MTH 3270 Notes 5

9 Statistical Foundations (9, E.1, E.2, E.5)

9.1 Samples and Populations

- **Data** are often collected by taking a ***random sample*** from a ***population***.

  Random sample data are needed if the goal is to draw ***statistical inferences*** (conclusions) from the data about the larger population.

9.2 Statistics, the Mean and Standard Error of a Statistic, and the Sampling Distribution of a Statistic

- A ***statistic*** is a numerical value computed from a set of data.

  Usually, the statistic **summarizes** some feature of the data (e.g. its central value or its spread).

  Statistics are often used to ***estimate*** the corresponding feature of the **population**.

  For example, with *numerical* data:

  - the ***sample mean*** $\bar{X}$ is used to *estimate* the ***population mean*** μ
  - the ***sample median*** $\tilde{X}$ is used to *estimate* the ***population median*** $\tilde{\mu}$
  - the ***sample standard deviation*** S is used to *estimate* the ***population standard deviation*** σ.

  and with *categorical* data:

  - the ***sample proportion*** $\hat{P}$ is used to *estimate* the ***population proportion*** P.

- Two **different random samples** from the **same population** will produce **different values** of a given **statistic**.

  This **chance variation** in the value of a statistic is called ***sampling variation***.

- The ***mean of a statistic*** is the value the statistic would take *on average* over repeatedly drawing samples from the **same population**.

- The ***standard error of a statistic*** is a value that measures the magnitude of its sampling variation, and is interpreted as the size of a **typical deviation** of the **statistic** away from its **mean**.

  A **smaller standard error** indicates the **statistic** is a **more precise** estimate of the corresponding **population parameter**.

- The ***sampling distribution*** of a statistic describes the **pattern** of the **sampling variation**.

  More formally, it's the **probability distribution** of the **statistic**, indicating the **values** the statistic might take and their **probabilities**.

  The **sampling distribution** can be interpreted as describing the **values** the statistic would take over repeated samples all drawn from the **same population**.

- The **standard error** of a statistic is just the **standard deviation** of the statistic's **sampling distribution**. The **mean** of the statistic is the **mean** of its **sampling distribution**.

- **Two ways** of determining a statistic's **sampling distribution** and the associated **standard error**:
  - Using ***mathematical theory***.
  - Using ***computer simulations***.

  Simulations are used when the mathematical theory is too difficult to work out.

9.3 Simulation

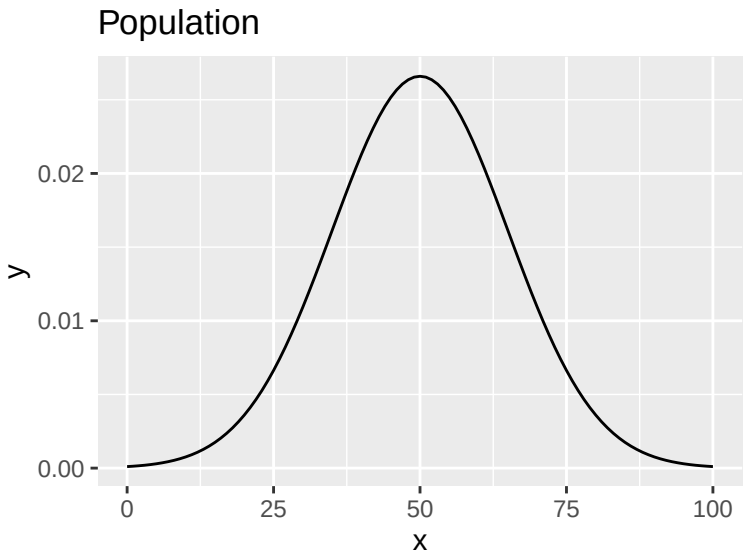
- **Simulation** is used to identify the **mean**, **standard error**, and **sampling distribution** of a **statistic** when the mathematical theory for identifying them is too difficult to work out.

  It's done by using a computer to generate (i.e. to ***simulate***) a large number of **random samples** (say, 1,000 of them) from the **population**, each sample of a given size $\boldsymbol{n}$, and computing the **statistic** of interest from each sample. Then:
  - The **mean** of the **statistic** is the *mean* of the 1,000 simulated values of the statistic.
  - The **standard error** of the **statistic** is the *standard deviation* of the 1,000 simulated values of the statistic.
  - The **sampling distribution** of the **statistic** is the *histogram* of the 1,000 simulated values of the statistic.

  In practice, it's usually *not possible* to simulate from the *actual* **population** (because we don't have data for the entire **population**). Two remedies are:
  - Simulation from a **probability distribution** that you think **mimics** the **population**.
  - Perform **bootstrap** simulations (more on this later).

- Consider a **population** represented by a ***normal distribution*** with **mean $\mu = 50$** and **standard deviation $\sigma = 15$**, i.e. a **N(50, 15)** distribution:

```
ggplot(data.frame(x = c(0, 100)), aes(x = x)) +
  stat_function(fun = dnorm, args = list(mean = 50, sd = 15)) +
  labs(title = "Population")
```



To **simulate** a random **sample** of size $\boldsymbol{n = 10}$ from the **N(50, 15) population** using the `rnorm()` function, type:

```
sim.sample <- rnorm(n = 10, mean = 50, sd = 15)
sim.sample
```

```
##  [1] 56.76978 51.14709 53.33778 64.04903 46.32967 45.88862 52.51233
##  [8] 58.09736 29.23560 32.23515
```

The **sample mean** $\bar{X}$ is

```
mean(sim.sample)

## [1] 48.96024
```

In this case the **error** in the *estimate* $\bar{X}$ of the population mean $\mu = 50$ is only $\bar{X} - \mu =$ -1.03976.

- You learned in your introductory statistics class that according to **mathematical theory**, *if* a **random sample** of size n is drawn from a $\mathbf{N(\mu, \sigma)}$ **population**,
  - The **mean** of $\bar{X}$ is μ.
  - The **standard error** of $\bar{X}$ is $\sigma/\sqrt{n}$ (which will be small when n is large).
  - The **sampling distribution** of $\bar{X}$ is $\mathbf{N(\mu, \sigma/\sqrt{n})}$

If the mathematical theory wasn't known, we could investigate the **mean**, **standard error**, and **sampling distribution** of $\bar{X}$ by **simulating**, say, 1,000 **random samples**, each of size $n = 10$, from the **population** and storing their $\bar{X}$ values in a 1,000-element *vector* named `sim.sample_means`:

```
# Create an empty vector to be filled in during loop iterations:
sim.sample_means <- rep(NA, 1000)

# Generate a new sample of size n = 10 during each loop iteration, save it's mean:
for(i in 1:1000) {
  sim.sample <- rnorm(n = 10, mean = 50, sd = 15)
  sim.sample_means[i] <- mean(sim.sample)
}
```

The **mean** of the **statistic** $\bar{X}$ is the *mean* of the 1,000 simulated $\bar{X}$ values:

```
sim.mean.xbar <- mean(sim.sample_means)
sim.mean.xbar

## [1] 49.84055
```

Note that this is quite close to the **mean** obtained via **mathematical theory**, $\mu = 50$.

The **standard error** of the **statistic** $\bar{X}$ is the *standard deviation* of the 1,000 simulated $\bar{X}$ values:

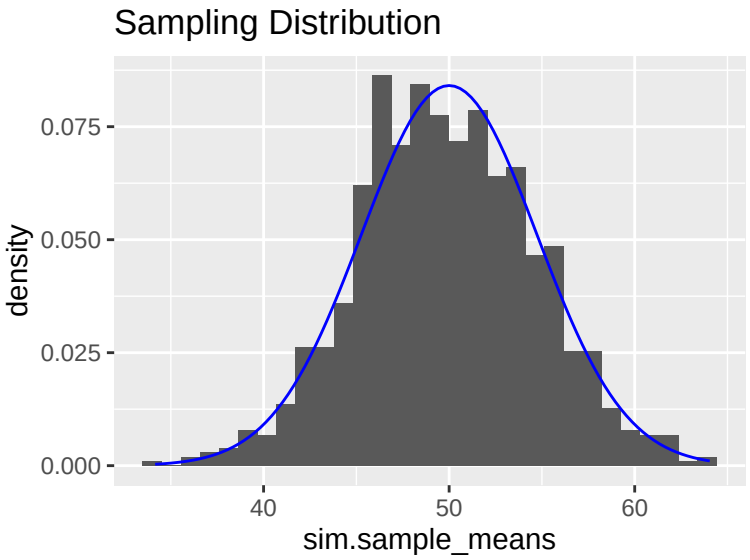
```
sim.se.xbar <- sd(sim.sample_means)
sim.se.xbar

## [1] 4.76193
```

Note that this is quite close to the **standard error** obtained via **mathematical theory**, $\sigma/\sqrt{n} = 15/\sqrt{10} =$ 4.74342.

The **sampling distribution** of the statistic $\bar{X}$ is depicted by a *histogram* of the 1,000 simulated $\bar{X}$ values:

```
ggplot(data = data.frame(sim.sample_means)) +
  geom_histogram(mapping = aes(x = sim.sample_means, y = stat(density))) +
  geom_function(fun = dnorm, args = list(mean = 50, sd = 15/sqrt(10)), color = "blue") +
  labs(title = "Sampling Distribution")
```



Note that this closely resembles the $N(\mu, \sigma/\sqrt{n}) = N(50, 15/\sqrt{10})$ **sampling distribution** obtained via **mathematical theory** (and shown as the blue curve with the histogram).

Section 9.3 Exercises

Exercise 1 This exercise involves simulations.

a) Using a `for()` loop and `rnorm()`, simulate 1,000 random samples of size $\boldsymbol{n = 10}$ from a N(50, 15) population (i.e. $\mu = 50$ and $\sigma = 15$), compute the sample mean $\bar{X}$ of each sample, and store the $\bar{X}$ values in a 1,000-element vector named, say, `sim.sample_means`. Report your R command(s).

b) Now use `mean()` and `sd()` to compute the mean and standard error of the 1,000 $\bar{X}$ values. Report these two values.

c) Recall that if a random sample of size n is drawn from a $N(\mu, \sigma)$ population, the *sampling distribution* of $\bar{X}$ (obtained via mathematical theory) is $N(\mu, \sigma/\sqrt{n})$.

Compare the two values of Part a to the theoretical mean and standard error, μ and $\sigma/\sqrt{n}$, of the sampling distribution of $\bar{X}$.

d) Make a histogram of the 1,000 simulated $\bar{X}$ values. Compare the shape, center, and spread of the histogram to the theoretical $N(\mu, \sigma/\sqrt{n})$ sampling distribution of $\bar{X}$.

Exercise 2 Simulate 1,000 random samples of size $\boldsymbol{n = 5}$ from a N(50, 15) population (i.e. $\mu = 50$ and $\sigma = 15$), and compute the four statistics below for each sample.

In each case: 1) Report the **mean** and **standard error** of the simulated statistic values, and 2) Plot the simulated values in a histogram and describe the shape, center, and spread of this **sampling distribution**.

a) The **sample median** $\boldsymbol{\tilde{X}}$ (use `median()`).

b) The **sample standard deviation** $\boldsymbol{S}$ (use `sd()`).

c) The **sample minimum** $\boldsymbol{X_{(1)}}$ (use `min()`).

d) The **sample maximum** $\boldsymbol{X_{(n)}}$ (use `max()`).

9.4 The Bootstrap

- When we *aren't able* to simulate from a **probability distribution** we think **mimics** the **population** (because we *don't know* which probability distribution mimics the population), an alternative is to perform **bootstrap** simulations to identify the **mean**, **standard error**, and **sampling distribution** of a **statistic**.

The ***bootstrap*** (B. Efron) refers to using the **original data set** (that's a **random sample** from the **population**) to **mimic** the **population**, then ***resampling*** (**with replacement**) from that **original data set**.

The ***resamples*** are drawn **with replacement** from the **original data set** using the **same sample size** n as the **original data set**:

```
Bootstrap:

1. Given a set of original data of size n, resample n observations with
   replacement from the data.

2. Compute the value of the statistic of interest from the resample.

3. Repeat steps 1 and 2 many, B, times (e.g. B = 1,000).

4. Use the mean, standard deviation, and histogram of the B values of the
   statistic as the mean, standard error, and sampling distribution of the
   statistic.
```

(Alternatively, the **mean** of the **statistic** could be taken to be the original value of the statistic as computed from the original data).

Data Set: `iris`

This famous (Fisher and Anderson's) `iris` data set, built in to R, gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of three species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

The five variables are:

`Sepal.Length`	The sepal length (cm)
`Sepal.Width`	The sepal width (cm)
`Petal.Length`	The petal length (cm)
`Petal.Width`	The petal width (cm)
`Species`	The species (Iris setosa, versicolor, or virginica).

- The following function (from the `"dplyr"` package) is useful for generating the **bootstrap resamples**.

```
slice_sample()      # Generate a random sample of n rows from a data frame
```

- For example, consider the famous built-in `iris` data set:

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

The **sample mean** petal width is $\bar{X} = 1.199$:

```
mean(iris$Petal.Width)

## [1] 1.199333
```

and this value ($\bar{X} = 1.199$) is an **estimate** of the (unknown) **population mean** petal width $\boldsymbol{\mu}$.

We want to know how **reliable** this **estimate** is. In other words, we'd like to know it's **standard error**.

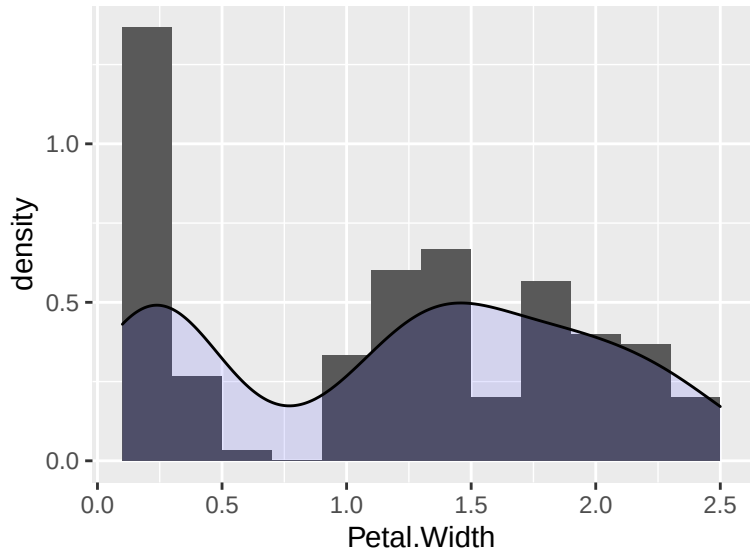
If we *knew* the `iris` petal width data were a random sample from a population that could be mimicked by a N(μ, σ) distribution, we could simulate samples from that normal distribution and assess the reliability of the estimate $\bar{X} = 1.199$ by determining its standard error as in Section 9.3.

However:

- We *wouldn't know* the values of μ and σ, so we'd have to use estimates of their values for performing the simulations.
- More importantly, we *don't know* whether the petal width population can be mimicked by a normal (bell-shaped) distribution.

In fact, the sample looks like it came from a **non-normal** (i.e. non-bell-shaped) population:

```
ggplot(data = iris, mapping = aes(x = Petal.Width, y = stat(density))) +
  geom_histogram(binwidth = 0.2) +
  geom_density(fill = "blue", alpha = 0.1)
```



So instead, we'll determine the **standard error** of our estimate ($\bar{X} = 1.199$) of μ using the **bootstrap** method.

For illustrative purposes, consider first a ***single* bootstrap resample** (i.e. $\boldsymbol{B = 1}$):

```
n.iris.rows <- nrow(iris)       # The original sample size, 150

# Set the seed (to allow regenerating the resample later):
set.seed(4)

# Generate a single resample (for now):
resamp <- slice_sample(.data = iris,
                       n = n.iris.rows,
                       replace = TRUE)

head(resamp)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          6.4         2.9          4.3         1.3 versicolor
## 2          7.0         3.2          4.7         1.4 versicolor
## 3          4.7         3.2          1.3         0.2     setosa
## 4          5.9         3.2          4.8         1.8 versicolor
## 5          5.8         2.8          5.1         2.4  virginica
## 6          7.0         3.2          4.7         1.4 versicolor
```

Setting `replace = TRUE` in `slice_sample()` indicates sampling **with replacement**. This allows the **resample** to include **duplicates** of rows from the original (`iris`) data set. Above, the 51st row from the original (`iris`) data set appears twice (labeled 2 and 6) in the **resample**.

The **sample mean** of this single **resample** from the original (`iris`) sample is $\bar{X} = 1.266$:

```
boot.sample_mean <- mean(resamp$Petal.Width)

boot.sample_mean

## [1] 1.266
```

Now we'll take $\boldsymbol{B = 1,000}$ bootstrap **resamples**, storing their $\bar{\boldsymbol{X}}$ values in a 1,000-element *vector* named `boot.sample_means`:

```
n.iris.rows <- nrow(iris)          # The original sample size, 150

B <- 1000                          # Number of bootstrap resamples, 1000

boot.sample_means <- rep(NA, B)    # Empty vector (initially), 1000 elements

for(i in 1:B) {
  resamp <- slice_sample(.data = iris,
                         n = n.iris.rows,
                         replace = TRUE)
  boot.sample_means[i] <- mean(resamp$Petal.Width)
}
```

The **mean** of the **statistic** $\bar{\boldsymbol{X}}$ is the *mean* of the 1,000 simulated $\bar{X}$ values:

```
boot.mean.xbar <- mean(boot.sample_means)
boot.mean.xbar

## [1] 1.199012
```

(Alternatively, the **mean** of the **statistic** $\bar{\boldsymbol{X}}$ could be taken to be the original value $\bar{X} = 1.199$ as computed from the original `iris` data).

The **standard error** of the **statistic** $\bar{\boldsymbol{X}}$ is the *standard deviation* of the 1,000 simulated $\bar{X}$ values:

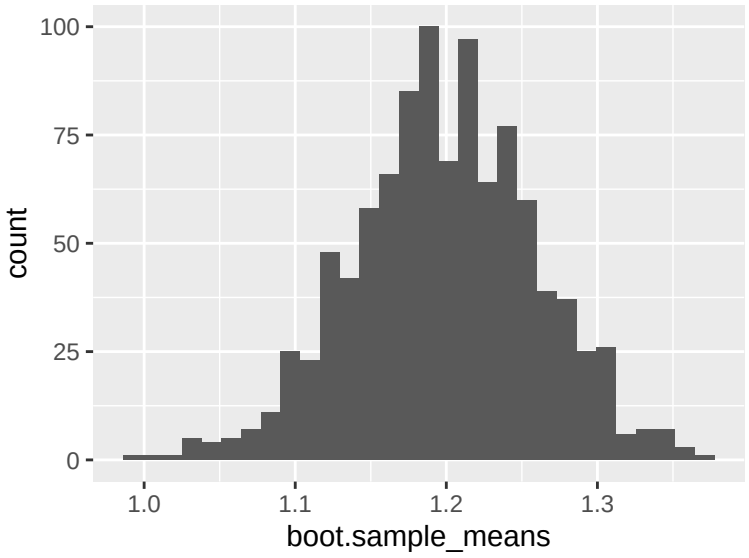
```
boot.se.xbar <- sd(boot.sample_means)
boot.se.xbar

## [1] 0.06044396
```

Interpretation: The **standard error**, 0.06, measures the magnitude of the **sampling variation** in the **original estimate** $\bar{X} = 1.199$ of the **population mean** petal width $\boldsymbol{\mu}$, and thus indicates how **reliable** this **estimate** is. Because the **standard error** 0.06 is small *relative* to the **estimate** $\bar{X} = 1.199$, the **estimate** is **reliable**.

The **sampling distribution** of the statistic $\bar{X}$ is depicted by a **histogram** of the 1,000 values:

```
ggplot(data = data.frame(boot.sample_means)) +
  geom_histogram(mapping = aes(x = boot.sample_means))
```



The fact that the **sampling distribution** of $\bar{X}$ is approximately **normal**, despite the **population** being **non-normal**, is a consequence of the ***Central Limit Theorem***.

Section 9.4 Exercises

Exercise 3 This exercise uses the famous built-in `iris` data set.

Use the **bootstrap** method to simulate $B = 1,000$ **resamples** each of size $n = 150$ from the original `iris` data set, and using the `Petal.Width` variable, compute the four statistics below for each **bootstrap resample**.

In each case: 1) Report the **mean** and **standard error** of the simulated statistic values, and 2) Plot the simulated values in a histogram and describe the shape, center, and spread of this **sampling distribution**.

a) The **sample median** $\tilde{X}$ (use `median()`).

b) The **sample standard deviation** S (use `sd()`).

c) The **sample minimum** $X_{(1)}$ (use `min()`).

d) The **sample maximum** $X_{(n)}$ (use `max()`).

9.5 Outliers

- An **outlier** is an observation that falls outside the overall pattern of observations in a data set.
- For example, the following data represent **numbers of deaths by lightning strikes** in the U.S. for each of the years 1959 - 2005 (in time order), as compiled by the *National Climatic Data Center* and the *National Weather Service*.

```
Year <- 1959:2005
Deaths <- c(75, 48, 61, 48, 150, 49, 57, 39, 27, 51, 46, 50, 62, 51, 50,
            58, 38, 34, 59, 44, 24, 39, 40, 33, 49, 33, 34, 32, 35, 30,
            23, 39, 36, 25, 20, 32, 43, 52, 42, 44, 46, 51, 47, 51, 44,
            33, 38)
LightningData <- data.frame(Year, Deaths)
```

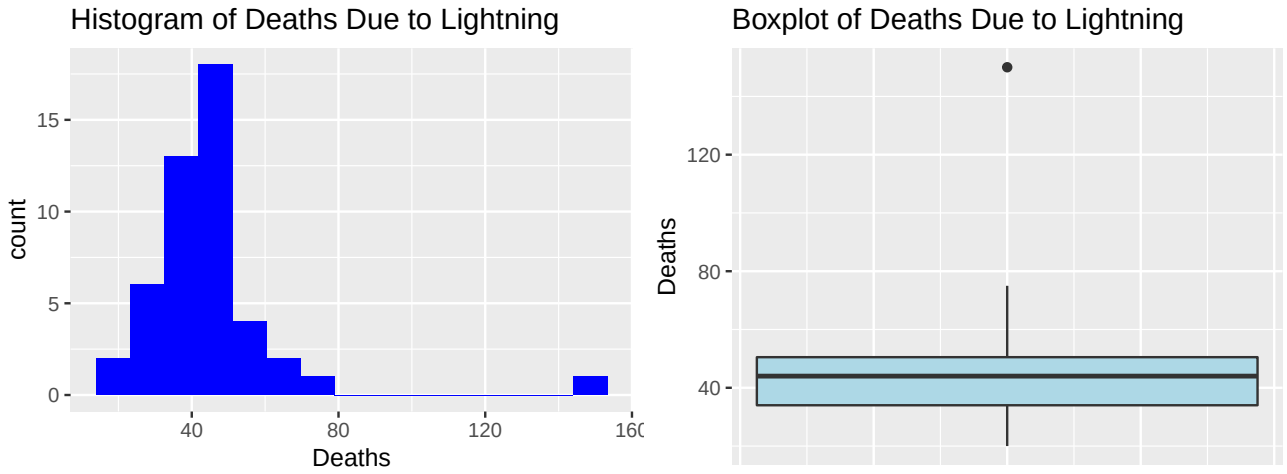


Figure 1

```
## Histogram
ggplot(data = LightningData) +
  geom_histogram(mapping = aes(x = Deaths), fill = "blue", bins = 15) +
  ggtitle("Histogram of Deaths Due to Lightning")
```

```
## Boxplot
ggplot(data = LightningData) +
  geom_boxplot(mapping = aes(y = Deaths), fill = "lightblue") +
  ggtitle("Boxplot of Deaths Due to Lightning") +
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())
```

Regarding the **outlier** (150 deaths in 1963), a *National Weather Service* report states:

> "On December 8, 1963 the crash of a jetliner killing 81 people near Elkin, Maryland, was attributed to lightning by the Civil Aeronautics Board investigators."

- **Comments**:
  - Outlying data values should be **checked for accuracy** (e.g. typos). Inaccuracies should be corrected.
  - Outliers may reveal **important insights**. Outliers **shouldn't** be dropped unless there's a clear rationale.
  - ***Robust*** statistical procedures are ones that aren't unduly affected by outliers (or other data irregularities).
  - ***Multivariate outliers*** might not show up in graphs. Instead, ***multivariate outlier detection*** procedures are needed to identify them.

Section 9.5 Exercises

Exercise 4 **Multivariate outliers** may not show up in graphs that don't show all the variables simultaneously.

Here are lengths (cm) and weights (g) of $\boldsymbol{n = 10}$ snakes, one of which is an **outlier**.

SnakeID	Length	Weight
1	85.7	331.9
2	64.5	121.5
3	84.1	382.2
4	82.5	287.3
5	78.0	224.3
6	65.9	380.4
7	81.3	245.2
8	71.0	208.2
9	86.7	393.4
10	78.7	228.3

```
SnakeID <- 1:10
Ln <- c(85.7, 64.5, 84.1, 82.5, 78.0, 65.9, 81.3, 71.0, 86.7, 78.7)
Wt <- c(331.9, 121.5, 382.2, 287.3, 224.3, 380.4, 245.2, 208.2, 393.4, 228.3)
Snakes <- data.frame(SnakeID, Ln, Wt)
```

a) Can you identify the **outlier** in either of these **univariate** graphs (histograms)?

```
ggplot(data = Snakes) +
  geom_histogram(mapping = aes(x = Ln),
                 fill = "blue",
                 color = "white",
                 bins = 5) +
  ggtitle("Histogram of Snakes Lengths")
```

```
ggplot(data = Snakes) +
  geom_histogram(mapping = aes(x = Wt),
                 fill = "blue",
                 color = "white",
                 bins = 5) +
  ggtitle("Histogram of Snakes Weights")
```

b) Can you identify the **outlier** in this **bivariate** graph (scatterplot)? If so, which snake (`SnakeID`) is the **outlier**?

```
ggplot(data = Snakes) +
  geom_point(mapping = aes(x = Ln, y = Wt)) +
  ggtitle("Scatterplot of Weights vs Lengths")
```

9.6 Statistical Models: Explaining Variation (9, E.1, E.2)

9.6.1 Simple Linear Regression: One Explanatory Variable

- ***Regression models*** describe **variation** in a ***response*** variable $\boldsymbol{Y}$ as a function of an ***explanatory*** variable $\boldsymbol{X}$.
- A ***simple linear regression analysis*** involves obtaining the **equation** of the **line** that best fits a scatterplot. The **equation** is useful for:
  1. **Predicting** the value of $\boldsymbol{Y}$ from a given value $\boldsymbol{X}$ (by plugging the X into the **equation** of the line).
  2. **Quantifying** a typical **change** in $\boldsymbol{Y}$ associated with a given **change** in $\boldsymbol{X}$ (using the **slope** of the line).

- We can carry out a **regression analysis** using the "linear model" function:

```
lm()          # Carry out a linear regression analysis by fitting a
              # linear model to a data set.
summary()     # Summarize the results of the regression analysis.
```

The resulting ***fitted regression line*** has an **equation** of the form

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X .$$

The **intercept** $\hat{\beta}_0$ and **slope** $\hat{\beta}_1$ are referred to as the ***coefficients*** of the **fitted model**.

- One way to obtain **predicted** values of Y is to use the following function.

```
predict()     # Returns the predicted response (Y) values from a fit-
              # ted regression model and a data frame of explanatory
              # (X) values.
```

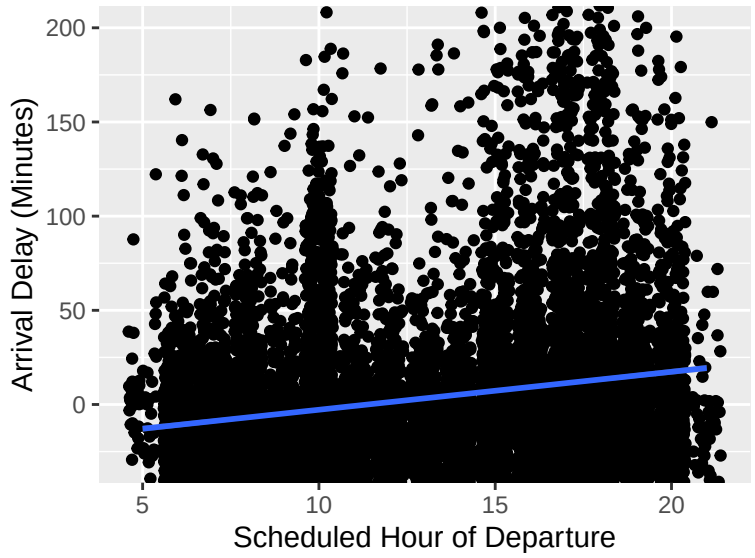
- Consider, as an example, the `flights` data (from the `"nycflights13"` package).

We'll investigate how the **arrival delay** (`arr_delay`) depends on the **scheduled departure hour** (`hour`) for flights to San Francisco. A plot of these variables is below.

```
library(nycflights13)

SF <- filter(.data = flights, dest == "SFO", !is.na(arr_delay))

ggplot(data = SF, mapping = aes(x = hour, y = arr_delay)) +
  geom_point(position = "jitter") +
  geom_smooth(method = "lm") +
  xlab("Scheduled Hour of Departure") + ylab("Arrival Delay (Minutes)") +
  coord_cartesian(ylim = c(-30, 200))
```



We carry out the **regression analysis** by typing:

```
my.reg <- lm(arr_delay ~ hour, data = SF)

summary(my.reg)
```

```
##
## Call:
## lm(formula = arr_delay ~ hour, data = SF)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -97.32 -25.22  -9.17   9.83 993.66
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -22.93267    1.23275  -18.60   <2e-16
## hour          2.01487    0.09154   22.01   <2e-16
##
## Residual standard error: 46.82 on 13171 degrees of freedom
## Multiple R-squared:  0.03548,Adjusted R-squared:  0.03541
## F-statistic: 484.5 on 1 and 13171 DF,  p-value: < 2.2e-16
```

In the output, the **coefficients** of the **equation** of the fitted line are in the `Estimate` column. Thus

$$\hat{\beta}_0 = -22.93 \qquad \text{and} \qquad \hat{\beta}_1 = 2.01,$$

so the **equation** is:

$$\hat{Y} = -22.93 + 2.01X$$

Using the equation, we **predict** the **arrival delay** for a flight whose scheduled departure is at **hour 15** by plugging **15** into the equation. Thus the **predicted delay** is:

$$\hat{Y} = -22.93 + 2.01(15) = \mathbf{7.22}$$

minutes, i.e.

```
-22.93 + 2.01 * 15
```

```
## [1] 7.22
```

Each **additional hour** in the **scheduled departure** time typically **delays** the **arrival** by an additional **2.01 minutes** (the **slope**).

- Another way to get **predicted values** from a **fitted model** is to use `predict()`.

To use `predict()`, we store the X value(s) for which we want to predict Y in a *data frame*. For example, to **predict** the **delay** for a flight whose scheduled departure is at **hour 15**, we create a *data frame* (`newHour`) containing the value **15** as the `hour`:

```
newHour <- data.frame(hour = 15)
newHour
```

```
##   hour
## 1   15
```

After fitting the model to the data in `SF`, saving the result as `my.reg` (as was done above), and creating the `newHour` data frame, we get the **predicted arrival delay** by typing:

```
predict(my.reg, newdata = newHour)
```

```
##        1
## 7.290446
```

Thus the **predicted arrival delay** is **7.29** minutes (which only differs from the earlier result because of round-off error).

For `predict()`, the variable name (`hour` above) in the new data frame (`newHour` above) needs to be the *same* as in the data frame used to build the model (`SF` above).

Section 9.6 Exercises

Exercise 5 Here are the data on lengths and weights of snakes (minus the outlier) from Exercise 4:

```
SnakeID <- 1:9
Ln <- c(85.7, 64.5, 84.1, 82.5, 78.0, 81.3, 71.0, 86.7, 78.7)
Wt <- c(331.9, 121.5, 382.2, 287.3, 224.3, 245.2, 208.2, 393.4, 228.3)
Snakes <- data.frame(SnakeID, Ln, Wt)
```

This command produces a scatterplot of the data:

```
ggplot(data = Snakes, mapping = aes(x = Ln, y = Wt)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  ggtitle("Scatterplot of Weights vs Lengths")
```

Obtain the equation of the fitted regression line, with `Wt` as the response (Y) and `Ln` as the explanatory variable (X), by typing:

```
my.reg <- lm(Wt ~ Ln, data = Snakes)
summary(my.reg)
```

a) From the output of `summary()`, the **equation** of the fitted line is:

$$\hat{Y} = -601.08 + 10.99X$$

Obtain the **predicted** weight for a snake whose length is **80** cm in two ways:

1. By plugging 80 into the equation for X.
2. By using `predict()`.

Report both sets of your R commands for obtaining the **predicted** weight.

b) What's a typical **change** in weight for each **1** cm **elongation**? What about for a **5** cm **elongation**?

Exercise 6 This exercise uses the `flights` data (from the `"nycflights13"` package).

We'll investigate how the **departure delay** (`dep_delay`) depends on the **scheduled departure hour** (`hour`) for flights to San Francisco.

Create the `SF` data set:

```
library(nycflights13)

SF <- filter(.data = flights, dest == "SFO", !is.na(arr_delay))
```

This command produces a plot showing the relationship between **departure delay** and **scheduled hour of departure**:

```
ggplot(data = SF, mapping = aes(x = hour, y = dep_delay)) +
  geom_point() +
  geom_smooth(method = "lm") +
  xlab("Scheduled Hour of Departure") + ylab("Departure Delay (Minutes)") +
  coord_cartesian(ylim = c(-30, 200))
```

a) Use `lm()` and `summary()` to obtain the equation of the fitted regression line, with with `dep_delay` as the response (Y) and `hour` as the explanatory variable (X). Report the **equation** of the fitted line.

b) Obtain the **predicted** departure delay for a flight whose departure hour is **15** in two ways:

1. By plugging 15 into the equation for X.
2. By using `predict()`.

Report both sets of your R commands for obtaining the **predicted** departure delay.

c) By how many minutes does the **departure delay** increase for each **additional hour** in the **scheduled departure**?

- The **vertical deviations** of the points in the scatterplot away from the fitted line are called ***residuals***.

Consider again the `Snakes` data frame from Exercise 5.

```
Snakes

##   SnakeID   Ln    Wt
## 1       1 85.7 331.9
## 2       2 64.5 121.5
## 3       3 84.1 382.2
## 4       4 82.5 287.3
## 5       5 78.0 224.3
## 6       6 81.3 245.2
## 7       7 71.0 208.2
## 8       8 86.7 393.4
## 9       9 78.7 228.3
```

After fitting the **regression line** to the data:

```
my.reg <- lm(Wt ~ Ln, data = Snakes)
```

the **residuals** are the *vertical line segments* shown in Fig. 2.

We can obtain the **residuals** (using `my.reg` from above) via `my.reg$residuals`. Below, we add them as a new column in `Snakes`:

```
library(dplyr)          # For mutate()

Snakes <- mutate(Snakes, Residuals = my.reg$residuals)
```

- A line fits the data "well" if the **residuals** are **small**.

The "best fitting" line (according to the ***least squares criterion***) is the one that minimizes the **sum of squared residuals**.

- The ***fitted values***, which we'll denote by $\hat{\boldsymbol{Y}}_{\boldsymbol{i}}$, are defined as

$$\hat{Y}_i \;=\; \hat{\beta}_0 + \hat{\beta}_1 X_i \,,$$

where the X_i's are the observed values of the explanatory variable (in the data set that the line was fitted to).

In Fig. 2, the **fitted values** are the y-coordinates of the points **on** the **fitted line** where the residual segments meet it.

We can obtain the **fitted values** (using `my.reg` from above) via `my.reg$fitted.values`. Below, we add them as a new column in `Snakes`:

```
Snakes <- mutate(Snakes, FittedVals = my.reg$fitted.values)
```

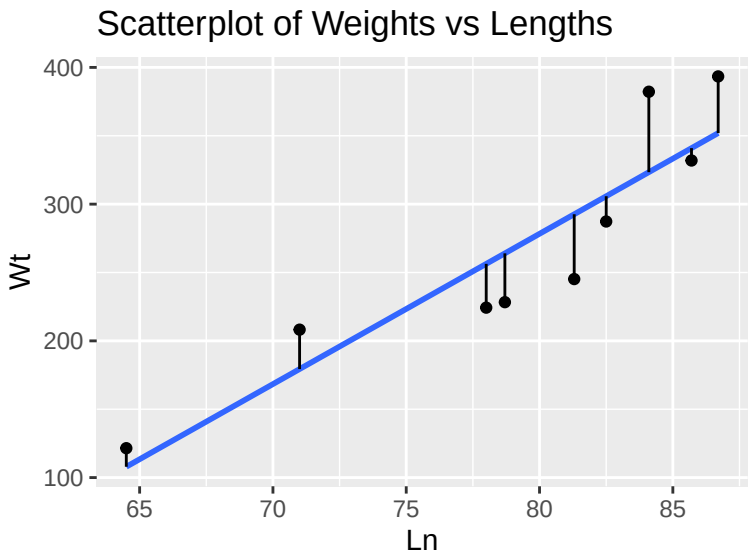


Figure 2

- Note that the ith **residual**, which we'll denote by $\boldsymbol{e_i}$, can be written as

$$e_i \;=\; Y_i - \hat{Y}_i\,,$$

where Y_i is the observed value of the response variable (in the data set that the line was fitted to) and $\hat{Y}_i$ is the ith fitted value.

Section 9.6 Exercises

Exercise 7 Here are the data on lengths and weights of snakes (minus the outlier) from Exercise 4:

```
SnakeID <- 1:9
Ln <- c(85.7, 64.5, 84.1, 82.5, 78.0, 81.3, 71.0, 86.7, 78.7)
Wt <- c(331.9, 121.5, 382.2, 287.3, 224.3, 245.2, 208.2, 393.4, 228.3)
Snakes <- data.frame(SnakeID, Ln, Wt)
```

Fit the linear regression model to the data:

```
my.reg <- lm(Wt ~ Ln, data = Snakes)
```

a) What class of object is returned by `lm()`? Find out by typing:

```
class(my.reg)
```

b) The `"lm"` class of objects is a special case of the `"list"` class. What does the following return?

```
is.list(my.reg)
```

c) How many objects are contained in the `my.reg` *list*? Find out by looking at their names:

```
names(my.reg)
```

d) Recall that the **equation** of the fitted line is

$$\hat{Y} = -601.08 + 10.99X$$

so the **nine** *fitted values* are defined as

$$\hat{Y}_i = -601.08 + 10.99X_i$$

where the X_i's are the **lengths** of the **nine** snakes in the `Snakes` data frame.

What would a plot of the **fitted values** versus the **lengths** look like? Try it, and describe the result:

```
library(dplyr)          # For mutate()

Snakes <- mutate(Snakes, FittedVals = my.reg$fitted.values)

ggplot(data = Snakes, mapping = aes(x = Ln, y = FittedVals)) +
  geom_point() +
  ggtitle("Scatterplot of Fitted Values vs Lengths")
```

e) What would a plot of the **residuals** versus the **lengths** look like? Try it (with a horizontal line at $y = 0$) and describe the result:

```
Snakes <- mutate(Snakes, Residuals = my.reg$residuals)

ggplot(data = Snakes, mapping = aes(x = Ln, y = Residuals)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  ggtitle("Scatterplot of Residuals vs Lengths")
```

9.6.2 Measuring the Fit of a Simple Linear Regression Model: SSE, MSE, and R^2

- A model fits the data "well" if the **residuals** are **small**.
- One measure of how well the model fits is the ***residual sum of squares*** (also called ***sum of squared errors***), denoted **SSE**:

$$\text{SSE} = \sum_{i=1}^{n} e_i^2 ,$$

where $\boldsymbol{e_i}$ is the ith **residual**.

A **smaller** SSE indicates a **better fit**.

One way to obtain the **SSE** is as follows (using the `Snakes` data, with residuals, from above):

```
my.sse <- sum(Snakes$Residuals^2)

my.sse

## [1] 11164.58
```

- SSE depends on the number of observations n, so it's better to measure the fit by the ***mean squared residual*** (also called ***mean squared error***), denoted **MSE** and defined as:

$$\text{MSE} = \frac{\text{SSE}}{n-2} .$$

MSE is the "average" *squared residual* (but using $n-2$ instead of n).

- MSE is measured in the *squared* units of Y (e.g. if Y is measured in dollars, MSE is measured in dollars *squared*).

  Its square root, $\boldsymbol{\sqrt{\text{MSE}}}$, called the ***residual standard error*** (also called ***root mean squared residual*** or ***root mean squared error***), is easier to interpret because it's measured in the same units as Y.

  A **smaller** $\sqrt{\text{MSE}}$ indicates a **better fit**.

  For example, here (again) are the results of the regression of **weights** on **lengths** of snakes:

```
my.reg <- lm(Wt ~ Ln, data = Snakes)

summary(my.reg)

##
## Call:
## lm(formula = Wt ~ Ln, data = Snakes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.395 -32.020  -9.061  28.826  58.827
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -601.083    154.333  -3.895 0.005939
## Ln            10.992      1.942   5.660 0.000767
##
## Residual standard error: 39.94 on 7 degrees of freedom
## Multiple R-squared:  0.8207,Adjusted R-squared:  0.795
## F-statistic: 32.03 on 1 and 7 DF,  p-value: 0.0007667
```

  From the output, the **residual standard error** is $\boldsymbol{\sqrt{\text{MSE}} = 39.94}$, which represents the **size** of a **typical residual**.

- The residual standard error depends on the units of Y (e.g. dollars vs euros, inches vs cm, etc.), so it's sometimes desirable instead to measure the fit by the $\boldsymbol{R^2}$, defined as:

$$R^2 \;=\; 1 \;-\; \frac{\text{SSE}}{(n-1)\text{Var}(Y_i\text{'s})} \;=\; 1 \;-\; \frac{\sum_{i=1}^n e_i^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2},$$

  where $\text{Var}(Y_i\text{'s})$ is the ***variance*** (squared standard deviation) of the Y_i's.

  $\boldsymbol{R^2}$ always falls between **0** and **1**. A **larger** R^2 indicates a **better fit** of the model to the data.

  For example, from the output of `summary()` above, the $\boldsymbol{R^2}$ (labeled `Multiple R-squared`) is $\boldsymbol{R^2 = 0.8207}$, indicating a **good fit** of the line to the snakes data.

Section 9.6 Exercises

Exercise 8 This exercise uses the `flights` data (from the `"nycflights13"` package).

From Exercise 6, here are the plot and linear regression of **departure delay** on **scheduled departure hour** for flights to San Francisco (using the `SF` data frame from Exercise 6):

```
ggplot(data = SF, mapping = aes(x = hour, y = dep_delay)) +
  geom_point() +
  geom_jitter() +
  geom_smooth(method = "lm") +
  xlab("Scheduled Hour of Departure") + ylab("Departure Delay (Minutes)") +
  coord_cartesian(ylim = c(-30, 200))

my.reg <- lm(dep_delay ~ hour, data = SF)

summary(my.reg)
```

a) From the output of `summary()`, what's the value of the **residual standard error**?

b) From the output of `summary()`, what's the value of $\boldsymbol{R^2}$ (labeled `Multiple R-squared`)?

c) Using the criteria below (and the R^2 from part *b*), how well does the linear model fit the data (poor, medium, or good)?

R^2	Model Fit
0.0 to 0.25	Poor
0.25 to 0.65	Medium
0.65 to 1.0	Good

9.6.3 Multiple Regression: Multiple Explanatory Variables

- ***Multiple regression models*** describe **variation** in a **response** variable $\boldsymbol{Y}$ as a function of **several** explanatory variables $\boldsymbol{X_1, X_2, \ldots, X_p}$.
- A ***multiple regression analysis*** involves obtaining the **equation** of the **plane** or **"hyperplane"** that best fits the data. The equation is useful for:
  1. **Predicting** the value of $\boldsymbol{Y}$ from given values of $\boldsymbol{X_1, X_2, \ldots, X_p}$.
  2. **Quantifying** a typical **change** in $\boldsymbol{Y}$ associated with given **changes** in $\boldsymbol{X_1, X_2, \ldots, X_p}$.

9.6.4 Multiple Regression with Two Explanatory Variables

- When $\boldsymbol{p = 2}$ (i.e. two explanatory variables), a multiple regression analysis involves obtaining the **equation** of the **plane** that best fits the data in a three-dimensional scatterplot.
- We carry out the analysis and view the results using `lm()` and `summary()`, as before.

The resulting ***fitted regression model*** is the **equation** of a **plane**, i.e. having the form

$$\hat{Y} \;=\; \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 \,.$$

The **intercept** is $\boldsymbol{\hat{\beta}_0}$, and $\boldsymbol{\hat{\beta}_1}$ and $\boldsymbol{\hat{\beta}_2}$ are referred to as ***coefficients*** of the **fitted model**.

Data Set: `waterUsage`

The `waterUsage` data contain, for $\boldsymbol{n = 28}$ U.S. cities, the **four** variables:

`City`	Name of the city.
`Water`	The city's water consumption (log of millions of liters/day).
`Population`	The population of the city (in millions, year 2000)
`Wealth`	A measure of the city's wealth (z-score of the city's median income).

Water Usage for U.S. Metropolitan Areas

City	Water Usage (Y)	Wealth (X_1)	Population (X_2)
New York	9.2	2.8	21.3
Los Angeles	9.1	0.1	16.4
Chicago	8.4	-0.2	9.2
DC/Baltimore	8.1	1.8	6.5
San Francisco	8.0	1.9	6.3
⋮	⋮	⋮	⋮
Stockton	5.7	-0.9	0.6
Mobile	6.4	-1.5	0.5

```
waterUsage <- data.frame(City = c("New_York", "Los_Angeles", "Chicago", "DC_Baltimore",
                                  "San_Francisco", "Detroit_Ann_Arbor", "Dallas",
                                  "Atlanta", "Seattle", "Miami", "Phoenix",
                                  "Minneapolis", "Denver", "Pittsburgh", "St_Louis",
                                  "Portland_Salem", "San_Antonio", "Salt_Lake_City",
                                  "Las_Vegas", "Providence", "Jacksonville",
                                  "Dayton_Springfield", "Albany_Schenectady",
                                  "Albuquerque", "Omaha", "Little_Rock", "Stockton",
                                  "Mobile"),
                         Water = c(9.2, 9.1, 8.4, 8.1, 8.0, 7.9, 7.9, 7.6, 7.6, 7.6,
                                   7.6, 6.8, 7.2, 6.7, 6.7, 7.1, 6.7, 7.0, 7.1, 6.2,
                                   6.0, 6.1, 6.1, 6.3, 6.0, 5.7, 5.7, 6.41),
                         Wealth = c(2.8, 0.1, -0.2, 1.8, 1.9, 0.3, 0.4, 0.9, -0.3,
                                    -0.5, 0.1, 0.9, 0.7, -1.3, 0.0, -0.4, -1.3, -1.2,
                                    0.1, 0.0, -0.3, -0.3, 0.1, -0.5, -0.4, -0.9, -0.9,
                                    -1.5),
                         Population = c(21.3, 16.4, 9.2, 6.5, 6.3, 5.5, 5.2, 4.3, 3.6,
                                    3.9, 3.3, 3.0, 2.6, 2.5, 2.2, 2.3, 1.6, 1.3, 1.6,
                                    1.5, 1.1, 1.0, 0.9, 0.7, 0.7, 0.6, 0.6, 0.5))
```

For example, using the `waterUsage` data set, we can obtain the ***fitted regression model***, with response variable **water consumption** ($\boldsymbol{Y}$) and *both* **wealth** ($\boldsymbol{X_1}$) *and* **population size** ($\boldsymbol{X_2}$) as explanatory variables, by typing:

```
my.reg <- lm(Water ~ Wealth + Population, data = waterUsage)

summary(my.reg)

##
## Call:
## lm(formula = Water ~ Wealth + Population, data = waterUsage)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.95881 -0.50504  0.06659  0.40889  0.58966
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.47674    0.14084  45.987  < 2e-16
## Wealth       0.11042    0.12603   0.876    0.389
## Population   0.15835    0.02633   6.014 2.78e-06
##
## Residual standard error: 0.5063 on 25 degrees of freedom
## Multiple R-squared:  0.7441,Adjusted R-squared:  0.7237
## F-statistic: 36.35 on 2 and 25 DF,  p-value: 3.985e-08
```

In the output, the **coefficients** of the **equation** of the fitted plane are in the `Estimate` column. Thus

$$\hat{\boldsymbol{\beta}}_{\mathbf{0}} = \mathbf{6.48}, \quad \hat{\boldsymbol{\beta}}_{\mathbf{1}} = \mathbf{0.11}, \quad \text{and} \quad \hat{\boldsymbol{\beta}}_{\mathbf{2}} = \mathbf{0.16},$$

so the **equation** is:

$$\hat{Y} \;=\; 6.48 \;+\; 0.11\,X_1 \;+\; 0.16\,X_2\,.$$

This is the **equation** of the **plane** shown below.

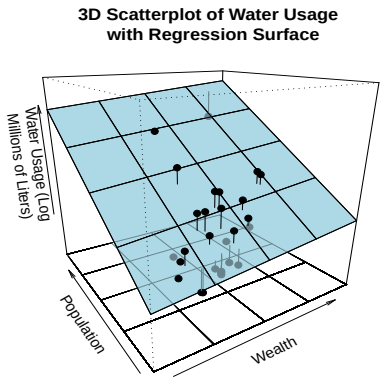


Figure 3

Using the equation, we **predict** the **water usage** for a city whose **wealth** is **1.5** *and* whose **population** is **3.0** by plugging **1.5** and **3.0** into the equation. Thus the **predicted water usage** is:

$$\hat{Y} \;=\; 6.48 \;+\; 0.11\,(1.5) \;+\; 0.16\,(3.0) \;=\; \mathbf{7.1}\,.$$

The **coefficient** $\hat{\boldsymbol{\beta}}_{\mathbf{1}} \mathbf{= 0.11}$ says each one-unit increase in a city's **wealth** leads to a water usage *increase* of **0.11** units when **population** is **fixed** (held **constant**).

- In general, when $\boldsymbol{p = 2}$ (i.e. two explanatory variables),

  1. The ***intercept*** $\hat{\boldsymbol{\beta}}_{\mathbf{0}}$ is the predicted value of $\boldsymbol{Y}$ when $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ are **both zero**. The intercept is usually **not** of interest.
  2. The ***coefficient*** $\hat{\boldsymbol{\beta}}_{\mathbf{1}}$ quantifies the **change** in $\boldsymbol{Y}$ for each **one-unit change** in $\boldsymbol{X_1}$, **while** $\boldsymbol{X_2}$ is **fixed** (held **constant**).
  3. The ***coefficient*** $\hat{\boldsymbol{\beta}}_{\mathbf{2}}$ quantifies the **change** in $\boldsymbol{Y}$ for each **one-unit change** in $\boldsymbol{X_2}$, **while** $\boldsymbol{X_1}$ is **fixed** (held **constant**).

- Another way to get **predicted values** from a **fitted model** is to use `predict()`.

To use `predict()`, we store the X_1 and X_2 values for which we want to predict Y in a *data frame*. For example, to **predict** the **water usage** for a city whose **wealth** score is **1.5** and whose **population** is **3.0** (million), we create a *data frame* (`newWealthPop`) containing the values **1.5** as the `Wealth` and **3.0** as the `Population`:

```
newWealthPop <- data.frame(Wealth = 1.5, Population = 3.0)
newWealthPop

##   Wealth Population
## 1    1.5          3
```

After fitting the model to the data in `waterUsage`, saving the result as `my.reg` (as was done above), and creating the `newWealthPop` data frame, we get the **predicted water usage** by typing:

```
predict(my.reg, newdata = newWealthPop)

##        1
## 7.117426
```

Thus the **predicted water usage** is **7.1** units (the same as the earlier result).

For `predict()`, the variable names (`Wealth` and `Population` above) in the new data frame (`newWealthPop` above) need to be the *same* as in the data frame used to build the model (`waterUsage` above).

Data Set: `portraitSales`

The `portraitSales` data set (below) was provided by Dwaine Studios, Inc., a portrait studio that specializes in portraits of children and operates in 21 cities. The data set includes information for each city about sales, number of persons under age 16, and per capita income.

`Sales`	Portrait sales (in thousands of dollars).
`Under16`	Number of persons under age 16 (in thousands of persons)
`Income`	Per capita disposable personal income (in thousands of dollars).

```
Sales <- c(174.4, 164.4, 244.2, 154.6, 181.6, 207.5, 152.8, 163.2,
           145.4, 137.2, 241.9, 191.1, 232.0, 145.3, 161.1, 209.7,
           146.4, 144.0, 232.6, 224.1, 166.5)

Under16 <- c(68.5, 45.2, 91.3, 47.8, 46.9, 66.1, 49.5, 52.0, 48.9,
             38.4, 87.9, 72.8, 88.4, 42.9, 52.5, 85.7, 41.3, 51.7,
             89.6, 82.7, 52.3)

Income <- c(16.7, 16.8, 18.2, 16.3, 17.3, 18.2, 15.9, 17.2, 16.6, 16.0,
            18.3, 17.1, 17.4, 15.8, 17.8, 18.4, 16.5, 16.3, 18.1, 19.1,
            16.0)

portraitSales <- data.frame(Sales, Under16, Income)
```

Section 9.6 Exercises

Exercise 9 This exercise uses the `portraitSales` data from above.

a) Use `lm()` to fit a multiple regression model with portrait `Sales` as the response (Y) and number of persons `Under16` (X_1) and per capita `Income` (X_2) as explanatory variables. Then use `summary()` to obtain the results. Write out the **equation** of the fitted plane.

b) Obtain the **predicted** sales for a city whose number of persons under 16 is **45.0** (thousand) and whose per capita income is **17.0** (thousand dollars) in two ways:

1. By plugging 45.0 and 17.0 into the equation for X_1 and X_2.
2. By using `predict()`.

Report both sets of your R commands for obtaining the **predicted** sales.

c) By how much do we expect **sales** to increase for each additional **1.0** thousand people **under 16** (holding income constant)?

d) By how much do we expect **sales** to increase for each additional **1.0** thousand dollars in per capita **income** (holding number of people under 16 constant)?

9.6.5 Multiple Regression with More than Two Explanatory Variables

- When $p > 2$ (i.e. more than two explanatory variables), a ***multiple regression model*** is no longer a plane in three dimensions, it's a **"hyperplane"** in higher dimensions, which shares many features of a plane.
- We carry out the analysis and view the results using `lm()` and `summary()`, as before.

The resulting ***fitted regression model*** is the **equation** of a **"hyperplane"**, i.e. having the form

$$\hat{Y} \;=\; \hat{\beta}_0 \,+\, \hat{\beta}_1\, X_1 \,+\, \hat{\beta}_2\, X_2 \,+\, \cdots \,+\, \hat{\beta}_p\, X_p\,.$$

The **intercept** is $\hat{\boldsymbol{\beta}}_{\mathbf{0}}$, and $\hat{\boldsymbol{\beta}}_{\mathbf{1}}, \hat{\boldsymbol{\beta}}_{\mathbf{2}}, \ldots, \hat{\boldsymbol{\beta}}_{\boldsymbol{p}}$ are referred to as ***coefficients*** of the **fitted model**.

- ***Scatterplot matrices*** and ***correlation matrices***, produced by the following functions, are useful for exploring and summarizing the data.

```
pairs()          # Make a scatterplot matrix of variables in a data frame.
cor()            # Make a correlation matrix of variables in a data frame.
```

Data Set: `waste`

The `waste` data below were used for the design of an efficient waste incinerator. On each of $\boldsymbol{n = 30}$ waste specimens, **six variables** were recorded:

`Specimen`	ID number identifying the waste specimen (1-30)
`EnergyContent`	The energy content of the specimen (kcal/kg), a measure of burnability
`Plastics`	Plastics in the specimen (percent, by weight)
`Paper`	Paper in the specimen (percent, by weight)
`Garbage`	Garbage in the specimen (percent, by weight)
`Water`	Moisture in the specimen (percent, by weight)

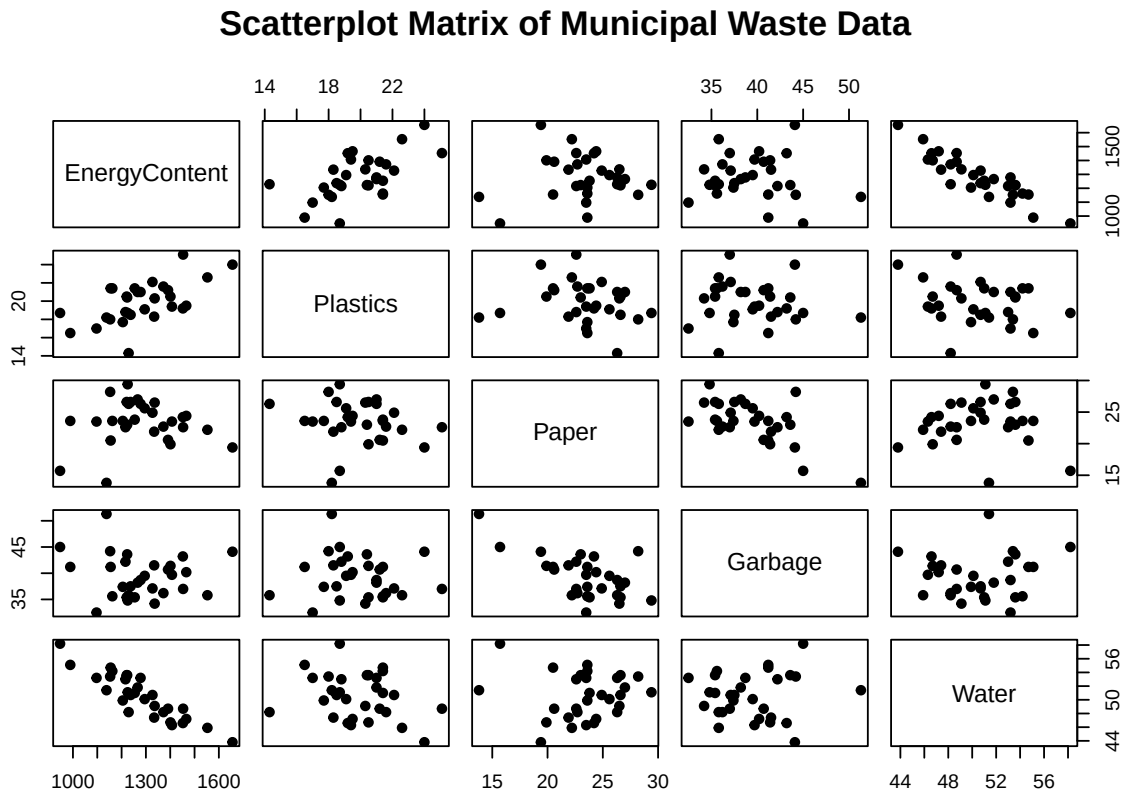
Municipal Waste Composition

Waste Specimen	Energy Content	Plastics	Paper	Garbage	Water
1	947	18.69	15.65	45.01	58.21
2	1407	19.43	23.51	39.69	46.31
3	1452	19.24	24.23	43.16	46.63
4	1553	22.64	22.20	35.76	45.85
5	989	16.54	23.56	41.20	55.14
⋮	⋮	⋮	⋮	⋮	⋮
29	1391	21.25	20.63	40.72	48.67
30	1372	21.62	22.71	36.22	48.19

```
waste <- data.frame(Specimen = 1:30,
                    EnergyContent = c(947, 1407, 1452, 1553, 989, 1162, 1466,
                        1656, 1254, 1336, 1097, 1266, 1401, 1223, 1216, 1334,
                        1155, 1453, 1278, 1153, 1225, 1237, 1327, 1229, 1205,
                        1221, 1138, 1295, 1391, 1372),
                    Plastics = c(18.7, 19.4, 19.2, 22.6, 16.5, 21.4, 19.5, 24.0,
                        21.4, 20.3, 17.0, 21.0, 20.5, 20.4, 18.8, 18.3, 21.4,
                        25.1, 21.0, 18.0, 18.7, 18.5, 22.1, 14.3, 17.7, 20.5,
                        18.2, 19.1, 21.2, 21.6),
                    Paper = c(15.7, 23.5, 24.2, 22.2, 23.6, 23.6, 24.4, 19.4,
                        23.8, 26.5, 23.5, 27.0, 19.9, 23.0, 22.6, 21.9, 20.5,
                        22.6, 26.3, 28.2, 29.4, 26.6, 24.9, 26.3, 23.6, 26.6,
                        13.8, 25.6, 20.6, 22.7),
                    Garbage = c(45.0, 39.7, 43.2, 35.8, 41.2, 35.6, 40.2, 44.1,
                        35.4, 34.2, 32.5, 38.2, 41.4, 43.6, 42.2, 41.5, 41.2,
                        37.0, 38.7, 44.2, 34.8, 37.5, 37.1, 35.8, 37.4, 35.4,
                        51.3, 39.5, 40.7, 36.2),
                    Water = c(58.2, 46.3, 46.6, 45.9, 55.1, 54.2, 47.2, 43.8,
                        51.0, 49.1, 53.2, 51.8, 46.7, 53.6, 53.0, 47.4, 54.7,
                        48.7, 53.2, 53.4, 51.1, 50.7, 50.7, 48.2, 49.9, 53.6,
                        51.4, 50.1, 48.7, 48.2))
```

- As an example, here's a **scatterplot matrix** of the `waste` data:

```
pairs(select(waste, -Specimen),
      main = "Scatterplot Matrix of Municipal Waste Data",
      pch = 19)
```



Here's the **correlation matrix**:

```
cor_mat <- cor(select(waste, -Specimen))
round(cor_mat, 2)

##               EnergyContent Plastics Paper Garbage Water
## EnergyContent          1.00     0.59  0.04   -0.09 -0.90
## Plastics               0.59     1.00 -0.15   -0.09 -0.26
## Paper                  0.04    -0.15  1.00   -0.63  0.00
## Garbage               -0.09    -0.09 -0.63    1.00  0.07
## Water                 -0.90    -0.26  0.00    0.07  1.00
```

The **correlation matrix** shows the ***correlations*** corresponding to the plots in the **scatterplot matrix**.

We can obtain the ***fitted regression model***, with response variable **energy content** ($\boldsymbol{Y}$) and **plastics** ($\boldsymbol{X_1}$), **paper** ($\boldsymbol{X_2}$), **garbage** ($\boldsymbol{X_3}$), and **water** ($\boldsymbol{X_4}$) as the $\boldsymbol{p = 4}$ explanatory variables by typing:

```
my.reg <- lm(EnergyContent ~ Plastics + Paper + Garbage + Water, data = waste)

summary(my.reg)

##
## Call:
## lm(formula = EnergyContent ~ Plastics + Paper + Garbage + Water,
##     data = waste)
```

```
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -39.47 -24.05 -11.72  21.45  60.56
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2234.508    178.576  12.513 2.91e-12
## Plastics      29.090      2.836  10.256 1.92e-10
## Paper          7.738      2.329   3.323  0.00274
## Garbage        4.353      1.926   2.260  0.03282
## Water        -37.291      1.840 -20.270  < 2e-16
##
## Residual standard error: 31.58 on 25 degrees of freedom
## Multiple R-squared:  0.9639,Adjusted R-squared:  0.9581
## F-statistic: 166.6 on 4 and 25 DF,  p-value: < 2.2e-16
```

In the output, the **coefficients** of the **equation** of the fitted hyperplane are in the `Estimate` column. Thus

$$\hat{\beta}_0 = \mathbf{2234.5}, \quad \hat{\beta}_1 = \mathbf{29.1}, \quad \hat{\beta}_2 = \mathbf{7.7}, \quad \hat{\beta}_3 = \mathbf{4.4}, \quad \text{and} \quad \hat{\beta}_4 = \mathbf{-37.3},$$

so the **equation** is: the **equation** is:

$$\hat{Y} \;=\; 2234.5 \,+\, 29.1\, X_1 \,+\, 7.7\, X_2 \,+\, 4.4\, X_3 \,-\, 37.3\, X_4.$$

This is the **equation** of a **"hyperplane"** in a five-dimensional coordinate system.

Using the equation, we **predict** the **energy content** for a waste specimen whose **plastics** is **20**, **paper** is **25**, **garbage** is **40**, and **water** is **15** by plugging **20**, **25**, **40**, and **15** into the equation. Thus the **predicted energy content** is:

$$\hat{Y} \;=\; 2234.5 \,+\, 29.1\,(20) \,+\, 7.7\,(25) \,+\, 4.4\,(40) \,-\, 37.3\,(15) \;=\; \mathbf{2625.5}\,.$$

The **coefficient $\hat{\beta}_1 = 29.1$** says each one-percent increase in a waste specimen's **plastics** leads to an energy content *increase* of **29.1** units when **paper**, **garbage**, and **water** are **fixed** (held **constant**).

- In general, when $\boldsymbol{p > 2}$, the coefficients have the same interpretation as they did when $p = 2$:

  1. The ***intercept*** $\hat{\beta}_0$ is the predicted value of $\boldsymbol{Y}$ when $\boldsymbol{X_1, X_2, \ldots, X_p}$ are **all zero**. The intercept is usually **not** of interest.
  2. For each $k = 1, 2, \ldots, p$, the ***coefficient*** $\hat{\beta}_k$ quantifies the **change** in $\boldsymbol{Y}$ for each **one-unit change** in $\boldsymbol{X_k}$, **while the other $\boldsymbol{p-1}$ $\boldsymbol{X_i}$'s** are **fixed** (held **constant**).

- Another way to get **predicted values** from a **fitted model** is to use `predict()`.

To use `predict()`, we store the $X_1, X_2, \ldots, X_p$ values for which we want to predict Y in a *data frame*. For example, to **predict** the **energy content** for a waste specimen whose **plastics** is **20**, **paper** is **25**, **garbage** is **40**, and **water** is **15**, we create a *data frame* (`newWaste`) containing the values **20** as the `Plastics`, **25** as the `Paper`, **40** as the `Garbage`, and **15** as the `Water`:

```
newWaste <- data.frame(Plastics = 20,
                       Paper = 25,
                       Garbage = 40,
                       Water = 15)
newWaste
```

```
##   Plastics Paper Garbage Water
## 1       20    25      40    15
```

After fitting the model to the data in `waste`, saving the result as `my.reg` (as was done above), and creating the `newWaste` data frame, we get the **predicted energy content** by typing:

```
predict(my.reg, newdata = newWaste)

##        1
## 2624.515
```

Thus the **predicted energy content** is **2624.5** units (the same as the earlier result up to round-off error).

For `predict()`, the variable names (`Plastics`,`Paper`, `Garbage`, and `Water` above) in the new data frame (`newWaste` above) need to be the *same* as in the data frame used to build the model (`waste` above).

Data Set: `cdi`

The `cdi` data set (**CDI.txt** on the course Canvas website) provides selective county demographic information (CDI) for 440 of the most populous counties in the U.S. Each line of the data set has an identification number with a county name and state abbreviation and provides information on 14 variables for a single county. Counties with missing data were deleted from the data set. The information generally pertains to the years 1990 and 1992. The **17 variables** are:

`ID`	Identification number (1-440)
`County`	County name
`State`	Two-letter state abbreviation
`LandArea`	Land area (square miles)
`TotPop`	Estimated 1990 population
`PctPop18_34`	Percent of 1990 CDI population aged 18-34
`PctPop65`	Percent of 1990 CDI population aged 65 or older
`nActPhys`	Number of professionally active nonfederal physicians during 1990
`nHospBeds`	Total number of beds, cribs, and bassinets during 1990
`nCrimes`	Total number of serious crimes in 1990, including murder, rape, robbery, aggravated assault, burglary, larceny-theft, and motor vehicle theft, as reported by law enforcement agencies
`PctHSGrad`	Percent of adult population (aged 25 or older) who completed 12 or more years of school
`PctBach`	Percent of adult population (aged 25 or older) with a bachelor's degree
`PctBelPov`	Percent of 1990 CDI population with income below poverty level
`PctUnemp`	Percent of 1990 CDI labor force that was unemployed
`PerCapInc`	Per capita income of 1990 CDI population (dollars)
`TotInc`	Total personal income of 1990 CDI population (in millions of dollars)
`Region`	Geographic region classification that is used by the U.S. Bureau of the Census, where 1=NE, 2=NC, 3=S, 4=W

Section 9.6 Exercises

Exercise 10 This exercise uses the `cdi` data set (**CDI.txt** on the course Canvas website).

a) Use `pairs()` to make a **scatterplot matrix** of these variables. Report your R command.

b) Use `cor()` to make the **correlation matrix** of the data. Report your R command.

c) Use `lm()` to fit a multiple regression model with **number of active physicians** as the response (Y) and **total population** (X_1), **land area** (X_2), and **total personal income** (X_3) as explanatory variables. Then use `summary()` to obtain the results. Write out the **equation** of the fitted model.

d) Obtain the **predicted** number of active physicians for a county with a total population of **400,000**, a land area of **1,000** square miles, and total personal income of **8,000** million dollars in two ways:

1. By plugging 400,000, 1,000, and 8,000 into the equation for X_1, X_2, and X_3.
2. By using `predict()`.

e) How much does **number of active physicians** increase for each additional **1**-person increase in total population (holding land area and total personal income constant)?

f) How much does **number of active physicians** increase for each additional **1.0** million dollars in total personal income (holding land area and total population constant)?

Exercise 11 This exercise also uses the `cdi` data set (**CDI.txt** on the course Canvas website).

The following uses `mutate()` (from the `"dplyr"` package) to create a new variable in the `cdi` data frame, `PopDens`, containing the **population density** of each county:

```
cdi <- mutate(cdi, PopDens = TotPop/LandArea)
```

a) Use `lm()` to fit a multiple regression model with **number of active physicians** as the response (Y) and **population density** (X_1), **percent of population 65 or older** (X_2), and **per capita income** (X_3) as explanatory variables. Then use `summary()` to obtain the results. Report the **equation** of the fitted model.

b) Obtain the **predicted** number of active physicians for a county with a population density of **900** per square mile, **15** percent of its population over 65, and per capita income of **20,000** dollars in two ways:

1. By plugging 900, 15, and 20,000 into the equation for X_1, X_2, and X_3.
2. By using `predict()`.

Report both sets of your R commands for obtaining the **predicted** number of active physicians.

- The **deviations** of the observations away from the fitted model are called ***residuals***.

  For example, consider (again) the `waterUsage` data frame (from above).

```
head(waterUsage)
```

```
##                City Water Wealth Population
## 1          New_York   9.2    2.8       21.3
## 2       Los_Angeles   9.1    0.1       16.4
## 3           Chicago   8.4   -0.2        9.2
## 4      DC_Baltimore   8.1    1.8        6.5
## 5     San_Francisco   8.0    1.9        6.3
## 6 Detroit_Ann_Arbor   7.9    0.3        5.5
```

  After fitting the **regression model** to the data:

```
my.reg <- lm(Water ~ Wealth + Population, data = waterUsage)
```

  the **residuals** are the *vertical line segments* shown in Fig. 3.

  We can obtain the **residuals** (using `my.reg` from above) via `my.reg$residuals`. Below, we add them as a new column in `waterUsage`:

```
waterUsage <- mutate(waterUsage, Residuals = my.reg$residuals)
```

- A model fits the data "well" if the **residuals** are **small**.

  The "best fitting" model (according to the ***least squares criterion***) is the one that minimizes the **sum of squared residuals**.

- The ***fitted values***, which we'll denote by $\hat{\boldsymbol{Y}}_{\boldsymbol{i}}$, are defined as

$$\hat{Y}_i \;=\; \hat{\beta}_0 \,+\, \hat{\beta}_1\, X_{1i} + \hat{\beta}_2 X_{2i} + \cdots + \hat{\beta}_p X_{pi}\,,$$

where the $X_{1i}, X_{2i}, \ldots, X_{pi}$'s are the observed values of the p explanatory variables (in the data set that the model was fitted to).

In Fig. 3, the **fitted values** are the y-coordinates of the points **on** the **fitted plane** where the residual segments meet it.

We can obtain the **fitted values** (using `my.reg` from above) via `my.reg$fitted.values`. Below, we add them as a new column in `waterUsage`:

```
waterUsage <- mutate(waterUsage, FittedVals = my.reg$fitted.values)
```

- Note that the ith **residual**, which we'll denote by $\boldsymbol{e_i}$, can be written as

$$e_i \;=\; Y_i - \hat{Y}_i\,,$$

where Y_i is the observed value of the response variable (in the data set that the line was fitted to) and $\hat{Y}_i$ is the ith fitted value.

9.6.6 Measuring the Fit of a Multiple Regression Model: SSE, MSE, and R^2

- A model fits the data "well" if the **residuals** are **small**.
- One measure of how well the model fits is the ***residual sum of squares*** (also called ***sum of squared errors***), denoted **SSE**:

$$\text{SSE} \;=\; \sum_{i=1}^{n} e_i^2\,,$$

where $\boldsymbol{e_i}$ is the ith **residual**.

A **smaller** SSE indicates a **better fit**.

One way to obtain the **SSE** is as follows (using the `waterUsage` data, with residuals, from above):

```
my.sse <- sum(waterUsage$Residuals^2)

my.sse

## [1] 6.408783
```

- SSE depends on the number of observations n, so it's better to measure the fit by the ***mean squared residual*** (also called ***mean squared error***), denoted **MSE** and defined as:

$$\text{MSE} \;=\; \frac{\text{SSE}}{n-(p+1)}\,.$$

MSE is the "average" *squared residual* (but using $n-(p+1)$ instead of n).

- MSE is measured in the *squared* units of Y (e.g. if Y is measured in dollars, MSE is measured in dollars *squared*).

Its square root, $\boldsymbol{\sqrt{\text{MSE}}}$, called the ***residual standard error*** (also called ***root mean squared residual*** or ***root mean squared error***), is easier to interpret because it's measured in the same units as Y.

A **smaller** $\sqrt{\text{MSE}}$ indicates a **better fit**.

For example, here (again) are the results of the regression of **water usage** on **wealth** and **population** of U.S. cities:

```
my.reg <- lm(Water ~ Wealth + Population, data = waterUsage)

summary(my.reg)

##
## Call:
## lm(formula = Water ~ Wealth + Population, data = waterUsage)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.95881 -0.50504  0.06659  0.40889  0.58966
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.47674    0.14084  45.987  < 2e-16
## Wealth       0.11042    0.12603   0.876    0.389
## Population   0.15835    0.02633   6.014 2.78e-06
##
## Residual standard error: 0.5063 on 25 degrees of freedom
## Multiple R-squared:  0.7441,Adjusted R-squared:  0.7237
## F-statistic: 36.35 on 2 and 25 DF,  p-value: 3.985e-08
```

From the output, the **residual standard error** is $\sqrt{\text{MSE}} = \mathbf{0.5063}$, which represents the **size** of a **typical residual**.

- The residual standard error depends on the units of Y (e.g. dollars vs euros, inches vs cm, etc.), so it's sometimes desirable instead to measure the fit by the $\boldsymbol{R^2}$, defined as:

$$R^2 \;=\; 1 \;-\; \frac{\text{SSE}}{(n-1)\text{Var}(Y_i\text{'s})} \;=\; 1 \;-\; \frac{\sum_{i=1}^{n} e_i^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2},$$

where $\text{Var}(Y_i\text{'s})$ is the ***variance*** (squared standard deviation) of the Y_i's.

$\boldsymbol{R^2}$ always falls between **0** and **1**. A **larger** R^2 indicates a **better fit** of the model to the data.

For example, from the output of `summary()` above, the $\boldsymbol{R^2}$ (labeled `Multiple R-squared`) is $\boldsymbol{R^2 = 0.7441}$, indicating a **decent fit** of the model to the water usage data.

Section 9.6 Exercises

Exercise 12 This exercise uses the `cdi` data set (**CDI.txt** on the course Canvas website). You'll compare the fits of the two models from Exercises 10 and 11.

After reading the data into a data frame named `cdi`, fit the model from Exercise 10:

```
my.reg <- lm(nActPhys ~ TotPop + LandArea + TotInc, data = cdi)

summary(my.reg)
```

Now re-create the `PopDens` variable and fit the model from Exercise 11:

```
cdi <- mutate(cdi, PopDens = TotPop/LandArea)

my.reg <- lm(nActPhys ~  PopDens + PctPop65 + PerCapInc, data = cdi)

summary(my.reg)
```

a) Using the **residual standard error** in the output from `summary()`, which model fits the data better?
Hint: A *smaller* residual standard error indicates a *better* fitting model.

b) Using the $\boldsymbol{R^2}$ (labeled `Multiple R-squared`) in the output from `summary()`, which model fits the data better? **Hint**: A *larger* R^2 indicates a *better* fitting model.

c) Based on your answers to Parts a and b, which model would you expect to give **better predictions** of the number of active physicians in a county?

9.7 Logistic Regression: Dichotomous (0 or 1) Response Variable (E.5)

- ***Logistic regression*** is used when the response variable $\boldsymbol{Y}$ is ***dichotomous***, that is, only takes **two** values (e.g. Yes/No, Healthy/Unhealthy, etc.), which we code as **0** and **1**.

For a **dichotomous** response variable, we (usually) want to estimate the **probability** that $\boldsymbol{Y}$ will equal **1** as a **function** of an explanatory variable $\boldsymbol{X}$.

Data Set: `dues`

The `dues` data set (**DUES.txt** on the course website) contains responses to a survey of **30** members conducted by the board of directors of a professional association to assess the effects of several possible amounts of dues increase. The **two variables** are:

`DuesIncr`	The amount of dues increase
`NotRenew`	Whether the interviewee would not renew their membership at that amount of dues increase (1 if they would not renew, 0 if they would renew)

- For example, consider the `dues` data set:

```
head(dues)

##   NotRenew DuesIncr
## 1        0       25
## 2        0       27
## 3        0       30
## 4        0       30
## 5        0       31
## 6        0       32
```

We might want to model the **probability** of a person **not renewing** their membership as a **function** of the dues increase $\boldsymbol{X}$.

A *linear regression model* is **not appropriate** because the line extends above and below the range of probability values 0 to 1 (see Fig. 4):

```
ggplot(data = dues, mapping = aes(x = DuesIncr, y = NotRenew)) +
  geom_point() +
  labs(title = "Not Renew vs Dues Increase", x = "Dues Increase", y = "Not Renew") +
  geom_smooth(method = "lm", se = FALSE) +
  scale_y_continuous(breaks = seq(-0.25, 1.25, 0.25),
                     labels = seq(-0.25, 1.25, 0.25),
                     limits = c(-0.25, 1.25))
```

Here's a (more appropriate) **fitted *logistic regression model*** (Fig. 5):

```
ggplot(data = dues, mapping = aes(x = DuesIncr, y = NotRenew)) +
  geom_point() +
  labs(title = "Not Renew vs Dues Increase", x = "Dues Increase", y = "Not Renew") +
  geom_smooth(method = "glm",
              method.args = list(family = "binomial"),
```

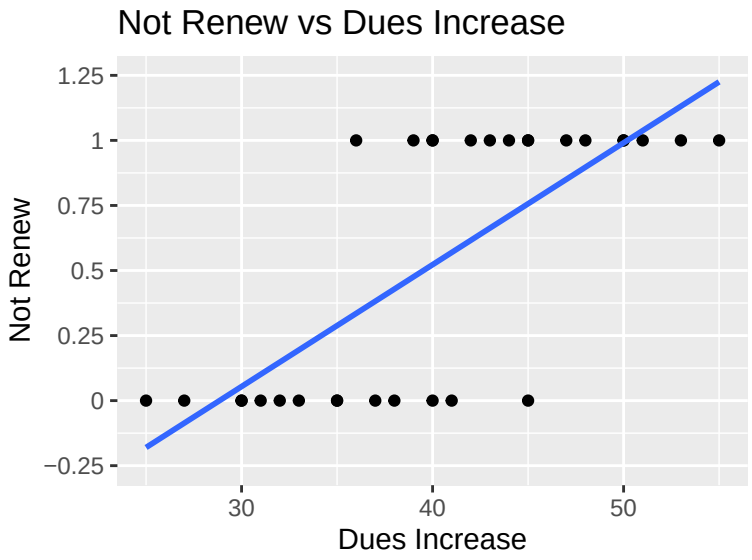


Figure 4

```
            se = FALSE) +
scale_y_continuous(breaks = seq(-0.25, 1.25, 0.25),
                   labels = seq(-0.25, 1.25, 0.25),
                   limits = c(-0.25, 1.25))
```

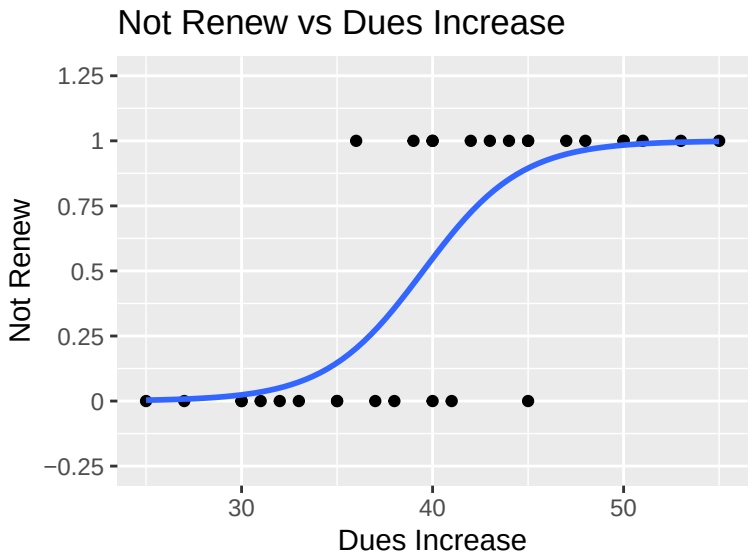


Figure 5

The curve in Fig. 5 gives the (estimated) **probability** of **not renewing** for any given value of the **dues increase** $\boldsymbol{X}$.

- We can carry out a **logistic regression analysis** using the "generalized linear model" function:

```
glm()       # Carry out a logistic regression analysis by fitting a logistic
            # regression model to a data set.  Specify family = "binomial".
            # Other so-called generalized linear models use different
            # family specifications.
summary()   # Summarize the results of the logistic regression analysis.
```

- For example, using the `dues` data set, with **not renewing** as the response (Y) and **dues increase** as the explanatory variable (X), we carry out the **logistic regression analysis** by typing:

```
my.logreg <- glm(NotRenew ~ DuesIncr, data = dues, family = "binomial")

summary(my.logreg)

##
## Call:
## glm(formula = NotRenew ~ DuesIncr, family = "binomial", data = dues)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.12290  -0.37290   0.08522   0.47113   1.78651
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -15.4157     5.6780  -2.715  0.00663
## DuesIncr      0.3902     0.1421   2.745  0.00605
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 41.455  on 29  degrees of freedom
## Residual deviance: 20.083  on 28  degrees of freedom
## AIC: 24.083
##
## Number of Fisher Scoring iterations: 6
```

  We'll interpret some of the output later.

- To understand the **fitted logistic regression model**, we'll define a function $\boldsymbol{p(X)}$ as

$$p(X) \;=\; P(Y = 1 \text{ when the value of the explanatory variable is } X)$$

  The ***fitted logistic regression model*** has the form

$$p(X) \;=\; \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} \tag{1}$$

  (were $\boldsymbol{e}$ is the *exponential constant*, $e = 2.718282...$).

  The **fitted logistic regression model** has the following properties (see Fig. 6):

  1. The fitted curve $p(X)$ is constrained to lie between zero and one.
  2. If $\hat{\beta}_1 > 0$, then $p(X)$ is an increasing function of X. Also, $p(X) \to 1$ as X increases and $p(X) \to 0$ as X decreases.
  3. If $\hat{\beta}_1 < 0$, $p(X)$ is a decreasing function of X. Also, $p(X) \to 0$ as X increases and $p(X) \to 1$ as X decreases.
  4. $\hat{\beta}_1$ determines the "steepness" of the "middle" part of the fitted curve $p(X)$. A larger $\hat{\beta}_1$ results in a "steeper" curve.
  5. $\hat{\beta}_0$ shifts the fitted curve left or right.

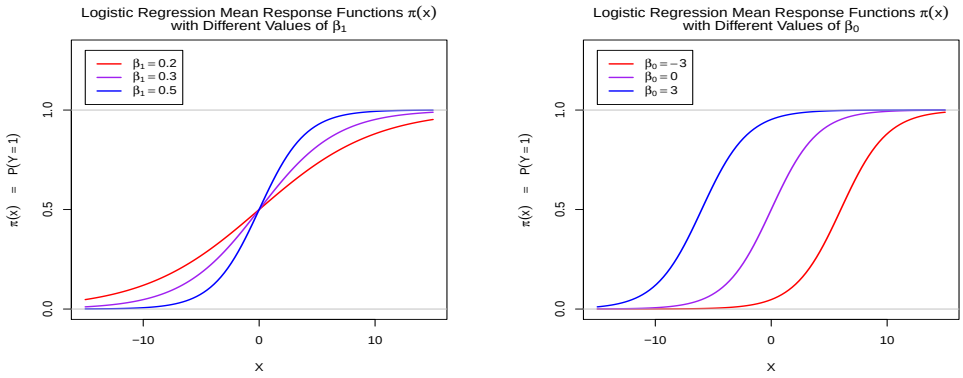


Figure 6

Refer to the output from `summary(my.logreg)` above. For the `dues` data, the **coefficients** of the **equation** of the *fitted logistic regression model* are in the `Estimate` column. Thus

$$\hat{\boldsymbol{\beta}}_{\mathbf{0}} = \mathbf{-15.42} \qquad \text{and} \qquad \hat{\boldsymbol{\beta}}_{\mathbf{1}} = \mathbf{0.39},$$

so the **equation** is:

$$p(X) \;=\; \frac{e^{-15.42+0.39X}}{1+e^{-15.42+0.39X}}\,.$$

This is the curve graphed in Fig. 5.

If we plug any value for X into the equation, we get the (estimated) **probability** that an individual **won't renew** their membership at that **dues increase** value X. For example, plugging **$42** in for the **dues increase** X gives

$$p(42) \;=\; \frac{e^{-15.42+0.39(42)}}{1+e^{-15.42+0.39(42)}} \;=\; 0.72\,,$$

i.e.

```
exp(-15.42 + 0.39 * 42)/(1 + exp(-15.42 + 0.39 * 42))

## [1] 0.7231218
```

There's a **72% chance** that a given individual **won't renew** their membership if the **dues increase** is **$42**.

We could also get this value using `predict()`:

```
newDues <- data.frame(DuesIncr = 42)

predict(my.logreg, newDues, type = "response")

##         1
## 0.7254854
```

Specifying `type = "response"` is necessary because the default `type` for `predict.glm()` (the `predict()` *method* for `"glm"` objects) is `"link"`, and for `type = "link"`, `predict.glm()` returns *not* the value of $p(X)$, but rather the value of the so-called ***logit function***, defined as:

$$\text{logit}(p(X)) \;=\; \log\left(\frac{p(X)}{1-p(X)}\right)\,.$$

It can be shown that when $p(X)$ is defined as in (1),

$$\text{logit}(p(X)) \;=\; \hat{\beta}_0 + \hat{\beta}_1 X\,.$$

Section 9.7 Exercises

Exercise 13 This exercise uses the `dues` data set (**DUES.txt** on the course Canvas website).

Fit the **logistic regression model** to the data, with `NotRenew` as the response variable (Y) and `DuesIncr` as the explanatory variable (X), using `glm()` with `family = "binomial"`:

```
my.logreg <- glm(NotRenew ~ DuesIncr, family = "binomial", data = dues)

summary(my.logreg)
```

a) From the output of `summary()`, the **equation** of the fitted logistic regression model is:

$$p(X) = \frac{e^{-15.42+0.39X}}{1+e^{-15.42+0.39X}}.$$

See Fig. 5.

Obtain the (estimated) **probability** that a person **won't renew** their membership if the **dues increase** is **45** dollars in two ways:

1. By plugging 45 into the equation for X.
2. By using `predict()`.

Report both sets of your R commands for obtaining the (estimated) **probability** of **not renewing**.

b) Now obtain the (estimated) **probability** that a person **won't renew** their membership if the **dues increase** is only **35** dollars dollars in two ways:

1. By plugging 35 into the equation for X.
2. By using `predict()`.

Report both sets of your R commands for obtaining the (estimated) **probability** of **not renewing**.

9.8 Model Complexity and Overfitting

- We'll want to know how "good" a given model is. There are *two ways* to think about this:
  - How well does it **predict** the response variable in **original data set** (that was used to build the model and to which the model was fitted)? We can use $\sqrt{\textbf{MSE}}$, $\boldsymbol{R^2}$, etc. to decide.
  - How well does it **predict** the response for **new** observations (**not** part of the original data set)? We can use **cross-validation** to decide (we'll see how later).
- A model ***overfits*** the **original data** if it predicts the *those* responses well, but *not* responses of **new** observations.

  Overfitting occurs when the fitted model **conforms too closely** to **random fluctuations** in the data instead of capturing the **systematic pattern** in the data.
- **Overfitting** results when the ***complexity*** of the model is *too high*.

  Model complexity is defined differently depending on the type of model being fitted. For example,
  - In **multiple regression**, **complexity** refers to the **number** of **explanatory variables** in the model.
  - In **polynomial regression** (see below), **complexity** refers to the **number** of **polynomial terms** in the model.

- For example, here are the data on **lengths** and **weights** of **nine** snakes:

```
SnakeID <- 1:9
Ln <- c(85.7, 64.5, 84.1, 82.5, 78.0, 81.3, 71.0, 86.7, 78.7)
Wt <- c(331.9, 121.5, 382.2, 287.3, 224.3, 245.2, 208.2, 393.4, 228.3)

Snakes <- data.frame(SnakeID, Ln, Wt)

g <- ggplot(Snakes, aes(x = Ln, y = Wt)) + geom_point()
g
```

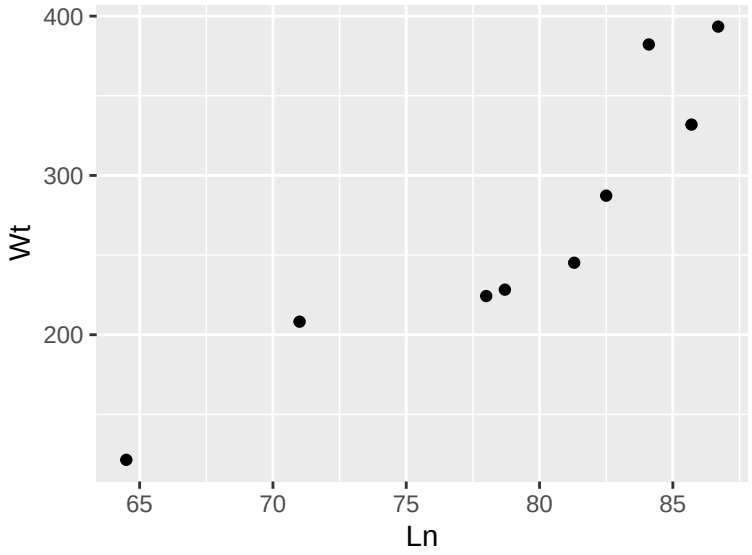


Figure 7

Below, we fit each of these ***polynomial regression models*** to the data:

$$\begin{aligned}
\text{Model 0}: \quad Y &= \hat{\beta}_0 \\
\text{Model 1}: \quad Y &= \hat{\beta}_0 + \hat{\beta}_1 X \\
\text{Model 2}: \quad Y &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 \\
\text{Model 3}: \quad Y &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 \\
\text{Model 4}: \quad Y &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_4 X^4 \\
\text{Model 5}: \quad Y &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_4 X^4 + \hat{\beta}_5 X^5
\end{aligned}$$

where Y is the **weight** and X the **length** of a snake.

```
# Constant model (Model 0):
g + stat_smooth(method = "lm", formula = y ~ 1, se = F) +
  ggtitle(label = "Model 0")
```

```
# Linear polynomial model (Model 1):
g + stat_smooth(method = "lm", formula = y ~ poly(x, 1), se = F) +
  ggtitle(label = "Model 1")
```

```
# Quadratic polynomial model (Model 2):
g + stat_smooth(method = "lm", formula = y ~ poly(x, 2), se = F) +
  ggtitle(label = "Model 2")
```

```
# Cubic polynomial model (Model 3):
g + stat_smooth(method = "lm", formula = y ~ poly(x, 3), se = F) +
  ggtitle(label = "Model 3")
```

```
# Quartic polynomial model (Model 4):
g + stat_smooth(method = "lm", formula = y ~ poly(x, 4), se = F) +
  ggtitle(label = "Model 4")
```

```
# Quintic polynomial model (Model 5):
g + stat_smooth(method = "lm", formula = y ~ poly(x, 5), se = F) +
  ggtitle(label = "Model 5")
```

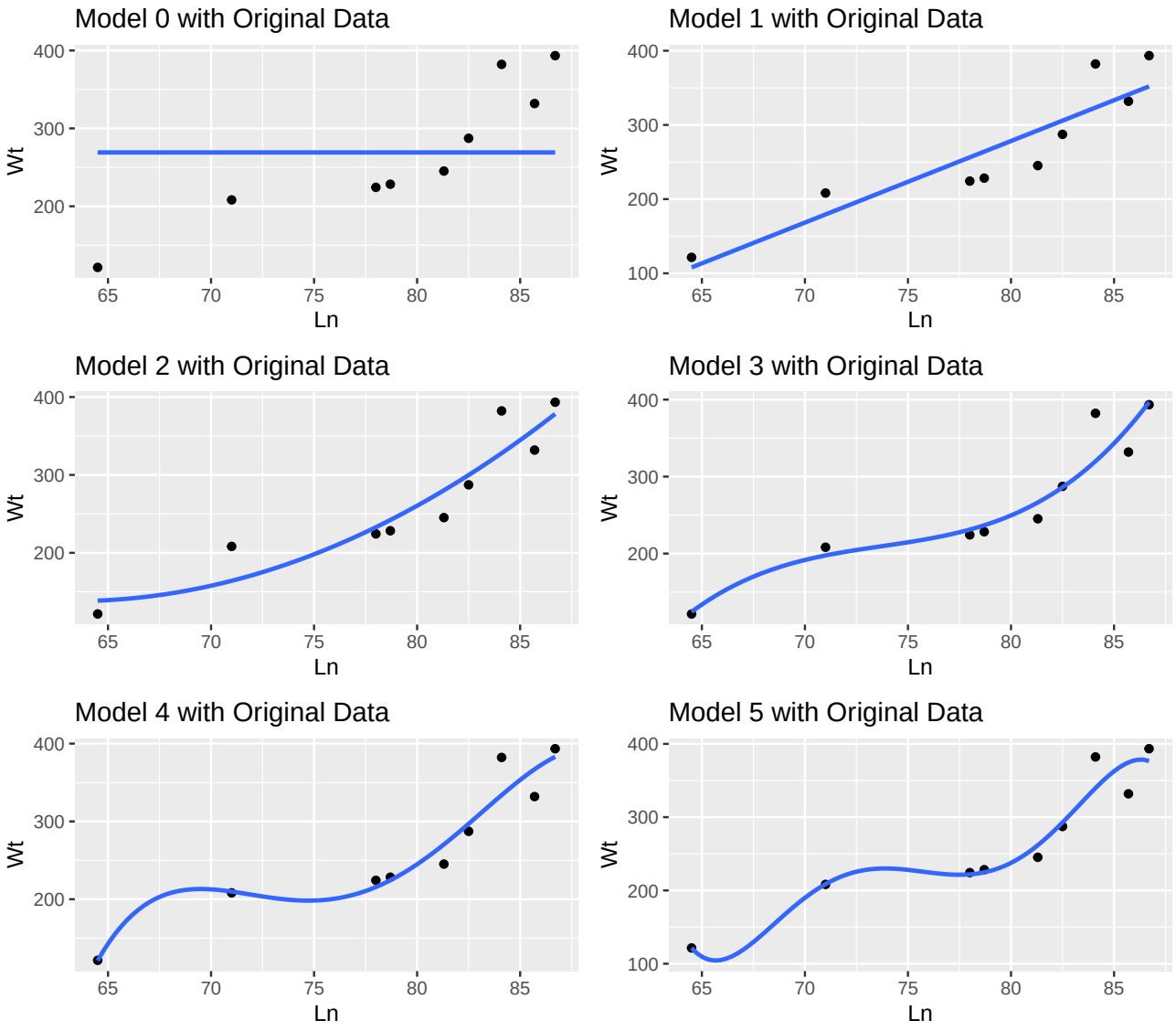


Figure 8

The models fit the **original** data progressively **better** as the model **complexity** gets **higher**, but they **don't necessarily** predict **new** observations better.

In other words, they perform progressively **better** in ***in-sample testing***, but **not necessarily** better in ***out-of-sample testing***.

For example, here are five **new** snakes:

```
newSnakes <- data.frame(Ln = c(67, 72, 77, 81, 86),
                        Wt = c(127.9, 153.7, 204.7, 300.6, 291.4))
newSnakes
```

```
##   Ln    Wt
## 1 67 127.9
## 2 72 153.7
## 3 77 204.7
## 4 81 300.6
## 5 86 291.4
```

Although the **fifth-degree** polynomial predicts the weights of the **original nine** snakes well (**in-sample testing**), it doesn't predict the weights of the **five new** snakes very well (**out-of-sample testing**).

The **linear** model does better in **out-of-sample testing**. See Fig. 9.

```
# Setting alpha = 0.05 makes the original data nearly transparent:
g_new <- ggplot(Snakes, aes(x = Ln, y = Wt)) +
  geom_point(alpha = 0.05)
```

```
# Five new observations with linear model fitted to original data:
g_new + stat_smooth(method = "lm", formula = y ~ poly(x, 1), se = F) +
  ggtitle(label = "Model 1 with New Snakes") +
  geom_point(data = newSnakes)
```

```
# Five new observations with quintic polynomial model fitted to original data:
g_new + stat_smooth(method = "lm", formula = y ~ poly(x, 5), se = F) +
  ggtitle(label = "Model 5 with New Snakes") +
  geom_point(data = newSnakes)
```

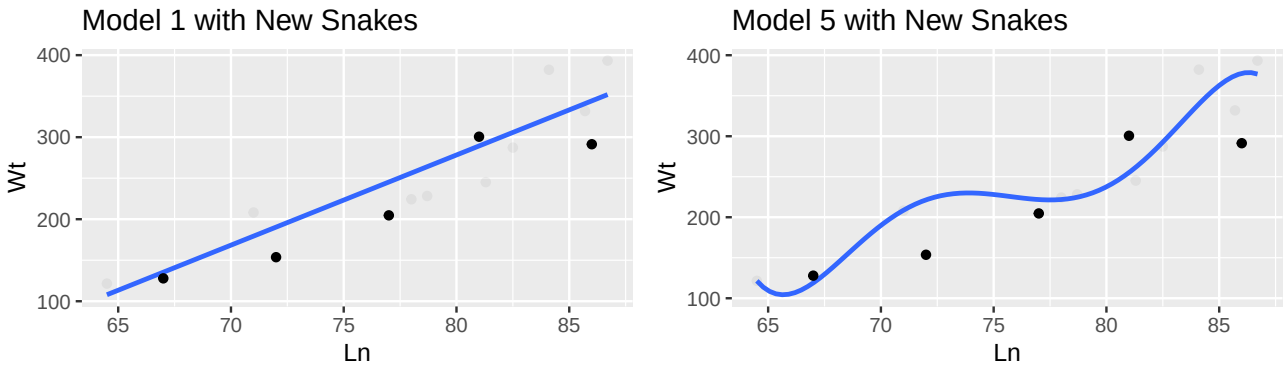


Figure 9

Section 9.8 Exercises

Exercise 14 Recall that in the context of *polynomial regression*, the **model complexity** refers to the number of polynomial terms in the model.

Recall also that **overfitting** occurs when the model complexity is too high.

We can *always* get a polynomial model to fit the data **perfectly** by using enough polynomial terms.

Here are the (original) `snakes` data:

```
Ln <- c(85.7, 64.5, 84.1, 82.5, 78.0, 81.3, 71.0, 86.7, 78.7)
Wt <- c(331.9, 121.5, 382.2, 287.3, 224.3, 245.2, 208.2, 393.4, 228.3)

snakes <- data.frame(Length = Ln, Weight = Wt)
```

Consider an **eighth degree** polynomial regression model for **predicting** *weight* from *length*:

$$\text{Model 8}: \quad Y = \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_4 X^4 + \hat{\beta}_5 X^5 + \hat{\beta}_6 X^6 + \hat{\beta}_7 X^7 + \hat{\beta}_8 X^8$$

where Y is the **weight** and X the **length** of a snake.

```
g <- ggplot(Snakes, aes(x = Ln, y = Wt)) + geom_point()

# Eighth degree polynomial model (Model 8):
g + stat_smooth(method = "lm", formula = y ~ poly(x, 8), se = F) +
  ggtitle(label = "Model 8")
g
```

a) How well does the fitted model **predict** the *weights* of the **original nine** snakes?

b) How well do you think the fitted model would **predict** the *weights* of **five new** snakes?