



**CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY (A)**

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. www.cbit.ac.in



ISO Certified
9001:2015

COMMITTED TO
RESEARCH,
INNOVATION AND
EDUCATION

44
years

UNIT – 3

Dr T Prathima, Dept. of IT, CBIT

Supervised Learning: Using Linear Regression, Using Logistic Regression.

Unsupervised methods: Cluster Analysis, Association rules.

Exploring Advanced Methods: Using bagging and random forest, using generalized additive models, using kernel methods to increase data separation.

Zumel, N., Mount, J., Porzak, J., “Practical data science with R”, 2nd Ed, Shelter Island, NY: Manning, 2019.

Supervised Learning: Linear Regression, Logistic Regression.

- Using linear regression to predict quantities
- Using logistic regression to predict probabilities or categories
- Extracting relations and advice from linear models
- Interpreting the diagnostics from R's `lm()` call
- Interpreting the diagnostics from R's `glm()` call
- Using regularization via the `glmnet` package to address issues that can arise with linear models.

Linear Regression

- Linear models are especially useful when you don't want only to predict an outcome, but also to know the relationship between the input variables and the outcome.
- In regression analysis, the coefficients in the regression equation are estimates of the actual population parameters. We want these coefficient estimates to be the best possible estimates!
- If you're trying to predict a numerical quantity like profit, cost, or sales volume, you should always try linear regression first. If it works well, you're done; if it fails, the detailed diagnostics produced can give you a good clue as to what methods you should try next.
- *Suppose you want to predict how many pounds a person on a diet and exercise plan will lose in a month. You will base that prediction on other facts about that person, like how much they reduce their average daily caloric intake over that month and how many hours a day they exercised. In other words, for every person i , you want to predict `pounds_lost[i]` based on `daily_cals_down[i]` and `daily_exercise[i]`*
- Linear regression assumes that the outcome `pounds_lost` is linearly related to each of the inputs `daily_cals_down[i]` and `daily_exercise[i]`. This means that the relationship between (for instance) `daily_cals_down[i]` and `pounds_lost` looks like a (noisy) straight line
- The relationship between `daily_exercise` and `pounds_lost` would similarly be a straight line $\text{pounds_lost} = bc_0 + b.cals * \text{daily_cals_down}$
- This means that for every unit change in `daily_cals_down` (every calorie reduced), the value of `pounds_lost` changes by `b.cals`, no matter what the starting value of `daily_cals_down` was. To make it concrete, suppose $\text{pounds_lost} = 3 + 2 * \text{daily_cals_down}$. Then increasing `daily_cals_down` by one increases `pounds_lost` by 2, no matter what value of `daily_cals_down` you start with.
- This would not be true for, say, $\text{pounds_lost} = 3 + 2 * (\text{daily_cals_down}^2)$.
- Linear regression further assumes that the total pounds lost is a *linear combination* of our variables `daily_cals_down[i]` and `daily_exercise[i]`, or the sum of the pounds lost due to reduced caloric intake, and the pounds lost due to exercise.

- Suppose that $y[i]$ is the numeric quantity you want to predict (called the *dependent* or *response* variable), and $x[i,]$ is a row of inputs that corresponds to output $y[i]$ (the $x[i,]$ are the *independent* or *explanatory* variables).
- Linear regression attempts to find a function $f(x)$ such that
- $$y[i] \sim f(x[i,]) + e[i] = b[0] + b[1] * x[i, 1] + \dots + b[n] * x[i, n] + e[i]$$
- You want numbers $b[0], \dots, b[n]$ (called the *coefficients* or *betas*) such that $f(x[i,])$ is as near as possible to $y[i]$ for all $(x[i,], y[i])$ pairs in the training data.
- R supplies a one-line command to find these coefficients: `lm()`.
- The last term in equation $e[i]$, represents what are called *unsystematic errors*, or noise.
- Unsystematic errors are defined to all have a mean value of 0 (so they don't represent a net upward or net downward bias) and are defined as uncorrelated with $x[i,]$.
- In other words, $x[i,]$ should not encode information about $e[i]$ (or vice versa).

EXTRAPOLATION IS NOT AS SAFE AS INTERPOLATION

- In general, you should try to use a model only for *interpolation*: predicting for new data that falls inside the range of your training data.
- *Extrapolation* (predicting for new data outside the range observed during training) is riskier for any model. It's especially risky for linear models, unless you know that the system that you are modeling is truly linear.

Listing 7.1 Loading the PUMS data and fitting a model

```
psub <- readRDS("psub.RDS")
```

```
set.seed(3454351)
```

```
gp <- runif(nrow(psub))
```

Makes a random variable to
group and partition the data

```
dtrain <- subset(psub, gp >= 0.5)
```

```
dtest <- subset(psub, gp < 0.5)
```

Splits 50–50 into training and test sets

```
model <- lm(log10(PINCP) ~ AGE + SEX + COW + SCHL, data = dtrain)
```

```
dtest$predLogPINCP <- predict(model, newdata = dtest)
```

```
dtrain$predLogPINCP <- predict(model, newdata = dtrain)
```

Fits a linear model
to log(income)

Gets the predicted log(income)
on the test and training sets

Statisticians call the quantity to be predicted the *dependent variable* and the variables/columns used to make the prediction the *independent variables*. We find it is easier to call the quantity to be predicted the y and the variables used to make the predictions the x s. Our formula is this: $\log_{10}(\text{PINCP}) \sim \text{AGEP} + \text{SEX} + \text{COW} + \text{SCHL}$, which is read “Predict the log base 10 of income as a function of age, sex, employment class, and education.”¹ The overall method is demonstrated in figure 7.4.

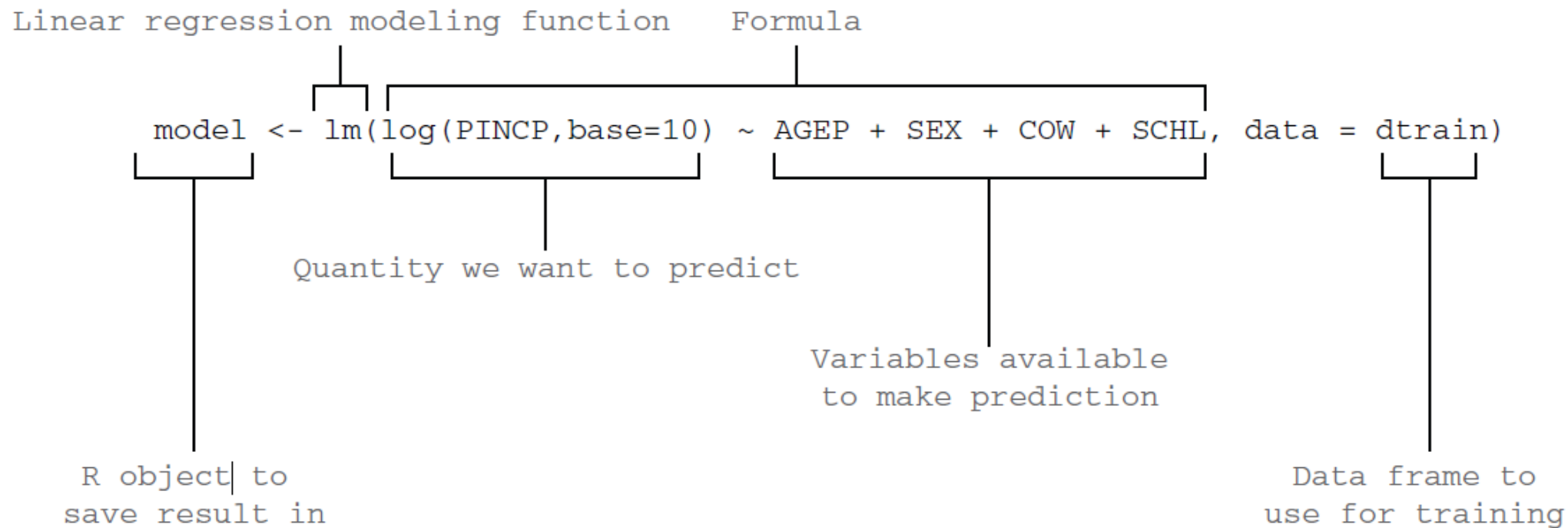


Figure 7.4 Building a linear model using `lm()`

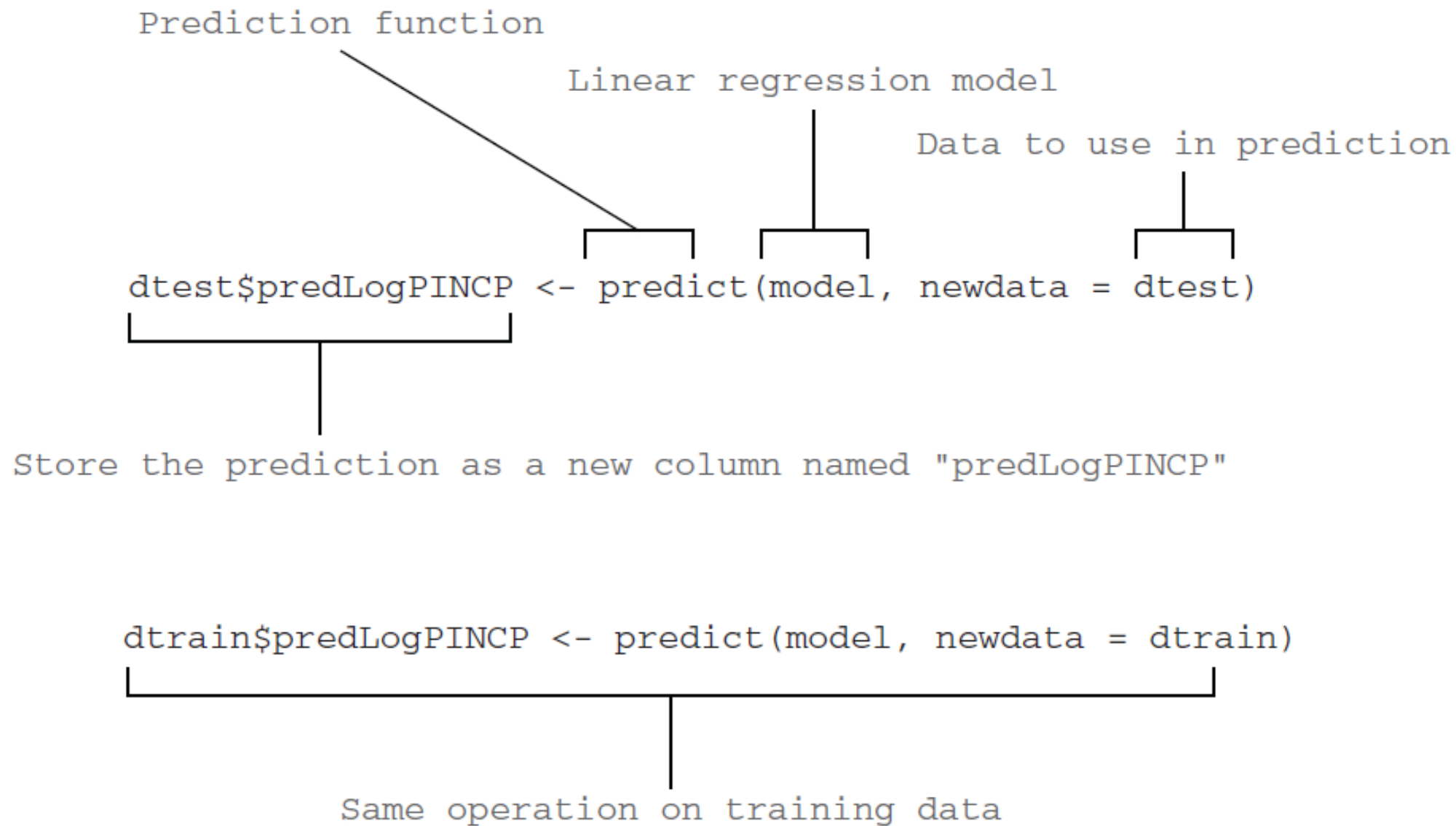


Figure 7.5 Making predictions with a linear regression model

CHARACTERIZING PREDICTION QUALITY

Before publicly sharing predictions, you want to inspect both the predictions and model for quality. We recommend plotting the actual y (in this case, predicted income) that you're trying to predict as if it were a function of your prediction. In this case, plot $\log_{10}(\text{PINCP})$ as if it were a function of predLogPINCP . If the predictions are very good, then the plot will be dots arranged near the line $y=x$, which we call *the line of perfect prediction* (the phrase is not standard terminology; we use it to make talking about the graph easier). The steps to produce this, illustrated in figure 7.6, are shown in the next listing.

Listing 7.2 Plotting log income as a function of predicted log income

```
library('ggplot2')
ggplot(data = dtest, aes(x = predLogPINCP, y = log10(PINCP))) +
  geom_point(alpha = 0.2, color = "darkgray") +
  geom_smooth(color = "darkblue") +
  geom_line(aes(x = log10(PINCP), ← Plots the line x=y
               y = log10(PINCP)),
            color = "blue", linetype = 2) +
  coord_cartesian(xlim = c(4, 5.25), ← Limits the range of the
                  ylim = c(3.5, 5.5)) graph for legibility
```

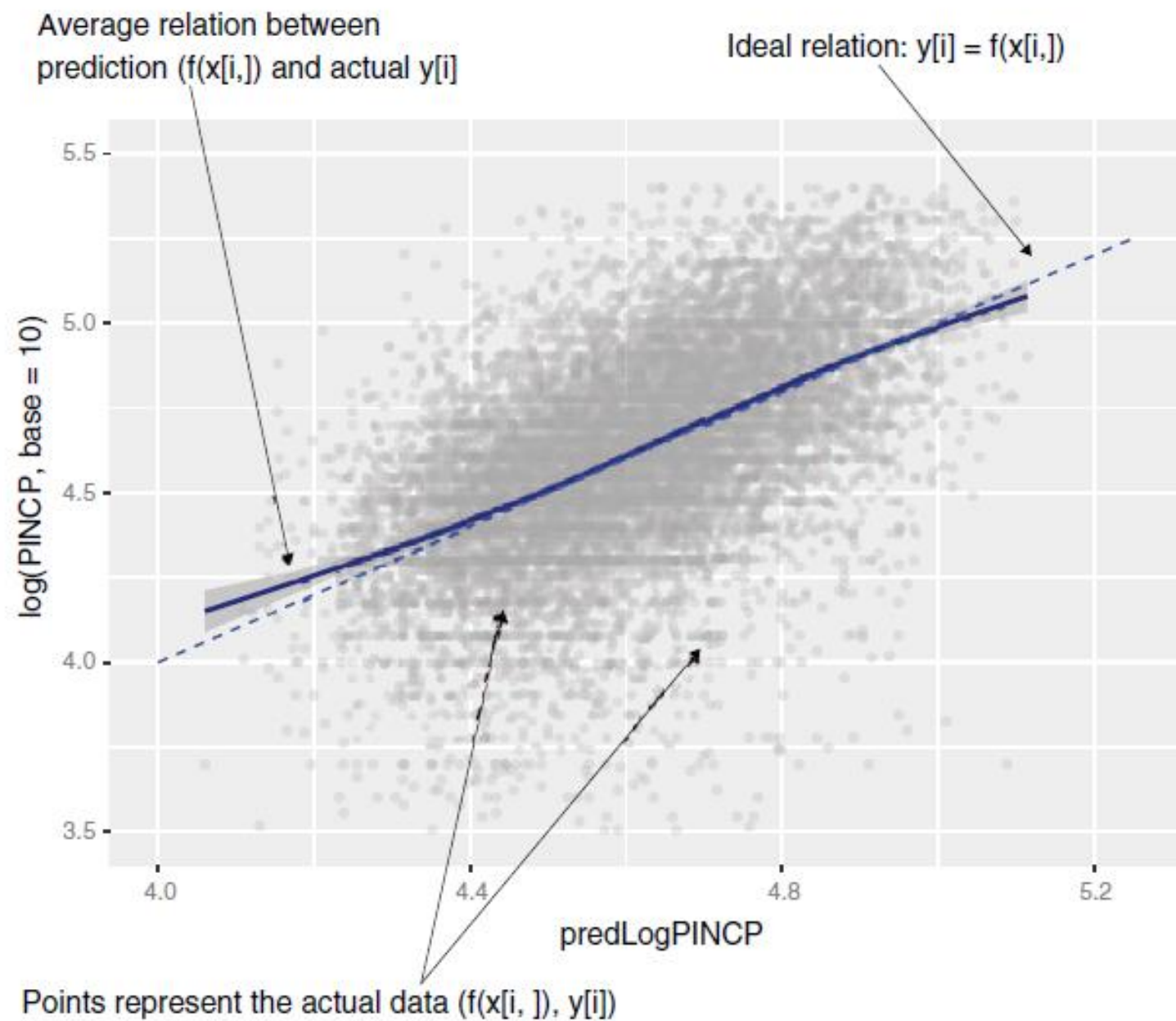


Figure 7.6 Plot of actual log income as a function of predicted log income

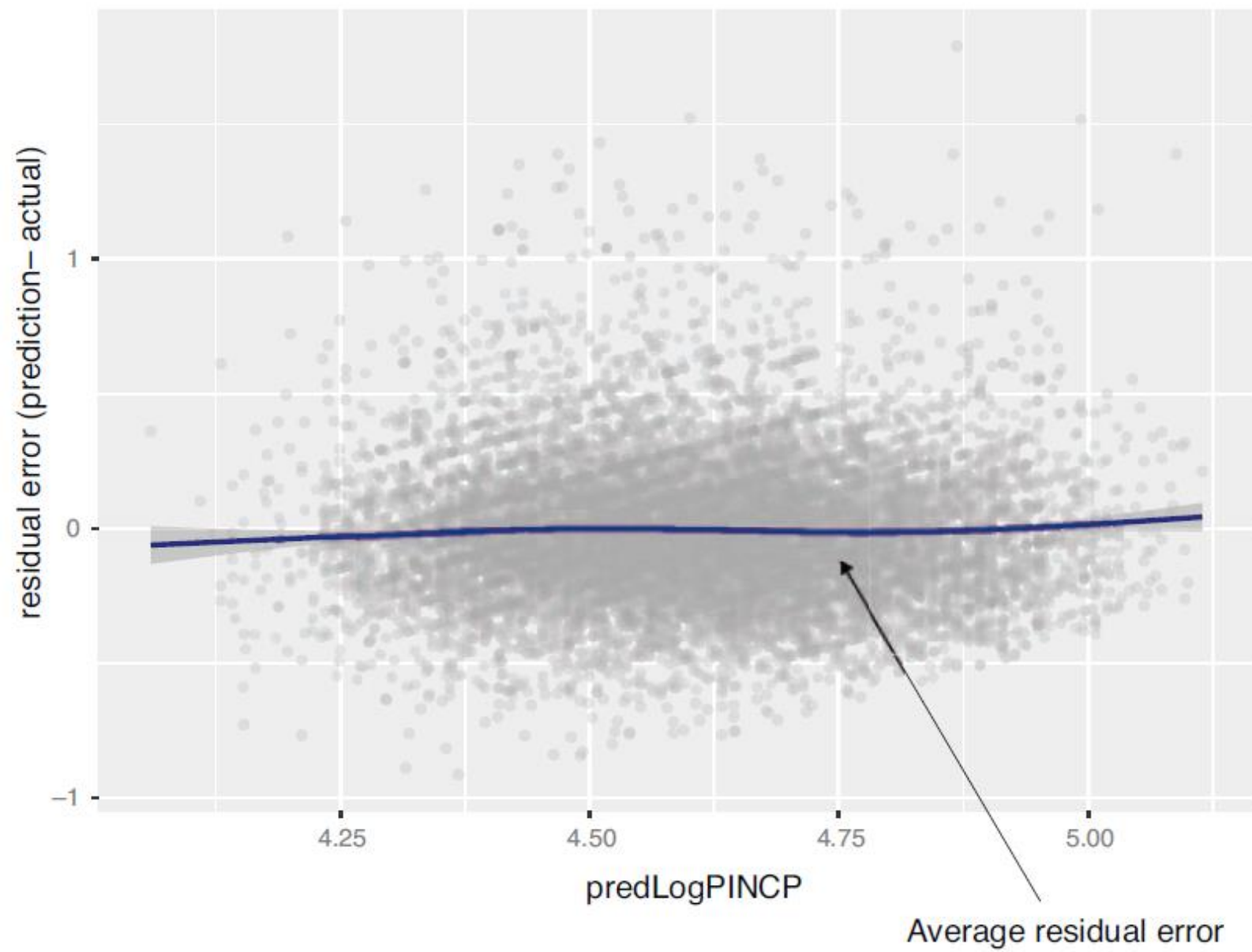


Figure 7.7 Plot of residual error as a function of prediction

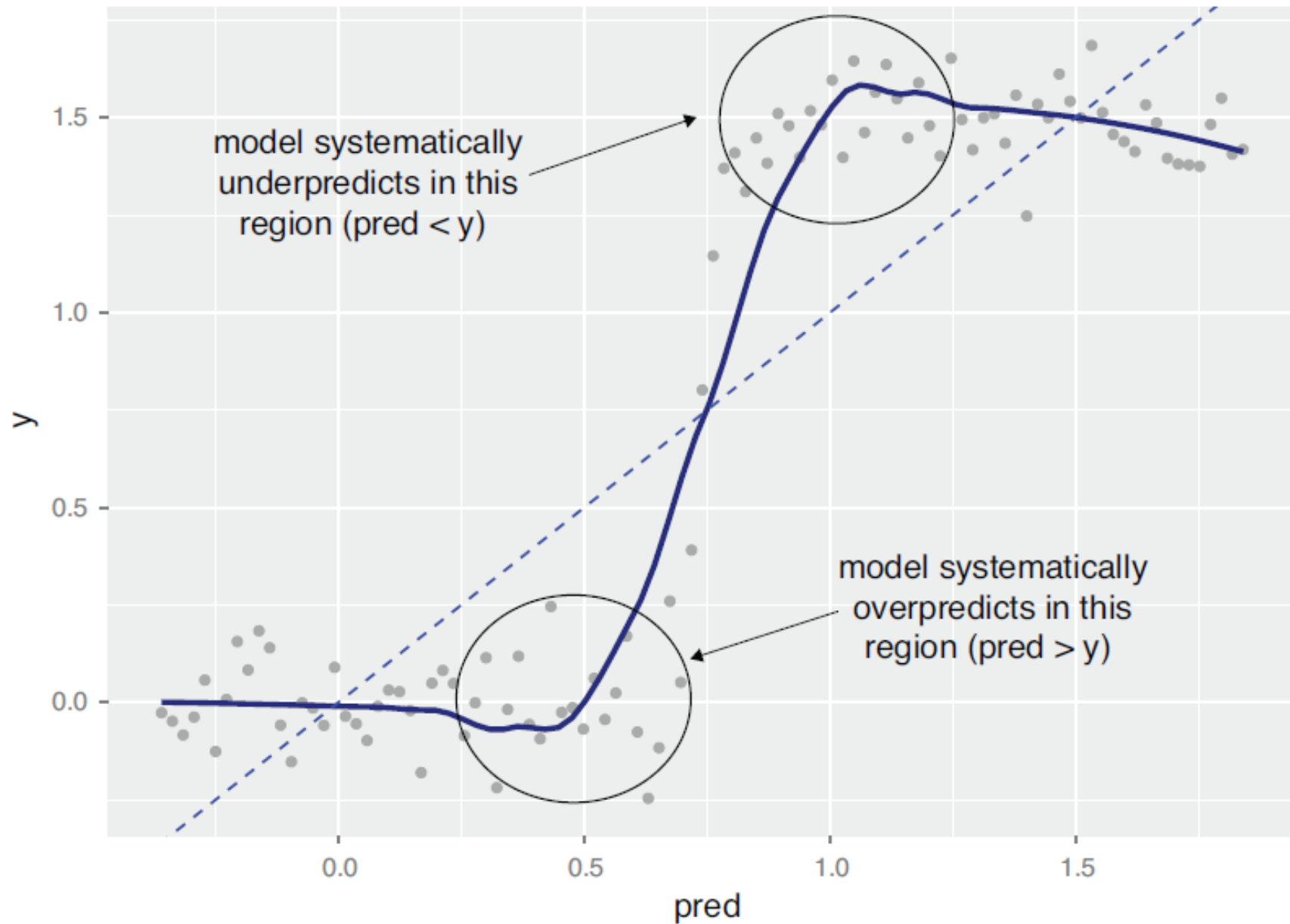




Figure 7.8 An example of systematic errors in model predictions



Prediction quality

- *On average, are the predictions correct?*
- Does the smoothing curve lie more or less along the line of perfect prediction? Ideally, the points will all lie very close to that line, but you may instead get a wider cloud of points if your input variables don't explain the output too closely. But if the smoothing curve lies along the line of perfect prediction and "down the middle" of the cloud of points, then the model predicts correctly on average: it underpredicts about as much as it overpredicts.
- *Are there systematic errors?*
- If the smoothing curve veers off the line of perfect prediction too much, as in figure 7.8, this is a sign of systematic under- or overprediction in certain ranges: the error is correlated with the prediction. Systematic errors indicate that the system is not "linear enough" for a linear model to be a good fit, so you should try one of the different modeling approaches
- *R-squared and RMSE*
- R-squared is a measure of how well the model "fits" the data, or its "goodness of fit." R-squared can be thought of as what fraction of the y variation is explained by the model. You want R-squared to be fairly large (1.0 is the largest you can achieve) and R-squareds that are similar on test and training. A significantly lower R-squared on test data is a symptom of an overfit model that looks good in training and won't work in production. In this case, the R-squareds were about 0.3 for both the training and test data. We'd like to see R-squareds higher than this (say, 0.7–1.0). So the model is of low quality, but not overfit. For well-fit models, R-squared is also equal to the square of the correlation between the predicted values and actual training values.
- *Another good measure to consider is root mean square error (RMSE).*
- You can think of the RMSE as a measure of the width of the data cloud around the line of perfect prediction. We'd like RMSE to be small, and one way to achieve this is to introduce more useful, explanatory variables.

Listing 7.4 Computing R-squared

```
rsq <- function(y, f) { 1 - sum((y - f)^2)/sum((y - mean(y))^2) }  
  
rsq(log10(dtrain$PINCP), dtrain$predLogPINCP)  R-squared of the model  
on the training data  
## [1] 0.2976165  
  
rsq(log10(dtest$PINCP), dtest$predLogPINCP)  R-squared of the model  
on the test data  
## [1] 0.2911965
```

Listing 7.5 Calculating root mean square error

```
rmse <- function(y, f) { sqrt(mean((y-f)^2)) }  
  
rmse(log10(dtrain$PINCP), dtrain$predLogPINCP)  RMSE of the model  
on the training data  
## [1] 0.2685855  
  
rmse(log10(dtest$PINCP), dtest$predLogPINCP)  RMSE of the model  
on the test data  
## [1] 0.2675129
```

Reading the model summary and characterizing coefficient quality

- THE ORIGINAL MODEL CALL

- The first part of the `summary()` is how the `lm()` model was constructed:
- Call:
- `lm(formula = log10(PINCP) ~ AGE + SEX + COW + SCHL, data = dtrain)`
- This is a good place to double-check whether you used the correct data frame, performed your intended transformations, and used the right variables. For example, you can double-check whether you used the data frame `dtrain` and not the data frame `dtest`.

- THE RESIDUALS SUMMARY

- The next part of the `summary()` is the residuals summary:
- Residuals:
- | Min | 1Q | Median | 3Q | Max |
|---------|---------|--------|--------|--------|
| -1.5038 | -0.1354 | 0.0187 | 0.1710 | 0.9741 |


```
Call:
lm(formula = log(PINCP, base = 10) ~ AGEP + SEX + COW + SCHL,
    data = dtrain)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.5038	-0.1354	0.0187	0.1710	0.9741

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	4.0058856	0.0144265	277.676	< 2e-16	***
AGEP	0.0115985	0.0003032	38.259	< 2e-16	***
SEXFemale	-0.1076883	0.0052567	-20.486	< 2e-16	***
COWFederal government employee	0.0638672	0.0157521	4.055	5.06e-05	***
COWLocal government employee	-0.0297093	0.0107370	-2.767	0.005667	**
COWPrivate not-for-profit employee	-0.0330196	0.0102449	-3.223	0.001272	**
COWSelf employed incorporated	0.0145475	0.0164742	0.883	0.377232	
COWSelf employed not incorporated	-0.1282285	0.0134708	-9.519	< 2e-16	***
COWState government employee	-0.0479571	0.0123275	-3.890	0.000101	***
SCHLRegular high school diploma	0.1135386	0.0107236	10.588	< 2e-16	***
SCHLGED or alternative credential	0.1216670	0.0173038	7.031	2.17e-12	***
SCHLsome college credit, no degree	0.1838278	0.0106461	17.267	< 2e-16	***
SCHLAssociate's degree	0.2387045	0.0123568	19.318	< 2e-16	***
SCHLBachelor's degree	0.3637114	0.0105810	34.374	< 2e-16	***
SCHLMaster's degree	0.4445777	0.0127100	34.978	< 2e-16	***
SCHLProfessional degree	0.5111167	0.0201800	25.328	< 2e-16	***
SCHLDoctorate degree	0.4818700	0.0245162	19.655	< 2e-16	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2688 on 11186 degrees of freedom
Multiple R-squared: 0.2976, Adjusted R-squared: 0.2966
F-statistic: 296.2 on 16 and 11186 DF, p-value: < 2.2e-16

Model call summary

Residuals summary

Coefficients table

Model quality summary

Figure 7.10 Model summary

Finding relations and extracting advice: REPORTED COEFFICIENTS

- All the information in a linear regression model is stored in a block of numbers called the *coefficients*. The coefficients are available through the `coefficients(model)` function.
- Our original modeling variables were only AGE, SEX, COW (class of work), and SCHL (schooling/education); yet the model reports many more coefficients than these four.
- The original variable SCHL took on these eight string values plus one more not shown: no high school diploma.
- Each of these possible strings is called a *level*, and SCHL itself is called a *categorical* or *factor* variable.
- The level that isn't shown is called the *reference level* ; the coefficients of the other levels are measured with respect to the reference level.
- For example, in SCHLBachelor's degree we find the coefficient 0.36, which is read as “The model gives a 0.36 bonus to log base 10 income for having a bachelor's degree, relative to not having a high school degree.”
- You can solve for the income ratio between someone with a bachelor's degree and the equivalent person (same sex, age, and class of work) without a high school degree

$$\begin{aligned}\log_{10}(\text{income_bachelors}) &= \log_{10}(\text{income_no_hs_degree}) + 0.36 \\ \log_{10}(\text{income_bachelors}) - \log_{10}(\text{income_no_hs_degree}) &= 0.36 \\ (\text{income_bachelors}) / (\text{income_no_hs_degree}) &= 10^{(0.36)}\end{aligned}$$

This means that someone with a bachelor's degree will tend to have an income about $10^{0.36}$, or 2.29 times higher than the equivalent person without a high school degree.

And under SCHLRegular high school diploma, we find the coefficient 0.11. This is read as “The model believes that having a bachelor's degree tends to add 0.36–0.11 units to the predicted log income, relative to having a high school degree.”

$$\begin{aligned}\log_{10}(\text{income_bachelors}) - \log_{10}(\text{income_no_hs_degree}) &= 0.36 \\ \log_{10}(\text{income_hs}) - \log_{10}(\text{income_no_hs_degree}) &= 0.11 \\ \log_{10}(\text{income_bachelors}) - \log_{10}(\text{income_hs}) &= 0.36 - 0.11 \quad \leftarrow \text{Subtracts the second equation from the first} \\ (\text{income_bachelors}) / (\text{income_hs}) &= 10^{(0.36 - 0.11)}\end{aligned}$$

The modeled relation between the bachelor's degree holder's expected income and the high school graduate's (all other variables being equal) is $10^{(0.36 - 0.11)}$, or about 1.8 times greater. The advice: college is worth it if you can find a job (remember that we limited the analysis to the fully employed, so this is assuming you can find a job).

- SEX and COW are also discrete variables, with reference levels Male and Employee of a private for profit [company], respectively.
- The coefficients that correspond to the different levels of SEX and COW can be interpreted in a manner similar to the education level.
- AGEP is a continuous variable with coefficient 0.0116. You can interpret this as saying that a one-year increase in age adds a 0.0116 bonus to log income;
- In other words, an increase in age of one year corresponds to an increase of income of $10^{0.0116}$, or a factor of 1.027—about a 2.7% increase in income (all other variables being equal).

Intercept

- One way to interpret the intercept is to think of it as “the prediction for the reference subject”—that is, the subject who takes on the values of all the reference levels for the categorical inputs, and zero for the continuous variables. Note that this may not be a physically plausible subject.
- In our example, the reference subject would be a male employee of a private for profit company, with no high school degree, who is zero years old.
- If such a person could exist, the model would predict their log base 10 income to be about 4.0, which corresponds to an income of \$10,000.

Indicator variables

- Most modeling methods handle a string-valued (categorical) variable with n possible levels by converting it to n (or $n-1$) binary variables, or *indicator variables*.
- R has commands to explicitly control the conversion of string-valued variables into well-behaved indicators: `as.factor()` creates categorical variables from string variables; `relevel()` allows the user to specify the reference level.
- But beware of variables with a very large number of levels, like ZIP codes. The runtime of linear (and logistic) regression increases as roughly the cube of the number of coefficients.
- Too many levels (or too many variables in general) will bog the algorithm down and require much more data for reliable inference.

Residuals

- Recall that the residuals are the errors in prediction: `log10(dtrain$PINCP) - predict(model, newdata=dtrain)`.
- In linear regression, the residuals are everything. Most of what you want to know about the quality of your model fit is in the residuals.
- You can calculate useful summaries of the residuals for both the training and test sets

Listing 7.6 Summarizing residuals

```
( resids_train <- summary(log10(dtrain$PINCP) -  
  predict(model, newdata = dtrain)) )  
##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.  
## -1.5038 -0.1354  0.0187  0.0000  0.1710  0.9741  
  
( resids_test <- summary(log10(dtest$PINCP) -  
  predict(model, newdata = dtest)) )  
##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.  
## -1.789150 -0.130733  0.027413  0.006359  0.175847  0.912646
```

In linear regression, the coefficients are chosen to minimize the sum of squares of the residuals. This is why the method is also often called the *least squares method*. So for good models, you expect the residuals to be small.

Likely error in estimate

Coefficient estimate

t value: How far coefficient estimate is from zero (in units of likely error)

p value: Probability of a t value this large by mere chance

Name of coefficient

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	4.0058856	0.0144265	277.676	< 2e-16	***
AGEP	0.0115985	0.0003032	38.259	< 2e-16	***
SEXFemale	-0.1076883	0.0052567	-20.486	< 2e-16	***
COWFederal government employee	0.0638672	0.0157521	4.055	5.06e-05	***
COWLocal government employee	-0.0297093	0.0107370	-2.767	0.005667	**
COWPrivate not-for-profit employee	-0.0330196	0.0102449	-3.223	0.001272	**
COWSelf employed incorporated	0.0145475	0.0164742	0.883	0.377232	
COWSelf employed not incorporated	-0.1282285	0.0134708	-9.519	< 2e-16	***
COWState government employee	-0.0479571	0.0123275	-3.890	0.000101	***
SCHLRegular high school diploma	0.1135386	0.0107236	10.588	< 2e-16	***
SCHLGED or alternative credential	0.1216670	0.0173038	7.031	2.17e-12	***
SCHLsome college credit, no degree	0.1838278	0.0106461	17.267	< 2e-16	***
SCHLAssociate's degree	0.2387045	0.0123568	19.318	< 2e-16	***
SCHLBachelor's degree	0.3637114	0.0105810	34.374	< 2e-16	***
SCHLMaster's degree	0.4445777	0.0127100	34.978	< 2e-16	***
SCHLProfessional degree	0.5111167	0.0201800	25.328	< 2e-16	***
SCHLDoctorate degree	0.4818700	0.0245162	19.655	< 2e-16	***

Figure 7.11 Model summary coefficient columns

Standard Error

- The standard error of the estimate, denoted se , is a measure of the standard deviation of the errors in a regression model. The standard error of the estimate is a measure of the average deviation of the errors, the difference between the \hat{y} -values predicted by the multiple regression model and the y -values in the sample. The standard error of the estimate for the regression model is the standard deviation of the errors/residuals.
- The value of se tells us, on average, how much the dependent variable differs from the regression model based on the independent variables. When interpreting the standard error of the estimate, remember to be specific to the question, using the actual names of the dependent and independent variables, and include appropriate units. The units of the standard error of the estimate are the same as the units of the dependent variable.
- **SE** represents the standard error of estimation which can be estimated using the following formula: **$SE = S / \sqrt{N}$** ; Where S represents the standard deviation and N represents the total number of data points.

t-test

- The coefficient t-value is a measure of how many standard deviations our coefficient estimate is far away from 0.
- We want it to be far away from zero as this would indicate we could reject the null hypothesis - that is, we could declare a relationship between speed and distance exist.
- In our example, the t-statistic values are relatively far away from zero and are large relative to the standard error, which could indicate a relationship exists.
- In general, t-values are also used to compute p-values.
- The t-value measures the size of the difference relative to the variation in your sample data.
- Put another way, T is simply the calculated difference represented in units of standard error. The greater the magnitude of T, the greater the evidence against the null hypothesis. This means there is greater evidence that there is a significant difference.

```
## Call:
## lm(formula = dist ~ speed.c, data = cars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -29.069  -9.525  -2.272   9.215  43.201
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   42.9800     2.1750  19.761  < 2e-16 ***
## speed.c        3.9324     0.4155   9.464 1.49e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.38 on 48 degrees of freedom
## Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
## F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

$Pr(>t)$

- The $Pr(>t)$ acronym found in the model output relates to the probability of observing any value equal or larger than t .
- A small p-value indicates that it is unlikely we will observe a relationship between the predictor (speed) and response (dist) variables due to chance.
- Typically, a p-value of 5% or less is a good cut-off point. In our model example, the p-values are very close to zero.
- Note the 'signif. Codes' associated to each estimate.
- Three stars (or asterisks) represent a highly significant p-value.
- Consequently, a small p-value for the intercept and the slope indicates that we can reject the null hypothesis which allows us to conclude that there is a relationship between speed and distance.

The p-value and significance

- Generally, people pick a threshold, and call all the coefficients with a p-value below that threshold *statistically significant*, meaning that those coefficients are likely not zero. A common threshold is $p < 0.05$;
- If the p-value for a variable is less than your significance level, your sample data provide enough evidence to reject the null hypothesis for the entire population.
- Your data favor the hypothesis that there is a non-zero correlation.
- Changes in the independent variable are associated with changes in the dependent variable at the population level.
- This variable is statistically significant and probably a worthwhile addition to your regression model.
- On the other hand, when a p value in regression is greater than the significance level, it indicates there is insufficient evidence in your sample to conclude that a non-zero correlation exists.
- The regression output example below shows that the South and North predictor variables are statistically significant because their p-values equal 0.000. On the other hand, East is not statistically significant because its p-value (0.092) is greater than the usual significance level of 0.05.

Coefficients

Term	Coef	SE Coef	T	P
Constant	389.166	66.0937	5.8881	0.000
East	2.125	1.2145	1.7495	0.092
South	5.318	0.9629	5.5232	0.000
North	-24.132	1.8685	-12.9153	0.000

Collinearity also lowers significance

- Sometimes, a predictive variable won't appear significant because it's collinear (or correlated) with another predictive variable.
- For example, if you did try to use both age and number of years in the workforce to predict income, neither variable may appear significant.
- This is because age tends to be correlated with number of years in the workforce. If you remove one of the variables and the other one gains significance, this is a good indicator of correlation.
- If you see coefficients that seem unreasonably large (often of opposite signs), or unusually large standard errors on the coefficients, that may indicate collinear variables.
- Another possible indication of collinearity in the inputs is seeing coefficients with an unexpected sign: for example, seeing that income is *negatively* correlated with years in the workforce.
- The overall model can still predict income quite well, even when the inputs are correlated;
- it just can't determine which variable deserves the credit for the prediction.

OVERALL MODEL QUALITY SUMMARIES

- Degrees of freedom:

- The *degrees of freedom* is the number of data rows minus the number of coefficients fit, in our case, this:
- `(df <- nrow(dtrain) - nrow(summary(model)$coefficients))`
- `## [1] 11186`
- The degrees of freedom is the number of training data rows you have after correcting for the number of coefficients you tried to solve for.
- You want the number of datums in the training set to be large compared to the number of coefficients you are solving for;
- in other words, you want the degrees of freedom to be high. A low degree of freedom indicates that the model you are trying to fit is too complex for the amount of data that you have, and your model is likely to be overfit.
- Overfitting is when you find chance relations in your training data that aren't present in the general population.
- Overfitting is bad: you think you have a good model when you don't.

- Degrees of freedom on test data

- On test data (data not used during training), the degrees of freedom equal the number of rows of data.
- This differs from the case of training data, where, as we have said, the degrees of freedom equal the number of rows of data minus the number of parameters of the model.
- The difference arises from the fact that model training “peeks at” the training data, but not the test data.

Residual standard error

- The *residual standard error* is the sum of the square of the residuals (or the sum of squared error) divided by the degrees of freedom.
- Similar to the RMSE (root mean squared error), except with the number of data rows adjusted to be the degrees of freedom; in R, this is calculated as follows:
- ```
(modelResidualError <- sqrt(sum(residuals(model)^2) / df))
```
- ```
## [1] 0.2687895
```
- The residual standard error is a more conservative estimate of model performance than the RMSE, because it's adjusted for the complexity of the model (the degrees of freedom is less than the number of rows of training data, so the residual standard error is larger than the RMSE).
- Again, this tries to compensate for the fact that more complex models have a higher tendency to overfit the data.

Multiple and adjusted R-squared

- Multiple R-squared is just the R-squared of the model on the training data
- The adjusted R-squared is the multiple R-squared penalized for the number of input variables.
- The reason for this penalty is that, in general, increasing the number of input variables will improve the R-squared on the training data, even if the added variables aren't actually informative.
- This is another way of saying that more-complex models tend to look better on training data due to overfitting, so the adjusted R-squared is a more conservative estimate of the model's goodness of fit.

F-static and its p-value

- The *F-statistic* is similar to the t-values for coefficients
- Just as the t-values are used to calculate p-values on the coefficients, the F-statistic is used to calculate a p-value on the model fit.
- It gets its name from the F-test, which is the technique used to check if two variances—in this case, the variance of the residuals from the constant model and the variance of the residuals from the linear model—are significantly different.
- The corresponding p-value is the estimate of the probability that we would've observed an F-statistic this large or larger if the two variances in question were in reality the same. So you want the p-value to be small (a common threshold: less than 0.05).
- In our example, the F-statistic p-value is quite small ($< 2.2e-16$): the model explains more variance than the constant model does, and the improvement is incredibly unlikely to have arisen only from sampling error.

Linear regression takeaways

Linear regression is the go-to statistical modeling method for predicting quantities. It is simple and has the advantage that the coefficients of the model can often function as advice. Here are a few points you should remember about linear regression:

- Linear regression assumes that the outcome is a linear combination of the input variables. Naturally, it works best when that assumption is nearly true, but it can predict surprisingly well even when it isn't.
- If you want to use the coefficients of your model for advice, you should only trust the coefficients that appear statistically significant.
- Overly large coefficient magnitudes, overly large standard errors on the coefficient estimates, and the wrong sign on a coefficient could be indications of correlated inputs.
- Linear regression can predict well even in the presence of correlated variables, but correlated variables lower the quality of the advice.
- Linear regression will have trouble with problems that have a very large number of variables, or categorical variables with a very large number of levels.
- Linear regression packages have some of the best built-in diagnostics available, but rechecking your model on test data is still your most effective safety check.

Logistic Regression

- Logistic regression is the most important (and probably most used) member of a class of models called *generalized linear models*.
- Unlike linear regression, logistic regression can directly predict values that are restricted to the $(0, 1)$ interval, such as probabilities.
- It's the go-to method for predicting probabilities or rates, and like linear regression, the coefficients of a logistic regression model can be treated as *advice*.
- It's also a good first choice for binary classification problems.
- Logistic regression is usually used to perform classification, but logistic regression and its close cousin *beta regression* are also useful in estimating *rates*.
- In fact, R's standard `glm()` call will work with predicting numeric values between 0 and 1 in addition to predicting classifications.

Understanding logistic regression

- *Suppose you want to predict whether or not a flight will be delayed, based on facts like the flight's origin and destination, weather, and air carrier.*
- *For every flight i , you want to predict `flight_delayed[i]` based on `origin[i]`, `destination[i]`, `weather[i]`, and `air_carrier[i]`.*
- *We'd like to use linear regression to predict the probability that a flight i will be delayed, but probabilities are strictly in the range $0:1$, and linear regression doesn't restrict its prediction to that range.*

- One idea is to find a function of probability that is in the range $-\text{Infinity}:\text{Infinity}$, fit a linear model to predict that quantity, and then solve for the appropriate probabilities from the model predictions.
- So let's look at a slightly different problem: instead of predicting the probability that a flight is delayed, consider the *odds* that the flight is delayed, or the ratio of the probability that the flight is delayed over the probability that it is not.
- $\text{odds}[\text{flight_delayed}] = P[\text{flight_delayed} == \text{TRUE}] / P[\text{flight_delayed} == \text{FALSE}]$
- The range of the odds function isn't $-\text{Infinity}:\text{Infinity}$; it's restricted to be a nonnegative number. But we can take the log of the odds---the *log-odds*---to get a function of the probabilities that *is* in the range $-\text{Infinity}:\text{Infinity}$.
- $\text{log_odds}[\text{flight_delayed}] = \log(P[\text{flight_delayed} == \text{TRUE}] / P[\text{flight_delayed} == \text{FALSE}])$
- Let: $p = P[\text{flight_delayed} == \text{TRUE}]$; then $\text{log_odds}[\text{flight_delayed}] = \log(p / (1 - p))$
- Note that if it's more likely that a flight will be delayed than on time, the odds ratio will be greater than one; if it's less likely that a flight will be delayed than on time, the odds ratio will be less than one.
- So the log-odds is positive if it's more likely that the flight will be delayed, negative if it's more likely that the flight will be on time, and zero if the chances of delay are 50-50.

- The log-odds of a probability p is also known as $\text{logit}(p)$.
- The inverse of $\text{logit}(p)$ is the *sigmoid* function, sigmoid function maps values in the range from $-\infty$ to ∞ to the range $0:1$ —in this case, the sigmoid maps unbounded log-odds ratios to a probability value that is between 0 and 1.
- `logit <- function(p) { log(p/(1-p)) }`
- `s <- function(x) { 1/(1 + exp(-x)) }`
- `s(logit(0.7))`
- `# [1] 0.7`
- `logit(s(-2))`
- `# -2`
- you can think of logistic regression as a linear regression that finds the log-odds of the probability that you're interested in.
- In particular, logistic regression assumes that $\text{logit}(y)$ is linear in the values of x .
- Like linear regression, logistic regression will find the best coefficients to predict y , including finding advantageous combinations and cancellations when the inputs are correlated.

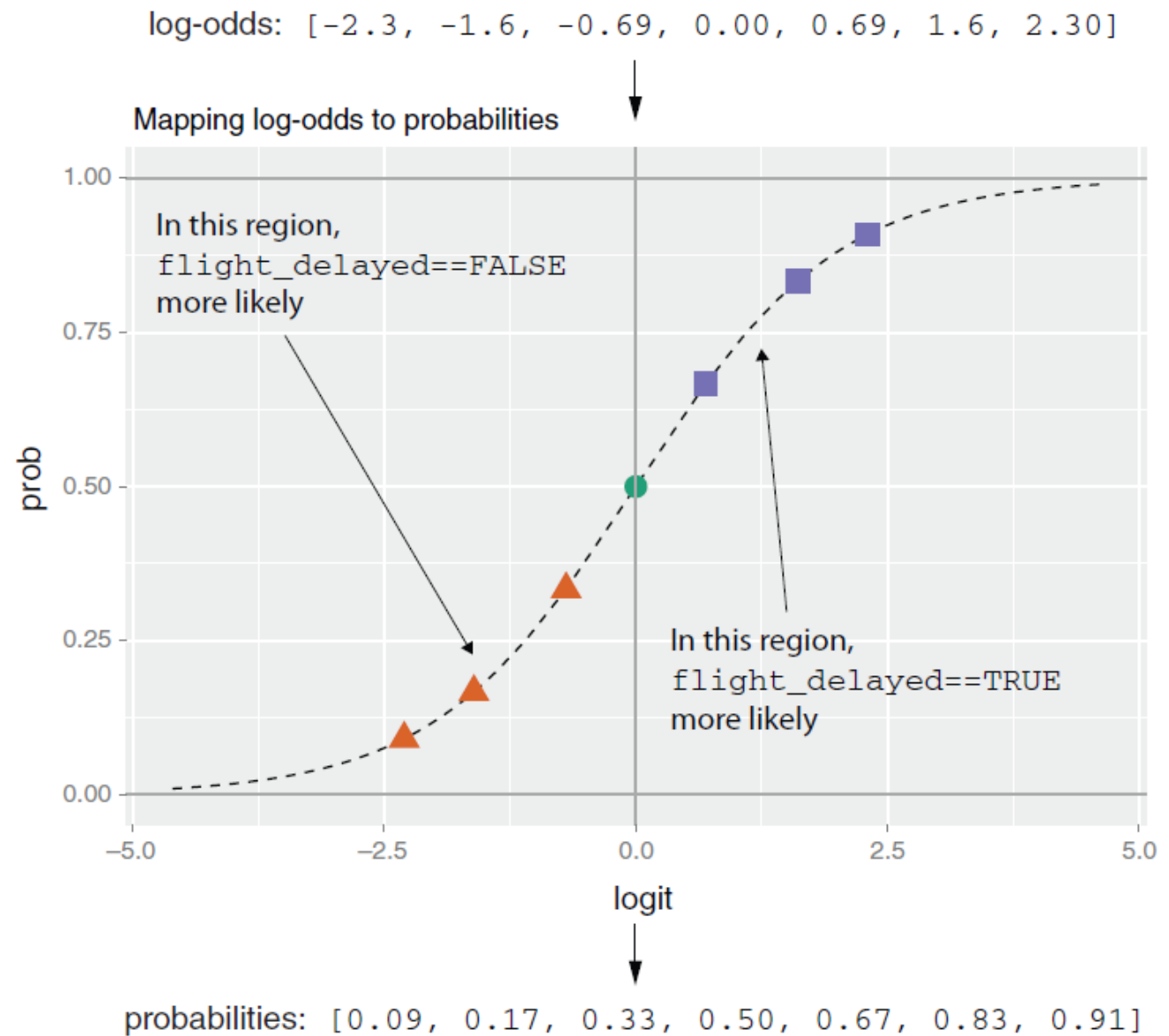


Figure 7.13 Mapping log-odds to the probability of a flight delay via the sigmoid function

- suppose $y[i]$ is the class of object i : TRUE or FALSE; delayed or on_time.
- Also, suppose that $x[i,]$ is a row of inputs, and call one of the classes the “class of interest” or *target class* —that is, the class you are trying to predict (you want to predict whether something is TRUE or whether the flight is in the class delayed).
- Then logistic regression attempts to a fit function $f(x)$ such that
- $P[y[i] \text{ in class of interest}] \sim f(x[i,]) = s(a + b[1] * x[i,1] + \dots + b[n] * x[i,n])$
- **The expression for a logistic regression model**
- If the $y[i]$ are the probabilities that the $x[i,]$ belong to the class of interest, then the task of fitting is to find the $a, b[1], \dots, b[n]$ such that $f(x[i,])$ is the best possible estimate of $y[i]$.
- R supplies a one-line statement to find these coefficients: `glm()`. Note that you don't need to supply $y[i]$ that are probability estimates to run `glm()`; the training method only requires $y[i]$ that say which class a given training example belongs to.

Case Study

- *Imagine that you're working at a hospital.*
- *The overall goal is to design a plan that provisions neonatal emergency equipment to delivery rooms.*
- *Newborn babies are assessed at one and five minutes after birth using what's called the Apgar test, which is designed to determine if a baby needs immediate emergency care or extra medical attention.*
- *A baby who scores below 7 (on a scale from 0 to 10) on the Apgar scale needs extra attention.*
- *Such at-risk babies are rare, so the hospital doesn't want to provision extra emergency equipment for every delivery.*
- *On the other hand, at-risk babies may need attention quickly, so provisioning resources proactively to appropriate deliveries can save lives.*
- *Your task is to build a model to identify ahead of time situations with a higher probability of risk, so that resources can be allocated appropriately.*
- *a sample dataset from the 2010 CDC natality public-use data file (<http://mng.bz/pnGy>). This dataset records statistics for all US births registered in the 50 states and the District of Columbia, including facts about the mother and father, and about the delivery*

Table 7.1 Some variables in the natality dataset

Variable	Type	Description
atRisk	Logical	TRUE if 5-minute Apgar score < 7; FALSE otherwise
PWGT	Numeric	Mother's prepregnancy weight
UPREVIS	Numeric (integer)	Number of prenatal medical visits
CIG_REC	Logical	TRUE if smoker; FALSE otherwise
GESTREC3	Categorical	Two categories: <37 weeks (premature) and >=37 weeks
DPLURAL	Categorical	Birth plurality, three categories: single/twin/triplet+
ULD_MECO	Logical	TRUE if moderate/heavy fecal staining of amniotic fluid
ULD_PRECIP	Logical	TRUE for unusually short labor (< three hours)
ULD_BREECH	Logical	TRUE for breech (pelvis first) birth position
URF_DIAB	Logical	TRUE if mother is diabetic
URF_CHYPER	Logical	TRUE if mother has chronic hypertension
URF_PHYPER	Logical	TRUE if mother has pregnancy-related hypertension
URF_ECLAM	Logical	TRUE if mother experienced eclampsia: pregnancy-related seizures

Building a logistic regression model

- The function to build a logistic regression model in R is `glm()`, supplied by the `stats` package.
- In our case, the dependent variable `y` is the logical (or Boolean) `atRisk`; all the other variables in table 7.1 are the independent variables `x`.
- The formula for building a model to predict `atRisk` using these variables is rather long to type in by hand;
- you can generate the formula using the `mk_formula()` function from the `wrapr` package
- If your goal is to use the model to classify new instances into one of two categories (in this case, at-risk or not-at-risk), then you want the model to give high scores to positive instances and low scores otherwise.

Listing 7.8 Building the model formula

```
complications <- c("ULD_MECO", "ULD_PRECIP", "ULD_BREECH")
riskfactors <- c("URF_DIAB", "URF_CHYPER", "URF_PHYPER",
                "URF_ECLAM")

y <- "atRisk"
x <- c("PWGT",
      "UPREVIS",
      "CIG_REC",
      "GESTREC3",
      "DPLURAL",
      complications,
      riskfactors)|
library(wrapr)
fm1a <- mk_formula(y, x)
```

Now we'll build the logistic regression model, using the training dataset.

Listing 7.9 Fitting the logistic regression model

```
print(fm1a)

## atRisk ~ PWGT + UPREVIS + CIG_REC + GESTREC3 + DPLURAL + ULD_MECO +
##      ULD_PRECIP + ULD_BREECH + URF_DIAB + URF_CHYPER + URF_PHYPER +
##      URF_ECLAM
## <environment: base>

model <- glm(fm1a, data = train, family = binomial(link = "logit"))
```

family = binomial(link = "logit")

- This is similar to the linear regression call to `lm()`, with one additional argument:
- `family = binomial(link = "logit")`.
- The `family` function specifies the assumed distribution of the dependent variable y .
- In our case, we're modeling y as a binomial distribution, or as a coin whose probability of heads depends on x .
- The `link` function "links" the output to a linear model—it's as if you pass y through the `link` function, and then model the resulting value as a linear function of the x values.
- Different combinations of `family` functions and `link` functions lead to different kinds of generalized linear models (for example, Poisson, or probit).
- Without an explicit `family` argument, `glm()` defaults to standard linear regression (like `lm`).
- The `family` argument can be used to select many different behaviors of the `glm()` function.
- For example, choosing `family = quasipoisson` chooses a "log" link, which models the logarithm of the prediction as linear in the inputs.

Making predictions

- `train$pred <- predict(model, newdata=train, type = "response")`
- `test$pred <- predict(model, newdata=test, type="response")`
- Note the additional parameter `type = "response"`.
- This tells the `predict()` function to return the predicted probabilities `y`.
- If you don't specify `type = "response"`, then by default `predict()` will return the output of the link function, `logit(y)`.
- One strength of logistic regression is that it preserves the marginal probabilities of the training data.
- That means that if you sum the predicted probability scores for the entire training set, that quantity will be equal to the number of positive outcomes (`atRisk == TRUE`) in the training set.
- This is also true for subsets of the data determined by variables included in the model. For example, in the subset of the training data that has `train$GESTREC == "<37 weeks"` (the baby was premature), the sum of the predicted probabilities equals the number of positive training examples

Listing 7.11 Preserving marginal probabilities with logistic regression

```
sum(train$atRisk == TRUE)  <— Counts the number of at-risk
## [1] 273                infants in the training set.

sum(train$pred)            <— Sums all the predicted
## [1] 273                probabilities over the
                           training set. Notice that
                           it adds to the number
                           of at-risk infants.

premature <- subset(train, GESTREC3 == "< 37 weeks") <—
sum(premature$atRisk == TRUE)
## [1] 112                Counts the number of
                           at-risk premature
                           infants in the training set

sum(premature$pred)        <— Sums all the predicted probabilities
## [1] 112                for premature infants in the training
                           set. Note that it adds to the number
                           of at-risk premature infants.
```

- Because logistic regression preserves marginal probabilities, you know that the model is in some sense consistent with the training data.
- When the model is applied to future data with distributions similar to the training data, it should then return results consistent with that data: about the correct probability mass of expected at-risk infants, distributed correctly with respect to the infants' characteristics.
- However, if the model is applied to future data with very different distributions (for example, a much higher rate of at-risk infants), the model may not predict as well.

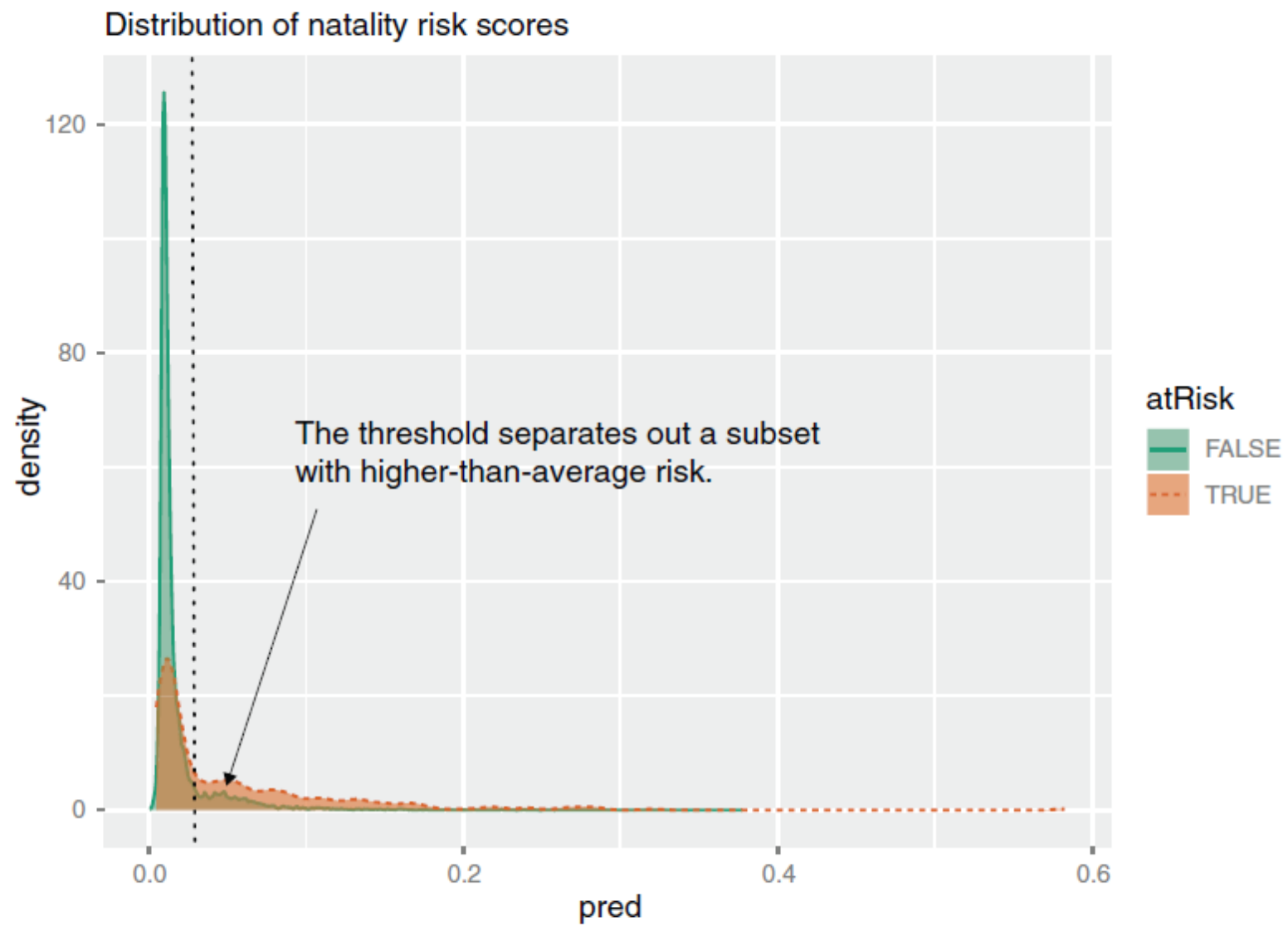


Figure 7.14 Distribution of score broken up by positive examples (TRUE) and negative examples (FALSE)

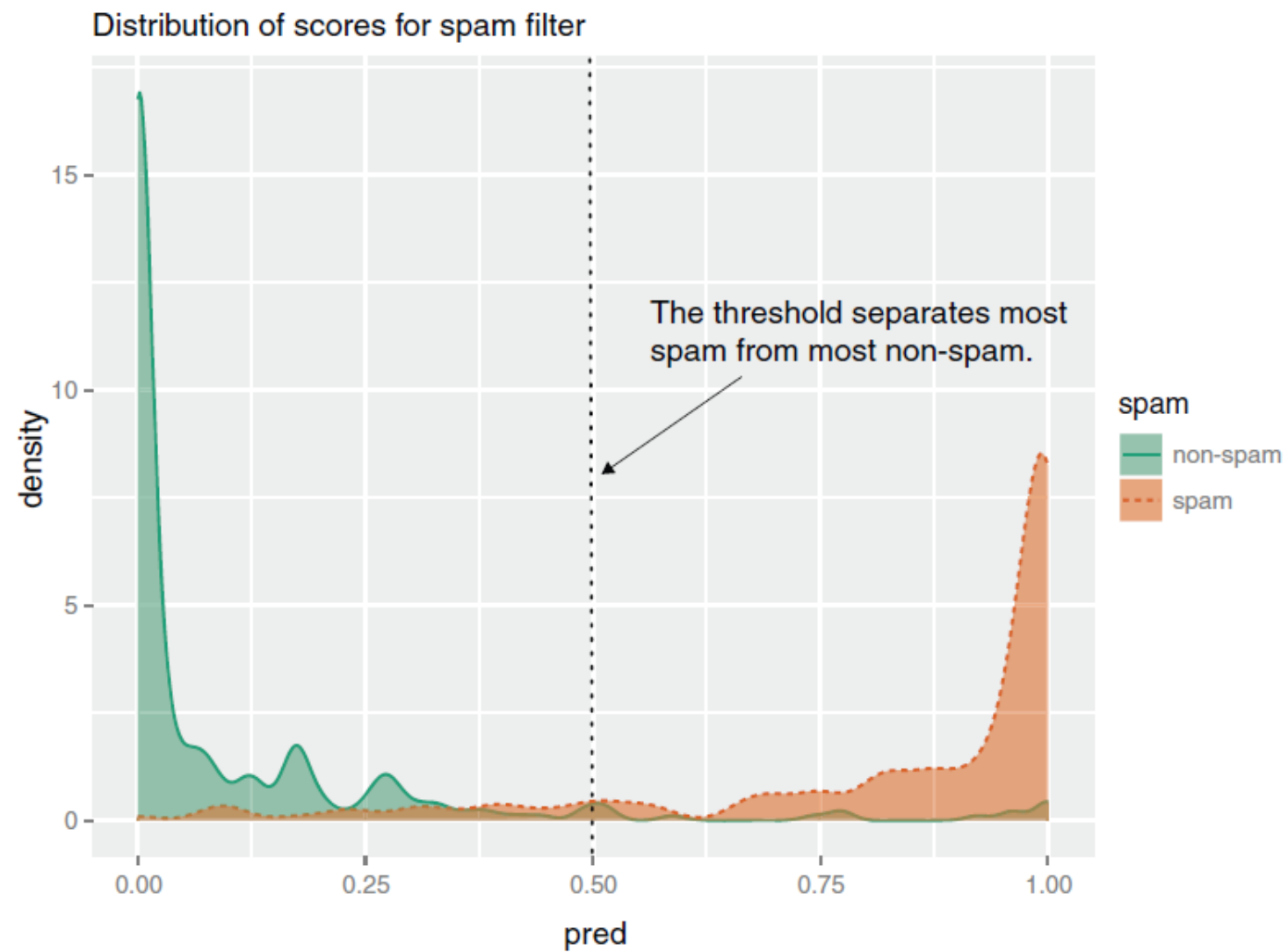
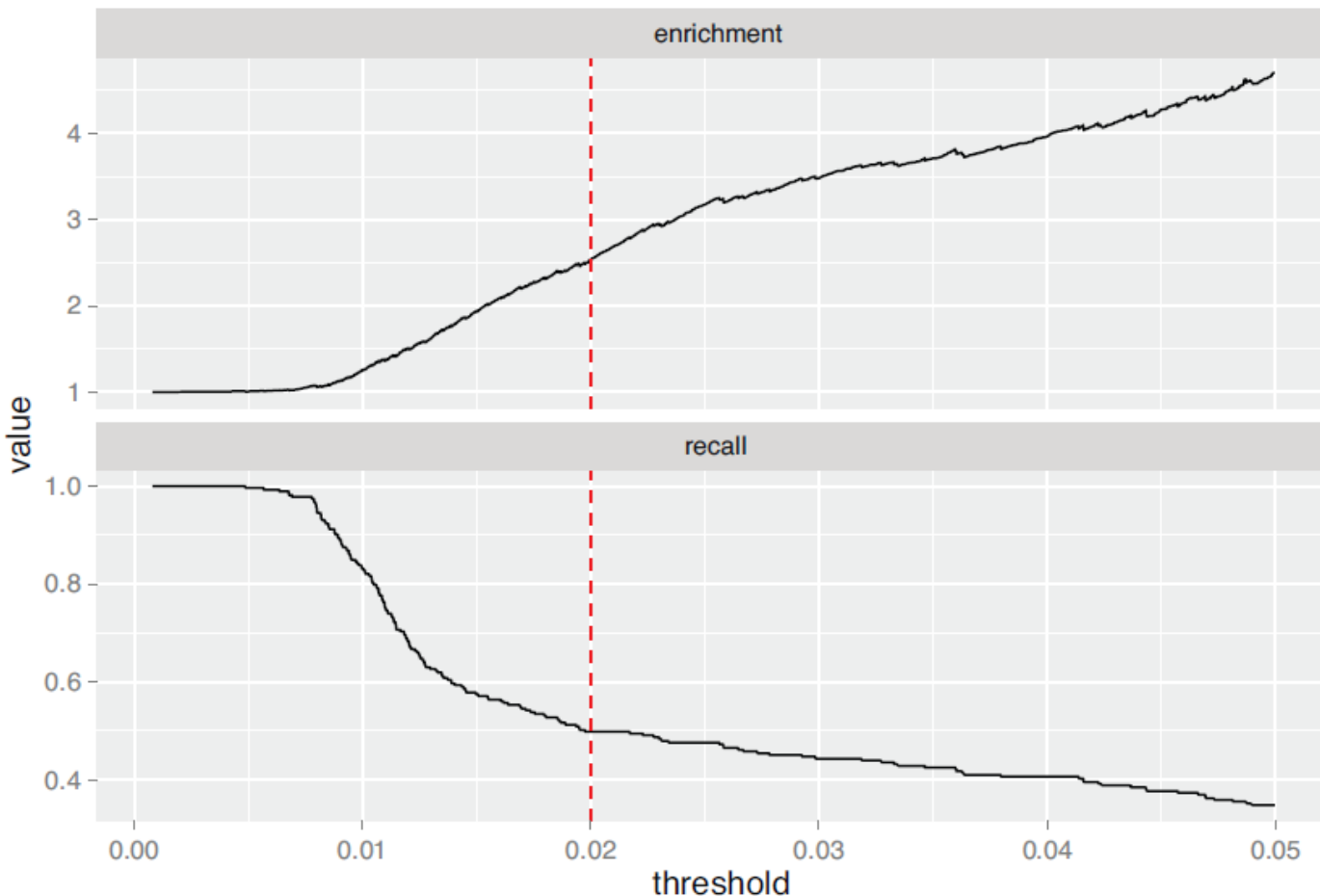


Figure 7.15 Reproduction of the spam filter score distributions from chapter 6

Threshold

- In order to use the model as a classifier, you must pick a threshold; scores above the threshold will be classified as positive, those below as negative.
- When you pick a threshold, you're trying to balance the *precision* of the classifier (what fraction of the predicted positives are true positives) and its *recall* (how many of the true positives the classifier finds).
- If the score distributions of the positive and negative instances are well separated, as in figure 7.15, you can pick an appropriate threshold in the “valley” between the two peaks.
- In the current case, the two distributions aren't well separated, which indicates that the model can't build a classifier that simultaneously achieves good recall and good precision.
- However, you might be able to build a classifier that identifies a subset of situations with a higher-than-average rate of at-risk births: for example, you may be able to find a threshold that produces a classifier with a precision of 3.6%.
- Even though this precision is low, it represents a subset of the data that has twice the risk as the overall population (3.6% versus 1.8%), so preprovisioning resources to those situations may be advised.
- We'll call the ratio of the classifier precision to the average rate of positives the *enrichment rate*.
- The higher you set the threshold, the more precise the classifier will be (you'll identify a set of situations with a much higher-than-average rate of at-risk births); but you'll also miss a higher percentage of at-risk situations, as well.
- When picking the threshold, you should use the training set, since picking the threshold is part of classifier-building.
- You can then use the test set to evaluate classifier performance.

Enrichment/recall vs. threshold for natality model



- The best trade-off between precision/enrichment and recall is a function of how many resources the hospital has available to allocate, and how many they can keep in reserve (or redeploy) for situations that the classifier missed.
- A threshold of 0.02 might be a good tradeoff.
- The resulting classifier will identify a subset of the population where the rate of risky births is 2.5 times higher than in the overall population, and which contains about half of all the true at-risk situations.

Figure 7.16 Enrichment (top) and recall (bottom) plotted as functions of threshold for the training set

Listing 7.13 Exploring modeling trade-offs

```
library("WVPlots")
library("ggplot2")
plt <- PRTPlot(train, "pred", "atRisk", TRUE,
               plotvars = c("enrichment", "recall"),
               thresholdrange = c(0,0.05),
               title = "Enrichment/recall vs. threshold for natality model")
plt + geom_vline(xintercept = 0.02, color="red", linetype = 2)
```

Calls PRTPlot() where pred is the column of predictions, atRisk is the true outcome column, and TRUE is the class of interest

Adds a line to mark threshold = 0.02.

Once you've picked an appropriate threshold, you can evaluate the resulting classifier by looking at the confusion matrix, as we discussed in section 6.2.3. Let's use the test set to evaluate the classifier with a threshold of 0.02.

Listing 7.14 Evaluating the chosen model

```
( ctab.test <- table(pred = test$pred > 0.02, atRisk = test$atRisk) )  
  
##           atRisk  
## pred    FALSE TRUE  
##  FALSE   9487   93  
##   TRUE   2405  116
```

Builds the confusion matrix. The rows contain predicted negatives and positives; columns contain actual negatives and positives.

```
( precision <- ctab.test[2,2] / sum(ctab.test[2,]) )  
## [1] 0.04601349  
  
( recall <- ctab.test[2,2] / sum(ctab.test[,2]) )  
## [1] 0.5550239  
  
( enrichment <- precision / mean(as.numeric(test$atRisk)) )  
## [1] 2.664159
```

The resulting classifier is low-precision, but identifies a set of potential at-risk cases that contains 55.5% of the true positive cases in the test set, at a rate 2.66 times higher than the overall average. This is consistent with the results on the training set.

Finding relations and extracting advice from logistic models

- The coefficients of a logistic regression model encode the relationships between the input variables and the output in a way similar to how the coefficients of a linear regression model do. You can get the model's coefficients with the call `coefficients (model)`.
- Negative coefficients that are statistically significant correspond to variables that are negatively correlated to the odds (and hence to the probability) of a positive outcome (the baby being at risk).
- Positive coefficients that are statistically significant are positively correlated to the odds of the baby being at risk.
- As with linear regression, every categorical variable is expanded to a set of indicator variables. If the original variable has n levels, there will be $n-1$ indicator variables; the remaining level is the reference level.
- For example, the variable `DPLURAL` has three levels corresponding to single births, twins, and triplets or higher.
- The logistic regression model has two corresponding coefficients: `DPLURALtwin` and `DPLURALtriplet or higher`. The reference level is single births.
- Both of the `DPLURAL` coefficients are positive, indicating that multiple births have higher odds of being at risk than single births do, all other variables being equal.

Listing 7.15 The model coefficients

```
coefficients(model)
##              (Intercept)              PWGT
##              -4.41218940              0.00376166
##              UPREVIS              CIG_RECTRUE
##              -0.06328943              0.31316930
##              GESTREC3< 37 weeks DPLURALtriplet or higher
##              1.54518311              1.39419294
##              DPLURALtwin              ULD_MECOTRUE
##              0.31231871              0.81842627
##              ULD_PRECIPTRUE              ULD_BREECHTRUE
##              0.19172008              0.74923672
##              URF_DIABTRUE              URF_CHYPERTRUE
##              -0.34646672              0.56002503
##              URF_PHYPERTRUE              URF_ECLAMTRUE
##              0.16159872              0.49806435
```

You can get the model's coefficients with the call `coefficients (model)`.

INTERPRETING THE COEFFICIENTS

- If the coefficient for the variable $x[, k]$ is $b[k]$, then the odds of a positive outcome are multiplied by a factor of $\exp(b[k])$ for every unit change in $x[, k]$.
- The coefficient for `GESTREC3 < 37 weeks` (for a premature baby) is 1.545183.
- So for a premature baby, the odds of being at risk are $\exp(1.545183) = 4.68883$ times higher compared to a baby that's born full-term, with all other input variables unchanged.
- The risk odds for a premature baby with the same characteristics as our hypothetical full-term baby are $0.0101 * 4.68883 = 0.047$.
- You can invert the formula $\text{odds} = p / (1 - p)$ to solve for p as a function of odds:
- $p = \text{odds} * (1 - p) = \text{odds} - p * \text{odds}$
- $p * (1 + \text{odds}) = \text{odds}$
- $p = \text{odds} / (1 + \text{odds})$
- The probability of this premature baby being at risk is $0.047 / 1.047$, or about 4.5%—quite a bit higher than the equivalent full-term baby
- Similarly, the coefficient for `UPREVIS` (number of prenatal medical visits) is about -0.06. This means every prenatal visit lowers the odds of an at-risk baby by a factor of $\exp(-0.06)$, or about 0.94.
- Suppose the mother of a premature baby had made no prenatal visits; a baby in the same situation whose mother had made three prenatal visits would have odds of being at risk of about $0.047 * 0.94 * 0.94 * 0.94 = 0.039$.
- This corresponds to a probability of being at risk of 3.75%.
- The general advice in this case might be to keep a special eye on premature births (and multiple births), and encourage expectant mothers to make regular prenatal visits.

Reading the model summary and characterizing coefficients

Listing 7.16 The model summary

```
summary(model)

## Call:
## glm(formula = fmla, family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -0.9732  -0.1818  -0.1511  -0.1358   3.2641
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.412189   0.289352 -15.249  < 2e-16 ***
## PWGT         0.003762   0.001487   2.530  0.011417 *
## UPREVIS      -0.063289   0.015252  -4.150  3.33e-05 ***
## CIG_RECTRUE   0.313169   0.187230   1.673  0.094398 .
## GESTREC3< 37 weeks 1.545183   0.140795  10.975  < 2e-16 ***
## DPLURALtriplet or higher 1.394193   0.498866   2.795  0.005194 **
## DPLURALtwin    0.312319   0.241088   1.295  0.195163
## ULD_MECOTRUE   0.818426   0.235798   3.471  0.000519 ***
## ULD_PRECIPTRUE 0.191720   0.357680   0.536  0.591951
## ULD_BREECHTRUE 0.749237   0.178129   4.206  2.60e-05 ***
## URF_DIABTRUE  -0.346467   0.287514  -1.205  0.228187
## URF_CHYPERTRUE 0.560025   0.389678   1.437  0.150676
## URF_PHYPERTRUE 0.161599   0.250003   0.646  0.518029
## URF_ECLAMTRUE  0.498064   0.776948   0.641  0.521489
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##    Null deviance: 2698.7  on 14211  degrees of freedom
## Residual deviance: 2463.0  on 14198  degrees of freedom
## AIC: 2491
##
## Number of Fisher Scoring iterations: 7
```

THE ORIGINAL MODEL CALL

- The first line of the summary is the call to `glm()`:
- Call:
- `glm(formula = fmla, family = binomial(link = "logit"), data = train)`
- Here is where we check that we've used the correct training set and the correct formula
- We can also verify that we used the correct family and link function to produce a logistic model.

THE DEVIANCE RESIDUALS SUMMARY

- The deviance residuals are the analog to the residuals of a linear regression model:
- Deviance Residuals:
- | Min | 1Q | Median | 3Q | Max |
|---------|---------|---------|---------|--------|
| -0.9732 | -0.1818 | -0.1511 | -0.1358 | 3.2641 |
- Linear regression models are found by minimizing the sum of the squared residuals;
- logistic regression models are found by minimizing the sum of the residual deviances, which is equivalent to maximizing the log likelihood of the data, given the model

- Logistic models can also be used to explicitly compute rates: given several groups of identical data points (identical except the outcome), predict the rate of positive outcomes in each group. This kind of data is called *grouped data*.
- In the case of grouped data, the deviance residuals can be used as a diagnostic for model fit.
- This is why the deviance residuals are included in the summary.
- We're using *ungrouped data*—every data point in the training set is potentially unique.
- In the case of ungrouped data, the model fit diagnostics that use the deviance residuals are no longer valid, so we won't discuss them here

THE SUMMARY COEFFICIENTS TABLE

The summary coefficients table for logistic regression has the same format as the coefficients table for linear regression:

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	-4.412189	0.289352	-15.249	< 2e-16	***
PWGT	0.003762	0.001487	2.530	0.011417	*
UPREVIS	-0.063289	0.015252	-4.150	3.33e-05	***
CIG_RECTRUE	0.313169	0.187230	1.673	0.094398	.
GESTREC3< 37 weeks	1.545183	0.140795	10.975	< 2e-16	***
DPLURALtriplet or higher	1.394193	0.498866	2.795	0.005194	**
DPLURALtwin	0.312319	0.241088	1.295	0.195163	
ULD_MECOTRUE	0.818426	0.235798	3.471	0.000519	***
ULD_PRECIPTRUE	0.191720	0.357680	0.536	0.591951	
ULD_BREECHTRUE	0.749237	0.178129	4.206	2.60e-05	***
URF_DIABTRUE	-0.346467	0.287514	-1.205	0.228187	
URF_CHYPERTRUE	0.560025	0.389678	1.437	0.150676	
URF_PHYPERTRUE	0.161599	0.250003	0.646	0.518029	
URF_ECLAMTRUE	0.498064	0.776948	0.641	0.521489	

Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'	0.1 ' ' 1

The columns of the table represent

- A coefficient
- Its estimated value
- The error around that estimate
- The signed distance of the estimated coefficient value from 0 (using the standard error as the unit of distance)
- The probability of seeing a coefficient value at least as large as we observed, under the null hypothesis that the coefficient value is really zero

- the *p-value* or *significance*, tells us whether we should trust the estimated coefficient value.
- The common practice is to assume that coefficients with p-values less than 0.05 are reliable, although some researchers prefer stricter thresholds.

- The coefficient magnitudes are non-negligible and the p-values indicate significance.
- Other variables that affect the outcome are
- *PWGT*—The mother's prepregnancy weight (heavier mothers indicate higher risk—slightly surprising)
- *UPREVIS*—The number of prenatal medical visits (the more visits, the lower the risk)
- *ULD_MECOTRUE*—Meconium staining in the amniotic fluid
- *ULD_BREECHTRUE*—Breech position at birth
- There might be a positive correlation between a mother's smoking and an at-risk birth, but the data doesn't indicate it definitively. None of the other variables show a strong relationship to an at-risk birth.

Null and residual deviances

- Deviance is a measure of how well the model fits the data.
- It is two times the negative *log likelihood* of the dataset, given the model.
- The idea behind log likelihood is that positive instances y should have high probability p_y of occurring under the model;
- negative instances should have low probability of occurring (or putting it another way, $(1 - p_y)$ should be large).
- The log likelihood function rewards matches between the outcome y and the predicted probability p_y , and penalizes mismatches (high p_y for negative instances, and vice versa).
- If you think of deviance as analogous to variance, then the *null deviance* is similar to the variance of the data around the average rate of positive examples.
- The *residual deviance* is similar to the variance of the data around the model.
- As with variance, you want the residual deviance to be small, compared to the null deviance.
- The model summary reports the deviance and null deviance of the model on the training data;

Logistic regression takeaways

Logistic regression is the go-to statistical modeling method for binary classification. As with linear regression, the coefficients of a logistic regression model can often function as advice. Here are some points to remember about logistic regression:

- Logistic regression is well calibrated: it reproduces the marginal probabilities of the data.
- Pseudo R-squared is a useful goodness-of-fit heuristic.
- Logistic regression will have trouble with problems with a very large number of variables, or categorical variables with a very large number of levels.
- Logistic regression can predict well even in the presence of correlated variables, but correlated variables lower the quality of the advice.
- Overly large coefficient magnitudes, overly large standard errors on the coefficient estimates, and the wrong sign on a coefficient could be indications of correlated inputs.
- Too many Fisher iterations, or overly large coefficients with very large standard errors, could be signs that your logistic regression model has not converged, and may not be valid.
- `glm()` provides good diagnostics, but rechecking your model on test data is still your most effective diagnostic.