# Chapter 9
# Linear Regression Models

## 1 Introduction

So far when forming forecasts we have only considered the past history of the variable that we are forecasting. However, it is often the case that we will have more information than this available to us. For example, if we were interested in forecasting total monthly sales at a company, we may also know monthly marketing expenditures, the months that new or improved versions of a particular product launched, the number of competitors operating in the same geographical area each month, etc. In order to take advantage of this extra information, we can use a linear regression model.

When beginning to learn about the linear regression model, we will start with only one "predictor" variable. Then we will consider the case where we have multiple predictor variables. Note that linear regression is the first method that we have encountered that can handle either time series or cross-sectional data; we will consider examples of each. Finally, we will discuss some potential problems related to time series regression, and how to account for these problems.

## 2 Simple Regression

To keep things simple at first, we will consider "simple regression" in which we only have two variables: the *forecast* variable, $Y$, and a *predictor* variable, $X$. We can write the linear regression as:

$$y = b_0 + b_1 x + \varepsilon$$

$$\varepsilon \sim N(0, \sigma)$$

Note that I am not using time subscripts (e.g. $y_t$ and $x_t$), because our data may either be time series or cross-sectional. This model says that our forecast variable, $Y$, is equal to an intercept,

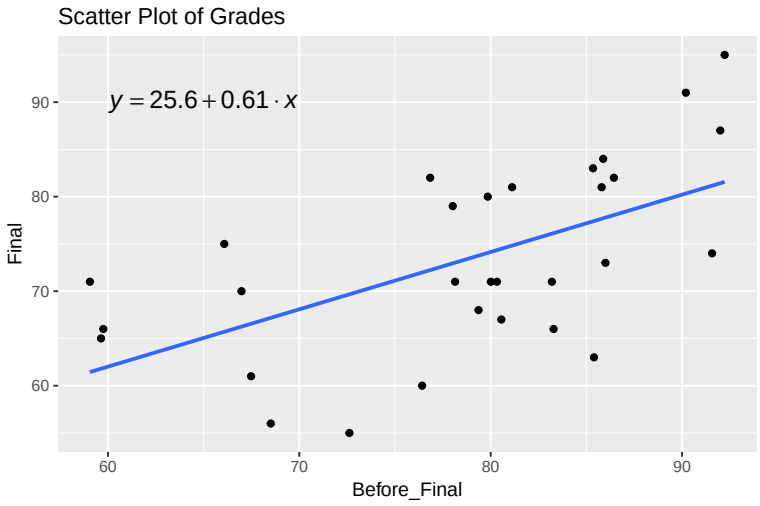
plus or minus some adjustment based on the value of the predictor variable, $X$, plus random error.

#### 2.0.1 Example:

Consider the following example from one section of Macro Principles I taught when I was a graduate student. Let $Y$ be the final exam grade (out of 100 points) and $X$ be the course grade (out of 100 points) on all assignments except for the final exam. After estimating the simple regression, we have:

$$y = 25.6 + 0.61x + \varepsilon$$

$$\varepsilon \sim N(0, 8.2)$$




### 2.1 Interpreting Coefficients

#### 2.1.1 The Intercept, $b_0$

The intercept term, $b_0$, will generally not be the mean of the forecast variable, $Y$. Instead, the mean of the forecast variable can be calculated by taking the unconditional expected value of both sides of the linear regression equation:

$$E(y) = E(b_0 + b_1 x + \varepsilon)$$

$$E(y) = E(b_0) + E(b_1 x) + E(\varepsilon)$$

$$E(y) = b_0 + b_1 E(x) + 0$$

Note that $E(x)$ is the expected value (i.e. the mean) of the predictor variable, $X$, and $E(y)$ is the expected value of the forecast variable, $Y$. Denote the mean of $X$ as $\bar{x}$ and the mean of $Y$ by $\bar{y}$. Then:

$$\begin{aligned} E(y) &= b_0 + b_1 E(x) + 0 \\ E(y) &= b_0 + b_1 \bar{x} \\ \bar{y} &= b_0 + b_1 \bar{x} \end{aligned}$$

We can see that if (and only if) the mean of the predictor variable is zero (i.e $\bar{x} = 0$), then $b_0$ will represent the mean of $Y$. Therefore, some people like to de-mean the predictor variable before using it in the regression, since this will give $b_0$ a "nicer" interpretation (rather than just the "intercept", it becomes the mean of the forecast variable).

### 2.1.2 The Slope, $b_1$

The slope parameter, $b_1$, tells us the *marginal change* in the forecast variable given a one-unit change in the predictor variable. To see this, take the derivative of the simple regression equation with respect to $x$

$$\begin{aligned} y &= b_0 + b_1 x + \varepsilon \\ dy &= b_1 \, dx \end{aligned}$$

So if $x$ increases by "$dx$" units, we would expect $y$ to be $b_1 * dx$ higher. If $x$ was one-unit higher, so that $dx = 1$, then:

$$\begin{aligned} dy &= b_1 \, dx \\ dy &= b_1(1) \\ dy &= b_1 \end{aligned}$$

we would expect $y$ to be $b_1$ units higher.

### 2.1.3 Example:

Consider the following example from the section of Macro Principles considered earlier. Let $Y$ be the final exam grade and $X$ be the course grade on all assignments except for the final.

$$y = 25.6 + 0.61x + \varepsilon$$
$$\varepsilon \sim N(0, 8.2)$$

If the mean of the course grade prior to the final was 78.6 (i.e. $\bar{x} = 78.6$), then we could find the mean of the final exam grade by:

$$\begin{aligned} y &= 25.6 + 0.61x + \varepsilon \\ E(y) &= 25.6 + 0.61\bar{x} \\ E(y) &= 25.6 + 0.61(78.6) \\ E(y) &= 25.6 + 47.9 \\ E(y) &= 73.5 \end{aligned}$$

If, coming into the final exam, Student A has a higher course than Student B by 1 percentage point, we would expect Student A to perform

$$\begin{aligned} dy &= b_1 \, dx \\ dy &= 0.61(1) \\ dy &= 0.61 \end{aligned}$$

points higher on the Final exam than Student B. Note that $b_1$ is not necessarily a *causal* estimate, meaning that we cannot necessarily say that if Student B increased her "before final" score by 1 point that her performance on the Final exam would be expected to increase by 0.61 points. In order to make that claim, we would need to perform an experiment or more sophisticated analysis that can better tackle questions of causality.

## 2.2 Calculating the Coefficients

The simple linear regression coefficients, $b_0$ and $b_1$, are calculated by minimizing the in-sample squared errors. This will give us the "least squares" solution — the values of $b_0$ and $b_1$ that provide the best in-sample fit (the least loss) under squared error loss.

To calculate them, we need to perform a bit of calculus. Let's assume that we are dealing with time series data, as having subscripts to track the individual observations of $y$ and $x$ will be helpful.

$$\min_{b_0,b_1} \sum_{t=1}^{T} \varepsilon_t^2$$

$$\text{where } \varepsilon_t = y_t - b_0 - b_1 x_t$$

Substitute in for $\varepsilon_t$ and take the derivative with respect to $b_0$ and the derivative with respect to $b_1$

$$\min_{b_0,b_1} \sum_{t=1}^{T} (y_t - b_0 - b_1 x_t)^2$$

$$\frac{\partial}{\partial b_0} = \sum_{t=1}^{T} -2(y_t - b_0 - b_1 x_t)$$

$$\frac{\partial}{\partial b_1} = \sum_{t=1}^{T} -2x_t(y_t - b_0 - b_1 x_t)$$

At the minimum, both partial derivatives will equal zero:

$$\frac{\partial}{\partial b_0} = \sum_{t=1}^{T} -2(y_t - b_0 - b_1 x_t) = 0$$

$$\frac{\partial}{\partial b_1} = \sum_{t=1}^{T} -2x_t(y_t - b_0 - b_1 x_t) = 0$$

We have two equations and two unknowns ($b_0$ and $b_1$). If we solved them (and performed some

substitutions), we would find:

$$b_1 = \frac{\sum_{t=1}^{T}(y_t - \bar{y})(x_t - \bar{x})}{\sum_{t=1}^{T}(x_t - \bar{x})^2}$$

$$b_0 = \bar{y} - b_1\bar{x}$$

## 2.3 Forecasting with Simple Regression

Suppose we know the predictor variable, $X$, and want to form a forecast of the forecast variable, $Y$. For instance, maybe we know a student's course grade and we want to forecast their final exam grade. We could use the simple regression equation to find the point forecast:

$$y_i = b_0 + b_1 x_i + \varepsilon$$

$$E(y_i) = E(b_0 + b_1 x_i + \varepsilon)$$

$$E(y_i) = b_0 + b_1 E(x_i)$$

where $y_i$ is the forecast variable for observation $i$ and $x_i$ is the predictor variable for observation $i$. If we know the true value of $x_i$, then we can simply plug it in. If we didn't know the true value, we would have to form an expectation of it somehow — either assume it will be equivalent to the in-sample mean from the series or use some other model to predict it (e.g. maybe previous collge GPA could predict the course grade before taking the final). Assuming we know the true value of $x_i$:

$$E(y_i) = b_0 + b_1 E(x_i)$$

$$E(y_i) = b_0 + b_1 x_i$$

$$\hat{y}_i = b_0 + b_1 x_i$$

### 2.4 Example

Let's return to our final exam model. Suppose we know a student has received a 91 so far in the course. Then, we would forecast that the student would receive a:

$$\hat{y}_i = b_0 + b_1 x_i$$
$$\hat{y}_i = 25.6 + 0.61 x_i$$
$$\hat{y}_i = 25.6 + 0.61(91)$$
$$\hat{y}_i = 25.6 + 55.5$$
$$\hat{y}_i = 81.1$$

on the exam. This is the optimal point forecast under squared error loss. As I will show later in this chapter, we can also use RStudio to find a forecast interval. The 95% forecast interval for this student's final exam grade is $(63.3, 98.4)$. This is a very wide interval (from a D- to an A), so it's a good thing the student will actually take the final exam!

## 3 Multiple Regression

Sometimes we may have more then one predictor variable. If that's the case, we will use multiple linear regression. For all intents & purposes, this is the same as simple regression, just with more predictor variables. We can still find the marginal effect that an increase in one variable has on the forecast variable, and we can still use the multiple regression model to make forecasts.

Suppose we had $k$ predictor variables. Then, we can denote variable $j$ as $x_j$ where $j \in \{1, 2, \ldots, k\}$. We can write the multiple regression equation as:

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_k x_k + \varepsilon$$
$$\varepsilon \sim N(0, \sigma)$$

#### 3.0.1 Example:

Suppose that we were still interested in forecasting the final exam grade. But now, instead of lumping all previous assignments together in one "course grade prior to the final," we will separate out the grades on midterms, HW's, writing assignments, and quizzes. Let:

- $x_1$ be the midterm 1 score for each student.

- $x_2$ be the midterm 2 score for each student.
- $x_3$ be the average HW score for each student.
- $x_4$ be the average WA score for each student.
- $x_5$ be the average quiz score for each student.

Then we have:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_5x_5 + \varepsilon$$
$$y = 3.8 + 0.44x_1 - 0.24x_2 + 0.61x_3 + 0.08x_4 - 0.04x_5 + \varepsilon$$

## 3.1 Interpreting Coefficients

### 3.1.1 The Intercept, $b_0$

The interpretation of the intercept is the same as it was in simple regression. It is the expected value of the forecast variable if all other coefficients were equal to zero. In other words, if someone received a zero on all assignments in the course, we would expect them to get a $b_0 = 3.8$ out of 100 on the Final. Again, this estimate isn't particularly interesting if our predictor variables are not centered around zero (virtually no student would ever get a zero on every single assignment in a class). Therefore, some people like to de-mean all of the predictor variables before fitting the linear regression model. If all of the predictor variables have mean zero, then $b_0$ will be equal to the mean of $Y$.

### 3.1.2 The Slope Coefficients

The slope coefficients have the same interpretation as earlier — each one expresses the marginal change in the forecast variable given a change in the predictor variable. To see this, take the total derivative of the linear regression equation with respect to the predictor variables (let's assume we had 5 predictor variables just do to a concrete example):

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_5x_5 + \varepsilon$$
$$\partial y = b_1\partial x_1 + b_2\partial x_2 + b_3\partial x_3 + b_4\partial x_4 + b_5\partial x_5$$

Suppose that we wanted to investigate the effect of a one unit change variable 1 ($x_1$) on the forecast variable. We set $\partial x_1 = 1$ and all other partial derivatives equal to 0, since none of the

other variables are changing. We have:

$$\partial y = b_1(1) + b_2(0) + b_3(0) + b_4(0) + b_5(0)$$
$$\partial y = b_1$$

So if $x_1$ is one-unit higher, then we would expect $y$ to be $b_1$ units higher.

### 3.1.3 Example

In our multiple regression for the final course grade example, we have:

$$y = 3.8 + 0.44x_1 - 0.24x_2 + 0.61x_3 + 0.08x_4 - 0.04x_5 + \varepsilon$$
$$\varepsilon \sim N(0, 6.7)$$

Recall that $x_1$ is the Midterm 1 grade. If a Student A's Midterm 1 grade was 1 point higher than Student B's, we would expect that their final exam grade would be:

$$\partial y = b_1 = 0.44$$

points higher.

Note that the expected effect of a one point increase in $x_2$, the grade on Midterm 2, is a 0.24 point *decrease* on the final exam. This is certainly not what we expected before seeing the data, as students who do well on midterms usually do better on the final. However, students who did better on Midterm 2 probably also tended to do better on Midterm 1, the HW assignments, etc. So in this case, it may not make sense to employ the "all else equal" standard, as someone who scored one point higher on Midterm 2 likely also had higher scores across the board.

Let's instead suppose that Student A scored 1 point higher then Student B on every assign-

ment. We would expect their final exam grade would be

$$
\begin{aligned}
y &= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_5x_5 + \varepsilon \\
\partial y &= b_1\partial x_1 + b_2\partial x_2 + b_3\partial x_3 + b_4\partial x_4 + b_5\partial x_5 \\
\partial y &= b_1(1) + b_2(1) + b_3(1) + b_4(1) + b_5(1) \\
\partial y &= b_1 + b_2 + b_3 + b_4 + b_5 \\
\partial y &= 0.44 - 0.24 + 0.61 + 0.08 - 0.04 \\
\partial y &= 0.85
\end{aligned}
$$

points higher than Student B's.

The problem of imprecise estimates for coefficients on each predictor variable, illustrated by the negative coefficient on the Midterm 2 grade, is called **multicollinearity**. This occurs when the predictor variables are highly correlated with each other. This can make it very difficult for the least squares algorithm to figure out the effect of each predictor variable independently. For instance, if students tend to perform well (or poorly) on both the first and second midterm, so that in some sense they are measuring one skill instead of two independent skills, it can be very hard for the least squares algorithm to determine the effect of each midterm separately. However, the joint effect will still be estimated fairly precisely, so that if a student scored higher on both Midterm 1 and Midterm 2, we will get a more realistic estimate of how well they should have done on the final exam.

## 3.2 Estimating Coefficients

RStudio will estimate the coefficients to minimize the in-sample sum of squared errors (just as it did under simple regression). With more coefficients, it becomes much harder to express an equation for each coefficient mathematically without using matrices. If you are familiar with matrix algebra, the formula for the coefficients is:

$$
B = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \end{bmatrix} = (X'X)^{-1}X'Y
$$

## 3.3 Forecasting with Multiple Regression

To form a forecast from multiple regression, you need to know (or forecast) the value of all of the predictor variables. For example, if we wanted to use the multiple regression model developed above to forecast a student's final exam score, we would need to know that student's score on Midterms 1&2, the HW, the writing assignments, and the quizzes.

Mathematically, we can take the expected value like we did above in the simple regression section. Let $y_i$ be the forecast variable for observation $i$ and $x_{j,i}$ be the value of predictor $j$ for observation $i$. Then, assuming $k = 5$, we could write:

$$y_i = b_0 + b_1 x_{1,i} + b_2 x_{2,i} + b_3 x_{3,i} + b_4 x_{4,i} + b_5 x_{5,i} + \varepsilon$$
$$E(y_i) = b_0 + b_1 E(x_{1,i}) + b_2 E(x_{2,i}) + b_3 E(x_{3,i}) + b_4 E(x_{4,i}) + b_5 E(x_{5,i}) + 0$$

Again, if we know the true values for all the predictors, we can plug them in instead of taking the expected value. If we don't know the true value, then we must form the expected value somehow. For example, suppose we had a student's scores for Midterm 1, the HW, the writing assignments, and the quizzes, but they missed Midterm 2. We would need to find a way to form the expected value, $E(x_{2,i})$. One simple way would be to use the class average for midterm 2. Another way would be to use the scores from the other students in the class to build a model to predict midterm 2 scores, and substitute the predicted value for that student's midterm 2 grade for the expected value.

Assuming that we know the true value for all of the predictor variables for individual $i$, we could substitute them in and write:

$$E(y_i) = b_0 + b_1 E(x_{1,i}) + b_2 E(x_{2,i}) + b_3 E(x_{3,i}) + b_4 E(x_{4,i}) + b_5 E(x_{5,i})$$
$$E(y_i) = b_0 + b_1 x_{1,i} + b_2 x_{2,i} + b_3 x_{3,i} + b_4 x_{4,i} + b_5 x_{5,i}$$
$$\hat{y}_i = b_0 + b_1 x_{1,i} + b_2 x_{2,i} + b_3 x_{3,i} + b_4 x_{4,i} + b_5 x_{5,i}$$

### 3.3.1 Example

Suppose we had a student with the following grades:

- Midterm 1: 82
- Midterm 2: 90

- HW: 92
- Writing Assignments: 78
- Quizzes: 84

We can forecast her final exam score by using the last equation from the previous section:

$$\hat{y}_i = b_0 + b_1 x_{1,i} + b_2 x_{2,i} + b_3 x_{3,i} + b_4 x_{4,i} + b_5 x_{5,i}$$
$$\hat{y}_i = 3.8 + 0.44 x_{1,i} - 0.24 x_{2,i} + 0.61 x_{3,i} + 0.08 x_{4,i} - 0.04 x_{5,i}$$
$$\hat{y}_i = 3.8 + 0.44(82) - 0.24(90) + 0.61(92) + 0.08(78) - 0.04(84)$$
$$\hat{y}_i = 77.3$$

So our best guess is that she will get a 77.3 on the final exam. We can use RStudio to find a prediction interval. The 95% interval is: (61.6,93.3); i.e., from a D- to an A.

# 4 Time Series Regression Examples

The examples above all used cross-sectional data — data that was collected during one semester across multiple individuals. However, our focus in this course is primarily on time series forecasting. We can also use the linear regression model to forecast time series variables.

## 4.1 Simple Regression: Linear Time Trend

One common simple regression used with time series data is a linear time trend. This assumes that on average, the data increases by exactly the same amount each period. This model estimates both the intercept and the slope of the trend line. We can write the model as:

$$y_t = b_0 + b_1 t + \varepsilon_t$$

Where $t$ is a time index, the same one we have been using throughout the course. However now we are using it directly as a variable, and not just as a subscript. For example, in the first time

period, we have $t = 1$, so we would write:

$$y_t = b_0 + b_1 t + \varepsilon_t$$

$$y_1 = b_0 + b_1(1) + \varepsilon_1$$

$$y_1 = b_0 + b_1 + \varepsilon_1$$

At time $t = 2$:

$$y_2 = b_0 + 2b_1 + \varepsilon_2$$

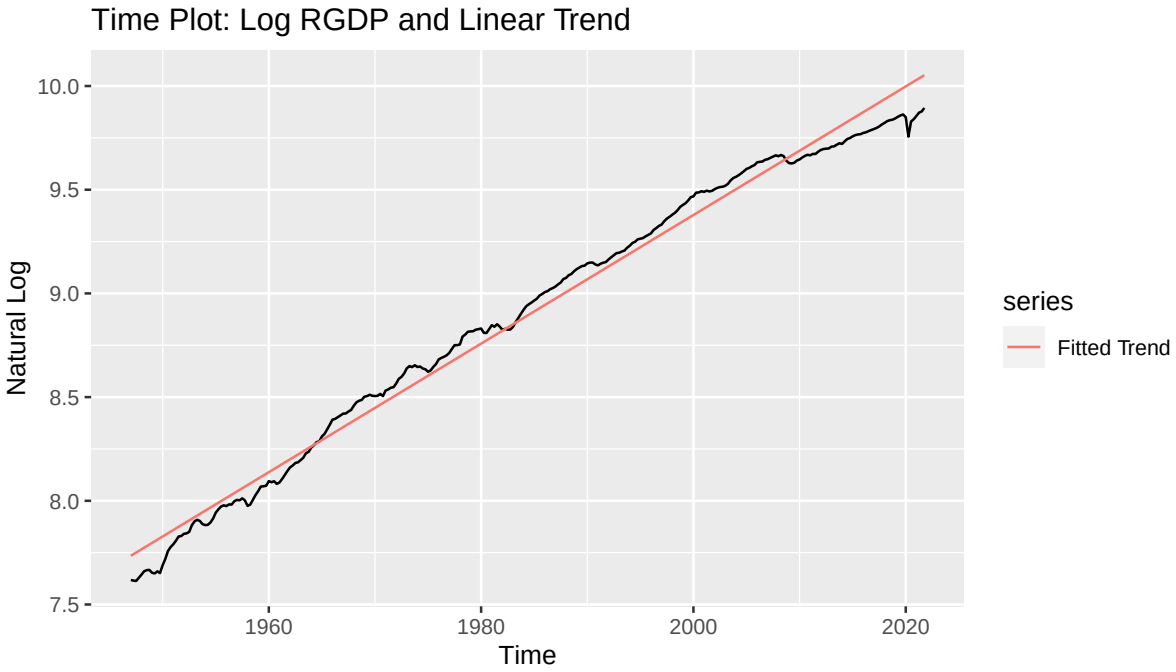
This model is saying that each period, the data will increase by $b_1$ units.

### 4.1.1 Example: RGDP Trend Regression

Suppose we were interested in forecasting Real GDP in the US, measured quarterly. Because the model is a linear trend model and RGDP has exponential growth, we must first take the natural log of RGDP to linearize the trend. Then, using logged RGDP, we can fit the linear time trend. Finally, we could convert back to the original units (typically we would have RStudio do this for us by using the "lambda=0" option).

```
  # Read in Data
  data <- fredr(
  series_id = "GDPC1"
)
  # Declare as Time Series
  Y <- ts(data$value,start=c(1947,1),frequency = 4)
  T <- length(Y)
  # Take log
  lnY <- log(Y)
  # Fit linear trend
  fit <- tslm(lnY ~ 1 + trend)
  # Time Plot
  autoplot(lnY) + autolayer(fit$fitted,series="Fitted Trend") +
    ggtitle("Time Plot: Log RGDP and Linear Trend") +
```

```
  ylab("Natural Log")
```




Using logged, RGDP, we can estimate the simple linear regression equation. We would find:

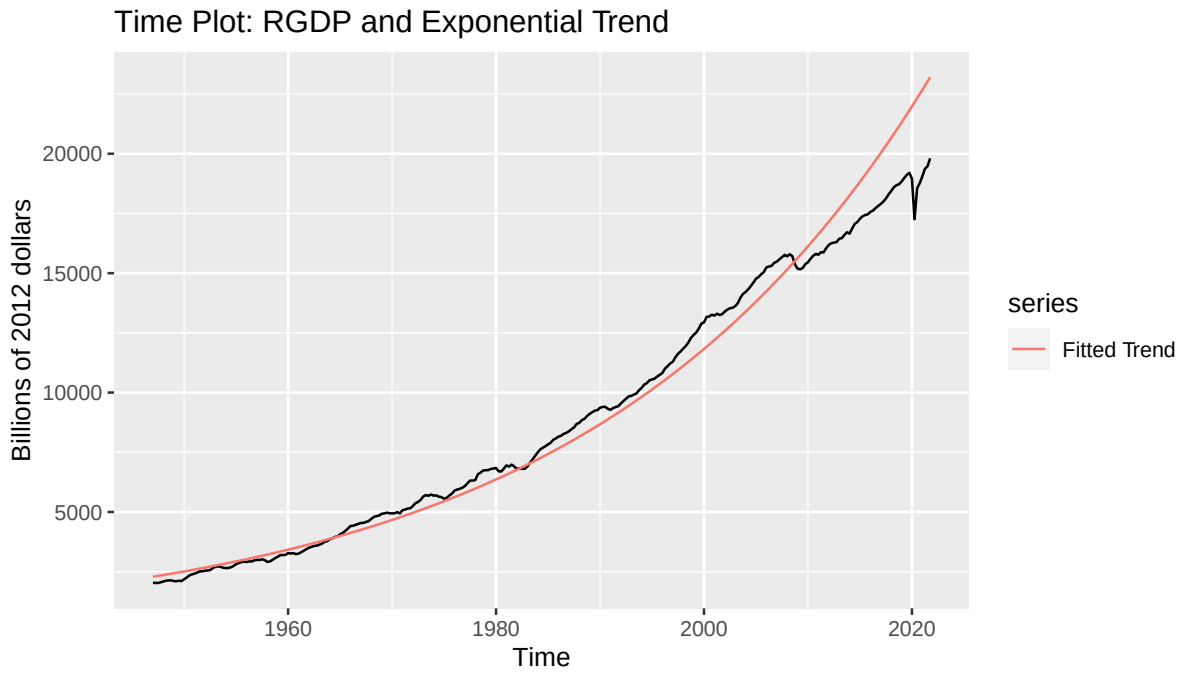
$$y_t = 7.7272 + 0.0078t + \varepsilon_t$$

where $y_t$ is logged RGDP. We can interpret the coefficient estimates as follows. First, the intercept, $b_0 = 7.7272$, is the expected value of the data at time $t = 0$, the time period immediately preceding our first observation. The slope coefficient, $b_1 = 0.0078$, is the expected change each period. In this case, since we have taken the natural log of the data, it is the expected change in the natural log each period. Alternatively, it is roughly equal to the expected *percent* change in the original data each period (in decimal form). In other words, we expect real GDP to grow by $0.0078 = 0.78\%$ each quarter, or roughly $4(0.78\%) \approx 3.1\%$ each year (since there are 4 quarters in a year).

If we convert these estimates back into the original units (billions of 2012 dollars) we would have:

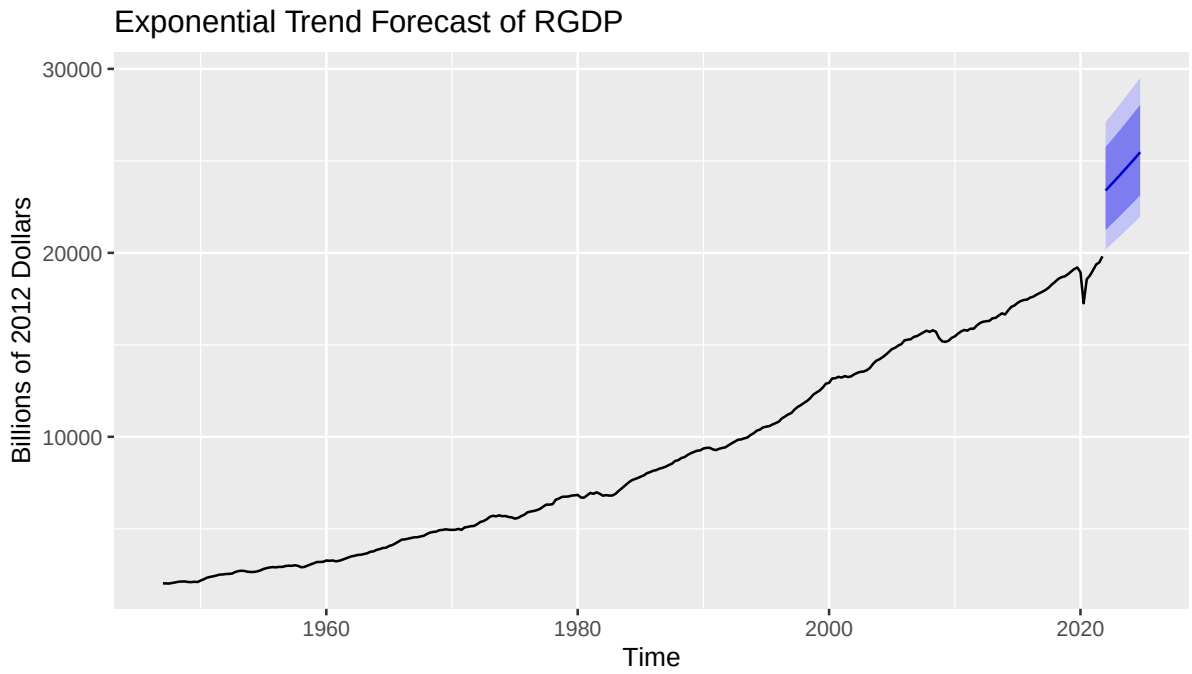
```
fit <- tslm(Y ~ 1 + trend,lambda=0)
autoplot(Y) + autolayer(fit$fitted,series="Fitted Trend") +
  ggtitle("Time Plot: RGDP and Exponential Trend") +
```

```
  ylab("Billions of 2012 dollars")
```




If we were to use this model to forecast RGDP growth, we would have a major problem — there is no requirement that the fitted value pass through the last data point. This can lead to forecasts that expect big declines or big jumps in the series in the first period out-of-sample. In this particular example, this model expects RGDP to rise from roughly \$19.81 trillion in the last in-sample period to roughly \$23.39 trillion in the first-period out-of-sample (an increase of 18.1% in a single quarter) before growing at a steady rate of 0.78% per quarter thereafter.

```
fcst <- forecast(fit,h=12)
autoplot(fcst) + ggtitle("Exponential Trend Forecast of RGDP") +
  ylab("Billions of 2012 Dollars")
```




There are ways around this problem (such as using a cubic spline or allowing for the trend to change over time), but they are fairly complicated and we will not discuss them in detail. Rather, I urge you to **use extreme caution if using the linear trend model to produce forecasts**.

## 4.2 Multiple Regression: Linear Time Trend with Seasonality

Let's consider an extension of the linear time trend model by also allowing for seasonality. Like the standard linear trend model, you should use extreme caution when using this model to form forecasts, as the trend component may suggest forecasts that imply implausibly large jumps or falls in the data one period out-of-sample, as discussed above. Therefore, we largely use this model as an example of the use of **dummy variables**, which will be used to estimate seasonality, and because the in-sample estimates of the trend and seasonal factors may be of interest.

### 4.2.1 A Digression on Dummy Variables

A dummy variable (sometimes just called a "dummy") represents category data and takes a value of 0 or 1. Some non time series examples are: a dummy for a sunny day (1 if the day is sunny, 0 if not), a dummy for a rainy day (1 if the day is rainy, 0 if not), a region dummy (0 if the observation is not in a certain region, 1 if it is in that region), etc. However, note that

we run into a problem if we include a dummy for all categories — perfect multicollinearity. If we have perfect multicollinearity, that means that one of our included variables is completely redundant, and the least squares algorithm will fail.

For example, suppose we were running a regression on ice cream sales, and we wanted to include a dummy for a sunny day and a dummy for a rainy day (for simplicity we will assume all days were either rainy or sunny). Let's pretend we had 5 observations and we wrote our predictor variables (and intercept) in a table or a matrix:

| Intercept | Sunny Dummy ($x_1$) | Rainy Dummy ($x_2$) |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 0 |

Note that if we added columns 2 and 3, then we would have a column of one's — identical to the intercept column. Therefore, we are using three columns to convey only two unique pieces of information. Instead, we should only use two of the three columns. We have two options: (1) delete either the sunny or rainy column, with the deleted column serving as the baseline or (2) delete the intercept column. In practice, option (1) is more common. Suppose we deleted the rainy column. We would have:

| Intercept | Sunny Dummy ($x_1$) |
|---|---|
| 1 | 1 |
| 1 | 0 |
| 1 | 0 |
| 1 | 1 |
| 1 | 1 |

Our regression model would be:

$$y = b_0 + b_1 x_1 + \varepsilon$$

where $y$ is income. Recall that $b_1$ is the marginal effect of a one unit increase in $x_1$. Since $x_1 = 0$ for rainy and $x_1 = 1$ for sunny, $b_1$ represents how much more (or less) is sold on sunny days, on average. For example, if $b_1 = 1,000$ it would mean that, on average, sales are \$1,000 more on sunny days than on rainy days.

### 4.2.2 Seasonal Dummies

In the seasonal time trend model, we will use "seasonal dummies" — a dummy variable for each season (except for one) in our data. In general, we will select one season to serve as the baseline (typically either the first or last season of the year). The coefficient on the dummy will then tell us the effect of that season, relative to the baseline season.

For instance, if we were using quarterly data, we could let quarter 1 (Q1) serve as the baseline, and include dummy variables for $Q2$, $Q3$, and $Q4$. In a table:

| Date | Intercept | Time Trend ($t$) | Q2 Dummy ($x_2$) | Q3 Dummy ($x_3$) | Q4 Dummy ($x_4$) |
|---|---|---|---|---|---|
| 2016 Q1 | 1 | 1 | 0 | 0 | 0 |
| 2016 Q2 | 1 | 2 | 1 | 0 | 0 |
| 2016 Q3 | 1 | 3 | 0 | 1 | 0 |
| 2016 Q4 | 1 | 4 | 0 | 0 | 1 |
| 2017 Q1 | 1 | 5 | 0 | 0 | 0 |
| 2017 Q2 | 1 | 6 | 1 | 0 | 0 |
| 2017 Q3 | 1 | 7 | 0 | 1 | 0 |

The regression model would be:

$$y = b_0 + b_1 t + b_2 x_2 + b_3 x_3 + b_4 x_4 + \varepsilon_t$$

Since these dummies represent specific quarters, we can change notation and write $Q_j$ instead of $x_j$ where appropriate, to remind ourselves that these variables represent dummies for different quarters:

$$y = b_0 + b_1 t + b_2 Q_2 + b_3 Q_3 + b_4 Q_4 + \varepsilon_t$$

$b_1$ will still be the average change per period. Each of $b_2$, $b_3$, and $b_4$ represent the average seasonal factor for their respective quarters relative to $Q1$ (since $Q1$ is the excluded season in this example). For example, if $b_4 = -15$ it would mean that on average, the value of $Y$ in the fourth quarter is 15 below what the value would have been in $Q1$.

### 4.2.3 Strengths and Limitations of Trend and Seasonal Regression

In order to forecast the future with linear regression, we either need to know or be able to forecast the future values of the predictor variables. This can make forecasting with linear regression difficult since it is impossible to know most time-series variables with certainty in

advance. The major benefit of the trend and seasonal regression is that we know the future values of the predictor variables with certainty until the end of time. Time will advance 1 period at a time, so it is trivial to find the value of the time trend variable at any future time period. Similarly, we will know what season we are forecasting for, so it will be trivial to set the seasonal dummies correctly at all future time periods.

However the trend and seasonal regression is not without drawbacks. First, it retains the major problems with trend regression — the trend does not need to go through the last data point, so our forecasts may be unrealistic. Second, the seasonality is very strict — each season will get a seasonal factor that cannot change over time. Therefore, if seasonal patterns have changed over time (e.g. if in the distant past, sales in September used to be above average, but then the economy changed and in more recent years sales in September tend to be below average) then this method will fare poorly. However, in practice, seasonal patterns tend to be very slow to change in most variables, so if you are using a data set that does not go back too far, this will typically not pose too big of a problem.

The trend and seasonal regression model as it is laid out in this chapter is largely for illustrative purposes. Typically, Holt-Winters' method or a seasonal ARIMA method will produce more accurate forecasts than the trend and seasonal method. However, there are ways to generalize this regression model (i.e. to allow for changes in trend and seasonality over time) that can greatly improve its performance. Unfortunately, we will not have time/space in this current chapter to talk about how to do this.

## 4.3 Multiple Regression: Several Time Series Variables

Aside from including a linear time trend and seasonal factors, linear regression allows for the use of additional time series variables as predictors. For example, maybe we are interested in forming a conditional forecast — what can we expect to happen to sales at our company if we increase the marketing budget by $1,000,000? Provided that we have historical data on both sales and the marketing budget, we could use linear regression to try and find an answer. Linear regression is the first method that we have seen that could tackle such a question.

Let's consider a different example. Suppose that we were interested in forecasting the one-period-ahead seasonally adjusted unemployment rate in the US and we think that variables such as industrial production growth, interest rates, and payroll employment growth can help us better predict the unemployment rate. We collect data on all of these series and then use

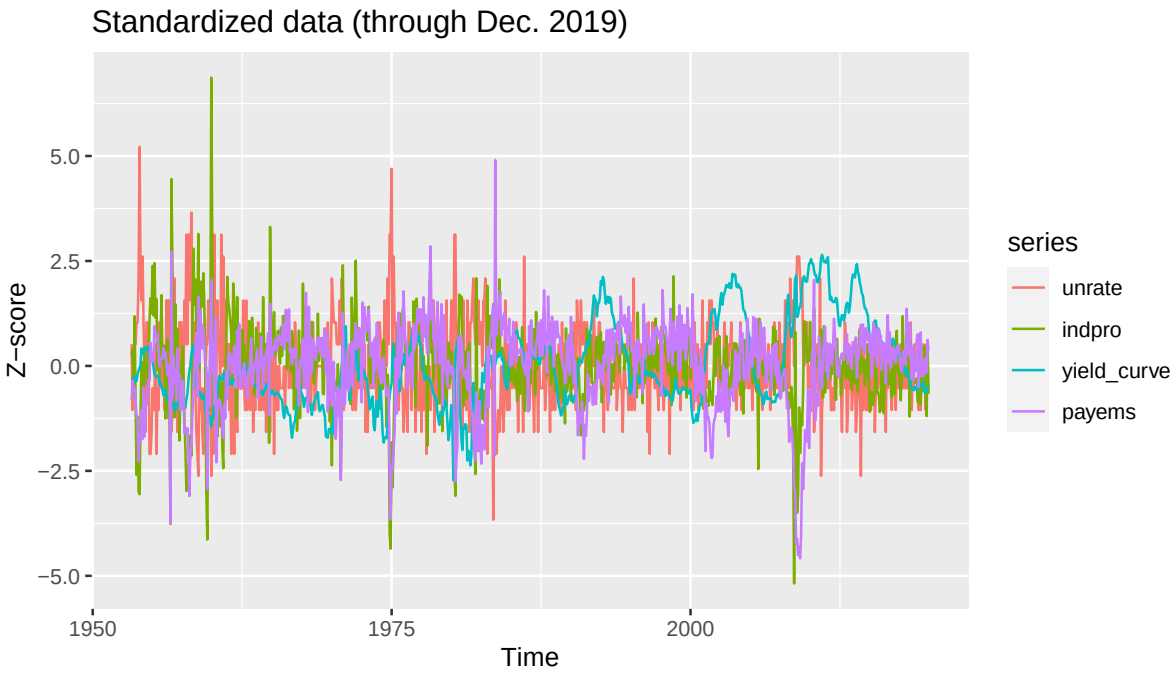
them in the following regression:

$$\Delta y_t = b_0 + b_1 x_{1,t} + b_2 x_{2,t} + b_3 x_{3,t} + \varepsilon_t$$

where:

- $\Delta y_t$ is the change in the unemployment rate in period $t$
- $x_{1,t}$ is the growth rate of industrial production in period $t$.
- $x_{2,t}$ is the interest rate differential between the 10-year and 3-month treasury bonds in period $t$.
- $x_{3,t}$ is the change in payroll employment in period $t$.

Below, I plot the the full sample of data used in the regression model. Due to the COVID pandemic, it is hard to see the typical variation in these series – therefore I also plot the data prior to the pandemic.



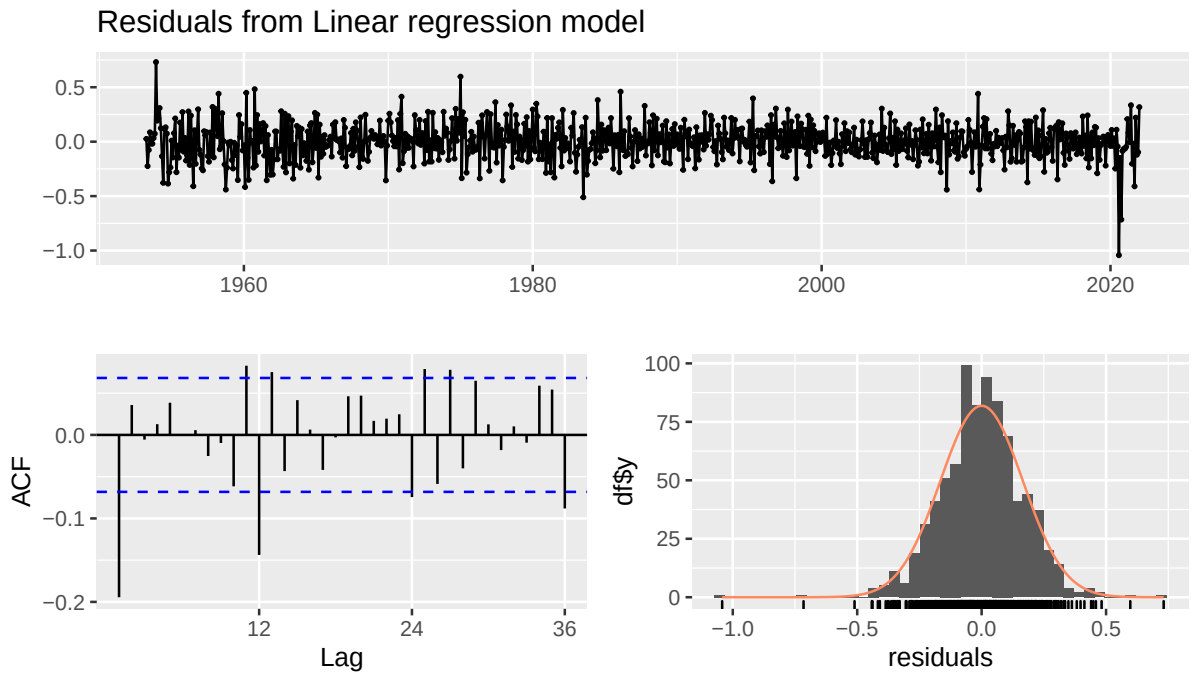

If we run this regression in R we find:

$$\Delta y_t = 0.084 - 0.0425x_{1,t} - 0.0305x_{2,t} - 0.00047x_{3,t} + \varepsilon_t$$

$$\varepsilon_t \sim N(0, 0.1664)$$

Note that even though the coefficients look small, their size is a function of the units we are measuring them in. For instance, we measure the change in payroll employment ($x_3$) in thousands of jobs, so this is saying that an increase of 1,000 jobs will *lower* the unemployment rate by $1 * 0.00047 = 0.00047$. Payroll employment growth typically fluctuates in the tens or hundreds of thousands. An increase of 300,000 jobs would imply an expected *reduction* in the unemployment rate of -0.14 percentage points, which is fairly substantial.

Below, we see that the residuals from this model look decent. Although many of the data sets have outlier values during the pandemic, they roughly cancel out on average (i.e., the increase in the unemployment rate was roughly what was expected given the large job loss). Therefore the residuals during this period look like they only contain one major outlier, at roughly -1, which is around -6.27 standard deviations from the mean. There is some negative autocorrelation at the first lag (and surprisingly, some small but significant negative autocorrelation at the seasonal frequencies), but otherwise the residuals look acceptable.




```
##
##  Breusch-Godfrey test for serial correlation of order up to 24
##
## data:  Residuals from Linear regression model
## LM test = 73.172, df = 24, p-value = 7.162e-07
```

### 4.3.1 Forecasting with Time-Series Regression with Time-Series Predictors

Suppose we wanted to use the above model to forecast the one-month-ahead unemployment rate. If we set $t = T + 1$ and take the expected value, we have:

$$\Delta y_t = b_0 + b_1 x_{1,t} + b_2 x_{2,t} + b_3 x_{3,t} + \varepsilon_t$$

$$\Delta y_{T+1} = b_0 + b_1 x_{1,T+1} + b_2 x_{2,T+1} + b_3 x_{3,T+1} + \varepsilon_{T+1}$$

$$E(\Delta y_{T+1}|Y, X) = b_0 + b_1 E(x_{1,T+1}|Y, X) + b_2 E(x_{2,T+1}|Y, X) + b_3 E(x_{3,T+1}|Y, X) + E(\varepsilon_{T+1}|Y, X)$$

$$E(\Delta y_{T+1}|Y, X) = b_0 + b_1 E(x_{1,T+1}|Y, X) + b_2 E(x_{2,T+1}|Y, X) + b_3 E(x_{3,T+1}|Y, X)$$

So, there is a bit of a problem. In order to forecast future values of the unemployment rate, we need to know future values of each of the predictor variables. Sometimes, for a one-period-ahead forecast, this is actually possible. For example, the estimate of the unemployment rate for November 2021 will be released on December 3, 2021 (it is always released on the first Friday

of the next month). If we wait until the end of November to make the forecast for the November unemployment rate, we would know the average interest rate in November, and would therefore have the true value for $x_{2,T+1}$. However, payroll employment is released at the same time as the unemployment rate, and industrial production is released later in the month. So what can we do?

There are three commonly used strategies to overcome this problem:

1. Forecast values for the predictor variables separately (e.g., using a benchmark, ETS, ARIMA, or separate linear regression model), and plug their forecasts into the equation above.

2. Rewrite the model so that $y_t$ depends on the h$^{\text{th}}$*lag* of the predictor variables (e.g., $x_{t-h}$). Then, the h-period ahead forecast (the forecast for $t = T + h$) will depend on $x_{T+h-h} = x_T$, which is within the sample and can therefore be plugged in directly to the forecast equation. This requires estimating a separate model for each $h$-period ahead forecast you would like to make.

3. Forecast all variables in a combined model in which all variables depend on lags of all the other variables. This is called a Vector Autoregression (VAR) model.

Each potential solution has its own strengths and weaknesses. We will focus on the first solution. In the current example, to keep things as simple as possible, we will use the relevant benchmark method for each predictor (if using an ETS or ARIMA model, we would need to account for the outliers somehow, which makes things unneccesarily complicated for our purposes). The appropriate benchmark model for industrial production and payroll employment is the intercept model, while for the yield curve it is the random walk model. This results in forecasts of 0.21% for industrial production growth ($x_1$), 0.51% for the yield curve ($x_2$), and 120.15 thousand new jobs ($x_3$). Using the estimated model and plugging this values in, we get a forecast of a change in the unemployment rate of:

$$E(\Delta y_{T+1}|Y,X) = b_0 + b_1E(x_{1,T+1}|Y,X) + b_2E(x_{2,T+1}|Y,X) + b_3E(x_{3,T+1}|Y,X)$$

$$E(\Delta y_{T+1}|Y,X) = 0.084 - 0.0425E(x_{1,T+1}|Y,X) - 0.0305E(x_{2,T+1}|Y,X) - 0.00047E(x_{3,T+1}|Y,X)$$

$$E(\Delta y_{T+1}|Y,X) = 0.084 - 0.0425(0.21) - 0.0305(0.51) - 0.00047(120.15)$$

$$E(\Delta y_{T+1}|Y,X) = 0.0032 \text{ percentage points}$$

Note that to actually calculate the value on the last line, I used all decimal places of each estimate and forecast in R, not the rounded values shown above.

We could also use the same method to create "scenario forecasts". For example, if we thought that each of the three predictors would come in at the 90th percentile of their forecast distribution, we would instead have:

$$E(\Delta y_{T+1}|Y,X) = 0.084 - 0.0425(1.54) - 0.0305(0.7) - 0.00047(1112.85)$$
$$E(\Delta y_{T+1}|Y,X) = -0.5253 \text{ percentage points}$$

So we would forecast a reduction in the unemployment rate of about one half percent (e.g., from 4.6% to 4.1%). If, for some reason, we later decided that any of these values was unrealistic, we could change it. For example, expecting payroll employment growth to be over one million jobs might seem overly optimistic. We could plug in a perhaps more realistic value of an increase of 500 thousand jobs:

$$E(\Delta y_{T+1}|Y,X) = 0.084 - 0.0425(1.54) - 0.0305(0.7) - 0.00047(500)$$
$$E(\Delta y_{T+1}|Y,X) = -0.2376 \text{ percentage points}$$

This way of forecasting is not without weaknesses. Probably the most important weakness of this method is that only plugging in point-forecasts for the predictor variables will understate the forecast interval for the forecast variable. For example, suppose that our forecast for payroll employment growth was 500 thousand jobs, with a 95% interval of (-500 thousand, 1.5 million). If we only plug in 500 thousand and pretend that we know it with certainty, our model will not be able to incorporate the uncertainty associated with it. In other words, the model will not be taking into account the fact that we are plugging in an uncertain forecast (e.g., if payroll employment is towards the upper end of its forecast interval, the unemployment rate would shrink by a lot more, and vice-versa). To properly account for this source of uncertainty, you would need to simulate a large number of values of the predictor variables from their forecast distributions. Using each of these simulations (as well as a simulated value of the error term), you would then form a simulated value of the forecast variable. Only after doing a large number of simulations could you calculate the correct 80%, 95%, etc. forecast intervals.

Generally speaking, this is not a trivial source of uncertainty. For example, when only

plugging in the point-forecasts from the intercept method for industrial production, the yield curve, and payroll employment, the 95% interval for the change in the unemployment rate spans $(-0.32, 0.33)$. On the other hand, simulating values from the forecast distributions from the intercept model for industrial production, the yield curve, and payroll employment and plugging those into the regression equation leads to a 95% interval for the change in the unemployment rate that spans $(-0.79, 0.8)$. While it is possible to perform these simulations in RStudio, it adds time and complexity.

## 4.4 Spurious Regression

So far we have covered the basics of time-series regression with time-series predictors. While any type of regression is prone to finding false relationships (see Ch.7 of *The Data Detective* for some examples), it is especially common in time-series regression. The problem is called **spurious regression** and is most likely to occur when the forecast variable and at least one of the predictor variables contains a trend or a unit root.

Spurious regression results in models that appear to fit very well in-sample and forecasts that have very small prediction intervals, but will actually perform very poorly out-of-sample. Some strange examples of variables that have a spurious relationship include (1) the number of people who drowned by falling into a pool and the number of films Nicolas Cage appeared in; and (2) divorce rate in Maine and per-capita consumption of margarine in the U.S.[1]

So how can spurious regression be detected and avoided? After fitting a time-series regression you should always inspect the residuals. If you find that the residuals have a unit root, you should go back and test each data series you are using (both forecast variable and all predictor variables) for a unit root — if you find a unit root in any of the series, you should first-difference that data series before using it in the regression. You can then re-run the regression with the differenced data, and check the residuals again. After differencing the appropriate variables and re-fitting the regression model, the residuals should look independent (or close to it) and should no longer contain a unit root. If the relationship before differencing was spurious, the relationship will break-down after differencing. If there is still a relationship between the forecast and predictor variables after differencing, the the relationship is much more likely to be "real" (i.e. not spurious) and to continue to hold out-of-sample.

[1] For these and others, see: https://www.tylervigen.com/spurious-correlations

# 5 Linear Regression in R

## 5.1 Simple Regression

Let's revisit the simple linear regression where we were forecasting the final exam grades using the course grade up to the final exam as the predictor variable. Assuming the forecast variable was imported as $Y$ and the predictor variable was imported as $X$, we could run the linear regression:

$$Y = b_0 + b_1 X + \varepsilon$$

We can put this data into a data frame, with informative column names

```
dat <- data.frame(Before_Final=X,Final=Y)
```

Next, we use the lm() function to fit the linear regression

```
fit_lm <- lm(Final ~ 1+Before_Final,data=dat)
```

The general rules for the function notation used above are:

- Use the $\sim$ symbol to denote the equal sign from the regression.
- Put the forecast variable to the left of $\sim$
- Put the predictor variables to the right of $\sim$
- Separate the predictor variables with a + sign
- Use "1" to denote that you want to include an intercept in the regression.
- If using column names in a data frame, set data=NameOfDataFrame

Once we have run the regression, we can view some summary statistics and coefficient estimates by using the summary() function and passing it the varaible that contains the regression results:

```
summary(fit_lm)

##
## Call:
```

```
## lm(formula = Final ~ 1 + Before_Final, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.670  -5.711   3.269   6.027  13.437
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   25.6284    12.5127   2.048 0.050022 .
## Before_Final   0.6065     0.1580   3.838 0.000649 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.211 on 28 degrees of freedom
## Multiple R-squared:  0.3447,Adjusted R-squared:  0.3213
## F-statistic: 14.73 on 1 and 28 DF,  p-value: 0.0006486
```

We can also plug in any "Before_Final" grade and see what the regression model would predict the final exam score to be. We could do this one at a time or several at a time. Suppose you were interested in forecasting the Final Exam grades for students who had a 95 and a 72 before taking it.

```
  test <- data.frame(Before_Final=c(95,72))
  forecast(fit_lm,newdata=test)

##   Point Forecast    Lo 80    Hi 80    Lo 95     Hi 95
## 1       83.24308 71.77268 94.71348 65.34169 101.14446
## 2       69.29426 58.25426 80.33427 52.06458  86.52395
```

The forecast() function produces the point forecasts of an 83.2 and a 69.2 for the two students, respectively. It also gives 80% and 95% prediction intervals. Note that to use the forecast() function you must have loaded the "fpp2" package, even if you are using cross-sectional data (as we are here).

## 5.2 Simple Time Trend Regression

If we wanted to use time series data in a regression, we will typically use the function tslm() which has some nice built-in time series features. To use this function, we need time series data. Let's use the RGDP example from earlier:

```
  data <- read.csv("RGDP_full.csv")
  Y <- ts(data[,2],start=c(1947,1),frequency = 4)
  # Since Y is growing exponentially, take the natural log:
  lnY <- log(Y)
  # Run a time trend regression:
  fit_tslm <- tslm(lnY~1+trend)
  # Print regression results:
  summary(fit_tslm)
```

```
##
## Call:
## tslm(formula = lnY ~ 1 + trend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.15796 -0.05214  0.02631  0.05280  0.09996
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.714e+00  7.970e-03   967.9   <2e-16 ***
## trend       7.875e-03  4.748e-05   165.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06769 on 288 degrees of freedom
## Multiple R-squared:  0.9896,Adjusted R-squared:  0.9896
## F-statistic: 2.751e+04 on 1 and 288 DF,  p-value: < 2.2e-16
```

To run the time trend regression, we will use the tslm() function. Note that this function allows us to include a time trend just by using the word "trend" as one of the predictor variables. It would also allow us to include seasonal dummy variables by simply using the word "seasonal" as one of the predictor variables, although we will not be doing that here since RGDP is already seasonally adjusted.

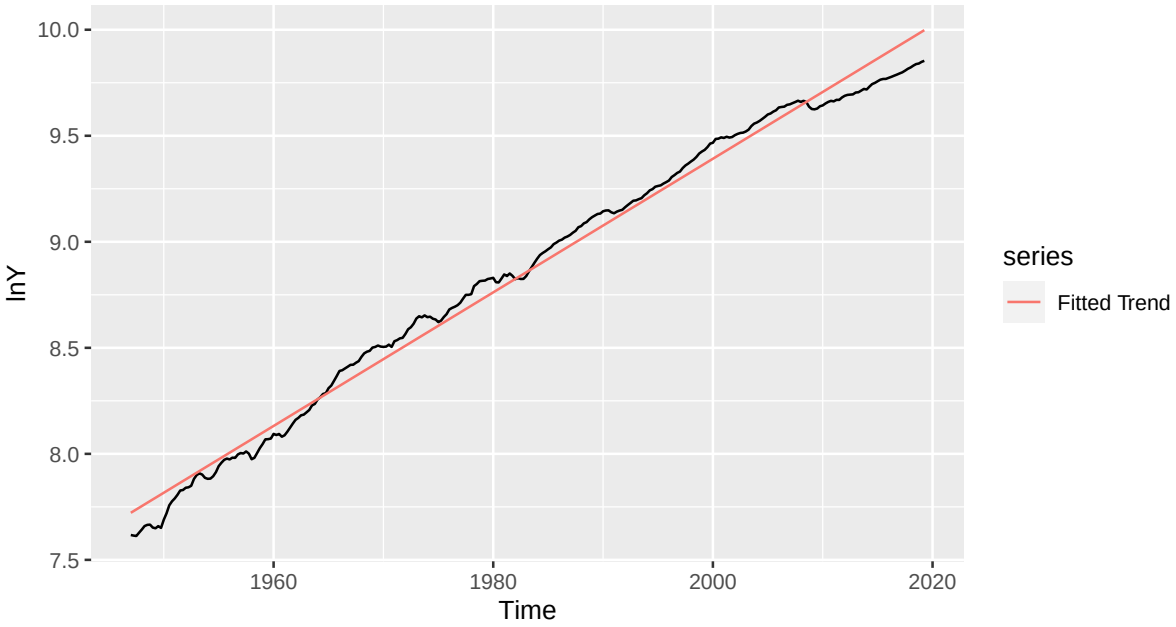
```
  # Run a time trend regression:
  fit_tslm <- tslm(lnY~1+trend)
  # Print regression results:
  summary(fit_tslm)
```

```
##
## Call:
## tslm(formula = lnY ~ 1 + trend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.15796 -0.05214  0.02631  0.05280  0.09996
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.714e+00  7.970e-03   967.9   <2e-16 ***
## trend       7.875e-03  4.748e-05   165.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06769 on 288 degrees of freedom
## Multiple R-squared:  0.9896,Adjusted R-squared:  0.9896
## F-statistic: 2.751e+04 on 1 and 288 DF,  p-value: < 2.2e-16
```

The syntax here is the same — you use the same formula notation that we used with the lm() function, we still separate the predictor variables with "+" signs, and we still use "1" as a predictor variable to make sure that R includes an intercept term in the regression model.

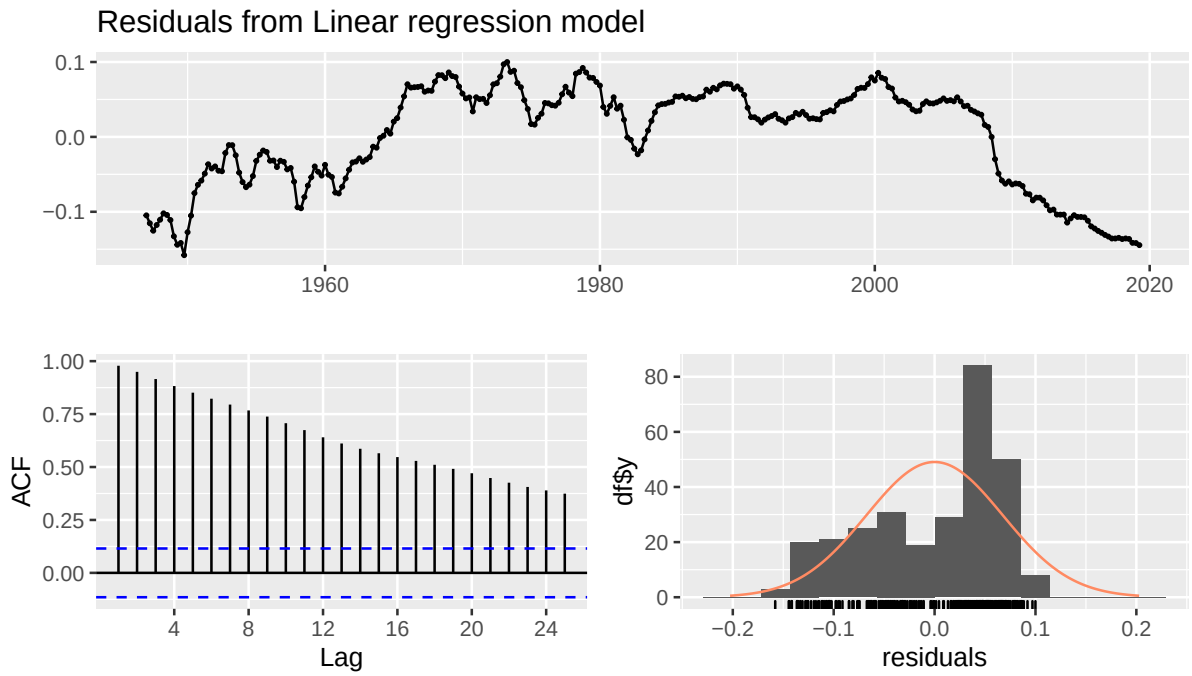
We can plot the time trend and the data using the following code:

```
autoplot(lnY) + autolayer(fit_tslm$fitted,series="Fitted Trend")
```




We could also inspect the residuals from this regression. I will first create a forecast object to make sure the checkresiduals() function uses the Ljung-Box test that we learned about. Note that we can forecast in the usual way, by using the forecast() function and passing it the model fit object.

```
fcst_tslm <- forecast(fit_tslm,h=12)

checkresiduals(fcst_tslm)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Linear regression model
## Q* = 1806.4, df = 6, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 8
```

Note that the residuals do not look very good — this is because the linear time trend is much too simplistic to fit our data. Notice especially the wide gap that opens up between the time trend and the data once the Great Recession hits. The linear time trend seems especially poor at fitting the most recent data.

## 5.3 Multiple Regression

We can also use either the lm() function (if using cross sectional data) or the tslm() function (if using time series data) to perform regression with multiple predictors. Following the cross sectional example from above, let's assume we have loaded in data containing the Midterm 1 and Midterm 2 scores, the HW score, the Quiz score, and the Writing Assignment score. Suppose they were each their own variable, and then put them in a data frame:

```
  dat <- data.frame(Midterm1=midterm1,Midterm2=midterm2,
                    HW=hw,WA=wa,Quiz=quiz,Final=final)
```

Let's see what the first few rows of "dat" look like.

```
  head(dat)
```

```
##   Midterm1 Midterm2   HW   WA Quiz Final
## 1     83.0     75.5 83.3 71.1 84.4    66
## 2     77.5     61.0 79.8 90.8 80.0    71
## 3     74.0     75.5 75.8 87.7 71.1    71
## 4     81.5     78.0 84.7 72.9 53.3    80
## 5     75.0     45.0 68.9 54.6 31.1    71
## 6     68.5     50.0 57.5 69.5 34.4    65
```

We can see that "dat" is essentially a big table with each row corresponding to a student and each column corresponding to one of the variables. To run the multiple regression we use the lm() function, and to see the results use the summary() function

```
  fit_lm <- lm(Final ~ 1+Midterm1+Midterm2+HW+WA+Quiz,data=dat)
  summary(fit_lm)
```

```
##
## Call:
## lm(formula = Final ~ 1 + Midterm1 + Midterm2 + HW + WA + Quiz,
##     data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.7736  -4.6956  -0.3787   3.4780  10.8382
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.80150   13.97978   0.272  0.78800
```

```
## Midterm1     0.44187    0.14018   3.152  0.00431 **
## Midterm2    -0.23922    0.15365  -1.557  0.13258
## HW           0.60894    0.19909   3.059  0.00540 **
## WA           0.08482    0.20147   0.421  0.67751
## Quiz        -0.04396    0.11412  -0.385  0.70349
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.745 on 24 degrees of freedom
## Multiple R-squared:  0.6209,Adjusted R-squared:  0.5419
## F-statistic:  7.86 on 5 and 24 DF,  p-value: 0.0001678
```

We can predict the final exam score for a student if we have that student's midterm, HW, WA, and Quiz grades. For example, let's assume that we had two students:

| Student ID | Midterm 1 | Midterm 2 | HW | WA | Quiz |
|---|---|---|---|---|---|
| 1 | 92 | 78 | 95 | 70 | 85 |
| 2 | 72 | 81 | 80 | 90 | 85 |

We could use their scores on these assignments and the model we built above to predict their final exam score.

```
  new <- data.frame(Midterm1=c(92,72),Midterm2=c(78,81),
                    HW=c(95,80),WA=c(70,90),Quiz=c(85,85))
  forecast(fit_lm,newdata = new)

##   Point Forecast    Lo 80    Hi 80    Lo 95     Hi 95
## 1       85.84451 75.03495 96.65406 68.91537 102.77365
## 2       68.85171 59.21843 78.48499 53.76477  83.93865
```

## 5.4 Time-Series Regression with Time-Series Predictors

Consider the example used in section 4.3 of this chapter, where we used industrial production growth, the slope of the yield curve, and payroll employment growth to forecast the change in the unemployment rate. Assuming the data has already been downloaded from FRED and loaded into RStudio, we could fit the model as shown below:

```r
# Fit linear regression model, assuming variables are saved in a ts data frame
# named "dat" with column names "unrate", "indpro", etc.
fit <- tslm(unrate ~ 1 + indpro + yield_curve + payems,data = dat)
```

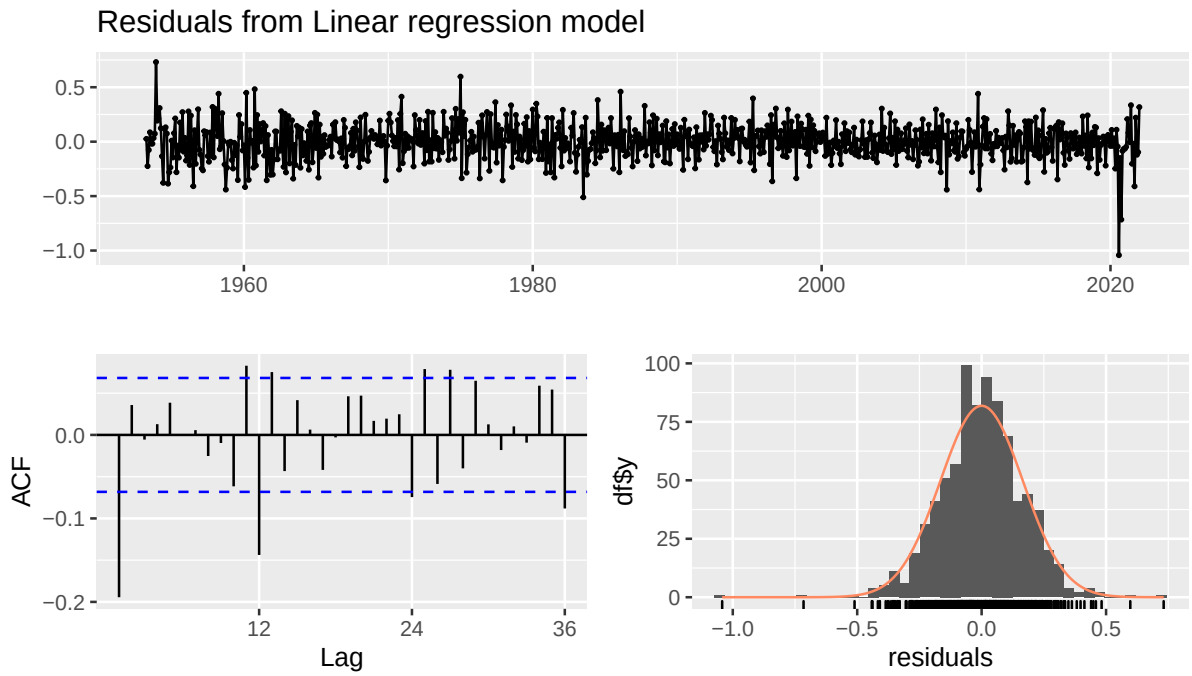
We could print the summary of the model:

```r
print(summary(fit))
```

```
##
## Call:
## tslm(formula = unrate ~ 1 + indpro + yield_curve + payems, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04259 -0.09528  0.00119  0.09863  0.73146
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.398e-02  7.631e-03  11.004  < 2e-16 ***
## indpro      -4.253e-02  7.057e-03  -6.027 2.52e-09 ***
## yield_curve -3.045e-02  8.599e-03  -3.542  0.00042 ***
## payems      -4.694e-04  9.498e-06 -49.419  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1664 on 822 degrees of freedom
## Multiple R-squared:  0.8478,Adjusted R-squared:  0.8473
## F-statistic:  1526 on 3 and 822 DF,  p-value: < 2.2e-16
```

And inspect the residuals:

```r
checkresiduals(fit)
```

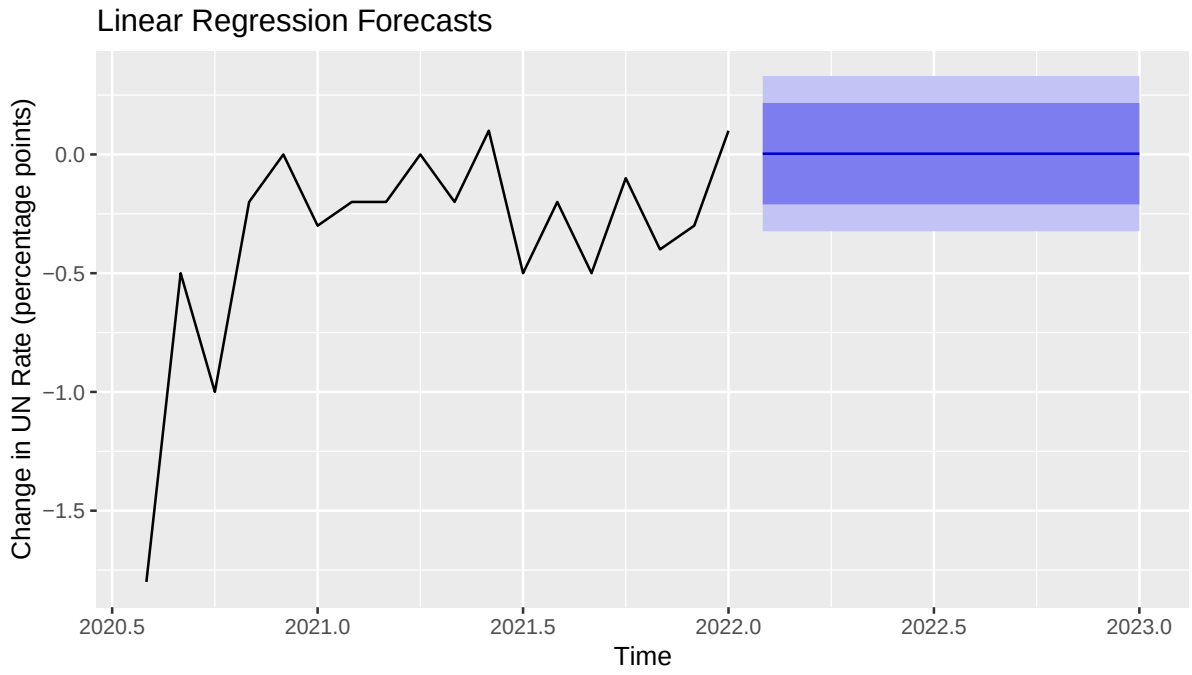



```
##
##  Breusch-Godfrey test for serial correlation of order up to 24
##
## data:  Residuals from Linear regression model
## LM test = 73.172, df = 24, p-value = 7.162e-07
```

To forecast from this model, we need future values of the predictors. As discussed in section 4.3, we could forecast them separately using any of the models we have learned to this point in class. For example, we could forecast each using the appropriate benchmark model:

```
# Set forecast horizon
H <- 12
# Forecast indpro growth
fcst_indpro <- meanf(indpro,h=H)
# Forecast the slope of the yield curve
fcst_yield <- rwf(yield_curve,h=H)
# Forecast the change in payroll employment
fcst_payems <- meanf(payems,h=H)


# Store point-forecasts in a data frame with the same column names as the
```

```r
# data frame holding the original data
fcst_x <- data.frame(indpro=fcst_indpro$mean,
                     yield_curve=fcst_yield$mean,
                     payems=fcst_payems$mean)

# Use these point-forecasts to forecast the change in the unemployment rate
fcst_unrate <- forecast(fit,newdata=fcst_x)

# Plot the forecasts
autoplot(fcst_unrate,include=18) +
  ggtitle("Linear Regression Forecasts") +
  ylab("Change in UN Rate (percentage points)")
```




Recall that the forecasts above understate the uncertainty associated with the unemployment forecast, since they are treating the forecasted predictor values as values that are known with certainty. In reality, these values are not known with certainty, and in fact have a large amount of uncertainty associated with them. To forecast with more accurate prediction intervals, we need to create a large number of simulated forecasts at each horizon. The steps are as follows:

1. At each forecast horizon, $h$, draw a value of each predictor variable from its forecast

distribution, and draw a value of the error term, $\varepsilon_{T+h}$, from its distribution.

2. Plug in the value of each predictor and the error term into the regression equation to create a simulated value of $y_{T+h}$.

3. Do this a large number of times, saving all simulated values of $y_{T+h}$.

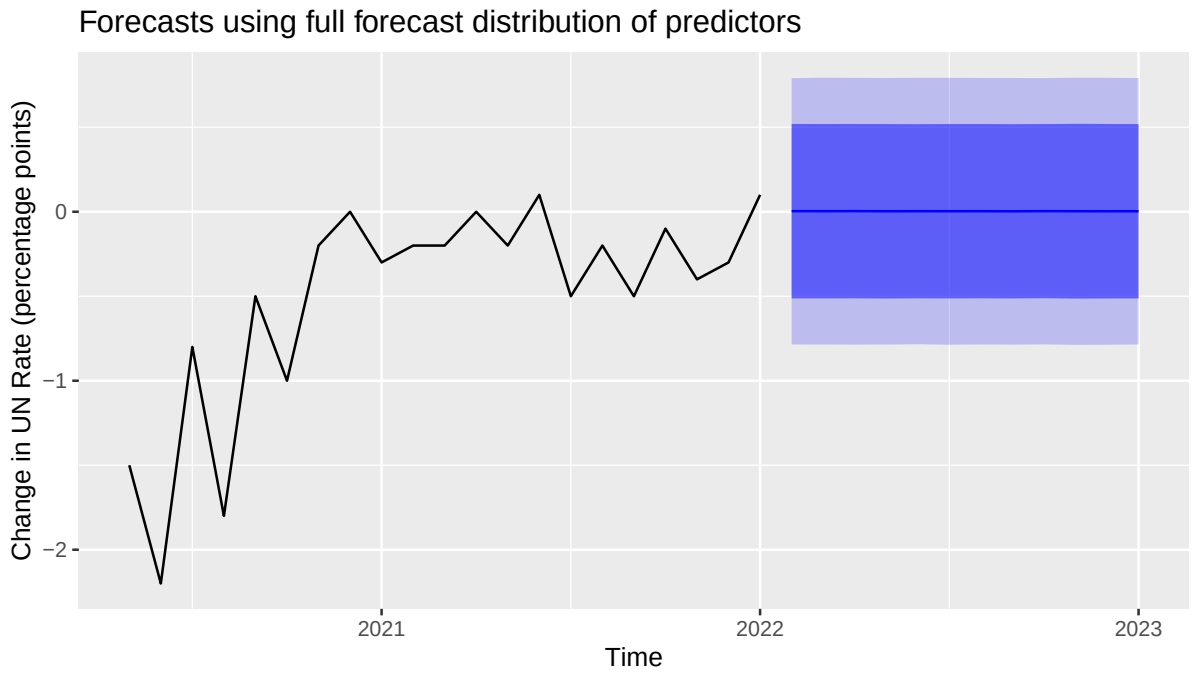
4. Go to the next forecast horizon, $h+1$, and repeat steps 1-3.

5. Once finished, find the relevant 80% and 95% intervals from the simulated values of $y$.

The code below goes through these steps, and then plots the forecasts with the more accurate (and wider) prediction intervals.

```
# Save SD of forecast distribution for each predictor
s_indpro <- (fcst_indpro$upper[,2]-fcst_indpro$mean)/1.96
s_yield <- (fcst_yield$upper[,2]-fcst_yield$mean)/1.96
s_payems <- (fcst_payems$upper[,2]-fcst_payems$mean)/1.96

# Save regression coefficients
b0 <- fit$coefficients[1]
b1 <- fit$coefficients[2]
b2 <- fit$coefficients[3]
b3 <- fit$coefficients[4]

# Number of simulations at each forecast horizon
N <- 1000000
# Empty storage matrix for simulatd forecasts of the change in UN Rate
Yfsto <- matrix(, nrow = H, ncol = N)

for (i in 1:H){
# At each forecast horizon, simulate "N" forecast values of each predictor,
# and draw "N" simulated error terms
indpro_f <- rnorm(N,mean=fcst_indpro$mean[i],sd=s_indpro[i])
yield_f <- rnorm(N,mean=fcst_yield$mean[i],sd=s_yield[i])
```

```
payems_f <- rnorm(N,mean=fcst_payems$mean[i],sd=s_payems[i])
sigma_f <- rnorm(N,mean=0,sd=summary(fit)$sigma)

# Using the estimated regression coefficients, the simulated values of the
# predictors, and the simulated error terms, form simulated forecasts
Yfsto[i,] <- b0 + b1*indpro_f + b2*yield_f + b3*payems_f + sigma_f
}

# Save relevant forecast percentiles for the 80% and 95% prediction intervals
quants <- c(0.025,0.1,0.50,0.90,0.975)
Yf_quants <- apply(Yfsto, 1 , quantile , probs = quants , na.rm = TRUE )

Yf_low80 <- ts(Yf_quants[2,],start=c(last_year,last_month+1),frequency = 12)
Yf_high80 <- ts(Yf_quants[4,],start=c(last_year,last_month+1),frequency = 12)
Yf_low95 <- ts(Yf_quants[1,],start=c(last_year,last_month+1),frequency = 12)
Yf_high95 <- ts(Yf_quants[5,],start=c(last_year,last_month+1),frequency = 12)

# Save the mean of the forecast distribution at each horizon; these are the
# point-forecasts
Yf_mean <- ts(apply(Yfsto, 1 , mean),start=c(last_year,last_month+1),frequency = 12)

# Plot forecasts
autoplot(Yf_mean,color="blue") +
  geom_ribbon(aes(ymin=Yf_low80,ymax=Yf_high80),alpha=0.5,fill="blue") +
  geom_ribbon(aes(ymin=Yf_low95,ymax=Yf_high95),alpha=0.2,fill="blue") +
  autolayer(window(unrate,start=c(2020,5)),color="black") +
  ggtitle("Forecasts using full forecast distribution of predictors") +
  ylab("Change in UN Rate (percentage points)")
```




## 6 Conclusion

In this chapter we have learned about linear regression. This is the first method that allows us to use external data to inform our forecasts. This comes at a cost, however, as the regressors must be known (or forecasted) prior to forming the forecast for the dependent variable. One method where that is not an issue is with a trend and seasonal dummy model, but these models tend to be overly restrictive and forecast relatively poorly.

Finally, linear regression with time-series data can result in spurious relationships. If either the dependent variable or the predictor variables have a trend or unit root, the assumptions of our model break down, and it is possible to find a very high degree of correlation when none should exist. If the residuals from a linear regression using time-series data have high autocorrelation, you need to be extremely cautious about any relationship you find. In this case, the best practice is to first-difference any variable that has a trend or unit root and then to re-fit the linear regression model. If the relationship is not spurious, you should still find a relationship; if it is, the relationship should break-down.