Teaching Introductory Statistics with R/RStudio

Regression (Simple, Multiple, Logistic)

Simple Linear Regression

Linear Regression statistical method for fitting a line to data. The relationship between two variables, x and y, can be modeled by a straight line with some error:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

β_0 is the y-intercept parameter, β_1 is the slope parameter, and ε represents the random error "noise" that expresses the data points landing somewhere off the straight line $y = \beta_0 + \beta_1 x$

Simple Linear Regression

Brushtail possums are a marsupial that lives in Australia. Researchers captured 104 of these animals and took body measurements before releasing them. We consider two measurements:
total length of possum (head to tail) and **length of possum's head**.

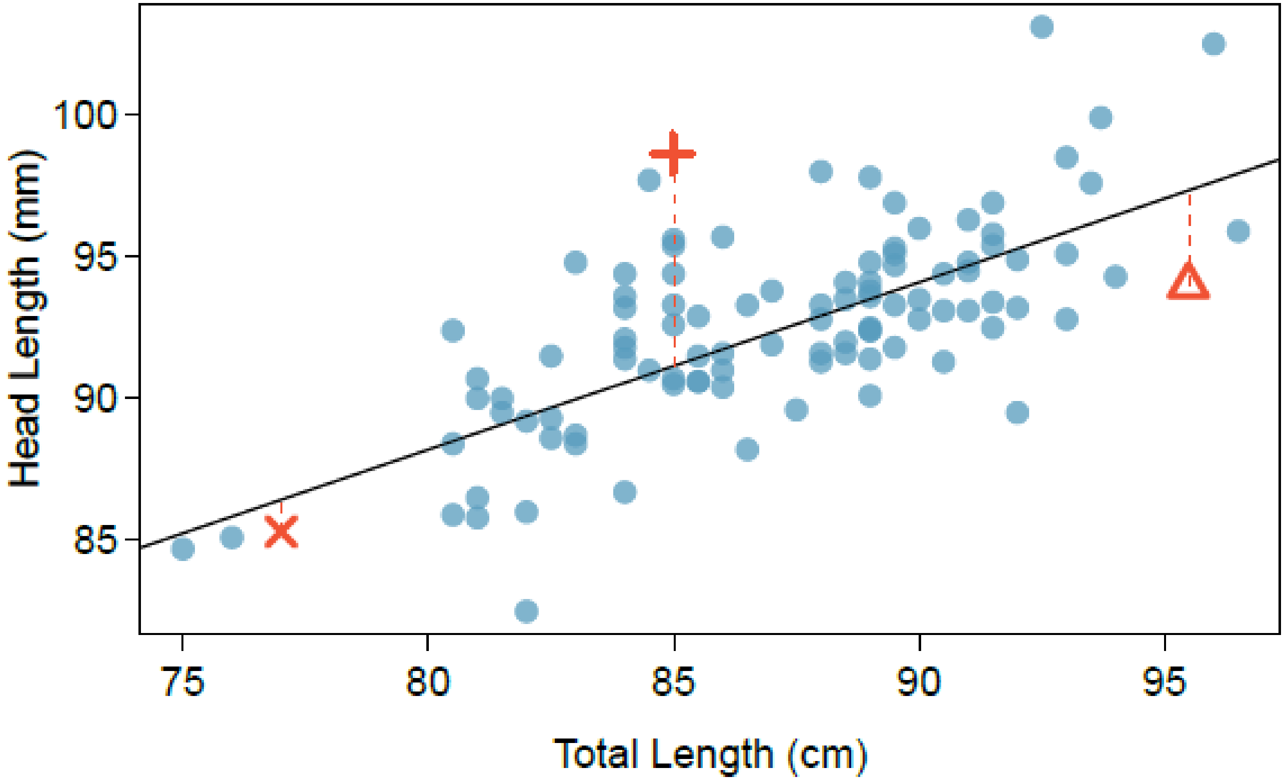
Data frame **possum** (openintro package)

To see the data: **view(possum)**

Let's get the scatterplot and correlation.

```
gf_point(head_l ~ total_l,
       data = possum, color = "blue")

cor(head_l ~ total_l, data = possum,
       use = "complete")
```



The function that creates the linear model is called **`lm()`**

`lm()` creates a lot of information about the model, so we save the results in a variable.

```
> model <-lm(head_l ~ total_l, data = possum)
```

Type **`model`** to see the main results:

```
> model

Call:
lm(formula = head_l ~ total_l, data = possum)

Coefficients:
(Intercept)      total_l
    42.7098       0.5729
```

Y-Intercept
$\hat{\beta}_0$

Slope
$\hat{\beta}_1$

Simple Linear Regression Model Line of Best Fit

$$\hat{y} = 42.7098 + 0.5729x$$

Use the model to predict head length of a possum which has a total length of $x = 80$ cm:

$$\hat{y} = 42.7098 + 0.5729(80) = 88.5418 \text{ mm}$$

Simple Linear Regression

To see more information about the model, use the **`summary()`** function:

```
> summary(model)

Call:
lm(formula = head_l ~ total_l, data = possum)

Residuals:
    Min      1Q  Median      3Q     Max
-7.1877 -1.5340 -0.3345  1.2788  7.3968

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 42.70979    5.17281   8.257 5.66e-13 ***
total_l      0.57290    0.05933   9.657 4.68e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.595 on 102 degrees of freedom
Multiple R-squared:  0.4776,     Adjusted R-squared:  0.4725
F-statistic: 93.26 on 1 and 102 DF,  p-value: 4.681e-16
```

5 NUMBER SUMMARY OF RESIDUALS

Describes Strength of Fit

Fitted Values with makeFun()

`makeFun()` in `MosaicCore` package. Makes a function of the regression model.

Let's name our function `fit`

```
> fit <-makeFun(model)
```

`Find expected head length of possum with total length 80mm:`

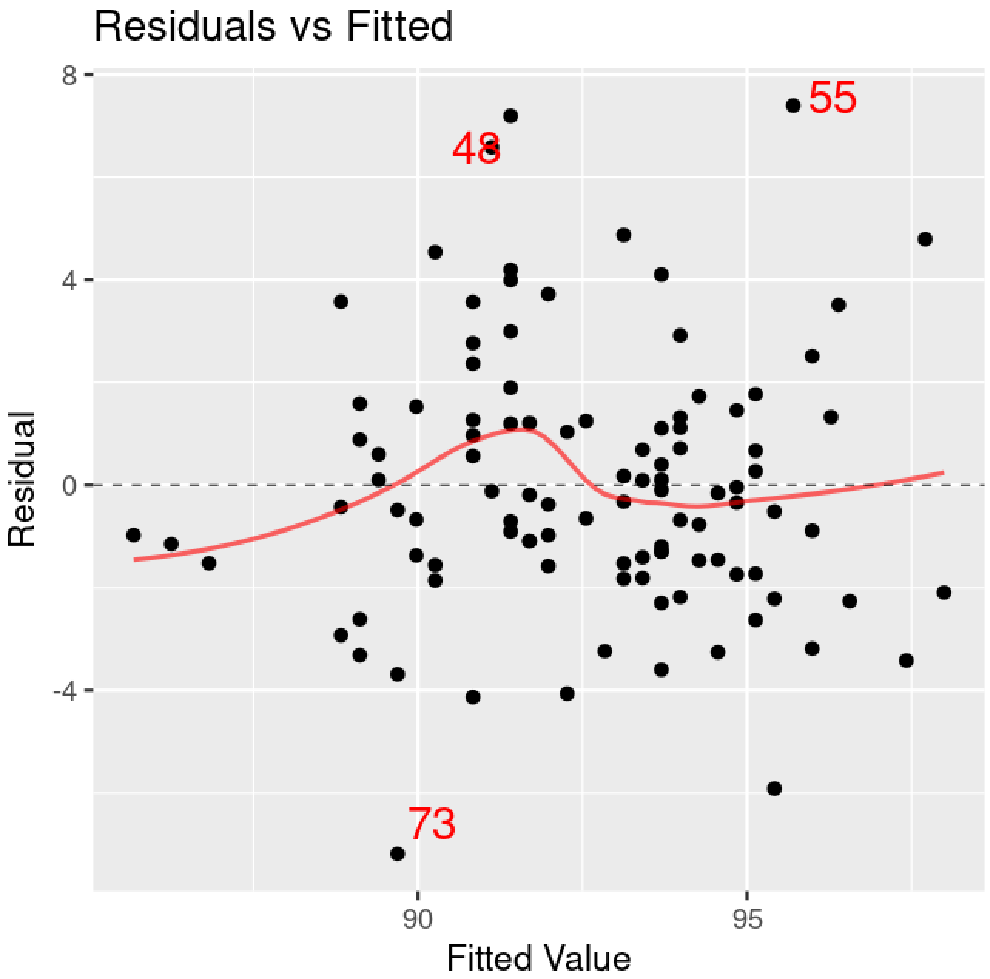
```
> fit(80)
  88.5419
```

Same as:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x \quad \rightarrow \quad \hat{y} = 42.7097931 + (0.5729013)(80) = 88.5419$$

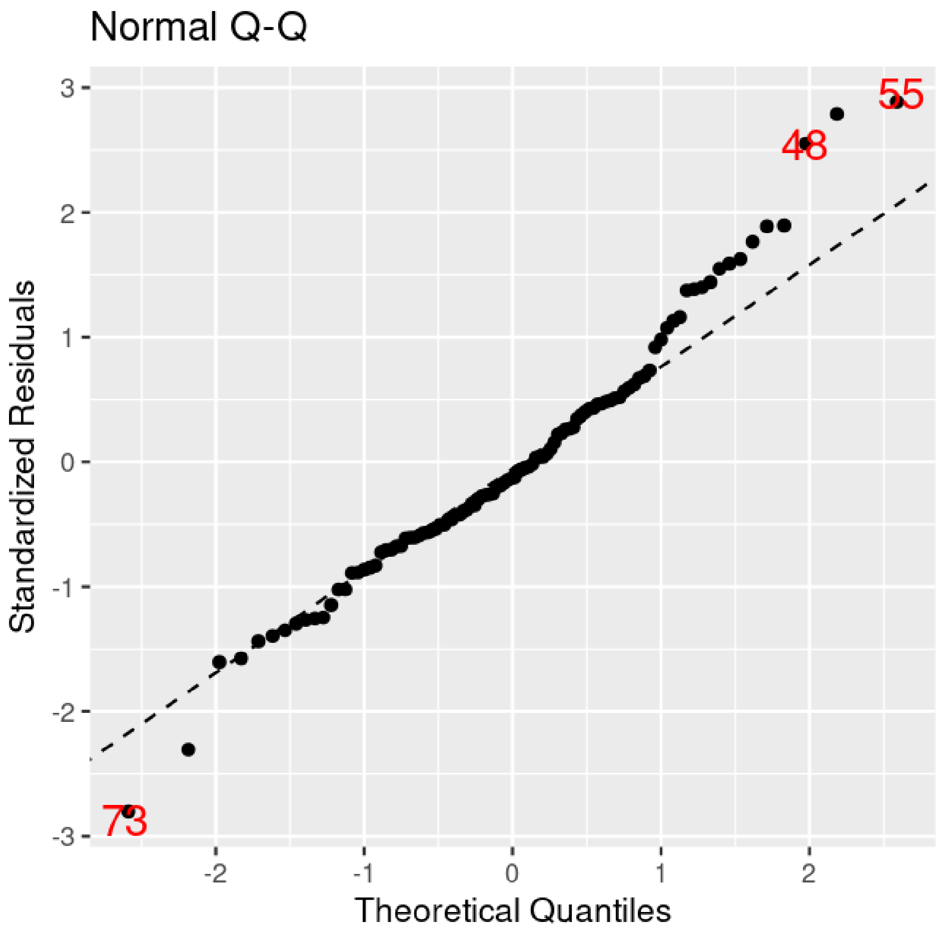
Diagnostic Plots

```
> mosaic::mplot(model, which = 1)
```



Look for Equal Spread to rule out non-linear

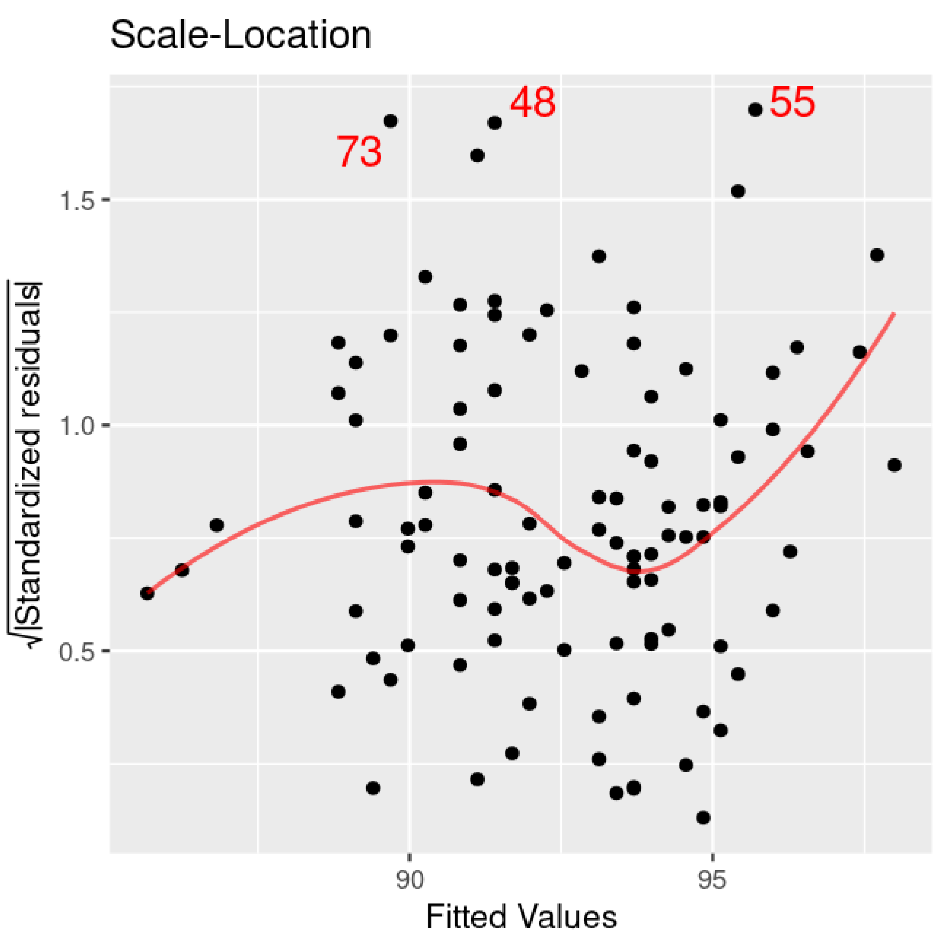
```
> mosaic::mplot(model, which = 2)
```



Assess normality of residuals (can also look at histogram)

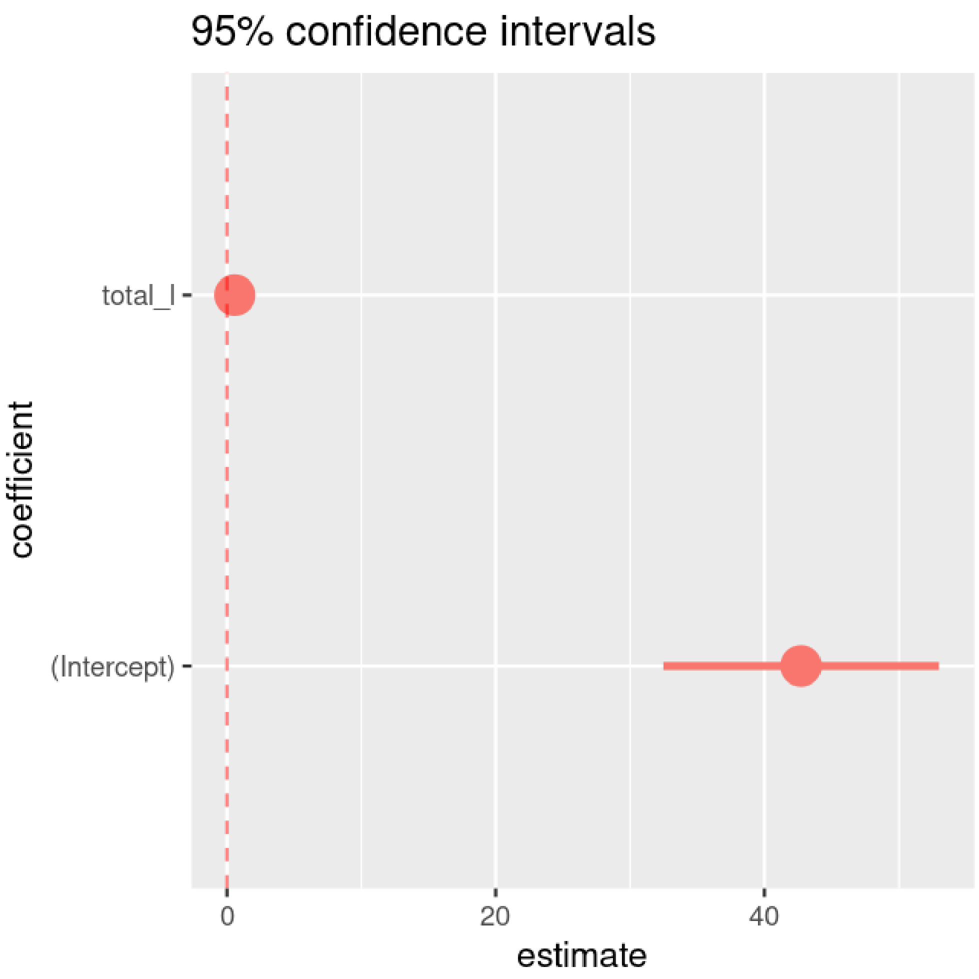
Diagnostic Plots

```
> mosaic::mplot(model, which = 3)
```



Homoscedasticity (Homogeneity of Variance)

```
> mosaic::mplot(model, which = 2)
```

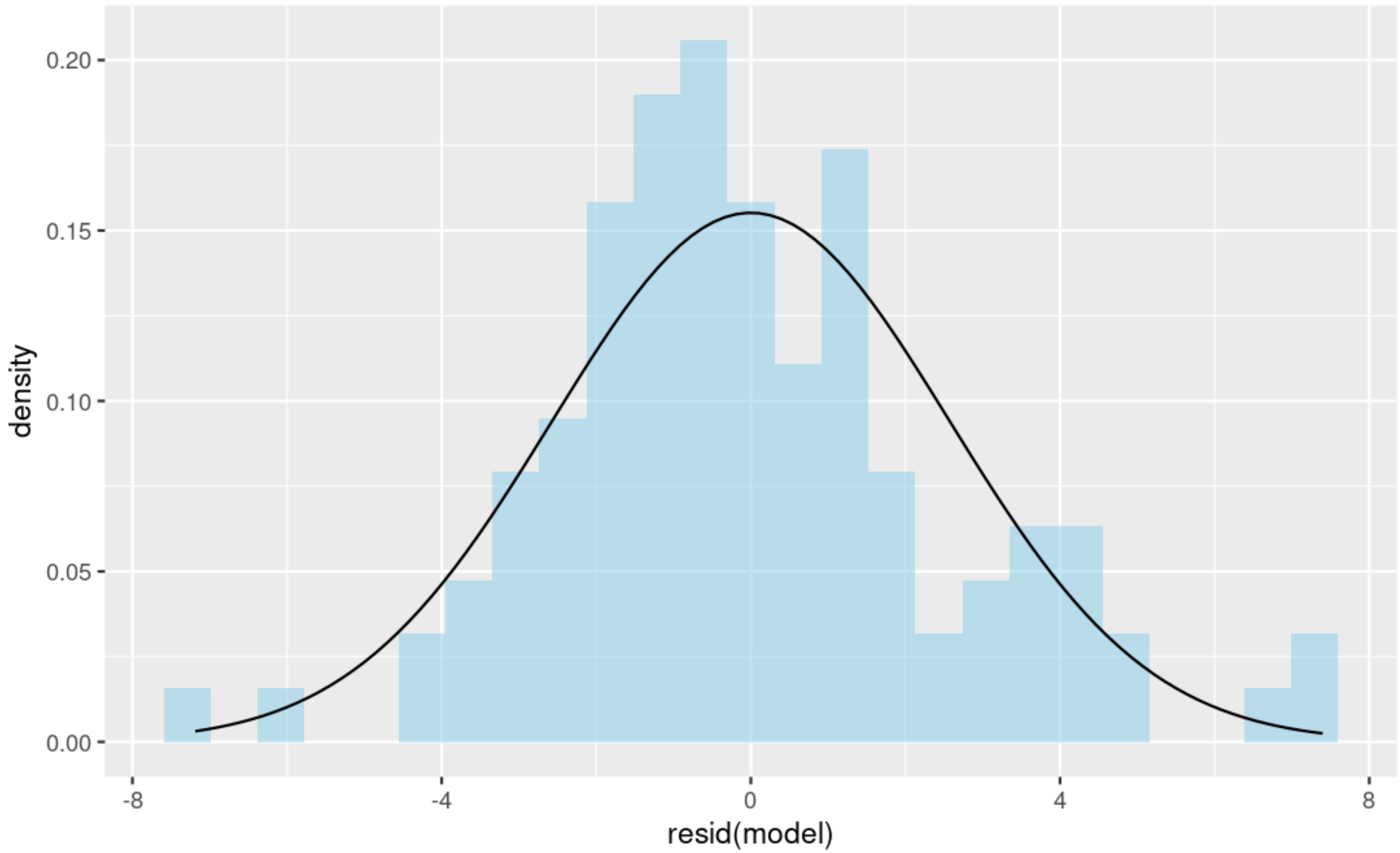


Visual of C.I. for each parameter estimate

Diagnostic Plots

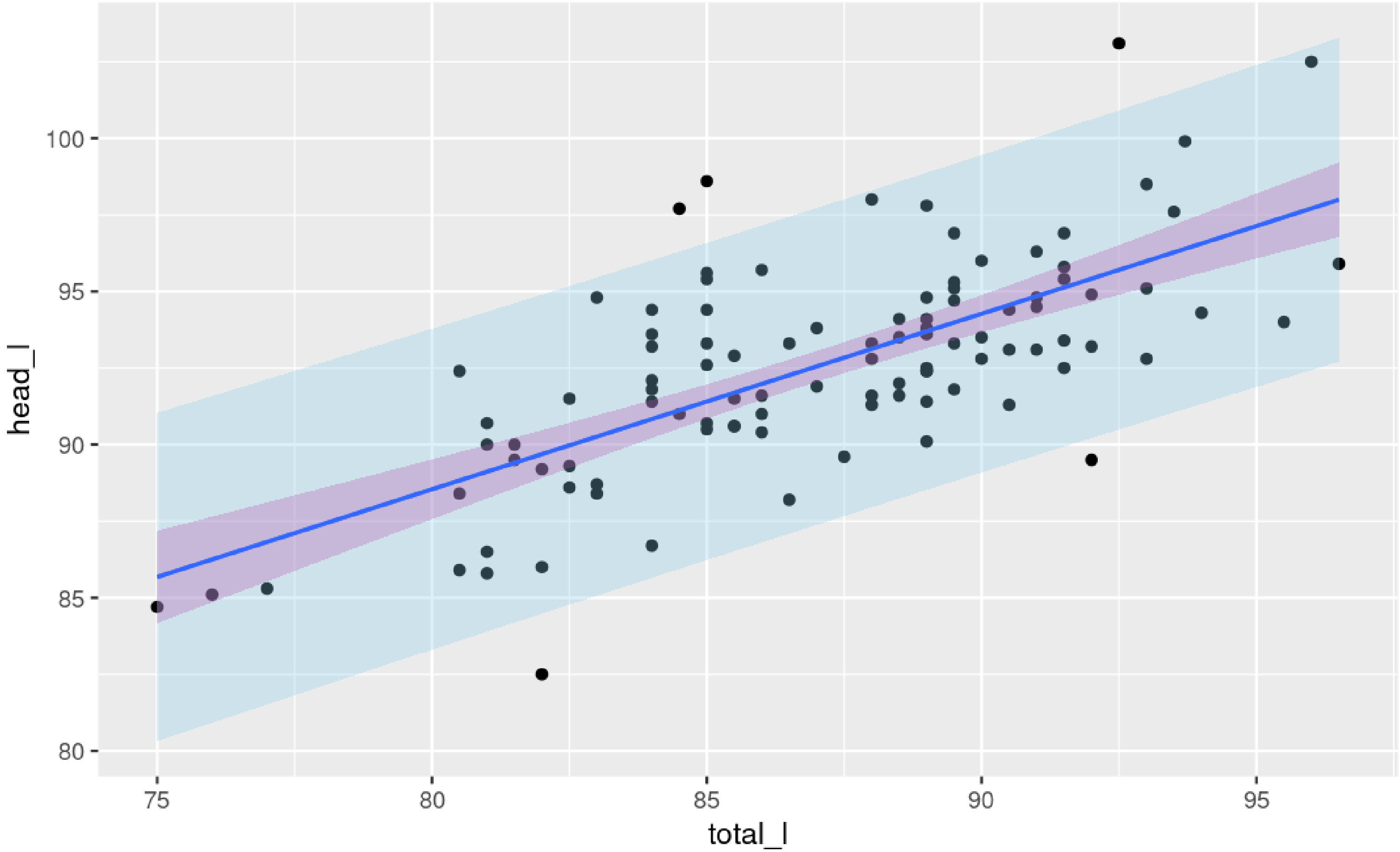
As with QQ-Plot, we can assess normality of residuals by looking at their distribution.

```
gf_dhistogram(~ resid(model), fill = "skyblue") %>%
     gf_fitdistr(dist = "dnorm")
```



Confidence and Prediction Bands

```
> gf_point(head_l ~ total_l, data = possum)%>%
      gf_lm(interval = "confidence", fill = "violetred") %>%
      gf_lm(interval = "prediction", fill = "skyblue")
```



Multiple Linear Regression

Build a model using all variables.

```
> mr_model <-lm(head_l ~ . , data = possum)


> msummary(mr_model)
               Estimate Std. Error t value       Pr(>|t|)
(Intercept) 35.18356    5.31147   6.624 0.00000000209 ***
total_l      0.51344    0.08143   6.305 0.00000000903 ***
tail_l      -0.40847    0.17455  -2.340        0.0214 *
skull_w      0.45850    0.08202   5.590 0.00000021645 ***
age          0.17028    0.11432   1.489        0.1397
sexm         1.05945    0.43733   2.423        0.0173 *
popother     0.82367    0.58568   1.406        0.1629

Residual standard error: 2.07 on 95 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.6761,    Adjusted R-squared:  0.6557
F-statistic: 33.05 on 6 and 95 DF,  p-value: < 0.0000000000000022
```

Model Selection

Model Selection using Backward Selection and AIC (Akaike Information Criteria).

Predictors are eliminated in steps according to which reduces AIC the most.

```
> step(model, method = “backward”)
```

```
Start:  AIC=155.16
head_l ~ total_l + tail_l + skull_w + age + sex + pop

          Df Sum of Sq    RSS    AIC
<none>                 407.03 155.16
- pop      1     8.474 415.51 155.26
- age      1     9.505 416.54 155.51
- tail_l   1    23.462 430.50 158.88
- sex      1    25.145 432.18 159.27
- skull_w  1   133.904 540.94 182.17
- total_l  1   170.328 577.36 188.82

Call:
lm(formula = head_l ~ total_l + tail_l + skull_w + age + sex + 
    pop, data = possum)

Coefficients:
(Intercept)      total_l       tail_l      skull_w          age         sexm     popother  
    35.1836       0.5134      -0.4085       0.4585       0.1703       1.0595       0.8237  
```

<u>CONCLUSION:</u>

NO VARIABLES ELIMINATED

KEEP ALL IN THE MODEL!

Logistic Regression - Classification

Logistic regression model (logit model) is used when response variable *Y* is categorical.

When *Y* is binary (only two possible values), we seek to classify outcomes such as:
0/1, yes/no, disease/no disease, survive/not survive, success/failure, etc.

Rather than model Y directly as with ordinary regression, we model the probability p(X) that Y belongs to a certain category based on one or more predictor variables X.

Does the presence of a predictor affect the probability of a given outcome?

When Y has more than two categories, multinomial logistic regression can be used.

Logistic Regression - Classification

For a single predictor variable, we model the probability of Y using the *logistic function*:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

Odds is defined as: $\frac{p(X)}{1 - p(X)}$ A little fun algebra shows that: $\frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X}$

Taking log of both sides, we arriver at the **log-odds** or **logit**, which is linear in X

$$log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

The beta parameters are estimated using the method of **maximum likelihood estimation**.

Logistic Regression in R

For our example, we will model the probability of credit card default.

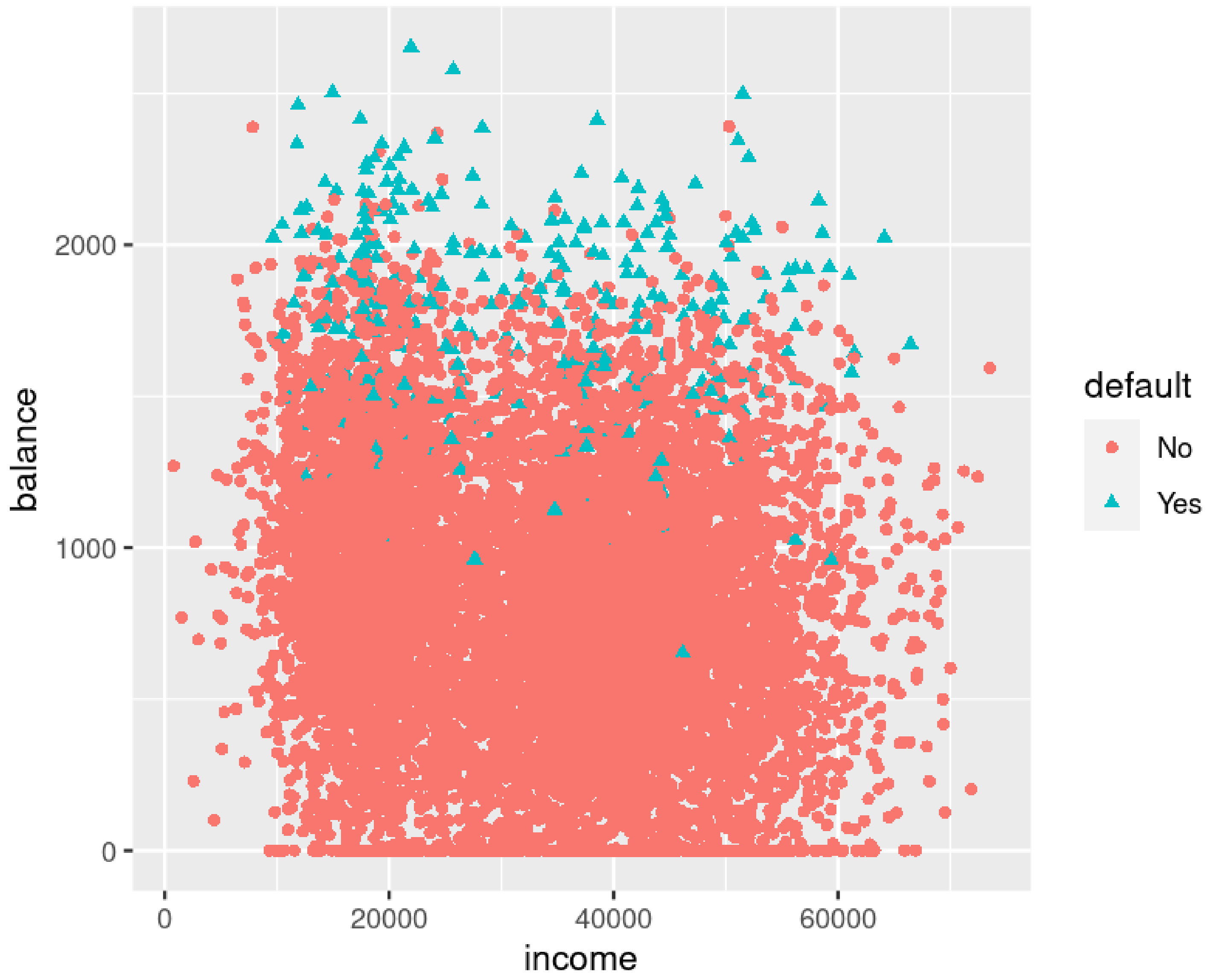
The categorical response variable is called **default** (“yes” or “no”).
There are numerical predictor variables **income** and **balance**.

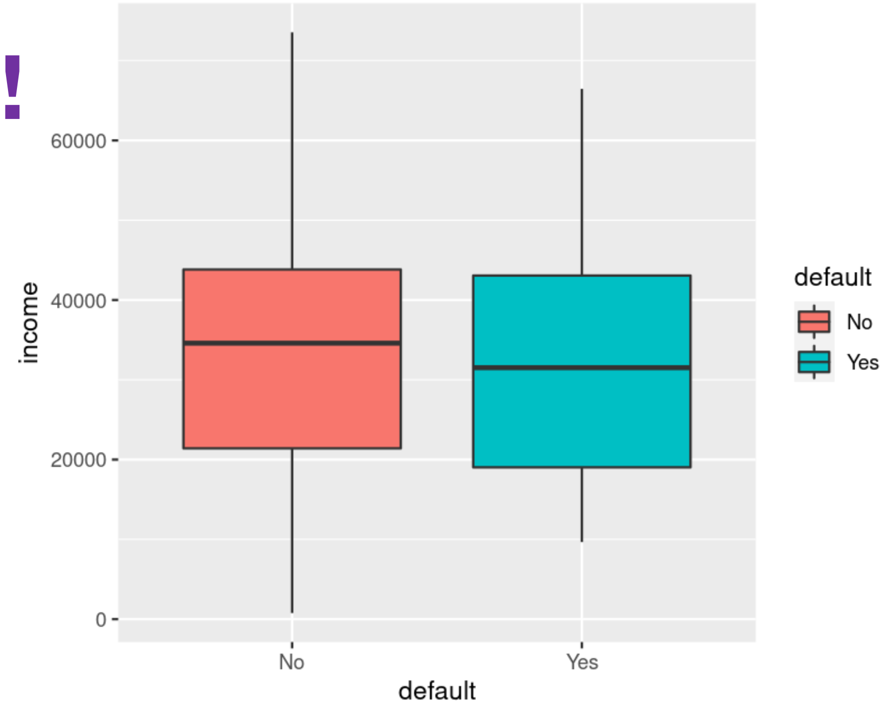
The `Default` data frame is in a package called ISLR, which we need to install.

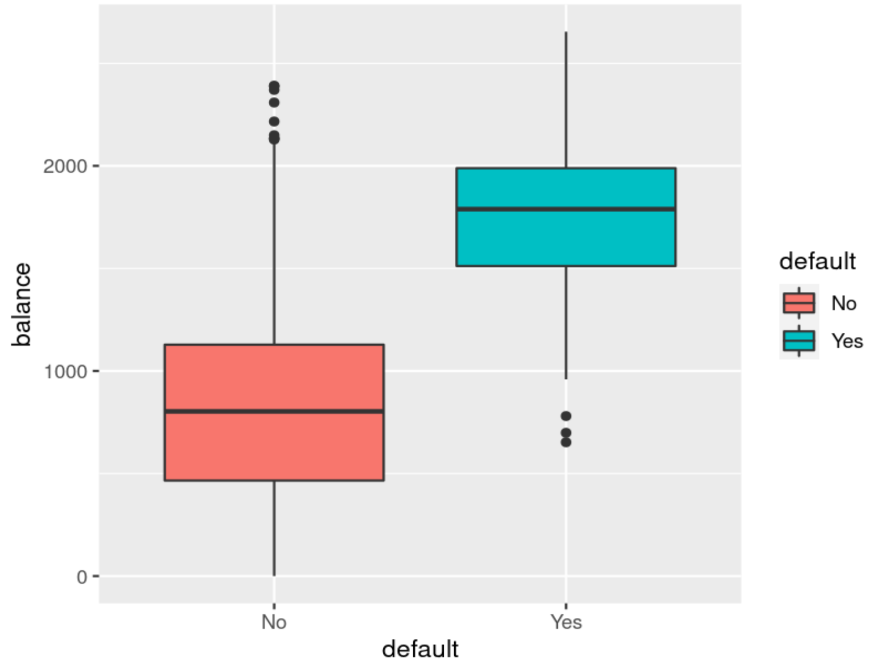
This package (as well as this example) are from the book:
“*An Introduction to Statistical Learning.*”

Let’s install and load the package, then view the data using `view(Default)`.

Strongly Related - Balance and Default!







Model Fit and Parameter Estimates

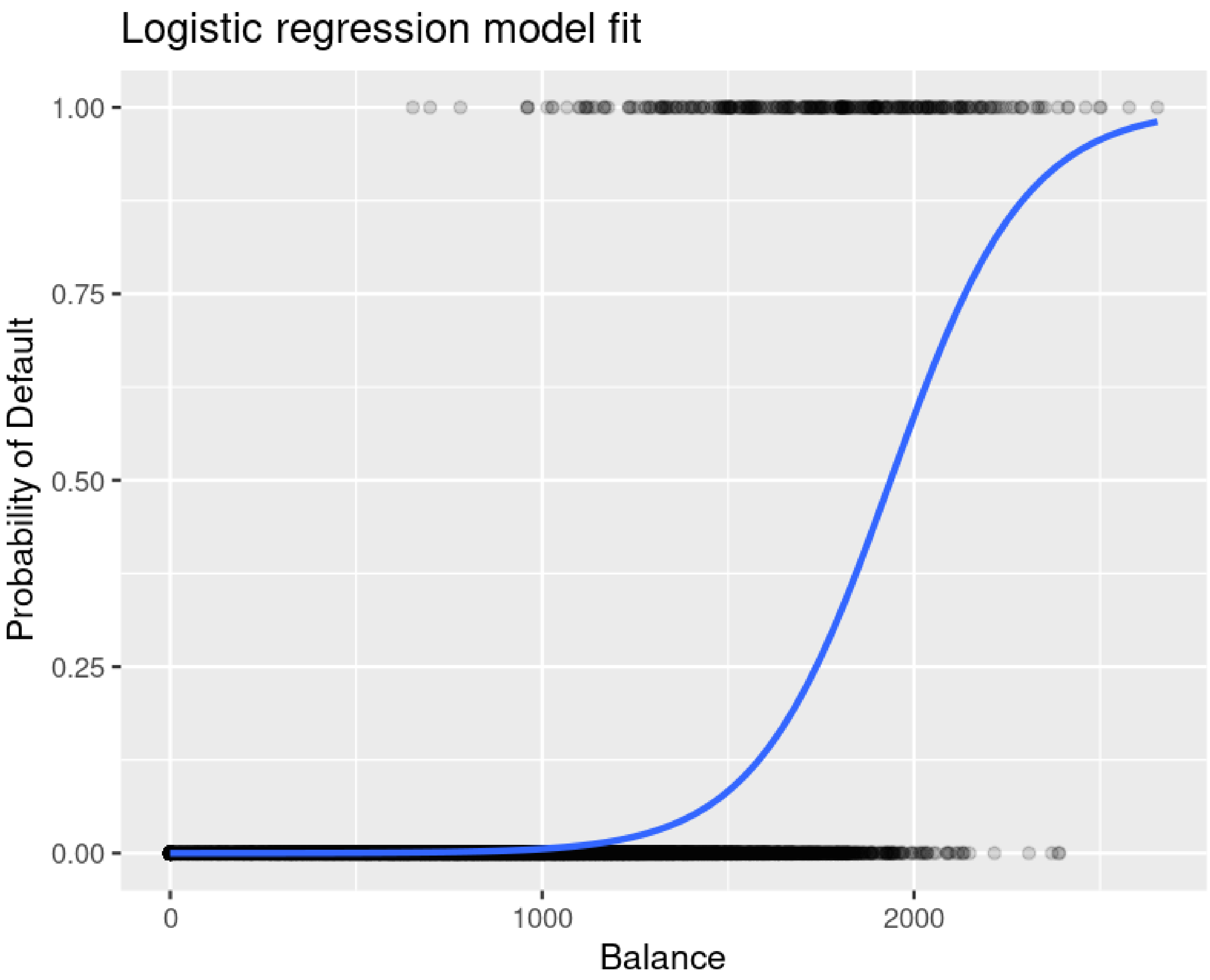
```
> options(scipen = 999)
> logitmodel <-glm(default01 ~ balance, data = Default01,
+                  family = "binomial")
> msummary(logitmodel)
Coefficients:
               Estimate  Std. Error z value             Pr(>|z|)
(Intercept) -10.6513306   0.3611574  -29.49 <0.0000000000000002 ***
balance       0.0054989   0.0002204   24.95 <0.0000000000000002 ***

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1596.5  on 9998  degrees of freedom
AIC: 1600.5
```

Plot of Regression Fit Line

```
gf_point(default01 ~ balance, data = Default01,
           alpha = 0.15,
           xlab ="Balance",
           ylab = "Probability of Default",
           title = "Logistic regression model fit") %>%
    gf_smooth(method = "glm", method.args = list(family = "binomial"))
```



Fitted Values with makeFun()

```
> logfit <-makeFun(logitmodel)
> logfit(1000)
          1
0.005752145
```

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1,000}}{1+e^{-10.6513+0.0055\times 1,000}} = 0.00576,$$

```
> exp(-10.6513306 + 0.0054989 * 1000)/ (1 + exp(-10.6513306 + 0.0054989 * 1000))
[1] 0.005752048
```

Fitted Values with predict()

```
> predict(logitmodel, data.frame(balance = c(1000,2000)), type = "response")
          1           2 
0.005752145 0.585769370 
```