_{i=1}^n (x_i - \bar{x}) u_i}{\sum_{i=1}^n (x_i - \bar{x})^2}\right) \quad (3.99)$$

$$= \frac{\text{Var}(\sum_{i=1}^n (x_i - \bar{x}) u_i)}{(\sum_{i=1}^n (x_i - \bar{x})^2)^2} \quad (3.100)$$

$$= \frac{\sum_{i=1}^n (x_i - \bar{x})^2 \sigma_i^2}{(\sum_{i=1}^n (x_i - \bar{x})^2)^2} \quad (3.101)$$

In the homoskedastic case, $\sigma_i^2 = \sigma^2$ for all i . Thus, σ^2 is a constant, and one set of $\sum_{i=1}^n (x_i - \bar{x})^2$ cancels, leaving $\text{Var}(\hat{\beta}_1) = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2}$. However, in the heteroskedastic case, the degree of deviation of x_i from \bar{x} is associated with the size of the squared errors, σ_i^2 .

3.8.4 Robust Standard Errors in Matrix Form

With multiple regressors we have a matrix of variances and covariances between the different parameters:

$$\text{Var}(\hat{\beta}) = E((\hat{\beta} - E(\hat{\beta}))(\hat{\beta} - E(\hat{\beta}))^t) \quad (3.102)$$

If $E(\hat{\beta}) = \beta$, this simplifies to:

$$\text{Var}(\hat{\beta}) = E((\hat{\beta} - \beta)(\hat{\beta} - \beta)^t) \quad (3.103)$$

Note that, if our population regression function is $Y = X\beta + e$, then:

$$\hat{\beta} = (X'X)^{-1}X'Y = (X'X)^{-1}X'(X\beta + e) \quad (3.104)$$

Multiplying this out reduces to:

$$\hat{\beta} = (X'X)^{-1}X'Y = \beta + (X'X)^{-1}X'e \quad (3.105)$$

Thus, the β cancel from each side, so our variance covariance matrix reduces to:

$$\text{Var}(\hat{\beta}) = E((X'X)^{-1}X'e e' X(X'X)^{-1}) \quad (3.106)$$

Because the X 's are treated as fixed, we can push the expected value operator through:

$$\text{Var}(\hat{\beta}) = (X'X)^{-1}X'E(ee')X(X'X)^{-1} \quad (3.107)$$

In the homoskedastic case $E(ee') = I\sigma^2$, so this further reduces to:

$$\text{Var}(\hat{\beta}) = \sigma^2(X'X)^{-1} \quad (3.108)$$

If we allow for heteroskedasticity, then we estimate $E(ee')$ with the regression residuals, giving us $E(\hat{e}\hat{e}')$. This is a matrix with the estimated variance of the errors on the diagonal and zeros off the diagonal. Thus, the robust variance covariance matrix is:

$$\text{Var}(\hat{\beta}) = (X'X)^{-1}X'\hat{e}\hat{e}'X(X'X)^{-1} \quad (3.109)$$

Taking the square root of the diagonal of the matrix gives us our standard errors for each estimated coefficient.

3.8.5 Box-Tidwell Transformation

This explanation follows [Box and Tidwell \(1962\)](#) section 2. Suppose we have $y_1, y_2 \dots y_n$ as a random sample from some population with associated conditions $x_1, x_2 \dots x_n$ as the independent variable. Also suppose that:

$$E(y) = \eta + \epsilon \quad (3.110)$$

and $\text{Var}(\epsilon) = \sigma^2$. We want to represent the response function η by some transformation of x multiplied by our β vector such that:

$$\eta = f(x^\lambda, \beta) = \beta x^\lambda \quad (3.111)$$

We start with λ^0 as our first guess of λ , then repeat the transformation until λ^i converges to λ . It makes the most sense to start with $\lambda = 1$ to see if we need any transformation at all, and under the same reasoning, use our initial guess for β as the coefficient from our first regression β^0 . We can then approximate η using the first order Taylor expansion of x^λ around our guess λ^0 :

$$\eta \approx \sum_{n=0}^1 \frac{f(x^{\lambda_0}, \beta^0)^{(n)}}{n!} (\lambda - \lambda^0)^n = f(x^{\lambda_0}, \beta^*) + (\lambda - \lambda^0) \left(\frac{\partial f(x^{\lambda_0}, \beta^0)}{\partial \lambda_0} \right) \quad (3.112)$$

In our case we are assuming $f(x^{\lambda_0}, \beta^0)$ has a power functional form $f(x^{\lambda_0}, \beta^0) = \beta^0 x^{\lambda_0}$, so:

$$\frac{\partial f(x^{\lambda_0}, \beta^0)}{\partial \lambda_0} = \beta^0 x^{\lambda_0} \ln(x) \quad (3.113)$$

This gives us our first Taylor approximation:

$$\eta \approx \sum_{n=0}^1 \frac{f(x^{\lambda_0}, \beta^0)^{(n)}}{n!} (\lambda - \lambda^0)^n = f(x^{\lambda_0}, \beta^*) + (\lambda - \lambda^0) \beta^0 x^{\lambda_0} \ln(x) \quad (3.114)$$

We can solve for our next estimate of λ , call it λ^1 , using the fact that if we estimate η by linear regression we get:

$$\eta \approx \sum_{n=0}^1 \frac{f(x^{\lambda_0}, \beta^0)^{(n)}}{n!} (\lambda - \lambda^0)^n = f(x^{\lambda_0}, \beta^*) + \phi x^{\lambda_0} \ln(x) \quad (3.115)$$

where ϕ^0 is the estimated coefficient of the regression for $x^{\lambda_0} \ln(x)$. Thus, we want to solve for λ^1 given that:

$$\phi^0 = (\lambda - \lambda^0) \beta^0 \quad (3.116)$$

Rearranging, our next estimate for λ (λ^1) is:

$$\lambda^1 = \frac{\phi^0}{\beta^0} + \lambda^0 \quad (3.117)$$

Each time we iterate this process the difference $\lambda - \lambda^i$ decreases, eventually giving us our estimate for λ .

3.8.6 Semi/Nonparametric Models

This material is covered in [Wood \(2006\)](#) chapter 3 in greater depth. What if we were to fit this model using a more flexible form where all we assumed about the relationship between y and x was that it included some smooth function $f(x)$:

$$y_i = f(x_i) + e_i \quad (3.118)$$

The question is then how do we choose that smooth function without assuming it is a power function, polynomial, or trying to eyeball it, all of which become more difficult with more regressors. One way to do this is to choose a space of functions B that $f(x)$ is some subspace of, meaning $f(x)$ is some combination of functions in our basis B . Suppose $b_i(x)$ is the i th basis function, then:

$$f(x) = \sum_{i=1}^q b_i(x)\beta_i \quad (3.119)$$

where the parameters β_i are unknown. For example, a fourth order polynomial basis would be:

$$f(x) = \beta_1 + \beta_2x + \beta_3x^2 + \beta_4x^3 + \beta_5x^4 \quad (3.120)$$

As can be seen, $b_1 = 1$, $b_2 = x$, $b_3 = x^2$, $b_4 = x^3$ and $b_5 = x^4$. Obviously this basis would easily fit the example population regression function. A more general basis function is called the *cubic spline*, which is just a bunch of cubic polynomials “spliced” together. Where the functions are spliced together are called the *knots* of the spline, and the spline must be smooth around the knots (with continuous first and second derivatives). Knots are usually evenly spaced across the regressor values.

The function $R(x, z)$ used for the cubic spline is sort of nasty. It transforms each input variable x according to its proximity to each knot z . Thus, a single regressor taking 1 column will be expanded to have one column for each z . The first two basis functions are $b_1(x) = 1$ and $b_2(x) = x$. The rest are ([Gu \(2013\)](#) pg. 37):

$$R(x, z) = \frac{[(z - 1/2)^2 - 1/12][(x - 1/2)^2 - 1/12]}{4} - \frac{[(|x - z| - 1/2)^4 - 1/2(|x - z| - 1/2)^2 + 7/240]}{24} \quad (3.121)$$

Here's an example implementing the cubic spline functions in R:

```
# R(x,z) for cubic spline on [0,1]
rk <- function(x, z){
  ((z - 0.5)^2 - 1/12)*((x - 0.5)^2 - 1/12)/4
  - ((abs(x - z) - 0.5)^4 - 0.5*(abs(x - z) - 0.5)^2 + 7/240)/24
}
```

Wood (2006) provided some example data on scaled engine capacity versus engine wear for 19 Volvo car engines:

```
# Engine size
size <- c(1.42,1.58,1.78,1.99,1.99,1.99,2.13,2.13,2.13,
2.32,2.32,2.32,2.32,2.32,2.43,2.43,2.78,2.98,2.98)

# Engine wear index
wear <- c(4.0,4.2,2.5,2.6,2.8,2.4,3.2,2.4,2.6,4.8,2.9,
3.8,3.0,2.7,3.1,3.3,3.0,2.8,1.7)

# Scale size to the unit interval [0, 1]
x <- size - min(size)
x <- x / max(x)
```

We now need to create a function that creates a model matrix for the regression spline. This just takes our 1 regressor and creates additional columns corresponding to the spline function $R(x, z)$, just like we would do if we were to create additional columns corresponding to x^2 , x^3 etc.:

```
# set up model matrix for cubic penalized regression spline
# x will be the regressor
# z will be the knot locations
spl.X <- function(x, z)
{
  # number of columns
  q <- length(z) + 2
  # number of data
  n <- length(x)

  # matrix of 1s
  X <- matrix(1, nrow = n, ncol = q)
  # set second column to x
  X[ , 2] <- x

  # set remaining columns to R(x, z)
```

```

# the outer product here makes the first row R(x1, z)
# the second row R(x2, z), etc.
# here z are the knot locations
X[, 3:q] <- outer(x, z, FUN = rk)
X
}

```

The function $spl.X(x, z)$ takes a regressor x_{i1} and knot sequence z and outputs a model matrix that looks like this:

$$X = \begin{pmatrix} 1 & x_{11} & R(x_{11}, z_1) & R(x_{11}, z_2) & R(x_{11}, z_3) & R(x_{11}, z_4) \\ 1 & x_{21} & R(x_{21}, z_1) & R(x_{21}, z_2) & R(x_{21}, z_3) & R(x_{21}, z_4) \\ 1 & x_{31} & R(x_{31}, z_1) & R(x_{31}, z_2) & R(x_{31}, z_3) & R(x_{31}, z_4) \\ 1 & x_{41} & R(x_{41}, z_1) & R(x_{41}, z_2) & R(x_{41}, z_3) & R(x_{41}, z_4) \\ 1 & x_{51} & R(x_{51}, z_1) & R(x_{51}, z_2) & R(x_{51}, z_3) & R(x_{51}, z_4) \\ 1 & x_{61} & R(x_{61}, z_1) & R(x_{61}, z_2) & R(x_{61}, z_3) & R(x_{61}, z_4) \\ 1 & x_{71} & R(x_{71}, z_1) & R(x_{71}, z_2) & R(x_{71}, z_3) & R(x_{71}, z_4) \\ 1 & x_{81} & R(x_{81}, z_1) & R(x_{81}, z_2) & R(x_{81}, z_3) & R(x_{81}, z_4) \end{pmatrix}$$

In this example there are four knot locations (z_1, z_2, z_3, z_4) and eight observations of x_{i1} . The dimension (rank) of the model matrix X is the number of knots plus 2, where the first two dimensions are the intercept and x_1 . Thus, the model matrix contains what we use for ordinary least squares as a special case in the first two columns.

Finally, we choose knots, generate the model matrix, fit the linear model, generate model predictions, and plot the model against the data:

```

# choose some knots
z <- 1:4/5

# generate model matrix
X <- spl.X(x, z)

# fit model
mod.1<-lm(wear ~ X - 1)

# x values for prediction
xp <- 0:100/100

Xp <- spl.X(xp, z) # prediction matrix

png(filename = "splines1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)

```

```

plot(x, wear, xlab = "Scaled engine size",
      ylab = "Wear index")
lines(xp, Xp %*% coef(mod.1)) # plot fitted spline
dev.off()

```

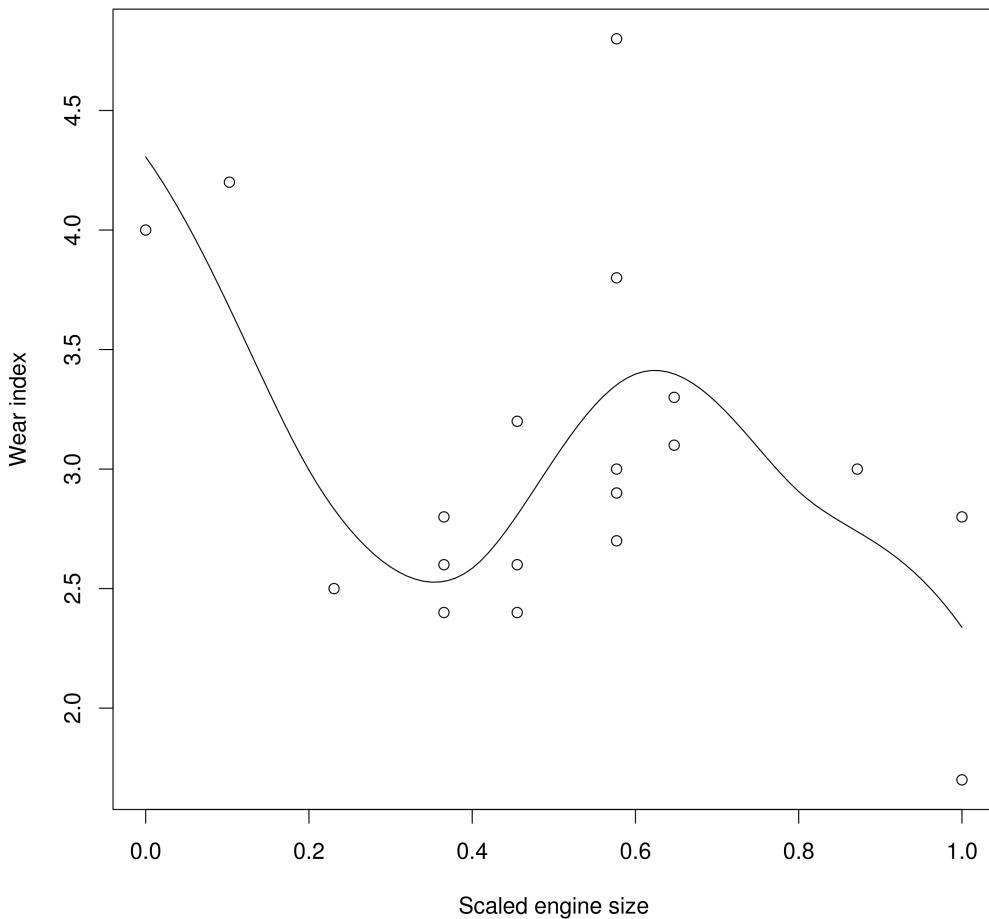


Figure 3.42: Cubic regression spline fit to engine size and wear data with 4 knots.

Unfortunately, the number of knots was chosen arbitrarily, and the performance of the regression spline will depend strongly on the knot locations. We would like to have a more robust approach that doesn't involve such arbitrary decisions. To do this, consider the following penalized regression:

$$\operatorname{argmin}_{\beta} [(y - X\beta)^2 + \lambda \int_x [f''(x)]^2 dx] \quad (3.122)$$

Here $f''(x)$ defines how “wiggly” the model’s smoothing functions are, where the wigglier the functions, the more penalty in the minimization. For example, if $f(x) = x$, then there is no penalty. The degree of penalization is determined by λ , called the *smoothing parameter*. As you would expect, large λ means a very high penalty, favoring linearity, while small $\lambda = 0$ allows for a model with arbitrary flexibility. We can rewrite the objective function as:

$$\operatorname{argmin}_{\beta} [(y - X\beta)^2 + \lambda \beta' S \beta] \quad (3.123)$$

The estimator of $\hat{\beta}$ is then:

$$\hat{\beta} = (X' X + \lambda S)^{-1} X' y \quad (3.124)$$

We can create S in the following way ([Wood \(2006\)](#) page 127):

```
spl.S <- function(xk)
# set up the penalized regression spline penalty matrix,
# given knot sequence xk
{ q <- length(xk) + 2;
  S <- matrix(0, q, q) # initialize matrix to 0
  S[3:q, 3:q] <- outer(xk, xk, FUN = rk) # fill in non-zero part
  S
}

S <- spl.S(z)
```

The matrix S has the first two rows and columns of zeros and the subsequent elements are $R(z_i, z_j)$ where z is the knot location:

$$S = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & R(z_1, z_1) & R(z_1, z_2) & R(z_1, z_3) & R(z_1, z_4) \\ 0 & 0 & R(z_2, z_1) & R(z_2, z_2) & R(z_2, z_3) & R(z_2, z_4) \\ 0 & 0 & R(z_3, z_1) & R(z_3, z_2) & R(z_3, z_3) & R(z_3, z_4) \\ 0 & 0 & R(z_4, z_1) & R(z_4, z_2) & R(z_4, z_3) & R(z_4, z_4) \end{pmatrix}$$

```
# Engine size
size <- c(1.42, 1.58, 1.78, 1.99, 1.99, 1.99, 2.13, 2.13, 2.13,
       2.32, 2.32, 2.32, 2.32, 2.43, 2.43, 2.78, 2.98, 2.98)

# Engine wear index
wear <- c(4.0, 4.2, 2.5, 2.6, 2.8, 2.4, 3.2, 2.4, 2.6, 4.8, 2.9,
        3.8, 3.0, 2.7, 3.1, 3.3, 3.0, 2.8, 1.7)

# Scale size to the unit interval [0, 1]
```

```

x <- size - min(size)
x <- x / max(x)

# choose some knots
z <- 1:4/5

prs.fit <- function(y, x, knots, lambda){
  x <- spl.X(x, knots)
  s <- spl.S(knots)
  beta.hat <- solve(t(x) %*% x + lambda * s) %*% t(x) %*% y
  beta.hat
}

prs.fit(wear, x, z, 1)
plot(x, wear)

```

It turns out that we can select λ using cross-validation by leaving a single observation out, fitting the model given a specific λ , then calculating the mean squared forecasting error over all observations, then repeating for different values of λ until the MSE is minimized.

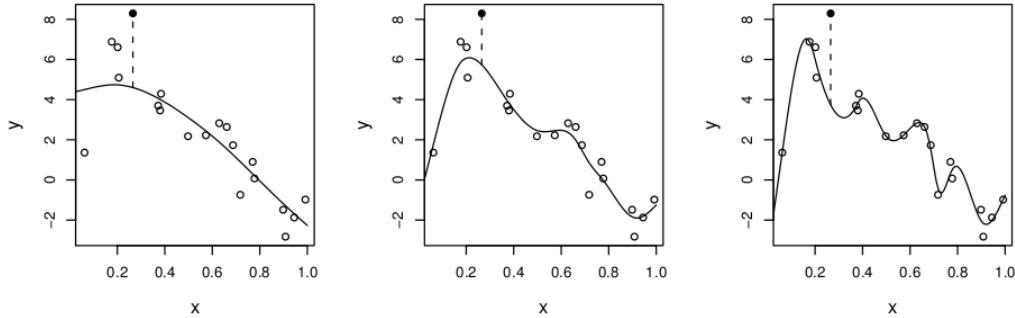


Figure 3.43: Cross-validation to select λ for cubic regression spline. As can be seen λ is too large on the far left plot (too much penalization for wigginess) and too small on the far right plot (not enough penalty). The plot in the middle shows the optimal value of λ selected by cross-validation. Image taken from Wood, S. (2006). Generalized additive models: An introduction with R. CRC press.

3.8.7 Box-Cox Transformation

This explanation follows Cohen et al. (2002) Box 6.4.1. Suppose we want to find a transformation $g(y) = \frac{y^\lambda - 1}{\lambda}$ such that the value λ produces residuals

that are normal and homoskedastic. We cannot compare models with different λ because they are not on the same scale, so we must normalize them in the following way:

$$Z^{(\lambda)} = \frac{Y^\lambda - 1}{\lambda(Y_G)^{\lambda-1}} \quad (3.125)$$

for $\lambda \neq 0$ and

$$Z^{(\lambda)} = Y_G \ln(Y) \quad (3.126)$$

for $\lambda = 0$. In each expression, Y_G is the geometric mean of the Y scores on the untransformed scale:

$$Y_G = (Y_1 \times Y_2 \times \cdots \times Y_n)^{1/n} \quad (3.127)$$

The reason behind this normalization is as follows. Let's call the size of the dependent variable it's volume, measured as the product of all the y observations:

$$\text{Volume}(y) = \prod_{i=1}^n y_i \quad (3.128)$$

Now, when we transform y to y^λ , the volume changes to:

$$\text{Volume}(y^\lambda) = \prod_{i=1}^n y_i^\lambda = \left(\prod_{i=1}^n y_i \right)^\lambda \quad (3.129)$$

We can see that changing λ has increased (or decreased) the size or volume of our dependent variable. This is going to cause a problem if we want to compare the quality of different regression models (in terms of MSE for example) across different models with different volumes. Those with larger volumes will have larger MSE, and those with smaller volumes will have smaller MSE. For example, consider the variance of y as we change λ :

```
# Generate y uniform from 1 to 2
y <- runif(100, 1, 2)

# Square y
y.2 <- (y^2-1)/2

# Calculate the geometric mean and divid y.2 by it
gm <- exp(mean(log(y)))
y.2.norm <- y.2/gm

# Calculate the variance of each dependent variable
var(y)
var(y.2)
var(y.2.norm)
```

As can be seen, the variance of the squared dependent variable is about twice as large as the untransformed dependent variable, but that difference is mostly eliminated by dividing by the geometric mean. In sum, to compensate for the changes in volume, we can calculate the relative volume:

$$\frac{\text{Volume}(y^\lambda)}{\text{Volume}(y)} = \frac{(\prod_{i=1}^n y_i)^\lambda}{\prod_{i=1}^n y_i} = (\prod_{i=1}^n y_i)^{\lambda-1} \quad (3.130)$$

Thus, if we divide y^λ by $(\prod_{i=1}^n y_i)^{\lambda-1}$ we can keep all of our dependent variables at constant volume. If we take the n th root of the untransformed dependent variable (the geometric mean), then this keeps the average volume constant across all observations (rather than having the total volume be constant).

With the normalized transformed dependent variable $Z^{(\lambda)}$ we can compare the residuals from the regression of $Z^{(\lambda)}$ on the independent variables X against each other. From this point it is simple to find the optimal value of λ by trying values of λ in some range, calculating the rMSE from each regression of $Z^{(\lambda)}$ on X , and choosing the value of λ that minimizes the rMSE ([Draper et al. \(1966\)](#) page 280). Usually λ values from -2 to 2 are tried. That demonstrates the general idea, but it is also possible to estimate λ using maximum likelihood estimation.

3.9 Homework 3

As always, we will have a replication plus extension format for the homework. We begin by replicating the results of [Anglin and Gencay \(1996\)](#). Read sections 1, 3 and 4.1 (Section 2 is optional). Write a short report that replicates Tables II and III (pg. 638-640). The data can be obtained from the package AER:

```
library(AER)
data(HousePrices)
```

In the replication report, I expect you to:

1. Conduct the ordinary least squares regressions and compare your results to those reported in the paper. (1 point)
2. Correctly interpret the coefficients for the DRV and LOT variables for both regressions. Would you recommend standardizing or mean-centering any variables? (1 point)

After attempting to reproduce these tables, write a short report telling the five stories of these data. Use the benchmark model in Table III as a starting point. Try to improve our understanding of the sample and population conditional distribution of house prices both qualitatively and quantitatively beyond this benchmark model. Here's what I expect to be included in the extension part of the report:

1. Create histograms of the house prices, log house prices, lot size, number of bedrooms, number of full bathrooms, and number of stories, with a summary of what you're seeing. (1 point)
2. Create tables (using the table() function) of the driveway, recreational room, full and finished basement, gas for hot water heating, central air conditioning, and preferred neighborhood dummy variables, with a summary of what you're seeing. (1 point)
3. Comment on whether you think the Gauss-Markov assumptions are met for the benchmark model in Table III. Which assumptions are likely to be met or not met and why? (1 point)
4. Plot the jackknife residuals versus fitted values for the benchmark regression using house prices rather than log house prices. What do you see? Using the Box-Cox scheme, what value of λ would you suggest for a transformation of the house prices? Is the log transform appropriate for the dependent variable? Why or why not? (1 point)

5. Create a component plus residual plot for the untransformed lot size variable in the benchmark regression model, using log transformed house prices. What transformation does the component plus residual plot suggest? Compare this to the value of λ that the Box-Tidwell function suggests. What transformation of lot size should we use (if any)? (1 point)
6. Create a semi-parametric Generalized Additive Model (GAM) as an extension of the benchmark model with log transformed house prices. Use smoothing only for the lot size variable. Look at the partial residual plot of the lot size variable from the GAM. What transformation to the lot size variable looks appropriate? How does this compare to your conclusion from the component plus residual plot and Box-Tidwell transformation? (1 point)
7. For the forecasting story, compare the semi-parametric GAM used in the previous section against the benchmark model with whatever you think is the most appropriate transformation for the lot size variable (based on what you've learned from the component plus residual plot, Box-Tidwell transform, and partial residual plot from the GAM). Does your model or the semi-parametric model perform better in terms of cross-validation? Summarize what you think this means. (1 point)
8. Compare heteroskedasticity robust standard error estimates to “classical” standard error estimates for the benchmark model in Table III with log transformed house prices. Is there any difference in the standard errors? Comment on what this says about potential model misspecification. (1 point)
9. Briefly comment on the statistical inference and causality stories. (1 point)

Here's a list of functions you might need:

- `hist()`
- `density()`
- `table()`
- `boxCox()` (car package)
- `boxTidwell()` (car package)
- `crPlots()` (car package)

- `gam()` (`mgcv` package)
- `lmtest()` (`lmtest` package)
- `vcovHC()` (`sandwich` package)

Make sure your report includes reproducible R code, either as an appendix, as a footnote through Harvard's Dataverse (or some other site of your choice), or through CMU's dropbox.

Chapter 4

Discrete Outcomes and The Generalized Linear Model

Discrete outcomes take on a finite number of values, usually two or three, such as 0 and 1, true or false, or cereal A, B, or C. When we try to understand discrete outcomes we do not expect to be able to add, subtract, or multiply the dependent variable and have a meaningful result. For example, in a clinical trial that looks at survival of patients given a treatment or placebo, there are two outcomes, alive and dead. We don't expect that alive +2 is meaningful, or that dead² is a potentially interesting transformation. Instead, we are interested in the *relative frequency* or probability of observations falling in one of the outcome categories as a function of a set of independent variables.

This chapter covers an extension of the linear least squares regression we are now so familiar with to situations where our outcome variable is discrete. This generalization encompasses linear least squares, and is thus appropriately called the *Generalized Linear Model*. We'll focus on binary outcome variables, although the generalized linear model applies to any variable that can be modeled with an exponential family distribution (e.g., count data with the Poisson distribution).

4.1 The Data Summary Story

Summarizing discrete outcomes involves looking at the relative frequency or probability of an outcome occurring. Let's take a look at some data made available by the vcd package:

```
install.packages("vcd", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(vcd)
data(Arthritis)
head(Arthritis)
```

These data include whether someone was treated for arthritis, their study ID#, their sex, age, and whether their arthritis status improved during the trial. We start by obtaining a simple tabulation of the frequency of improvements in arthritis symptoms among all people in the study, the dependent variable:

```
# Simple table of improvement status  
structable(~ Improved, Arthritis)
```

We can see that, overall, half of the participants (42 or 50%) had no status improvement, a few had some improvement (14 or 17%), and one-third had a marked status improvement (28 or 33%). We can also take a look at the frequency of improved status by treatment condition, and graph it using a spineplot:

```
# Table that shows improvement status by treatment condition  
structable(Treatment ~ Improved, Arthritis)  
png(filename = "spineplot1.png", width = 3000, height = 3000, res = 300)  
par(cex = 1.3)  
# Generate spineplot  
spineplot(Improved ~ Treatment, data = Arthritis)  
dev.off()
```

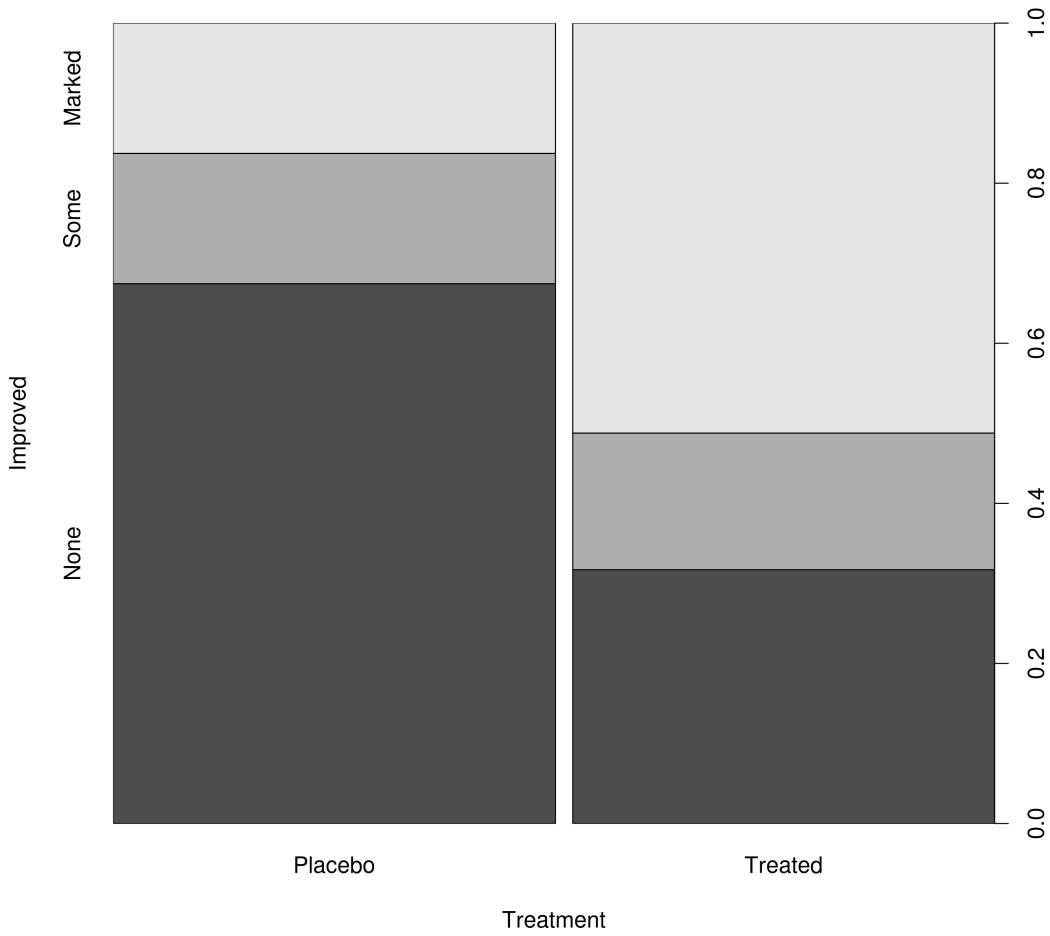


Figure 4.1: Spineplot showing the proportion of patients with no (dark grey), some (middle grey), or marked (light grey) improvement by treatment condition.

About half (43) of the participants in the study received the placebo, while the other half (41) received the treatment.

Improvement	Placebo	Treated
None	29	13
Some	7	7
Marked	7	21
Total	43	41

As the participants are about equally distributed between the two treatment groups, we should expect the frequency of improved status to be

roughly the same across the two groups if the arthritis treatment had no effect. In fact, the frequency of improved status seems to be very different between the two treatment groups. About one-third (13/41) of those who received the treatment showed no improvement in their condition, whereas about two-thirds (29/43) of those who received the placebo showed no improvement. The relative frequency of some improvement is about the same between the two groups. However, the relative frequency of marked improvement is much higher for the treated group (21/41 or 51%) than the placebo group (7/43 or 16%).

4.1.1 Absolute Risk, Relative Risk, and The Odds Ratio

There's three important ways to think about the difference in relative frequencies between the groups. The first is the *absolute risk difference*, which is just the difference in relative frequency of improvement. In our example, 7/43 showed marked improvement in the placebo group (16%), while 21/41 (51%) showed marked improvement in the treatment group. Thus, the absolute risk improvement is 0.35 ($= .51 - .16$) or 35% ($= 51\% - 16\%$).

We can also look at the *relative risk reduction*, which is like the absolute risk reduction, but scaled to the event rate in the control group. In our example, the relative risk reduction would be 2.19 ($= .35/.16$) or 219% ($= 51\% - 16\%)/(16\%$). This is closely related to the *relative risk* or *risk ratio*, which is just the rate of marked improvement in the treatment group divided by that in the control group 3.19 ($= .51/.16$) or 319% ($= 51\%/16\%$).

Lastly, and probably most importantly, is the *odds ratio*. The odds of marked improvement in the placebo group would be the number improved divided by the number who did not show marked improvement, or 0.19 ($= 7/36$). Likewise, the odds of marked improvement in the treated group would be 1.05 ($= 21/20$). Thus, the odds ratio is the ratio of these odds, which is 5.53 ($= 1.05/0.19$).

4.1.2 The Odds Ratio

Absolute risk improvement and relative risk make some sense intuitively. Absolute risk improvement is the easiest to understand: We go from some probability of the event occurring (either good or bad) to some new probability, and calculate the difference between the two.

The odds ratio is the most important, as we will see shortly. Unfortunately, the odds ratio has a long history of misinterpretation (Bland and Altman, 2000; Prasad et al., 2008; Sackett et al., 1996). It's so important (like the log transform), and so easy to misinterpret, that we should spend

some time on it. The critical question is: What does an odds ratio of say, 4, mean? If we are comparing two groups, then the odds of an event in one group is 4 times larger than the odds of the event in the other group. Let's say the probability of contracting a disease in a placebo group is .8, so the odds is 4 ($= .8/.2$). In the treatment group, the probability is .5, so the odds is 1 ($= .5/.5$). Thus, the odds ratio is 4 ($= 4/1$).

Fortunately or unfortunately, this does not tell us anything about the probabilities in each group. For example, the probability of contracting the disease in the placebo group could be .5 (odds = 1 = $.5/.5$), with a probability of contracting the disease in the treatment group of .25 (odds = $.25/.25$ = $.25/.8$), yielding an equivalent odds ratio of 4 ($= 1/.25$). Thus, *the odds ratio says nothing unique about probabilities*. For emphasis:

The odds ratio says nothing unique about probabilities.

The odds ratio says nothing about how much more likely something is.

The odds ratio says nothing about how much more probable something is.

The odds ratio only tells us the relative odds.

Because the odds ratio isn't telling us anything unique about probabilities, why not use something more intuitive, like relative risk? For example, a probability of contracting the disease of .8 in the placebo group with .5 in the treatment group gives us a relative risk of 1.6. We can interpret this as the probability of disease contraction is 1.6 times greater in the placebo group than treatment group. We're back to probabilities, which humans have reasonable intuitions about. The odds ratio is, again, 4.

4.1.3 The Odds Ratio

For a number of quantitative reasons, using the odds ratio is preferable to other risk measures. One reason we use the odds ratio is that it is sensitive even when probabilities are very high or very low, whereas the risk ratio is not. For example, suppose $p_1 = 0.99$ and $p_2 = 0.999$. The risk ratio is $\frac{p_2}{p_1} \approx 1$ whereas the odds ratio is $\frac{.999/.001}{.99/.01} = \frac{999}{99} = 10$. Thus, when probabilities are very high or very low, the risk ratio is insensitive to changes in probability, whereas the odds ratio is sensitive.

This is bad, because the odds ratio is not intuitive. Let's continue this line of trying to understand the odds ratio by looking at how it differs from the other risk measures. First let's consider how the odds ratio differs from the risk difference or $p_1 - p_2$:

```
set.seed(25)
```

```

# Generate probabilities uniform from 0 to 1
p1 <- runif(1000, 0, 1)
# Calculate the odds of p1
odds1 <- p1 / (1 - p1)
# Do the same for p2
p2 <- runif(1000, 0, 1)
odds2 <- p2 / (1 - p2)

png(filename = "oddsratios1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(2, 2), mar = c(5, 4, 1, 1))
plot(p1 - p2, odds1/odds2,
      ylim = c(0,1),
      pch = 19,
      col = rgb(0, 0, 0, .1))
plot(p1-p2, odds1/odds2,
      ylim = c(0, 10),
      pch = 19,
      col = rgb(0, 0, 0, .1))
plot(p1-p2, odds1/odds2,
      ylim = c(0, 20),
      pch = 19,
      col = rgb(0, 0, 0, .1))
plot(p1-p2, odds1/odds2,
      pch = 19,
      col = rgb(0, 0, 0, .1))
dev.off()

```

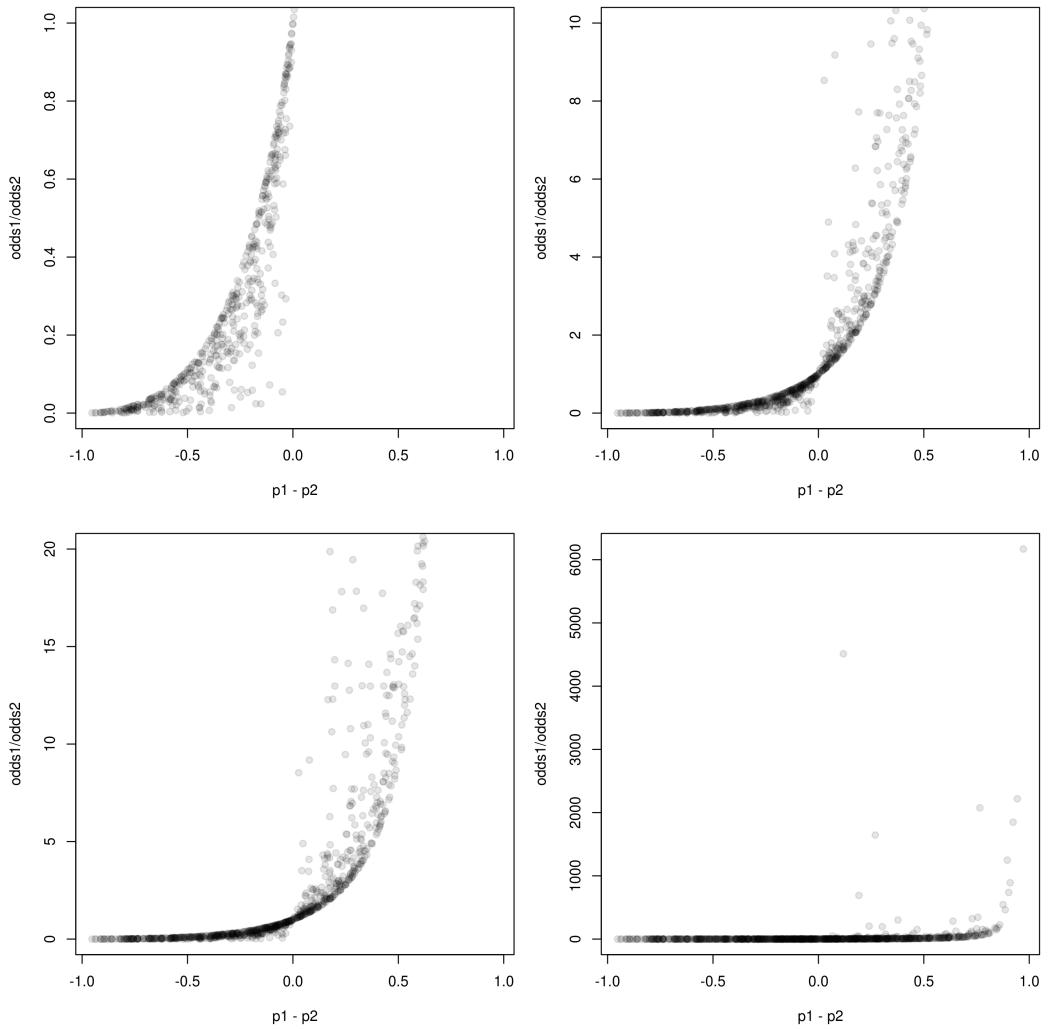


Figure 4.2: Plot of odds ratios by probability difference. The same data are used in each plot, with the y axis increasing from 1 to unconstrained, going from the top left to bottom right.

The relationship between the risk difference and odds ratio is non-linear, possibly linear after log transforming the odds ratio. Also notice that for each risk difference we can get just about any odds ratio, although there is a high density of odds ratios along the prevailing curve. Because the graph looks linear in the log odds ratio, we might as well take a look:

```
set.seed(26)
p1 <- runif(1000, 0, 1)
odds1 <- p1 / (1 - p1)
p2 <- runif(1000, 0, 1)
odds2 <- p2 / (1 - p2)
```

```
png(filename = "logoddsratios.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(1, 2))
plot(p1 - p2, log(odds1/odds2),
  pch = 19,
  col = rgb(0, 0, 0, .1))
plot(p1 - p2, log(odds1/odds2),
  pch = 19,
  col = rgb(0, 0, 0, .1))
diff <- p1 - p2
abline(lm(log(odds1/odds2) ~ diff), col = "red", lwd = 4, lty = 2)
# Fit GAM to prevailing curve
lines(sort(diff),
  predict(gam(log(odds1/odds2) ~ s(diff)),
  newdata = data.frame(diff = sort(diff))),
  col = "green",
  lwd = 4)
dev.off()
```

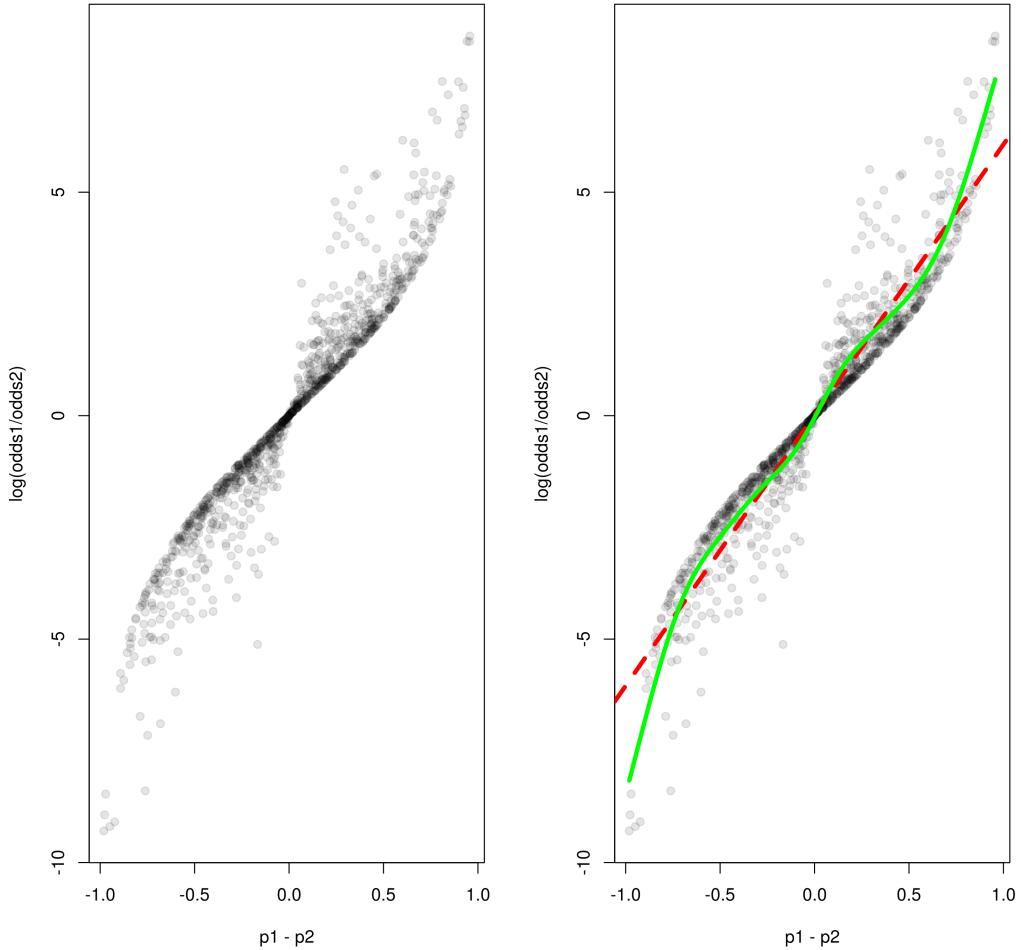


Figure 4.3: Plot of log odds ratios by probability difference (left), with linear and GAM fit (right).

The log odds ratio is roughly linearly proportional to the probability difference for probability differences ranging between 0 and 0.75. For large differences in probability, the linear approximation fails.

Also notice that sometimes the relationship is off the line, even when the probability difference is relatively small (i.e., less than 0.75). When does this occur? Let's look at a few risk differences, to see when the log odds ratio is off the line. First let's remind ourselves of the linear fit relating the log of the odds ratio to the difference in probabilities:

```
set.seed(27)
# Generate any probability and odds ratios uniformly
p1 <- runif(1000, 0, 1)
```

```

odds1 <- p1 / (1 - p1)
p2 <- runif(1000, 0, 1)
odds2 <- p2 / (1 - p2)
# Calculate the vector of probability differences
diff <- p1 - p2
# Regress the log odds ratio on the probability differences
logodds.diff <- lm(log(odds1/odds2) ~ diff)

```

We now have in the logodds.diff object a linear regression of the log odds on the probability differences. We now want to see how deviations from linearity relate to probabilities and probability differences. Let's start with a probability that varies almost the entire range, and a second probability that has a small difference of 0.1 from that initial probability:

```

# Allow p1.1 to vary between 0 and 0.899
p1.1 <- runif(1000, 0, 0.899)
# p2.1 has a small difference of 0.1 from p1.1
p2.1 <- p1.1 + 0.1
odds1.1 <- p1.1 / (1 - p1.1)
odds2.1 <- p2.1 / (1 - p2.1)
log.odds.1 <- log(odds1.1 / odds2.1)
# Make predictions for the log odds based on a risk difference of 0.1
# This prediction will be constant
pred.1 <- predict(logodds.diff, newdata = data.frame(diff = - 0.1))

```

We now have a range of probabilities associated with a probability difference of 0.1, and a point prediction ($\text{pred.1} = -0.59$) from a model that assumes the log odds is linear in the risk difference. Let's repeat this for three more risk differences (0.2, 0.3, and 0.4):

```

p1.2 <- runif(1000, 0, 0.799)
p2.2 <- p1.2 + 0.2
odds1.2 <- p1.2 / (1 - p1.2)
odds2.2 <- p2.2 / (1 - p2.2)
log.odds.2 <- log(odds1.2 / odds2.2)
pred.2 <- predict(logodds.diff, newdata = data.frame(diff = - 0.2))

p1.3 <- runif(1000, 0, 0.699)
p2.3 <- p1.3 + 0.3
odds1.3 <- p1.3 / (1 - p1.3)
odds2.3 <- p2.3 / (1 - p2.3)
log.odds.3 <- log(odds1.3 / odds2.3)
pred.3 <- predict(logodds.diff, newdata = data.frame(diff = - 0.3))

```

```

p1.4 <- runif(1000, 0, 0.599)
p2.4 <- p1.4 + 0.4
odds1.4 <- p1.4 / (1 - p1.4)
odds2.4 <- p2.4 / (1 - p2.4)
log.odds.4 <- log(odds1.4 / odds2.4)
pred.4 <- predict(logodds.diff, newdata = data.frame(diff = - 0.4))

```

Next, let's plot the actual log odds as it differs from the prediction of the model that assumes the log odds is linear in the risk differences:

```

png(filename = "logoddsdiffs.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(2, 2), mar = c(5, 4, 2, 1))
# Plot the difference from linearity in the log odds by probability difference
plot(p1.1, log.odds.1 - pred.1,
      pch = 19,
      col = rgb(0, 0, 0, .1),
      xlim = c(0, 1),
      xlab = "p1 Proportion",
      main = "p2 - p1 = 0.1",
      ylab = "Deviation from Linearity")
plot(p1.2, log.odds.2 - pred.2,
      pch = 19,
      col = rgb(0, 0, 0, .1),
      xlim = c(0, 1),
      xlab = "p1 Proportion",
      main = "p2 - p1 = 0.2",
      ylab = "Deviation from Linearity")
plot(p1.3, log.odds.3 - pred.3,
      pch = 19,
      col = rgb(0, 0, 0, .1),
      xlim = c(0, 1),
      xlab = "p1 Proportion",
      main = "p2 - p1 = 0.3",
      ylab = "Deviation from Linearity")
plot(p1.4, log.odds.4 - pred.4,
      pch = 19,
      col = rgb(0, 0, 0, .1),
      xlim = c(0, 1),
      xlab = "p1 Proportion",
      main = "p2 - p1 = 0.4",
      ylab = "Deviation from Linearity")
dev.off()

```

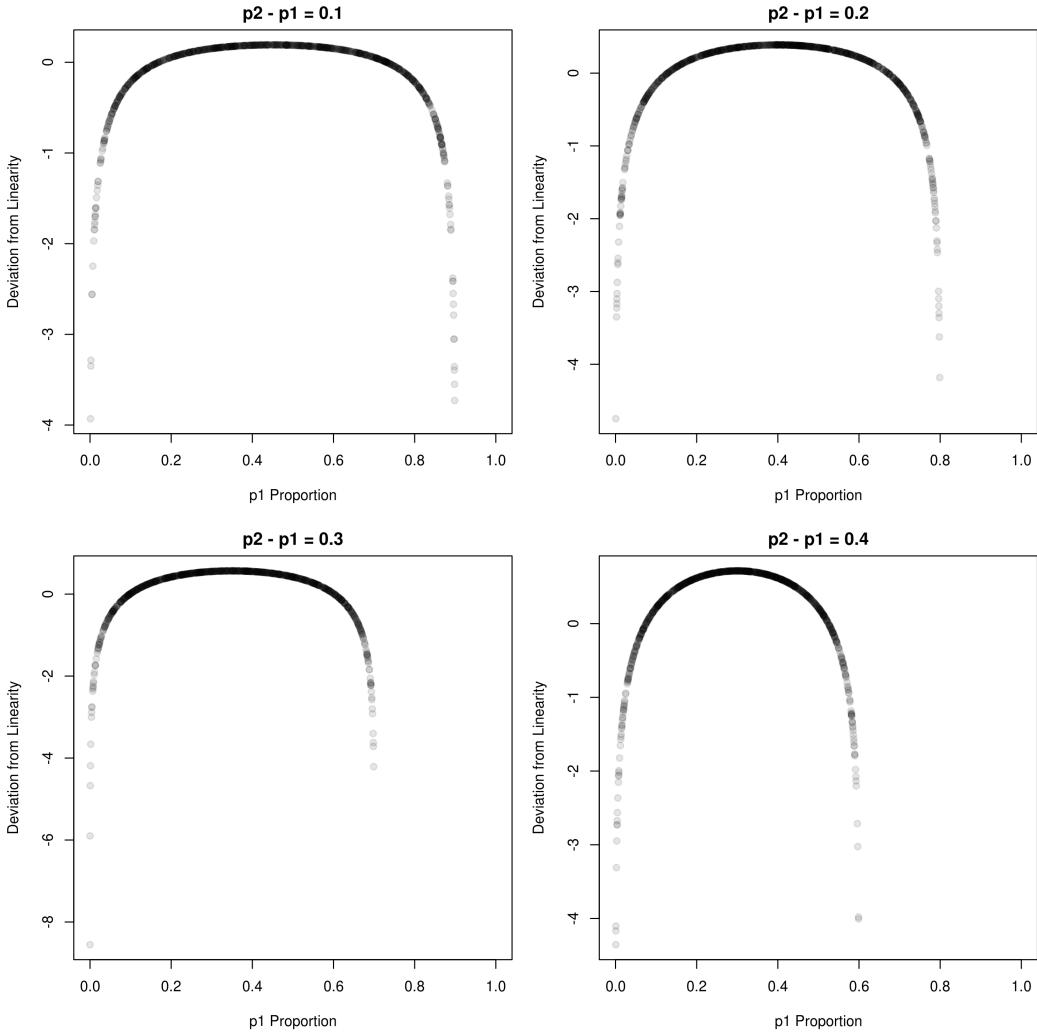


Figure 4.4: Plot of deviation of the log odds ratio from linearity depending on the proportion p_1 and the difference in proportions $p_2 - p_1$ by size of the difference (0.1, 0.2, 0.3, 0.4), from top left to bottom right respectively.

We can see that the log-odds ratio is highly non-linear in the risk difference near the boundaries, where $p_1 = 0$ or $p_1 + p_2 = 1$. This deviation from linearity is relatively insensitive to the magnitude of the difference between p_1 and p_2 . So, the % change in the odds ratio is roughly linear in probability difference, with the exceptions mentioned.

This doesn't really help intuitions that much, because knowing the log odds ratio is roughly linear except for large probability differences and except for very low probabilities or $p_1 + p_2 = 1$ is hardly intuitive. What about the relationship between the odds ratio and the risk ratio? The risk ratio has some intuitive meaning: If the risk ratio is 2, then the probability is twice as large:

```
set.seed(29)
p1 <- runif(1000, 0, 1)
odds1 <- p1 / (1 - p1)
p2 <- runif(1000, 0, 1)
odds2 <- p2 / (1 - p2)

png(filename = "rrr1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(p1/p2, odds1/odds2,
      pch = 19,
      xlim = c(0, 2),
      ylim = c(0, 2),
      col = rgb(0, 0, 0, .2),
      ylab = "Odds Ratio (odds1/odds2)",
      xlab = "Risk Ratio (p1/p2)")
abline(0, 1)
dev.off()
```

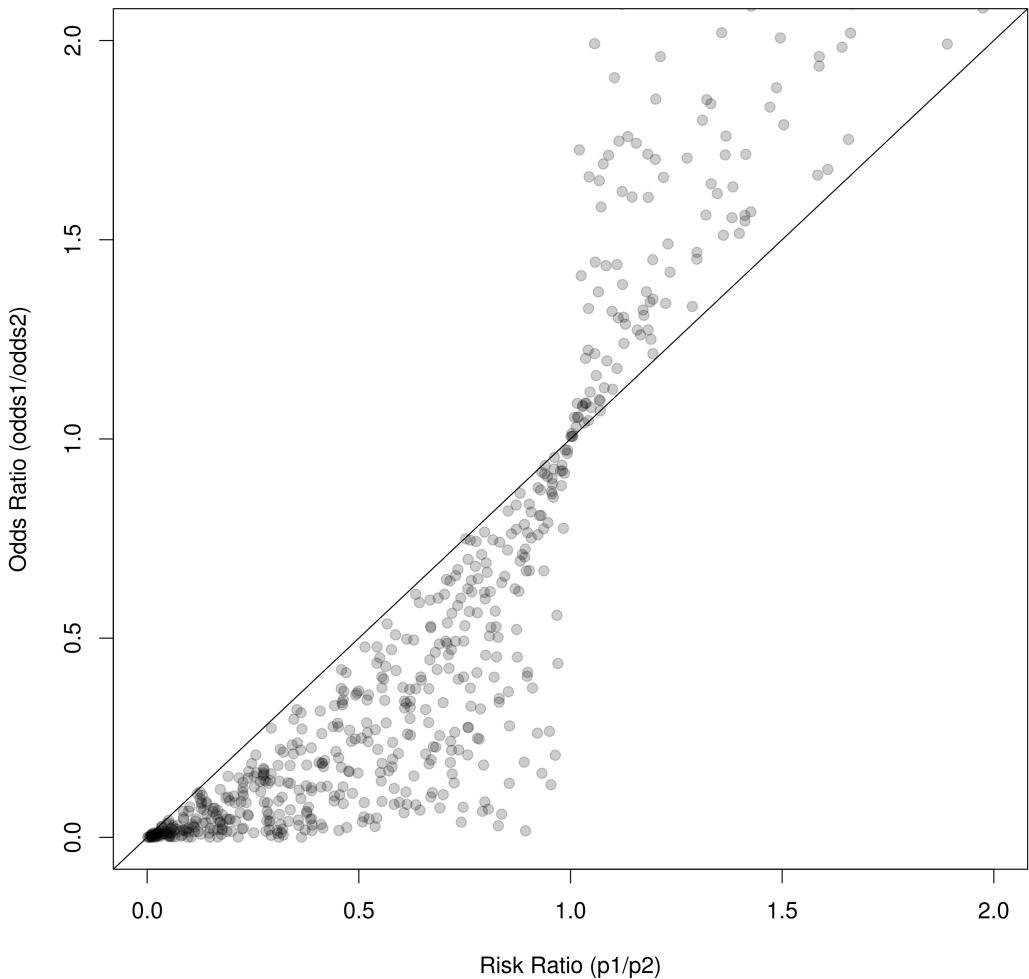


Figure 4.5: Plot of odds ratios by risk ratio.

We can learn a little bit from this plot: The odds ratio is always smaller than the risk ratio when less than one, and greater than the risk ratio when greater than one. So we can kind of bound our intuitions about relative probabilities using the odds ratio. Still, not that informative. When will the odds ratio be very different from the risk ratio? Consider the odds ratio:

$$OR = \frac{\frac{p_1}{1-p_1}}{\frac{p_2}{1-p_2}} = \frac{p_1(1-p_2)}{p_2(1-p_1)} = RR \times \frac{1-p_2}{1-p_1} \quad (4.1)$$

Where RR is the risk ratio or relative risk (p_1/p_2). So, the closer both p_1 and p_2 are to zero (and each other), the closer the odds ratio is to the risk ratio. This is called the “rare disease assumption” sometimes used in medical research. We can also see that the odds ratio and risk ratio can differ either because p_1 is large or p_2 is large (Grimes and Schulz, 2008).

So, under the rare disease assumption, the risk ratio is close to the odds ratio, so we can use that to help our intuitions. We shouldn't compare the odds ratio to the risk difference in this case, because when probabilities are very low or high, % change in odds ratio is non-linear in risk difference. Conversely, if we want to inform our intuitions when the probabilities are not near 0 or 1 (i.e., the rare disease assumption is false), then we'd do better thinking about the fact that the % change in the odds ratio is roughly proportional to the risk difference. We don't want to use the risk ratio as a comparator in this case, because the odds ratio and risk ratio can differ wildly when the rare disease assumption is violated. I suggest spending some time thinking about odds ratios and playing with numbers, using simulations or math by hand.

4.2 The Conditional Distribution Story

Modeling a conditional distribution of a discrete outcome variable is very much like modeling for a continuous one. In both cases we are interested in how some feature of the conditional distribution, such as the conditional mean, varies as a function of input variables. For discrete outcomes, the conditional mean is a conditional proportion of cases. For example, the conditional mean snow days in Pittsburgh on Tuesdays is the number of snow days (coded "1") divided by the total number of Tuesdays, which is just the proportion of snow days relative to Tuesdays. Thus:

$$E[Y|X = x] = P(Y = 1|X = x) \quad (4.2)$$

In words, the conditional mean of Y given some value of x is just the proportion of $Y = 1$ at that value of x .

4.2.1 Linear Probability Model

We could estimate this conditional mean using the usual least squares approach. Here's some data on a person's vote for a republican or democratic governor in relationship to this person's self-reported scale of conservatism (1 = very liberal, 5 = very conservative). The outcome is 1 if the person voted republican, and 0 if the person voted democrat:

```
votes <- read.table("VotingData.txt", col.names = c("conserv", "voterep"))
```

Let's start with a simple linear regression:

```
consreg <- lm(conserv ~ voterep, data = votes)
```

Those who voted for the republican were more conservative than those who voted for the democrat. The average conservatism among those who voted for a republican was 3.68, compared to 2.35 for the democrat. Plotting the data and regression line:

```
png(filename = "conservativevote.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(votes$voterep, amount = 0.025), jitter(votes$conserv),
     xlab = "Vote for Governor",
     ylab = "Conservativism",
     xaxt = "n",
     pch = 19,
     col = rgb(0, 0, 0, .5))
axis(side = 1, at = c(0, 1), labels = c("Democrat", "Republican"))
abline(consreg)
dev.off()
```

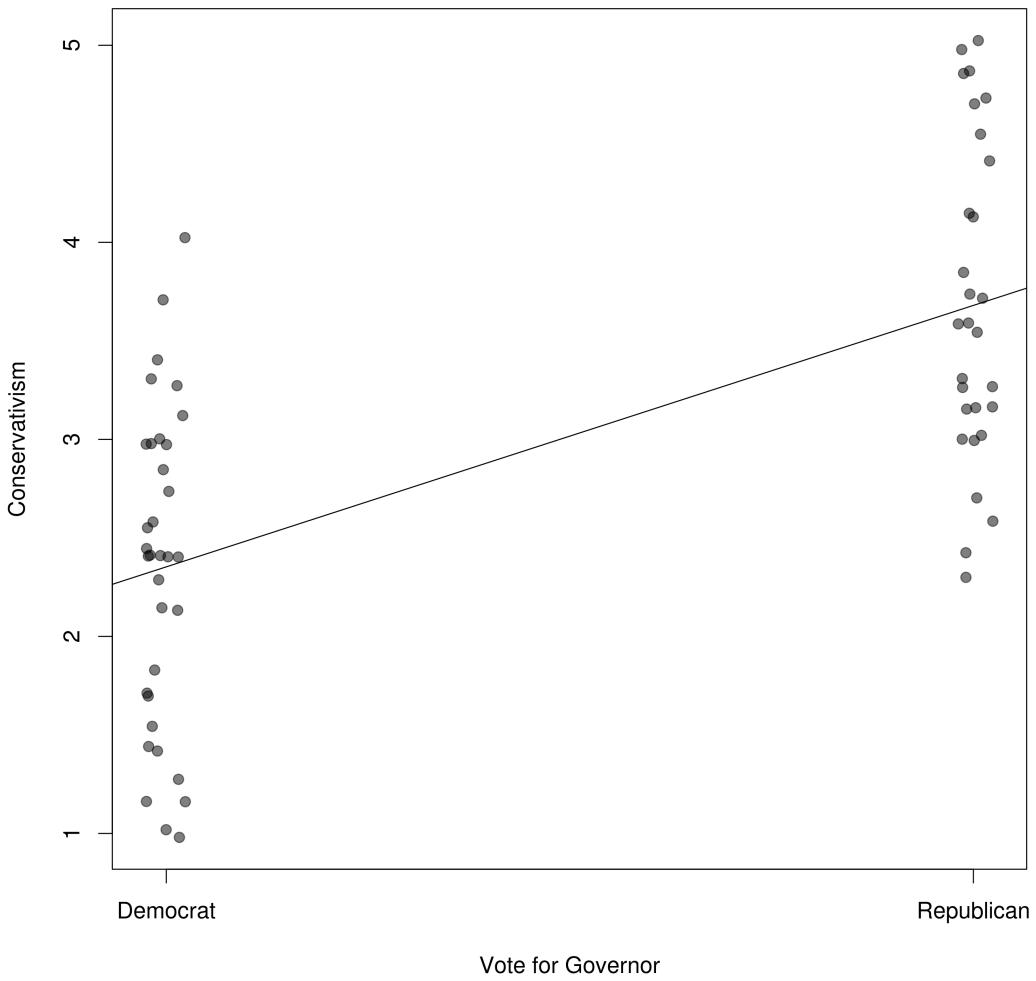


Figure 4.6: Conservativism as a function of the person's vote for governor.

With only a single dichotomous independent variable we are able to perfectly predict the means of the dependent variable. If we want to model the conditional probability of voting for a republican governor, we can just apply linear regression to data that are discrete binary outcomes (i.e., 0 or 1). This is called the *Linear Probability Model*. To do this let's turn the regression around. Because the dependent variable is dichotomous, our regression is predicting the probability of voting republican at each level of conservativism:

```
votereg <- lm(voterrep ~ conserv, data = votes)
```

Let's look at the predicted probability of voting republican given different levels of conservativism:

```
png(filename = "lpmvotes.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(votes$conserv), jitter(votes$votereg, amount = .025),
     xlab = "Conservativism",
     ylab = "Predicted Probability of Voting for Republican",
     ylim = c(-.2, 1.2),
     pch = 19,
     col = rgb(0, 0, 0, .5))
abline(votereg)
dev.off()
```

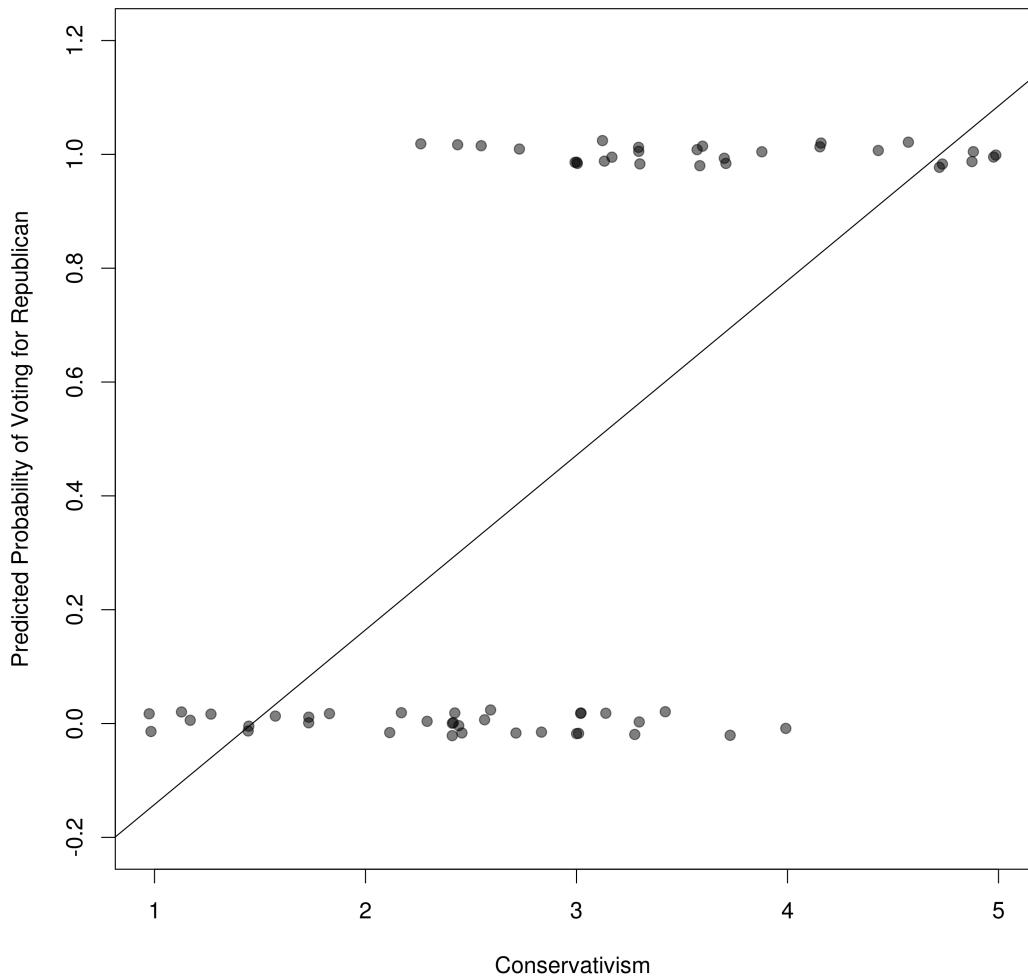


Figure 4.7: Probability of voting republican for different levels of reported conservativism using a linear probability model.

One clear problem from this graph is that for extreme values of conservativism, nonsense probabilities are predicted. At low values there is a negative predicted probability of voting republican. At high values, there is a predicted probability of voting republican greater than 1.

A second issue is heteroskedasticity. The variance of errors will be greater near the middle probabilities than those at the ends. For example, at the point where we predict a .50 probability of voting republican (with a conservativism score of about 3), half of those will vote republican and half won't. The error variability is at a maximum. However, when we predict a probability of 0 or 1, there is no variance, as all observations either voted republican or did not.

4.2.2 Non-Parametric Comparison

For qualitative comparison, we can fit a non-parametric curve to the graph, that doesn't assume a linear form, and uses cross-validation for bandwidth selection. We won't go into kernel regression methods in the course, but they can be a useful alternative to conventional parametric methods (e.g., OLS, Logistic Regression) or semi-parametric methods (e.g., GAM):

```
install.packages("cubature", repos = "http://lib.stat.cmu.edu/R/CRAN/")
install.packages("np", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(np)
# Fit nonparametric regression
np1 <- npreg(voterep ~ conserv, data = votes)
png(filename = "npvotes.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(votes$conserv), jitter(votes$voterep, amount = .025),
     xlab = "Conservativism",
     ylab = "Predicted Probability of Voting for Republican",
     ylim = c(-.2, 1.2),
     pch = 19,
     col = rgb(0, 0, 0, .5))
# Plot predictions of non-parametric regression
lines(votes$conserv, predict(np1), col = "red")
abline(votereg, lty = 2)
dev.off()
```

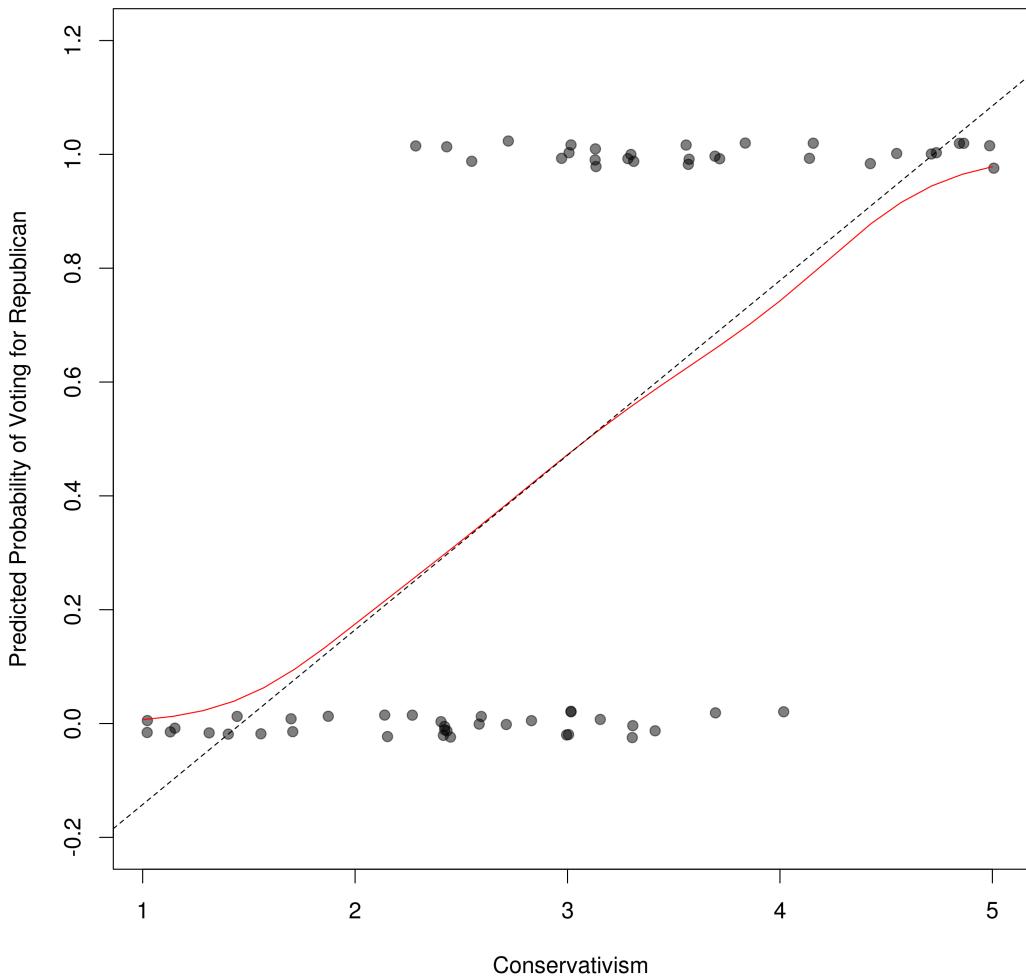


Figure 4.8: Probability of voting republican for different levels of reported conservativism. The black dashed line shows the linear fit, whereas the red line shows the non-parametric kernel regression.

We can see that there are very small differences between the non-parametric curve and the linear. For the most part, the relationship looks linear in the middle. Where the linear fit is really failing is near the extremes, where probabilities are close to 1 and 0. Here, the non-parametric curve correctly tapers off to probabilities of zero or one, whereas the linear fit keeps going, eventually predicting impossible probabilities.

4.2.3 Incorrectly Using the Logit Transform

The problem is that probabilities are bounded at either end, but straight lines are not. To take advantage of all we know from linear regression, that is,

allowing our estimate of the conditional mean to be a linear function of parameters (with any non-linear transformations allowed to the inputs themselves), we need to find a function that maps zeros and ones onto the real line. One approach is to use the odds:

$$\text{Odds} = \frac{p}{1-p} \quad (4.3)$$

Where p is $P(Y = 1|X = x)$. However, odds are still bounded at zero (i.e., $\frac{0}{1-0} = 0$). To solve the problem of the bounded range on the lower end, we can take the log of the odds, also called the *Logit transformation*:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) \quad (4.4)$$

Let's see how the logit (log-odds) function looks:

```
p <- seq(0, 1, .01)
logit.p <- log(p / (1 - p))
png(filename = "plogitp.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(p, logit.p,
      xlab = "p",
      ylab = "Logit(p)",
      pch = 19,
      col = rgb(0, 0, 0, .5))
lines(p, logit.p)
dev.off()
```

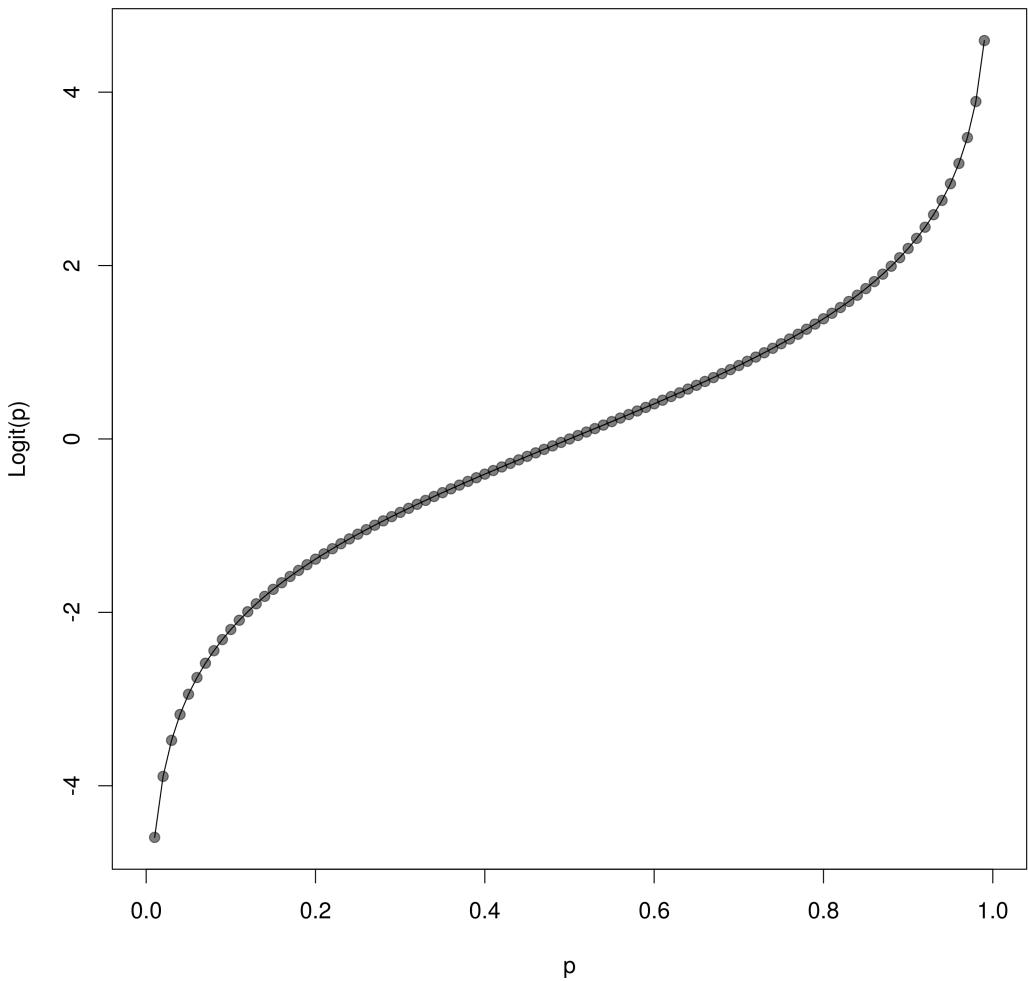


Figure 4.9: Plot of a proportion (p) and the logit transformation of the proportion ($\text{Logit}(p)$).

This looks very much like our plot of the log odds ratio versus risk difference earlier, but without the large deviations. Notice that the logit transformation approaches negative infinity as p goes to zero, and infinity as p goes to 1, making it unbounded, just like we wanted. Also notice that the logit transformation is approximately linear from about $p = 0.2$ to $p = 0.8$. If we want to get our probability back from the logit function, we invert it with the *logistic function*:

$$p = \frac{e^{\text{logit}(p)}}{1 + e^{\text{logit}(p)}} = \frac{1}{1 + e^{-\text{logit}(p)}} \quad (4.5)$$

The logistic function is the familiar s-shaped curve people talk about when they discuss logistic regression:

```

p <- seq(0, 1, .005)
logit.p <- log(p / (1 - p))
logistic.p <- 1/(1 + exp(-logit.p))
png(filename = "logitlogistic.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(logit.p, logistic.p,
      xlab = "Logit(p)",
      ylab = "p = Logistic(Logit(p))",
      col = rgb(0, 0, 0, .5),
      pch = 19)
dev.off()

```

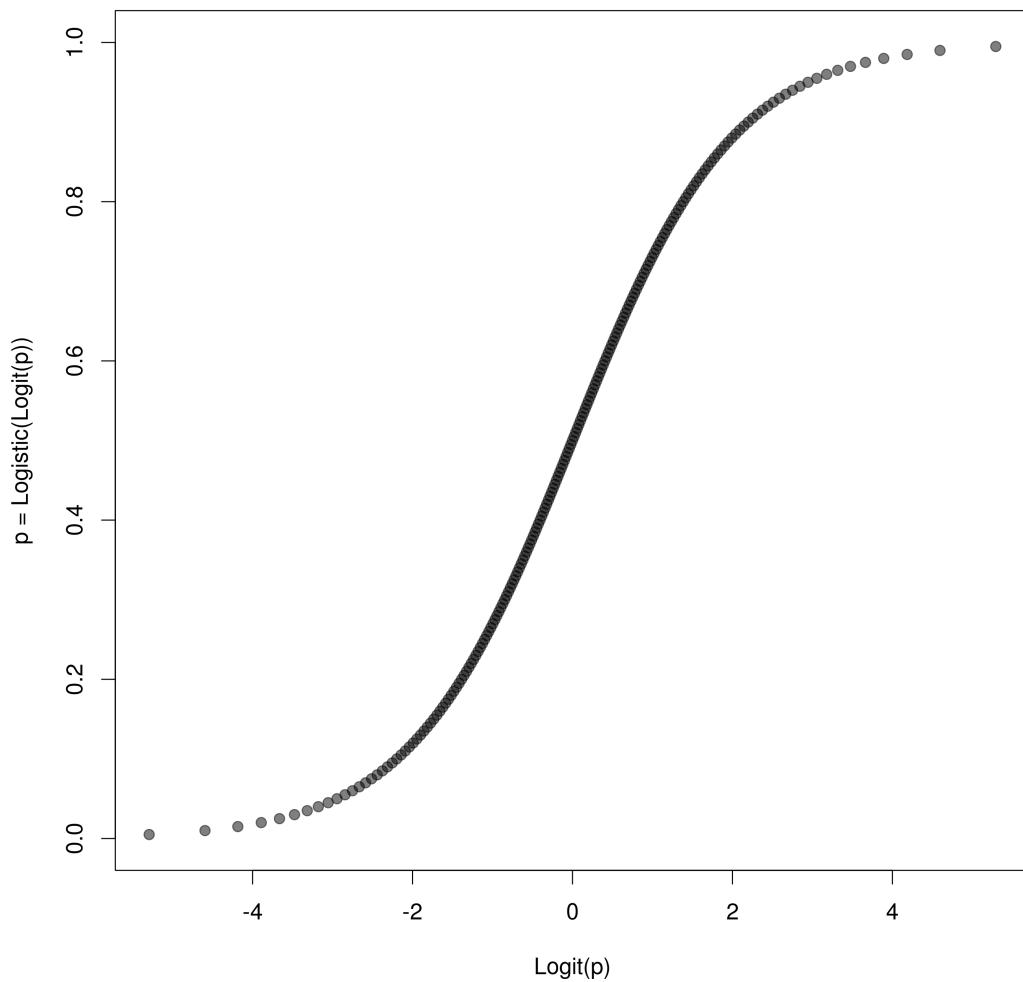


Figure 4.10: Plot of $\text{Logit}(p)$ and $p = \text{Logistic}(\text{Logit}(p))$.

Now that we've found a nice transformation to make to our dependent

variable, let's go ahead and transform then run linear regression, just like we did with the log transform. Unfortunately this won't work, because discrete outcomes are all 0 or 1, not a probability. At 0 and 1 the logit transform is $-\infty$ and ∞ , respectively. Not very useful.

4.3 Logistic Regression

Logistic regression solves these problems. Logistic regression does *not* use the logistic transformation on the dependent variable. Logistic regression also does *not* use the logit transform on the odds ratio. As we've noted above, both of these transformations just give us impossible values of $-\infty$ and ∞ if we input the dependent variable. Logistic regression has three components:

1. A *stochastic component* that models the data generating process.
2. A *deterministic linear predictor* $\eta(x) = x\beta$.
3. A *link function* that relates the linear predictors $x\beta$ to the conditional mean $p(x\beta)$.

Logistic regression is part of a broader family of models that all have these three components and are subsumed under the generalized linear model. They differ on their stochastic component (e.g., Poisson instead of Bernoulli) and their link functions.

4.3.1 Bernoulli Stochastic Component

The variability that enters into the logistic regression comes from its *stochastic* or *random* component. In the case of binary data, the random component that generates the outcome Y is a *Bernoulli* random variable:

$$Y \sim \text{Bernoulli}(p) \quad (4.6)$$

That is, our response variable Y takes on values 0 or 1, and can be modeled as a coin flip with $P(Y = 1) = p$ (that is the probability that we get a "heads" ($Y = 1$) is p). All of the random variation we observe in the data is accounted for by this mechanism. We can generate a Bernoulli random variable with different values of p and see what the data look like:

```
set.seed(40)
# Look at probabilities from 0.1 to 0.9
p <- seq(.1, .9, by = 0.1)
png(filename = "bernoulliplots.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(3, 3), mar = c(5, 4, 2, 1))
```

```

# Generate bar plots
for(i in 1:9){
  # Generate 100 random observations from a bernoulli random variable
  # The mean of the bernoulli random variable is p = p[i]
  y <- rbinom(100, 1, p[i])
  plot(factor(y), ylim = c(0, 100), main = paste("p = ", p[i]))
}
dev.off()

```

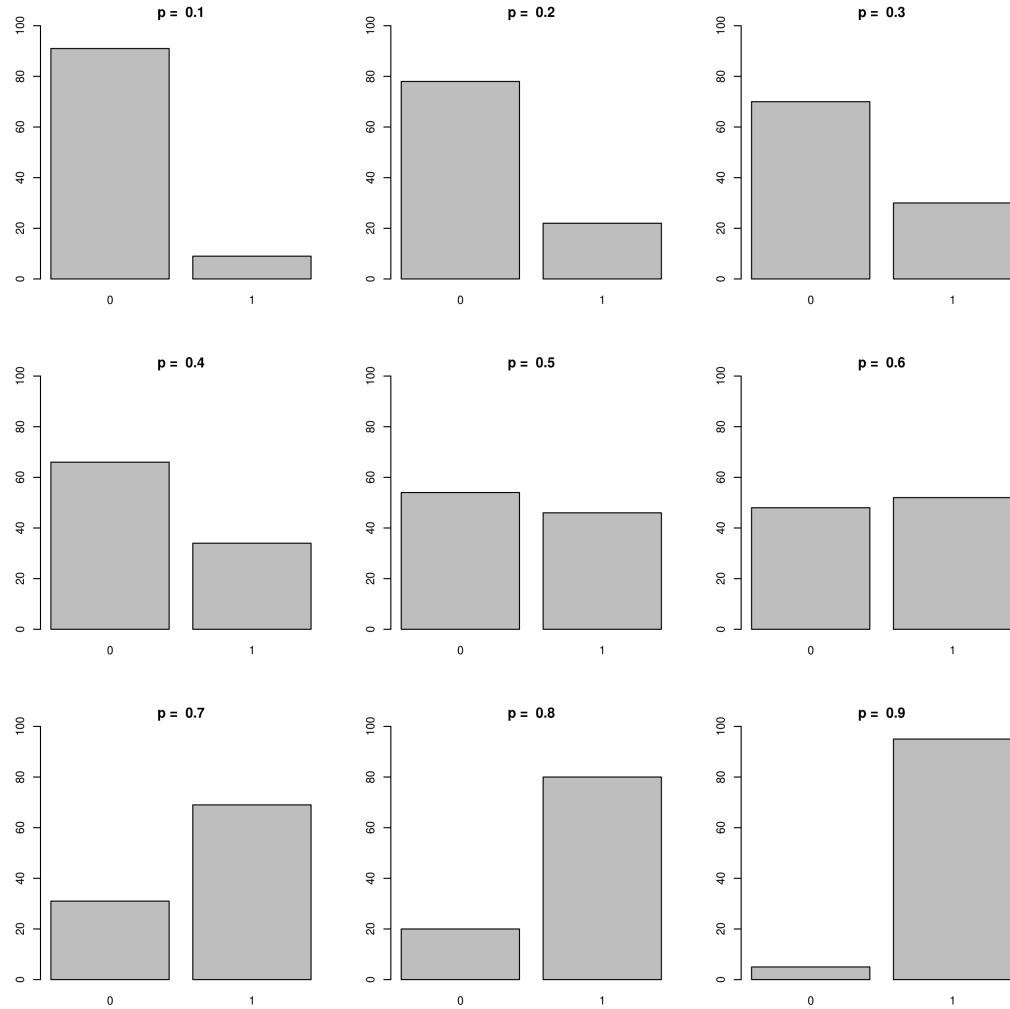


Figure 4.11: Plots of 100 draws from a Bernoulli random variable with $P(Y = 1) = p$ varying from 0.1 (top left), to 0.9 (bottom right).

The plots should confirm our intuition that as p increases, the proportion of outcomes where $Y = 1$ increases. For example, for $p = 0.1$, about 10% of

observations have $Y = 1$. There are serious implications to using this model of our outcome Y . Most importantly, the variance of a Bernoulli random variable is a function of the mean, and vice versa:

$$\text{Var}(Y) = p(1 - p) \quad (4.7)$$

So, for $Y \sim \text{Bernoulli}(0.1)$, the variance is 0.09 ($= 0.1 \times 0.9$). We can see this in the top left graph, as there is not much variability in the values; most are $Y = 0$. As previously mentioned, the variance of the Bernoulli random variable is at a maximum when $p = 1 - p = 0.5$. Because of this relationship between the mean (p) and the variance ($p(1 - p)$), we cannot separate them, and thus everything that messes up our estimation of the conditional mean also messes up our estimation of the conditional variance, and vice versa.

4.3.2 Bernoulli Likelihood

The *likelihood function* is like the stochastic component, but tells us something about how probable a *parameter* in our model is given data, rather than how probable data are given a parameter (i.e., how to generate data). We can use the likelihood function to tell how well a hypothesized parameter fits the data. For example, we can think of binary discrete outcomes as being generated by a sequence of independent coin flips characterized by the Bernoulli likelihood. With one coin flip, the likelihood function $L(p|y_1)$ is:

$$L(p|y_1) = p^{y_1} (1 - p)^{1-y_1} \quad (4.8)$$

So, suppose we observe $y_1 = 1$. With the likelihood function, we can ask how likely it is that this observation was generated by a Bernoulli random variable with parameter p . For example, suppose we think $p = 0.1$ is the most likely value for p . Then the likelihood is $L(p = 0.1|y_1 = 1)$:

$$L(p = 0.1|y_1 = 1) = 0.1^1 (1 - 0.1)^{1-1} = 0.1 \quad (4.9)$$

So, the likelihood that $p = 0.1$ is 0.1. What if we thought $p = 0.2$:

$$L(p = 0.2|y_1 = 1) = 0.2^1 (1 - 0.2)^{1-1} = 0.2 \quad (4.10)$$

We can plot the likelihood function for different values of p to find the value of p that makes the observed data $y_1 = 1$ most likely:

```
# Look at possible values of p
p <- seq(0, 1, by = .05)
likelihood <- function(p, y) {
  # Generate the likelihood function for one binary outcome
  # Args:
```

```

#   p is a vector of hypothesized parameters p
#   y is a single observed binary outcome (0 or 1)
# Returns:
#   The likelihood of y given different values of p
#    $p^y * (1 - p)^{1 - y}$ 
}
png(filename = "likelihood1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(p, likelihood(p, 1),
      xlab = "p",
      ylab = "Likelihood:  $L(p|y = 1)$ ",
      pch = 19)
lines(p, likelihood(p, 1))
dev.off()

```

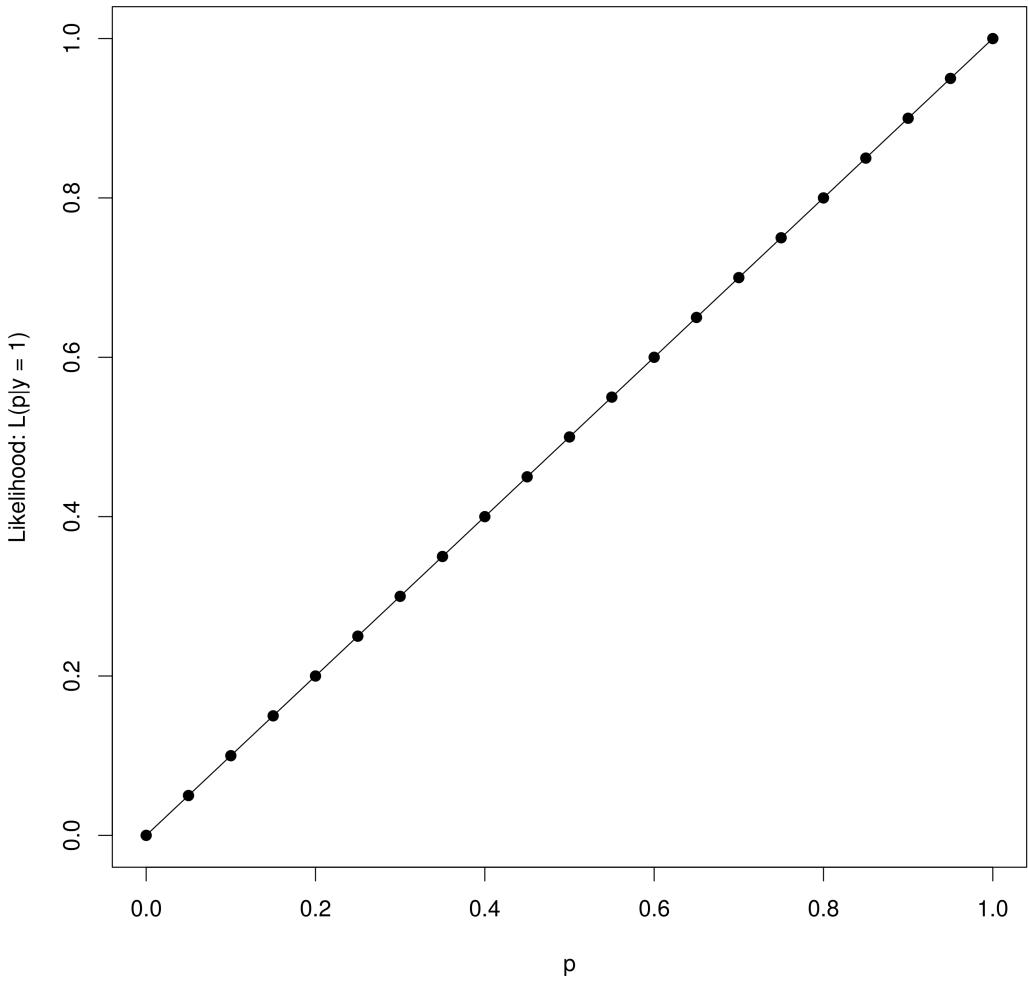


Figure 4.12: Plot of the Bernoulli likelihood function $L(p|y_1) = p^{y_1}(1-p)^{1-y_1}$ for $y_1 = 1$ versus the hypothesized parameters p .

The result is hopefully intuitive: With one observation where $y = 1$, it's most likely that $p = 1$, slightly less likely that $p = 0.9$, and much less likely that $p = 0.1$. Actually, we've already ruled out $p = 0$, because we can never see $y = 1$ if $p = 0$. What if we observe two outcomes, $y_1 = 1$ and $y_2 = 0$? The likelihood function is very similar, as it is just the product of the two likelihoods, *if the two observations are independent*:

$$L(p|y_1, y_2) = L(p|y_1) \times L(p|y_2) \quad (4.11)$$

We can expand this:

$$L(p|y_1) \times L(p|y_2) = p^{y_1}(1-p)^{1-y_1} \times p^{y_2}(1-p)^{1-y_2} \quad (4.12)$$

So, if $y_1 = 1$, and $y_2 = 0$, then the likelihood function is:

$$L(p|y_1 = 1) \times L(p|y_2 = 0) = p^1(1 - p)^{1-1} \times p^0(1 - p)^{1-0}$$
$$L(p|y_1 = 1) \times L(p|y_2 = 0) = p^1 \times (1 - p)^1$$

Now, take a moment to form a guess about the most likely value of p , given these two observations. Let's again plot the likelihood function, now with two observations:

```
# Look at possible values of p
p <- seq(0, 1, by = .05)
# Create vector of two observations y1 = 1 and y2 = 0
y <- c(1, 0)
likelihood <- function(p, y) {
  # Generate the likelihood function for two binary outcomes
  # Args:
  #   p is a vector of hypothesized parameters p
  #   y is a length 2 vector of observed binary outcomes (0 or 1)
  # Returns:
  #   The likelihood of y given different values of p

  p^y[1] * (1 - p)^(1 - y[1]) * p^y[2] * (1 - p)^(1 - y[2])
}

png(filename = "likelihood2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(p, likelihood(p, y),
     xlab = "p",
     ylab = "Likelihood: L(p|y1 = 1, y2 = 0)",
     pch = 19)
lines(p, likelihood(p, y))
dev.off()
```

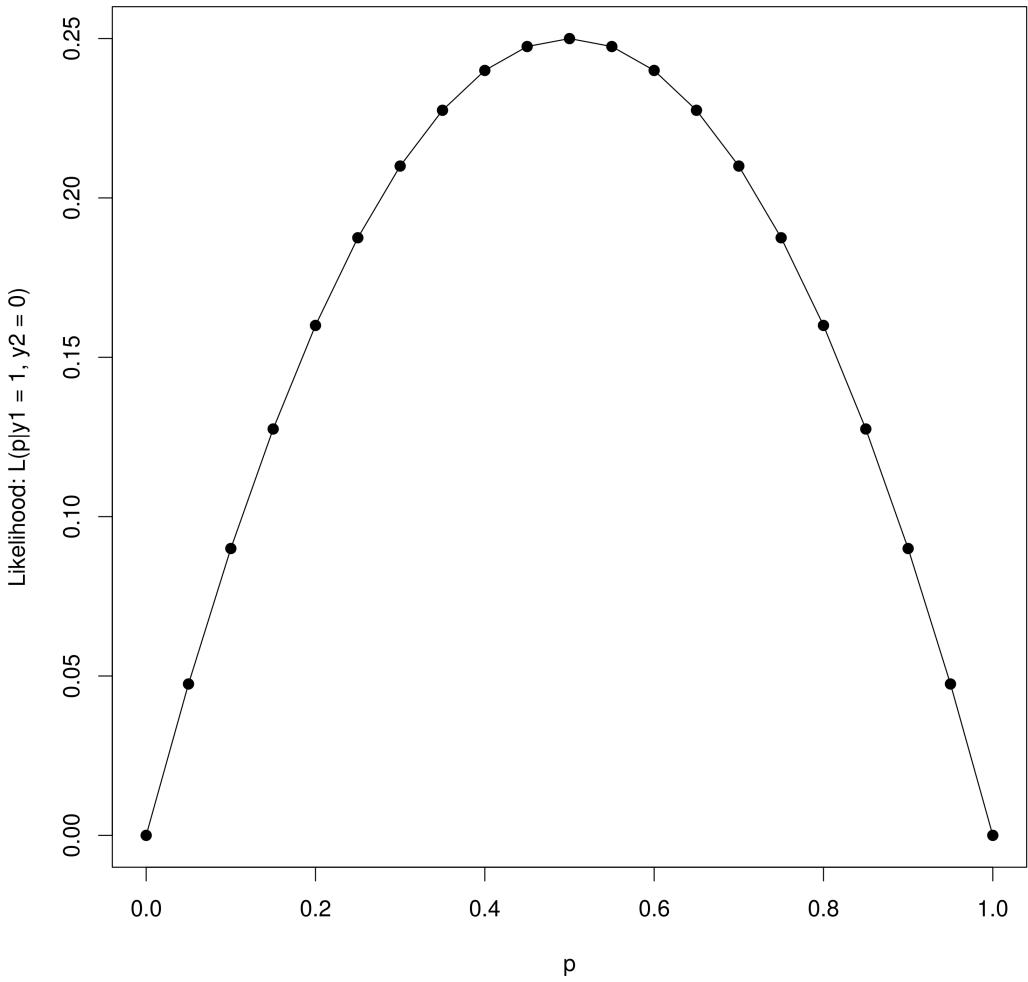


Figure 4.13: Plot of the Bernoulli likelihood function $L(p|y_1) \times L(p|y_2) = p^{y_1}(1-p)^{1-y_1} \times p^{y_2}(1-p)^{1-y_2}$ for $y_1 = 1$ and $y_2 = 0$ versus the hypothesized parameters p .

The plotted result is also hopefully intuitive: With two observations, one of which $y = 1$ and the other where $y = 0$, it's most likely that $p = 0.5$. We've now ruled out both $p = 0$ and $p = 1$, because we can never see $y = 1$ if $p = 0$ or $y = 0$ if $p = 1$. We can generalize this approach. The likelihood function for p given n independent observations is just the product of the likelihoods:

$$L(p|y_i) = \prod_{i=1}^n p^{y_i}(1-p)^{1-y_i} \quad (4.13)$$

With a likelihood function in hand, and a set of parameters that we want to use to find the maximum of the likelihood function, the task is to find the

parameter p that maximizes it. This is called *maximum likelihood estimation*.

Here's a couple more examples. Suppose out of four people who had a conservatism score of 1 ($c = 1$), only one voted republican. Thus, $p(c = 1)$ is the probability of voting republican given a conservatism score of 1. The likelihood is then:

$$L(p|y_{1,2,3,4}) = p(c = 1) \times (1 - p(c = 1)) \times (1 - p(c = 1)) \times (1 - p(c = 1))$$

So, let's find the p that maximizes this. How about $p(c = 1) = 1$. We substitute $p(c = 1) = 1$ into our likelihood and get:

$$p(c = 1) \times (1 - p(c = 1)) \times (1 - p(c = 1)) \times (1 - p(c = 1)) = 1 \times 0 \times 0 \times 0 = 0$$

The likelihood is zero, not a very good guess. So, $p = 1$ is a bad guess for the parameter that maximizes the likelihood function. What about $p = 0.5$?

$$0.5 \times 0.5 \times 0.5 \times 0.5 = 0.0625$$

So, $p = 0.5$ is a better guess than $p = 1$. Let's try one more, $p = 0.25$:

$$0.25 \times 0.75 \times 0.75 \times 0.75 = 0.105$$

Looks like $p = 0.25$ does even better. It turns out that the observed proportion, $1/4$, is the parameter that maximizes the likelihood function. This is intuitive: The most likely probability of voting republican given some level of conservatism is just the observed proportion of people who vote republican at that level of conservatism.

4.3.3 The Logit Link Function

Hopefully we've got some grasp on the mechanism that generates the observed data (the stochastic component) and a procedure for trying to estimate a parameter that is most likely given observed data (the likelihood function and maximum likelihood estimation). The last piece of the puzzle is the *link function*.

The logit link function extends the idea of transforming the data using the logit function, but gets around the $-\infty$ and ∞ problem. Rather than use the logit transform directly on the data, we can instead use the logit transform on the *conditional mean*, or the unobserved probability that $Y = 1$. The link function connects (links) a linear set of predictor variables ($x\beta$) to the conditional mean, using a non-linear function. It *does not* transform the dependent variable directly; instead it specifies the relationship between an unobserved, “latent”, or “hidden”, conditional mean that is not directly observed, and the observed binary outcomes (0 or 1).

In ordinary linear regression, the link function we use is the *identity link function*. That is, the conditional mean of the dependent variable is a linear function of the predictor variables, with no transformation. In this case, the “latent” conditional mean is identical to the observed conditional mean, aside from random error. More precisely, call the link function $g()$, and $E(y|x)$ the mean of y conditional on x , then the identity link function is just:

$$g(E(y|x)) = E(y|x) = \beta_0 + \beta_1 x_1 \quad (4.14)$$

Let’s call the linear predictors $\eta(x) = \beta_0 + \beta_1 x_1$. The identity link function makes no transformation to the conditional mean. Also notice that the link function $g()$ transforms the conditional mean ($E(y|x)$) into $\eta(x)$.

Now, back to the logit link function. Recall that we want to estimate some probability p given a bunch of observations y_i , using the likelihood function:

$$L(p|y_i) = \prod_{i=1}^n p^{y_i} (1-p)^{1-y_i} \quad (4.15)$$

More importantly, we want to extend this so that p is a function of the predictor variables x . The question is, how do we get from this likelihood function to a function that is linear in parameters. That’s exactly what the logit link function does. Consider the logit link function:

$$\text{logit}(p(x)) = \log\left[\frac{p(x)}{1-p(x)}\right] = \beta_0 + \beta_1 x_1 \quad (4.16)$$

In words, $p(x) = E(y|x)$ is the unobserved or latent probability that $y = 1$. We use the logit link function to connect $p(x)$ (the probability) to a set of linear predictors $\eta(x)$. Thus $g(p(x))$ is the logit link function that connects $p(x)$ to the linear predictors $\eta(x) = \beta_0 + \beta_1 x$. How does this relate to the likelihood function? Well, using a little algebra, we can solve the logit link function for $p(x)$:

$$p(x) = \frac{1}{1 + e^{-\eta(x)}} \quad (4.17)$$

Now, we substitute $p(x) = \frac{1}{1+e^{-\eta(x)}}$ into the likelihood function:

$$L(p(x)|y_i) = \prod_{i=1}^n p(x)^{y_i} (1-p(x))^{1-y_i} = \prod_{i=1}^n \left(\frac{1}{1 + e^{-\eta(x)}}\right)^{y_i} \left(1 - \frac{1}{1 + e^{-\eta(x)}}\right)^{1-y_i} \quad (4.18)$$

Now, we have a likelihood function, and we find the values of β_0 and β_1 in $\eta(x) = \beta_0 + \beta_1 x$ that maximize the likelihood, just like we did when we were maximizing the likelihood with respect to p .

Estimation

In general we cannot maximize this analytically, and instead must use a numerical optimization method to solve for $\hat{\beta}$. Here's an example. First let's generate our regressors x_1 and x_2 :

```
x1 <- runif(1000, -2, 2)
x2 <- runif(1000, -3, 3)
x <- cbind(rep(1, length(x1)), x1, x2)
```

Then let's define our linear predictor $\eta(x) = X\beta$:

```
beta0 <- .5
beta1 <- 2
beta2 <- 1
# Create linear predictor xb
xb <- beta0 + beta1*x1 + beta2*x2
```

The key to the simulation is defining y such that the $P(y = 1|x)$ follows the logistic function:

```
# Use logistic function
p <- exp(xb)/(1 + exp(xb))

# Generate coin flips where p(heads) = logistic(xb)
y <- rbinom(1000, 1, p)
```

Now we have generated a series of coin flips where the bias of the coin follows the logistic function of $X\beta$. Next we want to calculate the maximum likelihood estimate $\hat{\beta}$ of β . To do this we can take the log of the likelihood function:

$$\log(L(p(X\beta)|y_i)) = \log\left(\prod_{i=1}^n p(X\beta)^{y_i}(1 - p(X\beta))^{1-y_i}\right) \quad (4.19)$$

Because the log turns products into sums, we have:

$$\log(L(p(X\beta)|y_i)) = \sum_{i=1}^n \log(p(X\beta))^{y_i}(1 - p(X\beta))^{1-y_i} \quad (4.20)$$

This is equal to:

$$\sum_{i=1}^n y_i \log(p(X\beta)) + \sum_{i=1}^n (1 - y_i) \log(1 - p(X\beta)) \quad (4.21)$$

where $p(X\beta) = \frac{1}{1+e^{-X\beta}}$. We can write some code that will create the log likelihood function from our data and a hypothesized set $\hat{\beta} = \theta$:

```

# Based on code by Thomas Debray
# Define the negative log likelihood function
logl <- function(theta, x, y){
  y <- y
  x <- as.matrix(x)
  # theta is a vector of start values for beta
  # that will be updated
  beta <- theta[1:ncol(x)]

  # Define p.xb
  p.xb <- 1/(1 + exp(- x %*% beta))

  # Use the log-likelihood of the Bernouilli distribution
  loglik <- sum(y*log(p.xb)) + sum((1-y)*log(1 - p.xb))

  return(-loglik)
}

```

We can then use an iterative algorithm (Nelder-Mead) to find the set of parameters $\hat{\beta}$ that maximizes the log likelihood (actually we are minimizing the negative log-likelihood):¹

```

# Define initial values for the parameters
# Theta vector starts at zeros equal to the # of columns of x
theta.start <- rep(0, (dim(x)[2]))
names(theta.start) <- colnames(x)

# Calculate the maximum likelihood using optim
mle <- optim(theta.start,
              logl,
              x = x,
              y = y,
              hessian = F)

# Obtain regression coefficients
beta.hat <- mle$par

```

We can compare $\hat{\beta}$ from our code to the `glm` function in R:

```

glm1 <- glm(y ~ x1 + x2, family = binomial(link = "logit"))
summary(glm1)
beta.hat

```

The results should be identical.

¹There's a nice article about function optimization in R here from the creator of `optim` and `optimx`: <http://www.ibm.com/developerworks/library/ba-optimR-john-nash/>

4.3.4 The Logistic (Inverse Logit) Function

Because our unobserved conditional mean is in probabilities that follow the logistic function, we should think about the properties of the logistic function to develop some intuitions about how our conditional probabilities (rather than log odds) are related to the linear predictors. Our predicted probability $\hat{p}(x)$ is obtained by using the logistic function:

$$\hat{p}(x) = \frac{1}{1 + e^{-\hat{\eta}(x)}} \quad (4.22)$$

Recall, that $\hat{\eta}(x)$ is a function that is linear in parameters $\hat{\beta}$ with independent variables x :

$$\log\left(\frac{\hat{p}(x)}{1 - \hat{p}(x)}\right) = \hat{\eta}(x) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_k x_k \quad (4.23)$$

Let's take a look at what the logistic function looks like with respect to $\hat{\eta}(x)$:

```
set.seed(35)
# Draw a bunch of eta values
eta <- runif(200, -10, 10)
# Calculate the mean probability given eta
p <- 1/(1 + exp(-eta))

png(filename = "logistic1.png", width = 3000, height = 3000, res = 300)
par(cex = 2, mar = c(5, 4, 2, 1))
plot(eta, p,
      pch = 19,
      col = rgb(0, 0, 0, .5),
      ylab = "p(x)",
      xlab = expression(paste(eta, "(x)")))
lines(sort(eta), sort(p), col = "blue")
dev.off()
```

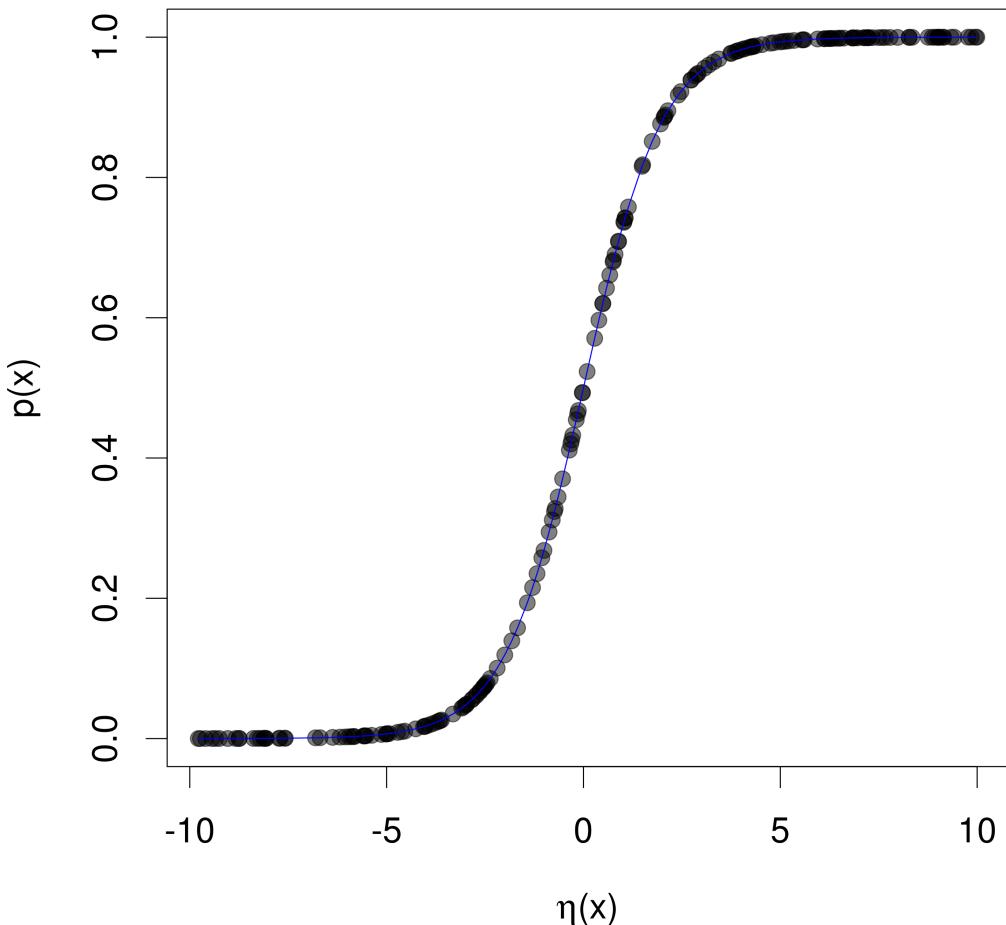


Figure 4.14: Plot of a proportion ($p(x)$) and linear predictors $\eta(x)$ related through the logistic function.

This “S-shaped curve” is the typical function people think about when doing logistic regression. Remember, however, that our response variable is a Bernoulli random variable: It has binary outcomes, not probabilities. Thus, we never actually observe something like this plot of $\eta(x)$ versus $p(x)$. Instead, in logistic regression we *assume* that there is some such logistic function relating $\eta(x)$ and $p(x)$. Also notice, that:

1. $p(x) = 0.5$ when $\eta(x) = 0$,
2. as $\eta(x)$ grows large, $p(x)$ approaches 1, and
3. as $\eta(x)$ gets small, $p(x)$ goes to zero.

This is similar to the linear case, with larger values of our regressors predicting larger values of the response variable, and vice versa. Let's look at some simulations of the Bernoulli random variable with different values of $\eta(x) = x\beta$:

```
set.seed(36)
x <- runif(200, -10, 10)
# Create our linear predictor eta(x) with beta0 = 0 and beta1 = .5
eta0 <- .5*x
# Generate Bernoulli outcomes using the logistic function
y0 <- rbinom(200, 1, 1/(1 + exp(-eta0)))
data0 <- data.frame(y0, x)
# Conduct logistic regression on the y0 data
glm0 <- glm(y0 ~ x, data = data0, family = binomial(link = "logit"))

# Create our linear predictor eta(x) with beta0 = 0 and beta1 = 1
eta1 <- x
y1 <- rbinom(200, 1, 1/(1 + exp(-eta1)))
data1 <- data.frame(y1, x)
glm1 <- glm(y1 ~ x, data = data1, family = binomial(link = "logit"))

# Create our linear predictor eta(x) with beta0 = 0 and beta1 = 2
eta2 <- 2*x
y2 <- rbinom(200, 1, 1/(1 + exp(-eta2)))
data2 <- data.frame(y2, x)
glm2 <- glm(y2 ~ x, data = data2, family = binomial(link = "logit"))
```

We've generated observations from Bernoulli random variables linked to the linear predictors through the logit function, with three values of β (.5, 1, 2) for the linear predictor $x\beta$:

```
png(filename = "logistic2.png", width = 3000, height = 3000, res = 300)
par(cex = 3, mar = c(5, 4, 2, 1), mfrow = c(3, 1), mar = c(4, 4, 2, 2))

plot(x, y0, pch = 19, col = rgb(0, 0, 0, .5), ylab = "Y0", xlab = "X")

# Generate predicted probabilities from the logistic regression
# Predictions from the logistic regression (glm) give us the logistic curve
curve(predict(glm0,
             data.frame(x = x),
             type = "resp"), # Type = "resp" gives us predicted probabilities
       add = TRUE, lwd = 2, col = "blue")
```

```
plot(x, y1, pch = 19, col = rgb(0, 0, 0, .5), ylab = "Y1", xlab = "X")
curve(predict(glm1,
             data.frame(x = x),
             type = "resp"),
       add = TRUE, lwd = 2, col = "blue")

plot(x, y2, pch = 19, col = rgb(0, 0, 0, .5), ylab = "Y2", xlab = "X")
curve(predict(glm2,
             data.frame(x = x),
             type = "resp"),
       add = TRUE, lwd = 2, col = "blue")
dev.off()
```

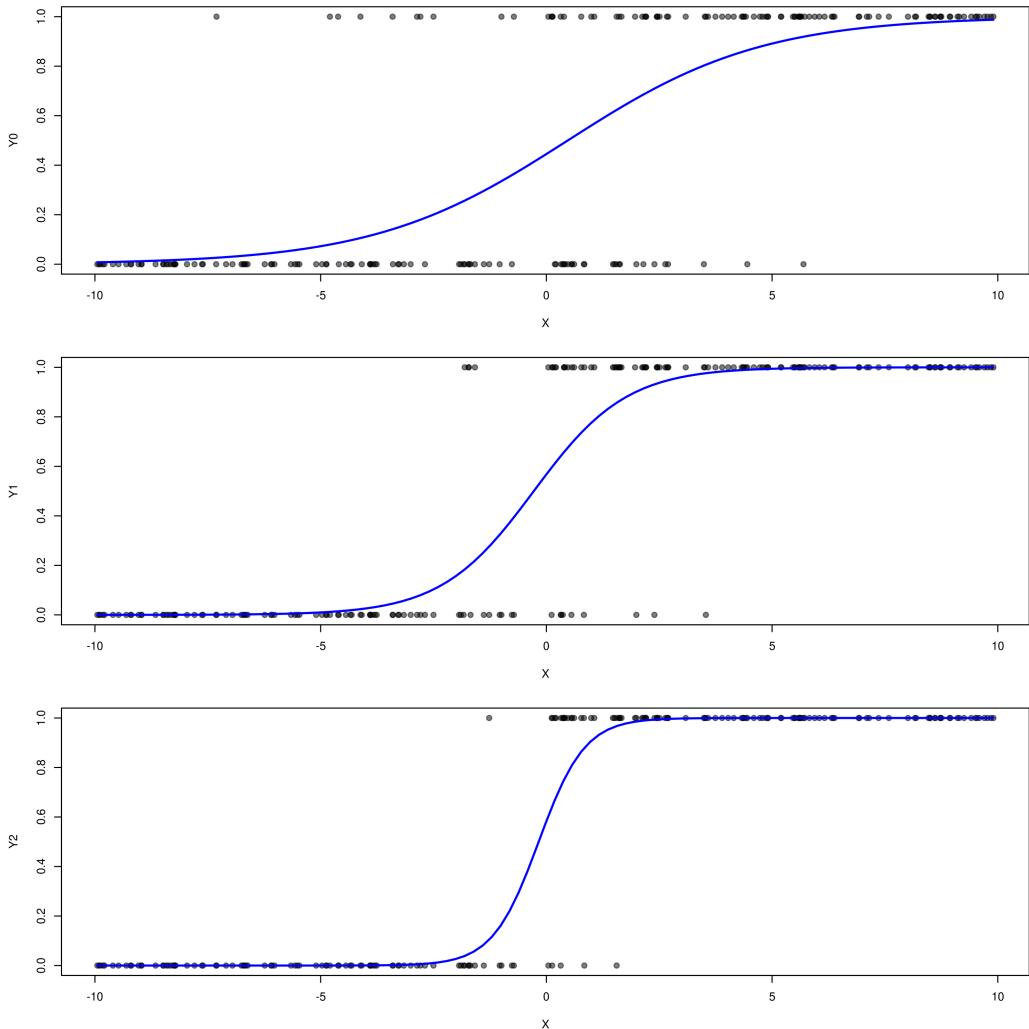


Figure 4.15: Plot of draws from a $\text{Bernoulli}\left(\frac{1}{1+e^{-\eta(x)}}\right)$ distribution and logistic regression curve for $\eta(x) = x\beta$ with $\beta \in \{0.5, 1, 2\}$ (from top to bottom).

We've reproduced our binary outcomes, and used logistic regression to discover the underlying relationship between x and p , assuming that relationship fits the logistic function (i.e., $\eta(x)$ and $p(x)$ are linked through the logit function).

Let's get a sense of how η changes the logistic curve. Roughly, the coefficient β on x can be thought of as the *discrimination* of x , or alternatively, the slope of the logistic curve. By discrimination, we mean it's ability to differentiate between the two classes (0 and 1, Yes or No). As β goes to zero, it cannot discriminate between the two classes, and the curve becomes flat.

As β approaches some threshold, the ability of x to discriminate between the classes becomes perfect. Past that threshold, x perfectly separates the two

classes, and the logistic regression algorithm will actually fail. This is called quasi-complete separation, and will be covered in more detail later in the chapter.

We can also see that the variance of the response variable y is a function of the curve. Closer to the tails there is very little variance: On the left side all values are zero (no variance), and on the right side all values are 1 (no variance). As we move toward the middle, and approach where there is equal probability in each class, the variance is at its maximum. Recall that the formula for the variance of a Bernoulli random variable is $\text{Var}(Y) = p(1 - p)$. Next, let's look at some simulations where β is negative:

```

set.seed(36)
x <- runif(200, -10, 10)
eta0 <- -.5*x
y0 <- rbinom(200, 1, 1/(1 + exp(-eta0)))
data0 <- data.frame(y0, x)
glm0 <- glm(y0 ~ x, data = data0, family = binomial(link = "logit"))

eta1 <- -x
y1 <- rbinom(200, 1, 1/(1 + exp(-eta1)))
data1 <- data.frame(y1, x)
glm1 <- glm(y1 ~ x, data = data1, family = binomial(link = "logit"))

eta2 <- -2*x
y2 <- rbinom(200, 1, 1/(1 + exp(-eta2)))
data2 <- data.frame(y2, x)
glm2 <- glm(y2 ~ x, data = data2, family = binomial(link = "logit"))

png(filename = "logistic3.png", width = 3000, height = 3000, res = 300)
par(cex = 3, mfrow = c(3, 1), mar = c(5, 4, 2, 1))
plot(x, y0, pch = 19, col = rgb(0, 0, 0, .5), ylab = "Y0", xlab = "X")
curve(predict(glm0,
             data.frame(x = x),
             type = "resp"),
       add = TRUE, lwd = 2, col = "blue")

plot(x, y1, pch = 19, col = rgb(0, 0, 0, .5), ylab = "Y1", xlab = "X")
curve(predict(glm1,
             data.frame(x = x),
             type = "resp"),
       add = TRUE, lwd = 2, col = "blue")

plot(x, y2, pch = 19, col = rgb(0, 0, 0, .5), ylab = "Y2", xlab = "X")

```

```

curve(predict(glm2,
            data.frame(x = x),
            type = "resp"),
      add = TRUE, lwd = 2, col = "blue")
dev.off()

```

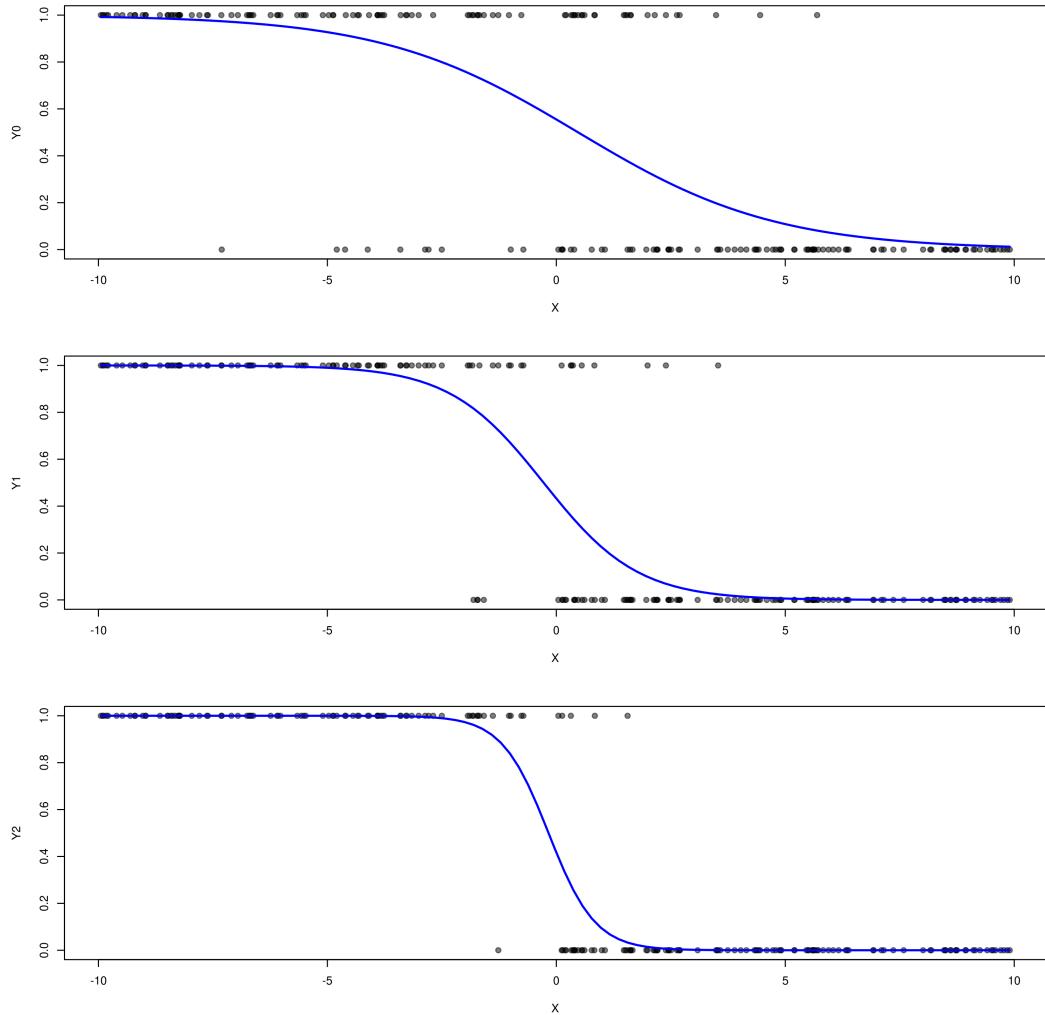


Figure 4.16: Plot of draws from a $\text{Bernoulli}\left(\frac{1}{1+e^{-\eta(x)}}\right)$ distribution and logistic regression curve for $\eta(x) = x\beta$ with $\beta \in \{-0.5, -1, -2\}$ (from top to bottom).

When β is negative, the curve is reflected around the mean of x . Now, as x increases, the probability of observing a 1 *decreases*. Nothing else changes. Next, let's add an intercept β_0 and see how that changes the logistic curve:

```
x <- runif(200, -10, 10)
```

```

# Set beta0 to -5
eta0 <- x - 5
y0 <- rbinom(200, 1, 1/(1 + exp(-eta0)))
data0 <- data.frame(y0, x)
glm0 <- glm(y0 ~ x, data = data0, family = binomial(link = "logit"))

# Set beta0 to 0
eta1 <- x
y1 <- rbinom(200, 1, 1/(1 + exp(-eta1)))
data1 <- data.frame(y1, x)
glm1 <- glm(y1 ~ x, data = data1, family = binomial(link = "logit"))

# Set beta0 to 5
eta2 <- x + 5
y2 <- rbinom(200, 1, 1/(1 + exp(-eta2)))
data2 <- data.frame(y2, x)
glm2 <- glm(y2 ~ x, data = data2, family = binomial(link = "logit"))

png(filename = "logistic4.png", width = 3000, height = 3000, res = 300)
par(cex = 3, mfrow = c(3, 1), mar = c(5, 4, 2, 1))
plot(x, y0, pch = 19, col = rgb(0, 0, 0, 0.5), ylab = "Y0", xlab = "X")
curve(predict(glm0,
             data.frame(x = x),
             type = "resp"),
       add = TRUE, lwd = 2, col = "blue")

plot(x, y1, pch = 19, col = rgb(0, 0, 0, .5), ylab = "Y1", xlab = "X")
curve(predict(glm1,
             data.frame(x = x),
             type = "resp"),
       add = TRUE, lwd = 2, col = "blue")

plot(x, y2, pch = 19, col = rgb(0, 0, 0, .5), ylab = "Y2", xlab = "X")
curve(predict(glm2,
             data.frame(x = x),
             type = "resp"),
       add = TRUE, lwd = 2, col = "blue")
dev.off()

```

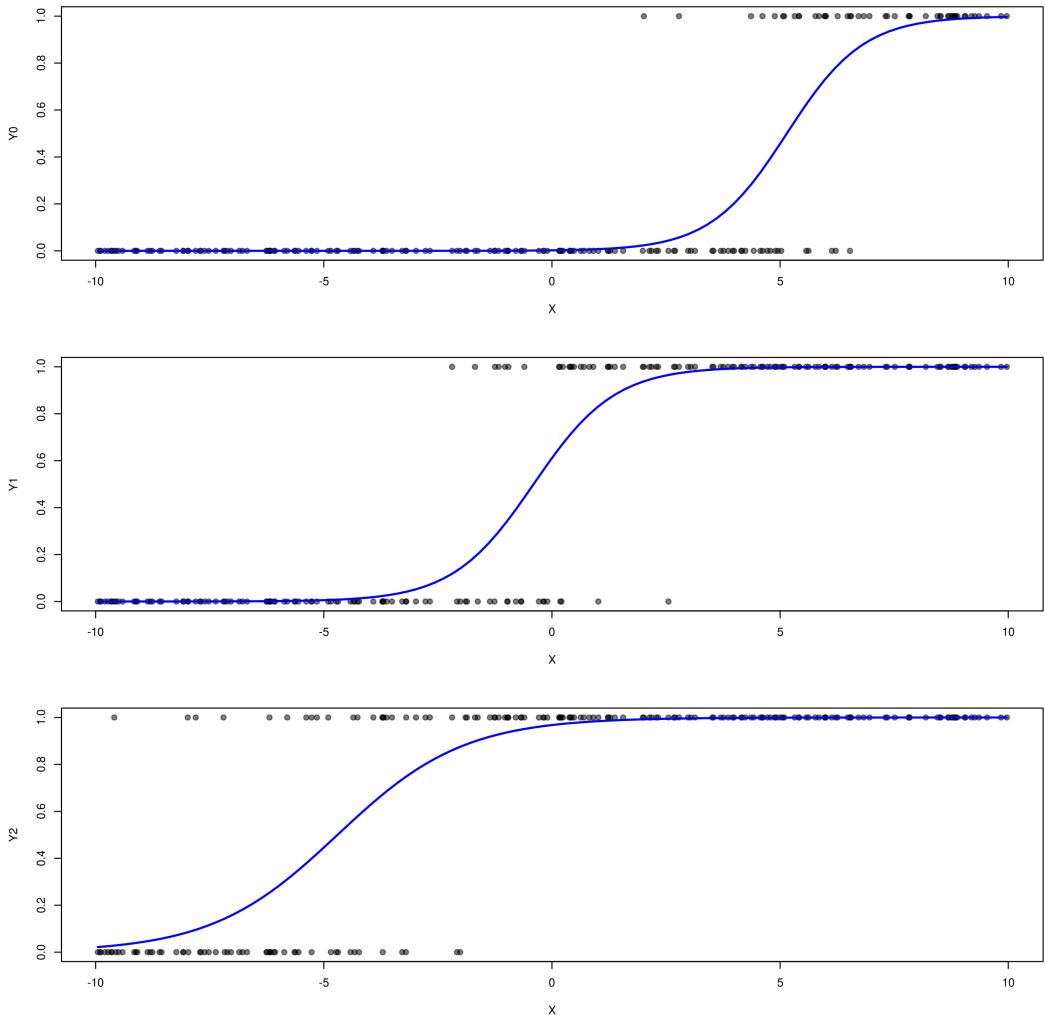


Figure 4.17: Plot of draws from a Bernoulli($\frac{1}{1+e^{-\eta}}$) distribution and logistic regression curve for $\eta(x) = \beta_0 + \beta x$ with $\beta_0 \in \{-5, 0, 5\}$ and $\beta = 1$ (from top to bottom).

With $\beta_0 = -5$ the curve shifts to the right relative to $\beta_0 = 0$. The point at which we begin to observe positive outcomes ($Y_0 = 1$) is shifted to the right. In general, decreasing β_0 (the intercept), decreases the proportion of positive outcomes, and shifts the logistic curve to the right. The opposite is true for $\beta_0 = 5$, where most observations are ($Y_3 = 1$), except for some where x has large negative values.

Another way of looking at β_0 is that it is a difficulty parameter. Suppose students are answering questions on a test. If they get it right $Y = 1$; if they get it wrong $Y = 0$. A test question with large positive β_0 is one that more students get right; a test question with large negative β_0 is one that more

students get wrong. All else equal, β_0 is a measure of the *difficulty* of the problem/task/question.

4.4 Interpreting Logistic Regression

Now that we have some idea of what logistic regression is, and how maximum likelihood estimation works, let's continue our voting example. We can fit a logistic regression using R's `glm` function (for generalized linear model). `glm` is very flexible and is able to model different exponential families, such as the Poisson or Negative Binomial distribution. We can also use different link functions besides the logit, such as probit or complementary log-log:

```
voteglm <- glm(voterep ~ conserv, data = votes,
# Specify the stochastic component (binomial/bernoulli)
# Specify the link function "logit"
family = binomial(link = "logit"))
summary(voteglm)
```

If we have a random sample, we can form confidence intervals around our parameters on the logit scale just as we did in linear regression as ± 2 standard errors. The `summary` function provides us the standard errors. Again, if the sampling is not random, the coverage probability of the confidence interval will *not* be 95%. In fact, we have no idea how far we are off. Let's look at the predicted probabilities of voting republican:

```
png(filename = "glmvotes.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(votes$conserv), jitter(votes$voterep, amount = 0.025),
     xlab = "Conservativism",
     ylab = "Predicted Probability of Voting for Republican",
     ylim = c(-.2, 1.2),
     pch = 19,
     col = rgb(0, 0, 0, .3))
# Add curve of predicted probabilities
curve(predict(voteglm, data.frame(conserv = x), type = "resp"), add = TRUE)
dev.off()
```

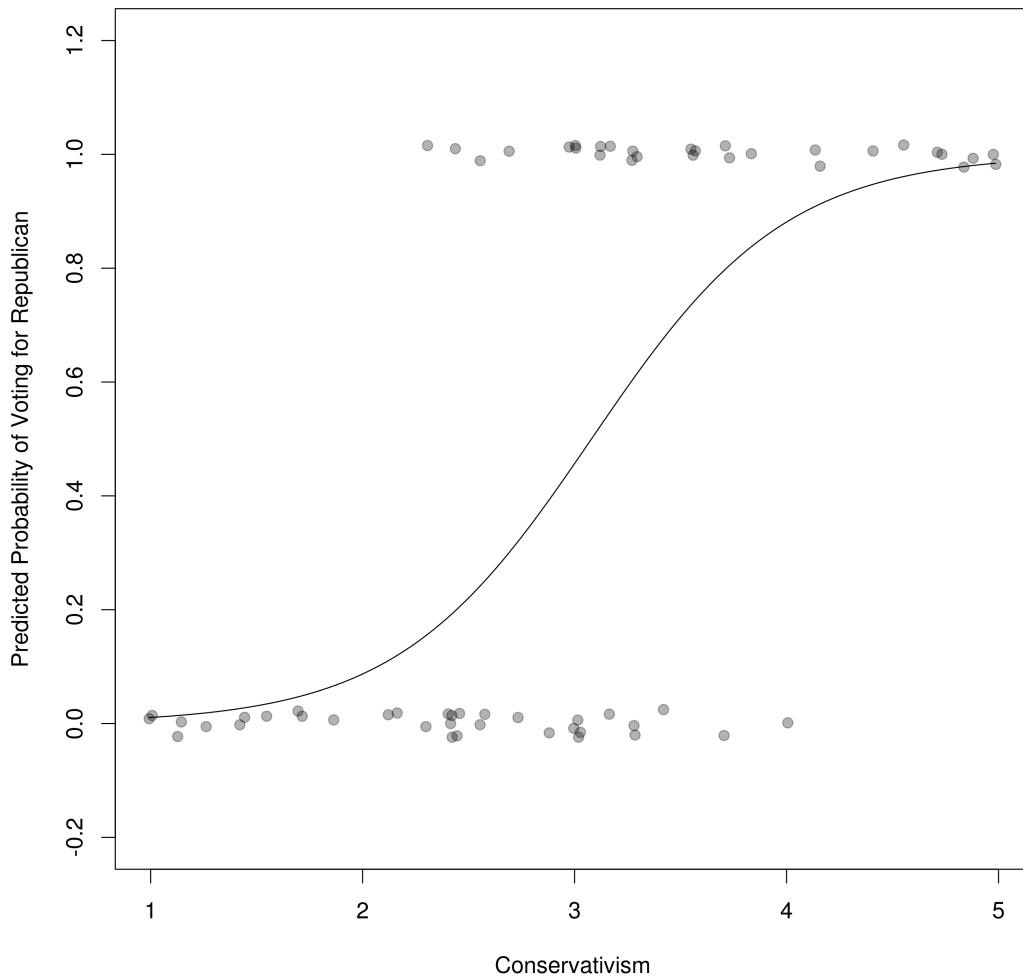


Figure 4.18: Predicted probability of voting for a republican given different levels of conservatism using logistic regression.

Now, using logistic regression there are no predicted probabilities greater than 1 or less than 0. We can see that the predicted probability of voting republican is a smooth logistic function of conservatism, as we should expect. Let's interpret the output:

```
summary(voteglm)
```

The output of the logistic regression tells us information about the parameters of $\hat{\eta}(x)$ on the *logit* scale:

$$\text{logit}(\text{Voterep}|x) = \log\left(\frac{P(\text{Voterep}|x)}{1 - P(\text{Voterep}|x)}\right) = -6.70 + 2.18 \times \text{Conserv}$$

For example, if the person's report on the conservativism scale is 1, then $\text{logit}(\text{Voterep}) = -6.7 + 2.18 \times 1 = -4.52$. Of course, this really isn't that helpful. We do know from the relationship between the logit function and p that because the logit is less than zero, it follows that $p < .5$. To get the actual average predicted probability of voting republican for people with a conservativism score of 1, we need to use our logistic transformation:

$$P(\text{Voterep} | \text{Conserv} = 1) = \frac{1}{1 + e^{-[-6.70 + 2.18 \times 1]}} = .011$$

Thus, the predicted probability of voting republican for people with a conservativism score of 1 is, on average, $P(\text{Voterep}) = 0.011$; that is they are very, very unlikely to vote conservative.

For the coefficient on conserv, a one unit increase in conservatism score is associated with an increase of 2.18 logit(voterep), or an increase of 2.18 in the log odds of voting republican. Again, this isn't a very useful interpretation of the coefficient, and importantly, the relationship between the predicted probability and change in the independent variable is non-linear. We can see this by looking at the logistic curve from our regression. A one unit increase in conservativism from a score of 4 to 5 has a relatively small increase on the probability of voting republican:

```
# Predict average probability of voting republican
# for conservative scores of 4 and 5
cons4 <- predict(voteglm, newdata = data.frame(conserv = 4), type = "response")
cons5 <- predict(voteglm, newdata = data.frame(conserv = 5), type = "response")
# Calculate the difference
# in predicted probabilities between 5 and 4
prob.diff4to5 <- cons5 - cons4
# Calculate the risk ratio of probabilities between 5 and 4
risk.ratio4to5 <- cons5/cons4
# Calculate the odds of voting republican
# for those with a conservativism score of 4/5
odds4 <- cons4/(1 - cons4)
odds5 <- cons5/(1 - cons5)
# Calculate the odds ratio
# for those with a conservativism score of 5 compared to 4
odds.ratio4to5 <- odds5/odds4
```

Thus, an increase of one score on the conservativism scale from 4 to 5 is associated with a probability difference of 0.10, a risk ratio of 1.2, and an odds ratio of 8.82. The odds ratio is much larger than the risk ratio, reflecting the fact that the risk ratio is highly insensitive to changes in probability near 0 and 1. What about an increase in conservativism from 3 to 4?

```

cons3 <- predict(voteglm, newdata = data.frame(conserv = 3), type = "response")
cons4 <- predict(voteglm, newdata = data.frame(conserv = 4), type = "response")
prob.diff3to4 <- cons4 - cons3
risk.ratio3to4 <- cons4/cons3
odds3 <- cons3/(1 - cons3)
odds4 <- cons4/(1 - cons4)
odds.ratio3to4 <- odds4/odds3

```

An increase of one score on the conservativism scale from 3 to 4 is associated with a probability difference of 0.42, a risk ratio of 1.93, and an odds ratio of 8.82. The probability difference and risk ratio indicate a much larger effect when we are near the middle probabilities (where conservatism is 3) than near the tail probabilities (where conservatism is very high (5) or very low (1)). This dependence on the underlying probability of the probability difference and risk ratio make them poor indicators of the marginal effect of our independent variable. That is, we want a single summary measure of the effect of our independent variable, not a different effect for each predicted probability.

We observed that the odds ratio is independent of the underlying probabilities we are using in our comparison. Thus, odds ratios are unintuitive because they are not linked to probabilities, but this is the same reason that they are a good measure of marginal effect. It's quite a conundrum for users of logistic regression that an invariant measure of marginal effects is so unintuitive.

One way to find the odds ratio is to use the definition of the logistic regression model. Recall that the model is linear in log-odds:

$$\text{logit}(\text{Voterep}|x) = \log\left(\frac{P(\text{Voterep}|x)}{1 - P(\text{Voterep}|x)}\right) = -6.70 + 2.18 \times \text{Conserv}$$

The marginal effect is the difference between the log odds when $x = \text{conserv} = 1$ and $x = \text{conserv} = 0$:

$$\log\left(\frac{P(\text{Voterep}|x = 1)}{1 - P(\text{Voterep}|x = 1)}\right) - \log\left(\frac{P(\text{Voterep}|x = 0)}{1 - P(\text{Voterep}|x = 0)}\right)$$

This gives us:

$$-6.70 + 2.18 \times 1 - (-6.70 + 2.18 \times 0) = 2.18$$

That is, the difference in log odds when the conservativism score increases by 1 unit is 2.18, just as we would expect from linear regression. However, again, we really don't care that much about differences on the log odds scale.

4.4.1 Odds Ratio in Logistic Regression

Recall that $\frac{P(\text{Voterep}|x)}{1-P(\text{Voterep}|x)}$ is the odds of voting republican given some value of the regressors x . As we saw above, the marginal effect is the difference in log odds ($\log(\text{Odds}(\text{Voterep}|x=1)) - \log(\text{Odds}(\text{Voterep}|x=0))$). One of the basic properties of logarithms is that the log of the difference between two numbers is equal to the ratio of the logs. Using this fact, we have:

$$\log(\text{Odds}(\text{Voterep}|x=1)) - \log(\text{Odds}(\text{Voterep}|x=0)) = \log\left(\frac{\text{Odds}(\text{Voterep}|x=1)}{\text{Odds}(\text{Voterep}|x=0)}\right)$$

We've already shown that, on the log odds scale, the difference in log odds when $x = 1$ and $x = 0$ was 2.18, so the log of the odds ratio is 2.18:

$$\log\left(\frac{\text{Odds}(\text{Voterep}|x=1)}{\text{Odds}(\text{Voterep}|x=0)}\right) = 2.18$$

If we want to find the odds *ratio*, all we need to do is exponentiate the log of the log odds ratio:

$$\text{Odds Ratio} = e^{\log\left(\frac{\text{Odds}(\text{Voterep}|x=1)}{\text{Odds}(\text{Voterep}|x=0)}\right)} = e^{2.18} = 8.82$$

Thus, we found the odds ratio, a measure of the marginal effect of the regressor of interest, by simply exponentiating the coefficient from the log-odds (logit) regression. In words, for a 1-unit increase in the conservativism score, the *odds* of voting for the republican are 8.82 times greater. What if we were to ask about a 2-unit increase in conservatism?

$$\log\left(\frac{P(\text{Voterep}|x=2)}{1-P(\text{Voterep}|x=2)}\right) - \log\left(\frac{P(\text{Voterep}|x=0)}{1-P(\text{Voterep}|x=0)}\right) = 2.18 \times 2 = 4.36$$

Then $e^{4.36} = e^{2.18} \times e^{2.18} = (\text{odds ratio})^2 = 78.3$. For a 2-unit increase in conserv, the odds increases by a factor of $8.82^2 = 78.3$. Thus, the effect is multiplicative. Alternatively, for a 1-unit decrease in conserv, the odds of voting for a republican decrease by a factor of 8.82. The odds ratio is $1/8.82 = 0.113$.

4.4.2 Average Marginal Effect

Because odds ratios are so unintuitive, whereas probabilities are intuitive, we often want to figure out a way to translate our logistic regression results into probabilities. Recall that the marginal effect in terms of probability differences or the risk ratio both depend on the predicted probability. The predicted probability is the conditional mean of the logistic function

calculated at the values of the regressors x_1 . For logistic regression, the marginal effect of a regressor x_1 is then:

$$\frac{\partial}{\partial x_1} \left(\frac{1}{1 + e^{-\hat{\eta}(x)}} \right) = \hat{\beta}_1 \frac{e^{\hat{\eta}(x)}}{(1 + e^{\hat{\eta}(x)})^2} \quad (4.24)$$

This is the coefficient $\hat{\beta}_1$ from the log-odds regression times the derivative of the logistic cumulative distribution function. The derivative of the logistic cumulative distribution is the logistic density $\frac{e^{\hat{\eta}(x)}}{(1 + e^{\hat{\eta}(x)})^2}$, which is also equal to $p(x)(1 - p(x))$. Here we're interpreting this as the “slope” of the logistic function at a specific value. Recalling the logistic function we've considered so far, it is steepest in the middle where $p = 0.5$ then the slope tapers symmetrically on both sides. Thus, we should expect that the predicted marginal effect is largest when the predicted probability is 0.5, which is when $\eta(x) = 0$. In fact, when the marginal effect is largest (slope of the logistic curve is steepest), the change in probability reduces to $\hat{\beta}_1 \frac{e^0}{(1 + e^0)^2} = \frac{\hat{\beta}_1}{4}$, meaning the largest change in probability associated with a change in x_1 is $\frac{\hat{\beta}_1}{4}$:

```
voteglm <- glm(voterep ~ conserv,
                 data = votes,
                 family = binomial(link = "logit"))
# Predictions from the logistic regression on the log odds scale
eta.hat <- predict(voteglm)
# Predictions from the logistic regression on the probability scale
preds.prob <- predict(voteglm, type = "response")
# Find the value of the logistic probability density
# Evaluated at each value of eta.hat
logistic.pdf <- dlogis(eta.hat)
# The marginal effect is just the coefficient in the regression
# times the logistic pdf evaluated at different eta.hat values
marginal.effect <- coef(voteglm)[2]*logistic.pdf

png(filename = "marginaleffects.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(preds.prob, amount = 0.025),
     jitter(marginal.effect, amount = 0.025),
     ylab = "Marginal Effect (Probability Difference)",
     xlab = "Predicted Probability",
     pch = 19,
     col = rgb(0, 0, 0, .3),
     ylim = c(0, 0.6))
lines(preds.prob, marginal.effect)
dev.off()
```

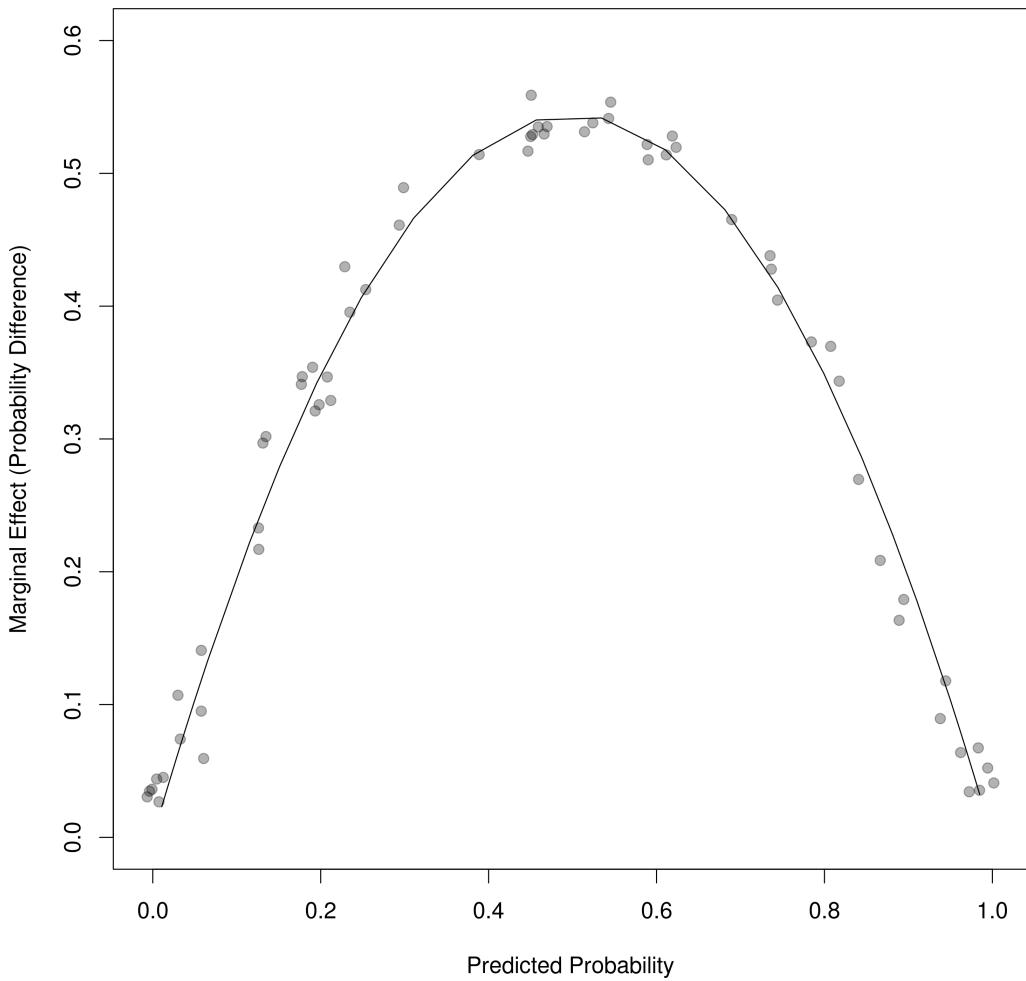


Figure 4.19: Marginal effect of conservatism on the predicted probability of voting for a republican given different levels of predicted probability.

The plot shows what we should expect. The marginal effect (probability difference) is largest when the predicted probability is 0.5 (recall that the logistic function is steepest here). The marginal effect is close to zero when probabilities are close to zero and one, as we should expect given the “tapering” of the logistic function at these endpoints. If we want to compute the *average marginal effect*, just take the average of our computed marginal effects:

```
AME <- mean(marginal.effect)
```

The average marginal effect tells us that, averaging across all observations, a small increase in the conservatism score is associated with a 0.31 increase

in the probability of voting republican. We can use the bootstrap get some sense of the uncertainty in the average marginal effect:

```

set.seed(40)
# Create an empty vector to store our bootstrapped AMEs
AME.boot <- c()
for(i in 1:100){
  # Take a random sample, with replacement from the rows of the votes data
  samp1 <- votes[sample(1:dim(votes)[1], replace = T, dim(votes)[1]), ]
  # Run the regression on this sample
  voteglm <- glm(voterep ~ conserv, data = samp1,
                  family = binomial(link = "logit"))

  # Calculate the marginal effects for the bootstrapped sample
  eta.hat <- predict(voteglm)
  preds.prob <- predict(voteglm, type = "response")
  logistic.pdf <- dlogis(eta.hat, 0, 1)
  marginal.effect <- coef(voteglm)[2]*logistic.pdf
  AME.boot <- append(AME.boot, mean(marginal.effect))
}
mean(AME.boot)
quantile(AME.boot, 0.025) # find the lower .025 quantile
quantile(AME.boot, 0.975) # find the upper .975 quantile

```

Over 100 bootstrap iterations, the mean average partial effect is 0.31, with 95% of the average marginal effects falling between 0.25 and 0.38. It might be better to look at the median marginal effect or the quantiles of the marginal effects.

If we considered the conservatism score discrete, it would be more appropriate to compare the effect on the predicted probability of voting republican for a one unit change in conservatism, (rather than an infinitesimal change as is done when using the derivative):

```

# Get the predicted probability for each observation
votes$prob <- predict(voteglm, type = "response")
# Create a new data frame with the conservative score + 1
new.data <- votes
new.data$conserv <- votes$conserv + 1

# Make predictions onto the new data frame
votes$prob.inc <- predict(voteglm,
                           type = "response",
                           newdata = new.data)

```

```

# Compare predicted probabilities for same observations
# with conserv + 1 vs old value; exclude those who
# will have conserv > 5 (an impossible value)
marginal.effect <- votes$prob.inc[votes$conserv < 5]
- votes$prob[votes$conserv < 5]

png(filename = "discreteme.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(votes$prob[votes$conserv < 5], amount = 0.025),
     jitter(avg.me, amount = 0.025),
     ylab = "Discrete Marginal Effect",
     xlab = "Predicted Probability",
     pch = 19,
     col = rgb(0, 0, 0, .5))
lines(votes$prob[votes$conserv < 5], avg.me)
dev.off()

```

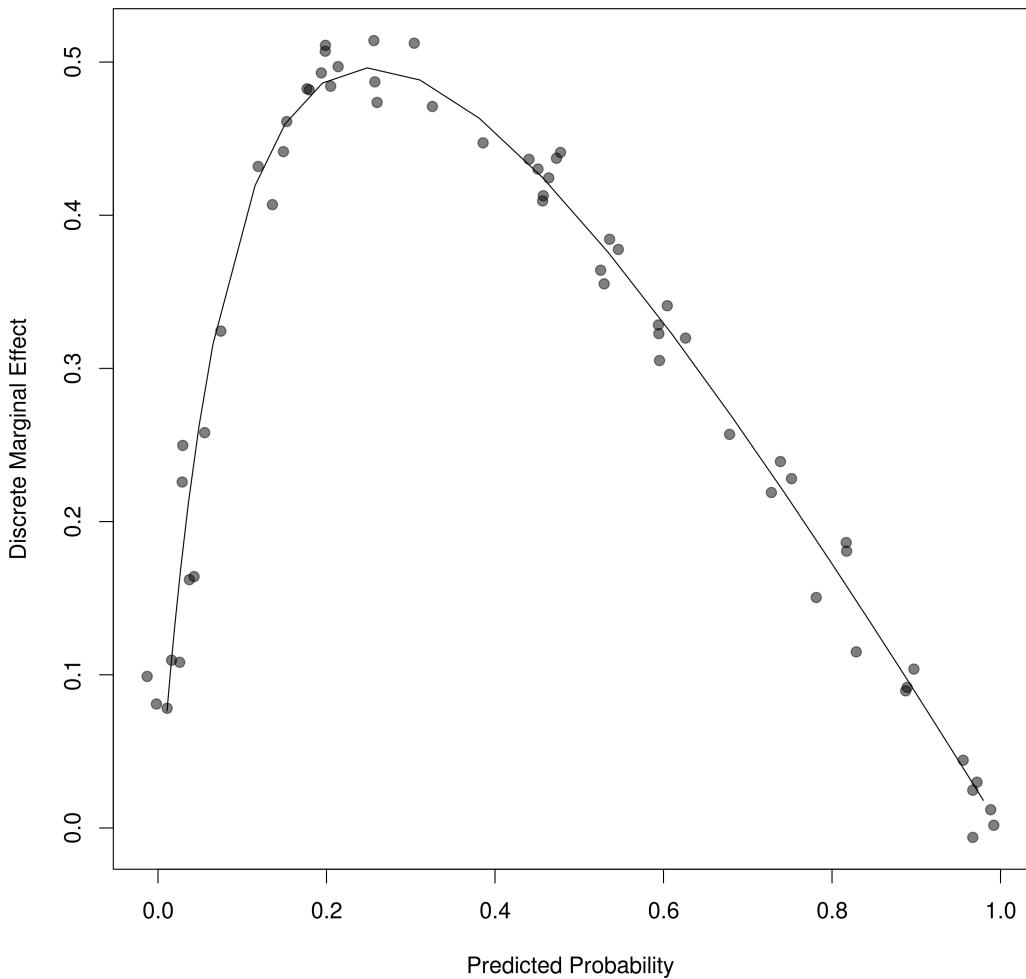


Figure 4.20: Discrete marginal effect of conservatism on the predicted probability of voting for a republican given different levels of predicted probability.

As expected, the marginal effects are largest for predicted probabilities below 0.5 because those observations are on the steepest part of the logistic curve.

4.4.3 Arsenic in Bangladesh Example

Here we use data from [Gelman and Hill \(2007\)](#) about whether households decide to switch drinking water wells when given a warning about unsafe arsenic levels in their well. If a well has too much arsenic, households must decide whether to walk to the nearest well with safe levels of arsenic. Researchers measured arsenic levels and encouraged households to switch wells if they found unsafe levels in their well. The outcome is whether the

household switched wells ($Y = 1$) or not ($Y = 0$). The data we have include the distance to the closest safe well, the arsenic level of the household's well, whether members of the household are active in community organizations, and the education level of the head of household:

```
wells <- read.table(
  "http://www.stat.columbia.edu/~gelman/arm/examples/arsenic/wells.dat")
wells.glm <- glm(switch ~ dist, data = wells,
                  family = binomial(link = "logit"))
summary(wells.glm)
```

The estimated regression function is:

$$\log\left(\frac{P(\text{switch}|\text{distance})}{1 - P(\text{switch}|\text{distance})}\right) = 0.61 - 0.006 \times \text{distance}$$

Distance here is measured in meters, so we can interpret the coefficient of distance in the regression as households that were one meter further from a safe well as having an odds of switching wells that is $0.994 (= e^{-0.006})$ times the odds off switching compared to a household one meter closer. If we were to compare a household that was 10 meters farther away, it would have an odds of switching wells $0.94 (= 0.994^{10})$ times the odds of a household that was 10 meters closer. I find it easier to interpret these in terms of greater odds, so we can say that a household that is one meter *closer* to the nearest safe drinking water well has an odds of switching wells that is 1.006 times greater ($= 1/e^{-0.006}$) than a household that is one meter further away, and a household that is 10 meters closer to a safe well has an odds of switching wells that is 1.06 times greater ($= 1/0.994^{10}$) than a household 10 meters further from the nearest well.

An easier rule of thumb is to evaluate the change in predicted probability of switching at its maximum value, for households with a predicted probability of switching wells of 0.5. As mentioned, this change in predicted probability is the “divide by 4” rule, where we just divide the coefficient from the log-odds model by 4, giving $-0.0015 (= -0.0062/4)$. Thus, the probability of switching decreases by, at most, 0.0015 points for an additional meter of distance to the nearest safe well, corresponding to a household with a predicted probability of switching of 0.50. Next, we can plot the fitted model:

```
png(filename = "wells1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(wells$dist), jitter(wells$switch, amount = .025),
      xlab = "Distance from Nearest Safe Well (Meters)",
      ylab = "Predicted Probability of Switching Wells",
      ylim = c(-.1, 1.1),
```

```
    pch  = 19,  
    col  = rgb(0, 0, 0, .05))  
  
# Add the fit from the model  
curve(1/(1 + exp(-(coef(wells.glm)[1] + coef(wells.glm)[2]*x))),  
      col = rgb(0, 0, 0, 1), add = TRUE)  
dev.off()
```

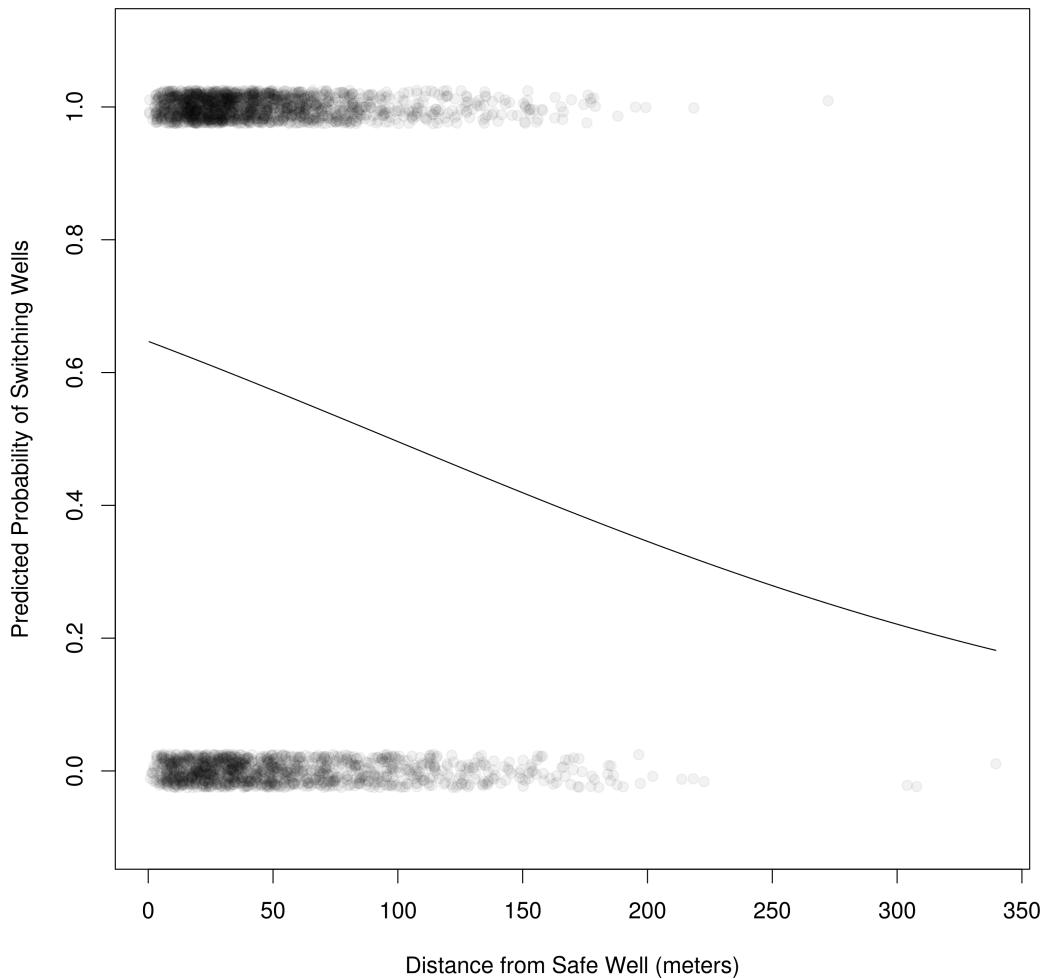


Figure 4.21: Plot of the predicted probability of switching wells from the logistic regression given the distance from the nearest safe well (in meters).

We can see that most distances from the nearest safe well are under 150 meters, with a distribution that is positively skewed with the mode between 0 and 50 meters.

```
png(filename = "wellsdist.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
hist(wells$dist,
     breaks = "FD",
     xlab    = "Distance (Meters)",
     ylim    = c(0, 300),
     main   = "")
dev.off()
```

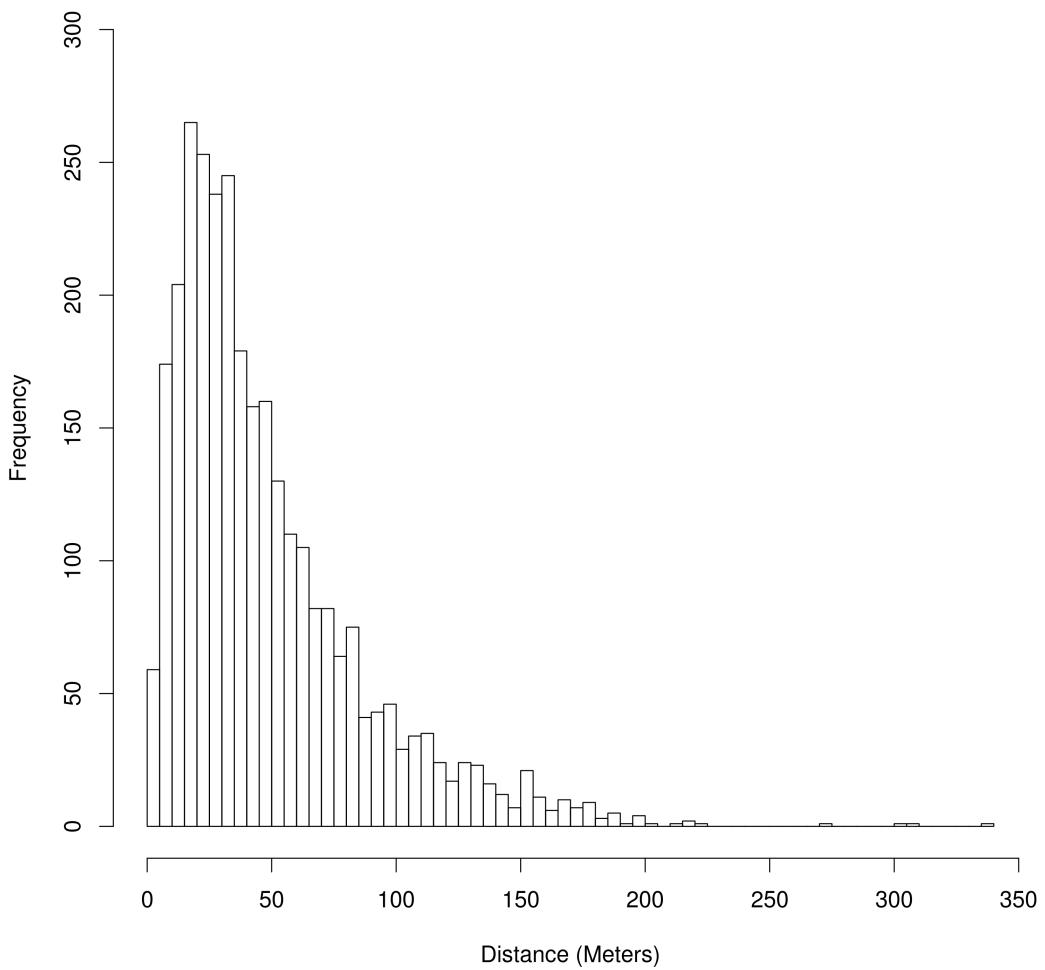


Figure 4.22: Histogram of distances to the nearest safe well, measured in meters.

We can also see that the predicted probability of switching wells decreases as the distance from the nearest safe well increases, as we would expect given that traveling is costly. There's also one household that was willing to travel over 250 meters to find a safe well! This would likely be an influential observation, which in the context of logistic regression, is an observation that is “unexpected” (i.e., we predict a low probability of switching, but they switch). Most people lived within 50 meters of a safe well and were willing to switch. The constant in the regression tells us how likely people are to switch when the distance is zero, which is $\frac{1}{1+e^{-0.61}}$ in this case, or 0.65.

Next, we'll add a second predictor variable, the level of arsenic measured in the household's nearest well. The arsenic levels are also highly positively

skewed:

```
png(filename = "wellsarsenic.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
hist(wells$arsenic,
      breaks = "FD",
      xlab   = "Arsenic Levels (micrograms/L)",
      ylim   = c(0, 500),
      xlim   = c(0, 10),
      main   = "")
dev.off()
```

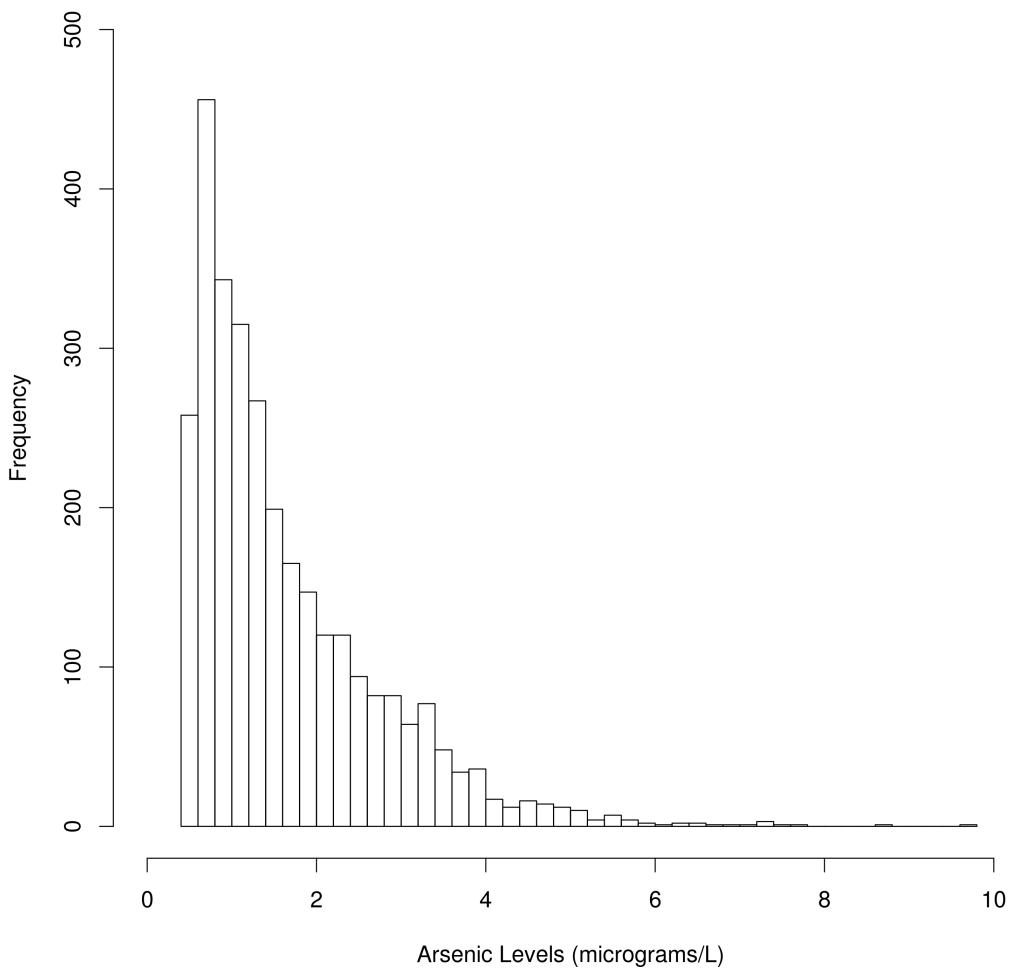


Figure 4.23: Histogram of arsenic levels (micrograms/L) measured in the household's well.

Here's the new regression:

```
wellss.glm2 <- glm(switch ~ dist + arsenic, data = wells,
                      family = binomial(link = "logit"))
summary(wellss.glm2)
```

The estimated regression function is:

$$\log\left(\frac{P(\text{switch}|\text{distance, arsenic})}{1 - P(\text{switch}|\text{distance, arsenic})}\right) = 0.003 - 0.009 \times \text{distance} + 0.46 \times \text{arsenic}$$

The coefficient of arsenic in the regression for the log-odds model is 0.46, indicating a maximum increase in probability of 0.12 ($= 0.46/4$) points for an

additional microgram/L of arsenic in the water for households with a predicted probability of switching of 0.5. The coefficient on distance is now more negative, indicating the probability of switching decreases even more than in the previous regression for households one meter further from the nearest safe well. There's two reasons this can happen: 1) arsenic and distance may be correlated, indicating there was omitted variable bias where arsenic is positively correlated to switching but negatively correlated to the distance from the nearest safe well,² and 2) because arsenic belongs in the regression, the variance of the conditional mean changes, which is equivalent to the conditional mean changing (as $\text{Var}(Y|x) = p(x)(1 - p(x))$).³ We can plot the probability of switching wells given the linear predictors $\hat{\eta}(x)$:

```

eta.hat <- predict(wells.glm2)
prob <- predict(wells.glm2, type = "response")
png(filename = "wells2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(eta.hat), jitter(wells$switch, amount = .025),
     xlab = expression(paste("Value of the Linear Predictors ", eta, "(x"))),
     ylab = "Predicted Probability of Switching Wells",
     xlim = c(-3, 4),
     ylim = c(-.1, 1.1),
     pch = 19,
     col = rgb(0, 0, 0, .1))

# Add the fit from the model
lines(sort(eta.hat), sort(prob), col = rgb(0, 0, 0, 1))
dev.off()

```

²Households with higher arsenic levels in their wells have a closer safe well that is nearest to them than those with lower arsenic levels.

³Because changes in the variance affect the conditional mean, omitting variables that are *uncorrelated* with a regressor changes the parameter on the regressor ([Yatchew and Griliches, 1985](#)).

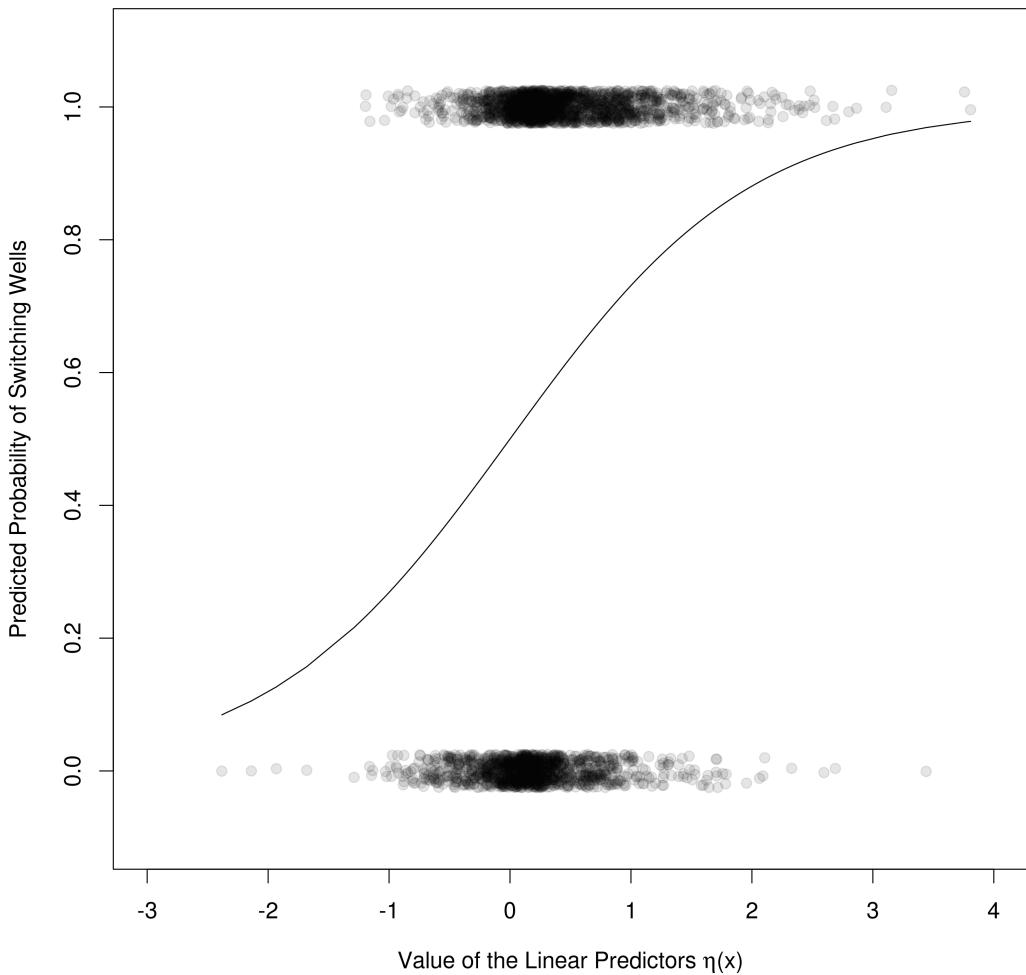


Figure 4.24: Plot of the predicted probability of switching wells from the logistic regression given the fitted linear predictors $\hat{\eta}(x)$, where $\hat{\eta}(x)$ includes arsenic levels (in micrograms/L) and distance from the nearest safe well (in meters).

From this plot we can see that there are only four households with large negative values of the linear predictors, households which likely have both low arsenic levels and high distance from the nearest well. There's the potential for these to be "outliers" as they will likely have a large influence on the fitted curve. There's also several houses that are "unexpected," in the sense that they have high predicted probabilities of switching wells (> 0.8), but did not switch.

We can also make plots of predicted switching probabilities versus one regressor at specific levels of the other. For example, we can look at the

relationship between probability of switching and distance at different levels of arsenic, or vice versa:

```
png(filename = "wells3.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(wells$dist), jitter(wells$switch, amount = .025),
     xlab = "Distance from Nearest Safe Well (Meters)",
     ylab = "Predicted Probability of Switching Wells",
     ylim = c(-.1, 1.1),
     pch = 19,
     col = rgb(0, 0, 0, .05))

#Calculate the mean arsenic levels
mean.arsenic <- mean(wells$arsenic)
# Plot curve at the mean of the arsenic levels
curve(1/(1 + exp(-(coef(wells.glm2)[1] +
                      coef(wells.glm2)[2]*x +
                      coef(wells.glm2)[3]*mean.arsenic))),
      col = rgb(0, 0, 0, 1), add = TRUE, lwd = 2)

# Create 4 evenly spaced arsenic levels
arsenic.level <- seq(0.5, 9.5, by = 3)
# Plot probability by distance at 4 levels of arsenic
for(i in 1:4){
  # Add the fit from the model for different levels of arsenic
  curve(1/(1 + exp(-(coef(wells.glm2)[1] +
                        coef(wells.glm2)[2]*x +
                        coef(wells.glm2)[3]*arsenic.level[i]))),
        col = rgb(1, 0, 0, i/4), add = TRUE, lwd = 2, lty = 2)
}
dev.off()
```

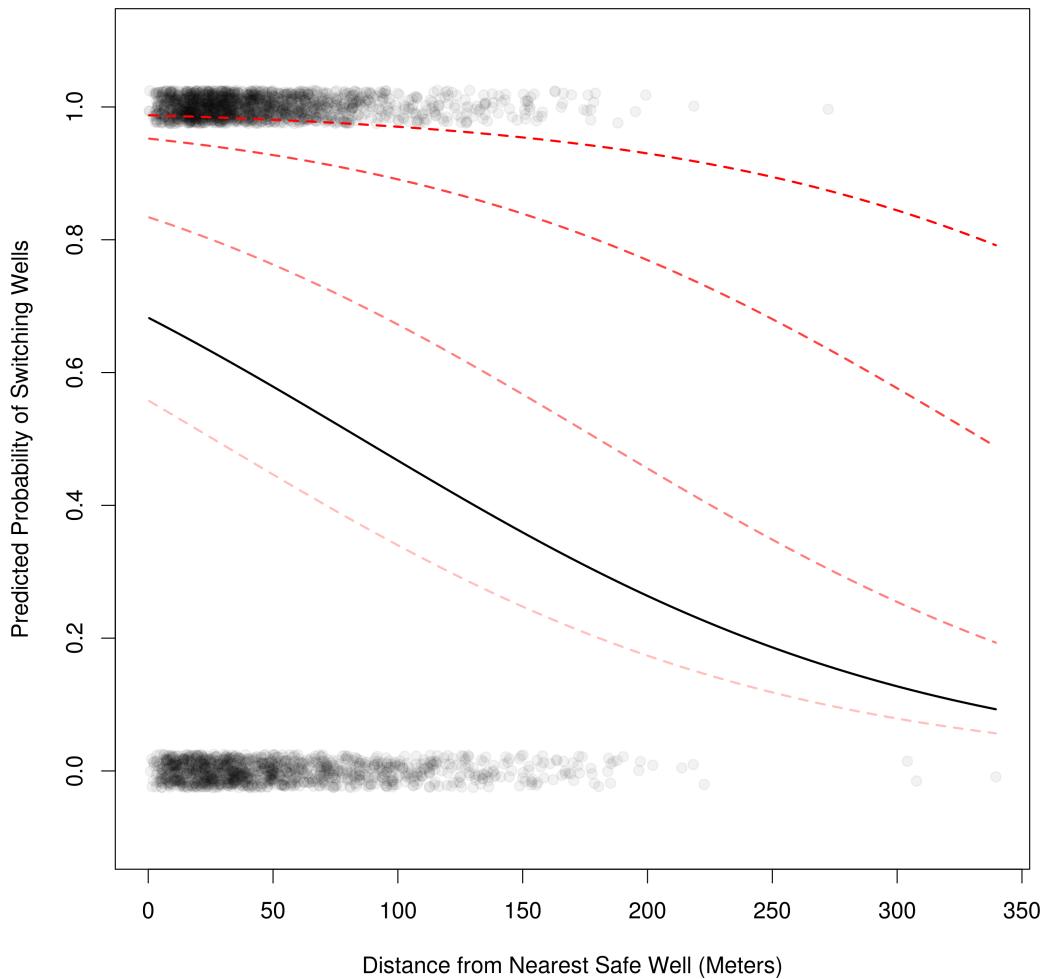


Figure 4.25: Plot of the predicted probability of switching wells from the logistic regression given the distance from the nearest safe well (in meters). The black curve shows the relationship between distance and probability of switching wells at the mean of arsenic levels (1.66 micrograms/L). Dashed red curves show the distance and probability relationship for increasing levels of arsenic (0.5, 3.5, 6.5, and 9.5 micrograms/L), with darker red lines indicating greater levels of arsenic.

It's somewhat misleading to make this sort of plot with evenly spaced arsenic levels across the range of arsenic levels, rather than say, plotting the most frequent arsenic levels. But, one thing we can see is that a change of arsenic levels of 3 units has a different change in the probability of switching depending on the arsenic level and the distance from the nearest safe well. For example, when the distance to the nearest safe well is small, the difference in

probability of well switching between the households with highest arsenic levels in their well (9.5) versus lower arsenic levels (6.5) is small, but larger between those with arsenic levels of 6.5 and 3.5, and even larger between those with arsenic levels of 3.5 and 0.5 (the minimum). This pattern reverses when the distance to the nearest safe well increases. However, because there are so few observations where the distance from the nearest safe well is large, we shouldn't be too sure about drawing any conclusions here. Plotting the other curves (arsenic at different levels of distance) is a suggested exercise.

A better approach might be to plot the probability of switching curves for different quantiles of arsenic, for example using 10 evenly spaced quantiles (i.e., deciles):

```

png(filename = "wells4.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(wells$dist), jitter(wells$switch, amount = .025),
     xlab = "Distance from Nearest Safe Well (Meters)",
     ylab = "Predicted Probability of Switching Wells",
     ylim = c(-.1, 1.1),
     pch = 19,
     col = rgb(0, 0, 0, .05))

#Calculate the mean arsenic levels
mean.arsenic <- mean(wells$arsenic)
# Plot curve at the mean of the arsenic levels
curve(1/(1 + exp(-(coef(wells.glm2)[1] +
                      coef(wells.glm2)[2]*x +
                      coef(wells.glm2)[3]*mean.arsenic))),
      col = rgb(0, 0, 0, 1), add = TRUE, lwd = 2)

# Get the deciles of arsenic levels
arsenic.level <- quantile(wells$arsenic, probs = seq(0, 1, by = .1))
# Plot probability by distance at 10 levels of arsenic
for(i in 1:11){
  # Add the fit from the model for different levels of arsenic
  curve(1/(1 + exp(-(coef(wells.glm2)[1] +
                        coef(wells.glm2)[2]*x +
                        coef(wells.glm2)[3]*arsenic.level[i]))),
        col = rgb(1, 0, 0, 1), add = TRUE, lwd = 2, lty = 2)
}
dev.off()

```

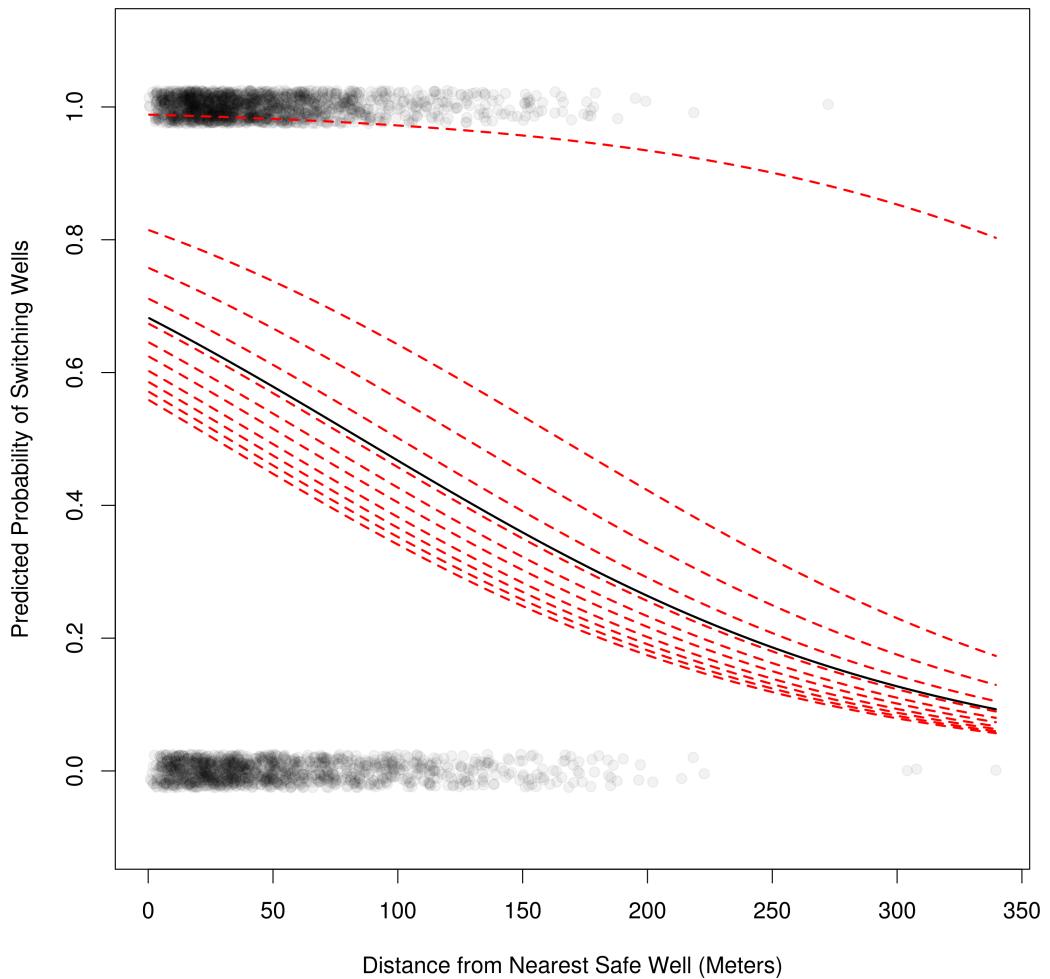


Figure 4.26: Plot of the predicted probability of switching wells from the logistic regression given the distance from the nearest safe well (in meters). Black curve shows the relationship between distance and probability of switching wells at the mean of arsenic levels (1.66 micrograms/L). Dashed red curves show the distance and probability relationship for increasing levels of arsenic, evaluated at the deciles of the arsenic levels.

This figure makes it clear that we have a highly skewed distribution of arsenic levels, and that for the most part the distance between the curves is similar across the different deciles. Lastly, we can include an interaction between distance and arsenic levels. First we should start by centering each variable:

```
wellss$c.dist <- wellss$dist - mean(wellss$dist)
wellss$c.arsenic <- wellss$arsenic - mean(wellss$arsenic)
```

```

wells.glm3 <- glm(switch ~ c.dist + c.arsenic + c.dist*c.arsenic, data = wells,
                    family = binomial(link = "logit"))
summary(wells.glm3)

```

The estimated regression function is:

$$\begin{aligned}
&= \log\left(\frac{P(\text{switch}|\text{c.dist}, \text{c.arsenic})}{1 - P(\text{switch}|\text{c.dist}, \text{c.arsenic})}\right) \\
&= 0.35 - 0.0087 \times \text{c.dist} + 0.47 \times \text{c.arsenic} - 0.002 \times \text{c.dist} \times \text{c.arsenic}
\end{aligned}$$

The intercept indicates a 0.59 probability ($= \frac{1}{1+e^{-0.35}}$) of switching wells when the distance from the nearest safe well and arsenic levels are at their means (48 meters and 1.66 micrograms/L, respectively). The coefficient on the centered distance variable indicates that when arsenic levels are at their mean (i.e., $\text{c.arsenic} = 0$), then the probability of switching wells decreases by 0.0022 points ($= 0.087/4$) for a one meter increase in distance from the nearest safe arsenic well, among households with a 0.5 probability of switching. Similarly, the coefficient on the centered arsenic variable indicates that when distance from the nearest safe well is at its mean (i.e., $\text{c.dist} = 0$), then the probability of switching wells increases by about 0.12 points ($= 0.469/4$) among households with a 0.5 probability of switching.

With the coefficient for the interaction term we can look at the effect in two ways. As distance from the nearest safe well increases by one meter, the coefficient on arsenic decreases by 0.0018, indicating that the importance of arsenic as a predictor of switching wells *decreases* as the distance from the nearest safe well increases. Alternatively, as the arsenic level increases by one unit, the coefficient on distance decreases by 0.0018, indicating that the importance of distance from the nearest safe well as a predictor of switching wells *increases* as arsenic levels increase.

```

# It's easier to plot without the centered variables
wells.glm4 <- glm(switch ~ dist + arsenic + dist*arsenic, data = wells,
                    family = binomial(link = "logit"))
png(filename = "wells5.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(wells$dist), jitter(wells$switch, amount = .025),
      xlab = "Distance from Nearest Safe Well (Meters)",
      ylab = "Predicted Probability of Switching Wells",
      ylim = c(-.1, 1.1),
      pch = 19,
      col = rgb(0, 0, 0, .05))

#Calculate the mean arsenic levels

```

```

mean.arsenic <- mean(wells$arsenic)
# Plot curve at the mean of the arsenic levels
curve(1/(1 + exp(-(coef(wells.glm4)[1] +
                      coef(wells.glm4)[2]*x +
                      coef(wells.glm4)[3]*mean.arsenic +
                      coef(wells.glm4)[4]*mean.arsenic*x))),
      col = rgb(0, 0, 0, 1), add = TRUE, lwd = 2)

# Get the deciles of arsenic levels
arsenic.level <- quantile(wells$arsenic, probs = seq(0, 1, by = .1))

# Plot probability by distance at 10 levels of arsenic
for(i in 1:11){
  # Add the fit from the model for different levels of arsenic
  curve(1/(1 + exp(-(coef(wells.glm4)[1] +
                        coef(wells.glm4)[2]*x +
                        coef(wells.glm4)[3]*arsenic.level[i] +
                        coef(wells.glm4)[4]*arsenic.level[i]*x))),
        col = rgb(1, 0, 0, 1), add = TRUE, lwd = 2, lty = 2)
}
dev.off()

```

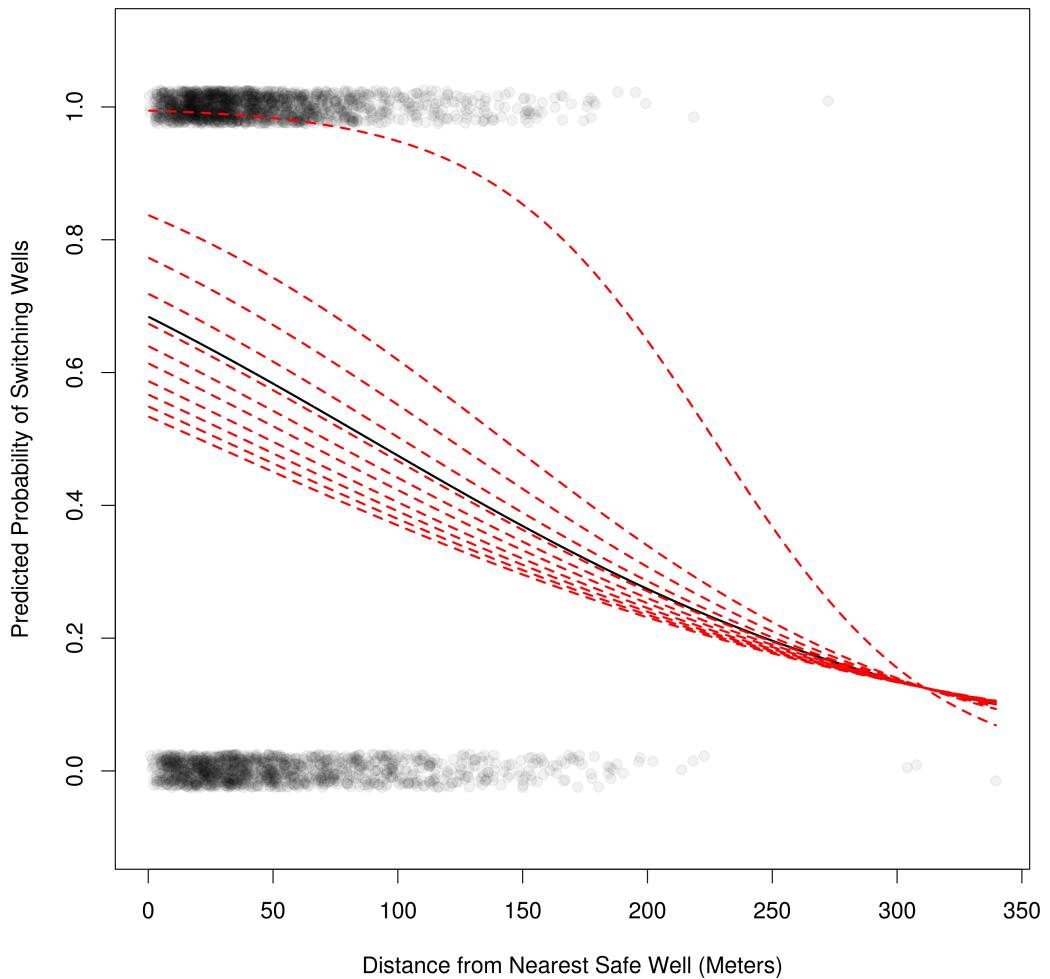


Figure 4.27: Plot of the predicted probability of switching wells from the logistic regression given the distance from the nearest safe well (in meters). Black curve shows the relationship between distance and probability of switching wells at the mean of arsenic levels (1.66 micrograms/L). Dashed red curves show the distance and probability relationship for increasing levels of arsenic, evaluated at the deciles of the arsenic levels, including an interaction between arsenic and distance.

We can see that, for most of the range of arsenic levels, as the distance from the nearest safe well increases, the difference in probability of switching between households with different arsenic levels becomes smaller. That is, arsenic levels matter more to households when it is relatively easy to switch to a safe well, but if switching to a safe well is difficult, the arsenic levels in their unsafe well doesn't matter much.

Finally, let's plot the relationship between well switching and arsenic levels holding several other variables at their means. First, let's include whether the head of household is a member of community organizations and education in the regression:

```
wells.glm5 <- glm(switch ~ dist + arsenic + educ + assoc, data = wells,
                    family = binomial(link = "logit"))
```

Next, let's calculate the mean of the other variables:

```
dist.mean <- mean(wells$dist)
educ.mean <- mean(wells$educ)
assoc.mean <- mean(wells$assoc)
```

Membership in community organizations is a binary (dummy) variable. If it were a factor variable, we would need to first translate it into a dummy variable before calculating the mean:

```
# Hypothetical example where wells$assoc is a factor
# If wells$assoc is equal to "yes", then assoc.mean = 1
# Otherwise, assoc.mean = 0
wells$assoc.dummy <- ifelse(wells$assoc == "yes", 1, 0)
```

We can now make the plot of probability of well switching holding other regressors at their means:

```
png(filename = "wells6.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(wells$arsenic), jitter(wells$switch, amount = .025),
     xlab = "Arsenic Levels (micrograms/L)",
     ylab = "Predicted Probability of Switching Wells",
     ylim = c(-.1, 1.1),
     pch = 19,
     col = rgb(0, 0, 0, .1))

# Plot curve by arsenic at the mean of the other regressors
curve(1/(1 + exp(-(coef(wells.glm5)[1] +
                      coef(wells.glm5)[2]*dist.mean +
                      coef(wells.glm5)[3]*x +
                      coef(wells.glm5)[4]*educ.mean+
                      coef(wells.glm5)[5]*assoc.mean))),
      col = rgb(0, 0, 0, 1), add = TRUE, lwd = 2)

dev.off()
```

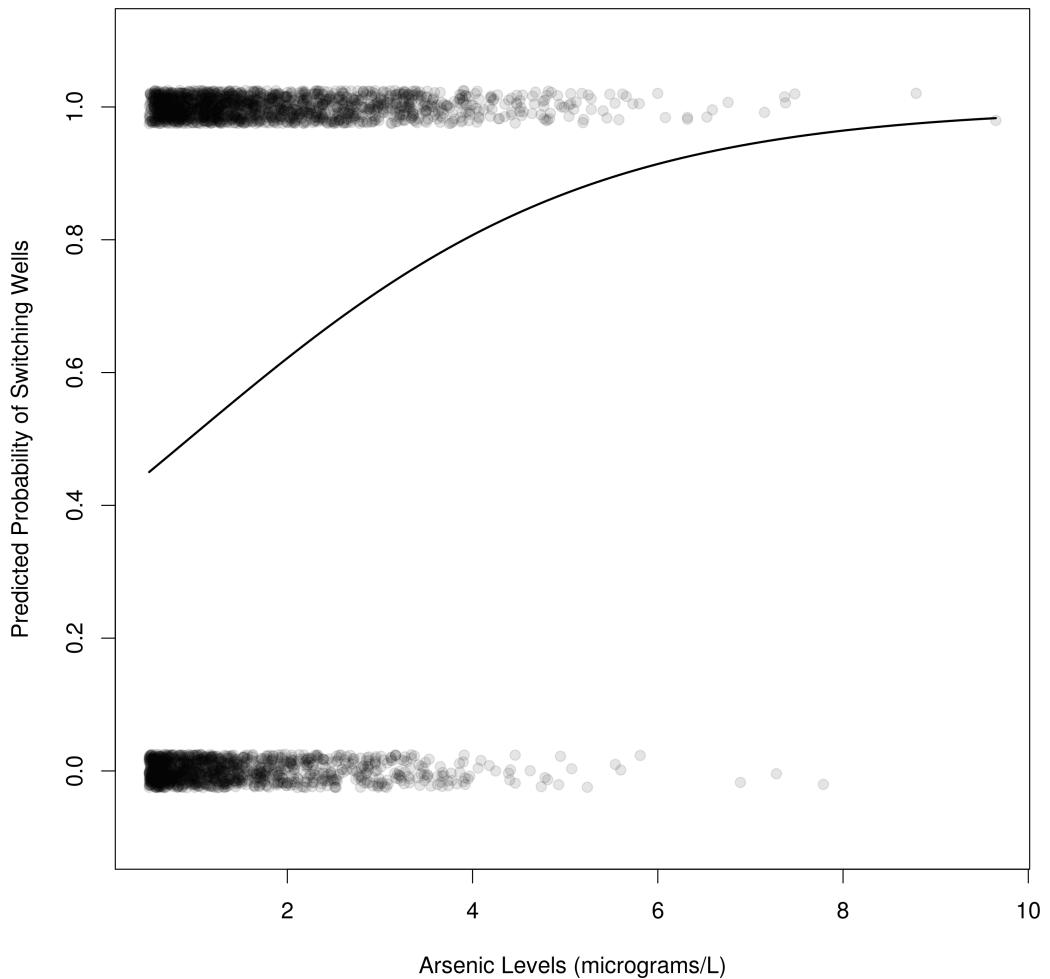


Figure 4.28: Plot of the predicted probability of switching wells from the logistic regression given arsenic levels (micrograms/L), with distance from the nearest safe well, education, and membership in community associations held at their means.

4.5 Model Checking for Logistic Regression

With the generalized linear model there are four ways our model could be misspecified: 1) an omitted variable, 2) an omitted transformation, 3) an omitted interaction, and 4) an incorrect link function. We generally can't do anything about 1) except test models with different variables that we've measured. For 2), we can handle arbitrary transformations using the GAM. On 3), omitting interactions is also a hard problem that we can't do much

about, although some of our goodness-of-fit tests (Calibration Plots, Stukel's test) might pick up a problem if there is one. We'll use calibration plots and Stukel's test to check 4), although these tests also pick up problems with 2) and 3).

4.5.1 Calibration

The simplest test of a logistic regression model is to compare the predicted probabilities of outcomes occurring to their actual frequencies. If these match closely, we say the model is *well calibrated*.

With discrete predictors, we only have a few combinations of regressors, and thus a few different probability predictions. If the model is otherwise correctly specified (i.e., no omitted variables or interactions), then a mismatch in predicted probabilities and actual frequencies indicates an incorrect link function, as it is the link function that connects the linear predictors to the predicted probabilities. However, with continuous predictors it is possible that each observation is assigned a different probability. It is therefore difficult to compare the frequency of occurrence to the predicted probability if the frequencies are only either zero or one.

Calibration Table

One way to overcome this problem is to group the observations based on their predicted probabilities, for example into deciles, then compare the relative frequency of actual outcomes to the average of the predicted probabilities within each decile (Hosmer et al., 2013). The results are summarized in a table showing the key pieces of information from each decile, including the cut point, actual relative frequency and expected frequency based on the model's predictions for each outcome, and the total number of observations in each decile. The first thing we need to do is create a set of cutpoints to group the predicted probabilities by quantiles:

```
# Code by the amazing Sarah Hailey
# get the raw fitted probabilities
wells$probs <- predict(wells.glm3, type = "response")
# Get the deciles of the fitted probabilities
decile.cutpoints <- quantile(wells$probs, probs = seq(0, 1, .1))
# Create a new variable that identifies
# the quantile that each fitted probability falls in
wells$decileID <- cut(wells$probs,
                        breaks = decile.cutpoints,
                        labels = 1:10,
                        include.lowest = TRUE)
```

Next, we need to calculate the number that actually switched wells in each decile:

```
# Calculate the number that switched in each decile
# You can see the counts by using table:
tab <- table(wells$decileID, wells$switch)
# To turn the table into a data.frame, use as.data.frame.matrix
observed <- as.data.frame.matrix(tab)
```

We then calculate the expected number of switches in each decile based on our model's predictions:

```
# To calculate the expected values for each decile,
# we need to sum the fitted probabilities for each decile
# We can do this by using tapply to compute
# the sum of wells$prob within each level of decileID:
expected1 <- tapply(wells$probs, wells$decileID, FUN = sum)
```

Now that we have the observed number of switches in each decile and the predicted number of switches from our model, we need to summarize them in a table:

```
# Create a summary calibration table
# Create cutpoints that represent the 9 intervals that exist for deciles
interval.cutpoints <- round(quantile(wells$probs, probs = seq(.1, 1, .1)), 2)

# Create a dataframe with these cutpoints:
cal <- data.frame(interval.cutpoints)

# Add a column of observed switches
cal$observed1 <- observed[, 2]
# Add a column of expected switches
cal$expected1 <- round(expected1, 0)

# Add columns for observed and expected non-switches:
cal$observed0 <- observed[, 1]
cal$expected0 <- round(302 - expected1, 0)

# Add a column for the total # of observations in each decile
cal$total <- table(wells$decileID)

cal
```

Table 4.1: Calibration Table. Observed (Obs) and estimated expected (Exp) frequencies within each decile of risk for Switch = 1 and Switch = 0 using the fitted logistic regression model.

Decile	Cut Point	Switch = 1		Switch = 0		Total
		Obs	Exp	Obs	Exp	
1	.45	107	117	195	185	302
2	.49	135	142	167	160	302
3	.51	147	151	155	151	302
4	.53	145	157	157	145	302
5	.55	178	163	124	139	302
6	.58	178	171	124	131	302
7	.62	195	180	107	122	302
8	.67	192	194	110	108	302
9	.74	219	212	83	89	302
10	.98	241	247	61	55	302

The resulting table should be examined for differences between the model's predicted frequencies and the frequencies of the actual outcome (or non-outcome), as well as examined for any systematic over or underprediction as a function of decile. It is also important to check that the total number of observations in each decile are equivalent, and if they are not, check how ties are being resolved.

Calibration Plot

We call a plot of the actual versus predicted frequencies a *calibration plot*. The calibration plot can help visualize any systematic deviations that exhibit a pattern as a function of the predicted probability decile. For example, modeled probabilities may be too high for the first decile, and too low for the last.

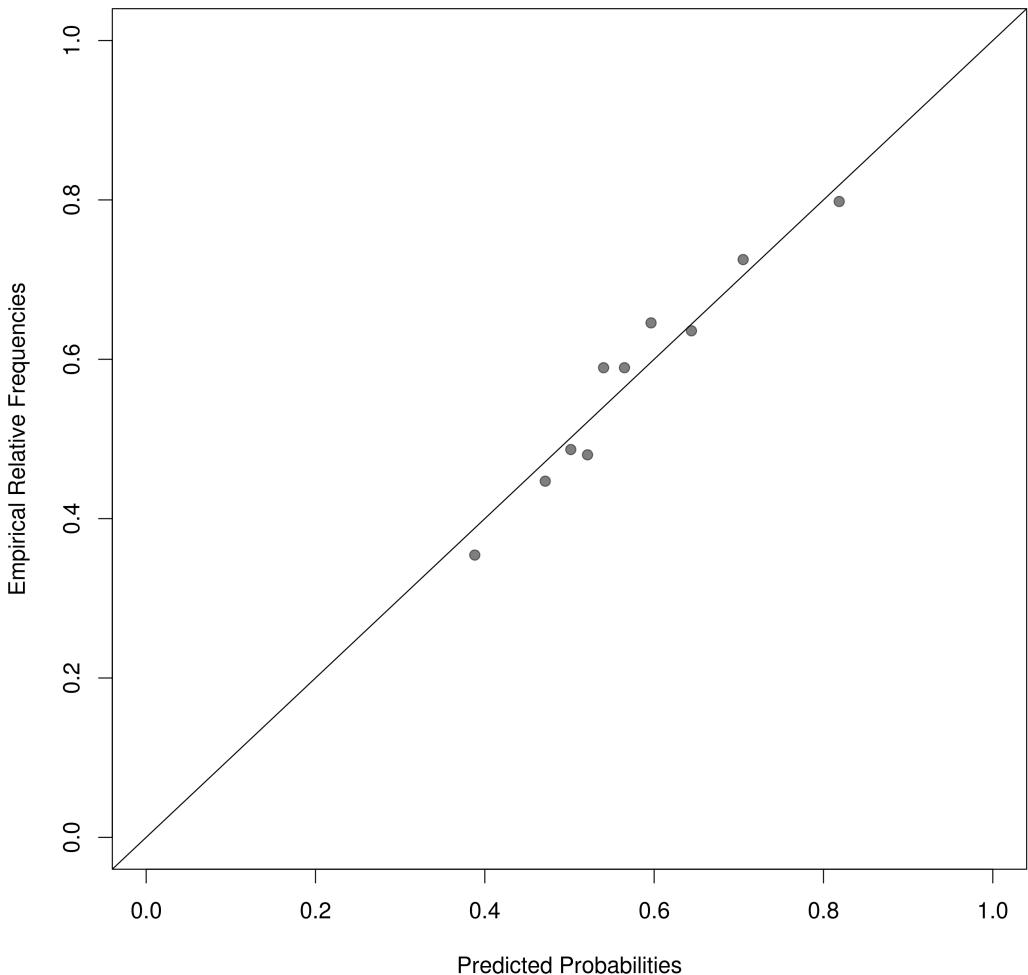


Figure 4.29: Calibration plot of empirical relative frequencies of switching and predicted probability of switching for 10 deciles of predicted probabilities.

In this calibration plot, there doesn't seem to be any systematic over- or under-prediction of relative frequencies.

4.5.2 Alternative Link Functions

Another way to check the logistic regression model was proposed by [Stukel \(1988\)](#) with the hope of testing the link function (although it also tends to pick up omitted transformations and interactions). The power of this test is modest, even though it is more powerful than any other method for detecting an incorrect link function ([Hosmer et al., 1997](#)). Note that a similar test was proposed by [Ramsey \(1969\)](#) for ordinary least squares regression. The

proposed test compares the logistic function against the following mean function:

$$\mu_\alpha(\eta) = \frac{e^{h_\alpha(\eta)}}{1 + e^{h_\alpha(\eta)}} \quad (4.25)$$

where for $\eta \geq 0$:

$$h_\alpha = \begin{cases} \alpha_1^{-1}(e^{\alpha_1|\eta|} - 1) & \text{if } \alpha_1 > 0 \\ \eta & \text{if } \alpha_1 = 0 \\ -\alpha_1^{-1} \log(1 - \alpha_1|\eta|) & \text{if } \alpha_1 < 0 \end{cases}$$

and for $\eta < 0$:

$$h_\alpha = \begin{cases} -\alpha_2^{-1}(e^{\alpha_2|\eta|} - 1) & \text{if } \alpha_2 > 0 \\ \eta & \text{if } \alpha_2 = 0 \\ \alpha_2^{-1} \log(1 - \alpha_2|\eta|) & \text{if } \alpha_2 < 0 \end{cases}$$

When $\alpha_1 = \alpha_2 = 0$ then we get the standard logistic model. If $\alpha_1, \alpha_2 > 0$ then h is exponential and the curve rises more quickly than the logistic, with slimmer tails. If $\alpha_1, \alpha_2 < 0$ then h is logarithmic and the curve rises more slowly than the logistic, with fatter tails. However, α_1 and α_2 independently govern the rate of change of Stukel's curve above and below $\eta = 0$. This model generalizes a number of important alternatives to the logistic function. Specifically, the mean function is:

$$\mu_\alpha(\eta) \approx \begin{cases} \text{Logistic} & \text{if } \alpha_1 = \alpha_2 = 0 \\ \text{Gaussian (Probit)} & \text{if } \alpha_1 = \alpha_2 = .165 \\ \text{Log-Log} & \text{if } \alpha_1 = -.037, \alpha_2 = .62 \\ \text{Complementary Log-Log} & \text{if } \alpha_1 = .62, \alpha_2 = -.037 \\ \text{Standard Laplace} & \text{if } \alpha_1 = \alpha_2 = -.077 \end{cases}$$

Probit

A prominent alternative to the logistic function to model $P(Y = 1|x)$ is the cumulative standard normal distribution:

$$P(Y = 1|x) = \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-t^2/2} dt \quad (4.26)$$

We can compare the shape of the normal cdf, logistic function, and Stukel's approximation to the normal cdf (using $\alpha_1 = \alpha_2 = 0.165$):

```
# Use a sequence of XB points
xb <- seq(from = -3, to = 3, by = .5)

# Use standard normal cdf
```

```

probits <- pnorm(xb)

# Use Stukel's approximation
stukel.probit <- function(xb){
  h <- ifelse(xb >= 0,
    (exp(0.165*abs(xb)) - 1)/0.165,
    (exp(0.165*abs(xb)) - 1)/-0.165)
  # Note we need to scale by 1.6 to make the standard logit
  # have unit variance
  exp(1.6*h)/(1 + exp(1.6*h))
}

# Calculate probabilities
stukel.probs <- stukel.probit(xb)
probits <- pnorm(xb)
logis <- plogis(1.6*xb)

png(filename = "probitstukel.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(xb, logis, type = "b",
      xlim = c(-3, 3),
      ylim = c(0, 1),
      col = "red",
      ylab = "P(Y = 1|XB)",
      xlab = "XB",
      lwd = 2)
lines(xb, probits, col = "blue", lty = 1, lwd = 2)
lines(xb, stukel.probs, col = "blue", lty = 2, lwd = 2)
dev.off()

```

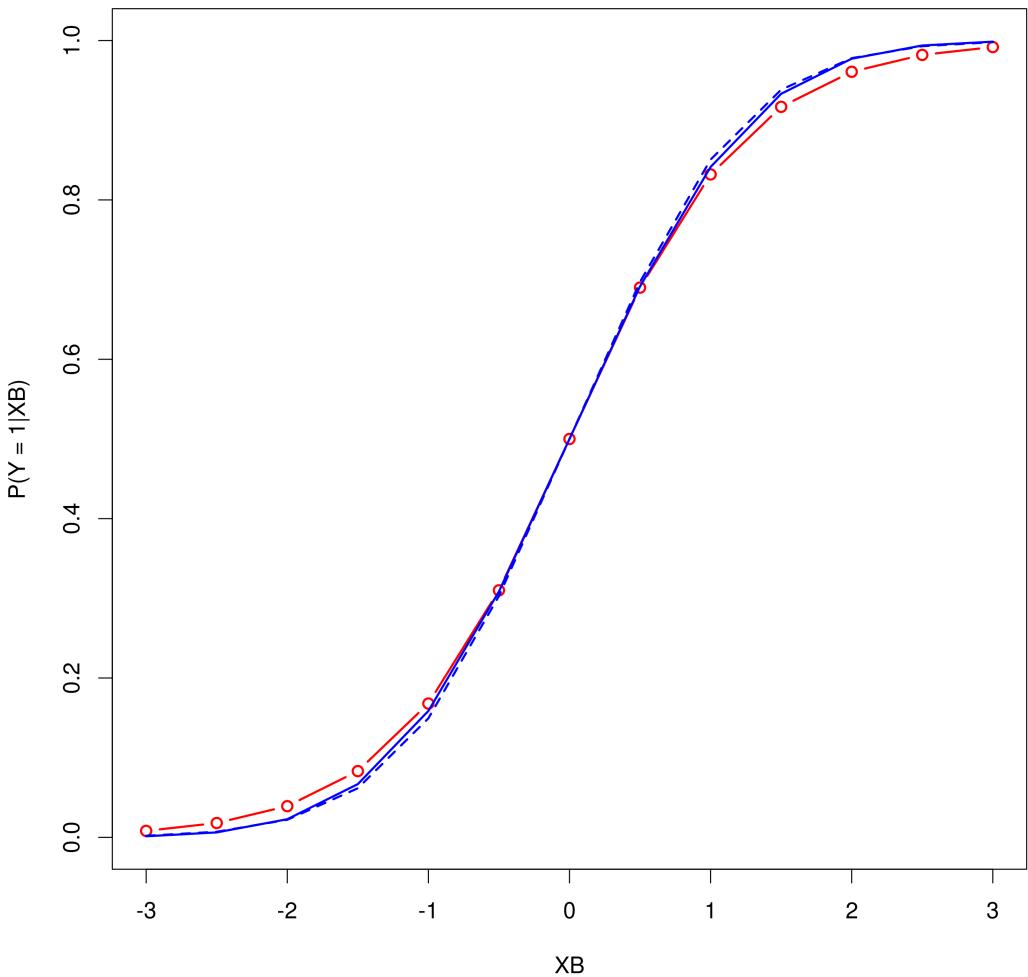


Figure 4.30: Logistic (red), probit (blue), and Stukel's approximation to the probit (dashed blue) ($\alpha_1 = \alpha_2 = 0.165$).

As can be seen, the logistic function and normal cumulative distribution are almost identical, meaning that a lot of data would be required to tell the two apart. The logistic function does have somewhat noticeably more density in the tails (heavier tails) than the normal. Stukel's approximation works very well.

Complementary Log Log

The complementary log-log function is an example of an alternative to the logistic function that is asymmetric above and below $X\beta = 0$.

$$P(Y = 1|x) = \text{clog-log}(X\beta) = 1 - e^{-e^{X\beta}} \quad (4.27)$$

where the link function is:

$$g(\eta(x)) = \log(-\log(1 - p(x))) = X\beta \quad (4.28)$$

It has a much faster rate of change for $X\beta > 0$ than $X\beta < 0$, giving it a short right tail and a left tail that is very close to the logistic function. Again, we can compare the shape of the complementary log-log cdf, logistic function, and Stukel's approximation to the complementary log-log (using $\alpha_1 = .62, \alpha_2 = -.037$):

```
# Use a sequence of XB points
xb <- seq(from = -3, to = 3, by = .5)

# Centered and scaled complementary log-log
# To have mean 0 variance 1
cloglog <- 1-exp(-exp(xb*pi/sqrt(6) - 0.577))

# Stukel's approximation to the clog-log
stukel.cloglog <- function(xb){
  h <- ifelse(xb >= 0,
    (exp(0.62*abs(xb)) - 1)/0.62,
    (log(1 + .037*abs(xb))/-.037))

  exp(h*1.6)/(1 + exp(h*1.6))
}

# Calculate probabilities
stukel.probs <- stukel.cloglog(xb)
clogs <- cloglog
logis <- exp(xb*1.6)/(1 + exp(xb*1.6))

png(filename = "clogstukel.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(xb, logis, type = "b",
      xlim = c(-3, 3),
      ylim = c(0, 1),
      col = "red",
      ylab = "P(Y = 1|XB)",
      xlab = "XB",
      lwd = 2)
lines(xb, clogs, col = "blue", lty = 1, lwd = 2)
lines(xb, stukel.probs, col = "blue", lty = 2, lwd = 2)
dev.off()
```

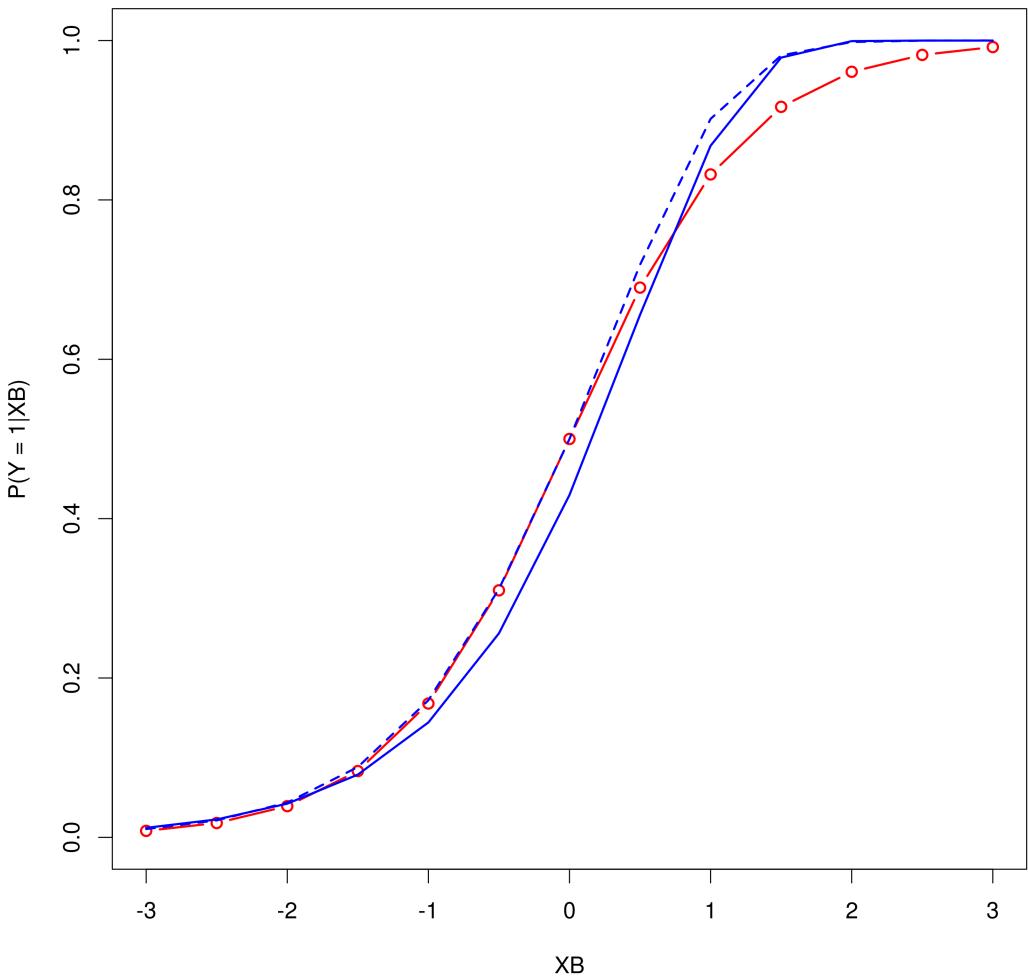


Figure 4.31: Logistic (red), complementary log-log (blue), and Stukel's approximation to the complementary log-log (dashed blue) with $\alpha_1 = .62$, $\alpha_2 = -.037$.

As can be seen, the complementary log-log function rises more quickly than the logistic for $X\beta > 0$ with a short right tail and long left tail. However, Stukel's approximation only works well when $X\beta > 0$.

Stukel's Test

If we can estimate α_1 and α_2 , we can not only test the logistic function, but also determine the source of misspecification. We can test whether such a model is more appropriate with the following approach. We can estimate a first-order approximation to Stukel's model using the following specification of

the logit model:

$$\text{logit}(p) = X\beta + \frac{1}{2}\alpha_1\hat{\eta}^2 I_{\eta>0} - \frac{1}{2}\alpha_2\hat{\eta}^2 I_{\eta<0} \quad (4.29)$$

where $\hat{\eta}$ is the fitted values (on the logit scale) from the original regression, and $I_{\eta>0}$ is an indicator variable that is 1 if $\eta > 0$ and 0 otherwise (and vice versa for $I_{\eta<0}$). To conduct this test, first we calculate $\hat{\eta}$ from the model:

```
wells.glm3 <- glm(switch ~ dist + arsenic + dist:arsenic,
                    family = binomial(link = "logit"),
                    data = wells)
eta.hat <- predict(wells.glm3)
```

Then create an indicator if $\hat{\eta} \geq 0$

```
positive <- ifelse(eta.hat >= 0, 1, 0)
negative <- ifelse(eta.hat < 0, 1, 0)
```

Then we square $\hat{\eta}$:

```
eta.hat.sq <- eta.hat^2
```

Then we run the model:

```
stukel1 <- glm(switch ~ dist + arsenic + dist:c.arsenic +
                 eta.hat.sq:positive + eta.hat.sq:negative,
                 data = wells,
                 family = binomial(link = "logit"))
```

Keep in mind that we want to estimate:

$$\text{logit}(p) = X\beta + \frac{1}{2}\hat{\alpha}_1\hat{\eta}^2 I_{\eta>0} - \frac{1}{2}\hat{\alpha}_2\hat{\eta}^2 I_{\eta<0} \quad (4.30)$$

from:

$$\text{logit}(p) = X\beta + \hat{\beta}_1\hat{\eta}^2 I_{\eta>0} + \hat{\beta}_2\hat{\eta}^2 I_{\eta<0} \quad (4.31)$$

Thus, $\hat{\alpha}_1 = 2\hat{\beta}_1$ and $\hat{\alpha}_2 = -2\hat{\beta}_2$. Here, we find $\hat{\beta}_1 = -0.38$ and $\hat{\beta}_2 = 0.19$, giving us $\hat{\alpha}_1 = -0.76$ and $\hat{\alpha}_2 = -0.38$. We might consider a function that rises and tapers more slowly than the logistic, with longer tails on both sides. Before we try a different link function, we should first check for non-linear transformations of the regressors (covered in the next section). Here's another example of testing the logistic function against alternatives using simulation:

```

set.seed(500)
x1 <- runif(10000, -2, 2)
x2 <- runif(10000, -3, 3)
x <- cbind(rep(1, length(x1)), x1, x2)
beta0 <- .5
beta1 <- 2
beta2 <- 1
# Create linear predictor xbeta
xb <- beta0 + beta1*x1 + beta2*x2

# Use complementary log-log
p <- 1-exp(-exp(xb))
# Generate coin flips where p(heads) = logistic(xb)
y <- rbinom(10000, 1, p)

# Try different link functions
glm1 <- glm(y ~ x1 + x2, family = binomial(link = "logit"))
glm2 <- glm(y ~ x1 + x2, family = binomial(link = "probit"))
glm3 <- glm(y ~ x1 + x2, family = binomial(link = "cloglog"))

eta.hat <- predict(glm1)
positive <- ifelse(eta.hat >= 0, 1, 0)
negative <- ifelse(eta.hat < 0, 1, 0)
eta.hat.sq <- eta.hat^2
stukel1 <- glm(y ~ x1 + x2 +
                 eta.hat.sq:positive + eta.hat.sq:negative,
                 family = binomial(link = "logit"))

```

Again, $\hat{\alpha}_1 = 2\hat{\beta}_1$ and $\hat{\alpha}_2 = -2\hat{\beta}_2$. Here, we find $\hat{\beta}_1 = 0.25$ and $\hat{\beta}_2 = 0.03$, giving us $\hat{\alpha}_1 = 0.50$ and $\hat{\alpha}_2 = -0.06$. The positive α_1 and negative α_2 matches the complementary log-log parameters proposed by Stukel ($\alpha_1 = 0.62, \alpha_2 = -0.037$) closely. We can verify this from our risk decile table, comparing the logistic to the complementary log-log. Also notice that the regression with the complementary log-log link has the lowest residual deviance, where residual deviance is a badness-of-fit measure similar to the rMSE in linear regression:

$$D = -2 \times \log(L(\hat{\beta})) \quad (4.32)$$

However, also notice that even with 10,000 observations the parameter β_2 is barely significant. This demonstrates a fundamental problem with diagnostic tests for logistic regression, where these tests are not very sensitive to model violations.

4.5.3 Partial Residual Plots and Non-Linear Transformations

Because Stukel's test does not unambiguously detect just link function misspecification, it makes sense to first look for non-linearity using the generalized additive model. Specifically, we look at the partial residual plot to figure out if the regressors need to be transformed, then re-run Stukel's test to see if this transformation removes any model misspecification.

Recall the residuals are the deviation between the actual y values and the model's predicted or fitted values \hat{y} . Alternatively, \hat{y} can be thought of as the fitted conditional mean, which in the case of binary outcomes, is the fitted conditional probability $\hat{p}(x)$. Thus, the residuals are:

$$y_i - \hat{y}_i = y_i - \hat{p}(x_i) \quad (4.33)$$

So, the residuals are the difference between the actual outcome y_i (1 or 0) and the predicted probability that $y_i = 1$ (or $\hat{p}(x_i)$). In this simple case, the residual is just $1 - \hat{p}(x_i)$ when $y_i = 1$ and $-\hat{p}(x_i)$ when $y_i = 0$. If we add the component to the residual, we get a component plus residual plot, like in linear regression (Landwehr et al., 1984):

```
wells.glm3 <- glm(switch ~ dist + arsenic + dist:arsenic, data = wells,
                     family = binomial(link = "logit"))
# Residual is the binary outcome minus the predicted probability
resids <- wells$switch - predict(wells.glm3, type = "response")
# Component is just xB
component <- coef(wells.glm3)[3]*wells$arsenic
# Component plus residual
cpr <- resids + component
# Component-plus-residual plot
png(filename = "wellscpr1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
# use local polynomial smooth
scatter.smooth(wells$arsenic, cpr,
               ylab = "Component plus residual",
               xlab = "Arsenic levels",
               pch = 19,
               col = rgb(0, 0, 0, .3))
dev.off()
```

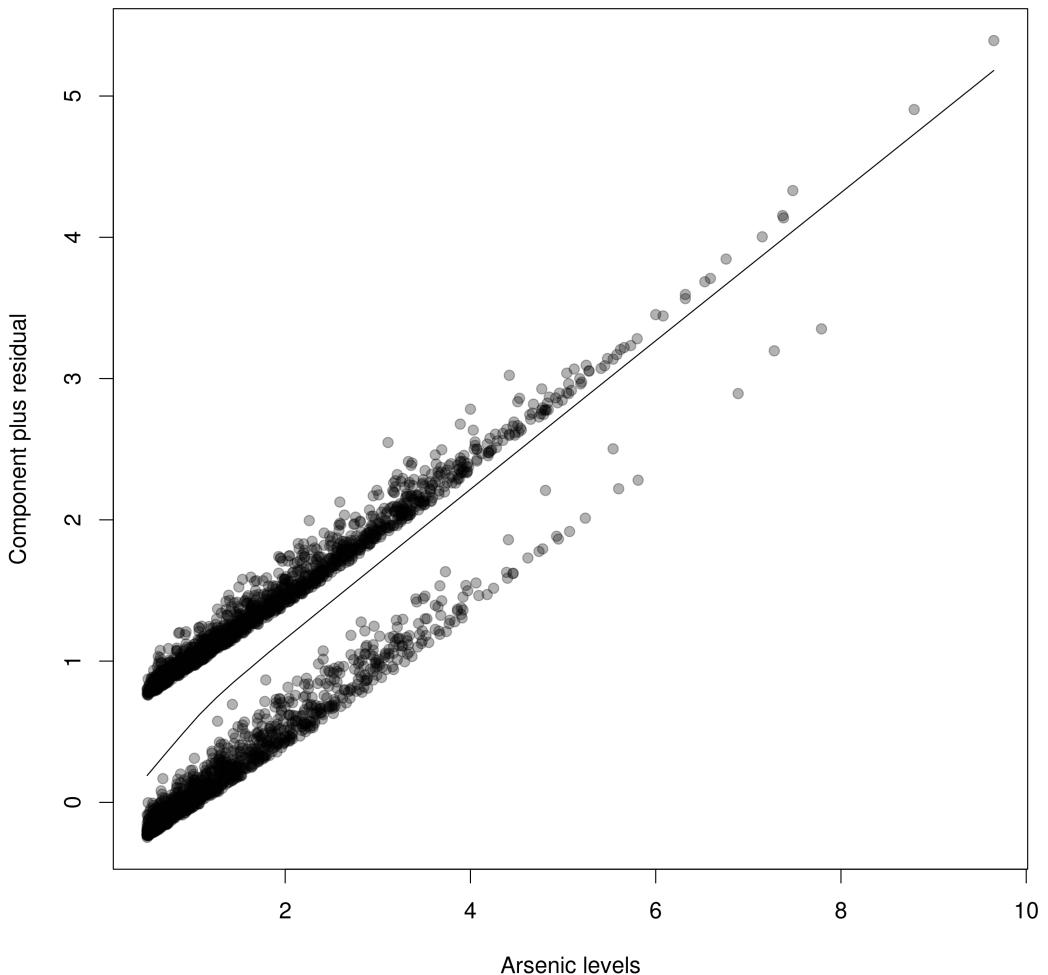


Figure 4.32: Component plus residual plots for arsenic levels from the logistic regression model predicting decision to switch wells.

As can be seen, the plot has the strange feature of having two separate groups of data, one above and one below the mean. This is what we should expect, as the residual plus component is:

$$y_i - \hat{p}(x_i) + \hat{\beta}X_i = \begin{cases} 1 - \hat{p}(x_i) + \hat{\beta}X_i & \text{if } y_i = 1 \\ -\hat{p}(x_i) + \hat{\beta}X_i & \text{if } y_i = 0 \end{cases}$$

which will differ by 1 for observations with the same fitted value ($\hat{p}(x_i)$) and x value. However, keep in mind that in logistic regression, the errors depend on the fitted values such that $\text{Var}(Y|x) = p(x)(1 - p(x))$, so we need to take that into account by normalizing our residuals by $\text{Var}(Y|x)$. Residuals

normalized in this way are called the *Pearson Residuals*.

$$e_{pi} = \frac{y_i - \hat{p}(x)_i}{\sqrt{\hat{p}(x)_i(1 - \hat{p}(x)_i)}} \quad (4.34)$$

We can then create the component plus residual plot using the Pearson residuals:

```
wells.glm3 <- glm(switch ~ dist + arsenic + dist:arsenic, data = wells,
                    family = binomial(link = "logit"))
# Residual is the binary outcome minus the predicted probability
prob <- predict(wells.glm3, type = "response")
resids <- (wells$switch - prob)/sqrt((prob*(1-prob)))
# Component is just xB
component <- coef(wells.glm3)[3]*wells$arsenic
# Component plus residual
cpr <- resids + component
# Component-plus-residual plot
png(filename = "wellscpr2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
# use local polynomial smooth
scatter.smooth(wells$arsenic, cpr,
                ylab = "Component plus residual",
                xlab = "Arsenic levels",
                pch = 19,
                col = rgb(0, 0, 0, .3))
dev.off()
```

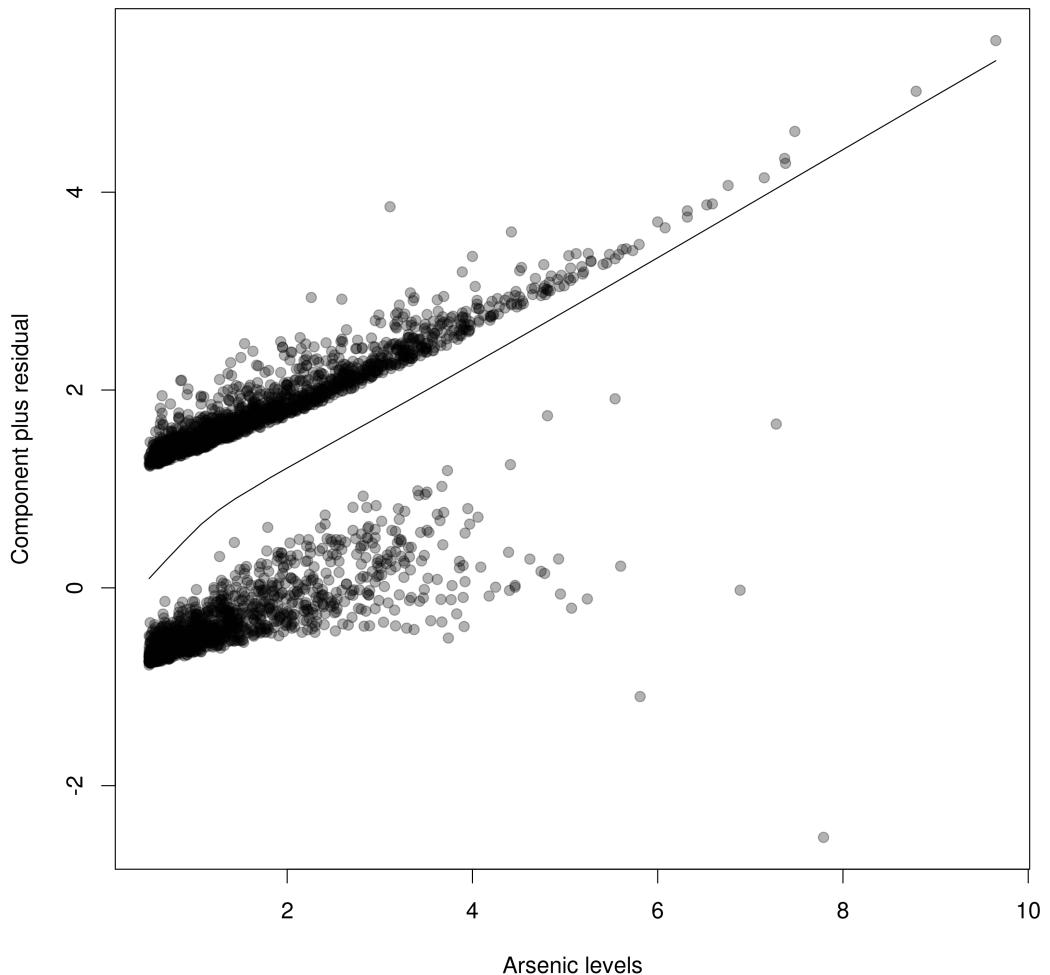


Figure 4.33: Component plus residual plot with Pearson residuals for arsenic levels from the logistic regression model predicting decision to switch wells.

As in the case of linear regression, the component plus residual plot is only valid if all the terms in the model are correctly specified. We can use the generalized additive model to provide a partial residual plot that allows arbitrary flexibility for each of the regressors:

```
library(mgcv)
# Check for any transformations to the regressors
gam.wells <- gam(switch ~ s(dist) + s(arsenic) + dist:arsenic, data = wells,
                  family = binomial(link = "logit"))
png(filename = "wellsgam.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1), mfrow = c(1, 2))
plot(gam.wells,
```

```
shade = TRUE,  
residuals = TRUE,  
scale = 0,  
cex = 1,  
pch = 19,  
col = rgb(0, 0, 0, .4))  
dev.off()
```

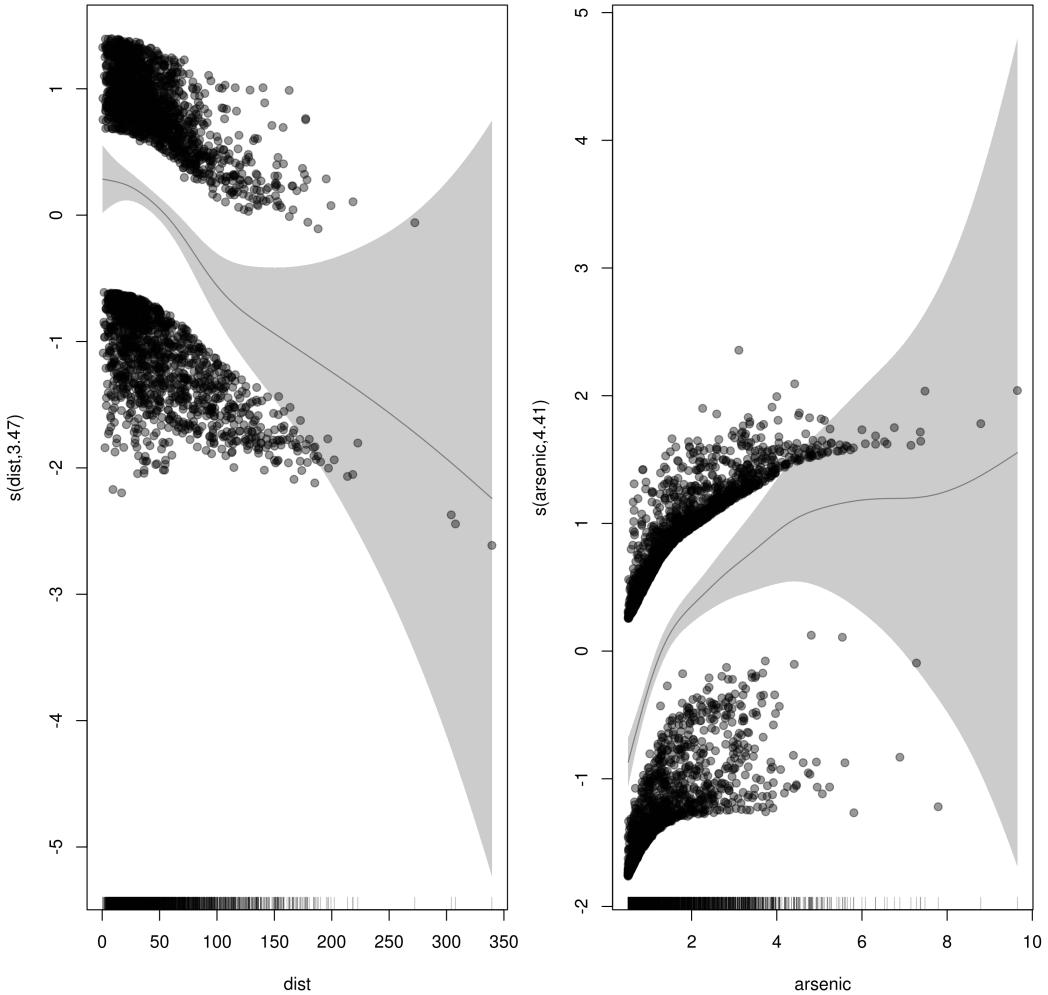


Figure 4.34: Partial residual plots for distance from the nearest safe well and arsenic levels from the generalized additive model predicting decision to switch wells.

The plots here show the partial residual, which is the pearson residual plus the smoothed component. As can be seen it looks like the log transform would be better for arsenic. Let's try that:

```

wells$log.arsenic <- log(wells$arsenic)
gam.wells2 <- gam(switch ~ s(dist) + s(log.arsenic) + dist:log.arsenic,
                   data = wells,
                   family = binomial(link = "logit"))
png(filename = "wellsgam2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1), mfrow = c(1, 2))
plot(gam.wells2,

```

```
shade = TRUE,  
residuals = TRUE,  
scale = 0,  
cex = 1,  
pch = 19,  
col = rgb(0, 0, 0, .4))  
dev.off()
```

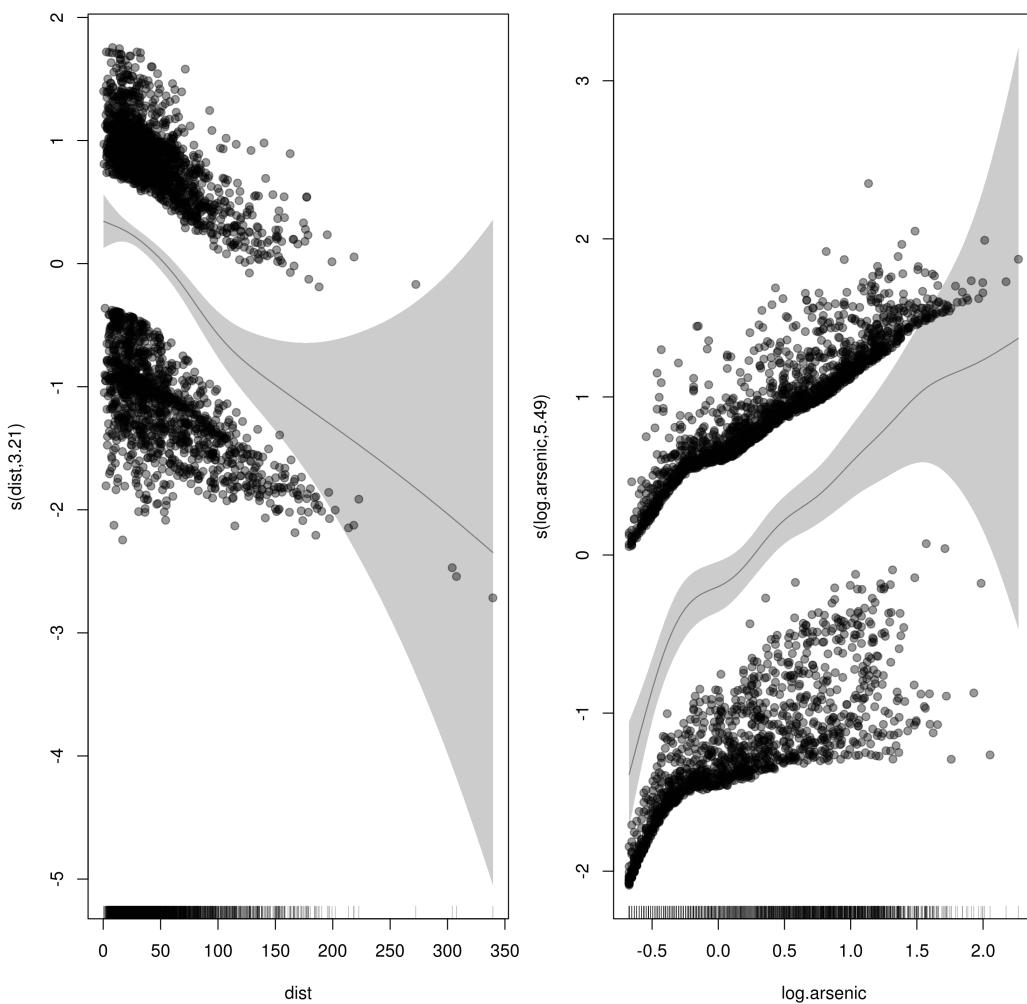


Figure 4.35: Partial residual plots for distance from the nearest safe well and arsenic levels from the generalized additive model predicting decision to switch wells.

Then redo Stukel's test with a log transform for arsenic:

```
wellss.log <- glm(switch ~ dist + log.arsenic + dist:log.arsenic,
                    data = wellss,
                    family = binomial(link = "logit"))

eta.hat <- predict(wellss.log)
positive <- ifelse(eta.hat >= 0, 1, 0)
negative <- ifelse(eta.hat < 0, 1, 0)
eta.hat.sq <- eta.hat^2
```

```

stukel2 <- glm(switch ~ dist + log.arsenic + dist:log.arsenic
+ eta.hat.sq:positive + eta.hat.sq:negative,
data = wells,
family = binomial(link = "logit"))

```

Stukel's test is much better after taking the log, but there still seems to be an issue with α_1 . We can test whether this is due to non-linearity again by trying another gam with smoothing on distance and log-arsenic:

```

stukel3 <- gam(switch ~ s(dist) + s(log.arsenic) + dist:log.arsenic
+ eta.hat.sq:positive + eta.hat.sq:negative,
data = wells,
family = binomial(link = "logit"))

```

```

png(filename = "wellsgam3.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1), mfrow = c(1, 2))
plot(stukel3,
      shade = TRUE,
      residuals = TRUE,
      scale = 0,
      cex = 1,
      pch = 19,
      col = rgb(0, 0, 0, .4))
dev.off()

```

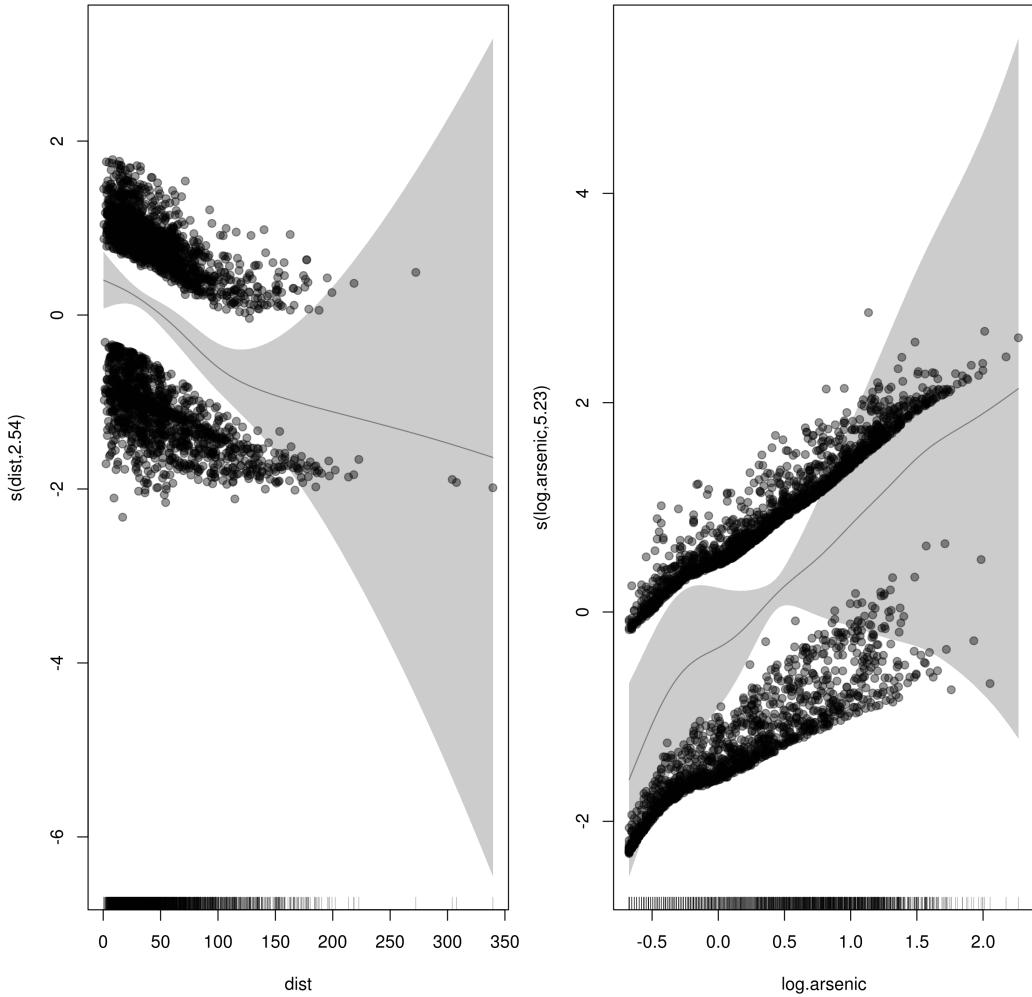


Figure 4.36: Partial residual plots for distance from the nearest safe well and arsenic levels from the generalized additive model predicting decision to switch wells.

Looks like we might need another term on log arsenic, let's try squaring it:

```

wells$log.arsenic.sq <- wells$log.arsenic^2
wells.log.sq <- glm(switch ~ dist + log.arsenic + log.arsenic.sq,
                     data = wells,
                     family = binomial(link = "logit"))

eta.hat <- predict(wells.log.sq)
positive <- ifelse(eta.hat >= 0, 1, 0)
negative <- ifelse(eta.hat < 0, 1, 0)
eta.hat.sq <- eta.hat^2

```

```

stukel4 <- glm(switch ~ dist + log.arsenic + log.arsenic.sq +
    + eta.hat.sq:positive + eta.hat.sq:negative,
    data = wells,
    family = binomial(link = "logit"))

```

After adding the square of the log the model passes Stukel's test, with a much smaller value of α_1 . This demonstrates that while Stukel's test can pick up problems with the link function, it is important to first check for non-linear transformations to the regressors. Before moving on, we can check alternative link functions:

```

wells.sq.logit <- glm(switch ~ dist + log.arsenic + log.arsenic.sq,
    data = wells,
    family = binomial(link = "logit"))
wells.sq.probit <- glm(switch ~ dist + log.arsenic + log.arsenic.sq,
    data = wells,
    family = binomial(link = "probit"))
wells.sq.clog <- glm(switch ~ dist + log.arsenic + log.arsenic.sq,
    data = wells,
    family = binomial(link = "cloglog"))

```

The parameter estimates are similar across models, and the logit has the lowest deviance, indicating we are unlikely to do much better with the complementary log-log. Note that we've used a very complex model, including the square of the log of arsenic as a predictor. While this works, it is not very elegant. It may be possible to include another predictor that is able to account for this curvature better, rather than continuing transformations of arsenic.

4.5.4 Complete and Quasi-Complete Separation

As previously mentioned, as $\hat{\beta}$ goes to ∞ or $-\infty$, the ability of x to discriminate between the classes becomes perfect. If this is the case the logistic regression algorithm (might) fail. This is called *complete* or *quasi-complete separation* (Albert and Anderson, 1984), and means either that you've got a really awesome model (congratulations!) or that you've done something really wrong. Either way you need to look at your regression output to see what's going on. Sometimes you might notice that R spits out the following warning after fitting logistic regression:

glm.fit: fitted probabilities numerically 0 or 1 occurred

R might also not spit out the warning, and instead you will have parameter estimates with very large standard errors. Both should tip you off that there

is something wrong with the model. Complete separation results from the following scenario ([Agresti, 2013](#)): Suppose that the results of the following table have been observed:

X	10	20	30	40	60	70	80
Y	0	0	0	0	1	1	1

As can be seen, $P(Y = 1|x < 50) = 0$, and $P(Y = 1|x > 50) = 1$. In this case, we can let $\hat{\beta} \rightarrow \infty$ and our likelihood function will increase indefinitely:

```
x <- runif(100, 10, 80)

# p is either 0 or 1 depending on x
p <- ifelse(x < 50, 0 , 1)

y <- rbinom(100, 1, p)
glm1 <- glm(y ~ x, family = binomial(link = "logit"))
```

As you can see, `glm` spits out the error and you get a large standard error for the estimate. The numerical approximation algorithm will iterate until the log-likelihood looks flat. Because the variances are from the inverse of a matrix of second derivatives, the standard errors will be large ([Agresti, 2013](#)). Complete or quasi-complete separation can occur in multiple regressors if a linear combination of them is associated with only one value of the dependent variable. [Kleiber and Zeileis \(2008\)](#) section 5.2 provide an example and debugging.

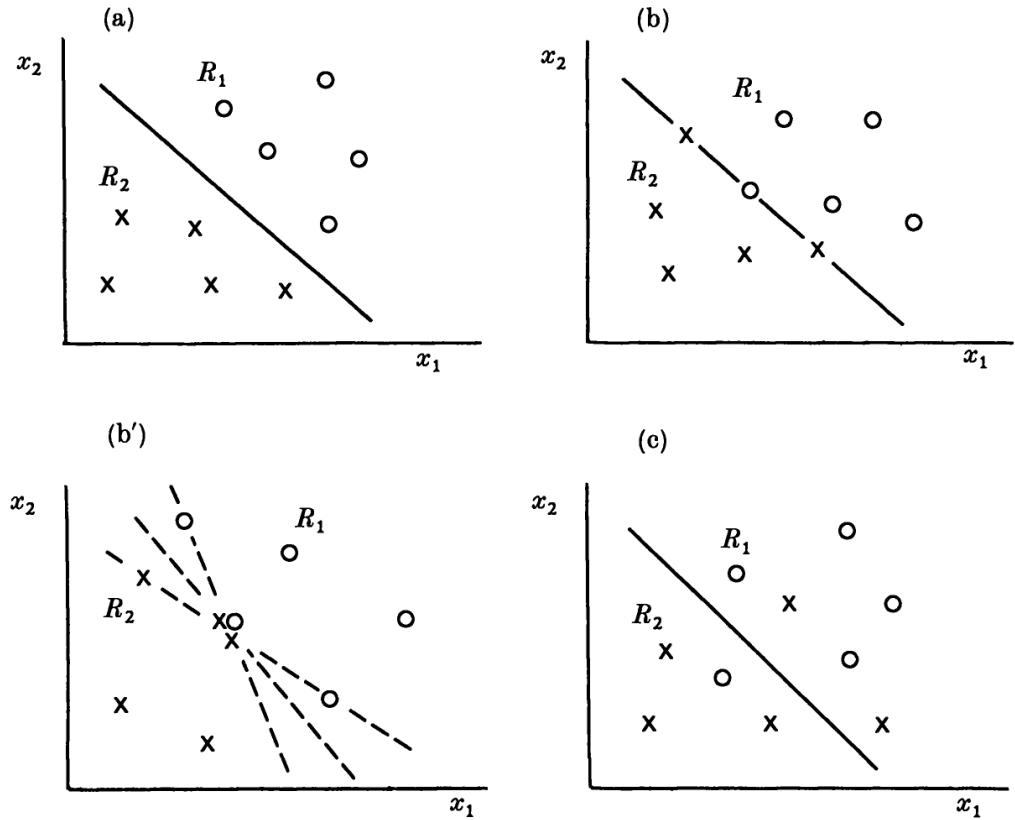


Figure 4.37: (a) Complete separation, (b) quasi-complete separation, (b') quasi-complete separation, (c) overlap. Figure taken from Albert, A., & Anderson, J. A. (1984). On the existence of maximum likelihood estimates in logistic regression models. *Biometrika*, 71(1), 1-10.

4.6 The Forecasting Story

After developing different models of our conditional means (conditional probabilities), we can take a look at how sensible their forecasts are compared to the actual data. Let's go back to our simple original data and obtain the predicted probability of voting republican given different levels of conservativism:

```
votes$prob <- predict(voteglm, type = "resp")
```

We can then set a cutoff point such that if the predicted probability is greater than the cutoff, we predict the person will vote republican, and democrat otherwise. A natural cutoff is predicted probability of 0.5:

```
# Using a cutoff point of p > 0.5
```

```
# Predict 1 (republican) if p > 0.5
# Predict 0 (democrat) otherwise
votes$cut.5 <- ifelse(votes$prob > .5, 1, 0)
```

4.6.1 Confusion Matrix

Now, let's look at the predicted voting behavior vs. actual voting behavior in a table. Such a table is called a *confusion matrix*:

```
table(votes$cut.5, votes$voterel)
```

From the table we see that our model predicted 35 observations would vote democrats (the first row), and 28 of them (80%) actually voted democrat. Our model also predicted 28 observations would vote republican, and 22 (79%) of them actually voted republican.

	Actual Democrat	Actual Republican
Predicted Democrat	28	7
Predicted Republican	6	22

As demonstrated, there are four possible outcomes of the confusion matrix. A true positive (TP) is a correct prediction about the outcome we are interested in, in this case whether someone voted republican. We had 22 true positives. Likewise, a true negative (TN) is a correct prediction about whether the observation does not have the outcome, in this case vote for a democrat. We had 28 true negatives. Sometimes we predicted a republican vote but the person voted democrat, this is a false positive (FP) and we had 6 of them. Likewise, sometimes we predicted democrat but the person voted for a republican, this is a false negative (FN) and we had 7 of them.

	Actual Democrat	Actual Republican
Predicted Democrat	TN	FN
Predicted Republican	FP	TP

Two useful measures based on the confusion matrix are:

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN} = \text{Sensitivity} \quad (4.35)$$

The TPR tells us the proportion of positives (republican votes) identified.

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN} = 1 - \text{Specificity} \quad (4.36)$$

The FPR tells us the proportion of negative instances that are incorrectly classified as positive.

Our model can trivially have a 100% true positive rate by always predicting republican, or a 100% true negative rate by always predicting democrat. However, always predicting republican will also lead to many misclassifications of votes for republican that were actually democrat (false positives). That is, increasing the true positives also increases false positives. Generally, there is a tradeoff between the true positive rate and false positive rate based on our model. A cutoff point of 0.5 is not necessarily the best, especially when the base rates of positive and negative instances are unequal. For example, if we know 90% of the people in the study voted republican, we would be more likely to predict a republican vote, even if the person scored low on the conservatism scale.

4.6.2 ROC Curve

We can summarize this tradeoff in a Reciever Operating Characteristic (ROC) curve, which plots the true positive rate against the false positive rate for different cutoff probabilities:

```
install.packages("ROCR", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(ROCR)
# Generate predictions from model
predictions <- prediction(predict(voteglm, type = "response"), votes$voterel)
# Generate true positive and false positive rates from predictions
perf <- performance(predictions, measure = "tpr", x.measure = "fpr")
png(filename = "roc1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(perf,
      colorize      = TRUE,
      color         = rainbow(10),
      main          = "Receiver Operating Characteristic",
      print.cutoffs.at = seq(.0, .9, by = 0.1),
      text.adj     = c(-.5, 1.2),
      xlab         = "False Positive Rate (1-Specificity) (FP/(FP+TN))",
      ylab         = "True Positive Rate (Sensitivity) (TP/(TP+FN))")
abline(0, 1)
dev.off()
```

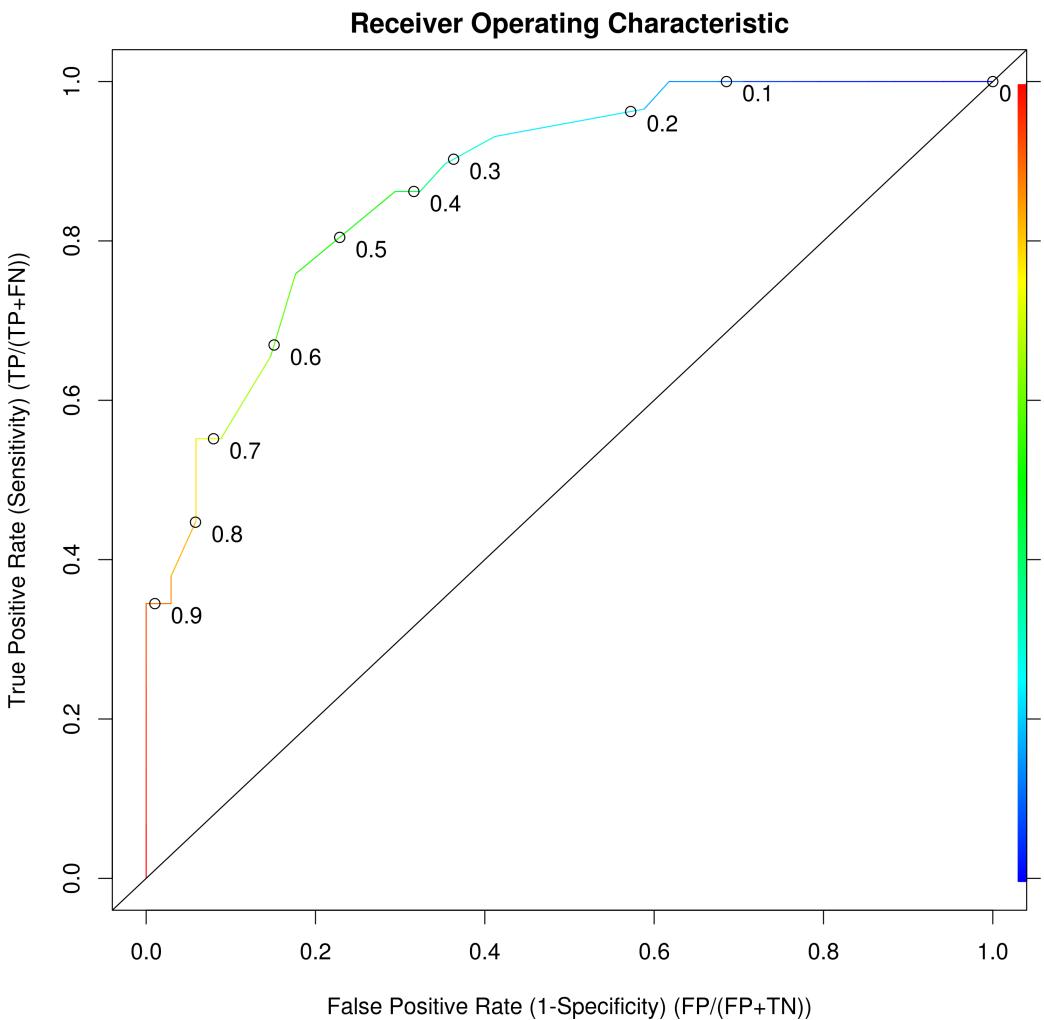


Figure 4.38: Receiver Operating Characteristic (ROC) curve showing true positive rates against false positive rates for several cutoff points from the logistic regression predicting voting for the republican candidate for governor depending on conservatism score.

The curve shows us the true positive rate that we would get for accepting a certain level of false positives. For example, if we could accept a 40% false positive rate, incorrectly identifying 40% of democrats as republicans, we would obtain about a 90% true positive rate, correctly identifying 90% of republicans as republicans. We would use a cutoff of a predicted probability of republican of 0.3, such that if the probability of republican vote from our model was greater than 0.3 we would predict the person votes republican, and democrat otherwise. The $y = x$ line shows how one would do with no model, just randomly guessing by flipping a coin. Along this line one has equal true

positive and false positive rates.

A fairly general way of summarizing the model's predictive ability is called the *Area Under the Curve* (AUC), which is just the area under the ROC curve compared to the random guessing $y = x$ line. Intuitively, the AUC tells us how likely the model is to assign a higher probability of voting republican to a randomly selected republican voter compared to a randomly selected democratic voter. We can compute the AUC using this intuition:

```
# Sample 10000 probabilities for a random person that voted republican
reps <- sample(votes$prob[votes$voterep == 1], size = 10000, replace = TRUE)
# Sample 10000 probabilities for a random person that voted democrat
dems <- sample(votes$prob[votes$voterep == 0], size = 10000, replace = TRUE)
# For each random observation, we are correct if we assign a higher probability
# to a republican being a republican than a democrat to being a republican
correct <- ifelse(reps > dems, 1, 0)
AUC <- sum(correct)/length(correct)
```

The AUC tells us that we will assign a higher probability of being a republican voter to randomly selected republican voters than democrat voters 86% of the time, which is pretty good. We can verify the AUC using ROCR:

```
AUC.ROCR <- performance(predictions, measure = "auc")
```

The ROCR AUC is 0.87, higher than our calculated AUC, which may reflect sampling variation. [Hosmer et al. \(2013\)](#) provide a nice example of plots that simultaneously show the estimated probabilities for each observation, estimated probabilities for each observation by outcome, and the ROC curve:

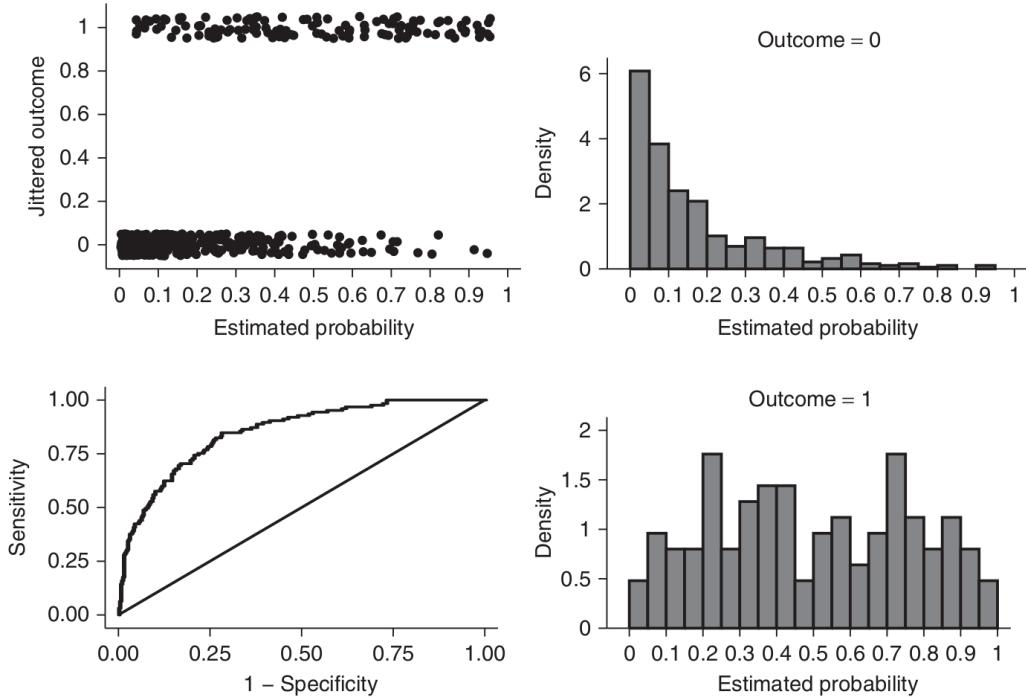


Figure 4.39: Plots of the estimated probabilities for each observation, estimated probabilities by outcome, and ROC curve. From Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression*. John Wiley Sons.

4.6.3 Cross-Validated ROC Curve

Remember that a good test of the forecasting ability of our model is to see how well it predicts new data, and that we can use cross-validation to do this. This holds for logistic regression too, and is especially useful for constructing a cross-validated ROC curve. This is especially important when building more complex models, for example with interactions or non-linear transformations, as their cross-validated ROC performance will usually be more affected than simpler models:

```
# Create an empty data frame to store the predictions
predictions <- data.frame(rep(NA, nrow(votes)/3))
# Remove the column of NAs
predictions <- predictions[, -1]
# Create an empty data frame to store the actual outcomes
labels <- data.frame(rep(NA, nrow(votes)/3))
# Remove the column of NAs
labels <- labels[, -1]
```

```

for(i in 1:1000){
  # Take a random sample, without replacement from 2/3 of the rows
  samp1 <- sample(rownames(votes), 2*nrow(votes)/3, replace = FALSE)
  # Training data takes the sampled rows
  training <- votes[samp1, ]
  # Test data takes the remaining rows
  testing <- votes[setdiff(rownames(votes), samp1), ]
  # Run glm on training data
  glm <- glm(voterep ~ conserv,
             data = training,
             family = binomial(link = "logit"))
  # Make probability predictions for test data
  glmpred <- predict(glm, testing, type = "response")
  # Add the predictions for this iteration to the data frame
  predictions <- cbind(predictions, glmpred)
  # Add the actual outcomes for this iteration to the data frame
  labels <- cbind(labels, testing$voterep)
}
# Create a list with the predictions and labels
cvdata <- list(predictions, labels)

# Run the ROC prediction and performance measures
glmerr <- prediction(cvdata[[1]], cvdata[[2]])
glmperf <- performance(glmerr, measure="tpr", x.measure="fpr")
# This gives a vector of AUCs
glmauc <- performance(glmerr, measure = "auc")
# Unlist the AUCs
glmauc <- unlist(glmauc@y.values)
# Take the average
glmauc <- mean(glmauc)

```

Plotting the cross-validated ROC curve:

```

png(filename = "roc2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(glmperf,
      colorize      = TRUE,
      color         = rainbow(10),
      main          = "Cross-Validated ROC Curve",
      avg           = 'threshold',
      spread.estimate = 'stddev',
      print.cutoffs.at = seq(.0, .9, by = 0.1),
      text.adj      = c(-.5, 1.2),

```

```
xlab      = "Average False Positive Rate",
ylab      = "Average True Positive Rate")
abline(0, 1)
dev.off()
```

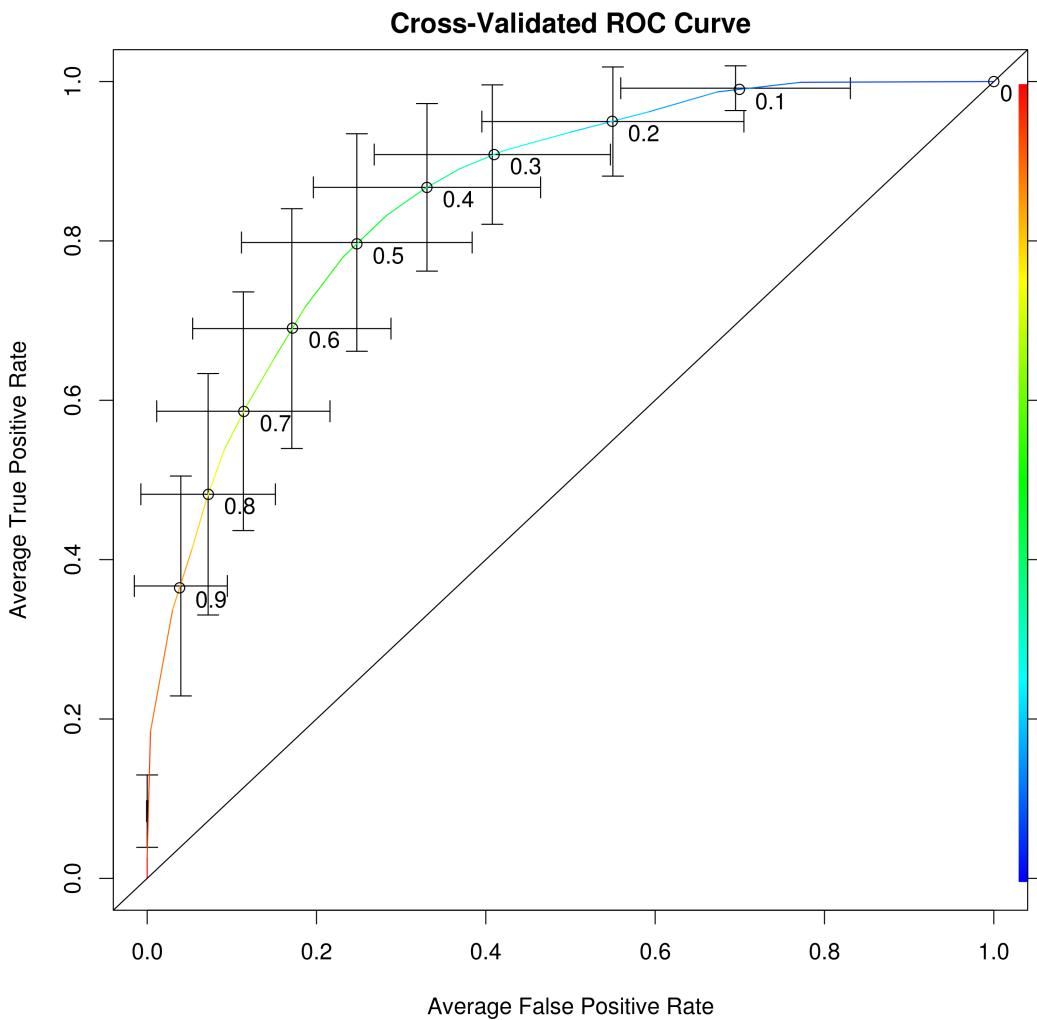


Figure 4.40: Cross-validated Receiver Operating Characteristic (ROC) curve showing true positive rates against false positive rates for several cutoff points from the logistic regression.

The cross-validated ROC curve shows 1000 iterations of the model using 3-fold cross-validation. Error bars show the standard deviation of the true positive rate and false positive rate for each cutoff. Because we have few observations, the error bars are pretty wide. For example, for the usual 0.5, cutoff, it is reasonable to expect anywhere from a 0.1 to a 0.40 false positive rate, and 0.65 to 0.85 true positive rate.

4.7 Statistical Inference Story

As always, the statistical inference story depends almost entirely on the relationship between our sample and population. Was it a random sample? If yes (and it never will be), we can proceed with our computations. If no, our confidence intervals will not have correct coverage probabilities (and we might even not know what they're supposed to cover).

To get the variance of the parameter estimates in a logistic regression model we need the matrix of second partial derivatives of the likelihood function with respect to the model parameters, called the *Hessian* (H). The negative of this matrix is called the *Observed Information* matrix ($I(\beta)$). The partial derivatives have the form (Hosmer et al., 2013):

$$\frac{\partial^2 \log(L(\beta))}{\partial \beta_j^2} = - \sum_{i=1}^n x_{ij}^2 \hat{p}_i (1 - \hat{p}_i) \quad (4.37)$$

and

$$\frac{\partial^2 \log(L(\beta))}{\partial \beta_j \partial \beta_l} = - \sum_{i=1}^n x_{ij} x_{il} \hat{p}_i (1 - \hat{p}_i) \quad (4.38)$$

With two regressors β_1 and β_2 , this would be a 2×2 matrix:

$$H = \begin{pmatrix} \frac{\partial^2 \log(L(\beta))}{\partial \beta_1^2} & \frac{\partial^2 \log(L(\beta))}{\partial \beta_1 \partial \beta_2} \\ \frac{\partial^2 \log(L(\beta))}{\partial \beta_1 \partial \beta_2} & \frac{\partial^2 \log(L(\beta))}{\partial \beta_2^2} \end{pmatrix}$$

or

$$H = \begin{pmatrix} - \sum_{i=1}^n x_{i1}^2 \hat{p}_i (1 - \hat{p}_i) & - \sum_{i=1}^n x_{i1} x_{i2} \hat{p}_i (1 - \hat{p}_i) \\ - \sum_{i=1}^n x_{i1} x_{i2} \hat{p}_i (1 - \hat{p}_i) & - \sum_{i=1}^n x_{i2}^2 \hat{p}_i (1 - \hat{p}_i) \end{pmatrix}$$

The observed information matrix is $I(\beta) = -H$. It is possible to rewrite $I(\beta) = X' \hat{V} X$ where \hat{V} is a matrix with $\hat{p}_i(1 - \hat{p}_i)$ on the main diagonal and zeros otherwise. The variance-covariance matrix of the parameter estimates is the inverse of the observed information matrix $\text{Var}(\beta) = I(\beta)^{-1}$, with the variance of each parameter on the main diagonal and covariance between parameter estimates on the off-diagonal. We can continue the example we used earlier to discuss maximum likelihood estimation:

```
# Generate the data
x1 <- runif(1000, -2, 2)
x2 <- runif(1000, -3, 3)
x <- cbind(rep(1, length(x1)), x1, x2)
beta0 <- .5
```

```

beta1 <- 2
beta2 <- 1
# Create linear predictor xb
xb <- beta0 + beta1*x1 + beta2*x2
# Use logistic function
p <- exp(xb)/(1 + exp(xb))

# Generate coin flips where p(heads) = logistic(xb)
y <- rbinom(1000, 1, p)

```

Again, define the log-likelihood function:

```

# Based on code by Thomas Debray
# Define the negative log likelihood function
logl <- function(theta, x, y){
  y <- y
  x <- as.matrix(x)
  beta <- theta[1:ncol(x)]

  # Define p.xb
  p.xb <- 1/(1 + exp(- x %*% beta))

  # Use the log-likelihood of the Bernouilli distribution
  loglik <- sum(y*log(p.xb)) + sum((1-y)*log(1 - p.xb))

  return(-loglik)
}

```

We find the $\hat{\beta}$ values that minimize the negative log-likelihood:

```

# Define initial values for the parameters
theta.start <- rep(0, (dim(x)[2]))
names(theta.start) <- colnames(x)

# Calculate the maximum likelihood
mle <- optim(theta.start,
  logl,
  x = x,
  y = y,
  hessian = F)

# Obtain regression coefficients
beta.hat <- mle$par

```

Finally, we calculate the observed information matrix and invert it to get the variance-covariance matrix of the parameters:

```
# Calculate the Information matrix
p <- 1/(1 + exp(-x %*% beta.hat))
V <- array(0, dim = c(dim(x)[1], dim(x)[1]))
# The variance of a Bernoulli RV is given by p(1-p)
diag(V) <- p*(1 - p)
IB <- t(x) %*% V %*% x
vcov <- solve(IB)

glm1 <- glm(y ~ x1 + x2, family = binomial(link = "logit"))
vcov(glm1)
vcov
```

4.7.1 Confidence Intervals For Coefficients and Odds Ratios

Like in previous chapters, statistical inferences depend mostly on how the sample was collected. From our previous voting example:

```
voteglm <- glm(voterep ~ conserv, data = votes,
                 family = binomial(link = "logit"))
summary(voteglm)
```

If we want to form a confidence interval around the odds ratio, we just exponentiate the confidence limits at ± 2 standard errors:

```
# We can pull out the standard errors by looking at the
# coefficients of the summary of our regression object
# This summary has a matrix of coefficients and standard errors
# Here we pull out row 2 column 2, which is the std error
# of the conserv regressor
se <- coef(summary(voteglm))[2, 2]

# Next, get the point estimate of the odds ratio
odds.est <- exp(coef(voteglm)["conserv"])
# Get the limits of the 95% confidence interval for the odds ratio
odds.lower <- exp(coef(voteglm)["conserv"] - 2*se)
odds.upper <- exp(coef(voteglm)["conserv"] + 2*se)
```

The odds ratio is 8.82, with a 95% confidence interval from 2.8 to 27.7.

4.7.2 Confidence Interval for the Conditional Mean

The confidence interval for the conditional mean is constructed in very much the same way as in linear regression, with a few additional transformations. Here the conditional mean is a conditional probability. The variance of the conditional mean (fitted value) on the log odds scale is (with two regressors):

$$\text{Var}(\log(\text{odds})|x) = \text{Var}(\beta_0 + \beta_1 x_1 + \beta_2 x_2) \quad (4.39)$$

this is equal to the sum of the variances and covariances of the betas:

$$= \text{Var}(\beta_0) + x_1^2 \text{Var}(\beta_1) + x_2^2 \text{Var}(\beta_2) + 2x_1 \text{Cov}(\beta_0, \beta_1) + 2x_2 \text{Cov}(\beta_0, \beta_2) + 2x_1 x_2 \text{Cov}(\beta_1, \beta_2) \quad (4.40)$$

Because the variance-covariance matrix of $\hat{\beta}$ is $I(\hat{\beta})^{-1}$, we can obtain:

$$\text{Var}(\log(\text{odds})|x) = x_*' (X' \hat{V} X)^{-1} x_* \quad (4.41)$$

where x_* is the value of the x variables that we are using to form the confidence interval (e.g., $x_1 = 5, x_2 = 3$). We can then form the confidence interval as $x_* \beta \pm 2 \times \sqrt{x_*' (X' \hat{V} X)^{-1} x_*}$. We then need to convert this to the probability scale. Here's an example. Suppose we want to form a confidence interval for the predicted probability of voting given a conservatism level of 2:

```
voteglm <- glm(voterep ~ conserv, data = votes,
                  family = binomial(link = "logit"))
# Get variance-covariance matrix of betas
v.cov <- vcov(voteglm)
# Create vector of values for x*
x.star <- c(1, 2)
# Variance and standard error of the log odds
var.logodds <- t(x.star) %*% v.cov %*% x.star
se <- sqrt(var.logodds)
# Fitted value on the log odds scale
xb <- predict(voteglm, newdata = data.frame(conserv = 2))
# Transform to probability
phat <- 1/(1+exp(- (xb)))
lower.ci <- 1/(1+exp(- (xb - 2*se)))
upper.ci <- 1/(1+exp(- (xb + 2*se)))
```

Our prediction for the average probability of voting republican among those with a conservatism score of 2 is .09, with a lower 95% confidence limit of .02 and upper 95% confidence limit of .27. We can conduct confidence bands like in linear regression:

```

voteglm <- glm(voterep ~ conserv, data = votes,
                 family = binomial(link = "logit"))
newdata = data.frame(conserv = seq(1, 5, by = .01))
# Make predictions for new data with normal standard errors
preds <- predict(voteglm, newdata = newdata, se = T)
# convert to probability scale
prob <- 1/(1 + exp(- preds$fit))

png(filename = "normci.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(jitter(votes$conserv), jitter(votes$voterep, amount = .025),
     xlab = "Conservativism",
     ylab = "Predicted Probability of Voting for Republican",
     ylim = c(-.2, 1.2),
     pch = 19,
     col = rgb(0, 0, 0, .5))
lines(newdata$conserv, sort(prob))
prob.lower <- 1/(1 + exp(- preds$fit - 2*preds$se.fit))
prob.upper <- 1/(1 + exp(- preds$fit + 2*preds$se.fit))
lines(newdata$conserv, sort(prob.lower), lty = 3)
lines(newdata$conserv, sort(prob.upper), lty = 3)
dev.off()

```

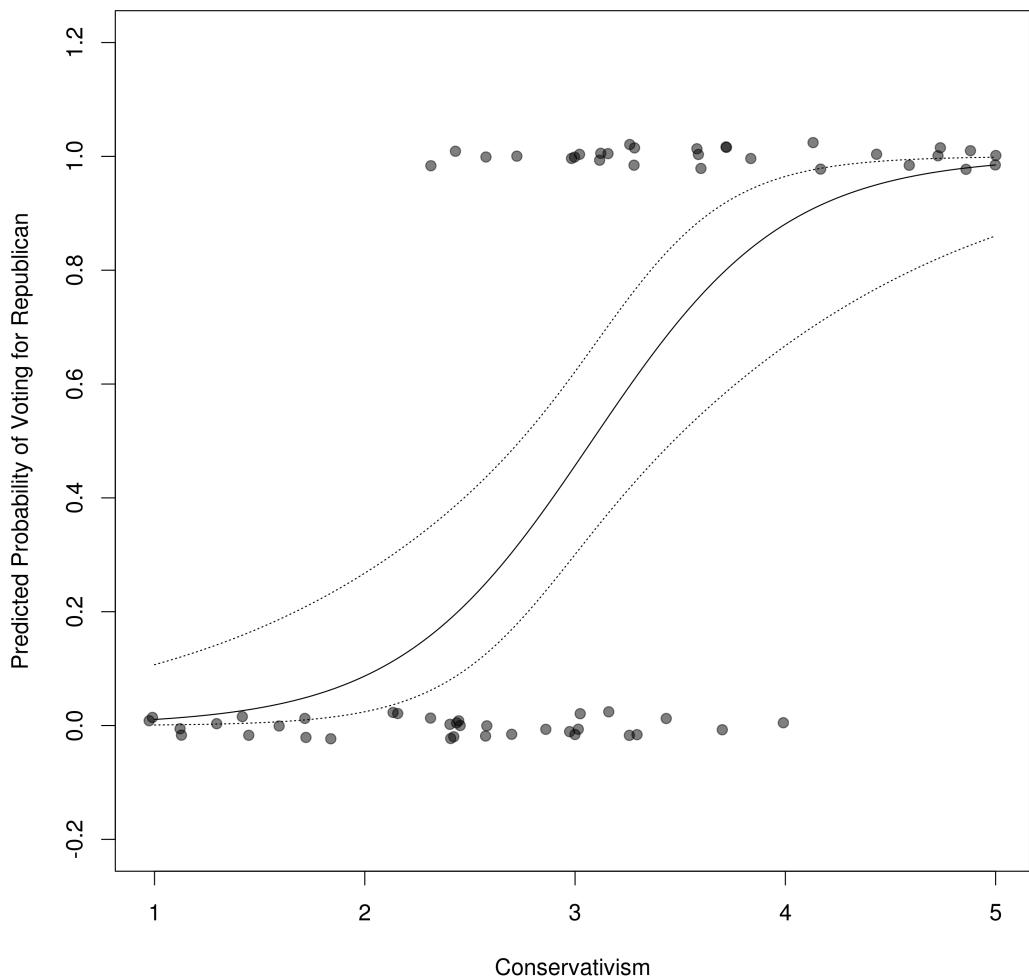


Figure 4.41: Predicted probabilities from logistic regression with 95% confidence bands as dotted lines.

4.8 Homework 4

Replicate the results of [Gerfin \(1996\)](#). Begin by reading section 3. The data can be obtained here:

```
library(AER)
data("SwissLabor")
```

Write a short report that replicates the top part of Table I for Switzerland (page 327). The probit regression can be executed using the following form:

```
glm(y ~ x, family = binomial(link = "probit"))
```

In the replication report, I expect you to:

1. Conduct the probit regression reported in the paper. (1 point)

After attempting to reproduce the table, write a short report telling the five stories of these data. Use the probit model in Table I as a starting point, but use logistic regression or other tools (i.e., not probit). Here's what I expect to be included in the extension part of the report:

1. Create histograms of age, education, number of young children, log of yearly non-labor income, number of older kids, and tables of whether the woman participates in the labor force and whether she is a permanent foreign resident, summarizing what you see. (1 point)
2. Write out the logit (log odds) regression equivalent of the model proposed in Table I (i.e., write the regression model in terms of a function that is linear in the log odds). Next, write the equivalent model in terms of the Bernoulli likelihood function using the logit link function. Explain how the logit function links the linear predictors to the conditional mean (probability). Explain whether it would be better or worse to just take the logit transform of the dependent variable directly. (1 point)
3. Conduct the logistic regression version of the model proposed in Table I. Interpret the logistic regression coefficients in terms of odds ratios. Note: Don't worry about interpreting the age coefficients. Plot the predicted probability of labor force participation against the woman's age, holding the other variables at their mean. (1 point)
4. Create and examine a calibration plot and calibration table for the model proposed in Table 1. Does there seem to be model misspecification? Why or why not? (1 point)

5. Compare logistic regression and a generalized additive model with smoothing on age and education for the model proposed in Table I. Look at the partial residual plots from the GAM. Do any transformations seem necessary to the age or education variables? Why or why not? (1 point)
6. Use Stukel's test for the logistic regression and previously suggested generalized additive model. Does the logit link function seem appropriate? Why or why not?
7. Create a cross-validated ROC curve for the logistic regression with and without any transformations or interactions you think are necessary based on your previous work. Which model performs better in terms of cross-validation and which model do you prefer? Explain your reasoning. (1 point)
8. Briefly comment on the statistical inference and causality stories. Use simulations or confidence intervals if you desire. (1 point)

Here's a list of functions you might need:

- `hist()`
- `table()`
- `gam()` (`mgcv` package)
- `npreg` (`np` package)
- `sims` (`arm` package)

Make sure your report includes reproducible R code, either as an appendix, as a footnote through Harvard's Dataverse (or some other site of your choice), or through CMU's dropbox.

Chapter 5

Multi-level Models

In previous chapters we've considered analyses of cross-sectional data, where observational units each appear once in the dataset. We've looked at different households and their decisions to switch drinking water wells, different women and their decisions to participate in the labor force, and different voters and their decisions about whether to vote for the republican or democratic candidate.

Sometimes we have additional information on each observational unit. For example, a single person may be followed over time, yielding multiple observations for that observational unit (the person), one observation during each time period. Many people may be located in the same city or town, yielding multiple observations in the same observational unit (the city/town). In sum, data often have a *multi-level* or *hierarchical* structure, with observations nested within observational units (e.g., people), which may in turn be nested within higher order observational units (e.g., towns), and we want to account for that structure in our analyses.

Our approach will be to allow these observations within the same observational unit to be more closely related to each other than observations obtained from different observational units. Criminal behavior of individuals will be more similar in the same town than if we randomly select criminals from different towns, possibly reflecting geographic differences such as the presence of gangs or lack of employment opportunities. Repeated judgments from the same individual, for example judgments of quality in Olympic figure skating competitions, likely have a pattern that would not be present if the same number of judgments were obtained from different individuals.

5.1 Introduction to Multi-Level Models

Recall the standard linear predictor formula that we've used, in different ways, for both linear least squares and logistic regression. We have some conditional

mean $E(Y|x)$ that is a linear function of predictors:

$$E(Y_i|x_i) = \beta_0 + \beta_1 x_{i1} + \dots + \beta_k x_{ik} \quad (5.1)$$

We usually consider the different observations i to come from different observational units j . That is, if $i \neq j$, then i and j are different people, places, molecules, etc., and for each observation i there is a unique observational unit j in the dataset that corresponds to i . We now want to extend this idea to allow for observational units j with observations i nested within these units. That is, for each observational unit j (e.g., people) we have multiple observations i (e.g., restaurant choices).

5.1.1 Varying Intercepts

Suppose we have $j = 5$ different observational units, each with some number of observations nested within the unit indexed by i . The simplest way to model the dependence between observations within these observational units is to construct a common intercept for them. Each observational unit has a different intercept from which the individual observations are drawn.

For example, the average height of people living in the Netherlands will be greater than those living in the U.S., and this can be modeled as a height intercept that is greater for the Netherlands than the U.S. Let's call this intercept that varies by observational unit the *varying intercept* $\alpha_{j[i]}$, where the subscript $j[i]$ indicates that we have observations i (e.g., people in the Netherlands and other countries) nested within observational units j (the Netherlands and other countries):

$$y_i = \alpha_{j[i]} + \beta x_i + \epsilon_i \quad (5.2)$$

We can also write out the intercepts explicitly, where there is a different intercept for each country:

$$y_i = \alpha_{1[i]} + \alpha_{2[i]} + \alpha_{3[i]} + \alpha_{4[i]} + \alpha_{5[i]} + \beta x_i + \epsilon_i \quad (5.3)$$

This model is called the *varying-intercept model*. The regressors x have the same relationship to y across the different observational units (i.e., there is just one β for x). For example, the regressor parent's height (x_i) is measured at the level of the person. A varying-intercept model proposes that the relationship between parent's height and child's height (β) is the same regardless of what country one lives in. To get some idea about what this looks like, let's use a numerical varying-intercept example:

```
set.seed(50)
# Overall intercept
```

```

beta0 <- 1
# Generate 10 intercepts, one for each group
# The intercepts are randomly drawn from a N(0, 1) distribution
alpha.j <- rnorm(10, 0, 1)
# Create a group identifier for 100 observations
group <- rep(seq(1, 10, by = 1), 100)
# Draw an error term at the observation level
e.y <- rnorm(1000, 0, 1)
# The dependent variable is the intercept
# plus the group intercept, plus a random component
y <- beta0 + alpha.j[group] + e.y
# Create a data frame with observation and group information
data <- data.frame(y = y, group = group)
png(filename = "interceptexample1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(as.factor(group), y,
      xlim = c(0, 11), pch = 19,
      col = rgb(0, 0, 0, .2),
      ylab = "y",
      xlab = "Group",
      ylim = c(-6, 6))
points(group, y, pch = 20, col = rgb(0, 0, 1, .3))
dev.off()

```

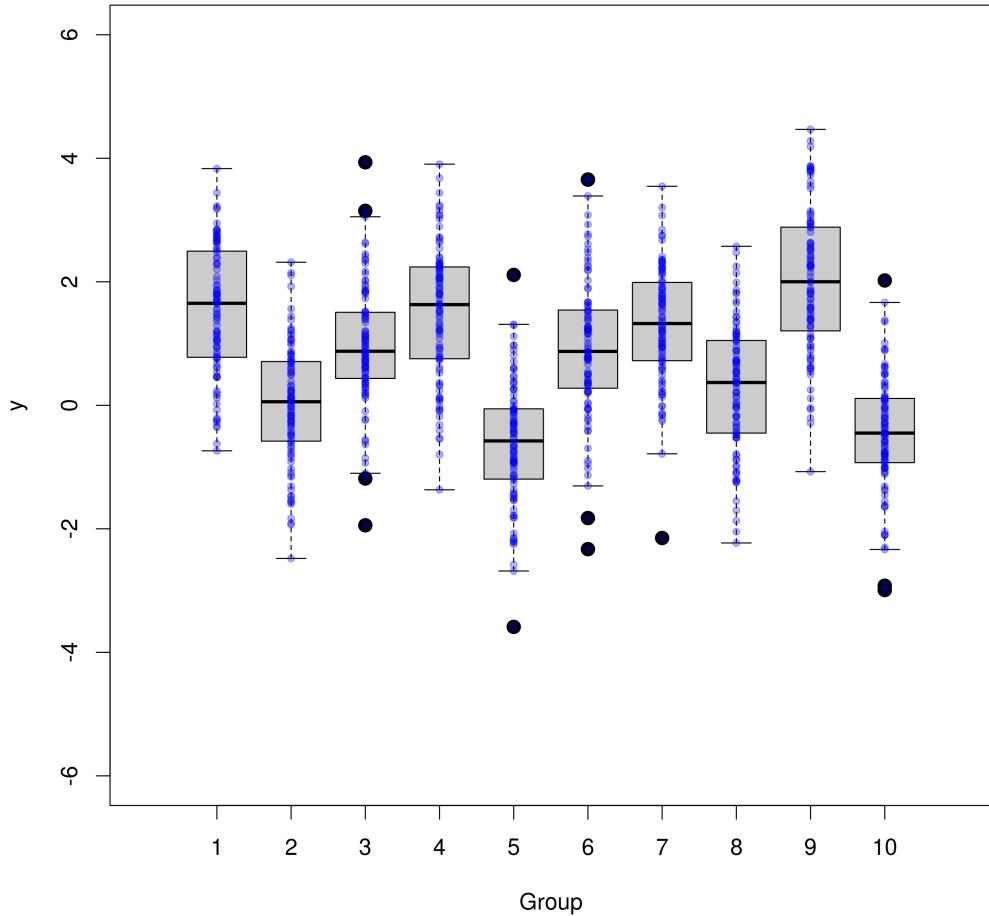


Figure 5.1: 1000 observations clustered within 10 observational units (groups). Each group has a random intercept. Observations within each group vary randomly around that intercept.

As we can see, the averages of each group differ, and this makes sense, as the group-level averages were drawn individually from a standard normal distribution. Within each group, observations then vary randomly around the group-level mean.

5.1.2 Varying Slopes

We may not think that the regressor x has the same relationship to y for all groups. Suppose some countries have issues with malnourishment across generations. In these countries, shorter parents will have shorter children, but so will taller parents. Thus, the relationship between parent's height and

child's height will vary across different observational units (countries) because of factors at the observational unit level (country-level socioeconomic factors). Where the socioeconomic factors are most conducive to normal human growth, we'll have a strong relationship between parent's height and child's height. Where socioeconomic factors harm human growth, we'll have a damped relationship between parent's height and child's height. As a result, we should allow for the regressor parent's height in country j , $x_{j[i]}$, to have a different relationship to y across different countries. This varying relationship is $\beta_{j[i]}$, that is, the slope for observation i within group j . In sum, we can also allow the different groups to have varying slopes, meaning the dependent variable is related to the independent variable to varying degrees across the different groups:

$$y_i = \alpha + \beta_{j[i]}x_i + \epsilon_i \quad (5.4)$$

We can, again, write out the slopes explicitly, where there is a different slope ($\beta_{j[i]}$) for each group j :

$$y_i = \alpha + \beta_{1[i]}x_i + \beta_{2[i]}x_i + \beta_{3[i]}x_i + \beta_{4[i]}x_i + \beta_{5[i]}x_i + \epsilon_i \quad (5.5)$$

Again, to gain insight into what this looks like and the data generating process, let's consider a numerical varying-slope example:

```
set.seed(51)
# Generate the independent variable uniform 0 to 10
x <- runif(1000, 0, 10)
# The overall intercept
beta0 <- 1
# Draw 5 different slopes from the standard normal distribution
beta.j <- rnorm(5, 0, 1)
# Create group assignments for each observation
group <- rep(seq(1, 5, by = 1), 200)
# Create a random error at the observation level N(0, 3)
e.y <- rnorm(1000, 0, 3)
# Create the dependent variable
y <- beta0 + beta.j[group]*x + e.y
data <- data.frame(x = x, y = y, group = group)
png(filename = "slopeexample1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(2, 3), mar = c(5, 4, 2, 1),
    cex.lab = 2, cex.axis = 1.5)
# Loop through the five groups and plot separately
for(i in 1:5){
  plot(data$x[data$group == i],
        data$y[data$group == i],
        xlim = c(0, 10),
```

```
ylim = c(-25, 25),
pch  = 19,
col  = rgb(0, 0, 0, .2),
ylab = "y",
xlab = "x")
# Plot the regression line separately for each group
abline(lm(y ~ x, data = data[group == i, ]), col = "red", lwd = 2)
}
dev.off()
```

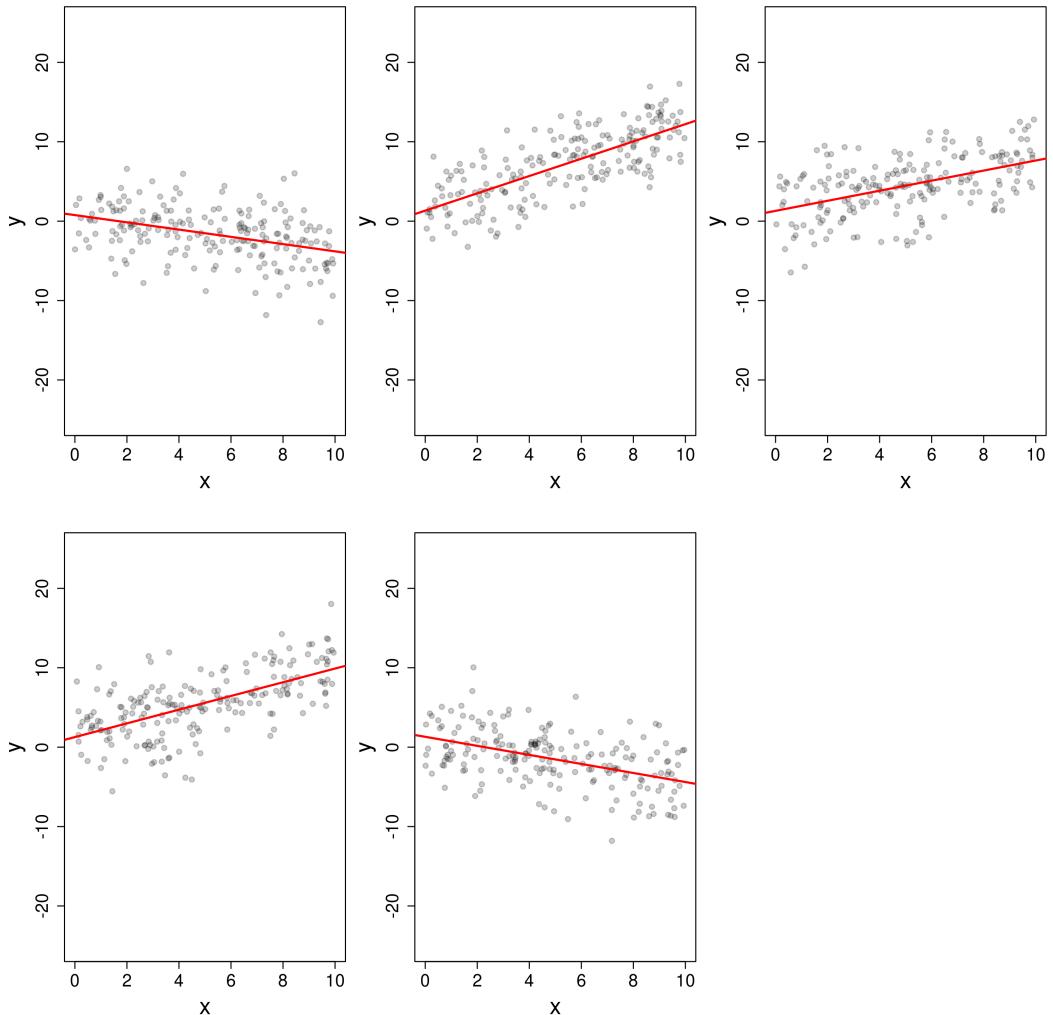


Figure 5.2: 1000 observations clustered within 5 observational units (groups). Each group has a random slope, but same intercept. Observations within each group vary randomly around that slope.

As we can see, the slopes of each group differ, and the individual observations vary randomly around each group-level slope.

5.1.3 Varying Intercepts and Slopes

Finally, we can allow the different groups to have both different intercepts ($\alpha_{j[i]}$) and different slopes ($\beta_{j[i]}$):

$$y_i = \alpha_{j[i]} + \beta_{j[i]}x_i + \epsilon_i \quad (5.6)$$

In general, it does not make sense to allow the group-level slope to vary

but not allow the group-level intercept to vary, so if we have a varying-slope model it will usually also have varying intercepts. Again, here's a numerical varying-slope varying-intercept example:

```
set.seed(53)
# Create the independent variable
x <- runif(1000, 0, 10)
# Draw 5 random intercepts from N(0, 4)
alpha.j <- rnorm(5, 0, 4)
# Draw 5 random slopes from N(0, 2)
beta.j <- rnorm(5, 0, 2)
# Create groups for each observation
group <- rep(seq(1, 5, by = 1), 200)
# Draw a random error term for each observation from N(0, 3)
e.y <- rnorm(1000, 0, 3)
# Create the dependent variable
y <- alpha.j[group] + beta.j[group]*x + e.y
data <- data.frame(x = x, y = y, group = group)
png(filename = "intslopeexample1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(2, 3), mar = c(5, 4, 2, 1),
     cex.lab = 1.5, cex.axis = 1.5)
for(i in 1:5){
  plot(data$x[data$group == i],
        data$y[data$group == i],
        xlim = c(0, 10),
        ylim = c(-40, 40),
        pch = 19,
        col = rgb(0, 0, 0, .2),
        ylab = "y",
        xlab = "x")
  abline(lm(y ~ x, data = data[group == i, ]), col = "red", lwd = 2)
}
dev.off()
```

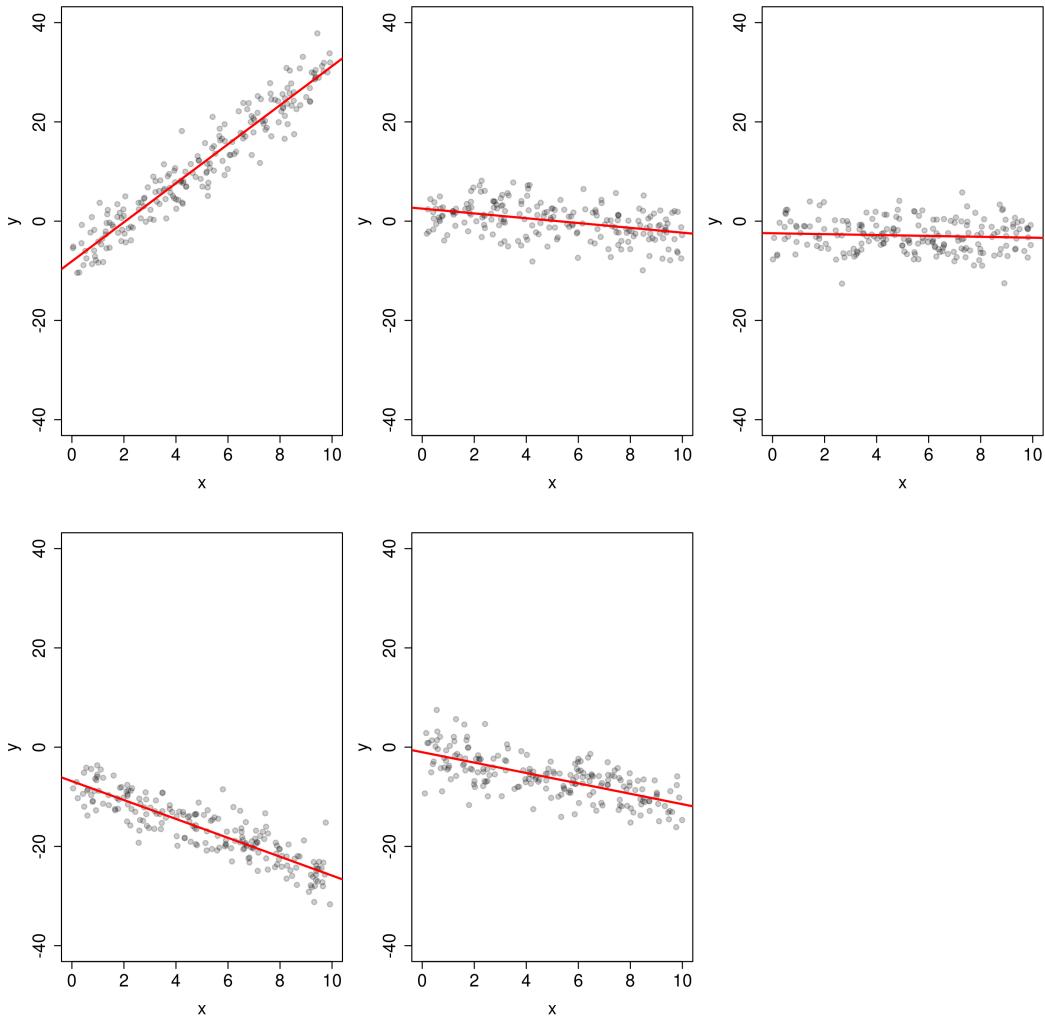


Figure 5.3: 1000 observations clustered within 5 observational units (groups). Each group has both a random intercept and random slope. Observations within each group vary randomly around that group-level intercept and slope.

In this case we could use a “usual” ordinary least squares model by fitting ordinary least squares with dummy variables interacted with the regressor for each group (varying slopes), and a dummy variable for each group (varying intercepts).

5.2 Radon Example

Let’s look at an example of hierarchically structured data of home radon levels in 85 counties in Minnesota, taken from [Gelman and Hill \(2007\)](#). In these data we are primarily interested in trying to understand if the radon levels are

too high in some counties and why. We'll look at household-level regressors, such as where the home radon measurements were taken (at the first floor (1) or in the basement (0)), as well as county-level soil uranium levels in parts per million. Importantly, home radon levels are grouped within each county, giving the observations (radon levels) a nested structure (within county):

```
radon <- read.table(  
  "http://www.stat.columbia.edu/~gelman/arm/examples/radon/srrs2.dat",  
  na.strings = "", header = T, sep = ",")
```

Let's pick out the Minnesota observations:

```
MNradon <- radon[radon$state == "MN", ]
```

Looking at the histogram of radon measurements, they are highly positively skewed and bounded below at zero:

```
png(filename = "radonactivity.png", width = 3000, height = 3000, res = 300)  
par(cex = 1.5, mar = c(5, 4, 2, 1))  
hist(MNradon$activity,  
      xlim = c(0, 50),  
      ylim = c(0, 200),  
      breaks = "FD",  
      main = "",  
      xlab = "Radon Levels (picoCuries/liter)")  
dev.off()
```

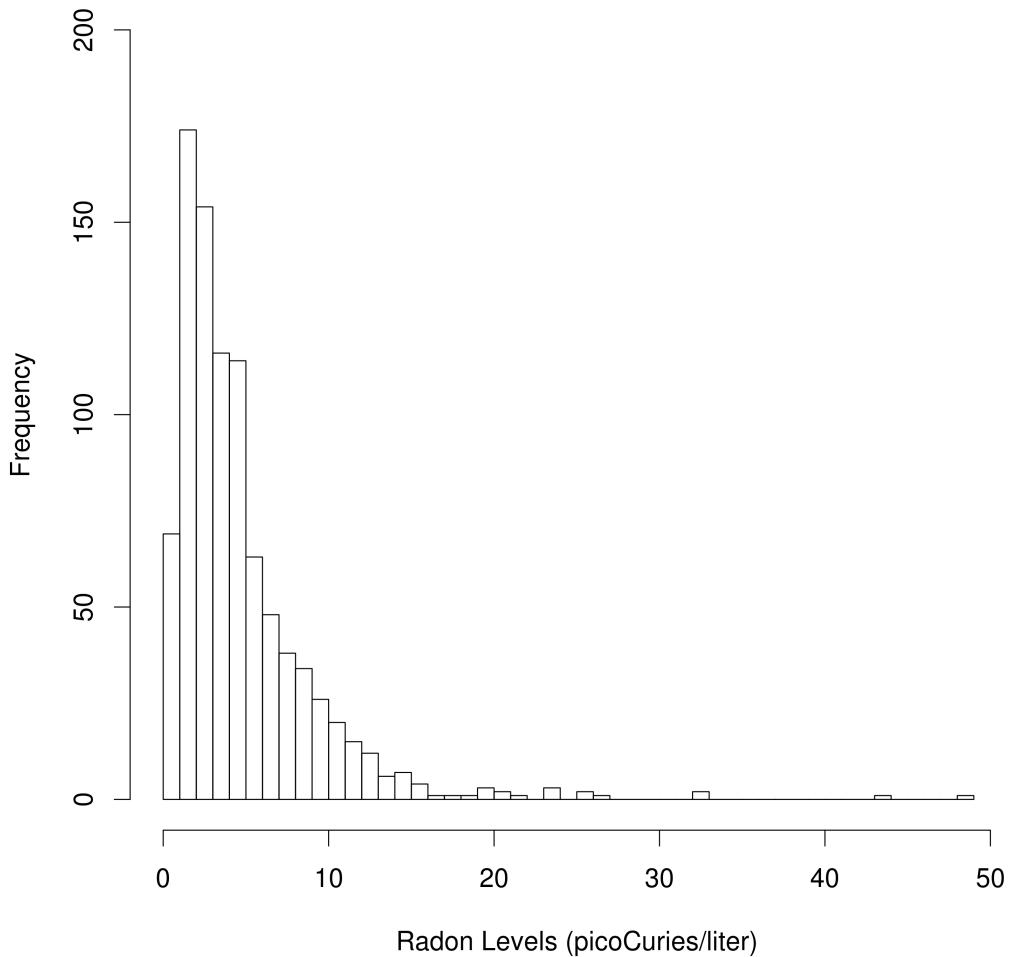


Figure 5.4: Radon levels (in picoCuries/L) in sampled Minnesota homes.

Let's see if the log transformation makes sense by testing different values of λ using the Box-Cox transformation using the Yeo-Johnson power family:

```
library(car)
png(filename = "radonboxcox.png", width = 3000, height = 3000, res = 300)
par(cex = 1.5, mar = c(5, 4, 2, 1))
boxCox(lm(activity ~ 1, data = MNradon), family = "yjPower")
dev.off()
```

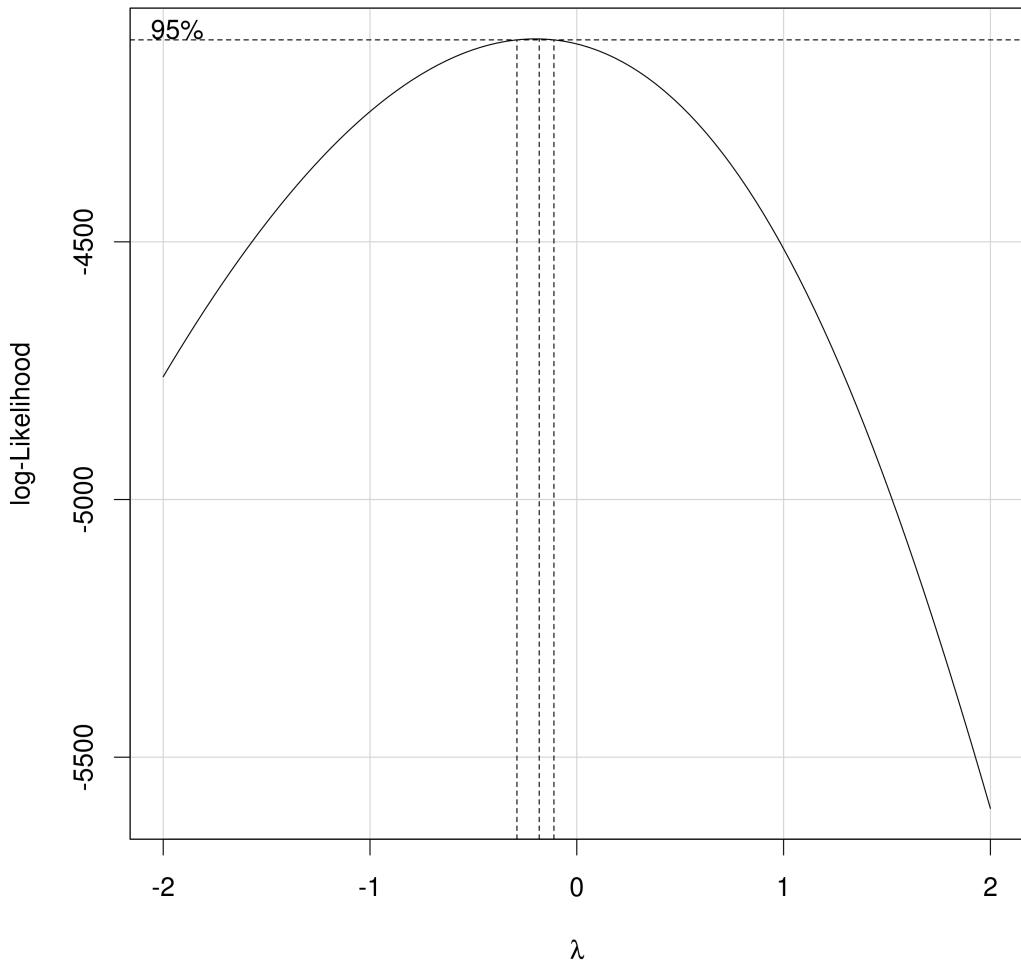


Figure 5.5: Log likelihood of the λ parameter for the Box-Cox transformation of the measured home radon levels.

As usual $\lambda = 0$ is not the maximum likelihood estimate. We're close enough that it might make sense to use the log transform to help with interpretation. Because some radon levels are zero, we'll use the Yeo-Johnson power transformation $g(y) = \log(y + 1)$ and the maximum likelihood estimate $\lambda \approx -0.2$:

```
MNradon$logradon <- log(MNradon$activity + 1)
MNradon$radon0.2 <- ((MNradon$activity+1)^(-0.2)-1)/-0.2
png(filename = "radonacthists.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1), mfrow = c(2, 1),
    cex.axis = 1.1)
hist(MNradon$logradon,
```

```
breaks = "FD",
ylim   = c(0, 140),
main   = "",
xlab   = "Log Radon Levels")
hist(MNradon$radon0.2,
      breaks = "FD",
      ylim   = c(0, 100),
      xlim   = c(0, 3),
      main   = "",
      xlab   = "Radon Levels with Lambda = -0.2 Transformation")
dev.off()
```

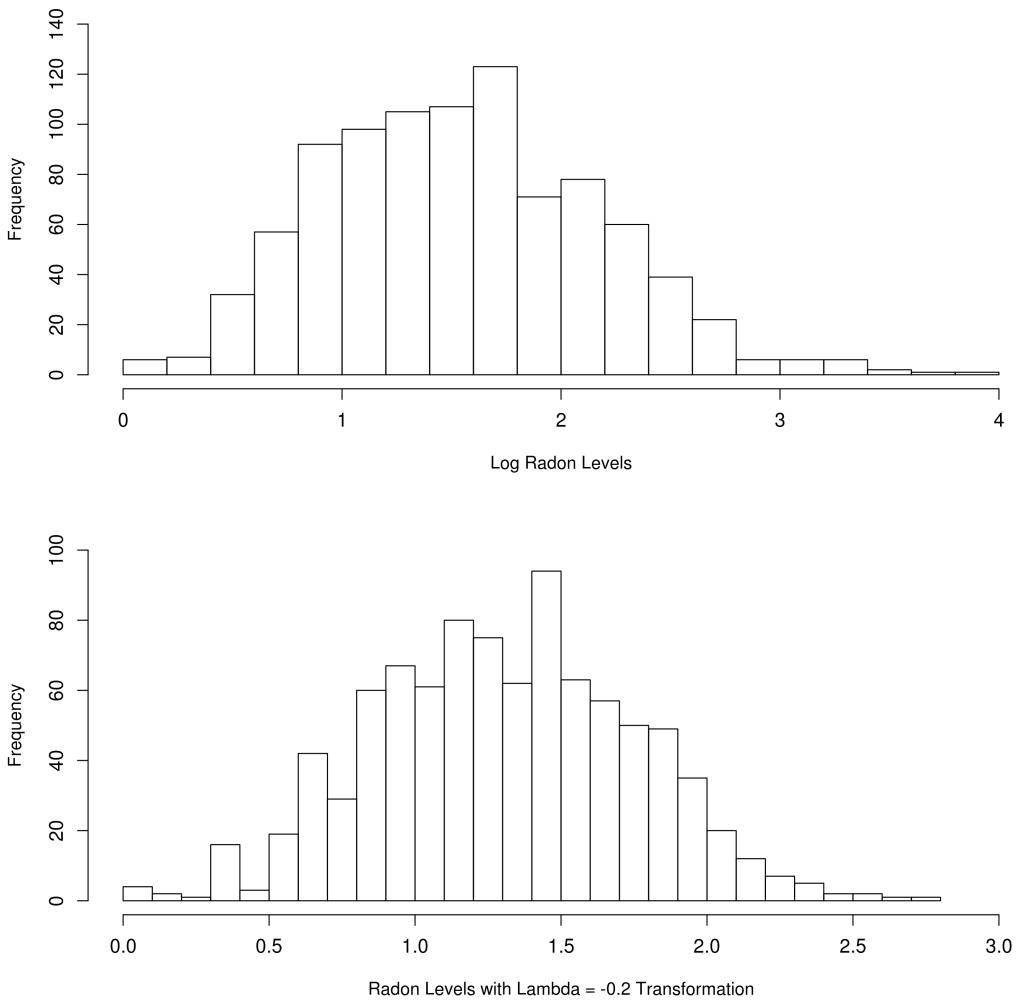


Figure 5.6: Histogram of log transformed radon activity and radon activity with $\lambda = -0.2$ Yeo-Johnson Power transformation.

Both look fairly normal except for the tails, which don't look particularly well-behaved in either case. Examining the qqplots:

```
png(filename = "logradonqq.png", width = 3000, height = 3000, res = 300)
par(cex = 1.5, mar = c(5, 4, 2, 1), mfrow = c(2, 1))
qqPlot(MNradon$logradon,
       ylab = "Log Radon Quantiles",
       xlab = "Normal Quantiles",
       pch = 19,
       col = rgb(0, 0, 0, .4))
qqPlot(MNradon$radon0.2,
       ylab = "Radon Quantiles with Lambda -0.2 Transformation",
```

```
    xlab = "Normal Quantiles",
    pch  = 19,
    col   = rgb(0, 0, 0, .4))
dev.off()
```

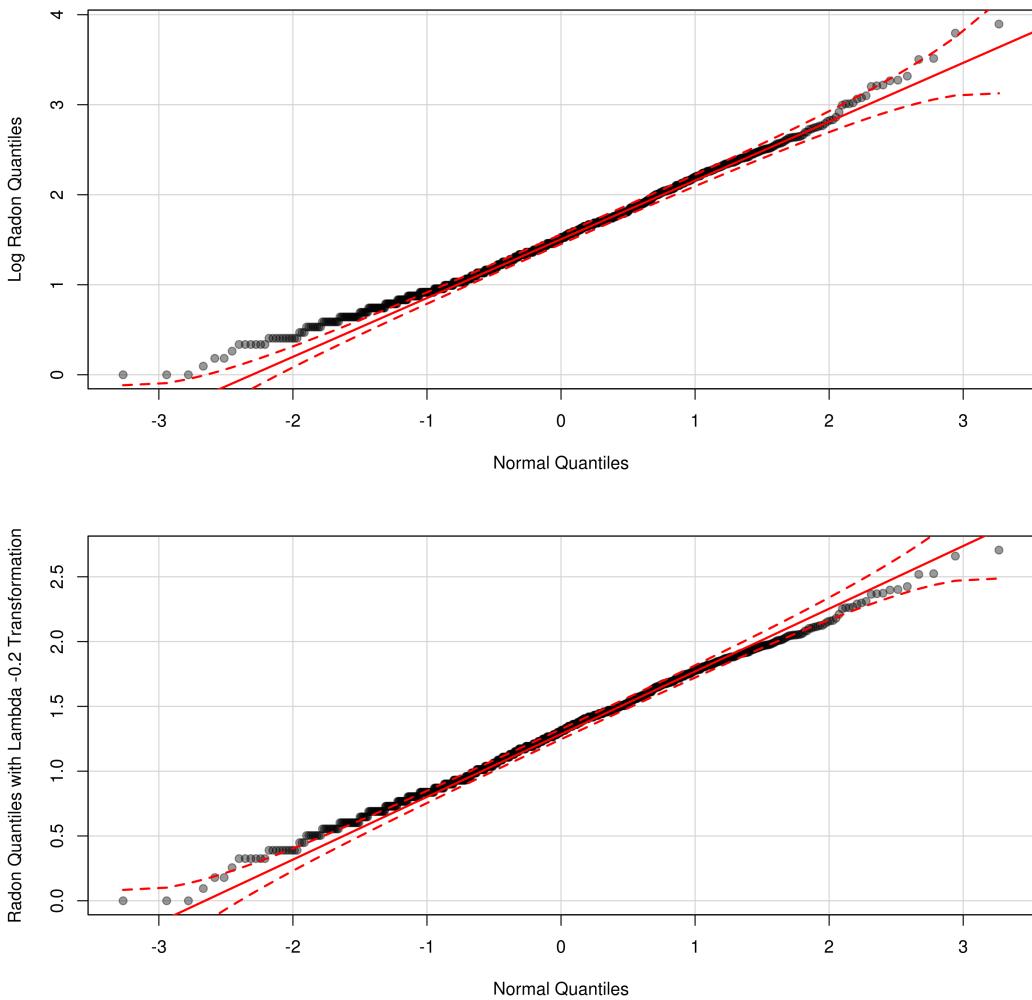


Figure 5.7: Quantile-Quantile plot of log-radon activity levels and $\lambda = -0.2$ transformed radon activity levels versus the normal distribution.

There is definitely some deviation from the log-normal distribution, with the $\lambda = -0.2$ transformation performing better. For the log transform, the lowest quantiles are not low enough compared to the quantiles of the normal distribution. We should expect this, given the Box-Cox method did not suggest $\lambda = 0$. Gelman and Hill chose the log transform but did not check whether it made sense, but we'll stick with their decision so we can continue with the multi-level model exposition (with appropriate reservations). Next, let's look at the logged radon data for each county. First, let's split the data into plots of 12 counties each:

```
# Get data for first 12 counties
plot1 <- MNradon[MNradon$county %in% unique(MNradon$county)[1: 12], ]
```

```

# We need to drop levels or else it will include irrelevant county labels
plot1 <- droplevels(plot1)
plot2 <- MNradon[MNradon$county %in% unique(MNradon$county) [13: 24], ]
plot2 <- droplevels(plot2)
plot3 <- MNradon[MNradon$county %in% unique(MNradon$county) [25: 36], ]
plot3 <- droplevels(plot3)
plot4 <- MNradon[MNradon$county %in% unique(MNradon$county) [37: 48], ]
plot4 <- droplevels(plot4)
plot5 <- MNradon[MNradon$county %in% unique(MNradon$county) [49: 60], ]
plot5 <- droplevels(plot5)
plot6 <- MNradon[MNradon$county %in% unique(MNradon$county) [61: 72], ]
plot6 <- droplevels(plot6)
plot7 <- MNradon[MNradon$county %in% unique(MNradon$county) [73: 84], ]
plot7 <- droplevels(plot7)

```

Next, let's plot the first 48 counties:

```

png(filename = "radonplots1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(2, 2), mar = c(5, 4, 2, 1), mgp = c(3, .3, 0))
plot(plot1$county, plot1$logradon,
      cex = 0.5,
      ylim = c(0, 4),
      ylab = "",
      xaxt = "n")
points(plot1$county, plot1$logradon,
       pch = 20,
       col = rgb(0, 0, 1, .3))
# Adjust the x axis labels
mtext(unique(MNradon$county)[1:12], side = 1,
      at = seq(1:12), adj = .75, las = 2, cex = .75)
# Adjust the y axis label
mtext("Log Radon Activity", side = 2,
      at = 1.25, adj = 0, padj = -3, las = 3, cex = 1)
# Add overall logradon mean, ignoring county
abline(h = mean(MNradon$logradon))

plot(plot2$county, plot2$logradon,
      cex = 0.5,
      ylim = c(0, 4),
      ylab = "",
      xaxt = "n")
points(plot2$county, plot2$logradon,
       pch = 20,

```

```

    col = rgb(0, 0, 1, .3))
# Adjust the x axis labels
mtext(unique(MNradon$county)[13:24], side = 1,
      at = seq(1:12), adj = .75, las = 2, cex = .75)
# Adjust the y axis label
mtext("Log Radon Activity", side = 2,
      at = 1.25, adj = 0, padj = -3, las = 3, cex = 1)
# Add overall logradon mean, ignoring county
abline(h = mean(MNradon$logradon))

plot(plot3$county, plot3$logradon,
      cex = 0.5,
      ylim = c(0, 4),
      ylab = "",
      xaxt = "n")
points(plot3$county, plot3$logradon,
       pch = 20,
       col = rgb(0, 0, 1, .3))
# Adjust the x axis labels
mtext(unique(MNradon$county)[25:36], side = 1,
      at = seq(1:12), adj = .75, las = 2, cex = .75)
# Adjust the y axis label
mtext("Log Radon Activity", side = 2,
      at = 1.25, adj = 0, padj = -3, las = 3, cex = 1)
# Add overall logradon mean, ignoring county
abline(h = mean(MNradon$logradon))

plot(plot4$county, plot4$logradon,
      cex = 0.5,
      ylim = c(0, 4),
      ylab = "",
      xaxt = "n")
points(plot4$county, plot4$logradon,
       pch = 20,
       col = rgb(0, 0, 1, .3))
# Adjust the x axis labels
mtext(unique(MNradon$county)[37:48], side = 1,
      at = seq(1:12), adj = .75, las = 2, cex = .75)
# Adjust the y axis label
mtext("Log Radon Activity", side = 2,
      at = 1.25, adj = 0, padj = -3, las = 3, cex = 1)
# Add overall logradon mean, ignoring county

```

```

abline(h = mean(MNradon$logradon))
dev.off()

```

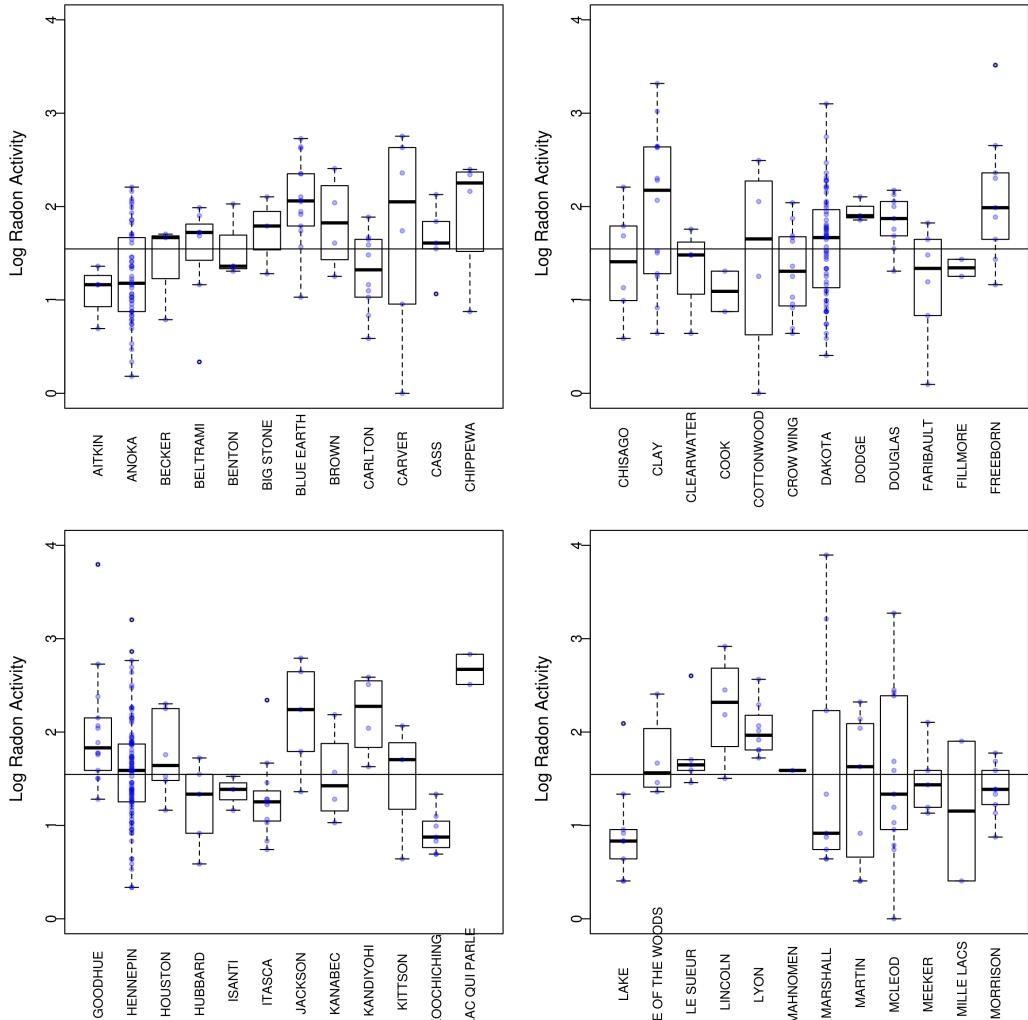


Figure 5.8: Boxplots of log radon activity in each county. Horizontal line is the overall mean log radon activity, ignoring county information.

Let's create the same plots for the remaining counties:

```

png(filename = "radonplots2.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(2, 2), mar = c(5, 4, 2, 1), mgp = c(3, .3, 0))
plot(plot5$county, plot5$logradon,
      cex = 0.5,
      ylim = c(0, 4),
      ylab = "",
      xaxt = "n")

```

```

points(plot5$county, plot5$logradon,
      pch = 20,
      col = rgb(0, 0, 1, .3))
# Adjust the x axis labels
mtext(unique(MNradon$county)[49:60], side = 1,
      at = seq(1:12), adj = .75, las = 2, cex = .75)
# Adjust the y axis label
mtext("Log Radon Activity", side = 2,
      at = 1.25, adj = 0, padj = -3, las = 3, cex = 1)
# Add overall logradon mean, ignoring county
abline(h = mean(MNradon$logradon))

plot(plot6$county, plot6$logradon,
      cex = 0.5,
      ylim = c(0, 4),
      ylab = "",
      xaxt = "n")
points(plot6$county, plot6$logradon,
      pch = 20,
      col = rgb(0, 0, 1, .3))
# Adjust the x axis labels
mtext(unique(MNradon$county)[61:72], side = 1,
      at = seq(1:12), adj = .75, las = 2, cex = .75)
# Adjust the y axis label
mtext("Log Radon Activity", side = 2,
      at = 1.25, adj = 0, padj = -3, las = 3, cex = 1)
# Add overall logradon mean, ignoring county
abline(h = mean(MNradon$logradon))

plot(plot7$county, plot7$logradon,
      cex = 0.5,
      ylim = c(0, 4),
      ylab = "",
      xaxt = "n")
points(plot7$county, plot7$logradon,
      pch = 20,
      col = rgb(0, 0, 1, .3))
# Adjust the x axis labels
mtext(unique(MNradon$county)[73:84], side = 1,
      at = seq(1:12), adj = .75, las = 2, cex = .75)
# Adjust the y axis label
mtext("Log Radon Activity", side = 2,

```

```

at = 1.25, adj = 0, padj = -3, las = 3, cex = 1)
# Add overall logradon mean, ignoring county
abline(h = mean(MNradon$logradon))
dev.off()

```

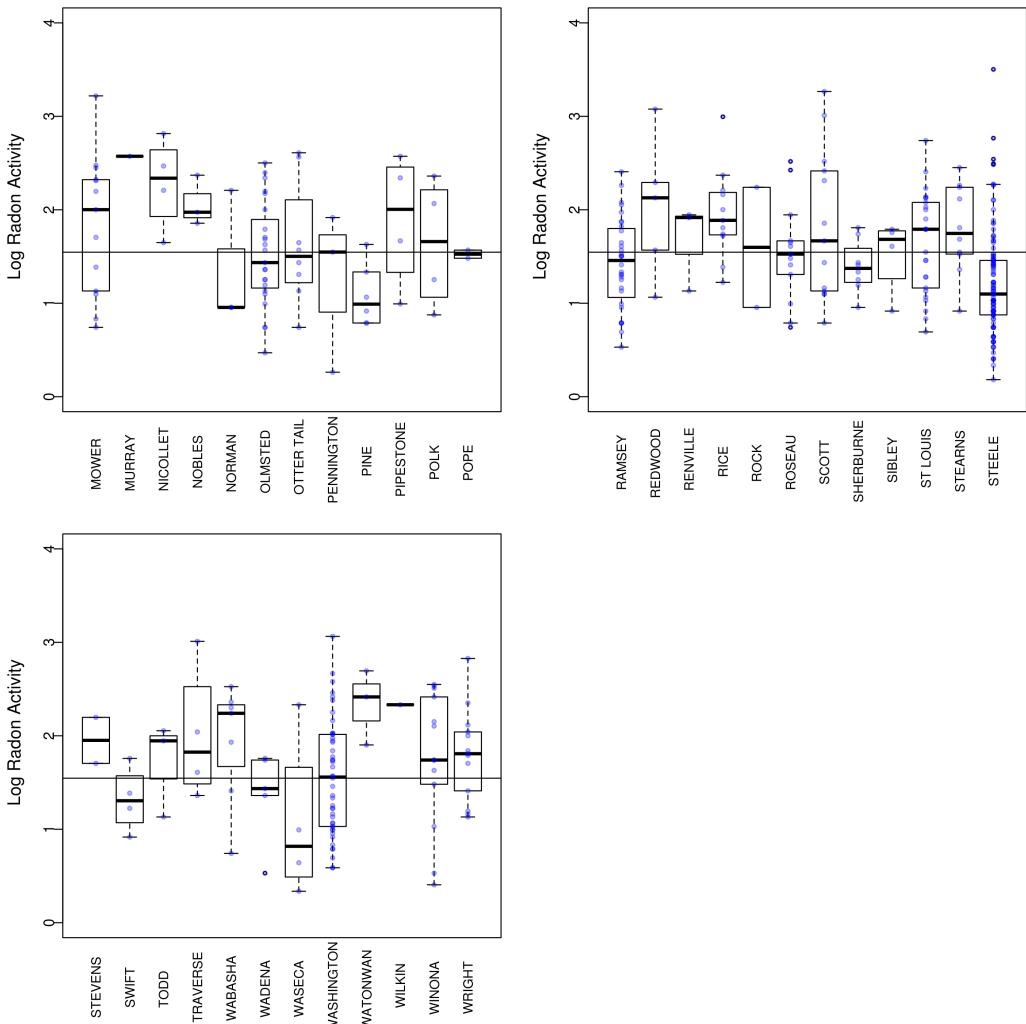


Figure 5.9: Boxplots of log radon activity in each county. Horizontal line is the overall mean log radon activity, ignoring county information.

We can see that many of the counties have log radon levels that are far away from the overall average log radon levels. There is also variation in the number of observations within each county, with some counties having many observations (e.g., Anoka, Dakota), and some having very few (e.g., Lac Qui Parle, Murray, Aitkin). What we are interested in is whether we should be concerned about radon levels in each county based on the measured radon

levels in each individual county, taking into account the variation in radon levels among counties and the sample sizes within each county.

5.3 Three Types of Pooling

We can think about expected radon levels in each household (i.e., the conditional mean radon levels) with hierarchically structured data according to two extremes and a compromise. At one extreme, we can say that the *hierarchical structure doesn't matter*; that is, observations are independent of each other even though they share a common observational unit. This is called *complete pooling*, and this is equivalent to running an ordinary least squares regression on all the data combined, ignoring the structure. From our introductory varying intercept and slope examples, it was clear that we could not ignore the hierarchical structure, as we could see the clustering of observations around the group mean or group slope. In the radon example, however, the pattern is harder to see, suggesting we might be able to ignore the fact that radon measurements are nested within counties.

At the other extreme, we may think that the hierarchical structure matters so much that observations in different observational units cannot be grouped together in any way, as they are simply not comparable. This is called *no pooling*, and is equivalent to running separate regressions for each observational unit. The compromise is *partial pooling*, which analyzes all the data together, but does not assume observations are independent if they come from the same observational unit. Summarizing the three possibilities:

1. *Complete Pooling*: Ignore the group structure and analyze all the data together. The regression parameters do not vary by group (i.e., the variance of the intercepts and slopes across the groups is zero).
2. *No Pooling*: Run separate regressions for each group. The parameters from different groups are not assumed to be related at all (i.e., the variance of intercepts and slopes across groups is ∞ , or the groups are treated as if they are unrelated). A more common version, that we will use, is to dummy code the group variables (sometimes called fixed effects or least squares dummy variable).
3. *Partial Pooling*: Somewhere between complete and no pooling is partial pooling, where intercepts and slopes vary by group, with the amount of pooling determined by the sample size in each group, the variance of each observation around its group mean, and the variance of the group means around the mean of the groups.

A complete pooling analysis is simple but fails to capture group structure. The higher the correlation between observations within each observational

unit, the more the *variance* of parameter estimates will be biased. In the limit, zero correlation between observations within observational units means no bias in the variance estimate, whereas perfect correlation between observations within observational unit indicates that our sample size is equal to the number of *observational units*, not the number of observations. Complete pooling also misses the point, in that we are not able to understand our model at the level of the observational unit (e.g., county), which is often the task at hand.

No pooling analysis, on the other hand, suffers from small sample sizes, where some groups may have very few observations. Often we are interested in every group, even though there are few observations. Another issue is called the incidental parameters problem ([Lancaster, 2000](#)), where the intercepts and slopes that vary by group using the no pooling regression are not consistent, meaning as we collect data on more groups, our estimates do not converge to their true values. Thus, if we are interested in modeling the groups themselves, no pooling analysis won't work.

Partial pooling overcomes these problems, and multi-level models use the partial pooling approach. The general idea of partial pooling is to be somewhere between complete pooling and no pooling. The multi-level model simultaneously estimates regression parameters at the individual and group levels. Group-level parameters are themselves modeled, and estimated from the data, just as individual level parameters.

5.3.1 Fixed and Random Effects

A multi-level partial pooling model with varying intercepts and/or slopes is often called a *random effects* model because the regression coefficients are treated as random variables with their own distributional assumptions. Recall from the simulations that each group was given a random intercept and/or slope that was drawn from a common normal distribution. For example, the intercepts (α_j) may be drawn from a normal distribution with mean zero and standard deviation two, while the slopes (β_j) may be drawn from a normal distribution with mean one and standard deviation three:

```
# Random intercepts
alpha.j <- rnorm(5, 0, 2)
# Random slopes
beta.j <- rnorm(5, 1, 3)
```

In general, these distributions do not need to be normal, although to generalize the multi-level model approach we would need to use hierarchical bayesian models ([Gelman and Hill, 2007](#)). *Fixed effects*, on the other hand, are not themselves modeled as a function of group structure, meaning we do not think group-level intercepts or slopes share a common distribution.

Instead they are estimated from the data in the usual way, for example by using a dummy variable for each group, picking one group as the “baseline” comparison.

A model using both fixed and random effects is often called a “mixed effects” model. Unfortunately, the terms fixed and random effects are not used consistently. [Gelman and Hill \(2007\)](#) identify 5 ways the terms fixed and random effects are used:

1. *The variation across individuals.* Fixed effects are constant across individuals, whereas random effects vary across individuals.
2. *The goals of inference.* Fixed effects model a parameter of interest, whereas random effects are ignored to focus on the underlying population.
3. *The exhaustiveness of the sample.* Fixed effects are used when the sample exhausts the population, whereas random effects are used when the sample is small relative to the population.
4. *The generative process.* A random effect is the realized value of a random variable.
5. *The estimation procedure.* Fixed effects are estimated using least squares or maximum likelihood, random effects are estimated using shrinkage or feasible generalized least squares.

When discussing multi-level models, we follow Gelman and Hill by avoiding the terms fixed and random effects where possible, and instead describe the models in terms of varying intercepts and slopes that are modeled. A more functional way of thinking about it is that fixed effects are allowed to be correlated with the regressors, whereas random effects are assumed to be uncorrelated with the regressors ([Wooldridge, 2002](#)).

5.3.2 Completely Pooled Intercept

As discussed, the complete pooling analysis estimates one intercept (and/or slope) for all observations, ignoring any grouping or hierarchical structure. Here’s an example where the data exhibit group structure but we ignore that structure, and instead fit a complete pooling model:

```
set.seed(1)
beta0 <- 1
# Generate random intercepts by group
alpha.j <- rnorm(10, 0, 1)
```

```

group <- rep(seq(1, 10, by = 1), 10)
e.y <- rnorm(100, 0, 1)
y <- beta0 + alpha.j[group] + e.y
data <- data.frame(y = y, group = group)
# Use complete pooling for the intercept, ignoring group structure
completetpool1 <- lm(y ~ 1, data = data)
# Make predictions for the intercept for each group
# With 95% confidence intervals
prd <- predict(completetpool1,
                newdata = data.frame(group = group),
                interval = c("confidence"),
                level = 0.95, type = "response")

png(filename = "completetpoolexample1.png", width = 3000,
     height = 3000, res = 300)
par(cex = 1.3, mfrow = c(1, 2), mar = c(5, 4, 2, 1),
     cex.axis = 1.3, cex.lab = 1.3)
plot(as.factor(group), y,
      xlim = c(0, 11),
      pch = 19,
      col = rgb(0, 0, 0, .2),
      ylab = "y",
      xlab = "Group",
      ylim = c(-1, 4))
points(group, y,
       pch = 20,
       col = rgb(0, 0, 1, .3))
plot(as.factor(group), y,
      xlim = c(0, 11),
      pch = 19,
      col = rgb(0, 0, 0, .2),
      ylab = "y",
      xlab = "Group",
      ylim = c(-1, 4))
points(group, y,
       pch = 20,
       col = rgb(0, 0, 1, .3))
# Add the complete pooling intercept line
abline(h = coef(completetpool1), col = "red", lty = 2, lwd = 4)
abline(h = 1, col = "blue", lwd = 4)
# Add line showing the lower confidence limit from the
# complete pooling regression

```

```
lines(group, prd[, 2], col ="red", lty = 2, lwd = 2)
# Add line showing the upper confidence limit from the
# complete pooling regression
lines(group, prd[, 3], col = "red", lty = 2, lwd = 2)
dev.off()
```

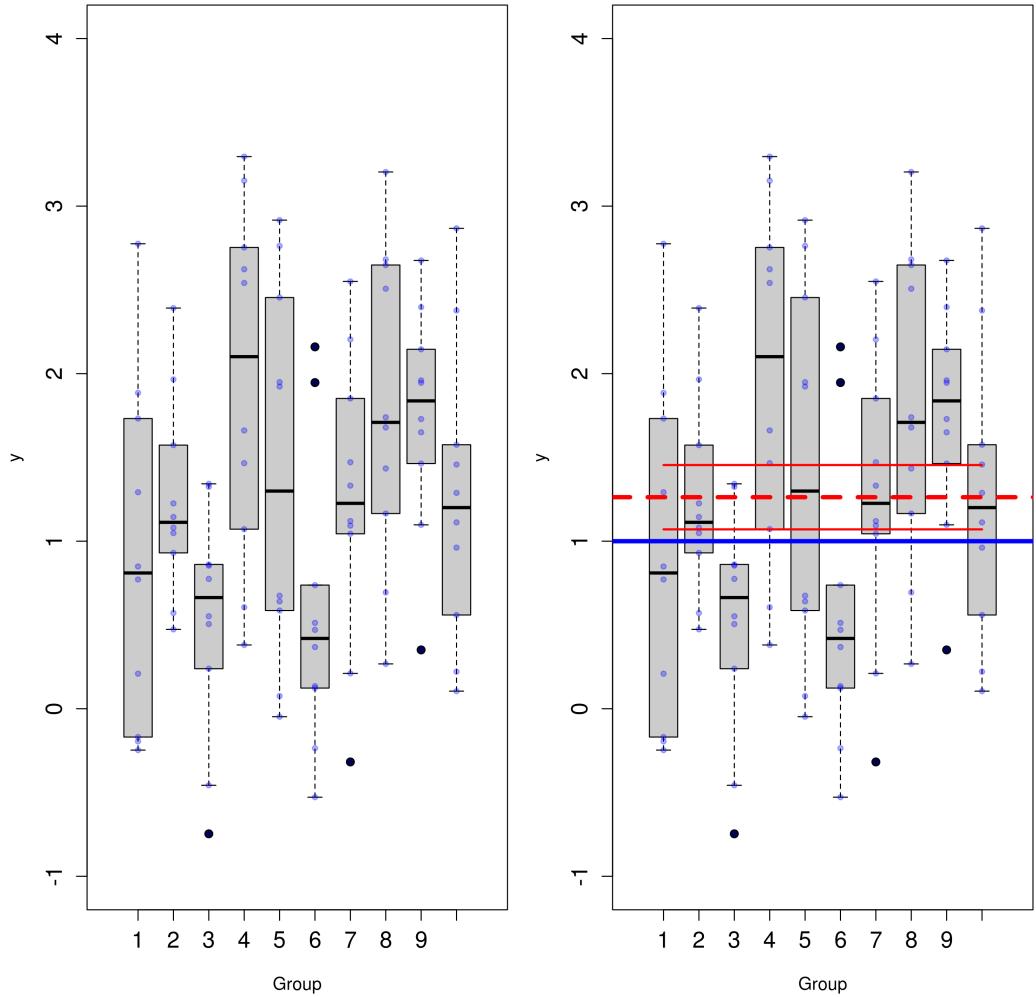


Figure 5.10: 100 observations clustered within 10 observational units (groups). Each group has a random intercept. Observations within each group vary randomly around that intercept. The plot on the right shows the complete pooled intercept and 95% confidence intervals for the intercept are shown in red. The actual population intercept is in blue, and in this sample lies outside the confidence interval.

Now, to take a real example, the complete pooling analysis just estimates one intercept for log radon levels in all Minnesota counties, ignoring the county-level structure:

```
radonreg <- lm(logradon ~ 1, data = MNradon)
```

The overall average log radon value in the sampled Minnesota houses is 1.55, or a geometric mean of $e^{1.55} - 1 = 3.71$ picoCuries/L on the original scale. This was shown earlier in our boxplots of log radon activity in each county:

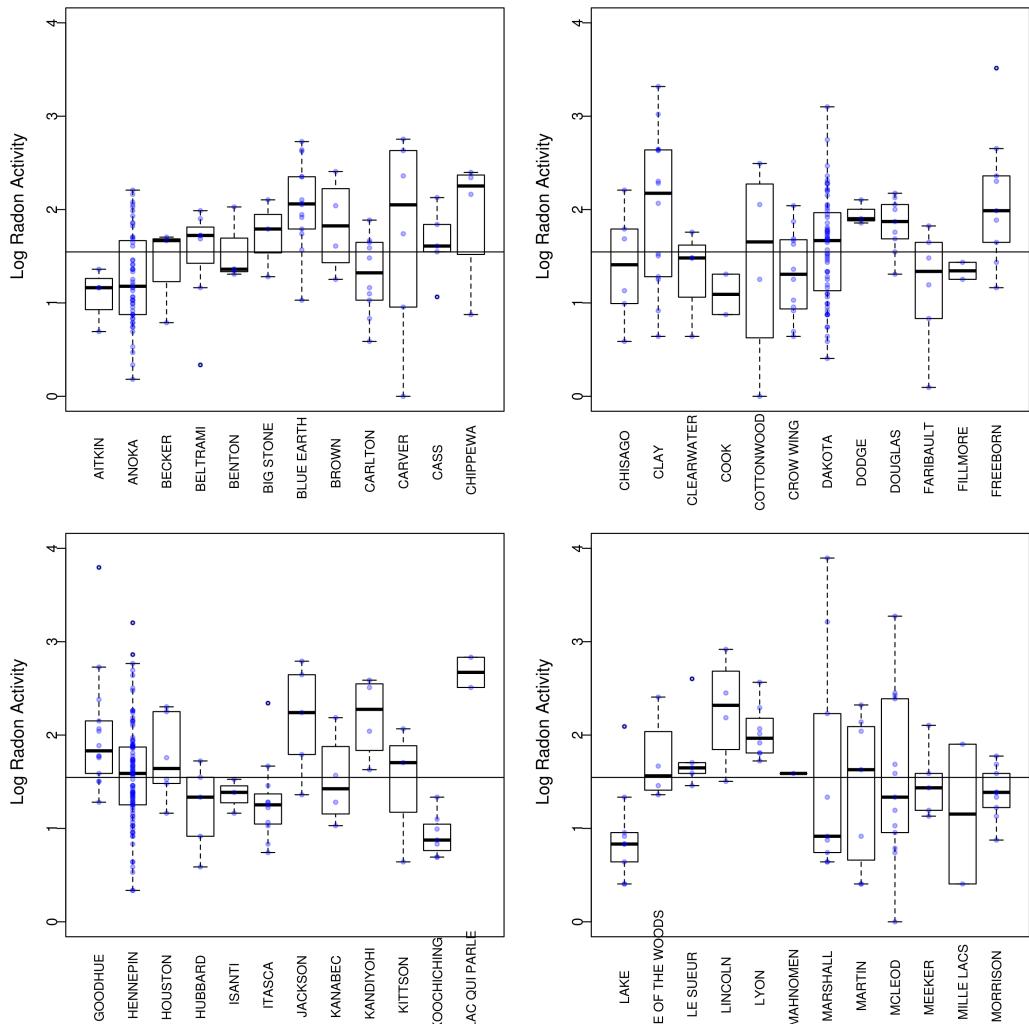


Figure 5.11: Boxplots of log radon activity in each county. Horizontal line is the complete pooling overall mean log radon activity, ignoring county information.

5.3.3 No Pooled Intercept

Now let's do the no-pooling analysis. To do this, we analyze each county seperately:

```
# Make an empty data frame to store
# the regression results for each county
no.pool <- data.frame(int      = rep(NA, 85),
                      std.error = rep(NA, 85),
                      county    = unique(MNradon$county),
                      sample.size = rep(NA, 85))

for(i in unique(MNradon$county)) {
  # Conduct the regression for data only in county i
  radonreg.np <- lm(logradon ~ 1, data = MNradon[MNradon$county == i, ])
  # Pull out the no pooling intercept
  no.pool$int[no.pool$county == i] <- coef(radonreg.np)
  # Pull out the no pooling standard error for the intercept
  no.pool$std.error[no.pool$county == i] <- coef(summary(radonreg.np))[1, 2]
  # Pull out the sample size for that county
  no.pool$sample.size[no.pool$county == i] <- nrow(MNradon[MNradon$county == i, ])
}
```

The parameter estimates in the no pooling regression are the same as if we use a dummy variable for each county in the regression:

```
radonreg2 <- lm(logradon ~ factor(county) - 1, data = MNradon)
summary(radonreg2)
no.pool
```

There are some slight but revealing differences between no pooling regression run separately versus using dummy coded counties. The intercepts are the same using the two methods. However, the standard errors are different. When using separate regressions, we cannot estimate a standard error for counties with only one observation (e.g., Mahnomen). In contrast, using the dummy coded regression, we have an overall error term (the rMSE of the regression) which fills in this gap, and otherwise accounts for the difference between the standard errors.

Notice that there are more extreme values and larger variation among the intercepts for counties with fewer observations. For example, Lac Qui Parle county has the highest measured radon levels but only two observations (measurements) in that county:

```
png(filename = "nopoolplot.png", width = 3000, height = 3000, res = 300)
```

```

par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(no.pool$sample.size, no.pool$int,
      pch = 19,
      col = rgb(0, 0, 0, .5),
      ylim = c(0, 3),
      ylab = "Sample Mean of Log Radon Activity in each County",
      xlab = "Sample Size in County",
      # Plot the sample size on the log scale
      log = "x")
abline(h = median(MNradon$logradon))
# Plot the standard errors for each county's mean logradon levels
for(j in no.pool$county){
  # Make a line that connects the standard error endpoints
  lines(
    # The two x-values are just the sample size for county j
    rep(no.pool$sample.size[no.pool$county == j], 2),
    # The two y-values are the point estimate
    no.pool$int[no.pool$county == j] +
    # Plus or minus one standard error
    c(-1, 1)*no.pool$std.error[no.pool$county == j],
    lwd = 0.5)
}
# Circle the Lac Qui Parle observation
points(no.pool$sample.size[36], no.pool$int[36], cex = 4)
dev.off()

```

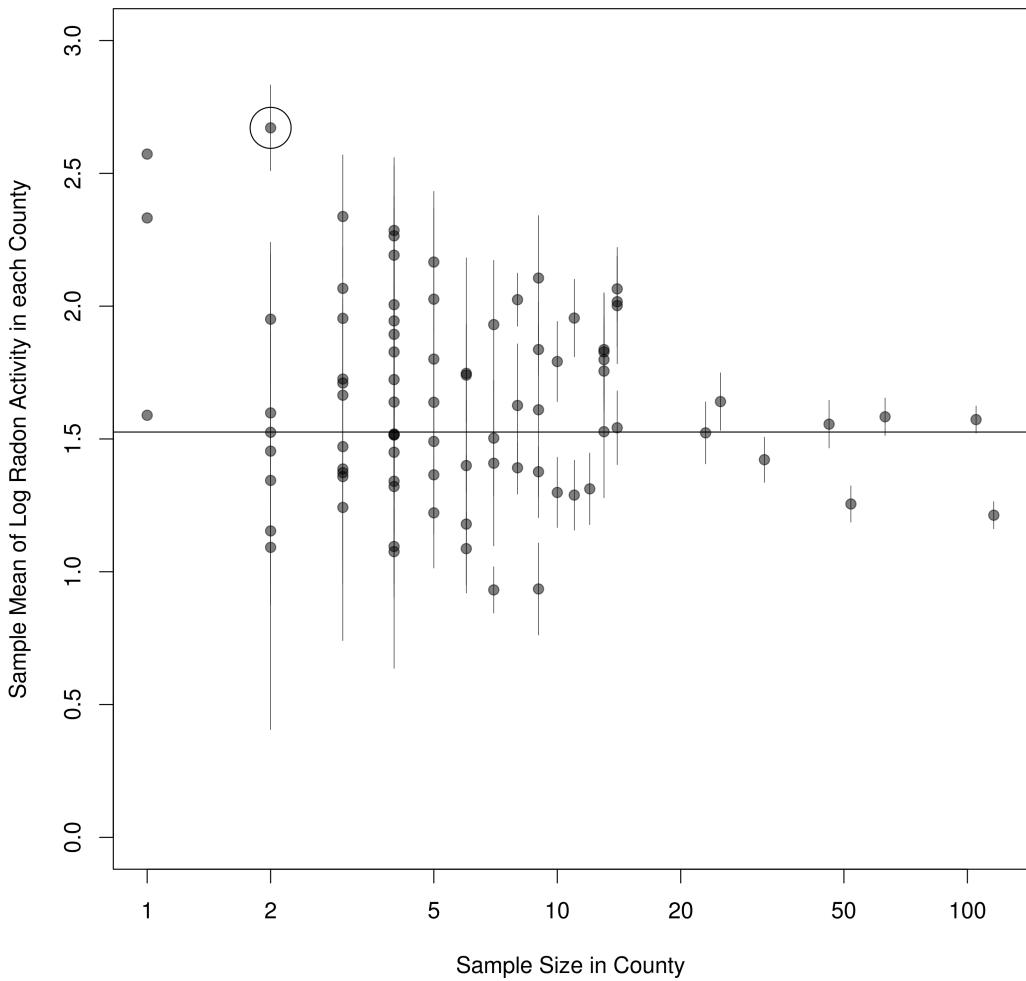


Figure 5.12: Plot of mean log radon activity in each county (y-axis) against the sample size in each county (x-axis), with ± 1 standard error bars shown. Lac Qui Parle county is circled.

Lac Qui Parle county seems to have abnormally high log radon levels, and these levels are estimated with high precision (low standard errors) relative to other counties with the same sample size. The two observations in this county were both high and very close to each other (2.51 and 2.83 log(picoCuries/L)).

Should we trust this result, indicating that Lac Qui Parle may have dangerous radon levels? Probably not, because we shouldn't expect counties with very small sample sizes to be as informative as those with large sample sizes. Our no pooling analysis will tend to put too much trust in counties with few observations, overstating the county-level variation. On the other hand, complete pooling treats all observations identically, thus understating

county-level variation.

5.3.4 Partially Pooled Intercept

The intuition behind partial pooling is that we can't trust the data from counties with very few observations too much, so we should move them closer to the overall average that ignores counties (the complete pooling estimate). On the other hand, for counties with many observations we should trust the estimate a lot, so we won't adjust their estimates much. Thus, partial pooling gives an estimated average $\hat{\alpha}_j$ for county j that shifts the observed county average \bar{y}_j for county j (no pooling) toward the observed average for all observations \bar{y}_{all} (complete pooling). Alternatively, we can think of \bar{y}_{all} as the mean of the counties μ_α :

$$\hat{\alpha}_j \approx \frac{\frac{n_j}{\sigma_y^2} \bar{y}_j + \frac{1}{\sigma_\alpha^2} \bar{y}_{all}}{\frac{n_j}{\sigma_y^2} + \frac{1}{\sigma_\alpha^2}} \quad (5.7)$$

This captures the weighted average intuition from before. We can see that as the sample size n_j of county j increases, more weight is placed on the measured county average \bar{y}_j when estimating the partially pooled average $\hat{\alpha}_j$ for county j . That is, the larger the sample size in the county, the closer the partially pooled mean $\hat{\alpha}_j$ for county j is to the county's unpooled mean \bar{y}_j .

This partial pooling effect is scaled by the variance of the within county data σ_y^2 , which is equivalent to the residual variance (rMSE squared) of the no pooling regression with dummy variables for counties. Thus, σ_y^2 will be low when there is little variability among observations once we account for the county-level means. As σ_y^2 gets smaller, that is, there is little variation within counties after accounting for their county-level intercept, then the intercepts are shifted less toward the complete pooling estimate \bar{y}_{all} .

The weighting is also determined by the variance of the means between counties σ_α^2 , that is, how much the county averages vary between each other. If this variance is very small the county-level averages will be close to the overall average, then we can just ignore county information and rely on \bar{y}_{all} . If it is very large, then we need not bother pooling the data at all because the county averages are very dissimilar, and the counties are probably not comparable.

From our simulation before, σ_y^2 is the variance of e.y of our dependent variable y , while σ_α^2 is the variance of alpha.j. Let's systematically change σ_y^2 and σ_α^2 to see how this changes our simulation:

```
beta0 <- 1
group <- rep(seq(1, 5, by = 1), 200)
set.seed(1)
```

```

# Set sigma.alpha's
sigma1.alpha <- 1
sigma2.alpha <- 2
# Draw intercepts from N(0, sigma.alpha)
alpha1.j <- rnorm(10, 0, sigma1.alpha)
alpha2.j <- rnorm(10, 0, sigma2.alpha)
# Set sigma.y's
sigma1.y <- 1
sigma2.y <- 2
# Draw errors from N(0, sigma.y)
e.y1 <- rnorm(1000, 0, sigma1.y)
e.y2 <- rnorm(1000, 0, sigma2.y)

# Create four different dependent variables
# Differeng in the size of sigma.alpha and sigma.y
y11 <- beta0 + alpha1.j[group] + e.y1
y12 <- beta0 + alpha1.j[group] + e.y2
y21 <- beta0 + alpha2.j[group] + e.y1
y22 <- beta0 + alpha2.j[group] + e.y2

data11 <- data.frame(x = x, y = y11, group = group)
data12 <- data.frame(x = x, y = y12, group = group)
data21 <- data.frame(x = x, y = y21, group = group)
data22 <- data.frame(x = x, y = y22, group = group)

```

Plotting the simulation:

```

png(filename = "alphayexample.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(2, 2), mar = c(5, 4, 2, 1))
plot(as.factor(group), y11,
     xlim = c(0, 6),
     pch = 19,
     col = rgb(0, 0, 0, .2),
     ylab = "y",
     xlab = "Group",
     ylim = c(-10, 10))
points(group, y11,
       pch = 20,
       col = rgb(0, 0, 1, .3))

plot(as.factor(group), y12,
     xlim = c(0, 6),
     pch = 19,

```

```

col  = rgb(0, 0, 0, .2),
ylab = "y",
xlab = "Group",
ylim = c(-10, 10))
points(group, y12,
       pch = 20,
       col = rgb(0, 0, 1, .3))

plot(as.factor(group), y21,
      pch = 19,
      xlim = c(0, 6),
      col = rgb(0, 0, 0, .2),
      ylab = "y",
      xlab = "Group",
      ylim = c(-10, 10))
points(group, y21,
       pch = 20,
       col = rgb(0, 0, 1, .3))

plot(as.factor(group), y22,
      pch = 19,
      xlim = c(0, 6),
      col = rgb(0, 0, 0, .2),
      ylab = "y",
      xlab = "Group",
      ylim = c(-10, 10))
points(group, y22,
       pch = 20,
       col = rgb(0, 0, 1, .3))
dev.off()

```

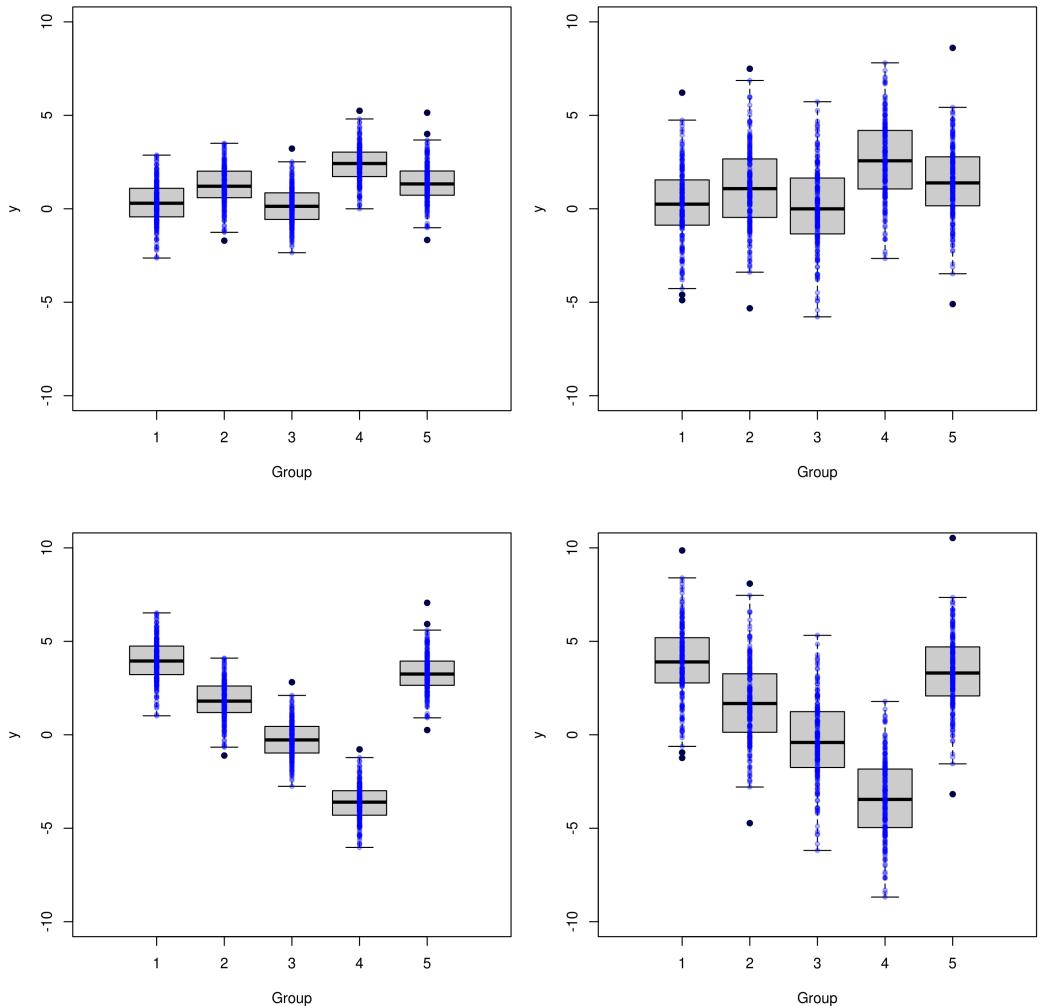


Figure 5.13: Top plots have low intercept variance $\sigma_\alpha^2 = 1$, while bottom plots have high intercept variance $\sigma_\alpha^2 = 4$. Plots on the left have low within-cluster variance $\sigma_y^2 = 1$, while plots on the right have high within-cluster variance $\sigma_y^2 = 4$.

Recapping, partial pooling will pull estimates closer to the complete pooling estimate \bar{y}_{all} when the sample size n_j in an observational unit (county) is small, when the variation within counties is large after accounting for the county mean σ_y^2 , and when the variation between county means σ_α^2 is small. We can summarize this in the following table:

Sample size n_j in group j	Estimate $\hat{\alpha}_j$
$n_j = 0$	$\hat{\alpha}_j = \bar{y}_{all}$ (complete pooling)
$n_j < \frac{\sigma_y^2}{\sigma_\alpha^2}$	$\hat{\alpha}_j$ closer to \bar{y}_{all}
$n_j = \frac{\sigma_y^2}{\sigma_\alpha^2}$	$\hat{\alpha}_j = \frac{1}{2}\bar{y}_{all} + \frac{1}{2}\bar{y}_j$
$n_j > \frac{\sigma_y^2}{\sigma_\alpha^2}$	$\hat{\alpha}_j$ closer to \bar{y}_j
$n_j = \infty$	$\hat{\alpha}_j = \bar{y}_j$ (no pooling)

Table 5.1: Degree of pooling as a function of sample size, within-group variance σ_y^2 and between group variance σ_α^2 .

To find the partially pooled intercepts, we need to estimate the within-county and between-county variance parameters, as they critically determine the amount of pooling. However, the variance estimates are themselves dependent on the partially pooled estimates of the county-level averages. The *lmer* function in R uses penalized maximum likelihood estimation to estimate the model parameters and variance components:¹

```
install.packages("arm", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(arm)
# (1|county) indicates that we should estimate partially pooled
# county-level intercepts
radonreg3 <- lmer(logradon ~ 1 + (1|county), data = MNradon)
summary(radonreg3)
```

Here we get an overall log-radon average of 1.62, similar to our complete pooling analysis of 1.55. The fitted model for a county j is:

$$\text{Log-Radon}_j = \alpha_j = 1.62 + u_j \quad (5.8)$$

Here u_j is the deviation from the overall mean of the α 's (i.e., $\mu_\alpha = 1.62$) for county j . For example, we can get $u_j = -0.21$ for county 1 (Aitkin):

```
# ranef gives a list, so you need to turn it into a matrix
as.matrix(ranef(radonreg3)$county)[1]
```

¹lmer stands for linear mixed effects in R. However, we can use lmer with any stochastic components and link functions from the generalized linear model (e.g., Binomial, Poisson), that is, this is an extension of the generalized linear model.

Giving us a county-level mean of:

$$\text{Log-Radon}_j = \alpha_j = 1.62 - 0.21 = 1.41 \quad (5.9)$$

Alternatively, we can get the county-level mean directly using the `coef()` function:

```
as.matrix(coef(radonreg3)$county) [1]
```

We can see the largest county-level deviations from the overall average by examining the deviations:

```
# Get the maximum and minimum county-level deviations
max(ranef(radonreg3)$county)
min(ranef(radonreg3)$county)
```

The largest fitted county mean log-radon activity is 0.31 above the overall average, giving us 1.93 ($= 1.62 + 0.31$). The lowest is 0.41 below the overall average, giving us 1.21 ($= 1.62 - 0.41$). How does this compare to the no pooling analysis? Let's find the average, largest, and smallest log-radon values from our no pooling regression:

```
mean(no.pool$int)
max(no.pool$int)
min(no.pool$int)
```

The average log-radon value across the different counties is 1.64, very similar to our partial pooling analysis of 1.62. However, the minimum is 0.93, and maximum 2.67. These are much smaller and larger (more variable), than the partial pooling analysis, indicating that these counties had small numbers of observations, as they were pooled toward the overall mean of 1.61 (more precisely, for these counties $n_j < \frac{\sigma_y^2}{\sigma_\alpha^2}$). The partial pooling approach moved the estimated intercept of these counties with few measurements closer to the overall complete pooling average.

5.4 Pooling with Predictors

We can also do complete, no, and partial pooling in a model with predictors. Using the radon data, we can add a predictor for where the radon measurement was taken, either in the basement of the house ($= 0$), or the first floor ($= 1$).

```

# Complete pooling regression
radonreg4.cp <- lm(logradon ~ floor, data = MNradon)
# No pooling dummy regression
radonreg4.np <- lm(logradon ~ floor + factor(county) - 1, data = MNradon)
# Partial pooling regression
radonreg4.pp <- lmer(logradon ~ floor + (1|county), data = MNradon)
# Grab data for the first 6 counties for plotting
plot1 <- MNradon[MNradon$county %in% unique(MNradon$county)[1: 6], ]
plot1 <- droplevels(plot1)

j <- 1
png(filename = "radonfloor.png", width = 3000, height = 3000, res = 300)
par(cex = 2, mfrow = c(2, 3), mar = c(5, 5, 2, 1),
     cex.axis = 1.5, cex.lab = 1.5)
# For each unique county ID
for(i in unique(plot1$county)){
  # Plot the logradon measurements by floor of measurement
  plot(jitter(plot1$floor[plot1$county == i], amount = 0.05),
       plot1$logradon[plot1$county == i],
       ylim = c(-1, 3),
       pch = 19,
       col = rgb(0, 0, 0, .5),
       xaxt = "n",
       xlab = "Floor",
       ylab = "Log Radon Levels",
       xlim = c(-0.1, 1.1),
       main = i,
       cex = 2)
  axis(side = 1, at = c(0, 1),
        labels = c("Basement", "First Floor"),
        cex = 2)
  # Add the complete pooling regression line
  abline(coef(radonreg4.cp)[1], coef(radonreg4.cp)[2],
         lty = 2,
         lwd = 2)
  # Add the no pooling intercept and slope
  # The slope is the first coefficient
  # so we need to advance j by one
  abline(coef(radonreg4.np)[j + 1],
         coef(radonreg4.np)[1],
         lty = 1,
         lwd = 2)
}

```

```
j <- j + 1  
}  
dev.off()
```

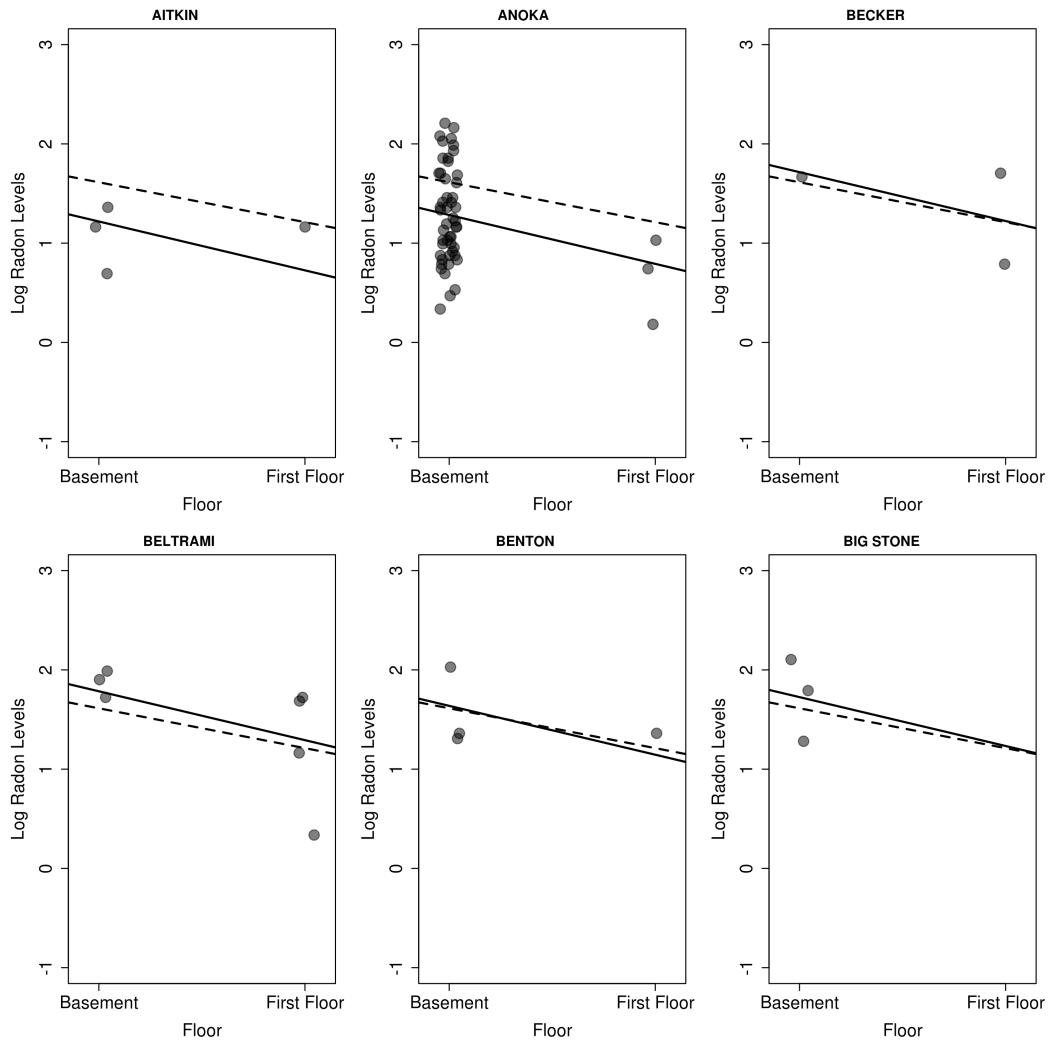


Figure 5.14: Plot of log radon activity in six counties by whether the measurements were taken in the basement or first floor of the house (x-axis). Complete pooling regression is shown on the dashed line, whereas the no pooling regression line, with different intercepts for each county, is shown in the solid line.

The estimate of the slope of log-radon on floor is similar across the three models:

```
coef(radonreg4.cp)["floor"]
coef(radonreg4.np)["floor"]
fixef(radonreg4.pp)["floor"]
```

The slope is steeper in the no pooling regression $\beta = -0.49$ than the complete pooling regression $\beta = -0.40$, indicating that there's some

correlation between the county-level radon levels (intercepts) and whether measurements in homes in that county were taken in basements or the first floor (i.e., there's county-level variation in whether houses have basements).

It's important to note here that with a regressor x , multi-level models that use "random" effects for the intercepts (i.e., varying intercepts) do *not* allow the regressors x and varying intercepts (or any other varying parameters) to be correlated. Thus, a large difference between estimates from the no pooling regression with dummy variables for each county, which takes any correlation among the regressors x and the intercepts into account, and the multi-level model, might indicate that we need to allow for a correlation between the varying parameters and the regressors. We'll explore this more later, but this is why economists rarely use random effects, because they are usually interested in trying to "control" for county-level variation to get unbiased estimates of the coefficient on x , and not particularly interested in understanding group-level variation for its own sake.

Next, we can also see from the plot that, while it seems reasonable for the intercept for Anoka to deviate from the complete pooling intercept because Anoka has a relatively large sample size, this is not the case for Aitkin, which has very few observations and a large deviation of its intercept from the complete pooling intercept. To see how the partial pooling approach handles this, let's add the partial pooling regression lines to the plot:

```
j <- 1
ranef.radon <- unlist(ranef(radonreg4.pp))
png(filename = "radonplots3.png", width = 3000, height = 3000, res = 300)
par(cex = 2, mfrow = c(2, 3), mar = c(5, 5, 2, 1),
    cex.axis = 1.5, cex.lab = 1.5)
for(i in unique(plot1$county)){
  plot(jitter(plot1$floor[plot1$county == i], amount = 0.05),
       plot1$logradon[plot1$county == i],
       ylim = c(-1, 3),
       pch = 19,
       col = rgb(0, 0, 0, .5),
       xaxt = "n",
       xlab = "Floor",
       ylab = "Log Radon Levels",
       xlim = c(-0.1, 1.1),
       main = i,
       cex = 2)
  axis(side = 1,
       at = c(0, 1),
       labels = c("Basement", "First Floor"),
       cex = 2)}
```

```
abline(coef(radonreg4.cp)[1] ,
       coef(radonreg4.cp)[2] ,
       lty = 2,
       lwd = 2)
abline(coef(radonreg4.np)[j + 1] ,
       coef(radonreg4.np)[1] ,
       lty = 1,
       lwd = 2)
abline(fixef(radonreg4.pp)[1] + ranef.radon[j] ,
       fixef(radonreg4.pp)[2] ,
       lty = 3,
       lwd = 2,
       col = "red")
j <- j + 1
}
dev.off()
```

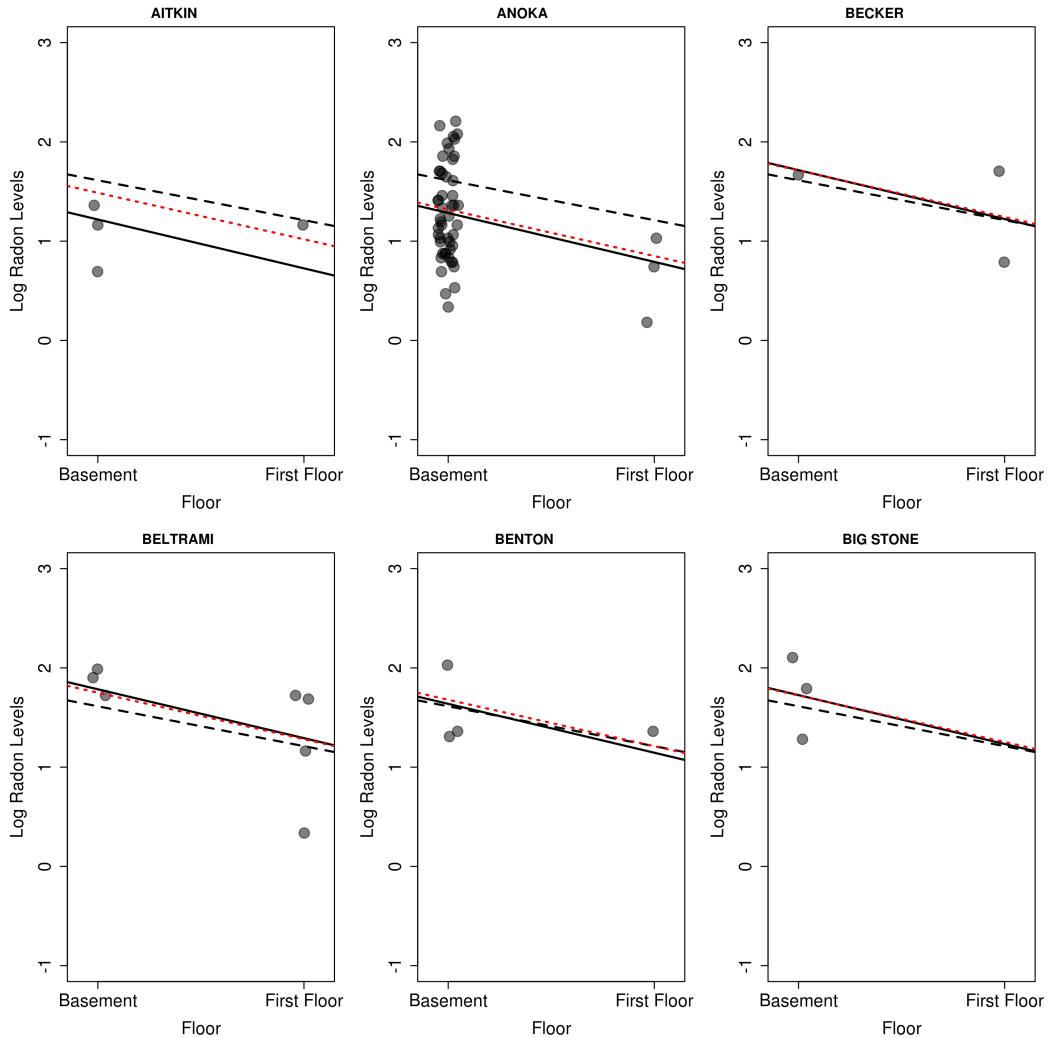


Figure 5.15: Plot of log radon activity in six counties by whether the measurements were taken in the basement or first floor of the house (x-axis). Complete pooling regression is shown on the dashed line. No pooling regression line, with different intercepts for each county, is shown in the solid line. Partial pooling line, with varying intercepts by county, is shown in the red dotted line.

The partial pooling line shows how the no pooling intercepts (solid lines) are pulled toward the complete pooling intercepts (dashed lines) more for the counties with few observations (e.g., Aitkin) than counties with many observations (e.g., Anoka). The complete pooling analysis forces a single intercept α for all counties. The no pooling analysis fits a separate intercept for each county, α_j . The partial pooling analysis fits intercepts for each county, α_j , that are between the complete and no pooling analyses. From the partial pooling model, our estimate of $\frac{\sigma_y^2}{\sigma_\alpha^2}$ is 4.7, indicating that a county

would need to have more than 5 observations to have the partially pooled intercept be closer to the no pooling intercept than complete pooling intercept. Because Aitkin has only 4 observations, the intercept for Aitkin is closer to the no pooling than complete pooling. On the other hand, Anoka has many more than 5 observations, and the partially pooled intercept is almost identical to the no pooling intercept.

This tradeoff is formalized by assigning the county intercepts a probability distribution themselves:

$$\alpha_j \sim N(\mu_\alpha, \sigma_\alpha^2) \quad (5.10)$$

Thus, the intercept for county j is drawn from a Normal distribution with mean equal to μ_α and variance σ_α^2 . As σ_α^2 grows larger partial pooling does nothing, as the intercept could be anywhere. As σ_α^2 goes to zero, the intercept goes to μ_α , as there is no variation in intercepts across counties. With one predictor variable, the average regression line for all counties is:

$$y_i = 1.72 - 0.47 \times \text{floor}_i$$

For a specific house i in county j , the regression line is:

$$y_{ij} = \alpha_j - 0.47 \times \text{floor}_i$$

where $\alpha_j = 1.72 + u_j$, meaning u_j is the deviation of the county mean from the mean of the counties. For example, for a house in Aitkin county (county 1), the random effect deviation is $u_j = -.23$:

```
as.matrix(ranef(radonreg4.pp)$county)[1]
```

Giving us a regression line for Aitkin county of:

$$\begin{aligned} \text{Log-Radon}_{ij} &= 1.72 - 0.23 - 0.47 \times \text{floor}_i \\ &= 1.49 - 0.47 \times \text{floor}_i \end{aligned}$$

Alternatively, we can get the county-level regression line directly:

```
as.matrix(coef(radonreg4.pp)$county)[1, ]
```

If we use the `summary()` function, we can get information about the variance components of the partial pooling regression model.

```
summary(radonreg4.pp)
```

The county-level intercept variance σ_α^2 is in the “Random effects” table under the “Group” labeled county and “Name” labeled “(Intercept)” in the “Variance” column. The residual variance σ_y^2 is similarly labeled Residual in the “Groups” column.

From this summary, we can see that at the individual level the error variance is $\sigma_y^2 = 0.30$ and at the county level the error variance is $\sigma_\alpha^2 = 0.06$. To see how much information we are getting from individual versus county-level data, we can look at the ratio of the variance of log-radon levels between counties to the average within-county variation $\frac{\sigma_\alpha^2}{\sigma_y^2}$. From our regression, this is $0.06/0.30 = 0.20$. Thus, we'd need five times the observations within a county to predict better than the information provided by the counties overall. This number of measurements is the cutoff where partial pooling moves closer to the complete pooling estimate (if the number of measurements is fewer), or no pooling estimate (if the number of measurements is greater). We can pull out standard errors for the intercept of different counties, plotting standard errors against sample size:

```
# In this case, if you don't define rownames this way
# things will get messed up in terms of alphabetization
# because of the way factors with spaces in their names are treated
rownames <- rownames(ranef(radonreg4.pp)$county)
p.pool <- data.frame(int = rep(NA, 85),
                      std.error = rep(NA, 85),
                      county = rownames,
                      sample.size = rep(NA, 85))
p.pool$int <- fixef(radonreg4.pp)[1] + unlist(ranef(radonreg4.pp))
p.pool$std.error <- unlist(se.ranef(radonreg4.pp))

for(i in rownames){
  p.pool$sample.size[p.pool$county == i] <- nrow(MNradon[MNradon$county == i, ])
}

png(filename = "ppoolplot.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3)
plot(p.pool$sample.size,
      p.pool$int,
      pch = 20,
      col = rgb(0, 0, 0, .5),
      ylim = c(0, 3),
      ylab = "Partially Pooled Intercept by County",
      xlab = "Sample Size in County",
      log = "x")
abline(fixef(radonreg4.pp)[1], 0)

for(j in p.pool$county){
  lines(rep(p.pool$sample.size[p.pool$county == j], 2),
        p.pool$int[p.pool$county == j]
```

```
+ c(-1, 1)*p.pool$std.error[p.pool$county == j],  
lwd = 0.5)  
}  
points(p.pool$sample.size[36], p.pool$int[36], cex = 4)  
dev.off()
```

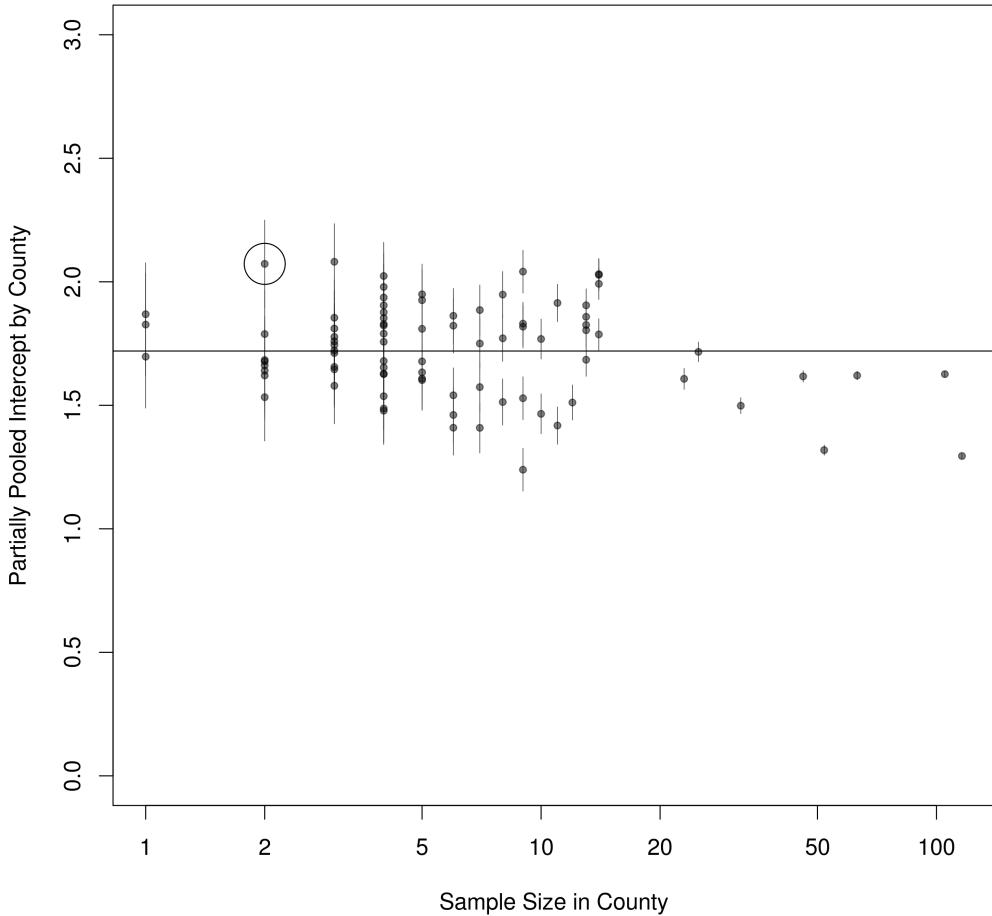


Figure 5.16: Plot of partially pooled intercept estimate of log radon activity in each county (y-axis) against the sample size in each county (x-axis), with ± 1 standard error bars shown. Lac Qui Parle county is circled.

The partially pooled intercepts with a predictor variable is different from when there is no predictor, but the same logic applies:

$$\hat{\alpha}_j \approx \frac{\frac{n_j}{\sigma_y^2}}{\frac{n_j}{\sigma_y^2} + \frac{1}{\sigma_\alpha^2}} \hat{\beta}_{0j} + \frac{\frac{1}{\sigma_\alpha^2}}{\frac{n_j}{\sigma_y^2} + \frac{1}{\sigma_\alpha^2}} \mu_\alpha \quad (5.11)$$

Here, μ_α is the mean of the varying intercept model $\alpha_j \sim N(\mu_\alpha, \sigma_\alpha^2)$ and $\hat{\beta}_{0j}$ is the intercept for group j from the no pooling model. This matches the linear least squares estimate of the intercept, which we derived in Chapter 3, which is $\beta_{0j} = \bar{y}_j - \beta_1 \bar{x}_j$, where \bar{y}_j is the average of the observations in group j , and \bar{x}_j is the average values of the predictor variable x in group j . Thus,

the partially pooled intercept with a predictor is a compromise between the no pooling intercept for group j and the average of the varying intercepts μ_α , where the compromise (degree of pooling) is smaller as the sample size n_j in group j is larger, the within-group variance σ_y^2 is smaller, and the between group variance σ_α^2 is smaller.

5.5 Group-Level Predictors

Next, let's look at group-level predictors, such as measured uranium in the county's soil in parts per million. First we need to retrieve the county-level data:

```
cty.lvl <- read.table(
  "http://www.stat.columbia.edu/~gelman/arm/examples/radon/cty.dat",
  na.strings = "", header = T, sep = ",")
# Get county level data for only Minnesota
# Only include county (cty), uranium (Uppm), and county id (ctfips)
uraniumMN <- cty.lvl[cty.lvl$st == "MN", c("cty", "Uppm", "ctfips")]
# Drop unused levels
uraniumMN <- droplevels(uraniumMN)
```

Let's translate the county-level uranium measurements into a uranium variable in our original data frame, then plot county-level log radon measurements by county-level log soil uranium measurements:

```
for(i in 1:nrow(MNradon)){
  # For each row in MNradon, find match the county id in the county-level data
  # with the county id in the household level data, then pull out the uranium
  MNradon$uranium[i] <- uraniumMN$Uppm[uraniumMN$ctfips == MNradon$cntyfips[i]]
}

# Get the no pooled county-level log-radon means
county.logradon <- c()
for(i in unique(MNradon$county)){
  county.logradon[i] <- mean(MNradon$logradon[MNradon$county == i])
}

# Regress county-level log radon on county-level log uranium
uranium1 <- lm(county.logradon ~ log(unique(MNradon$uranium)))

png(filename = "countyuranium.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(log(unique(MNradon$uranium)),
```

```
county.logradon,
pch  = 19,
col  = rgb(0, 0, 0, .5),
xlim = c(-1, 1),
ylim = c(0, 3),
xlab = "County-Level Log Uranium",
ylab = "County-Level Average Log Radon")
abline(uraniun1, lty = 2, lwd = 2, col = "red")
dev.off()
```

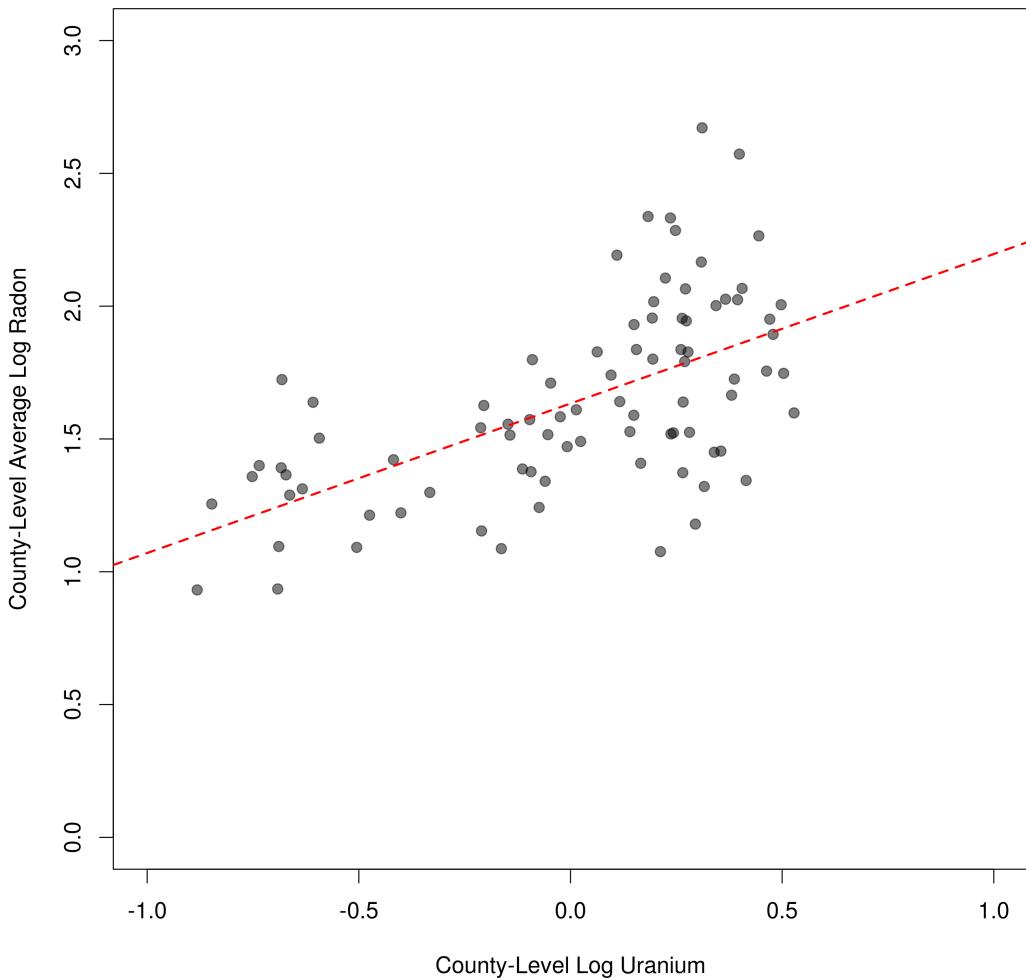


Figure 5.17: County-level log radon measurements plotted against county-level log uranium measurements.

There seems to be a roughly linear relationship between log radon levels and log uranium at the county level. At this point, we cannot go further without using a multi-level model, because we have county-level variables that will be perfectly correlated with the county-level intercept if we use classical regression. For example, uranium will be dropped from the following no-pooling regression because it is perfectly collinear with county-level intercepts:

```
# No pooling regression cannot include group-level predictors
MNradon$loguranium <- log(MNradon$uranium)
lm(logradon ~ factor(county) - 1 + loguranium, data = MNradon)
```

Notice the “NA” for the uranium predictor. On the other hand, the multi-level model does not have this problem because the intercepts are not directly modeled in the regression, instead their distribution is modeled. To include log-uranium as a predictor, we just add it to the regression:

```
# Add log uranium as a predictor
lmer.u <- lmer(logradon ~ floor + loguranium + (1|county), data = MNradon)
```

The model with a group level predictor has an overall average regression line of:

$$\text{Log-Radon}_{ij} = 1.72 - 0.44 \times \text{floor}_i + 0.57 \times \text{loguranium}_j$$

For a specific county j we need to get α_j , which is 1.72 plus the random deviation u_j for county j :

$$\text{Log-Radon}_{ij} = 1.72 + u_j - 0.44 \times \text{floor}_i + 0.57 \times \text{loguranium}_j$$

So, our regression line for Aitkin county, with log uranium levels of -0.689 and a random effect $u_j = -0.017$ is:

$$\begin{aligned}\text{Log-Radon}_{ij} &= 1.72 - 0.017 - 0.44 \times \text{floor}_i - 0.689 \times 0.57 \\ &= 1.31 - 0.44 \times \text{floor}_i\end{aligned}$$

As can be seen, adding a group level predictor has the effect of changing the *intercept* for each county, depending on the log-uranium levels in the county. Let's plot the partially pooled intercepts and their relationship to county-level log uranium levels:

```
rownames <- rownames(ranef(lmer.u)$county)
loguranium <- c()
for(i in rownames){
  loguranium[i] <- mean(MNradon$loguranium[MNradon$county == i])
}
u.pool <- data.frame(int = rep(NA,85),
                      std.error = rep(NA,85),
                      county = rownames,
                      loguranium = loguranium)

# The partially pooled intercept is the overall intercept (fixef)
u.pool$int <- fixef(lmer.u)[1]
# Plus the random intercept (ranef(lmer.u))
u.pool$int <- u.pool$int + unlist(ranef(lmer.u))
```

```

# Plus the prediction based on log uranium levels at the county
u.pool$int <- u.pool$int + fixef(lmer.u)[3]*u.pool$loguranium
# Get the standard error of the partially pooled intercept
u.pool$std.error <- unlist(se.ranef(lmer.u))

png(filename = "lmeru.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
plot(u.pool$loguranium,
      u.pool$int,
      pch = 20,
      col = rgb(0, 0, 0, .5),
      ylab = "Partially Pooled Log-Radon Intercept by County",
      xlab = "County-Level Log Uranium",
      ylim = c(1, 2.5))
abline(fixef(lmer.u)[1], fixef(lmer.u)[3])

for(j in u.pool$county){
  lines(rep(u.pool$loguranium[u.pool$county == j], 2),
        u.pool$int[u.pool$county==j] +
        c(-1, 1)*u.pool$std.error[u.pool$county == j],
        lwd = 0.5)
}
dev.off()

```

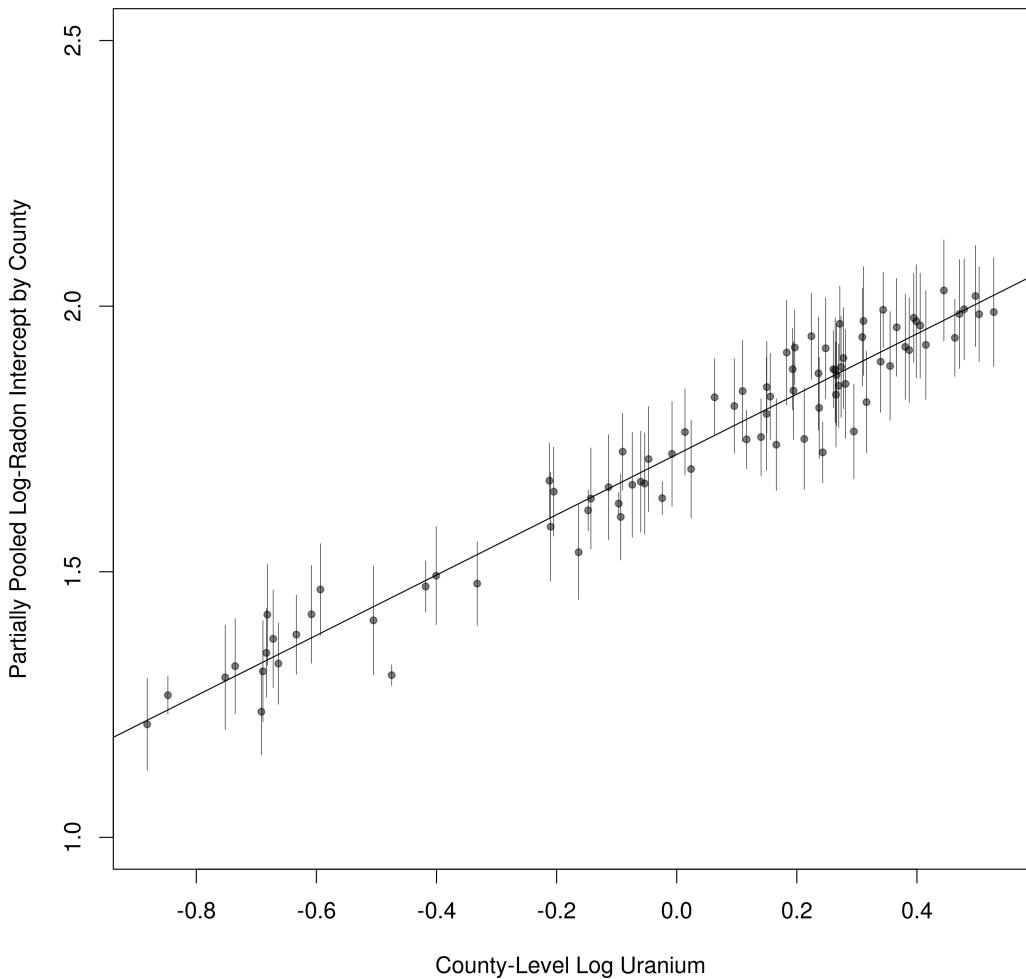


Figure 5.18: Partially pooled log radon intercept for each county from the multi-level model plotted against county-level log uranium in the soil.

We can see that the county-level predictor helped us drastically reduce the variability among counties in log-radon levels, in that their deviation from the log-uranium regression line is much smaller than their total variation. Recall that the varying-intercept model with no group-level predictor was:

$$\alpha_j \sim N(\mu_\alpha, \sigma_\alpha^2) \quad (5.12)$$

With a group-level predictor, we are basically regressing μ_α on group-level variables:

$$\alpha_j \sim N(\gamma_0 + \gamma_1 z_j, \sigma_\alpha^2) \quad (5.13)$$

Where z_j is our group-level predictor. With this interpretation, we can see how multi-level modeling gets its name, as we have a “level 1” model for

logradon at the observation level, and a “level 2” model for logradon at the observational unit (group) level:

$$y_i \sim N(\alpha_{j[i]} + \beta x_i, \sigma_y^2) \quad (5.14)$$

$$\alpha_j \sim N(\gamma_0 + \gamma_1 z_j, \sigma_\alpha^2) \quad (5.15)$$

Each level of the model has its own error; σ_y^2 for level 1 and σ_α^2 for level 2. Thus, multi-level modeling allows us to model outcomes at multiple structural levels. We can see what the model looks like at each level:

```

plot1 <- MNradon[MNradon$county %in% unique(MNradon$county)[1:6], ]
plot1 <- droplevels(plot1)
# Get the log uranium levels for Aitkin county
uran <- MNradon$loguranium[MNradon$county == levels(MNradon$county)[1]] [1]
# Get the county level random effect for the intercept
ran <- unlist(ranef(lmer.u)$county)[1]
# Get the county-level intercept for Aitkin county
int <- as.numeric(fixef(lmer.u)[1] + fixef(lmer.u)[3]*uran + ran)

png(filename = "multilevelmodel.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(2, 1), mar = c(5, 4, 2, 1), cex.lab = 1.3)
i <- unique(plot1$county)[1]

plot(jitter(plot1$floor[plot1$county == i], amount = 0.05),
     plot1$logradon[plot1$county == i],
     ylim = c(-1, 3),
     pch = 19,
     col = rgb(0, 0, 0, .5),
     xaxt = "n",
     xlab = "Floor",
     ylab = "Log Radon Levels",
     xlim = c(-0.1, 1.1),
     main = i,
     cex = 2)
axis(side = 1, at = c(0, 1), labels = c("Basement", "First Floor"))
abline(coef(radonreg4.cp)[1], coef(radonreg4.cp)[2], lty = 2, lwd = 2)
abline(coef(radonreg4,np)[2], coef(radonreg4,np)[1], lty = 1, lwd = 2)
abline(int,
       fixef(lmer.u)[2],
       lty = 3, lwd = 2, col = "red")

plot(u.pool$loguranium,
      u.pool$int,

```

```
pch  = 20,
col  = rgb(0, 0, 0, .5),
ylab = "Log-Radon Intercept by County",
xlab = "County-Level Log Uranium",
ylim = c(1, 2.5)
abline(fixef(lmer.u)[1], fixef(lmer.u)[3])

for(j in u.pool$county){
  lines(rep(u.pool$loguranium[u.pool$county == j], 2),
        u.pool$int[u.pool$county==j] +
        c(-1, 1)*u.pool$std.error[u.pool$county == j],
        lwd = 0.5)
}
dev.off()
```

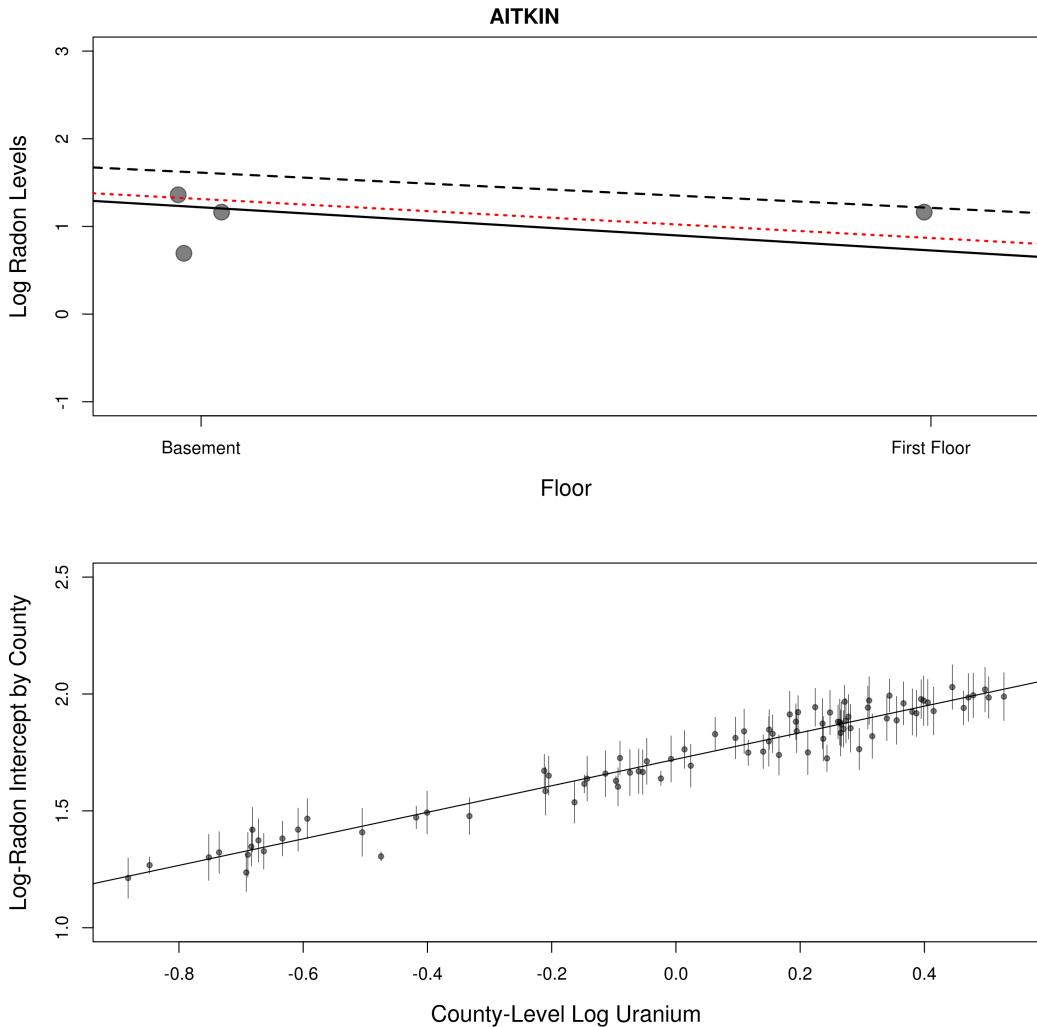


Figure 5.19: Two levels of a multi-level model. The level 1 regression (top) makes predictions about log radon levels at the level of individual observations. The level 2 regression (bottom) makes predictions about county-level log radon levels from county-level predictors.

The effect of the group-level predictor is striking in the radon example, indicating that we have a very strong level 2 model. The previous residual standard deviation of the county intercepts was 0.25 without county-level uranium, now it is about 0.11. With the group-level predictor the variance ratio is $\frac{\sigma_\alpha^2}{\sigma_y^2} = 0.11^2/0.55^2 = 0.033$, meaning the county-level model is as good as 30 ($= 1/0.033$) observations within any county.

Notice that the within-county residual standard error has not changed, as a county-level predictor cannot distinguish between observations within a county. Thus, we are good at modeling variation in log radon levels between

counties due to the strong relationship between county-level soil uranium content and radon activity, but not so good at modeling variation between households, due to the relatively weak relationship between household-level predictors and radon activity.

5.6 Varying Intercepts and Varying Slopes

We can also do partial pooling for predictor variables in the regression. The intuition is the same: The slope for each county is a tradeoff between the fitted slope for that county alone (no pooling) and the overall slope for all counties (complete pooling), with the weights determined by the county's sample size and the estimated variance parameters of the slopes. From the radon example, we can let the effect of floor of measurement vary by county, removing county-level uranium to keep it simple:

```
vary.slope <- lmer(logradon ~ floor + (1 + floor|county), data = MNradon)
summary(vary.slope)
```

The overall average regression line is:

$$\text{Log-Radon}_{ij} = 1.72 - 0.47 \times \text{floor}_i$$

For a specific county j we need to get α_j , which is 1.72 plus the random deviation u_j for county j , as well as β_j which is -0.47 plus the random slope deviation s_j :

$$\text{Log-Radon}_{ij} = 1.72 + u_j - 0.47 \times \text{floor}_i + s_j \times \text{floor}_i$$

So, our regression line for Aitkin county, with random intercept deviation $u_j = -0.286$ and random slope deviation $s_j = 0.147$ is:

$$\begin{aligned} \text{Log-Radon}_{ij} &= 1.72 - 0.286 - 0.47 \times \text{floor}_i + 0.147 \times \text{floor}_i \\ &= 1.43 - 0.32 \times \text{floor}_i \end{aligned}$$

We can see that adding a varying slope for the most part leaves our level 1 model the same, except β is now $\beta_{j[i]}$, but our level 2 model now has two components, with group-level models for both α_j and β_j :

$$y_i \sim N(\alpha_{j[i]} + \beta_{j[i]} x_i, \sigma_y^2) \quad (5.16)$$

$$\alpha_j \sim N(\mu_\alpha, (\sigma_\alpha^2, \rho_{\alpha\beta}\sigma_\alpha\sigma_\beta)) \quad (5.17)$$

$$\beta_j \sim N(\mu_\beta, (\rho_{\alpha\beta}\sigma_\alpha\sigma_\beta, \sigma_\beta^2)) \quad (5.18)$$

Notice that in addition to the variance of the intercepts σ_α^2 and variance in slopes σ_β^2 , we allow for a correlation $\rho_{\alpha\beta}$ between the county-level intercept and county-level slope. That is, counties with higher radon levels may also have higher (or lower) sensitivity to the floor of measurement. This correlation between varying intercepts and varying slopes is -0.55 for our model, and can be found by looking at the “Corr” column of the random effects table:

```
summary(vary.slope)
```

This indicates that counties with higher radon levels overall also have a larger difference in radon levels between houses where radon levels were measured in the basement and first floor, which is likely explained by the fact that higher radon levels are caused by uranium in the soil, which is better detected from the basement than the first floor.

We can also include the county-level uranium measurements as predictors of both county-level intercepts and slopes. This gives us the following multi-level model:

$$y_i \sim N(\alpha_{j[i]} + \beta_{j[i]}x_i, \sigma_y^2) \quad (5.19)$$

$$\alpha_j \sim N((\gamma_0^\alpha + \gamma_1^\alpha z_j), (\sigma_\alpha^2, \rho_{\alpha\beta}\sigma_\alpha\sigma_\beta)) \quad (5.20)$$

$$\beta_j \sim N((\gamma_0^\beta + \gamma_1^\beta z_j), (\rho_{\alpha\beta}\sigma_\alpha\sigma_\beta, \sigma_\beta^2)) \quad (5.21)$$

In R, including an interaction between floor and loguranium is equivalent to letting county-level loguranium be a predictor in the level 2 model of the county-level slope on floor:

```
vary.slope.u <- lmer(logradon ~ floor + loguranium +
                      floor*loguranium + (1 + floor|county),
                      data = MNradon)
summary(vary.slope.u)
```

With this model we've almost completely accounted for the county-level variance in radon levels. The overall average regression line is:

$$\text{Log-Radon}_{ij} = 1.72 - 0.46 \times \text{floor}_i + 0.63 \times \text{loguranium}_j - 0.33 \times \text{floor}_i \times \text{loguranium}_j$$

For a specific county j we need to get α_j , which is 1.72 plus the random deviation u_j for county j , as well as β_j which is -0.46 plus the random slope deviation s_j :

$$\begin{aligned} \text{Log-Radon}_{ij} \\ = 1.72 + u_j - 0.46 \times \text{floor}_i + 0.63 \times \text{loguranium}_j - 0.33 \times \text{floor}_i \times \text{loguranium}_j + s_j \times \text{floor}_i \end{aligned}$$

Our regression line for Aitkin county, with random intercept deviation $u_j = -0.015$, random slope deviation $s_j = 0.005$, and log uranium of -0.689 is:

$$\text{Log-Radon}_{ij}$$

$$= 1.72 - 0.015 - 0.46 \times \text{floor}_i - 0.63 \times 0.689 + 0.33 \times \text{floor}_i \times 0.689 + 0.005 \times \text{floor}_i \\ = 1.27 - 0.23 \times \text{floor}_i$$

We can use this procedure to calculate a vector of intercepts and slopes, one for each county:

```
# Again, need to make sure the counties are ordered the same
# as the log uranium values
rownames <- rownames(ranef(lmer.u)$county)
loguranium <- c()
for(i in rownames){
  loguranium[i] <- mean(MNradon$loguranium[MNradon$county == i])
}

# Get the county-level intercept
ints <- coef(vary.slope.u)$county[, 1]
# Add in the prediction from the log uranium levels
ints <- ints + coef(vary.slope.u)$county[, 3]*loguranium

# Get the county-level random slope
slopes <- coef(vary.slope.u)$county[, 2]
# Add in the prediction from the log uranium levels
slopes <- slopes + coef(vary.slope.u)$county[, 4]*loguranium
```

Next, let's plot the level 1 model for six counties:

```
plot1 <- MNradon[MNradon$county %in% unique(MNradon$county)[1:6], ]
plot1 <- droplevels(plot1)
j <- 1
png(filename = "varyingslope1.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mfrow = c(2, 3), mar = c(5, 4, 2, 1),
    cex.lab = 1.5, cex.axis = 1.5)
for(i in unique(plot1$county)){
  plot(jitter(plot1$floor[plot1$county == i], amount = 0.05),
       plot1$logradon[plot1$county == i],
       ylim = c(-1, 3),
       pch = 19,
```

```

col  = rgb(0, 0, 0, .5),
xaxt = "n",
xlab = "Floor",
ylab = "Log Radon Levels",
xlim = c(-0.1, 1.1),
main = i)
axis(side = 1, at = c(0, 1), labels = c("Basement", "First Floor"))
# Add complete pooling regression line
abline(coef(radonreg4.cp)[1], coef(radonreg4.cp)[2], lty = 2, lwd = 2)
# Add no pooling regression line
abline(coef(radonreg4.np)[j + 1], coef(radonreg4.np)[1], lty = 1, lwd = 2)
# Add partial pooling intercept and slopes
abline(ints[j], slopes[j], lty = 3, lwd = 2, col = "red")
j <- j + 1
}
dev.off()

```

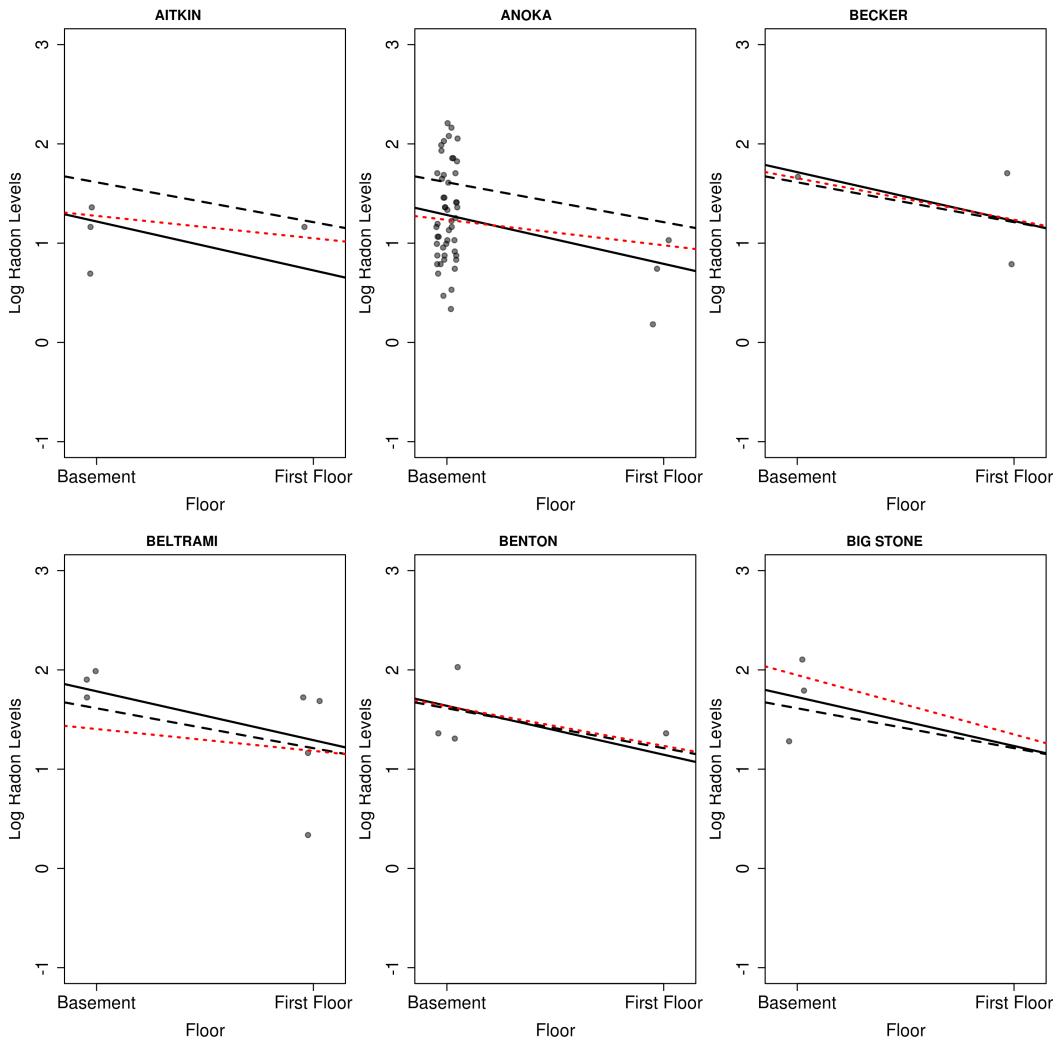


Figure 5.20: Level 1 model of log radon activity in six counties. Complete pooling regression is shown on the dashed line. No pooling regression line is shown in the solid line. Partial pooling varying-slope varying intercept is shown in red, with county-level uranium as a predictor of the intercept and slope.

Some counties have steeper slopes than others, and those slopes are even steeper if the log uranium levels in the county are higher. For example, the log uranium levels in Aitkin county are below the 25th percentile, explaining why the slope is much flatter than complete pooling would suggest. Similarly, some counties have higher intercepts, and those intercepts are higher if the log uranium levels are high. For example, Big Stone has a high intercept and steep slope, indicating high uranium levels in the soil (it's higher than the 75th percentile). Next, let's look at the level 2 model:

```

# Get the standard errors for the county-level intercepts and slopes
std.error.ints <- se.ranef(vary.slope.u)$county[, 1]
std.error.slopes <- se.ranef(vary.slope.u)$county[, 2]

png(filename = "varying2.png", width = 3000, height = 3000, res = 300)
par(cex = 2, mfrow = c(2, 1), mar = c(5, 4, 2, 1))
plot(loguranium, ints,
      pch = 20,
      col = rgb(0, 0, 0, .5),
      ylab = "Log-Radon Intercept by County",
      xlab = "County-Level Log Uranium",
      ylim = c(1, 2.5))
# This is the regression line for the intercepts
abline(fixef(vary.slope.u)[1], fixef(vary.slope.u)[3])

for(j in 1:85){
  lines(rep(loguranium[j], 2),
        ints[j] + c(-1, 1)*std.error.ints[j],
        lwd = 0.5)
}

plot(loguranium, slopes,
      pch = 20,
      col = rgb(0, 0, 0, .5),
      ylab = "Log-Radon Slope by County",
      xlab = "County-Level Log Uranium",
      ylim = c(-1.5, .5))
# This is the regression line for the slopes
abline(fixef(vary.slope.u)[2], fixef(vary.slope.u)[4])

for(j in 1:85){
  lines(rep(loguranium[j], 2),
        slopes[j] + c(-1, 1)*std.error.slopes[j],
        lwd = 0.5)
}
dev.off()

```

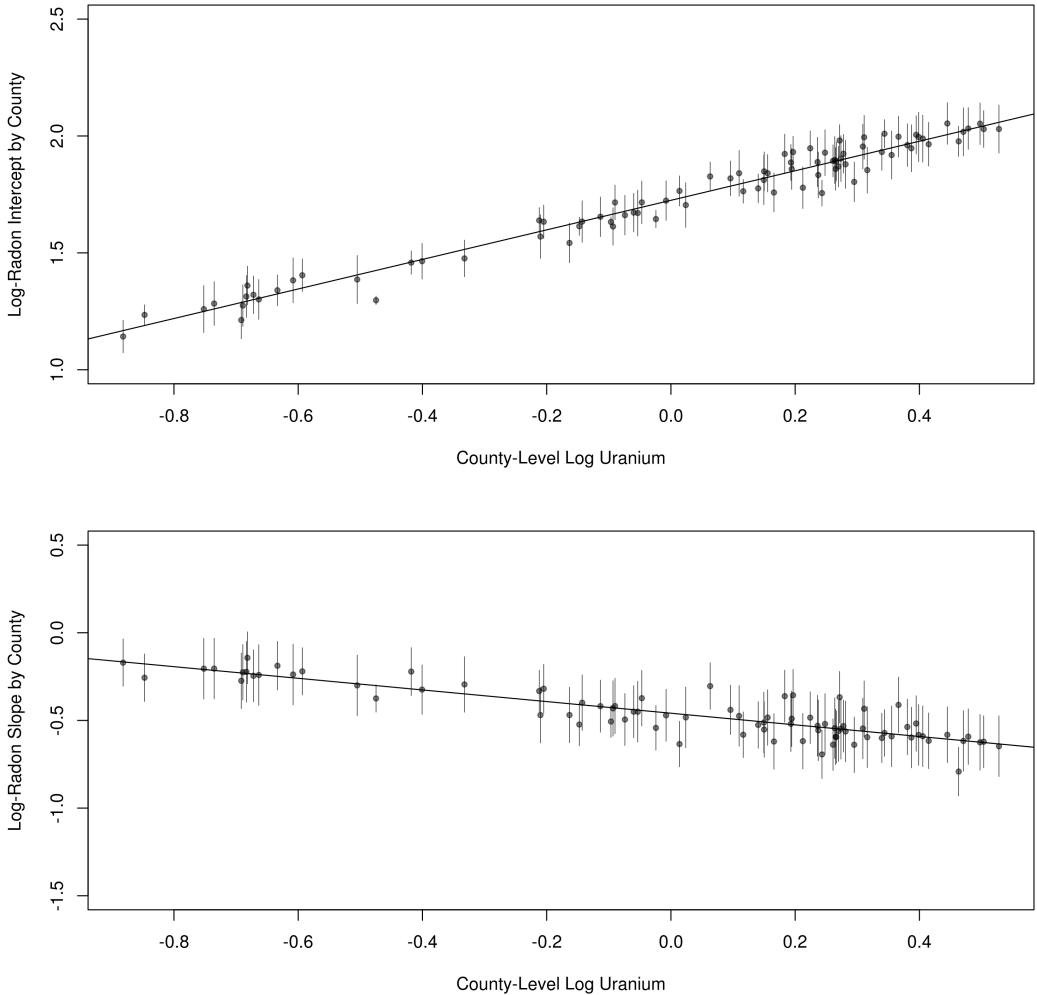


Figure 5.21: Level 2 model of partially pooled county-level intercepts (top) and slopes (bottom) plotted against county-level log uranium in the soil.

Our model of the county-level intercepts is much better than the county-level slopes. The discrepancy of each county's intercept from the regression line is a visual representation of the county-level variance $\sigma_\alpha^2 = 0.01$, and similarly for the discrepancy of each county's slope and $\sigma_\beta^2 = 0.03$. We can also see the correlation between the county-level intercept and county-level slope: As the intercept gets larger, indicating more uranium in the soil, the slope gets more negative (steeper) indicating a greater effect of measuring in the basement (close to the uranium) versus first floor. This visually captures $\rho_{\alpha\beta}$.

We can keep going pretty much indefinitely with more and more complex models at higher levels, including various types of crossed designs (where

observations are not nested within observational unit), and pretty much any distributional assumptions using Bayesian models. This includes models that are non-linear in parameters. Texts by [Bates \(2010\)](#), [Gelman and Hill \(2007\)](#) and [Faraway \(2004\)](#) include extensive resources.

5.7 Ordinary Least Squares Regression with Correlated Errors

An important way to think about multi-level models is that they are an extension of classical regression to allow for correlated errors. Recall from the simple linear regression model the simple random sampling assumption, which means that observations are independent. Independent means the the errors of each observation in the regression are uncorrelated. Multi-level models can be seen as an extension of this model to allow for errors to be correlated within group. Consider the following model:

$$y_i = \beta_0 + \beta_1 x_{i1} + \epsilon_i \quad (5.22)$$

$$\epsilon_i \sim N(0, \Omega) \quad (5.23)$$

In the classical least squares model, Ω is just a matrix of zeros in the off-diagonal and a constant σ^2 on the diagonal, which basically says the errors have a constant variance and there is no correlation in the errors between different observations (the off-diagonal). Suppose we have 8 observations, then Ω will look like this:

$$\Omega = \begin{pmatrix} \sigma^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma^2 \end{pmatrix}$$

Looking at this matrix, in the upper left corner $\Omega[1, 1]$ we have the covariance of the errors of the first observation with itself, which is just the error variance σ^2 . Below that, $\Omega[2, 1]$ shows the covariance of the second observation with the first observation, which is zero, due to our independence assumption.

In the multi-level model, the off-diagonal is not assumed to be zero, instead, the observations are correlated with each other depending on their grouping. If we assume that ϵ_i is composed of two parts, an idiosyncratic

error and a group-level error that are additive, then the error for each observation takes the following form:

$$\epsilon_i = \mu_{ig} + \alpha_g \quad (5.24)$$

Where μ_{ig} is the idiosyncratic error for observation i in group g and is distributed $N(0, \sigma_y^2)$, while α_g is an error term corresponding to a different intercept for group g with distribution $N(0, \sigma_\alpha^2)$. For example, observations of log-radon values are correlated with each other within county. Thus, if observations 1-4 are in the same group (e.g., same county), while observations 5-8 are in a different group, our covariance matrix Ω looks something like this:

$$\Omega = \begin{pmatrix} \sigma_y^2 + \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_\alpha^2 & 0 & 0 & 0 & 0 \\ \sigma_\alpha^2 & \sigma_y^2 + \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_\alpha^2 & 0 & 0 & 0 & 0 \\ \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_y^2 + \sigma_\alpha^2 & \sigma_\alpha^2 & 0 & 0 & 0 & 0 \\ \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_y^2 + \sigma_\alpha^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_y^2 + \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_\alpha^2 \\ 0 & 0 & 0 & 0 & \sigma_\alpha^2 & \sigma_y^2 + \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_\alpha^2 \\ 0 & 0 & 0 & 0 & \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_y^2 + \sigma_\alpha^2 & \sigma_\alpha^2 \\ 0 & 0 & 0 & 0 & \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_\alpha^2 & \sigma_y^2 + \sigma_\alpha^2 \end{pmatrix}$$

Here, the observations in rows 1-4 are correlated with each other (with σ_α^2 on the off diagonal) because they are from the same group (county). Their correlation is zero if they are in different groups, and $\rho = \frac{\sigma_\alpha^2}{\sigma_y^2 + \sigma_\alpha^2}$ if they are in the same group. Thus, multi-level models can be seen as an extension of typical ordinary least squares regression.

5.7.1 Estimation Using Feasible Generalized Least Squares

The extension of ordinary least squares that allows for different variances for each observation is called *weighted least squares*. A simple example is accounting for heteroskedasticity. Suppose the heteroskedasticity in our data follows a simple functional form:

$$\text{Var}(u|x) = \sigma^2 h(x) \quad (5.25)$$

where $h(x)$ is some known function. If we just divide everything by $\sqrt{h(x)}$, then the heteroskedasticity is eliminated, as:

$$\text{Var}\left(u \mid \frac{x}{\sqrt{h(x)}}\right) = \sigma^2 \frac{h(x)}{h(x)} = \sigma^2 \quad (5.26)$$

What we've done is divide each observation by its *inverse* variance to produce a new model that is homoskedastic. We can use the same logic to apply weighted least squares to the multi-level regression. Here, the matrix Ω is our weight matrix, weighting each observation inversely to its variance:

$$\hat{\beta}_{GLS} = (X' \Omega^{-1} X)^{-1} X' \Omega^{-1} y \quad (5.27)$$

where Ω^{-1} is the inverse of the covariance matrix Ω . The challenge is then to find the Ω matrix so that we can turn our problem into a weighted least squares problem. To do this, recall that we are assuming the errors follow a separable additive structure:

$$\epsilon_i = \mu_{ig} + \alpha_g \quad (5.28)$$

with $\mu_{ig} \sim N(0, \sigma_y^2)$ and $\alpha_g \sim N(0, \sigma_\alpha^2)$. Our general assumptions are that μ_{ig} , α_g and X are independent for all i and g . If we know σ_α^2 and σ_y^2 , we can then calculate ([Baltagi, 2008](#)):

$$\Omega = \sigma_\alpha^2 (I_N \otimes J_T) + \sigma_y^2 (I_N \otimes I_T) \quad (5.29)$$

where I_N is an $N \times N$ identity matrix, I_T is a $T \times T$ identity matrix, J_T is a $T \times T$ matrix of ones, and \otimes is the Kronecker product . For example, if $N = 2$, then I_N is:

$$I_N = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (5.30)$$

and J_T is a matrix of ones of dimension T where T is the maximum number of observations within each observational unit:

$$J_T = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \quad (5.31)$$

so $I_N \otimes J_T$ is:

$$I_N \otimes J_T = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad (5.32)$$

Multiplying by the scalar σ_α^2 gives:

$$\sigma_\alpha^2 (I_N \otimes J_T) = \begin{pmatrix} \sigma_\alpha^2 & \sigma_\alpha^2 & 0 & 0 \\ \sigma_\alpha^2 & \sigma_\alpha^2 & 0 & 0 \\ 0 & 0 & \sigma_\alpha^2 & \sigma_\alpha^2 \\ 0 & 0 & \sigma_\alpha^2 & \sigma_\alpha^2 \end{pmatrix} \quad (5.33)$$

Similarly, to calculate $\sigma_y^2(I_N \otimes I_T)$, consider that I_T is an identity matrix of dimension T :

$$I_T = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (5.34)$$

giving us:

$$I_N \otimes I_T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (5.35)$$

multiplying by σ_y^2 :

$$\sigma_y^2(I_N \otimes I_T) = \begin{pmatrix} \sigma_y^2 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & \sigma_y^2 & 0 \\ 0 & 0 & 0 & \sigma_y^2 \end{pmatrix} \quad (5.36)$$

adding it all up, we get:

$$\Omega = \sigma_\alpha^2(I_N \otimes J_T) + \sigma_y^2(I_N \otimes I_T) = \begin{pmatrix} \sigma_y^2 + \sigma_\alpha^2 & \sigma_\alpha^2 & 0 & 0 \\ \sigma_\alpha^2 & \sigma_y^2 + \sigma_\alpha^2 & 0 & 0 \\ 0 & 0 & \sigma_y^2 + \sigma_\alpha^2 & \sigma_\alpha^2 \\ 0 & 0 & \sigma_\alpha^2 & \sigma_y^2 + \sigma_\alpha^2 \end{pmatrix} \quad (5.37)$$

It is possible to write this in a more simple form:

$$\Omega = (T\sigma_\alpha^2 + \sigma_y^2)P + \sigma_y^2Q \quad (5.38)$$

where $P = I_N \otimes J_T/T$, a matrix that averages each observation across time for each observational unit:

$$P = I_N \otimes J_T/T = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \end{pmatrix} \quad (5.39)$$

That is, Py gives a vector of length NT with \bar{y}_i repeated T times, stacked on top of each other. Next, $Q = I_{NT} - P$, is a matrix which obtains the deviations from individual means:

$$Q = I_{NT} - P = \begin{pmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 \end{pmatrix} \quad (5.40)$$

This form is useful because we want to invert Ω , and writing Ω in terms of P and Q makes the following simple identity possible:

$$\Omega^{-1} = \frac{1}{(T\sigma_\alpha^2 + \sigma_y^2)} P + \frac{1}{\sigma_y^2} Q \quad (5.41)$$

Calculating P and Q only requires matrix algebra. However, because σ_α^2 and σ_y^2 are unknown, we need to estimate them. Using this regression with estimated variances is called *Feasible Generalized Least Squares* or FGLS.

There's also a more illuminating form of the FGLS estimator discovered by [Maddala \(1971\)](#):

$$\hat{\beta}_{FGLS} = W_1 \hat{\beta}_{within} + W_2 \hat{\beta}_{between} \quad (5.42)$$

where $\hat{\beta}_{within}$ and $\hat{\beta}_{between}$ are the estimates of β from the within and between regressions, and $W_1 = I - W_2$ are weights placed on the estimates from these two regressions. We will introduce the within and between regressions shortly, but the point is that the FGLS estimator is a weighted combination of the within and between estimates. A simpler form is ([Wooldridge, 2009](#)):

$$y_{it} - \lambda \bar{y}_i = \beta_0(1 - \lambda) + \beta_1(x_{it1} - \lambda \bar{x}_{it1}) + (v_{it} - \lambda \bar{v}_i) \quad (5.43)$$

where:

$$\lambda = 1 - \sqrt{\frac{\sigma_y^2}{\sigma_y^2 + T\sigma_\alpha^2}} \quad (5.44)$$

If $\lambda \approx 0$ then we just get our complete pooling estimate, which will happen when $\sigma_\alpha^2 \approx 0$. If $\lambda \approx 1$ then σ_y^2 is small relative to σ_α^2 and we have subtracted the group mean \bar{y}_i and \bar{x}_{ik} from each observation. This is called *demeaning* and results in the within estimator. Thus, one way to think of partial pooling is that it is a *quasi-demeaned* estimator, where the amount of demeaning is determined by λ .

Let's elaborate a bit more with an example. Suppose we have $N = 200$ observational units (e.g., firms, households), each with $T = 5$ measurements (e.g., time periods, responses on a test), yielding $N \times T = 1000$ total observations:

```

# Number of groups
N <- 200
# Number of observations per group
T <- 5

```

Next, we stack the group variable indicating each observational unit on top of each other, and draw a random intercept for each observational unit. Note, that you must stack the groups on top of each other so that all of the observations from unit 1 are above unit 2, and unit 2 above unit 3, or else the matrix algebra will fail:

```

# Create stacked groups
group <- sort(rep(1:N, 5))
# Set sigma.alpha
sigma1.alpha <- 1
# Draw intercepts from N(0, sigma.alpha)
alpha1.j <- rnorm(N, 0, sigma1.alpha)

```

Next, generate a regressor x from a uniform distribution and a bunch of ones for the intercept in our data matrix:

```

# Set X uniform from 1 to 2
x <- runif(N*T, 1, 2)
ones <- rep(1, length(x))
# Create data matrix
X <- cbind(ones, x)

```

Finally, we generate our population regression function $y_{ij} = 1 + x_i + \alpha_j + e_{ij}$ with normally distributed errors with standard deviation 1:

```

# Create y
# Set sigma.y's
sigma1.y <- 1
# Draw errors from N(0, sigma.y)
e.y1 <- rnorm(N*T, 0, sigma1.y)
beta0 <- 1
beta1 <- 1
y <- beta0 + alpha1.j[group] + beta1*x + e.y1

```

Then create $P = I_N \otimes J_T/T$ and $Q = I_{NT} - P$. Note that the Kronecker product looks similar to the matrix product, but with an x in between the % signs:

```

# Create P matrix
In <- diag(N)
JtT <- matrix(1/T, nrow = T, ncol = T)
P <- In %*% JtT

# Create Q matrix
Q <- diag(N*T) - P

```

With P and Q we can now calculate the within and between regressions.

Within Regression

The within regression is identical to the no pooling regression we've already discussed. The population regression function can be written:

$$y_{it} = \beta_0 + \beta_1 x_{it} + \alpha_i + v_{it} \quad (5.45)$$

If we average over the t for each observational unit i we get:

$$\bar{y}_{i\cdot} = \beta_0 + \beta_1 \bar{x}_{i\cdot} + \alpha_i + \bar{v}_{i\cdot} \quad (5.46)$$

If we subtract $\bar{y}_{i\cdot}$ from y_{it} we get:

$$y_{it} - \bar{y}_{i\cdot} = \beta_1(x_{it} - \bar{x}_{i\cdot}) + (v_{it} - \bar{v}_{i\cdot}) \quad (5.47)$$

This is the within regression, where we calculate the regression of y on x after subtracting the mean of y and the mean of x for each observational unit. This removes any of the variation between observational units α_i , meaning the within regression captures the relationship between x and y within each observational unit. To calculate the within regression using matrices we can use the Q matrix, which calculates deviations from means:

$$\hat{\beta}_{within} = ((QX)'QX)^{-1}(QX)'Qy \quad (5.48)$$

which reduces to:

$$\hat{\beta}_{within} = (X'QX)^{-1}X'Qy \quad (5.49)$$

because Q is idempotent. We can calculate this in R:

```

# Calculate the deviation of y from its group means
QY <- Q %*% y
# Calculate the deviation of x from its group means
QX <- Q %*% X

# To avoid dummy variable trap you need to remove the first column

```

```

QX <- Q %*% X[, -1]

# Within regression
within <- solve(t(QX) %*% QX) %*% t(QX) %*% QY
within

```

Let's compare this to the within function in the plm package:

```

# Within using the plm package
# install.packages("plm")
library(plm)
data1 <- data.frame(x = x, y = y, group = group)
data1.p <- pdata.frame(data1, "group")
within.plm <- plm(y ~ x, index = "group",
                    model = "within",
                    data = data1.p)
coef(within.plm)

```

The within regression should also be identical to the no pooling regression that we've previously conducted, which is also called the least-squares dummy variable (LSDV) regression:

```

lsdv.lm <- lm(y ~ x + factor(group), data = data1)
coef(lsdv.lm) ["x"]

# Least Squares Dummy Variable in matrix form
D <- diag(N) %x% rep(1, T)
MD <- diag(N*T) - D %*% solve(t(D) %*% D) %*% t(D)
# Avoid Dummy variable trap
lsdv <- solve(t(X[, -1]) %*% MD %*% X[, -1]) %*% t(X[, -1]) %*% MD %*% y
lsdv

```

The results should be identical. Using LSDV with R's lm function is a good way to check you've done your matrix computations correctly.

Between Regression

The between regression is a new, but simple, concept. Here we just average over the observations for each observational unit, treating each observational unit as a single observation:

$$\bar{y}_i = \alpha + \bar{X}'_i \beta + \bar{u}_i. \quad (5.50)$$

which is equivalent to:

$$Py = PX\beta + Pu \quad (5.51)$$

You can see that Py gives a vector of length NT with \bar{y}_i repeated T times, stacked on top of each other. This is the between transformation, in that all we observe is variation between groups, giving us our estimator:

$$\hat{\beta}_{between} = ((PX)'PX)^{-1}(PX)'Py \quad (5.52)$$

which reduces to:

$$\hat{\beta}_{between} = (X'PX)^{-1}X'Py \quad (5.53)$$

because P is also idempotent. We can calculate the between estimator in R:

```
# Between estimate in matrix form
PY <- P %*% y
PX <- P %*% X
between <- solve(t(PX) %*% PX) %*% t(PX) %*% PY
between

# Between estimate from plm
between.plm <- plm(y ~ x,
                      index = "id",
                      model = "between",
                      data = data1.p)
between.plm
```

The matrix form and plm function should produce identical results.

Estimating Variance Components

If we want to use either:

$$\hat{\beta}_{GLS} = (X'\Omega^{-1}X)^{-1}X'\Omega^{-1}y \quad (5.54)$$

or

$$\hat{\beta}_{FGLS} = W_1\hat{\beta}_{within} + W_2\hat{\beta}_{between} \quad (5.55)$$

to estimate $\hat{\beta}$, we need to know σ_y^2 and σ_α^2 . We've previously seen the importance of these variance components for calculating Ω . They are also critical to understanding W_1 because:

$$W_1 = (X'QX + \frac{\sigma_y^2}{T\sigma_\alpha^2 + \sigma_y^2}X'(P - J_{NT}/T)X)^{-1}X'QX \quad (5.56)$$

We can simplify W_1 by considering that $W_{xx} = X'QX$ and $B_{xx} = X'(P - J_{NT}/T)X$, giving us:

$$W_1 = (W_{xx} + \frac{\sigma_y^2}{T\sigma_\alpha^2 + \sigma_y^2} B_{xx})^{-1} W_{xx} \quad (5.57)$$

Now, if σ_y^2 is small compared to σ_α^2 , meaning that there is a lot of variation between groups, but not a lot within groups, then $W_1 \approx 1$, $W_2 \approx 0$, and $\hat{\beta}_{FGLS} \approx \hat{\beta}_{within}$. This should mirror the previous partial pooling intuition, where the within regression is the same as the no pooling regression, and the partial pooling estimates are close to the no pooling estimates when there is a lot of variation across groups (i.e., σ_α^2 is large compared to σ_y^2).

Conversely, if W_{xx} is small relative to $\frac{\sigma_y^2}{T\sigma_\alpha^2 + \sigma_y^2} B_{xx}$, then $W_1 \approx 0$, $W_2 \approx 1$, and $\hat{\beta}_{FGLS} \approx \hat{\beta}_{between}$. This would happen if there is little between group variation (meaning σ_α^2 is small relative to σ_y^2).

We can estimate σ_y^2 using the MSE from the within regression or LSDV regression. The MSE of the residuals from this regression are an unbiased estimate of σ_y^2 :

$$\hat{\sigma}_y^2 = \frac{\sum_{i=1}^n \sum_{t=1}^T ((y_{it} - \bar{y}_{i\cdot}) - \hat{y}_{it}^{within})^2}{N(T-1) - K} \quad (5.58)$$

in matrix form:

$$\hat{\sigma}_y^2 = \frac{\mathbf{y}' Q \mathbf{y} - \mathbf{y}' Q X (X' Q X)^{-1} X' Q \mathbf{y}}{N(T-1) - K} \quad (5.59)$$

in R:

```
# From LSDV regression just square the rmse
lsdv.MSE <- (summary(lsdv.lm)$sigma)^2

# In matrix form from within regression
sigma.y.sq <- (t(y) %*% QY - t(y) %*% QX %*% within)/(N*(T-1) - dim(QX)[2])
```

To estimate σ_α^2 we first need to estimate the quantity $\sigma_\alpha^2 + \sigma_y^2$. To do this, we can use the following shortcut proposed by [Greene \(2008\)](#). The reasoning is simple, in that if we conduct a regression with no covariates (only an intercept) the error should be:

$$\text{Var}(Y) = \sigma_{pooled}^2 = \sigma_\alpha^2 + \sigma_y^2 \quad (5.60)$$

From $\hat{\sigma}_{pooled}^2$ we can calculate $\hat{\sigma}_\alpha^2 = \hat{\sigma}_{pooled}^2 - \hat{\sigma}_y^2$:

```
pool.reg <- lm(y ~ 1, data = data1)
sigma.pooled.sq <- summary(pool.reg)$sigma^2
sigma.alpha.sq <- sigma.pooled.sq - sigma.y.sq
```

FGLS Estimate

We now have all of the components necessary to estimate $\hat{\beta}_{GLS}$:

$$\hat{\beta}_{GLS} = (X' \Omega^{-1} X)^{-1} X' \Omega^{-1} y \quad (5.61)$$

this is equal to:

$$\hat{\beta}_{GLS} = W_1 \tilde{\beta}_{Within} + W_2 \hat{\beta}_{Between} \quad (5.62)$$

in R:

```
# Calculate Wxx
Wxx <- t(X) %*% Q %*% X

# Calculate Bxx
JnT <- matrix(1/T, nrow = N*T, ncol = T*N)
Bxx <- t(X) %*% (P - JnT) %*% X

# Calculate phi.sq
phi.sq <- sigma.y.sq/(T*sigma.alpha.sq + sigma.y.sq)

# Calculate W1 and W2
W1 <- solve(Wxx + diag(phi.sq)*Bxx) %*% Wxx
W2 <- diag(dim(Bxx)[2]) - W1

# Set within to have the right dimension
within1 <- cbind(0, within)
beta.gls <- W1 %*% t(within1) + W2 %*% between
beta.gls
```

We can compare to the partial pooling multi-level models we've used:

```
library(lmer)
lmer1 <- lmer(y ~ x + (1|group), data = data1)
fixef(lmer1)
```

and the [Amemiya \(1971\)](#) random effects model in the plm package:

```
# Plm offers a bunch of different ways of calculating the random effects
amemiya <- plm(y ~ x,
                 data = data1.p,
                 index = "id",
                 model = "random",
                 random.method = "amemiya")
summary(amemiya)
```

The FGLS estimator, lmer, and Amemiya in the plm package should all agree.

5.8 Residual Diagnostics for Multi-Level Models

When conducting residual diagnostics for multi-level models we must be very careful to consider how the residuals are associated across different levels of the model. A multi-level model with one level is one regression with one set of residuals. A multi-level model with two levels has at least two sets of residuals (one set of residuals for each varying parameter). The residuals at the first level and at the second level depend on each other; a misspecification at level one may show up at level two, or vice versa. For example, heteroskedasticity at a level 1 model can be equivalently modeled with random slopes (consider the interaction-heteroskedasticity graph from earlier with z as a grouping variable) or an omitted interaction.

5.8.1 Level 1 Least Squares Diagnostics

There are two types of level 1 residuals that we'll look at: 1) least squares residuals (i.e., the residuals we are used to) and, 2) empirical bayes residuals (the residuals from the multi-level model) ([Loy and Hofmann, 2014](#)). Least squares residuals are just residuals from complete or no pooling regressions. Let's first use the complete pooling regression with floor of measurement and log-uranium levels as predictors:

```
library(car)
# Fit complete pooling regression with floor and uranium as predictors
LS.resids1 <- lm(logradon ~ floor + loguranium, data = MNradon)
png(filename = "LSresids1.png", width = 3000, height = 3000, res= 300)
par(cex = 1.3, mar = c(5, 4, 2, 1), mfrow = c(2, 2))
# Plot the fitted versus residuals
plot(fitted(LS.resids1), rstudent(LS.resids1),
      xlab = "Fitted Values",
      ylab = "Jackknife LS Residuals",
      pch = 19,
      col = rgb(0, 0, 0, .3))
# Add lowess smooth
lines(lowess(fitted(LS.resids1), rstudent(LS.resids1)),
      col = "green", lwd = 2)
abline(h = 0, lty = 2, col = "red", lwd = 2)
plot(MNradon$floor, rstudent(LS.resids1),
      xlab = "Floor Level",
      ylab = "Jackknife LS Residuals",
      pch = 19,
```

```
    col = rgb(0, 0, 0, .3))
plot(MNradon$loguranium, rstudent(LS.resids1),
      xlab = "Log Uranium Level",
      ylab = "Jackknife LS Residuals",
      pch = 19,
      col = rgb(0, 0, 0, .3))
lines(lowess(MNradon$loguranium, rstudent(LS.resids1)),
      col = "green", lwd = 2)
abline(h = 0, lty = 2, col = "red", lwd = 2)
qqPlot(LS.resids1, id.n = 3,
       distribution = "t",
       df = df.residual(LS.resids1),
       ylab = "Jackknife LS Residuals")
dev.off()
```

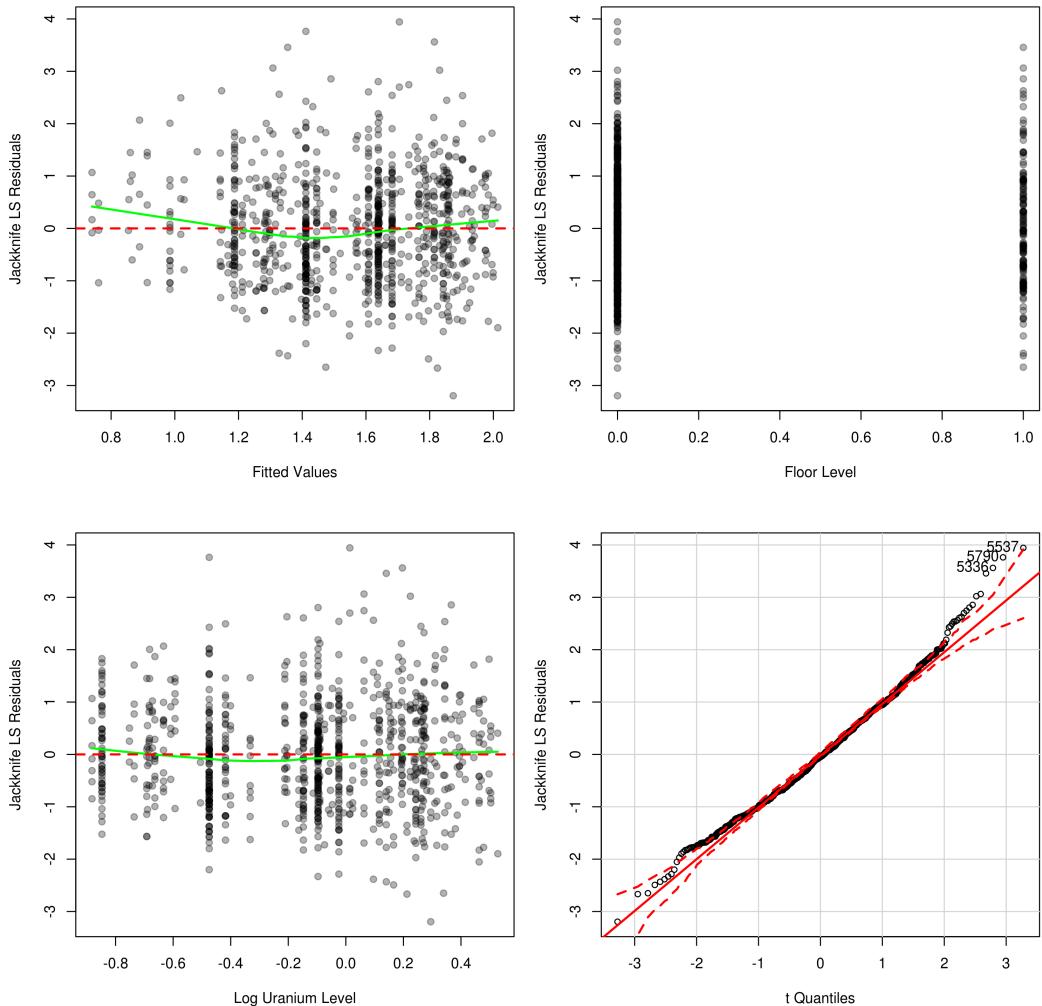


Figure 5.22: Level 1 Least Squares (LS) jackknife residuals from the complete pooling regression. The top left plot shows the jackknife residuals versus fitted values. The top right plot shows the jackknife residuals versus floor of radon level measurement (0 = basement, 1 = first floor). The bottom left plot shows the jackknife residuals versus log uranium levels. The bottom right plot shows the jackknife residuals against the quantiles of the t distribution.

There doesn't seem to be a relationship between the conditional mean of the residuals and the fitted values, but there does seem to be a heteroskedastic pattern, with the variance of the jackknife residuals increasing as a function of the fitted values. This seems to be related to the spread of the residuals as a function of the log uranium levels, as well as a greater spread among residuals for houses that measured radon levels in the basement. This makes sense, as we previously saw that we need a varying slope for level of measurement

depending on county-level uranium levels, as measurements in the basement will show more radon when uranium levels in the soil are high compared to when the soil uranium levels are low. The qqplot of the jaccknife residuals also seems to indicate that there are potential outliers with high discrepancy. To find this out, we fit a model with the appropriate interaction:

```

LS.resids2 <- lm(logradon ~ floor*loguranium, data = MNradon)
png(filename = "LSresids2.png", width = 3000, height = 3000, res= 300)
par(cex = 1.3, mar = c(5, 4, 2, 1), mfrow = c(2, 2))
plot(fitted(LS.resids2), rstudent(LS.resids2),
      xlab = "Fitted Values",
      ylab = "Jackknife Residuals",
      pch = 19,
      col = rgb(0, 0, 0, .3))
lines(lowess(fitted(LS.resids2), rstudent(LS.resids2)),
      col = "green", lwd = 2)
abline(h = 0, lty = 2, col = "red", lwd = 2)
plot(MNradon$floor, rstudent(LS.resids2),
      xlab = "Floor Level",
      ylab = "Jackknife LS Residuals",
      pch = 19,
      col = rgb(0, 0, 0, .3))
plot(MNradon$loguranium, rstudent(LS.resids2),
      xlab = "Log Uranium Level",
      ylab = "Jackknife LS Residuals",
      pch = 19,
      col = rgb(0, 0, 0, .3))
lines(lowess(MNradon$loguranium, rstudent(LS.resids2)),
      col = "green", lwd = 2)
abline(h = 0, lty = 2, col = "red", lwd = 2)
qqPlot(LS.resids2, id.n = 3,
       distribution = "t",
       df = df.residual(LS.resids1),
       ylab = "Jackknife LS Residuals")
dev.off()

```

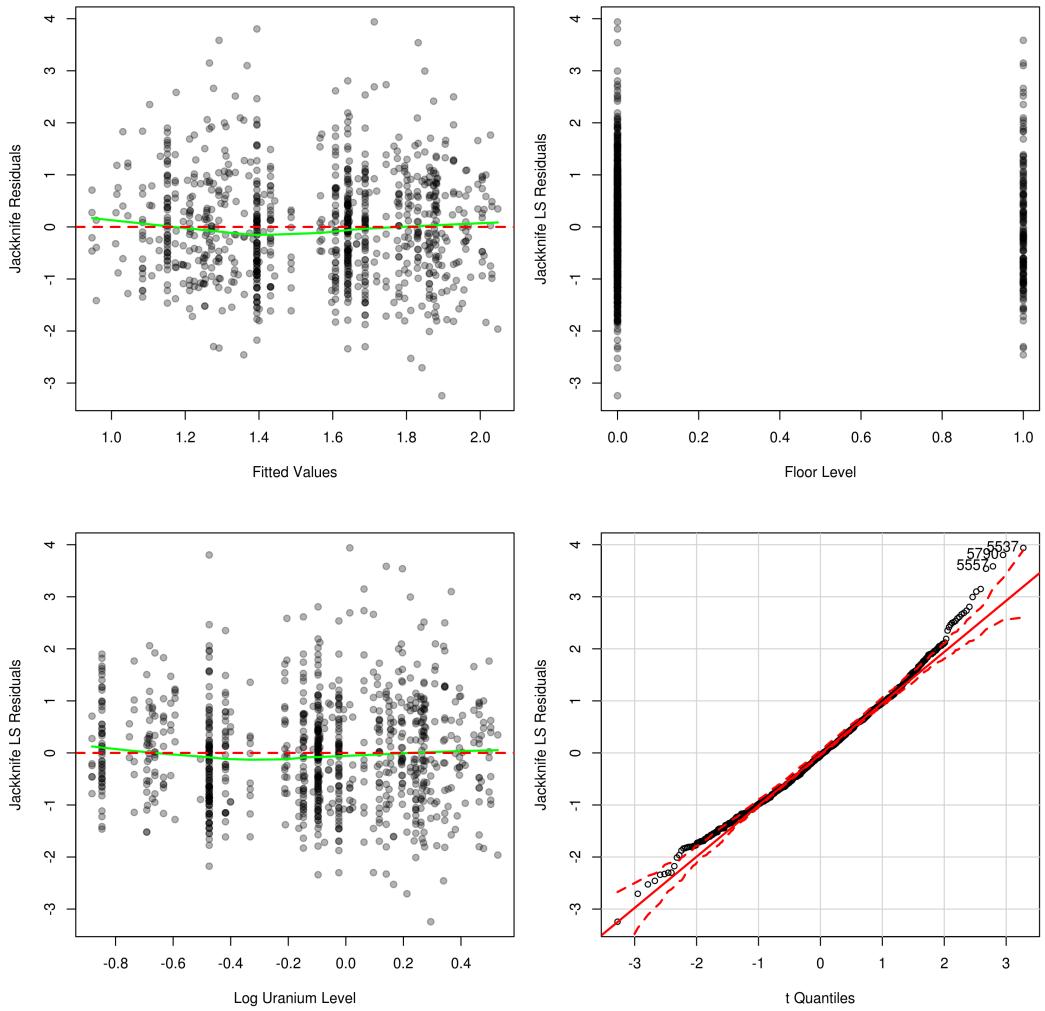


Figure 5.23: Level 1 Least Squares (LS) jackknife residuals from the complete pooling regression. The top left plot shows the jackknife residuals versus fitted values. The top right plot shows the jackknife residuals versus floor of radon level measurement (0 = basement, 1 = first floor). The bottom left plot shows the jackknife residuals versus log uranium levels. The bottom right plot shows the jackknife residuals against the quantiles of the t distribution.

The jackknife residuals still seem to be increasing in the uranium levels, and also seem to deviate from the t distribution. We found earlier that $\lambda = 0$ was not the maximum likelihood estimate for the Box-Cox transformation, so we should check whether using the maximum likelihood estimate of $\lambda = -0.2$ helps:

```
LS.resids3 <- lm(radon0.2 ~ floor*loguranium, data = MNradon)
png(filename = "LSresids3.png", width = 3000, height = 3000, res= 300)
```

```

par(cex = 1.3, mar = c(5, 4, 2, 1), mfrow = c(2, 2))
plot(fitted(LS.resids3), rstudent(LS.resids3),
      xlab = "Fitted Values",
      ylab = "Jackknife Residuals",
      pch = 19,
      col = rgb(0, 0, 0, .3))
lines(lowess(fitted(LS.resids3), rstudent(LS.resids3)),
      col = "green", lwd = 2)
abline(h = 0, lty = 2, col = "red", lwd = 2)
plot(MNradon$floor, rstudent(LS.resids3),
      xlab = "Floor Level",
      ylab = "Jackknife LS Residuals",
      pch = 19,
      col = rgb(0, 0, 0, .3))
plot(MNradon$loguranium, rstudent(LS.resids3),
      xlab = "Log Uranium Level",
      ylab = "Jackknife LS Residuals",
      pch = 19,
      col = rgb(0, 0, 0, .3))
lines(lowess(MNradon$loguranium, rstudent(LS.resids3)),
      col = "green", lwd = 2)
abline(h = 0, lty = 2, col = "red", lwd = 2)
qqPlot(LS.resids3, id.n = 3,
       distribution = "t",
       df = df.residual(LS.resids1),
       ylab = "Jackknife LS Residuals")
dev.off()

```

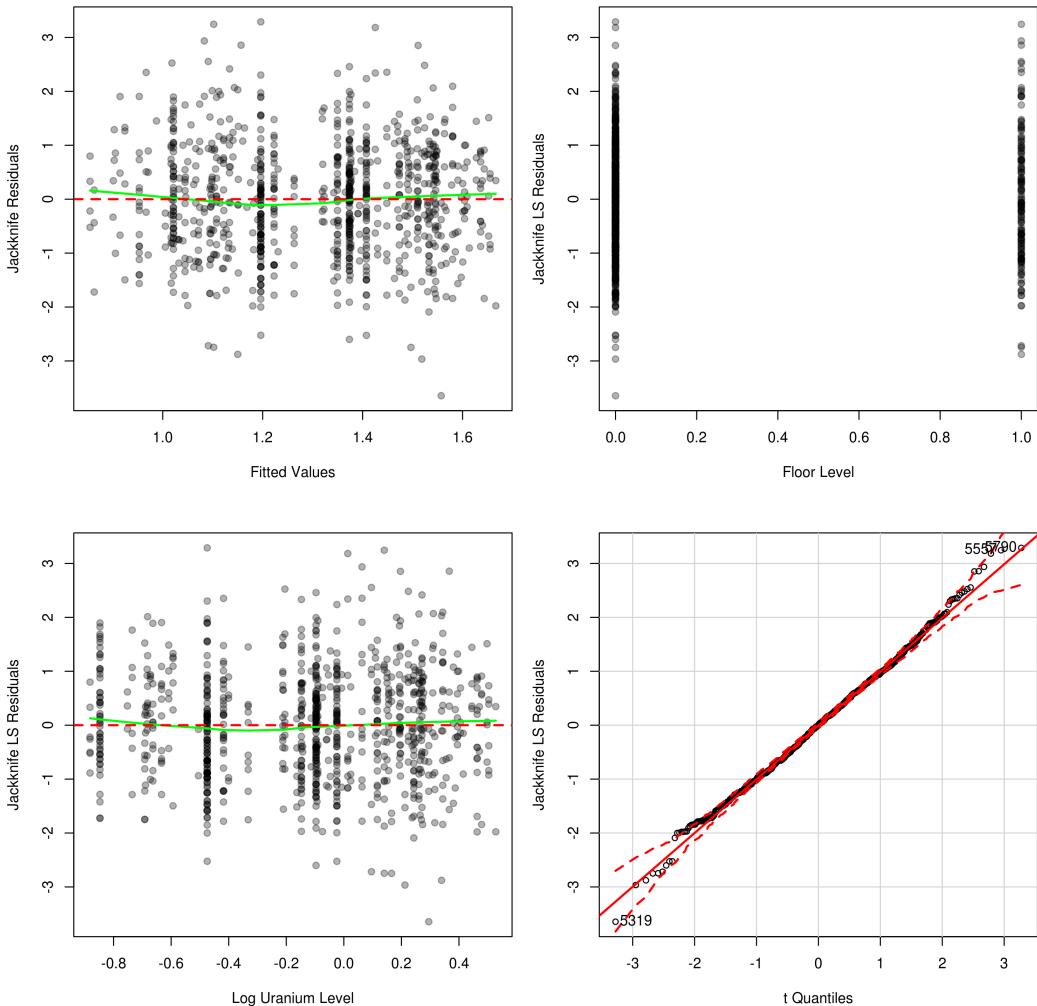


Figure 5.24: Level 1 Least Squares (LS) jackknife residuals from the complete pooling regression using the $\lambda = -0.2$ Box-Cox transformation on a house's measured radon activity. The top left plot shows the jackknife residuals versus fitted values. The top right plot shows the jackknife residuals versus floor of radon level measurement (0 = basement, 1 = first floor). The bottom left plot shows the jackknife residuals versus log uranium levels. The bottom right plot shows the jackknife residuals against the quantiles of the t distribution.

Using the right value of λ for the Box-Cox transformation seems to make most of the problems go away. Heteroskedasticity seems to be the main problem, so let's take a look at the heteroskedasticity robust standard errors and compare them to the homoskedastic standard errors:

```
library(sandwich)
library(lmtest)
```

```

# Get vector of homoskedastic standard errors
homs <- coef(summary(LS.resids3))[, 2]
# Get heteroskedasticity-robust standard errors
coefs <- coeftest(LS.resids3,
                    vcov = vcovHC(LS.resids3, type = "HC0"),
                    df = df.residual(LS.resids3))
hets <- coefs[, 2]
# Get ratios of heteroskedastic to homoskedastic SEs
ratios <- hets/homs

```

For the most part the heteroskedastic and homoskedastic standard errors are within 10% of each other, except for the floor main effect.

At level 1 we seem to be doing pretty well, with little pattern in the fitted versus residual plot, jackknife residuals that are roughly normally distributed (or t distributed with large degrees of freedom), and heteroskedasticity that is not too severe. We can also check for potentially influential observations:

```

png(filename = "ls4iindexplot.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
influenceIndexPlot(LS.resids4, id.n = 5)
dev.off()

```

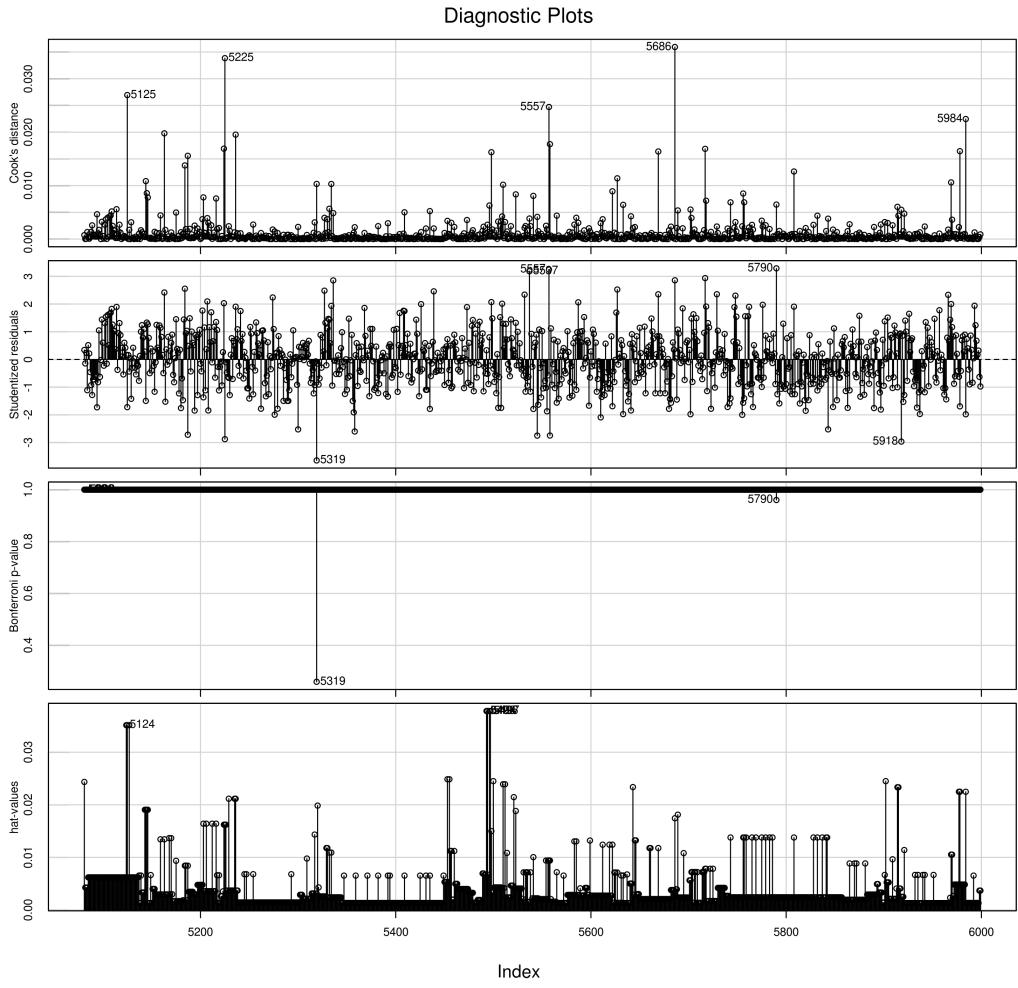


Figure 5.25: Level 1 Least Squares influence index plots. The top plot shows Cook's Distances, followed by the jackknife residuals, then the Bonferonni adjusted p-values, then the hat values for each observation.

The most influential observation is 5686:

```
MNradon["5686", ]
```

This observation has a Cook's D that is 110 times the median Cook's D, a potentially serious problem. We should look into this further, along with examining the other potential influential observations.

5.8.2 Level 2 Empirical Bayes Diagnostics

We now want to check our full multi-level model at the second (group) level, including what we've learned from the first level diagnostics. To do this we

evaluate the distribution of the varying slopes and intercepts. This is sometimes called the *Empirical Bayes* residuals, as the partially pooled estimates are sometimes called Empirical Bayes estimates. First, let's fit the multi-level model using the $\lambda = -0.2$ transformation:

```
vary.slope.0.2 <- lmer(radon0.2 ~ floor + loguranium +
                         floor*loguranium + (1 + floor|county),
                         data = MNradon)
```

Then, let's examine the distribution of the varying intercepts and slopes using histograms:

```
install.packages("HLMdiag", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(HLMdiag)
# Get the empirical Bayes residuals
EB.resids <- HLMresid(vary.slope.0.2, level = "county", type = "EB")
png(filename = "EBhists.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1), mfrow = c(2, 1),
    cex.lab = 1.3, cex.axis = 1.5)
hist(EB.resids[, 1],
     breaks = "FD",
     xlab = "Empirical Bayes Intercepts",
     xlim = c(-0.2, 0.2),
     ylim = c(0, 25),
     main = "")
hist(EB.resids[, 2],
     breaks = "FD",
     xlab = "Emperical Bayes Slopes",
     xlim = c(-0.2, 0.2),
     ylim = c(0, 25),
     main = "")
dev.off()
```

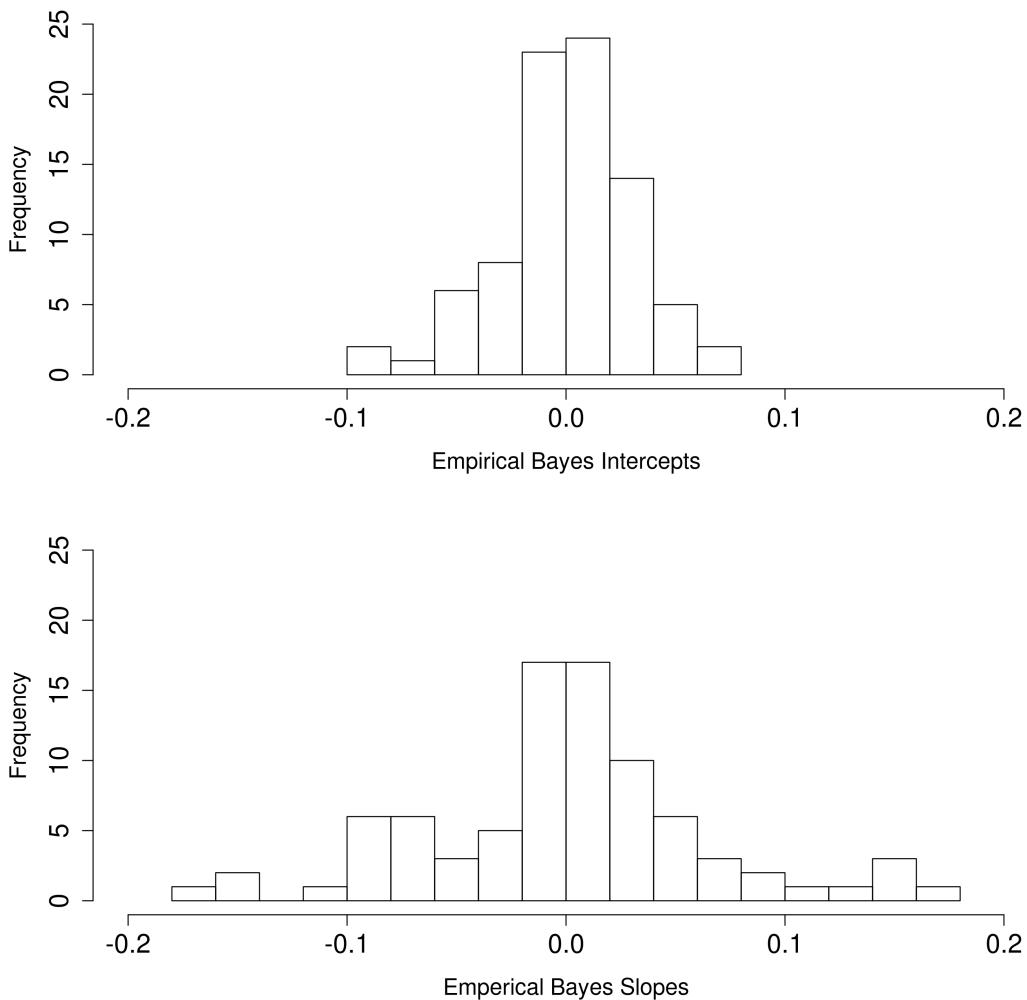


Figure 5.26: Level 2 Empirical Bayes (EB) residual histograms.

The Empirical Bayes residuals look relatively compressed for the intercepts, with a few observations sticking out at the bottom, and relatively spread out for the slopes. Let's use fitted versus residual plots and qqplots to examine the distribution of the Empirical Bayes residuals more carefully:

```
png(filename = "EBresids.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1), mfrow = c(2, 2))
# Plot the log uranium values against the intercept residuals
plot(loguranium, EB.resids[, 1],
      pch = 19,
      col = rgb(0, 0, 0, .5),
      ylim = c(-.1, .1),
      xlab = "County-Level Log Uranium",
```

```

ylab = "Empirical Bayes Intercept Residuals")
abline(h = 0, col = "red", lty = 2)
lines(lowess(loguranium, EB.resids[, 1]),
      col = "green", lwd = 2)
# Plot the log uranium values against the slope residuals
plot(loguranium, EB.resids[, 2],
      pch = 19,
      col = rgb(0, 0, 0, .5),
      xlab = "County-Level Log Uranium",
      ylab = "Empirical Bayes Slope Residuals")
abline(h = 0, col = "red", lty = 2)
lines(lowess(loguranium, EB.resids[, 2]),
      col = "green", lwd = 2)
qqPlot(EB.resids[, 1], id.n = 3,
       ylab = "Empirical Bayes Intercept Residuals",
       xlab = "Normal Quantiles")
qqPlot(EB.resids[, 2], id.n = 3,
       ylab = "Empirical Bayes Slope Residuals",
       xlab = "Normal Quantiles")
dev.off()

```

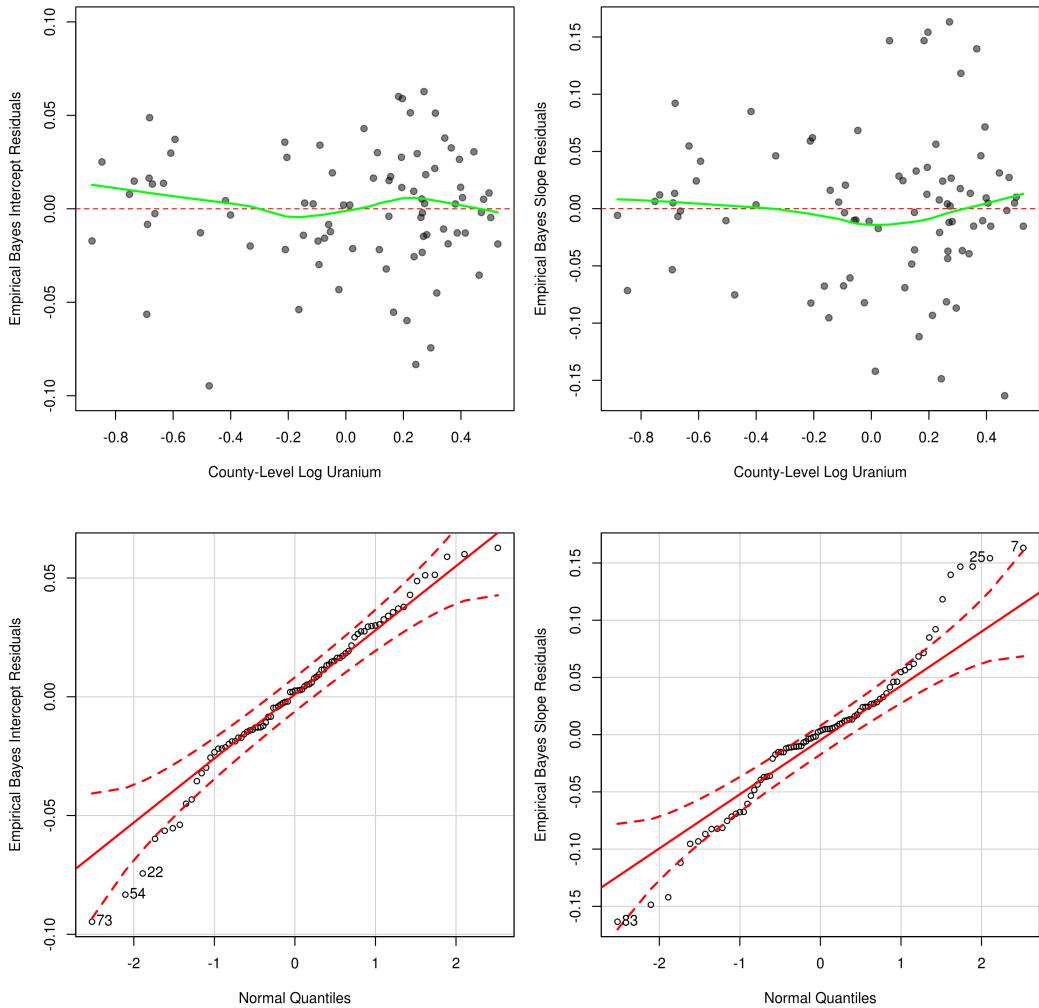


Figure 5.27: Level 2 Empirical Bayes (EB) residual plots. The plots on the left show the relationship between the model's residuals and fitted values (top) and normal distribution (bottom) for county-level intercepts. The plots on the right show the relationship between the model's residuals and fitted values (top) and normal distribution (bottom) for county-level slopes.

It looks like there's a problem with both heteroskedasticity for the log uranium model of the intercepts, and outliers for both the varying slope and varying intercepts. As we saw in the histograms, there's a lumpy lower tail for the Empirical Bayes residuals of the intercepts, with intercepts that are too low relative to the normal distribution. The slopes seem to have lumpy tails on both sides, especially for the larger slopes. To deal with heteroskedasticity at level 2 we'd have to either use a model that allows such heteroskedasticity or model it with other level 2 predictors.

Level 2 Deletion Influence Diagnostics

Just as in the case of level 1 diagnostics, we can evaluate the effect of deleting data on our model's predictions. For level 2 diagnostics, we delete *groups* rather than individual observations:

```
cooks.d <- cooks.distance(vary.slope.0.2, group = "county")
png(filename = "cooksdlm.png", width = 3000, height = 3000, res = 300)
par(cex = 1.3, mar = c(5, 4, 2, 1))
dotplot_diag(x = cooks.d,
              cutoff = "internal",
              name = "cooks.distance",
              ylab = "Cook's Distance",
              xlab = "County")
dev.off()
```

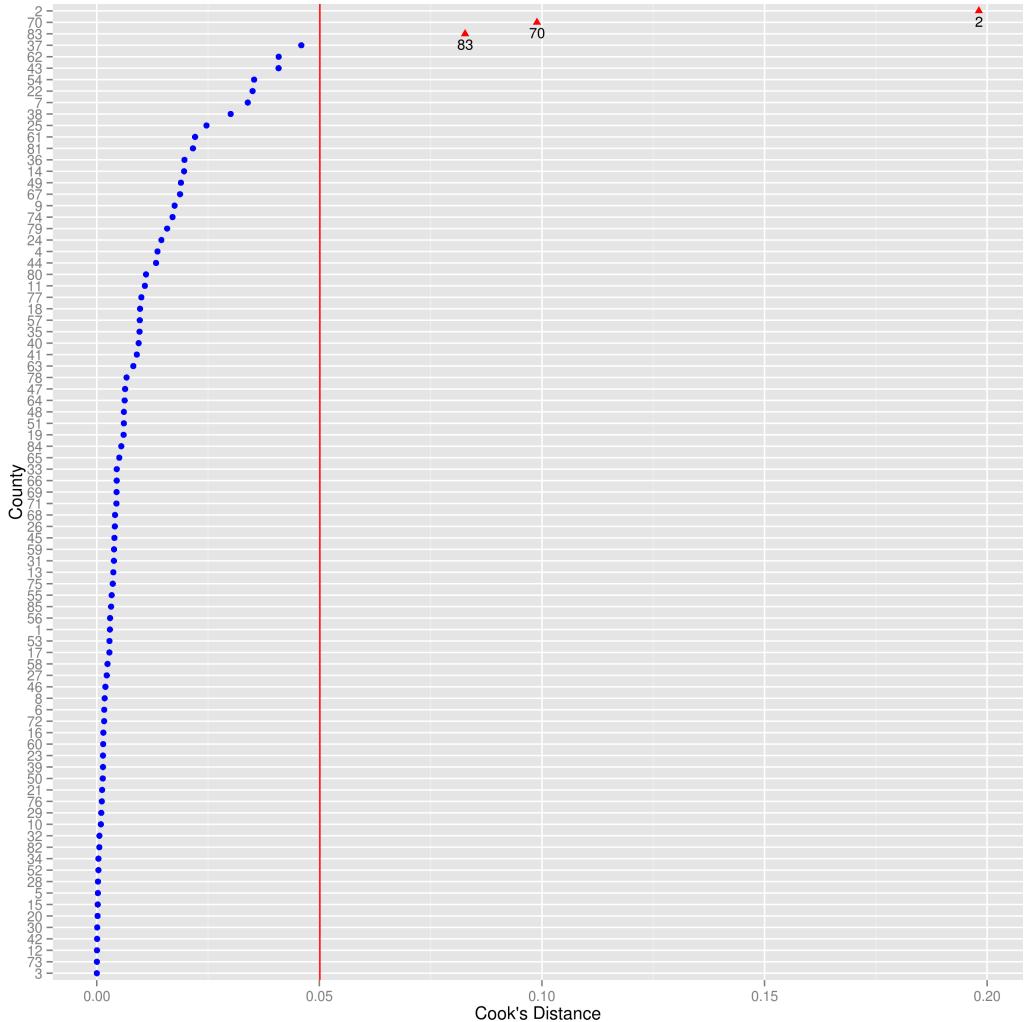


Figure 5.28: Cook's Distance as a function of deleted counties.

This deletion diagnostic indicates that Anoka is the most influential county. To see why Anoka is so influential, let's look at how much the coefficients change when Anoka is removed from the model:

```
# Obtain the change in the fixed effects when Anoka is dropped
beta.change.Anoka <- as.numeric(attr(cooks.d, "beta_cdd")[[2]])
# Label the fixed effects
names(beta.change.Anoka) <- names(fixef(vary.slope.0.2))
beta.change.Anoka
```

When we remove Anoka from the data the coefficient on the floor by loguranium interaction decreases from -0.24 to -0.164. This is a 33% drop in the magnitude of the coefficient, fairly large for a single county. One approach

to handling this would be to treat Anoka as a fixed effect with its own slopes and intercepts, essentially fitting a separate regression model for this county and not including it in the partial pooling. In other words, Anoka would be treated as if it were drawn from a different population than the rest of the counties.

5.9 An Iterative Approximate MLM Procedure

It's not so easy to understand multi-level models, given all the different patterns of correlations that are possible between variables and errors at multiple levels. This makes it difficult to wrap our heads around what is going right and wrong with the model. As a first approximation to our multi-level model, we can start with an iterative process of classical models:

1. Complete pooling: Fit a single complete pooling model. Include group-level predictors but no group-level indicators (dummy variables).
2. Dummy model: Fit a single no-pooling model with group-level indicators (as fixed effects), no group-level predictors, and no model of group-level coefficients.
3. Separate models: Fit a separate regression in each group.
4. Two-step analysis: Use the coefficients of the no pooling or separate models to fit a group-level (level 2) regression.
5. Separate Models: Fit separate regressions in each group again, but use information from the group-level regression to shrink coefficients toward the complete pooling when the sample size is small.

This approach actually maps on to the algorithm for actually fitting multi-level models ([Gelman, 2006](#)).

5.10 The Forecasting Story

5.10.1 Predictions for Existing and New Groups

We can make two types of predictions using our multi-level model: 1) a prediction for a house in a county that we've already considered, and 2) a prediction for a house in a county that we don't yet have data on.

Predictions for Existing Groups

For example, suppose we fit the following varying intercept model:

```
int1 <- lmer(logradon ~ floor + (1|county), data = MNradon)
summary(int1)
```

The resulting regression is:

$$\text{Log-Radon} = \alpha_j - 0.47 \times \text{floor} \quad (5.63)$$

If we have a house in Hennepin County ($j = 26$) where radon was measured on the first floor ($\text{floor} = 1$), then we can use the fitted county intercept and regression function to make predictions for this house:

```
set.seed(60)
x <- 1
# Get the fitted household-level standard deviation
sigma.y.hat <- sigma.hat(int1)$sigma$data
# Get the intercept and slope for county 26
coef.hat <- coef(int1)$county[26, ]
# Calculate the fitted mean for households in county 26
# With measurements on the first floor
mean.hat <- as.numeric(coef.hat[1] + coef.hat[2]*x)
# Simulate predictions
y.pred <- rnorm(1000, mean.hat, sigma.y.hat)
# Get quantiles of the predictions
quants <- quantile(y.pred, c(.025, .5, .975))
```

The median predicted log-radon for houses in Hennepin County is 1.17, with a 95% predictive interval of [0.08, 2.13]. The median and predictive interval on the original scale is 2.22 ($= e^{1.17} - 1$) [0.08, 7.41].

Predictions for New Groups

If we want to make a prediction for a new house in a county in Minnesota that we did not have any observations on, we first need to draw a random intercept for that county from the model of county-level intercepts, then make a prediction for that household within that county:

```
set.seed(62)
# Measurement on the first floor
x <- 1
# Contribution of measurement on the first floor
first <- as.numeric(coef.hat[2]*x)
```

```

# Get standard deviation of the county intercepts
sigma.alpha.hat <- sigma.hat(int1)$sigma$county
# Get the overall county mean
alpha.mean <- as.numeric(fixef(int1)[1])
# Generate a random county intercept
alpha.sim <- rnorm(1000, alpha.mean, sigma.alpha.hat)
# Make prediction for house in new county
# with measurements on the first floor
y.pred.new <- rnorm(1000, alpha.sim + first, sigma.y.hat)
quants.new <- quantile(y.pred.new, c(.025, .5, .975))

```

The median predicted log radon levels are 1.25 [0.06, 2.45], or on the original scale, 2.49 [0.06, 11.59]. As we should probably expect, the predictions for the new group are more uncertain than for an existing group because we don't have an estimate of the county's mean radon levels.

5.10.2 Cross-Validation

Just as in the case of residual diagnostics, we can analyze the forecasting skill of our model both at level 1 and level 2 (and higher levels if we use them). One way to do this is *leave-one-out-cross-validation*, a version of the cross-validation that we're now familiar with where we omit a single observation in our training data, train our model without that observation, then predict that single observation as our test. We then repeat this for each observation in the data. At level 2 we can do something similar by dropping individual *counties* in our training data, then making predictions for that county (Gelman, 2006).

Level 1 Leave-One-Out-Cross-Validation

At level 1 we are doing essentially what we've done previously by sampling two thirds of the data as our training data and predict the remaining third as our test, except here we are only dropping a single observation. The lmer function in R purposefully does not have a predict function, as it is not clear at what level or for what group predictions should be made. Thus, we need to calculate predictions by using the fixed and random effects from the regression directly. We can compare our model with varying slopes to one without using our leave-one-out cross-validation test:

```

simple <- c()
complex <- c()

for(i in 1:nrow(MNradon)){
  # Train complex model dropping the ith observation

```

```

lmer.complex <- lmer(logradon ~ floor + loguranium +
                      floor*loguranium + (1 + floor|county),
                      data = MNradon[-i, ])
# Find the county from the dropped observation
county <- which(rownames(coef(lmer.complex)$county) == MNradon$county[i])
# Let's break down the prediction into each of its parts
# There's the random intercept
rand.int <- coef(lmer.complex)$county[county, 1]
# Plus the county-level uranium prediction of the intercept
uran.int <- coef(lmer.complex)$county[1, 3]*MNradon$loguranium[i]
# Plus the random slope
rand.slope <- coef(lmer.complex)$county[county, 2]*MNradon$floor[i]
# Plus the county-level uranium prediction of the slope
uran.slope <- prod(coef(lmer.complex)$county[1, 4],
                     MNradon$floor[i],
                     MNradon$loguranium[i])
# The prediction is the sum of these four components
pred.comp <- rand.int + uran.int + rand.slope + uran.slope
# Calculate the squared residual
comp.sq.resid <- (MNradon$logradon[i] - pred.comp)^2
# Add the squared residual to the cv vector
complex <- append(complex, comp.sq.resid)

# Train simple model dropping the ith observation
lmer.simple <- lmer(logradon ~ floor + loguranium + (1|county),
                      data = MNradon[-i, ])
# There's the random intercept
rand.int <- coef(lmer.simple)$county[county, 1]
# Plus the county-level uranium prediction of the intercept
uran.int <- coef(lmer.simple)$county[1, 3]*MNradon$loguranium[i]
# Plus the floor of measurement effect
slope <- coef(lmer.simple)$county[county, 2]*MNradon$floor[i]
# The prediction is the sum of these three components
pred.simp <- rand.int + uran.int + slope
# Calculate the squared residual
simp.sq.resid <- (MNradon$logradon[i] - pred.simp)^2
# Add the squared residual to the cv vector
simple <- append(simple, simp.sq.resid)

}

comp.rMSE <- sqrt(sum(complex)/length(complex))

```

```
simp.rMSE <- sqrt(sum(simple)/length(simple))
```

For predictions at the individual house level the root-MSE of the more complex model with both varying intercepts and slopes is 0.562, compared to the root-MSE for the simpler model without varying slopes is 0.559.

Level 2 Leave-One-Out-Cross-Validation

To repeat this for county-level predictions we need to drop counties, fit the model, then make predictions for the omitted county. First we need to make a data frame with the county-level information, where information at the household level (whether the measurement was at the basement or first floor) is at the average county-level. In the case of the floor of measurement regressor, this is the proportion of houses in the county where measurements were made on the first floor:

```
county.floor <- tapply(MNradon$floor, MNradon$county, mean)
county.loguranium <- tapply(MNradon$loguranium, MNradon$county, mean)
county.logradon <- tapply(MNradon$logradon, MNradon$county, mean)
cnty.data <- data.frame(floor = county.floor,
                         loguranium = county.loguranium,
                         logradon = county.logradon)
```

Then we need to train the models on all the data, but make predictions using county-level information:

```
simple <- c()
complex <- c()

for(i in 1:length(unique(MNradon$county))){
  # train model dropping the ith county
  county.drop <- MNradon$county[i]
  # Include all counties except the dropped county
  train.data <- MNradon[!MNradon$county == county.drop, ]
  # Fit the complex model on the training data
  lmer.complex <- lmer(logradon ~ floor + loguranium +
                        floor*loguranium + (1 + floor|county),
                        data = train.data)
  # Get the overall intercept
  int <- fixef(lmer.complex)[1]
  # Multiply floor fixed effect times the
  slope <- fixef(lmer.complex)[2]*cnty.data$floor[i]
  # Multiply by the county-level loguranium
  uran <- fixef(lmer.complex)[3]*cnty.data$loguranium[i]
```

```

# Get the interaction
uran.floor <- prod(fixef(lmer.complex)[4] ,
                     cnty.data$floor[i] ,
                     cnty.data$loguranium[i])
pred.comp <- int + slope + uran + uran.floor
# Calculate the squared residual
comp.sq.resid <- (cnty.data$logradon[i] - pred.comp)^2
# Add the squared residual to the cv vector
complex <- append(complex, comp.sq.resid)
}

complex.rMSE <- sqrt(sum(complex)/length(complex))

```

5.11 Statistical Inference Story

5.11.1 Intraclass Correlation

As we've previously discussed, the variance of the parameters from the complete and no pooling regressions will not be correct when observations are correlated within observational unit. Consider the following regression, where observations i are clustered within group j , and we have regressors x_j that only vary at the group level (e.g., county-level uranium) ([Angrist and Pischke, 2008](#)):

$$y_{ij} = \beta_0 + \beta_1 x_j + \epsilon_{ij} \quad (5.64)$$

We can expect some correlation between the outcomes of observations within the same grouping (observational unit), that is:

$$E[\epsilon_{ij}\epsilon_{lj}] = \rho\sigma_e^2 \quad (5.65)$$

Here σ_e^2 is our familiar residual variance. If $i = l$ then $\rho = 1$ (the observation's residual is perfectly correlated with itself), and $\epsilon_{ij}\epsilon_{lj}$ is just the squared residual, which is equal to σ_e^2 in expectation (i.e., $E[\epsilon_{ij}^2] = \sigma_e^2$). If observations are independent, then there is no grouping structure and $\rho = 0$, meaning $E[\epsilon_{ij}\epsilon_{lj}] = 0$. If $0 < \rho < 1$, then there is some correlation between observations in the same group j . For this reason ρ is called the *Intra-Class Correlation* (ICC). Thus, ρ captures the correlation between grouped or clustered observations.

So far, the way we've modeled this group structure with multi-level models has been to assume the errors for each observation to have *additive* components. For example, we can decompose the error according to two separate terms, the grouping term j and individual term i :

$$\epsilon_{ij} = u_j + u_i \quad (5.66)$$

Here, $u_j \sim N(0, \sigma_\alpha^2)$ the component specific to group j , and $u_i \sim N(0, \sigma_y^2)$ is the idiosyncratic error for the observation i , regardless of grouping j . These are identical to the σ_α and σ_y we've talked about in our pooling discussion. If $\epsilon_{ij} = u_j + u_i$, then the covariance between observation i and l within group j is:²

$$E(\epsilon_{ij}\epsilon_{lj}) = E((u_j + u_i)(u_j + u_l)) \quad (5.67)$$

$$= E(u_j u_j + u_j u_i + u_j u_l + u_i u_l) \quad (5.68)$$

$$= E(u_j^2) \quad (5.69)$$

$$= \text{Var}(u_j) = \sigma_\alpha^2 \quad (5.70)$$

Thus, the covariance between two observations within the same cluster is σ_α^2 . This critically depends on the individual level errors u_i being uncorrelated with the group-level errors u_j so that $E(u_j u_i) = 0$. Similarly, this depends on the assumption that once we account for the group-level errors, the individual observations are uncorrelated with each other, that is $E(u_i u_l) = 0$. And, of course, this depends on the residual variance having the additive decomposition $\epsilon_{ij} = u_j + u_i$.

Continuing, the correlation between two observations is just the covariance divided by the square root of the product of the variances of the two observations. The variance of each observation i is:

$$E(u_j u_j + u_j u_i + u_j u_l + u_i u_l) \quad (5.71)$$

$$= E(u_j^2) + E(u_i^2) \quad (5.72)$$

$$= \text{Var}(y_{ij}) = \sigma_\alpha^2 + \sigma_y^2 \quad (5.73)$$

Giving us the following result for ρ :

$$\text{Corr}(y_{ij}, y_{lj}) = \rho = \frac{\sigma_\alpha^2}{\sqrt{(\sigma_\alpha^2 + \sigma_y^2)(\sigma_\alpha^2 + \sigma_y^2)}} = \frac{\sigma_\alpha^2}{\sigma_\alpha^2 + \sigma_y^2} \quad (5.74)$$

In the numerator we have σ_α^2 , which is analogous to the average sum of squared deviations of each group average from the average of the groups, $\hat{\sigma}_\alpha^2 = \frac{1}{J} \sum_{j=1}^J (\bar{\alpha}_j - \mu_\alpha)^2$. If the group averages are not different from the

²This is an intermediate step, if you're interested:

$$\begin{aligned} \text{Cov}(y_{ij}, y_{lj}) &= E((y_{ij} - E(y_{ij}))(y_{lj} - E(y_{lj}))) \\ &= E(\beta_0 + \beta_1 x_j + \epsilon_{ij} - (\beta_0 + \beta_1 x_j))(\beta_0 + \beta_1 x_j + \epsilon_{lj} - (\beta_0 + \beta_1 x_j)) \\ &= E(\epsilon_{ij}\epsilon_{lj}) \end{aligned}$$

average of the groups, then there is zero between-group variation, and we are not learning anything from using our grouping factor. In this case, $\rho = 0$ and our analysis reduces to the complete pooling estimate.

As σ_α^2 goes to infinity, the group means are very different from each other, and the mean of the groups. Essentially the groups are not comparable and we might as well analyze them separately, or treat each group as a single observation. Thus, $\rho = 1$ and the results reduce to the no pooling analysis.

5.11.2 The Moulton Factor

The variance of the partially pooled estimates, or FGLS estimates, is in general ([Cameron and Miller, 2013](#)):

$$\text{Var}(\hat{\beta}_1)_{pp} = (X' \hat{\Omega}^{-1} X)^{-1} \quad (5.75)$$

However, if the regressor x_1 is the same for all observations within a group (e.g., county-level uranium), and all groups have equal sample size n_j , then the variance of the partial pooled estimate of $\hat{\beta}_1$, let's call it $\text{Var}(\hat{\beta}_1)_{pp}$, is larger than the variance of the complete pooling OLS, let's call it $\text{Var}(\hat{\beta}_1)_{ols}$, by a specific amount:

$$\frac{\text{Var}(\hat{\beta}_1)_{pp}}{\text{Var}(\hat{\beta}_1)_{ols}} = \frac{\sigma_y^2 (X' X)^{-1} [1 + (n_j - 1)\rho]}{\sigma_y^2 (X' X)^{-1}} = 1 + (n_j - 1)\rho \quad (5.76)$$

Where ρ is the intraclass correlation and σ_y^2 is the within group variance. The square root of this ratio $\sqrt{1 + (n_j - 1)\rho}$ is sometimes called the *Moulton Factor* (after the economist Brent Moulton), and reflects the ratio of the partially pooled standard error of $\hat{\beta}_1$ ($\text{Var}(\hat{\beta}_1)_{pp}$) to the homoskedastic standard error ($\text{Var}(\hat{\beta}_1)_{ols}$) ([Moulton, 1990](#)).

As we can see, the Moulton Factor *increases* as the average sample size within each group increases, as well as when the correlation of observations within each group increases. Thus, collecting more data on groups with observations that are highly correlated within groups gets you much less information than an analysis that uses complete pooling with usual standard errors would suggest. For example, comparing a simple complete pooling regression versus partial pooling regression with county-level uranium as a predictor, the standard error of log uranium as a predictor is underestimated by 20% when using complete pooling:

```
cp1 <- lm(radon0.2 ~ loguranium, data = MNradon)
pp1 <- lmer(radon0.2 ~ loguranium + (1|county), data = MNradon)
summary(cp1)
summary(pp1)
coef(summary(pp1))[2, 2]/coef(summary(cp1))[2, 2]
```

In our case, ρ was very small ($.02 = \frac{.0037}{.0037+.1757}$), because county-level uranium accounts for a high level of between-county variation (σ_α^2). Even with such a small ρ , the standard error was substantially inflated. With only a modest intraclass correlation of say $\rho = .1$, large samples within each group can lead to a very large Moulton Factor (e.g., greater than 3) (Angrist and Pischke, 2008). Here's a numerical intercept-only example with equal group sizes and only an intercept:

```
set.seed(2)
beta0 <- 1
# Create 10 random intercepts
alpha.j <- rnorm(10, 0, 1)
group <- rep(seq(1, 10, by = 1), 100)
# Draw observation-level errors
e.y <- rnorm(1000, 0, 1)
y <- beta0 + alpha.j[group] + e.y
data1 <- data.frame(y = y, group = group)
# Complete pooling regression
lm1 <- lm(y ~ 1, data = data1)
# Partial pooling regression
lmer1 <- lmer(y ~ 1 + (1|group), data = data1)
```

Because the variance of the group means is $\sigma_\alpha^2 = 1$, and the residual variance of the observations is $\sigma_y^2 = 1$, the intraclass correlation $\rho = \frac{1}{1+1} = \frac{1}{2}$. The ratio of the standard errors for the partially pooled model, which gets the variance right by taking ρ into account, and the completely pooled model without clustered standard errors, which gets the variance wrong, is 7.21.

```
coef(summary(lmer1))[2]/coef(summary(lm1))[2]
```

From the Moulton factor, we have $n_j = 100$ and $\rho = 1/2$. We should expect the ratio to be $\sqrt{1 + (n_j - 1)\rho} = \sqrt{50} = 7.1$, verifying the simulation. Thus, with an intra-class correlation of 0.5 (i.e., equal variation between and within groups) we underestimate our standard errors by a factor of 7!

5.11.3 The Moulton Factor for Individual Level Regressors

The Moulton Factor changes if the regressor varies at the individual level, rather than just the group level. In this case the Moulton Factor is the square root of:

$$\frac{\text{Var}(\hat{\beta}_1)_{pp}}{\text{Var}(\hat{\beta}_1)_{ols}} = 1 + \left(\frac{\text{Var}(n_j)}{\bar{n}} + \bar{n} - 1 \right) \rho_x \rho \quad (5.77)$$

Where \bar{n} is the average group size and ρ_x is the intraclass correlation of x_{ij} :

$$\rho_x = \frac{\sum_j \sum_{i \neq k} (x_{ij} - \bar{x})(x_{kj} - \bar{x})}{\text{Var}(x_{ij}) \sum_j n_j(n_j - 1)} \quad (5.78)$$

Here, ρ_x is the degree to which the values of the regressor variable x are correlated among different observations within the same cluster. Thus clustering affects the variance estimate more when there is large variance in the sample sizes between groups, and when the intraclass correlations among regressors are high (i.e., observations in the same group have similar levels of the regressor).

5.11.4 Cluster-Robust Standard Errors

It turns out that just like in the case of heteroskedasticity, there is a “robust” standard error that takes grouping (clustering) into account, allowing us to worry less about the Moulton Factor. If the data follow the additive decomposition previously discussed, then the variance estimates of parameters from the partial pooling multi-level model will be the best.

However, it's nice to have some alternative that does not make so many assumptions. When we looked at heteroskedasticity, the robust variance estimate of the coefficient for a regressor allowed a correlation between the squared deviations of the regressor from its mean, $(x_i - \bar{x})^2$, and the squared residuals \hat{u}_i^2 , rather than assuming that they are uncorrelated. That is:

$$\text{Var}(\hat{\beta}_1) = \frac{\sum_{i=1}^n (x_i - \bar{x})^2 \hat{u}_i^2}{(\sum_{i=1}^n (x_i - \bar{x})^2)^2} \quad (5.79)$$

Cluster-robust standard errors allow for this relationship between $(x_i - \bar{x})^2$ and \hat{u}_i^2 , while also allowing for a correlation between the errors of two observations (e.g., people) within the same observational unit (e.g., cities), as well as a correlation between the values of the regressors in the same observational unit. The robust variance-covariance matrix is:

$$\text{Var}(\hat{\beta}) = (X' X)^{-1} \left[\sum_{i=1}^n X_i' u_i u_i' X_i \right] (X' X)^{-1} \quad (5.80)$$

where $u_i = y_i - X_i \beta$ for each observational unit i . This is almost identical to the sandwich estimator previously discussed, except now the meat is different, with each component of the meat being summed for each group. It is also important to note that the center of the sandwich is composed of N elements, rather than $N \times T$, because we are summing for each group. This means that the properties of the robust covariance matrix, for example

asymptotic consistency, rely on the number of *groups* growing large. For a small number of groups it cannot be expected to perform well. If we rewrite:

$$z_1 = \sum_{t=1}^{T_1} u'_{1t} X_{1t} \quad (5.81)$$

Which is the sum over all the observations t in observational unit 1. If we stack these on top of each other we get a vector z which has elements that are the sum of the estimating functions for each observation, repeated for each group i .

$$z = \begin{bmatrix} \sum_{t=1}^{T_1} u'_{1t} X_{1t} \\ \sum_{t=1}^{T_2} u'_{2t} X_{2t} \\ \sum_{t=1}^{T_3} u'_{3t} X_{3t} \\ \vdots \\ \sum_{t=1}^{T_n} u'_{nt} X_{nt} \end{bmatrix} \quad (5.82)$$

It turns out that:

$$z' z = \left[\sum_{i=1}^n X_i' u_i u_i' X_i \right] \quad (5.83)$$

so if we sum our estimating functions for each group then take the dot product we get the same answer as if we take the dot product for each group then sum. Let's run through the computations with a simple example using a complete pooling model with an intercept and floor as regressors:

```
radonreg <- lm(logradon ~ 1 + floor, data = MNradon)
```

First, we specify the clusters or groups:

```
# Specify the clustering variable
MNradon$county <- droplevels(MNradon$county)
cluster <- MNradon$county
```

Next, let's sum the estimating function from the regression over each cluster, using tapply. The estimating functions are provided by the sandwich package ([Zeileis, 2006](#)) and are equal to $u'_{it} X_{it} = (y_{it} - X_{it}\beta)X_{it}$:

```
# The sandwich library gives us a couple handy functions
library(sandwich)
# Sum the (y-b.hat*x)*x over each cluster for each column
z <- apply(estfun(radonreg), 2,
           function(x) tapply(x, cluster, sum))
meat <- t(z) %*% z
```

Next, we get the “bread” or $(X'X)^{-1}$:

```
# Get the "bread" which is n*(x'x)^-1 so we need to divide by n
bread <- bread(radonreg)/nrow(MNradon)
```

Finally, we use the sandwich formula to get the variance-covariance matrix of $\hat{\beta}$:

```
# Use the sandwich formula to get the cluster-robust variance
sandwich.variance <- (bread %*% meat %*% bread)
```

We can get the cluster-robust standard errors and test statistics by specifying the variance covariance matrix (vcov) in the coeftest function:

```
# Robust variance matrix
library(lmtest)
coeftest(radonreg, vcov = sandwich.variance)

# Robust vcov matrix using cluster.vcov
# install.packages("multiwayvcov")
library(multiwayvcov)
vcovCL <- cluster.vcov(radonreg,
                        MNradon$county,
                        df_correction = FALSE)
coeftest(radonreg, vcovCL)

# OLS standard errors
summary(radonreg)
```

The cluster-robust standard error of the intercept is 0.05, about twice as large as the standard error assuming homoskedasticity from the complete pooling regression (0.02). There's a degrees of freedom correction that we turned off, so we should leave it on for the rest of our computations.

Like all robust methods, cluster-robust standard errors have a positive and negative side. They are good because we are pretty sure that our model is misspecified in some way; we are unlikely to get the exact forms of heteroskedasticity, functional form misspecification, and clustering correct, so it's nice to have a fallback. On the bad side, the tendency is to be lazy about understanding our data. Rather than trying to model the correlation between observations within observational units, we can get standard errors that are “robust” to any such correlation. Using robust standard errors in some sense implies that we don't really care about trying to understand what is making our variance estimate wrong. Instead, we just “robust” the problem away.³

³Ed Leamer calls the use of robust standard errors “Whitewashing” the problem of model misspecification, as robust standard errors are named after Halbert White, although this is undoubtedly unfair to the late Professor White ([Leamer, 2010](#)).

Instead of sweeping the standard errors under the table, we can use heteroskedasticity-robust standard errors, cluster-robust standard errors, and partial pooling to diagnose model misspecifications. Consider the following regression:

```
radonreg5.cp <- lm(logradon ~ floor + loguranium, data = MNradon)
```

Examining the cluster-robust standard errors will tell us how bad the variance of our parameter estimates are if we don't take grouping into account:

```
# Get robust standard errors
robust <- sqrt(cluster.vcov(radonreg5.cp, MNradon$county))
# Get the classical homoskedastic standard errors
classical <- coef(summary(radonreg5.cp))

# Get the Moulton Factor for the intercept
robust[1, 1]/classical[1, 2]
# Get the Moulton Factor for the floor dummy variable
robust[2, 2]/classical[2, 2]
# Get the Moulton Factor for the loguranium variable
robust[3, 3]/classical[3, 2]
```

The cluster-robust standard errors are 20% larger for the intercept, 12% larger for the floor independent variable, and 41% larger for the loguranium independent variable, all indicating some sort of model misspecification. Recall that cluster-robust standard errors pick up both heteroskedasticity and within-cluster correlations. Let's try to pick out the contribution of heteroskedasticity to the variance estimates:

```
het <- coeftest(radonreg5.cp,
                 vcov = vcovHC(radonreg5.cp, type = "HC0"),
                 df = df.residual(radonreg5.cp))
# Check for heteroskedasticity for the intercept
het[1, 2]/classical[1, 2]
# Check for heteroskedasticity for the floor dummy variable
het[2, 2]/classical[2, 2]
# Check for heteroskedasticity for the uranium variable
het[3, 2]/classical[3, 2]
```

The heteroskedasticity-robust standard errors are almost identical to the classical homoskedastic standard errors, except for the floor dummy variable. This suggests that the inflation of standard errors is due to clustering for the intercept and uranium regressors, but heteroskedasticity for the floor dummy variable. Partial pooling will take the additive components form of clustering

into account. If that holds, then the standard errors for the partial pooling model should be similar to the cluster-robust standard errors, except for the floor dummy variable, as our multi-level model does not take heteroskedasticity into account:

```
radonreg5.pp <- lmer(logradon ~ floor + loguranium + (1|county),
                      data = MNradon)
summary(radonreg5.pp)
```

Let's compare the standard errors from the partial pooling model to the cluster-robust standard errors:

```
# Partially pooled standard errors
partial.pool <- coef(summary(radonreg5.pp))

robust[1, 1]/partial.pool[1, 2]
robust[2, 2]/partial.pool[2, 2]
robust[3, 3]/partial.pool[3, 2]
```

As expected, there is a difference of almost the same 12% between the partial pooling and cluster-robust standard errors for the floor regressor, confirming that this is due to heteroskedasticity. For the intercept and uranium regressors the difference between the partial pooling and cluster-robust standard errors is less than 10%. To be conservative we might use complete pooling with cluster-robust standard errors, however, it is important to keep in mind that cluster-robust standard errors are not unbiased, but instead asymptotically consistent. Even more importantly, they are consistent *as the number of groups* grows large, not the number of observations. Thus, it could be the case that the difference between the robust and partial pooling standard errors reflects bias.

An additional way of checking the cluster-robust and partial pooling standard errors would be to use the *block bootstrap* or *simple cluster bootstrap*, where we bootstrap but instead of sampling from individual observations with replacement, we sample from groups (or alternatively blocks/clusters) with replacement. Such an approach can provide an additional comparison to the partial pooling and cluster-robust estimates.

```
intercept <- c()
floor <- c()
log.u <- c()

for(i in 1:1000){
  # Sample counties
  cnty.samps <- sample(unique(MNradon$county),
```

```

    size = length(unique(MNradon$county)),
    replace = T)
# Create a data frame with the sampled counties
cnty <- data.frame(county = cnty.samps)
# Create a new data frame with the correct data for each county
sample <- merge(cnty, MNradon, by = "county")
# Run the regression on the bootstrap sample
lmer.boot <- lmer(logradon ~ floor + loguranium + (1|county),
                    data = sample)
# Get the regressin parameters
intercept <- append(intercept, fixef(lmer.boot)[1])
floor <- append(floor, fixef(lmer.boot)[2])
log.u <- append(log.u, fixef(lmer.boot)[3])
}

# Look at the standard errors
sd(intercept)
sd(floor)
sd(log.u)

```

We can summarize the standard errors from the different approaches, taking into account the various types of misspecification they take into account:

Approach	Heteroskedasticity	ρ	ρ_x
OLS			
Partial Pool		Y	
Het	Y		
Cluster	Y	Y	Y
Bootstrap	Y	Y	Y

It's possible to try to eliminate some possible causes of different standard errors using this approach. For example, if partial pooling and cluster-robust standard errors do not match, but cluster-robust standard errors and heteroskedastic standard errors match, then the problem is likely heteroskedasticity. A difference between clustering and the bootstrap may indicate model misspecification, rather than just misspecification of the variance estimate, or that we have too few groups to get accurate variance estimates using either of these approaches. Here's the comparison of these different variance estimation approaches for our previous regression:

For the intercept estimate, the partial pooling, cluster-robust, and bootstrap standard errors are close to each other, indicating agreement that the change in the variance of this parameter is due to the intraclass correlation.

Parameter	OLS	Partial Pool	Het	Cluster	Bootstrap
Intercept	0.021	0.027	0.021	0.026	0.025
Floor	0.050	0.050	0.055	0.056	0.061
Loguranium	0.051	0.066	0.049	0.071	0.057

Table 5.2: Standard errors for the intercept, floor, and loguranium regressors using homoskedastic standard errors (OLS), partial pooling (Partial Pool), heteroskedastic standard errors (Het), cluster-robust standard errors (Cluster), and block bootstrap (Bootstrap).

For the floor estimate, the heteroskedastic and cluster-robust standard errors are very close, and greater than the partial pooling standard error, indicating that heteroskedasticity is the main problem, although the block bootstrap is picking up something else. Lastly, for the loguranium estimate there doesn't seem to be heteroskedasticity at the individual level, but there is a fair amount of disagreement between the partial pooling, cluster-robust, and bootstrap standard errors. This would probably be worth looking into.

We can also look at confidence intervals based on the quantiles of the bootstrap distribution, as the distribution of the floor estimates is skewed:

```
quantile(intercept, c(.025, .5, .975))
quantile(floor, c(.025, .5, .975))
quantile(log.u, c(.025, .5, .975))
```

In sum, a difference between cluster-robust variance estimate and the partial pooling variance estimate can signal a model misspecification, which can be used to improve either model. We can use heteroskedasticity-robust standard errors to try to determine whether such a difference is due to heteroskedasticity, the intraclass correlation of the regressor (i.e., whether the regressor is correlated within-cluster), or both. Specifying heteroskedasticity in multi-levels in R using the lmer function is not possible, so we'd have to use either block bootstrap or a more complex approach, such as Bayesian models using Gibbs sampling (in BUGS for example).

5.12 Homework 5

Read the introduction, data, and total examination score sections of [Goldstein et al. \(1993\)](#). Don't bother replicating their results, as we do not have the same data shown in the paper.⁴ The data can be obtained here:

```
install.packages("mlmRev", repos = "http://lib.stat.cmu.edu/R/CRAN/")
library(mlmRev)
data("Exam", package = "mlmRev")
head(Exam)
```

The variables in the data are:

- *school*: the student's school
- *normexam*: the score on the General Certificate of Secondary Education (GCSE) examination at age 16
- *schgend*: school gender type (mixed, boys only, girls only)
- *schavg*: average London Reading Test intake score for the school
- *vr*: verbal reasoning subscore of the London Reading Test at intake
- *intake*: overall London Reading Test score at intake
- *standLRT*: standardized intake score on the London Reading Test at age 11
- *sex*: the student's gender
- *type*: whether the school is mixed or single gender
- *student*: student ID within school (not unique)

Write a short report telling the five stories of these data. Here's what I expect to be included in the report:

1. Create histograms of the number of students in each school, the scores on the GCSE exams, the average London Reading Test intake scores for the schools, and the standardized London Reading Test intake scores, summarizing what you see. Next, make tables of the school gender types, verbal reasoning subscores at intake, overall London Reading Test scores at intake, the student's gender, and the type of schools, summarizing what you see. (1 point)

⁴In fact, it was pretty disappointing to be unable to find a good multi-level model dataset worth replicating, due to both poor documentation and data sharing.

2. Conduct complete, no, and partial pooling regressions of the GCSE exam scores using only an intercept, summarizing the results. Examine the distribution of the no pooling and partial pooling intercepts. How are they similar or different? Provide the fitted intercept for an example school using no pooling and partial pooling. (1 point)
3. For the previous regression, explain when the partially pooled school intercepts will be closer to complete pooling or no pooling. (1 point)
4. Add the student's standardized intake score on the London Reading Test at age 11 as a predictor to the complete, no, and partial pooling regressions. How do the intercepts change as the predictor is added? How do the complete, no, and partial pooling regressions compare? Should we be concerned about any correlation between the students' standardized intake score and the school level intercepts (if there is any correlation)? (1 point)
5. Add the average London Reading Test intake score for the school as a group-level predictor in a multi-level model. How does the variance of the intercepts change? Plot the regression line of the average London Reading Test scores against the no pooled and partially pooled intercepts, summarizing what you see. Write the fitted regression line for an example school. (1 point)
6. Extend the previous model by allowing the standardized intake score to vary at the school level, summarizing the results. Write the fitted regression line for an example school. Interpret the variance components of the new model. (1 point)
7. Use level 1 diagnostics to discover any problems with the first level of the previous model. Allow for potential transformations of the dependent and independent variables. What revised model do you suggest? (1 point)
8. Use level 2 diagnostics to discover problems with the second level of the previous model. Do the Empirical Bayes residuals look appropriate? Are there any potentially influential schools? (1 point)
9. Compare a no pooling dummy variable version of your revised model with a partial pooling version. How do the standard errors of your no pooling model change when using heteroskedasticity and cluster-robust standard errors? What does this suggest about potential model misspecification? Should we use our partial pooling or no pooling model? (1 point)

10. Compare your model to a simpler version using level 1 and level 2 leave-one-out-cross-validation. Summarize their performance. Which model do you prefer? (1 point)
11. Explain why econometricians, statisticians, and machine learning researchers are so hungry.⁵ (1 bonus point)

Here's a list of functions you might need:

- `hist()`
- `table()`
- `tapply()`
- `gam()` (mgcv package)
- `boxCox()` (car package)
- `boxTidwell()` (car package)
- `qqPlot()` (car package)
- `lmer()` (arm package)
- `fixef()` (lme4 package)
- `ranef()` (lme4 package)
- `cl.robust()` (lecture notes/Arai)
- `vcovHC` (sandwich)
- `coeftest` (lmtest)
- `HLMresid` (HLMdiag)
- `cooks.distance` (HLMdiag)

Make sure your report includes reproducible R code, either as an appendix, as a footnote through Harvard's Dataverse (or some other site of your choice), or through CMU's dropbox.

⁵For more background check out the Chinese Restaurant Process and Indian Buffet Processes, as well as the newly discovered Fried Chicken Bucket process <http://sigbovik.org/2007/papers/proceedings.pdf>

Bibliography

- Agresti, A. (2013). *Categorical data analysis*. John Wiley & Sons. 401
- Albert, A. and Anderson, J. (1984). On the existence of maximum likelihood estimates in logistic regression models. *Biometrika*, 71(1):1–10. 400
- Altman, D., Schulz, K., and Moher, D. (2004). Turning a blind eye: Testing the success of blinding and the consort statement. *British Medical Journal*, 328(7448):1135. 162
- Amemiya, T. (1971). The estimation of the variances in a variance-components model. *International Economic Review*, pages 1–13. 492
- Anglin, P. M. and Gencay, R. (1996). Semiparametric estimation of a hedonic price function. *Journal of Applied Econometrics*, 11(6):633–648. 305
- Angrist, J. D. and Pischke, J.-S. (2008). *Mostly harmless econometrics: An empiricist’s companion*. Princeton university press. 513, 516
- Anscombe, F. J. (1973). Graphs in statistical analysis. *The American Statistician*, 27(1):17–21. 52, 124
- Baltagi, B. (2008). *Econometric analysis of panel data*, volume 1. John Wiley & Sons. 484
- Bates, D. M. (2010). lme4: Mixed-effects modeling with r. URL <http://lme4.r-forge.r-project.org/book>. 482
- Behnken, D. W. and Draper, N. R. (1972). Residuals and their variance patterns. *Technometrics*, 14(1):101–111. 114, 116
- Berk, R. A. (2004). *Regression analysis: A constructive critique*. Sage Publications: Los Angeles, CA. 36, 51, 52, 109, 147, 148, 163
- Berk, R. A. (2008). *Statistical learning from a regression perspective*. Springer. 51

- Berk, R. A. and Freedman, D. A. (2003). Statistical assumptions as empirical commitments. In Bloomberg, T. and Cohen, S., editors, *Law, Punishment, and Social Control: Essays in Honor of Sheldon Messinger*, pages 235–254. Transaction Books. [147](#), [148](#)
- Bierens, H. J. and Ginther, D. K. (2001). Integrated conditional moment testing of quantile regression models. *Empirical Economics*, 26(1):307–324. [166](#), [167](#)
- Bland, J. and Altman, D. (2000). Statistics notes: The odds ratio. *British Medical Journal*, 320(7247):1468. [311](#)
- Box, G. E. and Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 211–252. [255](#)
- Box, G. E. and Tidwell, P. W. (1962). Transformation of the independent variables. *Technometrics*, 4(4):531–550. [221](#), [296](#)
- Breiman, L. (1992). The little bootstrap and other methods for dimensionality selection in regression: X-fixed prediction error. *Journal of the American Statistical Association*, 87(419):738–754. [139](#)
- Breiman, L. and Spector, P. (1992). Submodel selection and evaluation in regression. the x-random case. *International statistical review/revue internationale de Statistique*, pages 291–319. [139](#)
- Buchinsky, M. (1994). Changes in the U.S. wage structure 1963-1987: Application of quantile regression. *Econometrica*, pages 405–458. [191](#)
- Cameron, A. C. and Miller, D. L. (2013). A practitioner’s guide to cluster-robust inference. [515](#)
- Cohen, J. (1994). The earth is round (p_i. 05). *American Psychologist*, 49(12):997–1003. [35](#)
- Cohen, P., Cohen, J., West, S. G., and Aiken, L. S. (2002). *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Hillsdale, NJ: Erlbaum. [194](#), [255](#), [302](#)
- Cook, R. D. (1993). Exploring partial residual plots. *Technometrics*, 35(4):351–362. [219](#)
- Draper, N. R., Smith, H., and Pownell, E. (1966). *Applied regression analysis*, volume 3. Wiley New York. [29](#), [115](#), [304](#)
- Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, pages 1–26. [154](#)

- Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*. New York, NY: Chapman & Hall. 154
- Faraway, J. J. (2004). *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*. CRC press. 482
- Feynman, R. (1974). Cargo cult science. *Engineering and Science*, 37(7):10–13. 22
- Gelman, A. (2006). Multilevel (hierarchical) modeling: What it can and cannot do. *Technometrics*, 48(3):432–435. 508, 510
- Gelman, A. (2011). Why tables are really much better than graphs. *Journal of Computational and Graphical Statistics*, 20(1):3–7. 52
- Gelman, A. (2013). Ethics and statistics: Is it possible to be an ethicist without being mean to people? *Chance*, 26(4):52–55. 23, 27
- Gelman, A. and Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. New York, NY: Cambridge University Press. 103, 198, 361, 427, 441, 442, 482
- Gelman, A. and Shalizi, C. (2013). Philosophy and the practice of Bayesian statistics. *British Journal of Mathematical and Statistical Psychology*, 66(1):8–38. 51
- Gerfin, M. (1996). Parametric and semi-parametric estimation of the binary response model of labor market participation. *Journal of Applied Econometrics*, 11(3):321–39. 417
- Goldstein, H., Rasbash, J., Yang, M., Woodhouse, G., Pan, H., Nuttall, D., and Thomas, S. (1993). A multilevel analysis of school examination results. *Oxford Review of Education*, 19(4):425–433. 524
- Gopnik, A., Glymour, C., Sobel, D., Schulz, L., Kushnir, T., and Danks, D. (2004). A theory of causal learning in children: Causal maps and bayes nets. *Psychological Review*, 111(1):3–32. 160
- Gøtzsche, P. (1996). Blinding during data analysis and writing of manuscripts. *Controlled Clinical Trials*, 17(4):285–290. 25
- Greene, W. H. (2008). *Econometric analysis*. Granite Hill Publishers. 491
- Grimes, D. A. and Schulz, K. F. (2008). Making sense of odds and odds ratios. *Obstetrics & Gynecology*, 111(2):423–426. 321

- Gu, C. (2013). *Smoothing spline ANOVA models*, volume 297. Springer Science & Business Media. [297](#)
- Haahr, M. and Hróbjartsson, A. (2006). Who is blinded in randomized clinical trials? *The Cochrane Collaboration Methods Groups Newsletter*, 3:14. [162](#)
- Halvorsen, R. and Palmquist, R. (1980). The interpretation of dummy variables in semilogarithmic equations. *American Economic Review*, 70(3):474–75. [264](#)
- Hastie, T., Tibshirani, R., and Friedman, J. (2004). The elements of statistical learning: Data mining, inference, and prediction. *BeiJing: Publishing House of Electronics Industry*. [142](#)
- Hosmer, D., Lemeshow, S., and Sturdivant, R. (2013). *Applied Logistic Regression*. John Wiley Sons: New York. [379](#), [406](#), [411](#)
- Hosmer, D. W., Hosmer, T., Le Cessie, S., Lemeshow, S., et al. (1997). A comparison of goodness-of-fit tests for the logistic regression model. *Statistics in medicine*, 16(9):965–980. [382](#)
- Judd, C. M. and McClelland, G. H. (1989). *Data analysis: A model comparison approach*. Routledge. [235](#)
- King, G. (2007). An introduction to the Dataverse Network as an infrastructure for data sharing. *Sociological Methods & Research*, 36(2):173–199. [36](#)
- King, G. and Roberts, M. (2013). How robust standard errors expose methodological problems they do not fix. [286](#)
- Kleiber, C. and Zeileis, A. (2008). *Applied econometrics with R*. Springer. [401](#)
- Lancaster, T. (2000). The incidental parameter problem since 1948. *Journal of Econometrics*, 95(2):391–413. [441](#)
- Landwehr, J. M., Pregibon, D., and Shoemaker, A. C. (1984). Graphical methods for assessing logistic regression models. *Journal of the American Statistical Association*, 79(385):61–71. [390](#)
- Leamer, E. (1983). Let's take the con out of econometrics. *The American Economic Review*, 73(1):31–43. [25](#)
- Leamer, E. E. (2010). Tantalus on the road to asymptopia. *The Journal of Economic Perspectives*, 24(2):31–46. [519](#)

- Loy, A. and Hofmann, H. (2014). Hlmdiag: A suite of diagnostics for hierarchical linear models in r. *Journal of Statistical Software*, 56(5). [493](#)
- Ludwig, J., Duncan, G. J., Gennetian, L. A., Katz, L. F., Kessler, R. C., Kling, J. R., and Sanbonmatsu, L. (2012). Neighborhood effects on the long-term well-being of low-income adults. *Science*, 337(6101):1505–1510. [47](#)
- Maddala, G. S. (1971). The use of variance components models in pooling cross section and time series data. *Econometrica: Journal of the Econometric Society*, pages 341–358. [486](#)
- Mahoney, M. (1977). Publication prejudices: An experimental study of confirmatory bias in the peer review system. *Cognitive Therapy and Research*, 1(2):161–175. [23](#)
- Moulton, B. R. (1990). An illustration of a pitfall in estimating the effects of aggregate variables on micro units. *The Review of Economics and Statistics*, pages 334–338. [515](#)
- Neuroskeptic (2012). The nine circles of scientific hell. *Perspectives on Psychological Science*, 7(6):643–644. [23](#)
- Nosek, B. A., Spies, J. R., and Motyl, M. (2012). Scientific utopia II: Restructuring incentives and practices to promote truth over publishability. *Perspectives on Psychological Science*, 7(6):615–631. [27](#)
- Prasad, K., Jaeschke, R., Wyer, P., Keitz, S., and Guyatt, G. (2008). Tips for teachers of evidence-based medicine: Understanding odds ratios and their relationship to risk ratios. *Journal of General Internal Medicine*, 23(5):635–640. [311](#)
- Ramsey, J. B. (1969). Tests for specification errors in classical linear least-squares regression analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 350–371. [382](#)
- Sackett, D. L., Deeks, J. J., and Altman, D. G. (1996). Down with odds ratios! *Evidence Based Medicine*, 1(6):164–166. [311](#)
- Schulz, K. (1995a). Subverting randomization in controlled trials. *Journal of the American Medical Association*, 274(18):1456–1458. [162](#)
- Schulz, K. (1995b). Unbiased research and the human spirit: The challenges of randomized controlled trials. *Canadian Medical Association Journal*, 153(6):783–786. [162](#)

- Schulz, K. (1996). Randomised trials, human nature, and reporting guidelines. *The Lancet*, 348(9027):596–598. [162](#)
- Schulz, K., Chalmers, I., Hayes, R., and Altman, D. (1995). Empirical evidence of bias. *Journal of the American Medical Association*, 273(5):408–412. [161](#)
- Schulz, K. and Grimes, D. (2002a). Allocation concealment in randomised trials: Defending against deciphering. *The Lancet*, 359(9306):614–618. [162](#)
- Schulz, K. and Grimes, D. (2002b). Generation of allocation sequences in randomised trials: Chance, not choice. *The Lancet*, 359(9305):515–519. [162](#)
- Schulz, K., Grimes, D., Altman, D., and Hayes, R. (1996). Blinding and exclusions after allocation in randomised controlled trials: Survey of published parallel group trials in obstetrics and gynaecology. *British Medical Journal*, 312(7033):742–744. [162](#)
- Schwartz, D., Fischhoff, B., Krishnamurti, T., and Sowell, F. (2013). The Hawthorne Effect and energy awareness. *Proceedings of the National Academy of Sciences*, In Press. [163](#)
- Simmons, J., Nelson, L., and Simonsohn, U. (2011). False-positive psychology. *Psychological Science*, 22(11):1359–1366. [26](#), [27](#), [31](#), [33](#), [38](#)
- Stodden, V. C. (2009). Enabling reproducible research: Open licensing for scientific innovation. *International Journal of Communications Law and Policy*, 13:1–25. [36](#)
- Stodden, V. C. (2011). Trust your science? Open your data and code. *Amstat News*, 409:21–22. [36](#)
- Stukel, T. A. (1988). Generalized logistic models. *Journal of the American Statistical Association*, 83(402):426–431. [382](#)
- Tse, T., Williams, R., and Zarin, D. (2009). Reporting basic results in clinicaltrials.gov. *Chest*, 136(1):295–303. [25](#)
- Turner, E., Matthews, A., Linardatos, E., Tell, R., and Rosenthal, R. (2008). Selective publication of antidepressant trials and its influence on apparent efficacy. *New England Journal of Medicine*, 358(3):252–260. [27](#)
- Tversky, A. and Kahneman, D. (1977). Causal schemata in judgments under uncertainty. Technical report, DTIC Document. [160](#)
- Wackerly, D. D., Mendenhall, W., and Scheaffer, R. L. (2008). *Mathematical statistics with applications*. Cengage Learning. [29](#), [40](#), [165](#)

- Wand, M. (1997). Data-based choice of histogram bin width. *The American Statistician*, 51(1):59–64. [56](#)
- Weisberg, S. (2001). Yeo-johnson power transformations. *Department of Applied Statistics, University of Minnesota*. Retrieved June, 1:2003. [256](#)
- White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48(4):817–838. [291](#)
- Wood, S. (2006). *Generalized additive models: An introduction with R*. CRC press. [227](#), [230](#), [297](#), [298](#), [301](#)
- Wooldridge, J. (2002). *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT press. [442](#)
- Wooldridge, J. (2009). *Introductory econometrics: A Modern Approach*. Nashville, TN: South-Western Publishers. [193](#), [264](#), [268](#), [282](#), [486](#)
- Yatchew, A. and Griliches, Z. (1985). Specification error in probit models. *The Review of Economics and Statistics*, 67(1):134–39. [368](#)
- Zeileis, A. (2006). Object-oriented computation of sandwich estimators. *Journal of Statistical Software*, 16(9):1–16. [280](#), [518](#)