An Introduction to the R Statistical Computing Environment

John Fox

McMaster University

ICPSR 2021

Outline

Outline

- Defining a dummy-variable regressor for a dichotomous explanatory variable — e.g., gender in the regression of income Y on gender and education X.

- Defining a dummy-variable regressor for a dichotomous explanatory variable — e.g., gender in the regression of income Y on gender and education X.
- Let $D = 0$ for women and $D = 1$ for men.

- Defining a dummy-variable regressor for a dichotomous explanatory variable — e.g., gender in the regression of income Y on gender and education X.
- Let $D = 0$ for women and $D = 1$ for men.
- Then the additive dummy-regression model is

$$Y = \alpha + \beta X + \gamma D + \varepsilon$$

- Defining a dummy-variable regressor for a dichotomous explanatory variable — e.g., gender in the regression of income Y on gender and education X.
- Let $D = 0$ for women and $D = 1$ for men.
- Then the additive dummy-regression model is

$$Y = \alpha + \beta X + \gamma D + \varepsilon$$

- So, for women (treating X as conditionally fixed)

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 0 + \varepsilon \\ E(Y) &= \alpha + \beta X \end{aligned}$$

- Defining a dummy-variable regressor for a dichotomous explanatory variable — e.g., gender in the regression of income Y on gender and education X.
- Let $D = 0$ for women and $D = 1$ for men.
- Then the additive dummy-regression model is

$$Y = \alpha + \beta X + \gamma D + \varepsilon$$

- So, for women (treating X as conditionally fixed)

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 0 + \varepsilon \\ E(Y) &= \alpha + \beta X \end{aligned}$$

- And, for men

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 1 + \varepsilon \\ E(Y) &= (\alpha + \gamma) + \beta X \end{aligned}$$

- Defining a dummy-variable regressor for a dichotomous explanatory variable — e.g., gender in the regression of income Y on gender and education X.
- Let $D = 0$ for women and $D = 1$ for men.
- Then the additive dummy-regression model is

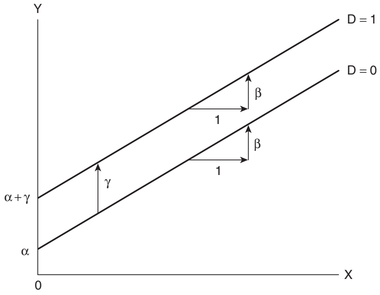
$$Y = \alpha + \beta X + \gamma D + \varepsilon$$

- So, for women (treating X as conditionally fixed)

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 0 + \varepsilon \\ E(Y) &= \alpha + \beta X \end{aligned}$$

- And, for men

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 1 + \varepsilon \\ E(Y) &= (\alpha + \gamma) + \beta X \end{aligned}$$



- Defining a dummy-variable regressor for a dichotomous explanatory variable — e.g., gender in the regression of income Y on gender and education X.
- Let $D = 0$ for women and $D = 1$ for men.
- Then the additive dummy-regression model is

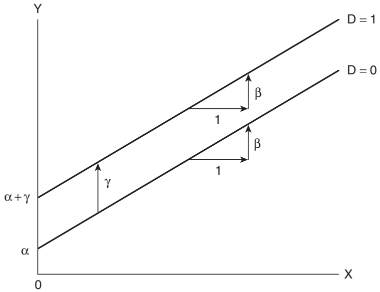
$$Y = \alpha + \beta X + \gamma D + \varepsilon$$

- So, for women (treating X as conditionally fixed)

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 0 + \varepsilon \\ E(Y) &= \alpha + \beta X \end{aligned}$$

- And, for men

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 1 + \varepsilon \\ E(Y) &= (\alpha + \gamma) + \beta X \end{aligned}$$



- In R notation with data in `Data`:

```
model <- lm(income ~ education
                + gender, data=Data).
```

- Different slopes for women and men ("different slopes for different folks") can be modelled by introducing an interaction regressor, the product of X and D, into the model:

$$Y = \alpha + \beta X + \gamma D + \delta(X \times D) + \varepsilon$$

- Different slopes for women and men ("different slopes for different folks") can be modelled by introducing an interaction regressor, the product of X and D, into the model:

$$Y = \alpha + \beta X + \gamma D + \delta(X \times D) + \varepsilon$$

- Then, for women

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 0 + \delta(X \times 0) + \varepsilon \\ E(Y) &= \alpha + \beta X \end{aligned}$$

- Different slopes for women and men ("different slopes for different folks") can be modelled by introducing an interaction regressor, the product of X and D, into the model:

$$Y = \alpha + \beta X + \gamma D + \delta(X \times D) + \varepsilon$$

- Then, for women

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 0 + \delta(X \times 0) + \varepsilon \\ E(Y) &= \alpha + \beta X \end{aligned}$$

- And, for men

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 1 + \delta(X \times 1) + \varepsilon \\ E(Y) &= (\alpha + \gamma) + (\beta + \delta)X \end{aligned}$$

- Different slopes for women and men ("different slopes for different folks") can be modelled by introducing an interaction regressor, the product of X and D, into the model:

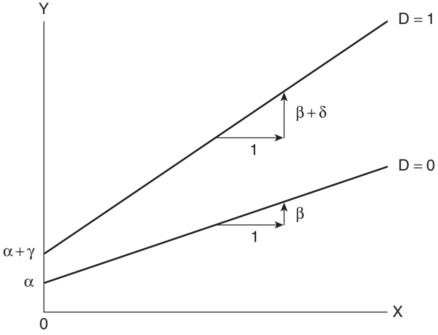
$$Y = \alpha + \beta X + \gamma D + \delta(X \times D) + \varepsilon$$

- Then, for women

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 0 + \delta(X \times 0) + \varepsilon \\ E(Y) &= \alpha + \beta X \end{aligned}$$

- And, for men

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 1 + \delta(X \times 1) + \varepsilon \\ E(Y) &= (\alpha + \gamma) + (\beta + \delta)X \end{aligned}$$



- Different slopes for women and men ("different slopes for different folks") can be modelled by introducing an interaction regressor, the product of X and D, into the model:

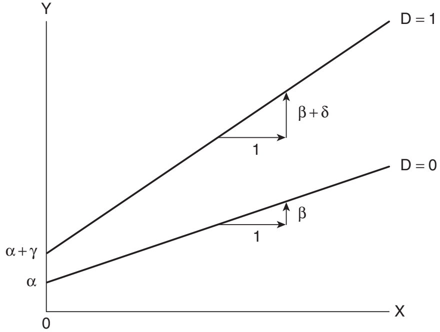
$$Y = \alpha + \beta X + \gamma D + \delta(X \times D) + \varepsilon$$

- Then, for women

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 0 + \delta(X \times 0) + \varepsilon \\ E(Y) &= \alpha + \beta X \end{aligned}$$

- And, for men

$$\begin{aligned} Y &= \alpha + \beta X + \gamma \times 1 + \delta(X \times 1) + \varepsilon \\ E(Y) &= (\alpha + \gamma) + (\beta + \delta) X \end{aligned}$$



- In R (compact) notation:
`model <- lm (income ~ education*gender, data=Data).`

- Polytomous explanatory variables—i.e., factors with more than two levels—are handled by creating a set of dummy regressors, one fewer than the number of levels.

- Polytomous explanatory variables—i.e., factors with more than two levels—are handled by creating a set of dummy regressors, one fewer than the number of levels.
- For example, for gender with levels female, male, and nonbinary, we can code two dummy regressors:

- Polytomous explanatory variables—i.e., factors with more than two levels—are handled by creating a set of dummy regressors, one fewer than the number of levels.
- For example, for gender with levels female, male, and nonbinary, we can code two dummy regressors:

Gender	D_1	D_2
female	0	0
male	1	0
nonbinary	0	1

- Then we can fit the model

$$Y = \alpha + \beta X + \gamma_1 D_1 + \gamma_2 D_3 + \delta_1 (X \times D_1) + \delta_2 (X \times D_2) + \varepsilon$$

- Then we can fit the model

$$Y = \alpha + \beta X + \gamma_1 D_1 + \gamma_2 D_3 + \delta_1 (X \times D_1) + \delta_2 (X \times D_2) + \varepsilon$$

- and

$$\begin{aligned}
\text{female} : E(Y) &= \alpha + \beta X + \gamma_1 \times 0 + \gamma_2 \times 0 + \delta_1 (X \times 0) + \delta_2 (X \times 0) \\
&= \alpha + \beta X \\
\text{male} : E(Y) &= \alpha + \beta X + \gamma_1 \times 1 + \gamma_2 \times 0 + \delta_1 (X \times 1) + \delta_2 (X \times 0) \\
&= (\alpha + \gamma_1) + (\beta + \delta_1) X \\
\text{nonbinary} : E(Y) &= \alpha + \beta X + \gamma_1 \times 0 + \gamma_2 \times 1 + \delta_1 (X \times 0) + \delta_2 (X \times 1) \\
&= (\alpha + \gamma_2) + (\beta + \delta_2) X
\end{aligned}$$

- Type II tests are constructed in conformity to the *principle of marginality*: Each term in the model is tested assuming that its higher-order relatives are zero (and hence are ignored).

- Type II tests are constructed in conformity to the *principle of marginality*: Each term in the model is tested assuming that its higher-order relatives are zero (and hence are ignored).
- Thus, a main effect (e.g., `X`) is tested assuming that the interaction or interactions to which the main effect is marginal (e.g., `X:A`, `X:A:B`) are zero.

- Type II tests are constructed in conformity to the *principle of marginality*: Each term in the model is tested assuming that its higher-order relatives are zero (and hence are ignored).
- Thus, a main effect (e.g., `X`) is tested assuming that the interaction or interactions to which the main effect is marginal (e.g., `X:A`, `X:A:B`) are zero.
- For example, consider the model `y ~ a*b*c` or in longer form
  `y ~ 1 + a + b + c + a:b + a:c + b:c + a:b:c`.

- For Type-II tests of all terms, we implicitly fit the following models (all in longer form):

Model	*Formula*
1	`y ~ 1 + a + b + c + a:b + a:c + b:c + a:b:c`
2	`y ~ 1 + a + b + c + a:b + a:c + b:c`
3	`y ~ 1 + a + b + c + a:c + b:c`
4	`y ~ 1 + a + b + c + a:b + b:c`
5	`y ~ 1 + a + b + c + a:b + a:c`
6	`y ~ 1 + a + b + c + b:c`
7	`y ~ 1 + b + c + b:c`
8	`y ~ 1 + a + b + c + a:c`
9	`y ~ 1 + a + c + a:c`
10	`y ~ 1 + a + b + c + a:b`
11	`y ~ 1 + a + b + a:b`

- Contrasting pairs of models by subtracting the regression sum of squares for the smaller model from that for the larger model produces the Type-II ANOVA table:

Term	*Models Contrasted*
a	$6 - 7$
b	$8 - 9$
c	$10 - 11$
a:b	$2 - 3$
a:c	$2 - 4$
b:c	$2 - 5$
a:b:c	$1 - 2$

- Contrasting pairs of models by subtracting the regression sum of squares for the smaller model from that for the larger model produces the Type-II ANOVA table:

Term	*Models Contrasted*
a	6 – 7
b	8 – 9
c	10 – 11
a:b	2 – 3
a:c	2 – 4
b:c	2 – 5
a:b:c	1 – 2

- The degrees of freedom for each term are the number of regressors used for that term.

- Contrasting pairs of models by subtracting the regression sum of squares for the smaller model from that for the larger model produces the Type-II ANOVA table:

Term	*Models Contrasted*
`a`	$6-7$
`b`	$8-9$
`c`	$10-11$
`a:b`	$2-3$
`a:c`	$2-4$
`b:c`	$2-5$
`a:b:c`	$1-2$

- The degrees of freedom for each term are the number of regressors used for that term.
- The estimated error variance used for the denominator of the F-tests comes from the largest model fit to the data, here Model 1, and the denominator degrees of freedom for F are the residual degrees of freedom for this model.

- `lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)`

- `lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)`
- Operators for the formula argument:

Expression	*Interpretation*	*Example*
A + B	include both A and B	income + education
A - B	exclude B from A	a*b*d - a:b:d
A:B	interaction of A and B	type:education
A*B	A + B + A:B	type*education
B %in% A	B nested within A	education %in% type
A/B	A + B %in% A	type/education
A^k	effects crossed to order k	(a + b + d)^2

- `data`: A data frame containing the data for the model.

- `data`: A data frame containing the data for the model.
- `subset`:

Linear Models in R

Arguments of the `lm()` Function

- `data`: A data frame containing the data for the model.
- `subset`:
  - a logical vector: `subset = gender == "F"`

Arguments of the `lm()` Function

- `data`: A data frame containing the data for the model.
- `subset`:
  - a logical vector: `subset = gender == "F"`
  - a numeric vector of observation indices: `subset = 1:100`

- `data`: A data frame containing the data for the model.
- `subset`:
  - a logical vector: `subset = gender == "F"`
  - a numeric vector of observation indices: `subset = 1:100`
  - a negative numeric vector with observations to be omitted: `subset = -c(6, 16)`

- `data`: A data frame containing the data for the model.
- `subset`:
  - a logical vector: `subset = gender == "F"`
  - a numeric vector of observation indices: `subset = 1:100`
  - a negative numeric vector with observations to be omitted: `subset = -c(6, 16)`
- `weights`: for weighted-least-squares regression

- `data`: A data frame containing the data for the model.
- `subset`:
  - a logical vector: `subset = gender == "F"`
  - a numeric vector of observation indices: `subset = 1:100`
  - a negative numeric vector with observations to be omitted: `subset = -c(6, 16)`
- `weights`: for weighted-least-squares regression
- `na.action`: name of a function to handle missing data; default given by the `na.action` option, initially `"na.omit"`

- `data`: A data frame containing the data for the model.
- `subset`:
  - a logical vector: `subset = gender == "F"`
  - a numeric vector of observation indices: `subset = 1:100`
  - a negative numeric vector with observations to be omitted: `subset = -c(6, 16)`
- `weights`: for weighted-least-squares regression
- `na.action`: name of a function to handle missing data; default given by the `na.action` option, initially `"na.omit"`
- `method`, `model`, `x`, `y`, `qr`, `singular.ok`: technical arguments

- `data`: A data frame containing the data for the model.
- `subset`:
  - a logical vector: `subset = gender == "F"`
  - a numeric vector of observation indices: `subset = 1:100`
  - a negative numeric vector with observations to be omitted: `subset = -c(6, 16)`
- `weights`: for weighted-least-squares regression
- `na.action`: name of a function to handle missing data; default given by the `na.action` option, initially `"na.omit"`
- `method`, `model`, `x`, `y`, `qr`, `singular.ok`: technical arguments
- `contrasts`: specify a list of contrasts for factors; e.g., `contrasts=list(partner.status=contr.sum, fcategory=contr.poly))`

Linear Models in R

Arguments of the `lm()` Function

- `data`: A data frame containing the data for the model.
- `subset`:
  - a logical vector: `subset = gender == "F"`
  - a numeric vector of observation indices: `subset = 1:100`
  - a negative numeric vector with observations to be omitted: `subset = -c(6, 16)`
- `weights`: for weighted-least-squares regression
- `na.action`: name of a function to handle missing data; default given by the `na.action` option, initially `"na.omit"`
- `method`, `model`, `x`, `y`, `qr`, `singular.ok`: technical arguments
- `contrasts`: specify a list of contrasts for factors; e.g., `contrasts=list(partner.status=contr.sum, fcategory=contr.poly))`
- `offset`: term added to the right-hand-side of the model with a fixed coefficient of 1.

- Influence on the regression coefficients = leverage $\times$ outlyingness.

- Influence on the regression coefficients $=$ leverage $\times$ outlyingness.
- *Hat-values* measure leverage.

- Influence on the regression coefficients $=$ leverage $\times$ outlyingness.
- *Hat-values* measure leverage.
    - The fitted linear regression model in matrix form is $\mathrm{y} = \mathrm{Xb} + \mathrm{e}$, where y is the $(n \times 1)$ response vector, X is the $(n \times p)$ model matrix, and $\mathrm{b} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y}$ is the $(p \times 1)$ vector of least squares coefficients.

- Influence on the regression coefficients = leverage × outlyingness.
- *Hat-values* measure leverage.
  - The fitted linear regression model in matrix form is $\mathrm{y} = \mathrm{Xb} + \mathrm{e}$, where y is the $(n \times 1)$ response vector, X is the $(n \times p)$ model matrix, and $\mathrm{b} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y}$ is the $(p \times 1)$ vector of least squares coefficients.
  - The fitted values are then $\widehat{\mathrm{y}} = \mathrm{Xb} = \mathrm{X}(\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y} = \mathrm{Hy}$, where the $(n \times n)$ *hat-matrix* is $\mathrm{H} = \mathrm{X}(\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T$.

- Influence on the regression coefficients $=$ leverage $\times$ outlyingness.
- *Hat-values* measure leverage.
  - The fitted linear regression model in matrix form is $\mathrm{y} = \mathrm{Xb} + \mathrm{e}$, where y is the $(n \times 1)$ response vector, X is the $(n \times p)$ model matrix, and $\mathrm{b} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y}$ is the $(p \times 1)$ vector of least squares coefficients.
  - The fitted values are then $\widehat{\mathrm{y}} = \mathrm{Xb} = \mathrm{X}(\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y} = \mathrm{Hy}$, where the $(n \times n)$ *hat-matrix* is $\mathrm{H} = \mathrm{X}(\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T$.
  - The h_{ij} element of H gives the weight of Y_j in determining $\widehat{Y}_i$.

- Influence on the regression coefficients = leverage × outlyingness.
- *Hat-values* measure leverage.
  - The fitted linear regression model in matrix form is $\mathrm{y} = \mathrm{Xb} + \mathrm{e}$, where y is the $(n \times 1)$ response vector, X is the $(n \times p)$ model matrix, and $\mathrm{b} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y}$ is the $(p \times 1)$ vector of least squares coefficients.
  - The fitted values are then $\widehat{\mathrm{y}} = \mathrm{Xb} = \mathrm{X}(\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y} = \mathrm{Hy}$, where the $(n \times n)$ *hat-matrix* is $\mathrm{H} = \mathrm{X}(\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T$.
  - The h_{ij} element of H gives the weight of Y_j in determining $\widehat{Y}_i$.
  - The H matrix is symmetric ($\mathrm{H} = \mathrm{H}^T$) and idempotent ($\mathrm{H}^2 = \mathrm{H}$), and it follows that the jth diagonal element of H, $h_j = h_{jj} = \sum_{i=1}^{n} h_{ij}^2$ summarizes the size of all of the elements in the jth column of of H and hence the leverage of the jth case in determining the fit.

- Influence on the regression coefficients $=$ leverage $\times$ outlyingness.
- *Hat-values* measure leverage.
    - The fitted linear regression model in matrix form is $\mathrm{y} = \mathrm{X}\mathrm{b} + \mathrm{e}$, where y is the $(n \times 1)$ response vector, X is the $(n \times p)$ model matrix, and $\mathrm{b} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y}$ is the $(p \times 1)$ vector of least squares coefficients.
    - The fitted values are then $\widehat{\mathrm{y}} = \mathrm{X}\mathrm{b} = \mathrm{X}(\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y} = \mathrm{H}\mathrm{y}$, where the $(n \times n)$ *hat-matrix* is $\mathrm{H} = \mathrm{X}(\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T$.
    - The h_{ij} element of H gives the weight of Y_j in determining $\widehat{Y}_i$.
    - The H matrix is symmetric ($\mathrm{H} = \mathrm{H}^T$) and idempotent ($\mathrm{H}^2 = \mathrm{H}$), and it follows that the jth diagonal element of H, $h_j = h_{jj} = \sum_{i=1}^{n} h_{ij}^2$ summarizes the size of all of the elements in the jth column of of H and hence the leverage of the jth case in determining the fit.
    - The diagonal entries h_j of H are the hat-values.

- Influence on the regression coefficients = leverage × outlyingness.
- *Hat-values* measure leverage.
  - The fitted linear regression model in matrix form is $\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e}$, where $\mathbf{y}$ is the $(n \times 1)$ response vector, $\mathbf{X}$ is the $(n \times p)$ model matrix, and $\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ is the $(p \times 1)$ vector of least squares coefficients.
  - The fitted values are then $\widehat{\mathbf{y}} = \mathbf{X}\mathbf{b} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{H}\mathbf{y}$, where the $(n \times n)$ *hat-matrix* is $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$.
  - The h_{ij} element of $\mathbf{H}$ gives the weight of Y_j in determining $\widehat{Y}_i$.
  - The $\mathbf{H}$ matrix is symmetric ($\mathbf{H} = \mathbf{H}^T$) and idempotent ($\mathbf{H}^2 = \mathbf{H}$), and it follows that the jth diagonal element of $\mathbf{H}$, $h_j = h_{jj} = \sum_{i=1}^{n} h_{ij}^2$ summarizes the size of all of the elements in the jth column of of $\mathbf{H}$ and hence the leverage of the jth case in determining the fit.
  - The diagonal entries h_j of $\mathbf{H}$ are the hat-values.
  - The hat-values are bounded between $1/n$ (if the model has an intercept, otherwise 0) and 1, and the average hat-values is $\overline{h} = p/n$.

- *Studentized residuals* measure outlyingness.

- *Studentized residuals* measure outlyingness.
  - The studentized residuals are

$$E_{Ti} = \frac{E_i}{S_{E(-i)}\sqrt{1 - h_i}}$$

  where E_i is the ith element of the least-squares residuals vector e and $S_{E(-i)}$ is the standard deviation of the residuals when the regression is refit with the ith case removed.

- *Studentized residuals* measure outlyingness.
  - The studentized residuals are
$$E_{Ti} = \frac{E_i}{S_{E(-i)}\sqrt{1-h_i}}$$
where E_i is the ith element of the least-squares residuals vector e and $S_{E(-i)}$ is the standard deviation of the residuals when the regression is refit with the ith case removed.
  - If the model is correct, then each studentized residual is distributed at t with $n-p-1$ degrees of freedom, providing a basis for an outlier test based on the the largest absolute studentized residual.

- *Studentized residuals* measure outlyingness.
  - The studentized residuals are

$$E_{Ti} = \frac{E_i}{S_{E(-i)}\sqrt{1 - h_i}}$$

    where E_i is the ith element of the least-squares residuals vector e and $S_{E(-i)}$ is the standard deviation of the residuals when the regression is refit with the ith case removed.
  - If the model is correct, then each studentized residual is distributed at t with $n - p - 1$ degrees of freedom, providing a basis for an outlier test based on the the largest absolute studentized residual.
  - But because there are n studentized residuals, it's necessary to correct for simultaneous statistical inference—e.g., a Bonferroni correction, which multiplies the two-sided P-value for the t-test by n.

- Measuring influence on the regression coefficients with dfbeta and Cook's D:

- Measuring influence on the regression coefficients with dfbeta and Cook's D:
  - The most direct measure is to refit the model without the ith case and see how the coefficients change.

- Measuring influence on the regression coefficients with dfbeta and Cook's D:
  - The most direct measure is to refit the model without the ith case and see how the coefficients change.
  - The answer is $\text{dfbeta}_i = \mathbf{b} - \mathbf{b}_{(-i)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i E_i/(1-h_i)$, where $\mathbf{b}_{(-i)}$ is the vector of least-squares coefficients computed with the ith case deleted, and $\mathbf{x}_i$ is the ith row of $\mathbf{X}$ (written as a column vector).

- Measuring influence on the regression coefficients with dfbeta and Cook's D:
  - The most direct measure is to refit the model without the ith case and see how the coefficients change.
  - The answer is $\text{dfbeta}_i = \mathbf{b} - \mathbf{b}_{(-i)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i E_i/(1-h_i)$, where $\mathbf{b}_{(-i)}$ is the vector of least-squares coefficients computed with the ith case deleted, and $\mathbf{x}_i$ is the ith row of $\mathbf{X}$ (written as a column vector).
  - Because there are a lot $(n \times p)$ of dfbeta_{ij}, it's useful to summarize the p values for each case i. The most common such measure is *Cook's distance*:

$$D_i = \frac{\text{dfbeta}_i^T \mathbf{X}^T\mathbf{X}\ \text{dfbeta}_i}{pS_E^2} = \frac{(\widehat{\mathbf{y}} - \widehat{\mathbf{y}}_{(-i)})^T(\widehat{\mathbf{y}} - \widehat{\mathbf{y}}_{(-i)})}{pS_E^2} \approx \frac{E_{Ti}^2}{p} \times \frac{h_i}{1-h_i}$$
$$= \text{outlyingness} \times \text{leverage}$$

  where $\widehat{\mathbf{y}}_{(-i)}$ is the vector of fitted values computed when the ith case is removed.

- Added-variable plots visualize leverage, outlyingness, and influence on each regression coefficient, reducing the p-dimensional scatterplot of the data to a series of p two-dimensional scatterplots, one for each coefficient.

- Added-variable plots visualize leverage, outlyingness, and influence on each regression coefficient, reducing the p-dimensional scatterplot of the data to a series of p two-dimensional scatterplots, one for each coefficient.
- For example, focusing on the coefficient B_1 of X_1 in the regression $Y = A + B_1X_1 + B_2X_2 + \cdots + B_kX_k + E$ (so $p = k + 1$):

- Added-variable plots visualize leverage, outlyingness, and influence on each regression coefficient, reducing the p-dimensional scatterplot of the data to a series of p two-dimensional scatterplots, one for each coefficient.
- For example, focusing on the coefficient B_1 of X_1 in the regression $Y = A + B_1X_1 + B_2X_2 + \cdots + B_kX_k + E$ (so $p = k + 1$):
  - Regress Y on $X_2, \ldots, X_k$ (and an intercept), obtaining residuals $E^{(Y_1)}$ (i.e., what remains of Y when the effects of $X_2, \ldots, X_k$ are removed).

- Added-variable plots visualize leverage, outlyingness, and influence on each regression coefficient, reducing the p-dimensional scatterplot of the data to a series of p two-dimensional scatterplots, one for each coefficient.
- For example, focusing on the coefficient B_1 of X_1 in the regression $Y = A + B_1 X_1 + B_2 X_2 + \cdots + B_k X_k + E$ (so $p = k + 1$):
  - Regress Y on $X_2, \ldots, X_k$ (and an intercept), obtaining residuals $E^{(Y_1)}$ (i.e., what remains of Y when the effects of $X_2, \ldots, X_k$ are removed).
  - Regress X_1 on $X_2, \ldots, X_k$ (and an intercept), obtaining residuals $E^{(X_1)}$ (i.e., what remains of X_1 when the effects of $X_2, \ldots, X_k$ are removed).

- Added-variable plots visualize leverage, outlyingness, and influence on each regression coefficient, reducing the p-dimensional scatterplot of the data to a series of p two-dimensional scatterplots, one for each coefficient.
- For example, focusing on the coefficient B_1 of X_1 in the regression $Y = A + B_1X_1 + B_2X_2 + \cdots + B_kX_k + E$ (so $p = k + 1$):
  - Regress Y on $X_2, \ldots, X_k$ (and an intercept), obtaining residuals $E^{(Y_1)}$ (i.e., what remains of Y when the effects of $X_2, \ldots, X_k$ are removed).
  - Regress X_1 on $X_2, \ldots, X_k$ (and an intercept), obtaining residuals $E^{(X_1)}$ (i.e., what remains of X_1 when the effects of $X_2, \ldots, X_k$ are removed).
  - plot $E^{(Y_1)}$ versus $E^{(X_1)}$.

- Added-variable plots visualize leverage, outlyingness, and influence on each regression coefficient, reducing the p-dimensional scatterplot of the data to a series of p two-dimensional scatterplots, one for each coefficient.
- For example, focusing on the coefficient B_1 of X_1 in the regression $Y = A + B_1X_1 + B_2X_2 + \cdots + B_kX_k + E$ (so $p = k + 1$):
  - Regress Y on $X_2, \ldots, X_k$ (and an intercept), obtaining residuals $E^{(Y_1)}$ (i.e., what remains of Y when the effects of $X_2, \ldots, X_k$ are removed).
  - Regress X_1 on $X_2, \ldots, X_k$ (and an intercept), obtaining residuals $E^{(X_1)}$ (i.e., what remains of X_1 when the effects of $X_2, \ldots, X_k$ are removed).
  - plot $E^{(Y_1)}$ versus $E^{(X_1)}$.
- Repeat for each of $X_2, \ldots, X_k$ (and even, if desired, for the constant regressor, $X_0 = 1$).

- The AV plot for X_j has the following remarkable properties:

- The AV plot for X_j has the following remarkable properties:
  - The slope of the least-squares line in the plot is the coefficient B_j of X_j in the multiple regression.

- The AV plot for X_j has the following remarkable properties:
  - The slope of the least-squares line in the plot is the coefficient B_j of X_j in the multiple regression.
  - The residuals from this line are the same as the residuals E_i in the multiple regression.

- The AV plot for X_j has the following remarkable properties:
  - The slope of the least-squares line in the plot is the coefficient B_j of X_j in the multiple regression.
  - The residuals from this line are the same as the residuals E_i in the multiple regression.
  - The horizontal variation of X_j in the plot is its conditional variation holding the other Xs constant: $S^2_{X_j|\text{other } X\text{s}} = \sum E^{(X_j)^2} / (n - k)$.

- The AV plot for X_j has the following remarkable properties:
  - The slope of the least-squares line in the plot is the coefficient B_j of X_j in the multiple regression.
  - The residuals from this line are the same as the residuals E_i in the multiple regression.
  - The horizontal variation of X_j in the plot is its conditional variation holding the other Xs constant: $S^2_{X_j|\text{other } X\text{s}} = \sum E^{(X_j)^2} / (n-k)$.
  - Consequently, the standard error of B_j computed from the simple regression corresponding to the plot, $\text{SE}(B_j) = S_E / \sqrt{\sum E^{(X_j)^2}}$ is the same as the standard error of B_j from the multiple regression.

- Component-plus-Residuals plots are even a simpler way of reducing the p-dimensional scatterplot to a series of 2D plots:

- Component-plus-Residuals plots are even a simpler way of reducing the *p*-dimensional scatterplot to a series of 2D plots:
    - Add the residuals from the full regression to the linear component representing X_1 to form the *partial residuals*: $E^{(1)} = B_1X_1 + E$.

- Component-plus-Residuals plots are even a simpler way of reducing the *p*-dimensional scatterplot to a series of 2D plots:
  - Add the residuals from the full regression to the linear component representing X_1 to form the *partial residuals*: $E^{(1)} = B_1 X_1 + E$.
  - Plot $E^{(1)}$ versus X_1, enhancing the graph with a scatterplot smoother (nonparametric regression line) to judge nonlinearity.

- Component-plus-Residuals plots are even a simpler way of reducing the p-dimensional scatterplot to a series of 2D plots:
  - Add the residuals from the full regression to the linear component representing X_1 to form the *partial residuals*: $E^{(1)} = B_1X_1 + E$.
  - Plot $E^{(1)}$ versus X_1, enhancing the graph with a scatterplot smoother (nonparametric regression line) to judge nonlinearity.
- By construction, the least-squares slope of the C+R plot for X_1 is B_1 from the multiple regression, and the residuals in the C+R plot are just the Es.

- Component-plus-Residuals plots are even a simpler way of reducing the *p*-dimensional scatterplot to a series of 2D plots:
    - Add the residuals from the full regression to the linear component representing X_1 to form the *partial residuals*: $E^{(1)} = B_1X_1 + E$.
    - Plot $E^{(1)}$ versus X_1, enhancing the graph with a scatterplot smoother (nonparametric regression line) to judge nonlinearity.
- By construction, the least-squares slope of the C+R plot for X_1 is B_1 from the multiple regression, and the residuals in the C+R plot are just the Es.
- Under certain reasonably general (but not bulletproof) circumstances, if the partial relationship between Y and X_1 is nonlinear but incorrectly modelled as linear, the nature of the nonlinearity will be apparent in the C+R plot for X_1.

- Component-plus-Residuals plots are even a simpler way of reducing the *p*-dimensional scatterplot to a series of 2D plots:
  - Add the residuals from the full regression to the linear component representing X_1 to form the *partial residuals*: $E^{(1)} = B_1X_1 + E$.
  - Plot $E^{(1)}$ versus X_1, enhancing the graph with a scatterplot smoother (nonparametric regression line) to judge nonlinearity.
- By construction, the least-squares slope of the C+R plot for X_1 is B_1 from the multiple regression, and the residuals in the C+R plot are just the Es.
- Under certain reasonably general (but not bulletproof) circumstances, if the partial relationship between Y and X_1 is nonlinear but incorrectly modelled as linear, the nature of the nonlinearity will be apparent in the C+R plot for X_1.
- Repeat for each of $X_2, \ldots, X_k$.

- It's often possible to linearize a nonlinear relationship between Y and X by transforming one or the other (or both) by a power transformation.

- It's often possible to linearize a nonlinear relationship between Y and X by transforming one or the other (or both) by a power transformation.
- By power transformations, I mean $X \rightarrow X^p$ or similarly for Y.

- It's often possible to linearize a nonlinear relationship between Y and X by transforming one or the other (or both) by a power transformation.
- By power transformations, I mean $X \rightarrow X^p$ or similarly for Y.
    - The power p may be positive or negative, and need not be a whole number.

- It's often possible to linearize a nonlinear relationship between Y and X by transforming one or the other (or both) by a power transformation.
- By power transformations, I mean $X \rightarrow X^p$ or similarly for Y.
    - The power p may be positive or negative, and need not be a whole number.
    - For example, $X^{1/2} = \sqrt{X}$ and $X^{-1} = 1/X$.

- It's often possible to linearize a nonlinear relationship between Y and X by transforming one or the other (or both) by a power transformation.
- By power transformations, I mean $X \rightarrow X^p$ or similarly for Y.
  - The power p may be positive or negative, and need not be a whole number.
  - For example, $X^{1/2} = \sqrt{X}$ and $X^{-1} = 1/X$.
  - $p = 1$ is no transformation: $X^1 = X$.

- It's often possible to linearize a nonlinear relationship between Y and X by transforming one or the other (or both) by a power transformation.
- By power transformations, I mean $X \rightarrow X^p$ or similarly for Y.
    - The power p may be positive or negative, and need not be a whole number.
    - For example, $X^{1/2} = \sqrt{X}$ and $X^{-1} = 1/X$.
    - $p = 1$ is no transformation: $X^1 = X$.
    - If $p = 0$, we use $\log(X)$.

- It's often possible to linearize a nonlinear relationship between Y and X by transforming one or the other (or both) by a power transformation.
- By power transformations, I mean $X \rightarrow X^p$ or similarly for Y.
  - The power p may be positive or negative, and need not be a whole number.
  - For example, $X^{1/2} = \sqrt{X}$ and $X^{-1} = 1/X$.
  - $p = 1$ is no transformation: $X^1 = X$.
  - If $p = 0$, we use $\log(X)$.
  - Following John Tukey, we say that $p > 1$ (e.g., X^2, X^3) is a transformation "up the ladder of powers" and $p < 1$ (e.g., $X^{1/2}$, $\log(X)$, $1/X$) is "down the ladder of powers."

- This approach works if

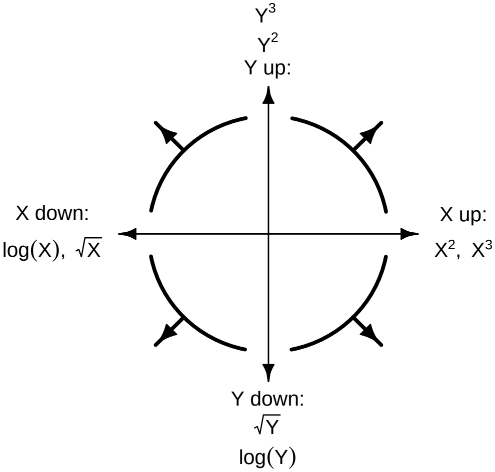
- This approach works if
    1. The values of the variable to be transformed are all *positive*.

- This approach works if
  1. The values of the variable to be transformed are all *positive*.
  2. The relationship between the variables is *monotone* (strictly increasing or decreasing).

- This approach works if
  1. The values of the variable to be transformed are all *positive*.
  2. The relationship between the variables is *monotone* (strictly increasing or decreasing).
  3. The relationship is *simple*, in the sense that the direction of curvature doesn't change.

- This approach works if
  1. The values of the variable to be transformed are all *positive*.
  2. The relationship between the variables is *monotone* (strictly increasing or decreasing).
  3. The relationship is *simple*, in the sense that the direction of curvature doesn't change.
  4. There are then only four patterns, summarized by Mosteller and Tukey's *bulging rule*:

- This approach works if
  1. The values of the variable to be transformed are all *positive*.
  2. The relationship between the variables is *monotone* (strictly increasing or decreasing).
  3. The relationship is *simple*, in the sense that the direction of curvature doesn't change.
  4. There are then only four patterns, summarized by Mosteller and Tukey's *bulging rule*:



Outline

- A generalized linear model consists of three components:

- A generalized linear model consists of three components:

1. A *random component*, specifying the conditional distribution of the response variable, Y_i, given the predictors. Traditionally, the random component is an exponential family — the normal (Gaussian), binomial, Poisson, gamma, or inverse-Gaussian.

- A generalized linear model consists of three components:

1. A *random component*, specifying the conditional distribution of the response variable, Y_i, given the predictors. Traditionally, the random component is an exponential family — the normal (Gaussian), binomial, Poisson, gamma, or inverse-Gaussian.
2. A linear function of the regressors, called the *linear predictor*,

$$\eta_i = \alpha + \beta_1 X_{i1} + \cdots + \beta_k X_{ik}$$

on which the expected value μ_i of Y_i depends.

- A generalized linear model consists of three components:

1. A *random component*, specifying the conditional distribution of the response variable, Y_i, given the predictors. Traditionally, the random component is an exponential family — the normal (Gaussian), binomial, Poisson, gamma, or inverse-Gaussian.
2. A linear function of the regressors, called the *linear predictor*,

$$\eta_i = \alpha + \beta_1 X_{i1} + \cdots + \beta_k X_{ik}$$

   on which the expected value μ_i of Y_i depends.
3. A *link function* $g(\mu_i) = \eta_i$, which transforms the expectation of the response to the linear predictor. The inverse of the link function is called the *mean function*: $g^{-1}(\eta_i) = \mu_i$.

- In the following table, the logit, probit and complementary log-log links are for binomial or binary data:

Link	$\eta_i = g(\mu_i)$	$\mu_i = g^{-1}(\eta_i)$
identity	μ_i	η_i
log	$\log_e \mu_i$	e^{η_i}
inverse	μ_i^{-1}	η_i^{-1}
inverse-square	μ_i^{-2}	$\eta_i^{-1/2}$
square-root	$\sqrt{\mu_i}$	η_i^2
logit	$\log_e \dfrac{\mu_i}{1-\mu_i}$	$\dfrac{1}{1+e^{-\eta_i}}$
probit	$\Phi(\mu_i)$	$\Phi^{-1}(\eta_i)$
complementary log-log	$\log_e[-\log_e(1-\mu_i)]$	$1-\exp[-\exp(\eta_i)]$

- Generalized linear models are fit with the `glm()` function. Most of the arguments of `glm()` are similar to those of `lm()`:

- Generalized linear models are fit with the `glm()` function. Most of the arguments of `glm()` are similar to those of `lm()`:
  - The response variable and regressors are given in a model `formula`.

- Generalized linear models are fit with the `glm()` function. Most of the arguments of `glm()` are similar to those of `lm()`:
  - The response variable and regressors are given in a model `formula`.
  - `data`, `subset`, and `na.action` arguments determine the data on which the model is fit.

- Generalized linear models are fit with the `glm()` function. Most of the arguments of `glm()` are similar to those of `lm()`:
  - The response variable and regressors are given in a model `formula`.
  - `data`, `subset`, and `na.action` arguments determine the data on which the model is fit.
  - The additional `family` argument is used to specify a *family-generator function*, which may take other arguments, such as a link function.

- The following table gives family generators and default links:

Family	*Default Link*	*Range of* Y_i	$V(Y_i \mid \eta_i)$
`gaussian`	`identity`	$(-\infty, +\infty)$	ϕ
`binomial`	`logit`	$\frac{0, 1, \ldots, n_i}{n_i}$	$\mu_i(1-\mu_i)$
`poisson`	`log`	$0, 1, 2, \ldots$	μ_i
`Gamma`	`inverse`	$(0, \infty)$	$\phi\mu_i^2$
`inverse.gaussian`	`1/mu^2`	$(0, \infty)$	$\phi\mu_i^3$

- The following table gives family generators and default links:

Family	*Default Link*	*Range of* Y_i	$V(Y_i \mid \eta_i)$
`gaussian`	`identity`	$(-\infty, +\infty)$	ϕ
`binomial`	`logit`	$\frac{0, 1, \ldots, n_i}{n_i}$	$\mu_i(1-\mu_i)$
`poisson`	`log`	$0, 1, 2, \ldots$	μ_i
`Gamma`	`inverse`	$(0, \infty)$	$\phi\mu_i^2$
`inverse.gaussian`	`1/mu^2`	$(0, \infty)$	$\phi\mu_i^3$

- For distributions in the exponential families, the variance is a function of the mean and a dispersion parameter ϕ (fixed to 1 for the binomial and Poisson distributions).

- The following table shows the links available (✓) for each family in R, with the default link marked by ★:

	link							
family	identity	inverse	sqrt	1/mu^2	log	logit	probit	cloglog
gaussian	★	✓			✓			
binomial					✓	★	✓	✓
poisson	✓		✓		★			
Gamma	✓	★			✓			
inverse.gaussian	✓	✓		★	✓			
quasi	★	✓	✓	✓	✓	✓	✓	✓
quasibinomial						★	✓	✓
quasipoisson	✓		✓		★			

- The following table shows the links available (✓) for each family in R, with the default link marked by ★:

family	link							
	identity	inverse	sqrt	1/mu^2	log	logit	probit	cloglog
gaussian	★	✓			✓			
binomial					✓	★	✓	✓
poisson	✓		✓		★			
Gamma	✓	★			✓			
inverse.gaussian	✓	✓		★	✓			
quasi	★	✓	✓	✓	✓	✓	✓	✓
quasibinomial						★	✓	✓
quasipoisson	✓		✓		★			

- The quasi, quasibinomial, and quasipoisson family generators do not correspond to exponential families.

- The response for a binomial GLM may be specified in several forms:

- The response for a binomial GLM may be specified in several forms:
  - For binary data, the response may be

- The response for a binomial GLM may be specified in several forms:
  - For binary data, the response may be
    - a variable or an R expression that evaluates to 0s (‘failure’) and 1s (‘success’).

- The response for a binomial GLM may be specified in several forms:
  - For binary data, the response may be
    - a variable or an R expression that evaluates to `0`s (‘failure’) and `1`s (‘success’).
    - a logical variable or expression, such as `voted == "yes"` (with `TRUE` representing success, and `FALSE` failure).

- The response for a binomial GLM may be specified in several forms:
  - For binary data, the response may be
    - a variable or an R expression that evaluates to `0`s (‘failure’) and `1`s (‘success’).
    - a logical variable or expression, such as `voted == "yes"` (with `TRUE` representing success, and `FALSE` failure).
    - a factor (in which case the first category is taken to represent failure and the others success).

- The response for a binomial GLM may be specified in several forms:
  - For binary data, the response may be
    - a variable or an R expression that evaluates to `0`s ('failure') and `1`s ('success').
    - a logical variable or expression, such as `voted == "yes"` (with `TRUE` representing success, and `FALSE` failure).
    - a factor (in which case the first category is taken to represent failure and the others success).
  - For binomial data, the response may be

- The response for a binomial GLM may be specified in several forms:
  - For binary data, the response may be
    - a variable or an R expression that evaluates to `0`s (‘failure’) and `1`s (‘success’).
    - a logical variable or expression, such as `voted == "yes"` (with `TRUE` representing success, and `FALSE` failure).
    - a factor (in which case the first category is taken to represent failure and the others success).
  - For binomial data, the response may be
    - a two-column matrix, with the first column giving the count of successes and the second the count of failures for each binomial observation.

- The response for a binomial GLM may be specified in several forms:
  - For binary data, the response may be
    - a variable or an R expression that evaluates to `0`s ('failure') and `1`s ('success').
    - a logical variable or expression, such as `voted == "yes"` (with `TRUE` representing success, and `FALSE` failure).
    - a factor (in which case the first category is taken to represent failure and the others success).
  - For binomial data, the response may be
    - a two-column matrix, with the first column giving the count of successes and the second the count of failures for each binomial observation.
    - a vector giving the *proportion* of successes, while the binomial denominators (total counts or numbers of trials) are given by the `weights` argument to `glm()`.

- Poisson generalized linear models are commonly used when the response variable is a count (Poisson regression) and for modeling associations in contingency tables (loglinear models). The two applications are formally equivalent.

- Poisson generalized linear models are commonly used when the response variable is a count (Poisson regression) and for modeling associations in contingency tables (loglinear models). The two applications are formally equivalent.
- Poisson GLMs are fit in R using the `poisson` family generator with `glm()`.

- Poisson generalized linear models are commonly used when the response variable is a count (Poisson regression) and for modeling associations in contingency tables (loglinear models). The two applications are formally equivalent.
- Poisson GLMs are fit in R using the `poisson` family generator with `glm()`.
- Overdispersed binomial and Poisson models may be fit via the `quasibinomial` and `quasipoisson` families.

- Poisson generalized linear models are commonly used when the response variable is a count (Poisson regression) and for modeling associations in contingency tables (loglinear models). The two applications are formally equivalent.
- Poisson GLMs are fit in R using the `poisson` family generator with `glm()`.
- Overdispersed binomial and Poisson models may be fit via the `quasibinomial` and `quasipoisson` families.
- The `glm.nb()` function in the **MASS** package fits negative-binomial GLMs to count data.

- Poisson generalized linear models are commonly used when the response variable is a count (Poisson regression) and for modeling associations in contingency tables (loglinear models). The two applications are formally equivalent.
- Poisson GLMs are fit in R using the `poisson` family generator with `glm()`.
- Overdispersed binomial and Poisson models may be fit via the `quasibinomial` and `quasipoisson` families.
- The `glm.nb()` function in the **MASS** package fits negative-binomial GLMs to count data.
- The `multinom()` function in the **nnet** package fits multinomial GLMs for nominal polytomous responses.

- Poisson generalized linear models are commonly used when the response variable is a count (Poisson regression) and for modeling associations in contingency tables (loglinear models). The two applications are formally equivalent.
- Poisson GLMs are fit in R using the `poisson` family generator with `glm()`.
- Overdispersed binomial and Poisson models may be fit via the `quasibinomial` and `quasipoisson` families.
- The `glm.nb()` function in the **MASS** package fits negative-binomial GLMs to count data.
- The `multinom()` function in the **nnet** package fits multinomial GLMs for nominal polytomous responses.
- The `polr()` function in the **MASS** package fits the proportional-odds logit model and the ordered probit model to ordinal polytomous responses.

- Poisson generalized linear models are commonly used when the response variable is a count (Poisson regression) and for modeling associations in contingency tables (loglinear models). The two applications are formally equivalent.
- Poisson GLMs are fit in R using the `poisson` family generator with `glm()`.
- Overdispersed binomial and Poisson models may be fit via the `quasibinomial` and `quasipoisson` families.
- The `glm.nb()` function in the **MASS** package fits negative-binomial GLMs to count data.
- The `multinom()` function in the **nnet** package fits multinomial GLMs for nominal polytomous responses.
- The `polr()` function in the **MASS** package fits the proportional-odds logit model and the ordered probit model to ordinal polytomous responses.
- The `clm()` function in the **ordinal** package fits a variety of models (including the proportional-odds model) to ordinal polytomous responses.

Outline

The Linear Mixed-Effects Model

- The *Laird-Ware form* of the linear mixed model:

$$
\begin{aligned}
Y_{ij} &= \beta_1 + \beta_2 X_{2ij} + \cdots + \beta_p X_{pij} + B_{1i} Z_{1ij} + \cdots + B_{qi} Z_{qij} + \varepsilon_{ij} \\
B_{ki} &\sim N(0, \psi_k^2), \mathrm{Cov}(B_{ki}, B_{k'i}) = \psi_{kk'} \\
& \quad B_{ki}, B_{k'i'} \text{ are independent for } i \neq i' \\
\varepsilon_{ij} &\sim N(0, \sigma^2 \lambda_{ijj}), \mathrm{Cov}(\varepsilon_{ij}, \varepsilon_{ij'}) = \sigma^2 \lambda_{ijj'} \\
& \quad \varepsilon_{ij}, \varepsilon_{i'j'} \text{ are independent for } i \neq i'
\end{aligned}
$$

- where:

The Linear Mixed-Effects Model

- where:
  - Y_{ij} is the value of the response variable for the jth of n_i observations in the ith of m groups or clusters.

The Linear Mixed-Effects Model

- where:
  - Y_{ij} is the value of the response variable for the jth of n_i observations in the ith of m groups or clusters.
  - $\beta_1, \beta_2, \ldots, \beta_p$ are the fixed-effect coefficients, which are identical for all groups.

The Linear Mixed-Effects Model

- where:
  - Y_{ij} is the value of the response variable for the jth of n_i observations in the ith of m groups or clusters.
  - $\beta_1, \beta_2, \ldots, \beta_p$ are the fixed-effect coefficients, which are identical for all groups.
  - $X_{2ij}, \ldots, X_{pij}$ are the fixed-effect regressors for observation j in group i; there is also implicitly a constant regressor, $X_{1ij} = 1$.

The Linear Mixed-Effects Model

- where:
  - Y_{ij} is the value of the response variable for the jth of n_i observations in the ith of m groups or clusters.
  - $\beta_1, \beta_2, \ldots, \beta_p$ are the fixed-effect coefficients, which are identical for all groups.
  - $X_{2ij}, \ldots, X_{pij}$ are the fixed-effect regressors for observation j in group i; there is also implicitly a constant regressor, $X_{1ij} = 1$.
  - $B_{1i}, \ldots, B_{qi}$ are the random-effect coefficients for group i, assumed to be multivariately normally distributed, independent of the random effects of other groups. The random effects, therefore, vary by group.

The Linear Mixed-Effects Model

- where:
    - Y_{ij} is the value of the response variable for the jth of n_i observations in the ith of m groups or clusters.
    - $\beta_1, \beta_2, \ldots, \beta_p$ are the fixed-effect coefficients, which are identical for all groups.
    - $X_{2ij}, \ldots, X_{pij}$ are the fixed-effect regressors for observation j in group i; there is also implicitly a constant regressor, $X_{1ij} = 1$.
    - $B_{1i}, \ldots, B_{qi}$ are the random-effect coefficients for group i, assumed to be multivariately normally distributed, independent of the random effects of other groups. The random effects, therefore, vary by group.
        - The B_{ik} are thought of as random variables, not as parameters, and are similar in this respect to the errors ε_{ij}.

The Linear Mixed-Effects Model

- where:
  - Y_{ij} is the value of the response variable for the jth of n_i observations in the ith of m groups or clusters.
  - $\beta_1, \beta_2, \ldots, \beta_p$ are the fixed-effect coefficients, which are identical for all groups.
  - $X_{2ij}, \ldots, X_{pij}$ are the fixed-effect regressors for observation j in group i; there is also implicitly a constant regressor, $X_{1ij} = 1$.
  - $B_{1i}, \ldots, B_{qi}$ are the random-effect coefficients for group i, assumed to be multivariately normally distributed, independent of the random effects of other groups. The random effects, therefore, vary by group.
    - The B_{ik} are thought of as random variables, not as parameters, and are similar in this respect to the errors ε_{ij}.
  - $Z_{1ij}, \ldots, Z_{qij}$ are the random-effect regressors.

The Linear Mixed-Effects Model

- where:
  - Y_{ij} is the value of the response variable for the jth of n_i observations in the ith of m groups or clusters.
  - $\beta_1, \beta_2, \ldots, \beta_p$ are the fixed-effect coefficients, which are identical for all groups.
  - $X_{2ij}, \ldots, X_{pij}$ are the fixed-effect regressors for observation j in group i; there is also implicitly a constant regressor, $X_{1ij} = 1$.
  - $B_{1i}, \ldots, B_{qi}$ are the random-effect coefficients for group i, assumed to be multivariately normally distributed, independent of the random effects of other groups. The random effects, therefore, vary by group.
    - The B_{ik} are thought of as random variables, not as parameters, and are similar in this respect to the errors ε_{ij}.
  - $Z_{1ij}, \ldots, Z_{qij}$ are the random-effect regressors.
    - The Zs are almost always a subset of the Xs (and may include *all* of the Xs).

The Linear Mixed-Effects Model

- where:
  - Y_{ij} is the value of the response variable for the jth of n_i observations in the ith of m groups or clusters.
  - $\beta_1, \beta_2, \ldots, \beta_p$ are the fixed-effect coefficients, which are identical for all groups.
  - $X_{2ij}, \ldots, X_{pij}$ are the fixed-effect regressors for observation j in group i; there is also implicitly a constant regressor, $X_{1ij} = 1$.
  - $B_{1i}, \ldots, B_{qi}$ are the random-effect coefficients for group i, assumed to be multivariately normally distributed, independent of the random effects of other groups. The random effects, therefore, vary by group.
    - The B_{ik} are thought of as random variables, not as parameters, and are similar in this respect to the errors ε_{ij}.
  - $Z_{1ij}, \ldots, Z_{qij}$ are the random-effect regressors.
    - The Zs are almost always a subset of the Xs (and may include *all* of the Xs).
    - When there is a random intercept term, $Z_{1ij} = 1$.

The Linear Mixed-Effects Model

- The remaining parameters specify the variance-covariance components (don't get lost!):

The Linear Mixed-Effects Model

- The remaining parameters specify the variance-covariance components (don't get lost!):
  - ψ_k^2 are the variances and $\psi_{kk'}$ the covariances among the random effects, assumed to be constant across groups.

The Linear Mixed-Effects Model

- The remaining parameters specify the variance-covariance components (don't get lost!):
  - ψ_k^2 are the variances and $\psi_{kk'}$ the covariances among the random effects, assumed to be constant across groups.
    - In some applications, the ψs are parametrized in terms of a smaller number of fundamental parameters.

The Linear Mixed-Effects Model

- The remaining parameters specify the variance-covariance components (don't get lost!):
  - ψ_k^2 are the variances and $\psi_{kk'}$ the covariances among the random effects, assumed to be constant across groups.
    - In some applications, the ψs are parametrized in terms of a smaller number of fundamental parameters.
  - ε_{ij} is the error for observation j in group i.

The Linear Mixed-Effects Model

- The remaining parameters specify the variance-covariance components (don't get lost!):
  - ψ_k^2 are the variances and $\psi_{kk'}$ the covariances among the random effects, assumed to be constant across groups.
    - In some applications, the ψs are parametrized in terms of a smaller number of fundamental parameters.
  - ε_{ij} is the error for observation j in group i.
    - The errors for group i are assumed to be multivariately normally distributed, and independent of errors in other groups.

The Linear Mixed-Effects Model

- The remaining parameters specify the variance-covariance components (don't get lost!):
  - ψ_k^2 are the variances and $\psi_{kk'}$ the covariances among the random effects, assumed to be constant across groups.
    - In some applications, the ψs are parametrized in terms of a smaller number of fundamental parameters.
  - ε_{ij} is the error for observation j in group i.
    - The errors for group i are assumed to be multivariately normally distributed, and independent of errors in other groups.
  - $\sigma^2\lambda_{ijj'}$ are the covariances between errors in group i.

The Linear Mixed-Effects Model

- The remaining parameters specify the variance-covariance components (don't get lost!):
  - ψ_k^2 are the variances and $\psi_{kk'}$ the covariances among the random effects, assumed to be constant across groups.
    - In some applications, the ψs are parametrized in terms of a smaller number of fundamental parameters.
  - ε_{ij} is the error for observation j in group i.
    - The errors for group i are assumed to be multivariately normally distributed, and independent of errors in other groups.
  - $\sigma^2\lambda_{ijj'}$ are the covariances between errors in group i.
    - Generally, the $\lambda_{ijj'}$ are parametrized in terms of a few basic parameters, and their specific form depends upon context.

The Linear Mixed-Effects Model

- The remaining parameters specify the variance-covariance components (don't get lost!):
  - ψ_k^2 are the variances and $\psi_{kk'}$ the covariances among the random effects, assumed to be constant across groups.
    - In some applications, the ψs are parametrized in terms of a smaller number of fundamental parameters.
  - ε_{ij} is the error for observation j in group i.
    - The errors for group i are assumed to be multivariately normally distributed, and independent of errors in other groups.
  - $\sigma^2\lambda_{ijj'}$ are the covariances between errors in group i.
    - Generally, the $\lambda_{ijj'}$ are parametrized in terms of a few basic parameters, and their specific form depends upon context.
    - When observations are sampled independently within groups and are assumed to have constant error variance (as is typical in hierarchical models), $\lambda_{ijj} = 1$, $\lambda_{ijj'} = 0$ (for $j \neq j'$), and thus the only free parameter to estimate is the common error variance, σ^2.

The Linear Mixed-Effects Model

- The remaining parameters specify the variance-covariance components (don't get lost!):
  - ψ_k^2 are the variances and $\psi_{kk'}$ the covariances among the random effects, assumed to be constant across groups.
    - In some applications, the ψs are parametrized in terms of a smaller number of fundamental parameters.
  - ε_{ij} is the error for observation j in group i.
    - The errors for group i are assumed to be multivariately normally distributed, and independent of errors in other groups.
  - $\sigma^2 \lambda_{ijj'}$ are the covariances between errors in group i.
    - Generally, the $\lambda_{ijj'}$ are parametrized in terms of a few basic parameters, and their specific form depends upon context.
    - When observations are sampled independently within groups and are assumed to have constant error variance (as is typical in hierarchical models), $\lambda_{ijj} = 1$, $\lambda_{ijj'} = 0$ (for $j \neq j'$), and thus the only free parameter to estimate is the common error variance, σ^2.
    - If the observations in a "group" represent longitudinal data on a single individual, then the structure of the λs may be specified to capture serial (i.e., over-time) dependencies among the errors.

- In the **nlme** package (Pinheiro, Bates, DebRoy, and Sarkar):

Fitting Mixed Models in R

with the **nlme** and **lme4** packages

- In the **nlme** package (Pinheiro, Bates, DebRoy, and Sarkar):
  - `lme()`: linear mixed-effects models with nested random effects; can model serially correlated errors.

Fitting Mixed Models in R

with the **nlme** and **lme4** packages

- In the **nlme** package (Pinheiro, Bates, DebRoy, and Sarkar):
  - `lme()`: linear mixed-effects models with nested random effects; can model serially correlated errors.
  - `nlme()`: nonlinear mixed-effects models.

- In the **nlme** package (Pinheiro, Bates, DebRoy, and Sarkar):
  - `lme()`: linear mixed-effects models with nested random effects; can model serially correlated errors.
  - `nlme()`: nonlinear mixed-effects models.
- In the **lme4** package (Bates, Maechler, Bolker, and Walker):

Fitting Mixed Models in R

with the **nlme** and **lme4** packages

- In the **nlme** package (Pinheiro, Bates, DebRoy, and Sarkar):
  - `lme()`: linear mixed-effects models with nested random effects; can model serially correlated errors.
  - `nlme()`: nonlinear mixed-effects models.
- In the **lme4** package (Bates, Maechler, Bolker, and Walker):
  - `lmer()`: linear mixed-effects models with nested or crossed random effects; no facility (yet) for serially correlated errors.

Fitting Mixed Models in R

with the **nlme** and **lme4** packages

- In the **nlme** package (Pinheiro, Bates, DebRoy, and Sarkar):
  - `lme()`: linear mixed-effects models with nested random effects; can model serially correlated errors.
  - `nlme()`: nonlinear mixed-effects models.
- In the **lme4** package (Bates, Maechler, Bolker, and Walker):
  - `lmer()`: linear mixed-effects models with nested or crossed random effects; no facility (yet) for serially correlated errors.
  - `glmer()`: generalized-linear mixed-effects models.

Fitting Mixed Models in R

with the **nlme** and **lme4** packages

- In the **nlme** package (Pinheiro, Bates, DebRoy, and Sarkar):
  - `lme()`: linear mixed-effects models with nested random effects; can model serially correlated errors.
  - `nlme()`: nonlinear mixed-effects models.
- In the **lme4** package (Bates, Maechler, Bolker, and Walker):
  - `lmer()`: linear mixed-effects models with nested or crossed random effects; no facility (yet) for serially correlated errors.
  - `glmer()`: generalized-linear mixed-effects models.
- There are many other CRAN packages that fit a variety of mixed-effects models, perhaps most notably **glmmTMB** (see `https://bbolker.github.io/mixedmodels-misc/glmmFAQ.html`).

Fitting Mixed Models in R

with the **nlme** and **lme4** packages

- In the **nlme** package (Pinheiro, Bates, DebRoy, and Sarkar):
  - `lme()`: linear mixed-effects models with nested random effects; can model serially correlated errors.
  - `nlme()`: nonlinear mixed-effects models.
- In the **lme4** package (Bates, Maechler, Bolker, and Walker):
  - `lmer()`: linear mixed-effects models with nested or crossed random effects; no facility (yet) for serially correlated errors.
  - `glmer()`: generalized-linear mixed-effects models.
- There are many other CRAN packages that fit a variety of mixed-effects models, perhaps most notably **glmmTMB** (see `https://bbolker.github.io/mixedmodels-misc/glmmFAQ.html`).
- There are also Bayesian approaches to modeling hierarchical and longitudinal data that offer certain advantages; see in particular the **rstan**, **rstanarm**, and **blme** packages.

- A level-1 model specifying a linear “growth curve” for log exercise for each subject:

$$\text{log-exercise}_{ij} = \alpha_{0i} + \alpha_{1i}(\text{age}_{ij} - 8) + \varepsilon_{ij}$$

- A level-1 model specifying a linear "growth curve" for log exercise for each subject:

$$\log\text{-exercise}_{ij} = \alpha_{0i} + \alpha_{1i}(\text{age}_{ij} - 8) + \varepsilon_{ij}$$

- Our interest in detecting differences in exercise histories between subjects and controls suggests the level-2 model

$$\alpha_{0i} = \gamma_{00} + \gamma_{01}\text{group}_i + \omega_{0i}$$
$$\alpha_{1i} = \gamma_{10} + \gamma_{11}\text{group}_i + \omega_{1i}$$

where group is a dummy variable coded 1 for subjects and 0 for controls.

- Substituting the level-2 model into the level-1 model produces

$$
\begin{aligned}
\text{log-exercise}_{ij} &= (\gamma_{00} + \gamma_{01}\text{group}_i + \omega_{0i}) + (\gamma_{10} + \gamma_{11}\text{group}_i + \omega_{1i})(\text{age}_{ij} - 8) + \varepsilon_{ij} \\
&= \gamma_{00} + \gamma_{01}\text{group}_i + \gamma_{10}(\text{age}_{ij} - 8) + \gamma_{11}\text{group}_i \times (\text{age}_{ij} - 8) \\
&\qquad + \omega_{0i} + \omega_{1i}(\text{age}_{ij} - 8) + \varepsilon_{ij}
\end{aligned}
$$

A Mixed Model for the Blackmore Exercise Data

Laird-Ware form of the Model

- Substituting the level-2 model into the level-1 model produces

$$\begin{aligned}\text{log-exercise}_{ij} &= (\gamma_{00} + \gamma_{01}\text{group}_i + \omega_{0i}) + (\gamma_{10} + \gamma_{11}\text{group}_i + \omega_{1i})(\text{age}_{ij} - 8) + \varepsilon_{ij} \\ &= \gamma_{00} + \gamma_{01}\text{group}_i + \gamma_{10}(\text{age}_{ij} - 8) + \gamma_{11}\text{group}_i \times (\text{age}_{ij} - 8) \\ &\quad + \omega_{0i} + \omega_{1i}(\text{age}_{ij} - 8) + \varepsilon_{ij}\end{aligned}$$

- in Laird-Ware form,

$$Y_{ij} = \beta_1 + \beta_2 X_{2ij} + \beta_3 X_{3ij} + \beta_4 X_{4ij} + \delta_{1i} + \delta_{2i} Z_{2ij} + \varepsilon_{ij}$$

- Substituting the level-2 model into the level-1 model produces

$$\begin{aligned}
\text{log-exercise}_{ij} &= (\gamma_{00} + \gamma_{01}\text{group}_i + \omega_{0i}) + (\gamma_{10} + \gamma_{11}\text{group}_i + \omega_{1i})(\text{age}_{ij} - 8) + \varepsilon_{ij} \\
&= \gamma_{00} + \gamma_{01}\text{group}_i + \gamma_{10}(\text{age}_{ij} - 8) + \gamma_{11}\text{group}_i \times (\text{age}_{ij} - 8) \\
&\quad + \omega_{0i} + \omega_{1i}(\text{age}_{ij} - 8) + \varepsilon_{ij}
\end{aligned}$$

- in Laird-Ware form,

$$Y_{ij} = \beta_1 + \beta_2 X_{2ij} + \beta_3 X_{3ij} + \beta_4 X_{4ij} + \delta_{1i} + \delta_{2i} Z_{2ij} + \varepsilon_{ij}$$

- Continuous first-order autoregressive process for the errors:

$$\text{Cor}(\varepsilon_{it}, \varepsilon_{i,t+s}) = \rho(s) = \phi^{|s|}$$

where the time-interval between observations, s, need not be an integer.

- Using lme() in the **nlme** package:

```
lme(log.exercise ~ I(age - 8)*group,
            random = ~ I(age - 8) | subject,
            correlation = corCAR1(form = ~ age |subject)
            data=Blackmoore)
```

A Mixed Model for the Blackmore Exercise Data

Specifying the Model in lme() and lmer()

- Using `lme()` in the **nlme** package:

```
lme(log.exercise ~ I(age - 8)*group,
            random = ~ I(age - 8) | subject,
            correlation = corCAR1(form = ~ age |subject)
            data=Blackmoore)
```

- Using `lmer()` in the **lme4** package, but without autocorrelated errors:

```
lmer(log.exercise ~ I(age - 8)*group + (I(age - 8) | subject),
            data=Blackmoore)
```

Outline

- The "Tidyverse" is an integrated set of R packages developed by Hadley Wickham and his collaborators at RStudio (see `https://www.tidyverse.org/`).

- The "Tidyverse" is an integrated set of R packages developed by Hadley Wickham and his collaborators at RStudio (see `https://www.tidyverse.org/`).
- The packages are meant to provide a straightforward way to import data into R and to manipulate the data.

- The “Tidyverse” is an integrated set of R packages developed by Hadley Wickham and his collaborators at RStudio (see `https://www.tidyverse.org/`).
- The packages are meant to provide a straightforward way to import data into R and to manipulate the data.
- There are also Tidyverse tools for R programming and statistical graphics.

- The "Tidyverse" is an integrated set of R packages developed by Hadley Wickham and his collaborators at RStudio (see `https://www.tidyverse.org/`).
- The packages are meant to provide a straightforward way to import data into R and to manipulate the data.
- There are also Tidyverse tools for R programming and statistical graphics.
- A central goal of the data-oriented Tidyverse packages is to construct, modify, and maintain "tidy data"—rectangular data sets in which the rows represent cases and the columns represent variables.

- The "Tidyverse" is an integrated set of R packages developed by Hadley Wickham and his collaborators at RStudio (see `https://www.tidyverse.org/`).
- The packages are meant to provide a straightforward way to import data into R and to manipulate the data.
- There are also Tidyverse tools for R programming and statistical graphics.
- A central goal of the data-oriented Tidyverse packages is to construct, modify, and maintain "tidy data"—rectangular data sets in which the rows represent cases and the columns represent variables.
  - Of course, the idea of a rectangular data set greatly antedates the Tidyverse and is incorporated in the standard R data frame.

- There are eight "core" Tidyverse packages, which can be installed and loaded via the master **tidyverse** package:

- There are eight "core" Tidyverse packages, which can be installed and loaded via the master **tidyverse** package:
  1. **readr**: Imports rectangular data sets from plain-text files.

- There are eight "core" Tidyverse packages, which can be installed and loaded via the master **tidyverse** package:
  1. **readr**: Imports rectangular data sets from plain-text files.
  2. **tibble**: The specific implementation of rectangular data sets in the Tidyverse is called a "tibble," and tibble objects inherit from the `"data.frame"` class.

- There are eight "core" Tidyverse packages, which can be installed and loaded via the master **tidyverse** package:
  1. **readr**: Imports rectangular data sets from plain-text files.
  2. **tibble**: The specific implementation of rectangular data sets in the Tidyverse is called a "tibble," and tibble objects inherit from the `"data.frame"` class.
  3. **tidyr**: Provides functions to create and maintain rectangular data sets (e.g., to transform rectangular data sets between "wide" and "long" form).

- There are eight "core" Tidyverse packages, which can be installed and loaded via the master **tidyverse** package:
  1. **readr**: Imports rectangular data sets from plain-text files.
  2. **tibble**: The specific implementation of rectangular data sets in the Tidyverse is called a "tibble," and tibble objects inherit from the `"data.frame"` class.
  3. **tidyr**: Provides functions to create and maintain rectangular data sets (e.g., to transform rectangular data sets between "wide" and "long" form).
  4. **dplyr**: Provides functions for data manipulation (e.g., adding variables to an existing data set).

- There are eight "core" Tidyverse packages, which can be installed and loaded via the master **tidyverse** package:
  1. **readr**: Imports rectangular data sets from plain-text files.
  2. **tibble**: The specific implementation of rectangular data sets in the Tidyverse is called a "tibble," and tibble objects inherit from the `"data.frame"` class.
  3. **tidyr**: Provides functions to create and maintain rectangular data sets (e.g., to transform rectangular data sets between "wide" and "long" form).
  4. **dplyr**: Provides functions for data manipulation (e.g., adding variables to an existing data set).
  5. **stringr**: Provides functions for manipulating text (character-string) data (e.g., searching for text).

- There are eight "core" Tidyverse packages, which can be installed and loaded via the master **tidyverse** package:
  1. **readr**: Imports rectangular data sets from plain-text files.
  2. **tibble**: The specific implementation of rectangular data sets in the Tidyverse is called a "tibble," and tibble objects inherit from the `"data.frame"` class.
  3. **tidyr**: Provides functions to create and maintain rectangular data sets (e.g., to transform rectangular data sets between "wide" and "long" form).
  4. **dplyr**: Provides functions for data manipulation (e.g., adding variables to an existing data set).
  5. **stringr**: Provides functions for manipulating text (character-string) data (e.g., searching for text).
  6. **forcats**: Provides functions for manipulating R factors (e.g., changing the order of levels of a factor).

- There are eight "core" Tidyverse packages, which can be installed and loaded via the master **tidyverse** package:
  1. **readr**: Imports rectangular data sets from plain-text files.
  2. **tibble**: The specific implementation of rectangular data sets in the Tidyverse is called a "tibble," and tibble objects inherit from the `"data.frame"` class.
  3. **tidyr**: Provides functions to create and maintain rectangular data sets (e.g., to transform rectangular data sets between "wide" and "long" form).
  4. **dplyr**: Provides functions for data manipulation (e.g., adding variables to an existing data set).
  5. **stringr**: Provides functions for manipulating text (character-string) data (e.g., searching for text).
  6. **forcats**: Provides functions for manipulating R factors (e.g., changing the order of levels of a factor).
  7. **purrr**: Provides R programming tools (e.g., alternatives to iteration).

- There are eight "core" Tidyverse packages, which can be installed and loaded via the master **tidyverse** package:
  1. **readr**: Imports rectangular data sets from plain-text files.
  2. **tibble**: The specific implementation of rectangular data sets in the Tidyverse is called a "tibble," and tibble objects inherit from the `"data.frame"` class.
  3. **tidyr**: Provides functions to create and maintain rectangular data sets (e.g., to transform rectangular data sets between "wide" and "long" form).
  4. **dplyr**: Provides functions for data manipulation (e.g., adding variables to an existing data set).
  5. **stringr**: Provides functions for manipulating text (character-string) data (e.g., searching for text).
  6. **forcats**: Provides functions for manipulating R factors (e.g., changing the order of levels of a factor).
  7. **purrr**: Provides R programming tools (e.g., alternatives to iteration).
  8. **ggplot2**: A comprehensive alternative graphics system for R (to be discussed when we take up R graphics, and a package that is slightly out-of-place in the Tidyverse).

- There are other Tidyverse packages, which can be installed and loaded separately, most notably:

- There are other Tidyverse packages, which can be installed and loaded separately, most notably:
  - **haven**: Imports data from other statistical packages.

- There are other Tidyverse packages, which can be installed and loaded separately, most notably:
  - **haven**: Imports data from other statistical packages.
  - **readxl**: Imports data from Excel files.

- There are other Tidyverse packages, which can be installed and loaded separately, most notably:
  - **haven**: Imports data from other statistical packages.
  - **readxl**: Imports data from Excel files.
  - **lubridate**: For working with dates.

- There are other Tidyverse packages, which can be installed and loaded separately, most notably:
  - **haven**: Imports data from other statistical packages.
  - **readxl**: Imports data from Excel files.
  - **lubridate**: For working with dates.
  - **magrittr**: The style of data manipulation encouraged by the developers of the Tidyverse makes extensive use of the "pipe" operator, %>%, which is provided by the **magritr** package.

- There are other Tidyverse packages, which can be installed and loaded separately, most notably:
  - **haven**: Imports data from other statistical packages.
  - **readxl**: Imports data from Excel files.
  - **lubridate**: For working with dates.
  - **magrittr**: The style of data manipulation encouraged by the developers of the Tidyverse makes extensive use of the "pipe" operator, %>%, which is provided by the **magritr** package.
    - **magrittr** also includes some other programming-oriented functions.

- There are other Tidyverse packages, which can be installed and loaded separately, most notably:
  - **haven**: Imports data from other statistical packages.
  - **readxl**: Imports data from Excel files.
  - **lubridate**: For working with dates.
  - **magrittr**: The style of data manipulation encouraged by the developers of the Tidyverse makes extensive use of the "pipe" operator, %>%, which is provided by the **magritr** package.
    - **magrittr** also includes some other programming-oriented functions.
    - The pipe operator is supplied by several of the core Tidyverse packages.

- There are other Tidyverse packages, which can be installed and loaded separately, most notably:
  - **haven**: Imports data from other statistical packages.
  - **readxl**: Imports data from Excel files.
  - **lubridate**: For working with dates.
  - **magrittr**: The style of data manipulation encouraged by the developers of the Tidyverse makes extensive use of the "pipe" operator, %>%, which is provided by the **magritr** package.
    - **magrittr** also includes some other programming-oriented functions.
    - The pipe operator is supplied by several of the core Tidyverse packages.
    - Pipes can be used with standard R functions.

- There are few, if any, Tidyverse functions that don't have close analogs in the standard R distribution, but the Tidyverse functions are more uniform and many people claim that they are easier to use (possibly because they're unfamiliar with standard R).

Should You Commit to the Tidyverse?

- There are few, if any, Tidyverse functions that don't have close analogs in the standard R distribution, but the Tidyverse functions are more uniform and many people claim that they are easier to use (possibly because they're unfamiliar with standard R).
  - There are hundreds of functions in the core Tidyverse packages. It isn't obvious that it's easier to learn the Tidyverse than to learn standard R.

- There are few, if any, Tidyverse functions that don't have close analogs in the standard R distribution, but the Tidyverse functions are more uniform and many people claim that they are easier to use (possibly because they're unfamiliar with standard R).
    - There are hundreds of functions in the core Tidyverse packages. It isn't obvious that it's easier to learn the Tidyverse than to learn standard R.
- There are both advantages and disadvantages to Tidyverse implementations of ideas.

- There are few, if any, Tidyverse functions that don't have close analogs in the standard R distribution, but the Tidyverse functions are more uniform and many people claim that they are easier to use (possibly because they're unfamiliar with standard R).
  - There are hundreds of functions in the core Tidyverse packages. It isn't obvious that it's easier to learn the Tidyverse than to learn standard R.
- There are both advantages and disadvantages to Tidyverse implementations of ideas.
  - For example, the `print()` method for tibbles is nicer than that for data frames (cf., the `brief()` function in the **car** package), but tibbles don't support row names.

- There are few, if any, Tidyverse functions that don't have close analogs in the standard R distribution, but the Tidyverse functions are more uniform and many people claim that they are easier to use (possibly because they're unfamiliar with standard R).
    - There are hundreds of functions in the core Tidyverse packages. It isn't obvious that it's easier to learn the Tidyverse than to learn standard R.
- There are both advantages and disadvantages to Tidyverse implementations of ideas.
    - For example, the `print()` method for tibbles is nicer than that for data frames (cf., the `brief()` function in the **car** package), but tibbles don't support row names.
- Tidyverse tools often don't play well with non-Tidyverse tools.

- There are few, if any, Tidyverse functions that don't have close analogs in the standard R distribution, but the Tidyverse functions are more uniform and many people claim that they are easier to use (possibly because they're unfamiliar with standard R).
  - There are hundreds of functions in the core Tidyverse packages. It isn't obvious that it's easier to learn the Tidyverse than to learn standard R.
- There are both advantages and disadvantages to Tidyverse implementations of ideas.
  - For example, the `print()` method for tibbles is nicer than that for data frames (cf., the `brief()` function in the **car** package), but tibbles don't support row names.
- Tidyverse tools often don't play well with non-Tidyverse tools.
  - For example, the **data.table** package implements a data frame alternative that is superior to tibbles for large data sets, but data.tables aren't well supported by Tidyverse functions.

- R is a programming language, and in many cases the simplest and most direct solution to a problem is to write a program.

- R is a programming language, and in many cases the simplest and most direct solution to a problem is to write a program.
  - Using the Tidyverse tools effectively requires some programming skills, and a beginner's time might be better spent learning more general basic R programming.

- R is a programming language, and in many cases the simplest and most direct solution to a problem is to write a program.
  - Using the Tidyverse tools effectively requires some programming skills, and a beginner's time might be better spent learning more general basic R programming.
- For an interesting general critique of the Tidyverse (with which I don't entirely agree), see an essay by Norm Matloff at `https://github.com/matloff/TidyverseSkeptic`.

Outline

- The binary logit model is

$$\Pr(Y_i = 1) = \phi_i = \frac{1}{1 + exp(-\mathrm{x}_i^T \boldsymbol{\beta})}$$

where

- The binary logit model is

$$\Pr(Y_i = 1) = \phi_i = \frac{1}{1 + exp(-\mathrm{x}_i^T \boldsymbol{\beta})}$$

where

  - X is the model matrix, with x_i^T as its ith row;

- The binary logit model is

$$\Pr(Y_i = 1) = \phi_i = \frac{1}{1 + exp(-\mathrm{x}_i^T \boldsymbol{\beta})}$$

where
  - X is the model matrix, with x_i^T as its ith row;
  - y is the response vector (containing 0s and 1s) with Y_i as its ith element;

- The binary logit model is

$$\Pr(Y_i = 1) = \phi_i = \frac{1}{1 + exp(-\mathrm{x}_i^T \boldsymbol{\beta})}$$

where
  - X is the model matrix, with x_i^T as its ith row;
  - y is the response vector (containing 0s and 1s) with Y_i as its ith element;
  - $\boldsymbol{\beta}$ is the vector of logistic-regression parameters.

- The log-likelihood for the model is

$$\log_e L(\boldsymbol{\beta}) = \sum y_i \log_e \phi_i + (1 - y_i) \log_e (1 - \phi_i)$$

- The log-likelihood for the model is

$$\log_e L(\boldsymbol{\beta}) = \sum y_i \log_e \phi_i + (1 - y_i) \log_e (1 - \phi_i)$$

- The gradient (the vector of partial derivatives) of the log-likelihood with respect to the parameters is

$$\frac{\partial \log_e L}{\partial \boldsymbol{\beta}} = \sum (y_i - \phi_i)\mathbf{x}_i$$

- The log-likelihood for the model is

$$\log_e L(\boldsymbol{\beta}) = \sum y_i \log_e \phi_i + (1 - y_i) \log_e (1 - \phi_i)$$

- The gradient (the vector of partial derivatives) of the log-likelihood with respect to the parameters is

$$\frac{\partial \log_e L}{\partial \boldsymbol{\beta}} = \sum (y_i - \phi_i) \mathrm{x}_i$$

- The Hessian (the matrix of second-order partial derivatives) of the log-likelihood is

$$\frac{\partial \log_e L}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = \mathrm{X}^T \mathrm{V} \mathrm{X}$$

where $\mathrm{V} = \mathrm{diag}\{\phi_i(1 - \phi_i)\}$. The variance-covariance matrix of the estimated regression coefficients is the inverse of the Hessian.

- The log-likelihood for the model is

$$\log_e L(\boldsymbol{\beta}) = \sum y_i \log_e \phi_i + (1 - y_i) \log_e (1 - \phi_i)$$

- The gradient (the vector of partial derivatives) of the log-likelihood with respect to the parameters is

$$\frac{\partial \log_e L}{\partial \boldsymbol{\beta}} = \sum (y_i - \phi_i)\mathrm{x}_i$$

- The Hessian (the matrix of second-order partial derivatives) of the log-likelihood is

$$\frac{\partial \log_e L}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = \mathrm{X}^T \mathrm{V} \mathrm{X}$$

  where $\mathrm{V} = \mathrm{diag}\{\phi_i(1 - \phi_i)\}$. The variance-covariance matrix of the estimated regression coefficients is the inverse of the Hessian.
- Setting the gradient to 0 produces nonlinear estimating equations for $\boldsymbol{\beta}$, which have to be solved iteratively, possibly using the information in the Hessian.

- Newton-Raphson is a general method for solving nonlinear equations iteratively.

- Newton-Raphson is a general method for solving nonlinear equations iteratively.
- Here:

- Newton-Raphson is a general method for solving nonlinear equations iteratively.
- Here:
  1. Choose initial estimates of the regression coefficients, such as $b_0 = 0$.

- Newton-Raphson is a general method for solving nonlinear equations iteratively.
- Here:
  1. Choose initial estimates of the regression coefficients, such as $\mathsf{b}_0 = 0$.
  2. At each iteration t, update the coefficients:

$$\mathsf{b}_t = \mathsf{b}_{t-1} + (\mathsf{X}^T \mathsf{V}_{t-1} \mathsf{X})^{-1} \mathsf{X}^T (\mathsf{y} - \mathsf{p}_{t-1})$$

  where

- Newton-Raphson is a general method for solving nonlinear equations iteratively.
- Here:
  1. Choose initial estimates of the regression coefficients, such as $\mathsf{b}_0 = 0$.
  2. At each iteration t, update the coefficients:

$$\mathsf{b}_t = \mathsf{b}_{t-1} + (\mathsf{X}^T \mathsf{V}_{t-1} \mathsf{X})^{-1} \mathsf{X}^T (\mathsf{y} - \mathsf{p}_{t-1})$$

  where

  - $\mathsf{p}_{t-1} = \{1/[1 + \exp(-\mathsf{x}_i^T \mathsf{b}_{t-1})]\}$ is the vector of fitted response probabilities from the previous iteration.

- Newton-Raphson is a general method for solving nonlinear equations iteratively.
- Here:
  1. Choose initial estimates of the regression coefficients, such as $\mathbf{b}_0 = 0$.
  2. At each iteration t, update the coefficients:

$$\mathbf{b}_t = \mathbf{b}_{t-1} + (\mathbf{X}^T \mathbf{V}_{t-1} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{y} - \mathbf{p}_{t-1})$$

where

- $\mathbf{p}_{t-1} = \{1/[1 + \exp(-\mathbf{x}_i^T \mathbf{b}_{t-1})]\}$ is the vector of fitted response probabilities from the previous iteration.
- $\mathbf{V}_{t-1} = \mathrm{diag}\{p_{i,t-1}(1 - p_{i,t-1})\}$.

- Newton-Raphson is a general method for solving nonlinear equations iteratively.
- Here:
  1. Choose initial estimates of the regression coefficients, such as $\mathbf{b}_0 = 0$.
  2. At each iteration t, update the coefficients:

$$\mathbf{b}_t = \mathbf{b}_{t-1} + (\mathbf{X}^T \mathbf{V}_{t-1} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{y} - \mathbf{p}_{t-1})$$

     where
     - $\mathbf{p}_{t-1} = \{1/[1 + \exp(-\mathbf{x}_i^T \mathbf{b}_{t-1})]\}$ is the vector of fitted response probabilities from the previous iteration.
     - $\mathbf{V}_{t-1} = \text{diag}\{p_{i,t-1}(1 - p_{i,t-1})\}$.
  3. Step 2 is repeated until $\mathbf{b}_t$ is close enough to $\mathbf{b}_{t-1}$, at which point the MLE $\widehat{\boldsymbol{\beta}} \approx \mathbf{b}_t$. The estimated asymptotic covariance matrix of the coefficients is given by $\widehat{V}(\widehat{\boldsymbol{\beta}}) \approx (\mathbf{X}^T \mathbf{V}_t \mathbf{X})^{-1}$.

- Three standard object-oriented programming systems in R: S3, S4, reference classes. Of these, the S3 object system is the one most commonly used in applications.

- Three standard object-oriented programming systems in R: S3, S4, reference classes. Of these, the S3 object system is the one most commonly used in applications.
- How the S3 object system works:

- Three standard object-oriented programming systems in R: S3, S4, reference classes. Of these, the S3 object system is the one most commonly used in applications.
- How the S3 object system works:
  - Method dispatch of the generic function `generic()` for the object named `object`, which is of of class `"class"` (where ⇒ means "the interpreter looks for and dispatches"):
    `generic(object)` ⇒ `generic.class(object)` ⇒ `generic.default(object)`

- Three standard object-oriented programming systems in R: S3, S4, reference classes. Of these, the S3 object system is the one most commonly used in applications.
- How the S3 object system works:
  - Method dispatch of the generic function `generic()` for the object named `object`, which is of of class `"class"` (where ⇒ means "the interpreter looks for and dispatches"):
    `generic(object)` ⇒ `generic.class(object)` ⇒ `generic.default(object)`
    - For example, summarizing an object `mod` of class `"lm"`:
      `summary(mod)` ⇒ `summary.lm(mod)`

- Three standard object-oriented programming systems in R: S3, S4, reference classes. Of these, the S3 object system is the one most commonly used in applications.
- How the S3 object system works:
  - Method dispatch of the generic function `generic()` for the object named `object`, which is of of class `"class"` (where ⇒ means "the interpreter looks for and dispatches"):
    `generic(object)` ⇒ `generic.class(object)` ⇒ `generic.default(object)`
    - For example, summarizing an object `mod` of class `"lm"`:
      `summary(mod)` ⇒ `summary.lm(mod)`
  - Objects can have more than one class, in which case the first applicable method is used.

- Three standard object-oriented programming systems in R: S3, S4, reference classes. Of these, the S3 object system is the one most commonly used in applications.
- How the S3 object system works:
  - Method dispatch of the generic function `generic()` for the object named `object`, which is of of class `"class"` (where ⇒ means "the interpreter looks for and dispatches"):
    `generic(object)` ⇒ `generic.class(object)` ⇒ `generic.default(object)`
    - For example, summarizing an object `mod` of class `"lm"`:
      `summary(mod)` ⇒ `summary.lm(mod)`
  - Objects can have more than one class, in which case the first applicable method is used.
    - For example, objects produced by `glm()` are of class `c("glm", "lm")` and therefore can *inherit* methods from class `"lm"`.

- Three standard object-oriented programming systems in R: S3, S4, reference classes. Of these, the S3 object system is the one most commonly used in applications.
- How the S3 object system works:
  - Method dispatch of the generic function `generic()` for the object named `object`, which is of of class `"class"` (where ⇒ means "the interpreter looks for and dispatches"):
    `generic(object)` ⇒ `generic.class(object)` ⇒ `generic.default(object)`
    - For example, summarizing an object `mod` of class `"lm"`:
      `summary(mod)` ⇒ `summary.lm(mod)`
  - Objects can have more than one class, in which case the first applicable method is used.
    - For example, objects produced by `glm()` are of class `c("glm", "lm")` and therefore can *inherit* methods from class `"lm"`.
    - Methods are searched from left to right, so if `mod` is produced by a call to `glm()`, and if `generic(mod)` is called, then methods are invoked in the order
      `generic(mod)` ⇒ `generic.glm(mod)` ⇒ `generic.lm(mod)` ⇒ `generic.default(mod)`
      and will fail if none of these three methods are available.

- Generic functions take the form:

```
generic <- function(object, other, named, arguments, ...){
    UseMethod("generic")
}
```

where the ellipses (...) "soak up" additional arguments not named in the generic function that may be passed to specific methods when `generic()` is called.

- Generic functions take the form:

```
generic <- function(object, other, named, arguments, ...){
    UseMethod("generic")
}
```

where the ellipses (`...`) "soak up" additional arguments not named in the generic function that may be passed to specific methods when `generic()` is called.

- For example, the R `summary()` function is defined as

```
summary <- function(object, ...){
    UseMethod("summary")
}
```

and `summary.lm()` is

```
summary.lm <- function (object, correlation=FALSE, symbolic.cor=FALSE, ...){
     etc.
}
```