Chapter 10 Dynamic regression models

Ankit Gupta

23/03/2024

The time series models in the previous two chapters allow for the inclusion of information from past observations of a series, but not for the inclusion of other information that may also be relevant. For example, the effects of holidays, competitor activity, changes in the law, the wider economy, or other external variables, may explain some of the historical variation and may lead to more accurate forecasts. On the other hand, the regression models in Chapter 7 allow for the inclusion of a lot of relevant information from predictor variables, but do not allow for the subtle time series dynamics that can be handled with ARIMA models. In this chapter, we consider how to extend ARIMA models in order to allow other information to be included in the models.

In Chapter 7 we considered regression models of the form

$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \varepsilon_t,$

where y_t is a linear function of the k predictor variables $(x_{1,t}, \ldots, x_{k,t})$, and ε_t is usually assumed to be an uncorrelated error term (i.e., it is white noise). We considered tests such as the Ljung-Box test for assessing whether the resulting residuals were significantly correlated.

In this chapter, we will allow the errors from a regression to contain autocorrelation. To emphasise this change in perspective, we will replace ε_t with η_t in the equation. The error series η_t is assumed to follow an ARIMA model. For example, if η_t follows an $ARIMA(1,1,1)$ model, we can write

$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \eta_t,$

$(1 - \phi_1 B)(1 - B)\eta_t = (1 + \theta_1 B)\varepsilon_t$

where ε_t is a white noise series.

Here the second equation specifies the dynamics of regression errors

Notice that the model has two error terms here — the error from the regression model, which we denote by η_t, and the error from the ARIMA model, which we denote by ε_t. Only the ARIMA model errors are assumed to be white noise.

10.1 Estimation

If we minimize $\Sigma \eta_t^2$ by using ordinary regression:

- Estimated coefficients $\hat{\beta}_0, \ldots \hat{\beta}_k$ are no longer optimal as some information ignored.
- Statistical tests associated with the model (e.g. t-tests on the coefficients) are incorrect.
- p-values for coefficients usually too small ("spurious regression")
- AIC of fitted models misleading

Solution

(1) Minimizing $\Sigma \epsilon_t^2$ avoids these problems.

(2) Maximizing likelihood similar to minimizing $\Sigma \epsilon_t^2$ (we will look this solution here)

Stationarity

Regression with ARMA errors

$y_t = \beta_0 + \beta_1 x_{1,t} + \ldots + \beta_k x_{k,t} + \eta_t$

where η_t is an ARMA process

- All variables in the model are *stationary*.
- If we estimate the model while any of these are non-stationary, the estimated coefficients *can be incorrect*.
- *Difference* variables until all stationary.
- If necessary, apply same differencing to all the variables.

Regression with ARIMA errors

Model with ARIMA(1,1,1) errors $y_t = \beta_0 + \beta_1 x_{1,t} + \ldots + \beta_k x_{k,t} + \eta_t$,

$(1 - \phi_1 B)(1 - B)\eta_t = (1 + \theta_1 B)\varepsilon_t$

Equilvalent to Model with ARIMA(1,0,1) errors by differencing variables $y'_t = \beta_0 + \beta_1 x'_{1,t} + \ldots + \beta_k x'_{k,t} + \eta'_t$,

$(1 - \phi_1 B)(1 - B)\eta'_t = (1 + \theta_1 B)\varepsilon_t$

where $y'_t = y_t - y_{t-1}, x'_{t,i} = x_{t,i} - x_{t-1,i}, \eta'_t = \eta_t - \eta_{t-1}$

So, in general, any regression with an ARIMA error can be rewritten as a regression with an ARMA error by differencing all variables with the same differencing operator as in the ARIMA model. So,

Original data

$y_t = \beta_0 + \beta_1 x_{1,t} + \ldots + \beta_k x_{k,t} + \eta_t$,

where $\phi(B)(1 - B)^d \eta_t = \theta(B)\varepsilon_t$

After differencing all variables

$y'_t = \beta_1 x'_{1,t} + \ldots + \beta_k x'_{k,t} + \eta'_t$,

where $\phi(B)\eta'_t = \theta(B)\varepsilon_t, y'_t = (1 - B)^d y_t, x'_{i,t} = (1 - B)^d x_{i,t}, \eta'_t = (1 - B)^d \eta_t$

Theory

When we estimate the parameters from the model, we need to minimise the sum of squared ε_t values. If we minimise the sum of squared η_t values instead (which is what would happen if we estimated the regression model ignoring the autocorrelations in the errors), then several problems arise.

The estimated coefficients $\hat{\beta}_0, \ldots, \hat{\beta}_k$ are no longer the best estimates, as some information has been ignored in the calculation; Any statistical tests associated with the model (e.g., t-tests on the coefficients) will be incorrect. The AICc values of the fitted models are no longer a good guide as to which is the best model for forecasting.

In most cases, the p-values associated with the coefficients will be too small, and so some predictor variables will appear to be important when they are not. This is known as "spurious regression".

Minimising the sum of squared ε_t values avoids these problems. Alternatively, maximum likelihood estimation can be used; this will give similar estimates of the coefficients.

An important consideration when estimating a regression with ARMA errors is that all of the variables in the model must first be stationary. Thus, we first have to check that y_t and all of the predictors $(x_{1,t}, ..., x_{k,t})$ appear to be stationary. If we estimate the model when any of these are non-stationary, the estimated coefficients will not be consistent estimates (and therefore may not be meaningful). One exception to this is the case where non-stationary variables are co-integrated. If there exists a linear combination of the non-stationary y_t and the predictors that is stationary, then the estimated coefficients will be consistent.

We therefore first difference the non-stationary variables in the model. It is often desirable to maintain the form of the relationship between y_t and the predictors, and consequently it is common to difference all of the variables if any of them need differencing. The resulting model is then called a "model in differences", as distinct from a "model in levels", which is what is obtained when the original data are used without differencing.

If all of the variables in the model are stationary, then we only need to consider an ARMA process for the errors. It is easy to see that a regression model with ARIMA errors is equivalent to a regression model in differences with ARMA errors.

10.2 Regression with ARIMA errors using fabel

Regression with ARIMA Errors

- In *fable,* we can specify an ARIMA(p,d,q) for the errors, and d levels of differencing will be applied to all variables $(y, x_{1,t}, ..., x_{k,t})$ during estimation.
- Check that ϵ_t series looks like white noise.
- AICc can be calculated for final model.
- Repeat procedure for all subsets of predictors to be considered, and select model with lowest AICc value like in Chapter 7. AIC for models with different levels of differencing is not comparable.

Example: US Personal Consumption and Income

Here, we are just going to do a simple regression as in Chapter 7. So we regress Consumption on Incomes as

```r
library(fpp3)
```

```
## -- Attaching packages ---------------------------------------------- fpp3 0.5 --

## v tibble      3.2.1     v tsibble     1.1.3
## v dplyr       1.1.3     v tsibbledata 0.4.1
## v tidyr       1.3.0     v feasts      0.3.1
## v lubridate   1.9.3     v fable       0.3.3
## v ggplot2     3.4.4     v fabletools  0.3.4

## -- Conflicts --------------------------------------------------- fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```r
fit <- us_change |> model(ARIMA(Consumption ~ Income))
report(fit)
```

```
## Series: Consumption
## Model: LM w/ ARIMA(1,0,2) errors
##
## Coefficients:
##          ar1      ma1     ma2  Income  intercept
##       0.7070  -0.6172  0.2066  0.1976     0.5949
## s.e.  0.1068   0.1218  0.0741  0.0462     0.0850
##
## sigma^2 estimated as 0.3113:  log likelihood=-163.04
## AIC=338.07   AICc=338.51   BIC=357.8
```

Here, fable chooses the ARIMA(1,0,2) and we can write the equations for fitted model as:

Here, y_t = Consumption and x_t = Income

Now, equations will be:

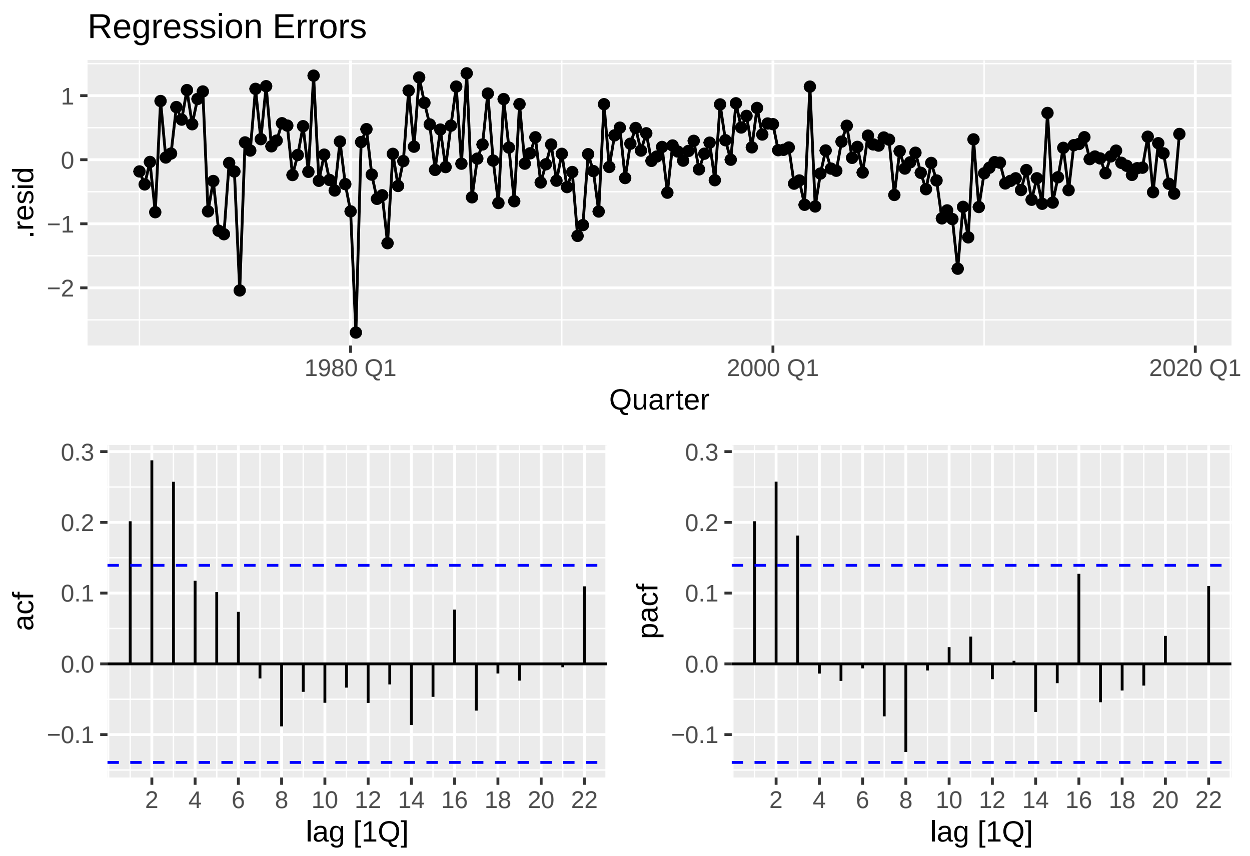
$y_t = 0.5949 + 0.1976x_t + \eta_t$ (Regression Equation)

$(1 - 0.7070B)y_t = (1 - 0.6172B + 0.206B^2)\epsilon_t$

$\epsilon_t \sim (0, 0.3113)$ (Innovation Residuals)

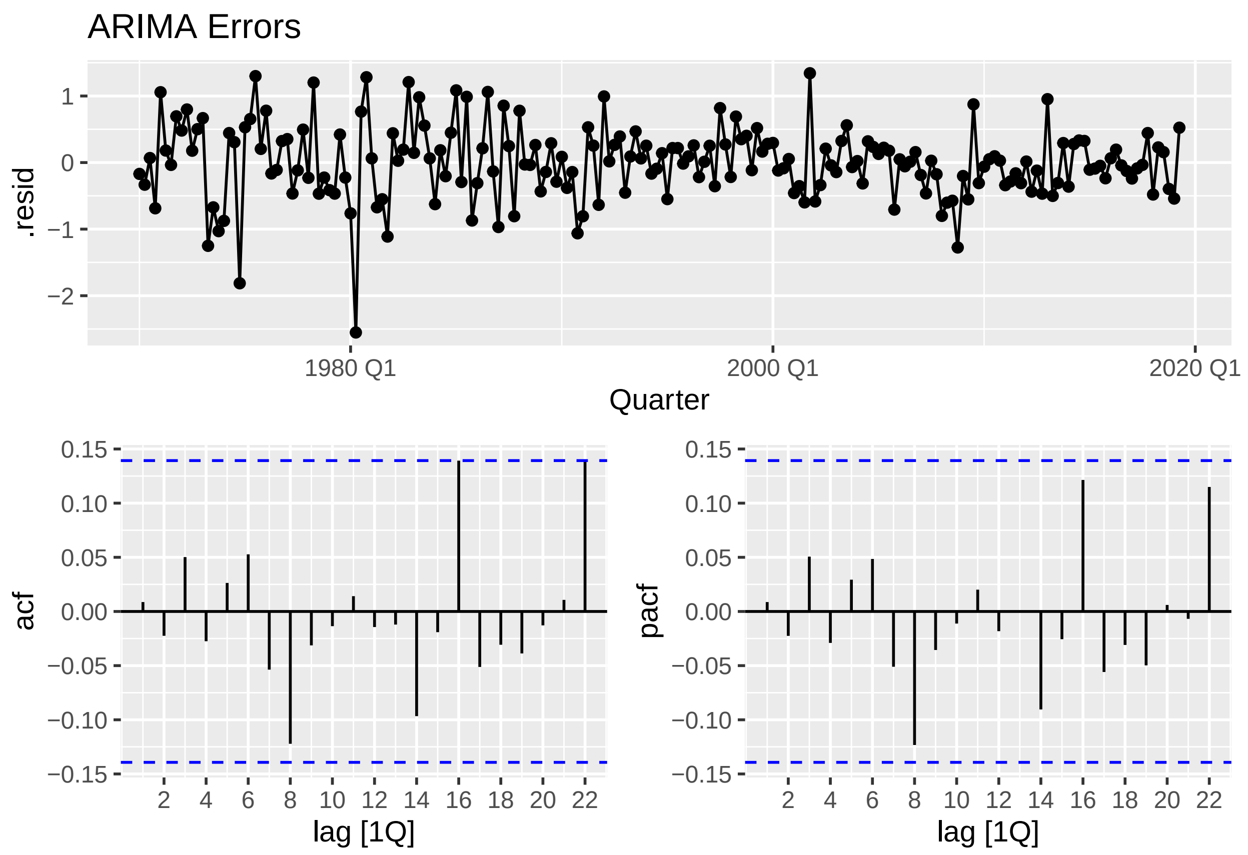
Now, to see the residuals:

```r
residuals(fit, type="regression") |>
  gg_tsdisplay(.resid,plot_type = "partial")+
  labs(title = "Regression Errors")
```



Now, if see the innovation residuals

```
residuals(fit, type="innovation") |>
  gg_tsdisplay(.resid,plot_type = "partial")+
  labs(title = "ARIMA Errors")
```



type=innovation is the default setting. Here, we can see the white noise and we have done a good job while fitting the model.

```
augment(fit) |>
  features(.innov, ljung_box, dof=3, lag=12)
```

```
## # A tibble: 1 x 3
##   .model                        lb_stat lb_pvalue
##   <chr>                           <dbl>     <dbl>
## 1 ARIMA(Consumption ~ Income)      5.54     0.785
```

So, if we do the ljung_box test then p-value clearly does not reject the all of white noise.

Theory

The function *ARIMA()* will fit a regression model with ARIMA errors if exogenous regressors are included in the formula. As introduced in Section 9.5, the pdq() special specifies the order of the ARIMA error model. If differencing is specified, then the differencing is applied to all variables in the regression model before the model is estimated. For example, the command

```
ARIMA(y ~ x + pdq(1,1,0))
```

```
## <ARIMA model definition>
```

will fit the model $y'_t = \beta_1 x'_t + \eta'_t$, where $\eta'_t = \phi_1 \eta'_{t-1} + \varepsilon_t$ is an $AR(1)$ error. This is equivalent to the model:

$y_t = \beta_0 + \beta_1 x_t + \eta_t$,

where η_t is an ARIMA(1,1,0) error. Notice that the constant term disappears due to the differencing. To include a constant in the differenced model, we would add 1 to the model formula.

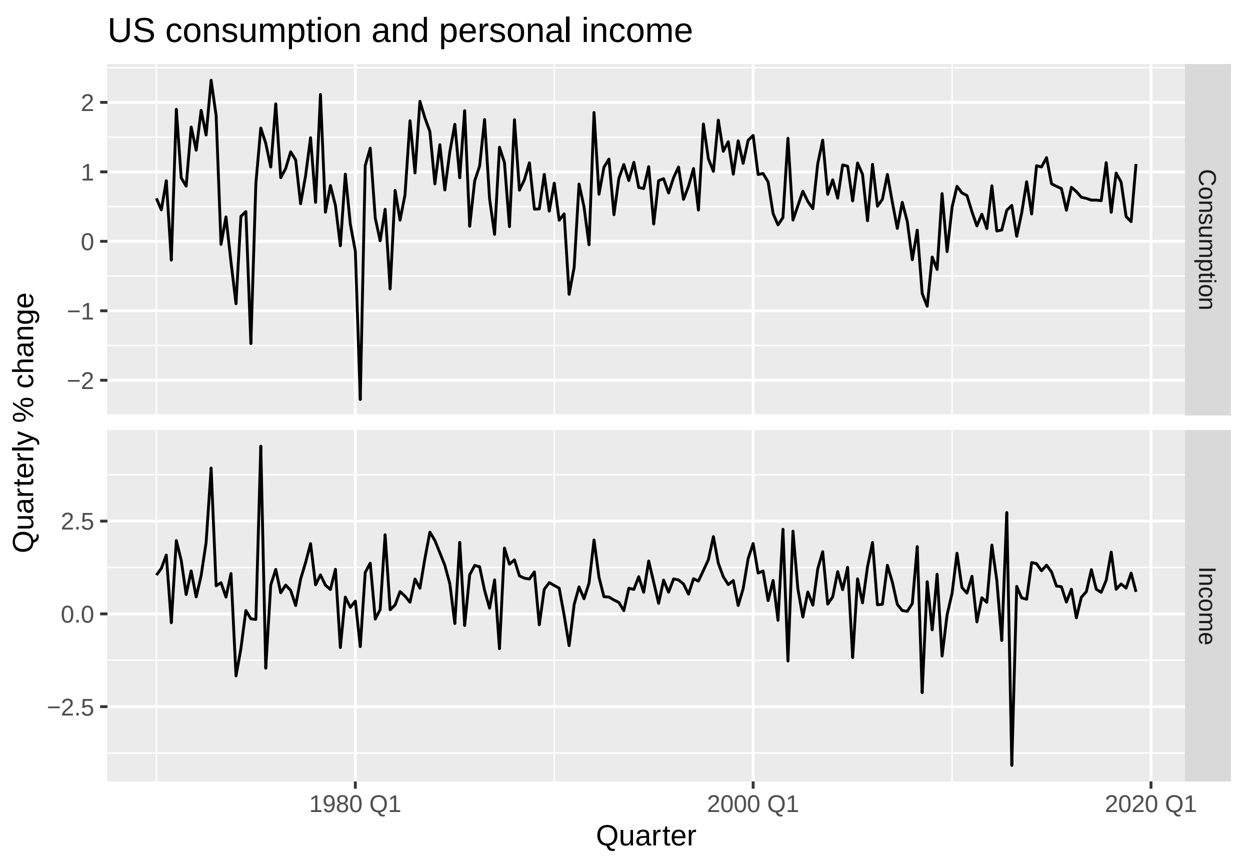
The ARIMA() function can also be used to select the best ARIMA model for the errors. This is done by not specifying the pdq() special. Whether differencing is required is determined by applying a KPSS test to the residuals from the regression model estimated using ordinary least squares. If differencing is required, then all variables are differenced and the model re-estimated using maximum likelihood estimation. The final model will be expressed in terms of the original variables, even if it has been estimated using differenced variables.

The AICc is calculated for the final model, and this value can be used to determine the best predictors. That is, the procedure should be repeated for all subsets of predictors to be considered, and the model with the lowest AICc value selected.

Example: US Personal Consumption and Income

Figure 10.1 shows the quarterly changes in personal consumption expenditure and personal disposable income from 1970 to 2019 Q2. We would like to forecast changes in expenditure based on changes in income. A change in income does not necessarily translate to an instant change in consumption (e.g., after the loss of a job, it may take a few months for expenses to be reduced to allow for the new circumstances). However, we will ignore this complexity in this example and try to measure the instantaneous effect of the average change of income on the average change of consumption expenditure.

```r
us_change |>
  pivot_longer(c(Consumption, Income),
               names_to = "var", values_to = "value") |>
  ggplot(aes(x = Quarter, y = value)) +
  geom_line() +
  facet_grid(vars(var), scales = "free_y") +
  labs(title = "US consumption and personal income",
       y = "Quarterly % change")
```



```r
fit <- us_change |>
  model(ARIMA(Consumption ~ Income))
report(fit)
```

```
## Series: Consumption
## Model: LM w/ ARIMA(1,0,2) errors
##
## Coefficients:
##          ar1      ma1     ma2  Income  intercept
##       0.7070  -0.6172  0.2066  0.1976     0.5949
## s.e.  0.1068   0.1218  0.0741  0.0462     0.0850
##
## sigma^2 estimated as 0.3113:  log likelihood=-163.04
## AIC=338.07   AICc=338.51   BIC=357.8
```

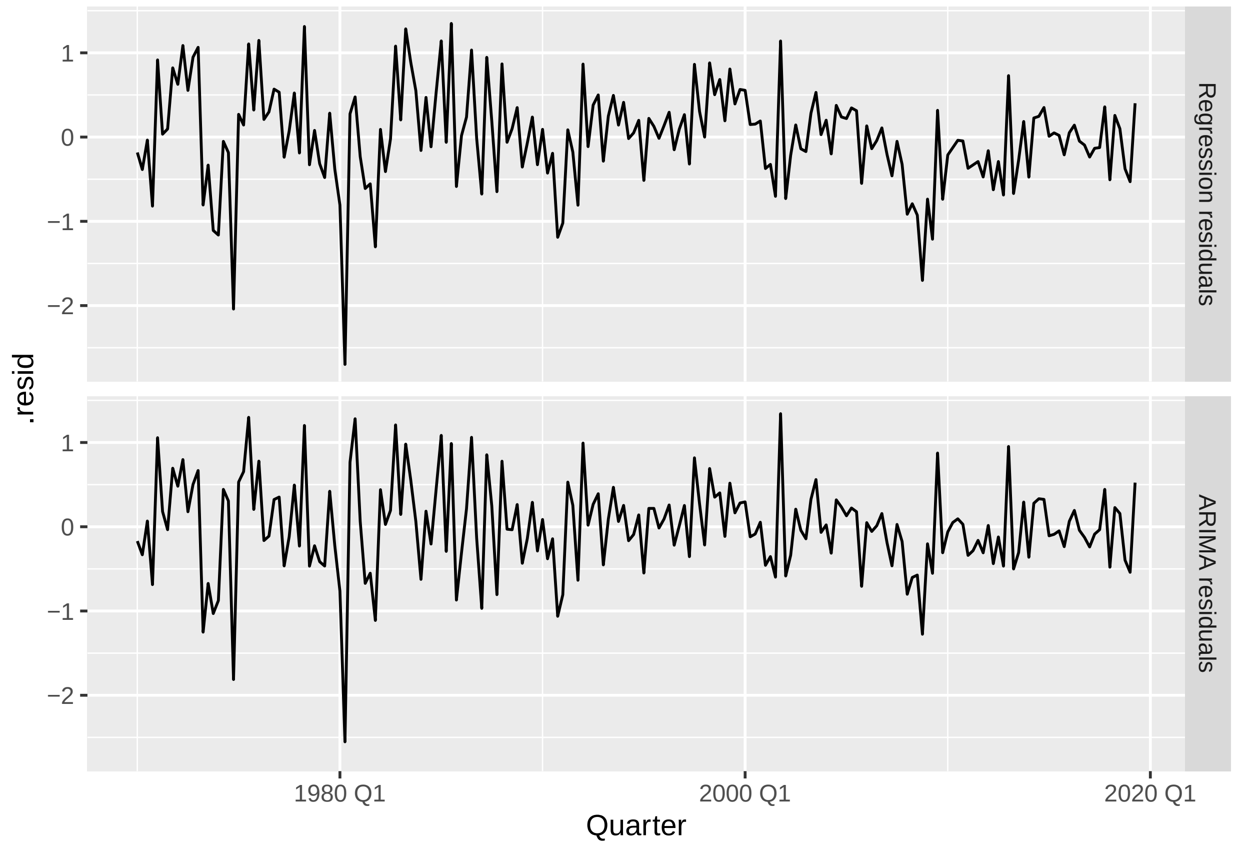
The data are clearly already stationary (as we are considering percentage changes rather than raw expenditure and income), so there is no need for any differencing. The fitted model is

$y_t = 0.595 + 0.198x_t + \eta_t,$

$\eta_t = 0.707\eta_{t-1} + \varepsilon_t - 0.617\varepsilon_{t-1} + 0.207\varepsilon_{t-2}, \varepsilon_t \sim NID(0, 0.311).$

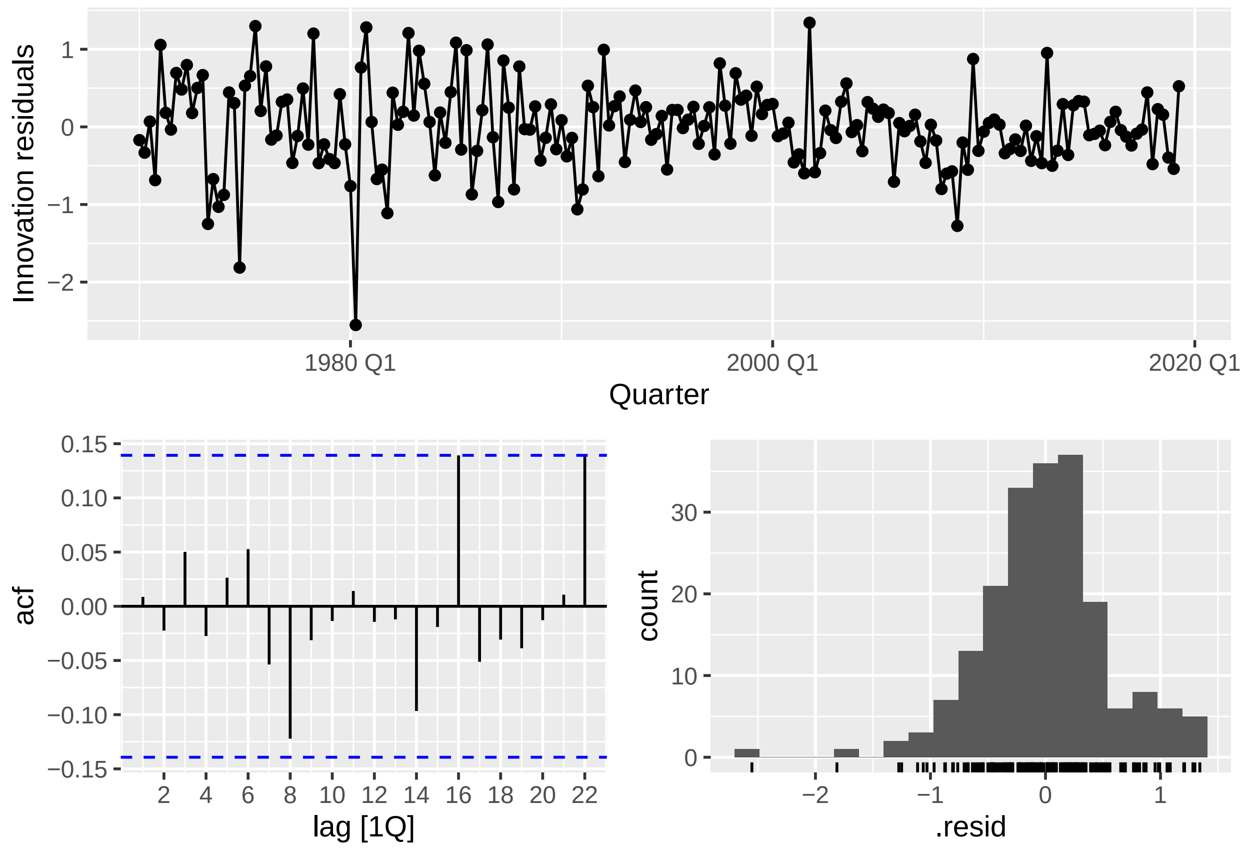
We can recover estimates of both the η_t and ε_t series using the residuals() function

```r
bind_rows(
    `Regression residuals` =
        as_tibble(residuals(fit, type = "regression")),
    `ARIMA residuals` =
        as_tibble(residuals(fit, type = "innovation")),
    .id = "type"
  ) |>
  mutate(
    type = factor(type, levels=c(
      "Regression residuals", "ARIMA residuals"))
  ) |>
  ggplot(aes(x = Quarter, y = .resid)) +
  geom_line() +
  facet_grid(vars(type))
```



It is the ARIMA estimated errors (the innovation residuals) that should resemble a white noise series.

```
fit |> gg_tsresiduals()
```



```r
augment(fit) |>
  features(.innov, ljung_box, dof = 3, lag = 8)
```

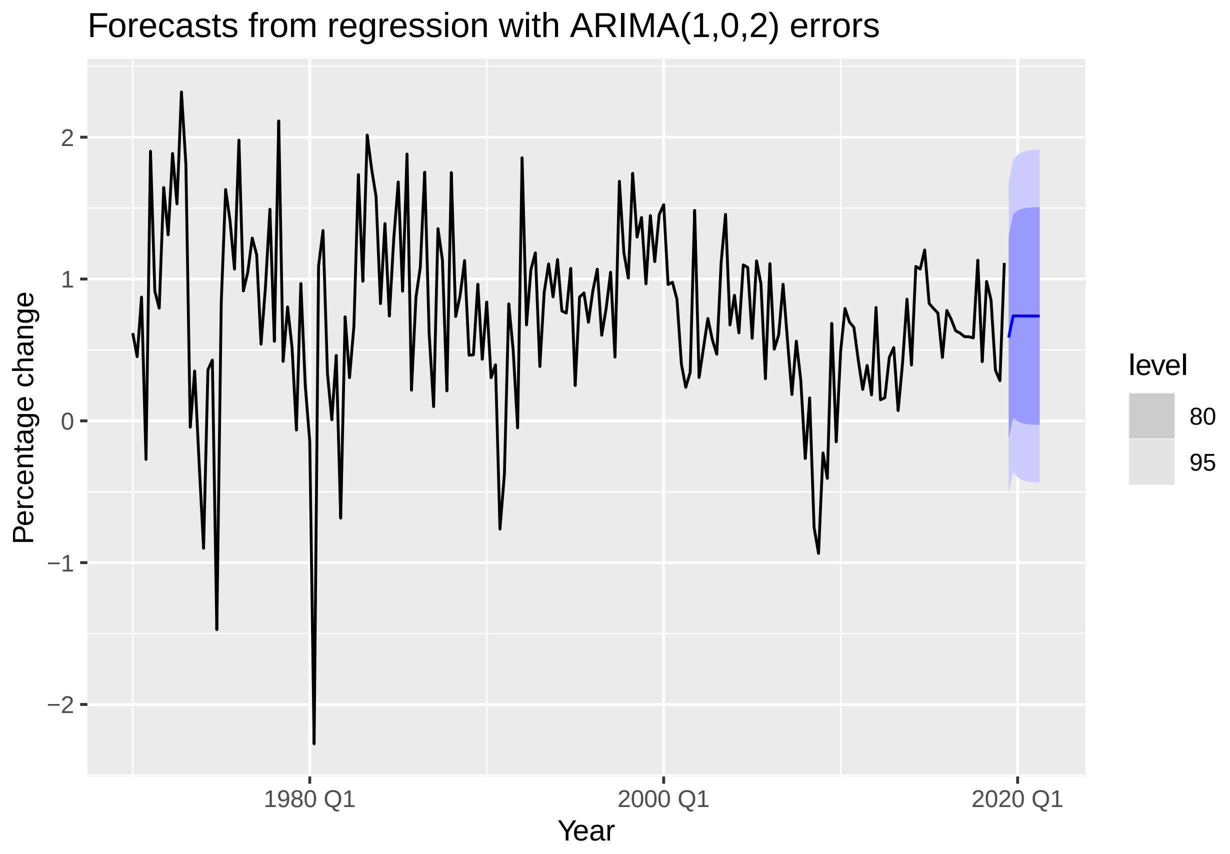
```
## # A tibble: 1 x 3
##   .model                          lb_stat lb_pvalue
##   <chr>                             <dbl>     <dbl>
## 1 ARIMA(Consumption ~ Income)        5.21     0.391
```

Forecasting

- To forecast a regression model with ARIMA errors, we need to forecast the regression part of the model and the ARIMA part of the model combine the results.
- Some predictors are known into the future (e.g. time, dummies).
- Separate forecasting models may be needed for other predictors.
- Forecast intervals ignore the uncertainty in forecasting the predictors.

Example

```r
fit <- us_change |> model(ARIMA(Consumption ~ Income))
us_change_future <- new_data(us_change,8) |>
  mutate(Income=mean(us_change$Income))
forecast(fit,new_data = us_change_future) |>
  autoplot(us_change) +
  labs(x="Year",y="Percentage change",
       title = "Forecasts from regression with ARIMA(1,0,2) errors")
```

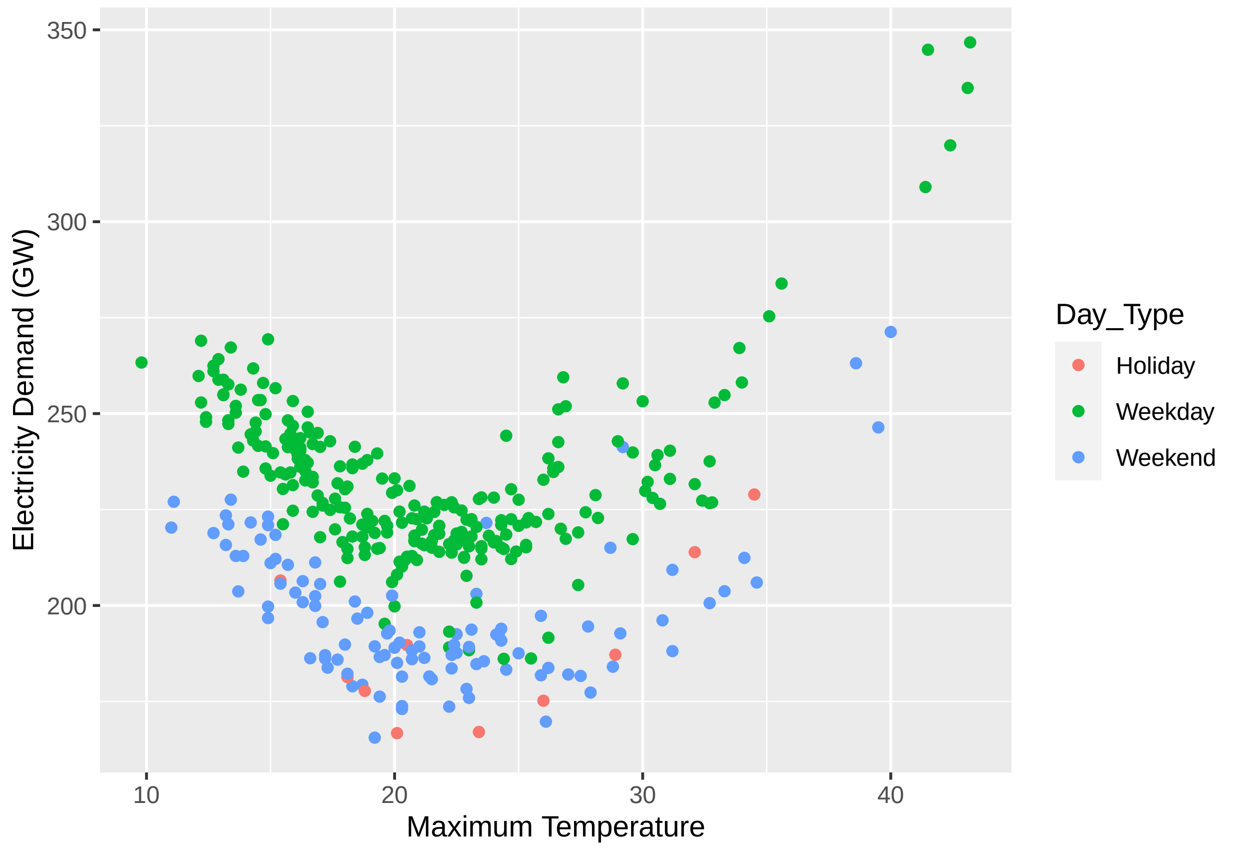


Here, we are going to generate forecast for next 8 steps and here forecast function will generate some forecasts for us. We assume here the historical average of income of us change. Hence the forecast we get for the next 8 quarters look quite reasonable. Here, in practice, prediction interval need to probably be wider as they take the values for the income for the future as fixed as given.

Example

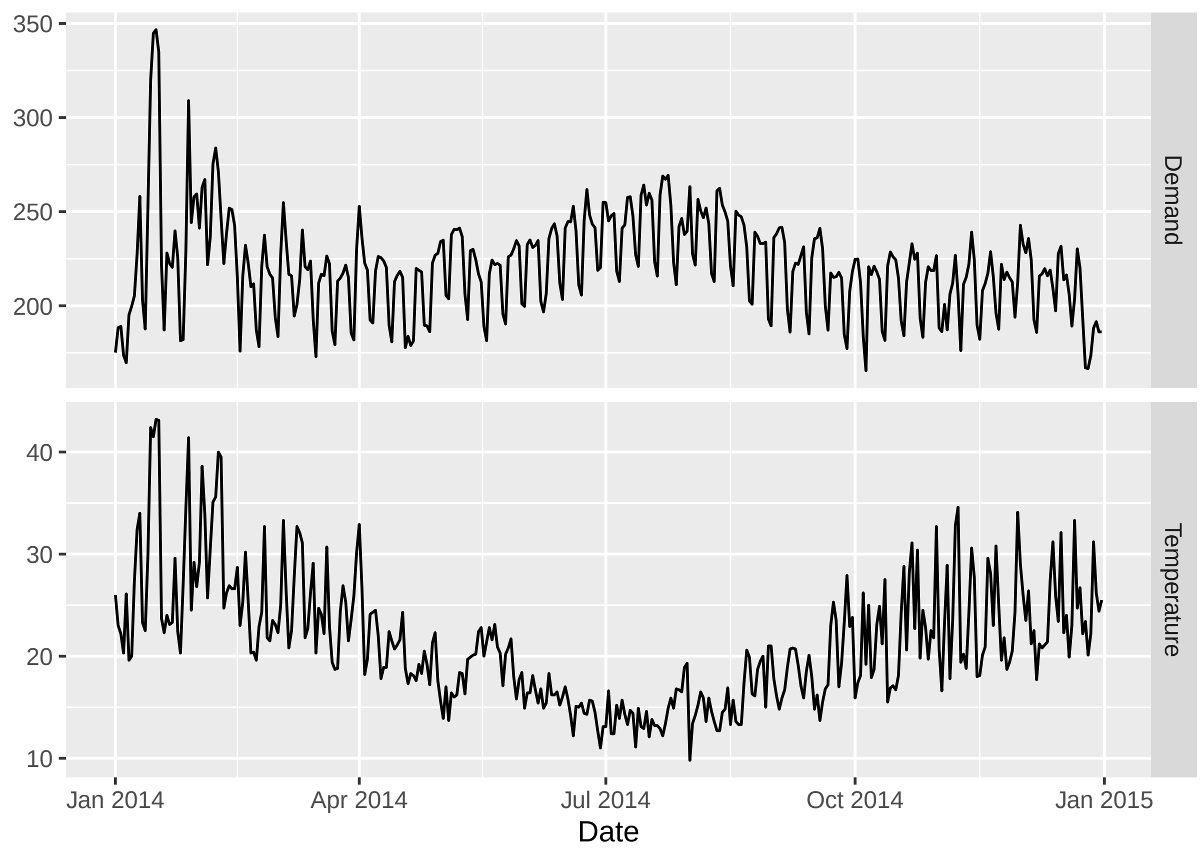
```
vic_elec_daily <- vic_elec |>
  filter(year(Time) == 2014) |>
  index_by(Date = date(Time)) |>
  summarise(
    Demand = sum(Demand) / 1e3,
    Temperature = max(Temperature),
    Holiday = any(Holiday)
  ) |>
  mutate(Day_Type = case_when(
    Holiday ~ "Holiday",
    wday(Date) %in% 2:6 ~ "Weekday",
    TRUE ~ "Weekend"
  ))

vic_elec_daily |>
  ggplot(aes(x=Temperature,y=Demand,colour=Day_Type))+
  geom_point()+
  labs(x="Maximum Temperature", y = "Electricity Demand (GW)")
```



Here, we can see that there is a non-linear relationship between maximum temperature and the electricity demand.

```
vic_elec_daily |>
  pivot_longer(c(Demand,Temperature)) |>
  ggplot(aes(x=Date, y= value)) +
  geom_line() +facet_grid(name ~ .,scales = "free_y")+
  labs(y="")
```



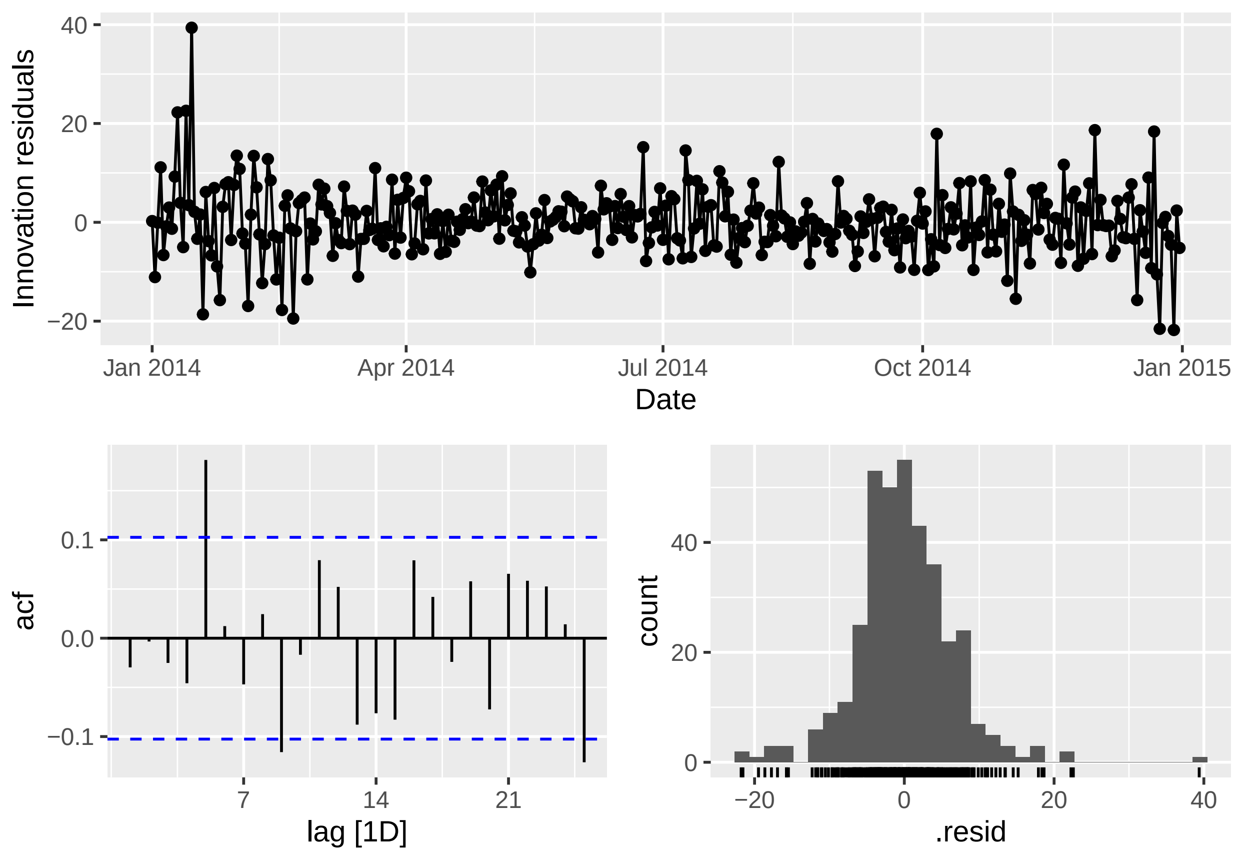
In australia, Jun,Jul, Aug are winter months so temperature is down and demand increases.

```
fit <- vic_elec_daily |>
  model(arima=ARIMA(Demand ~ Temperature + I(Temperature^2)+
                    (Day_Type=="Weekday")))
report(fit)
```

```
## Series: Demand
## Model: LM w/ ARIMA(2,1,2)(2,0,0)[7] errors
##
## Coefficients:
##           ar1     ar2      ma1      ma2    sar1    sar2  Temperature
##       -0.1093  0.7226  -0.0182  -0.9381  0.1958  0.4175      -7.6135
## s.e.   0.0779  0.0739   0.0494   0.0493  0.0525  0.0570       0.4482
##       I(Temperature^2)  Day_Type == "Weekday"TRUE
##                 0.1810                    30.4040
## s.e.            0.0085                     1.3254
##
## sigma^2 estimated as 44.91:  log likelihood=-1206.11
## AIC=2432.21   AICc=2432.84   BIC=2471.18
```

To capture the non-linearity in temperature, we use temperature^2.

```
gg_tsresiduals(fit)
```



```
augment(fit) |>
  features(.resid,ljung_box,dof=6,lag=14)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 arima     28.4  0.000404
```

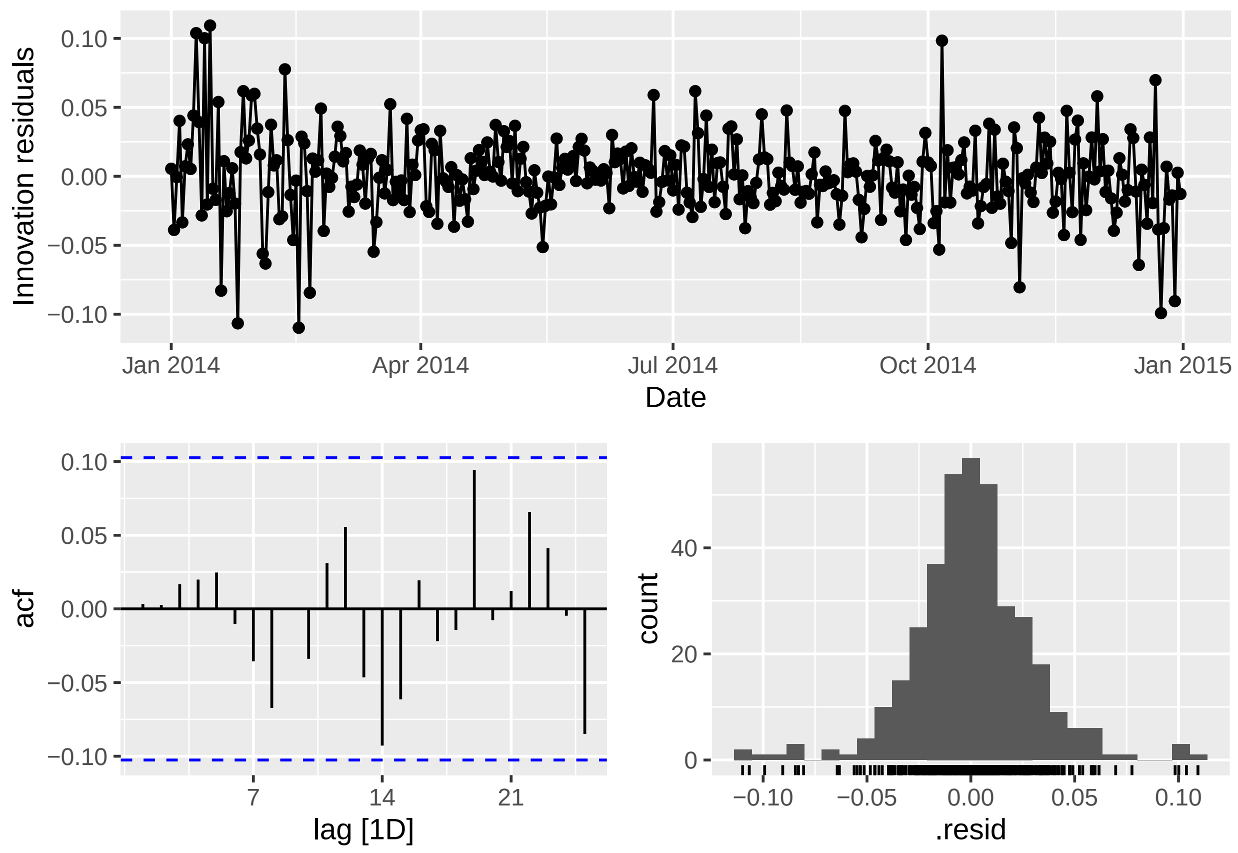
Now, revisit this example again and do the log transformation of the demand.

```
fit <- vic_elec_daily |>
  model(arima=ARIMA(log(Demand) ~ Temperature + I(Temperature^2)+
                      (Day_Type=="Weekday"),stepwise = FALSE,
                    order_constraint = (p+q <=8 & P+Q <=5)))
report(fit)
```

```
## Series: Demand
## Model: LM w/ ARIMA(5,1,3)(2,0,0)[7] errors
## Transformation: log(Demand)
##
## Coefficients:
##          ar1      ar2     ar3     ar4     ar5      ma1      ma2      ma3
##       0.0262  -0.0290  0.5387  -0.087  0.1854  -0.2122  -0.1405  -0.6215
## s.e.  0.2311   0.1481  0.1167   0.061  0.0710   0.2303   0.1793   0.1276
##         sar1    sar2  Temperature  I(Temperature^2)  Day_Type == "Weekday"TRUE
##       0.3217  0.3866      -0.0311             8e-04                    0.1396
## s.e.  0.0594  0.0593       0.0004             0e+00                    0.0061
##
```

```
## sigma^2 estimated as 0.0008747:  log likelihood=769.48
## AIC=-1510.95   AICc=-1509.75   BIC=-1456.39
```

```
gg_tsresiduals(fit)
```



Here, we have not got any significant spikes and hence, we have got residuals of white noise.

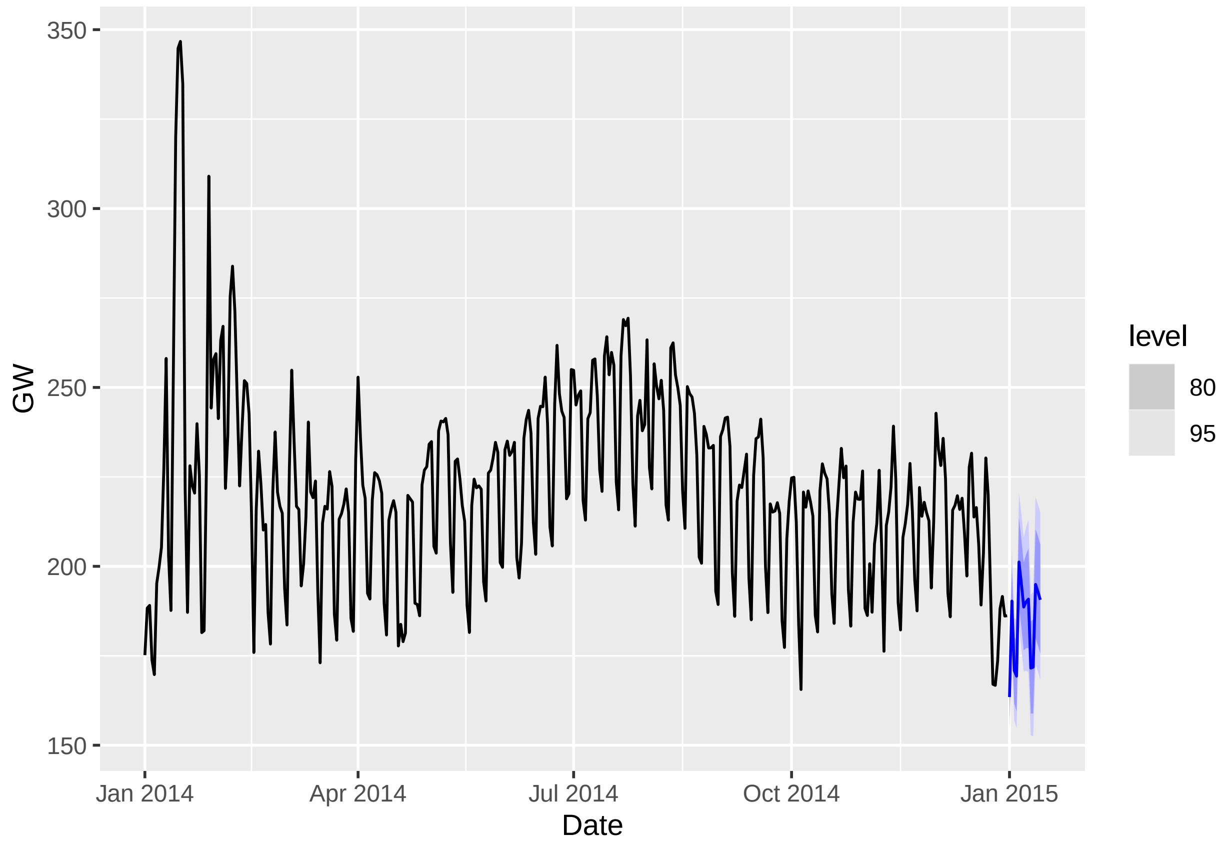
```
# Forecast one day ahead
vic_next_day <- new_data(vic_elec_daily,1) |>
  mutate(Temperature =26, Day_Type="Holiday")
forecast(fit, vic_next_day)
```

```
## # A fable: 1 x 6 [1D]
## # Key:     .model [1]
##   .model Date                  Demand .mean Temperature Day_Type
##   <chr>  <date>                <dist> <dbl>       <dbl> <chr>
## 1 arima  2015-01-01 t(N(5.1, 0.00087))  163.          26 Holiday
```

It returns a mean value of 163 for electricity demand.

```
vic_elec_future <- new_data(vic_elec_daily,14) |>
  mutate(
    Temperature = 26,
    Holiday = c(TRUE,rep(FALSE,13)),
    Day_Type=case_when(
      Holiday~"Holiday",
      wday(Date) %in% 2:6 ~ "Weekday",
      TRUE ~ "Weekend"
    )
  )
```

```
forecast(fit, new_data = vic_elec_future) |>
  autoplot(vic_elec_daily) + labs(y="GW")
```



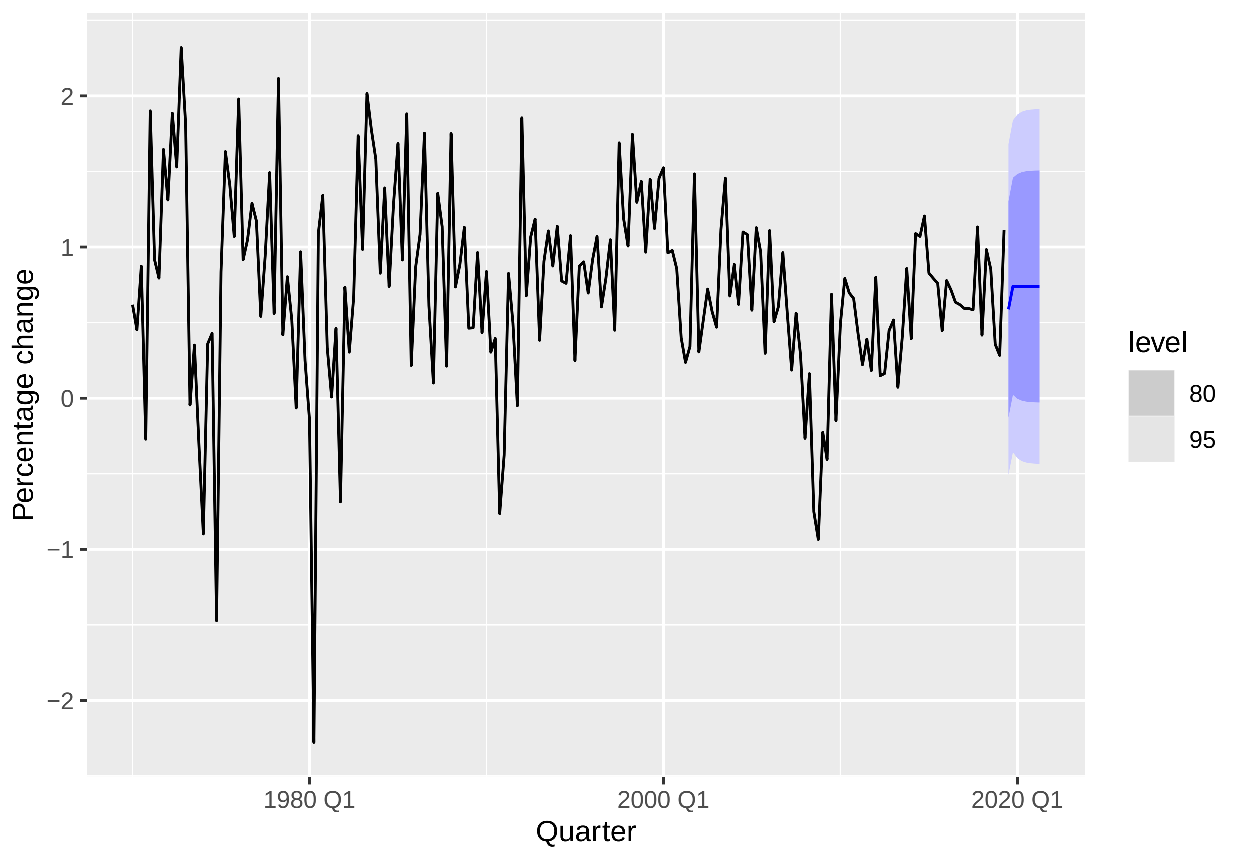
Theory

To forecast using a regression model with ARIMA errors, we need to forecast the regression part of the model and the ARIMA part of the model, and combine the results. As with ordinary regression models, in order to obtain forecasts we first need to forecast the predictors. When the predictors are known into the future (e.g., calendar-related variables such as time, day-of-week, etc.), this is straightforward. But when the predictors are themselves unknown, we must either model them separately, or use assumed future values for each predictor.

Example: US Personal Consumption and Income

We will calculate forecasts for the next eight quarters assuming that the future percentage changes in personal disposable income will be equal to the mean percentage change from the last forty years.

```
fit <- us_change |> model(ARIMA(Consumption ~ Income))
us_change_future <- new_data(us_change, 8) |>
  mutate(Income = mean(us_change$Income))
forecast(fit, new_data = us_change_future) |>
  autoplot(us_change) +
  labs(y = "Percentage change")
```



The prediction intervals for this model are narrower than if we had fitted an ARIMA model without covariates, because we are now able to explain some of the variation in the data using the income predictor.

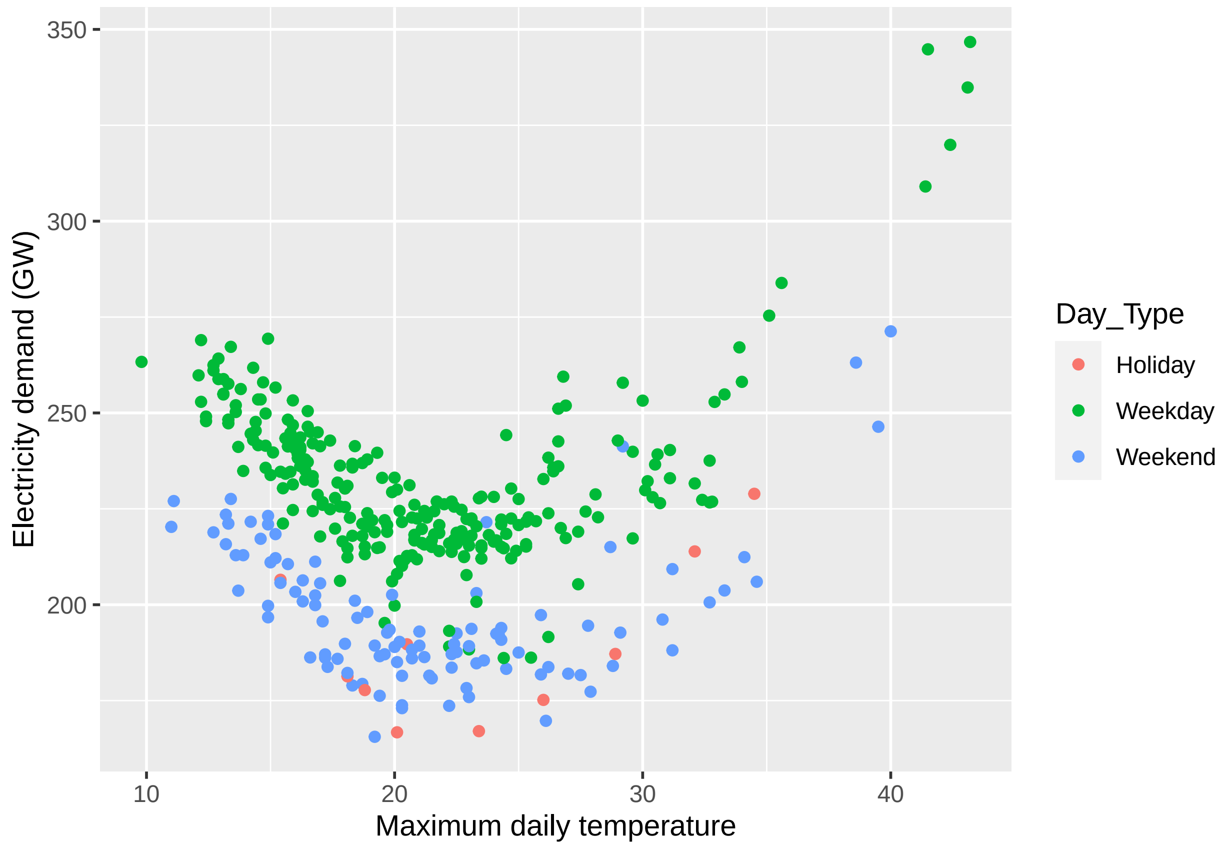
It is important to realise that the prediction intervals from regression models (with or without ARIMA errors) do not take into account the uncertainty in the forecasts of the predictors. So they should be interpreted as being conditional on the assumed (or estimated) future values of the predictor variables.

Example: Forecasting electricity demand

Daily electricity demand can be modelled as a function of temperature. As can be observed on an electricity bill, more electricity is used on cold days due to heating and hot days due to air conditioning. The higher demand on cold and hot days is reflected in the U-shape of Figure 10.5, where daily demand is plotted versus daily maximum temperature.

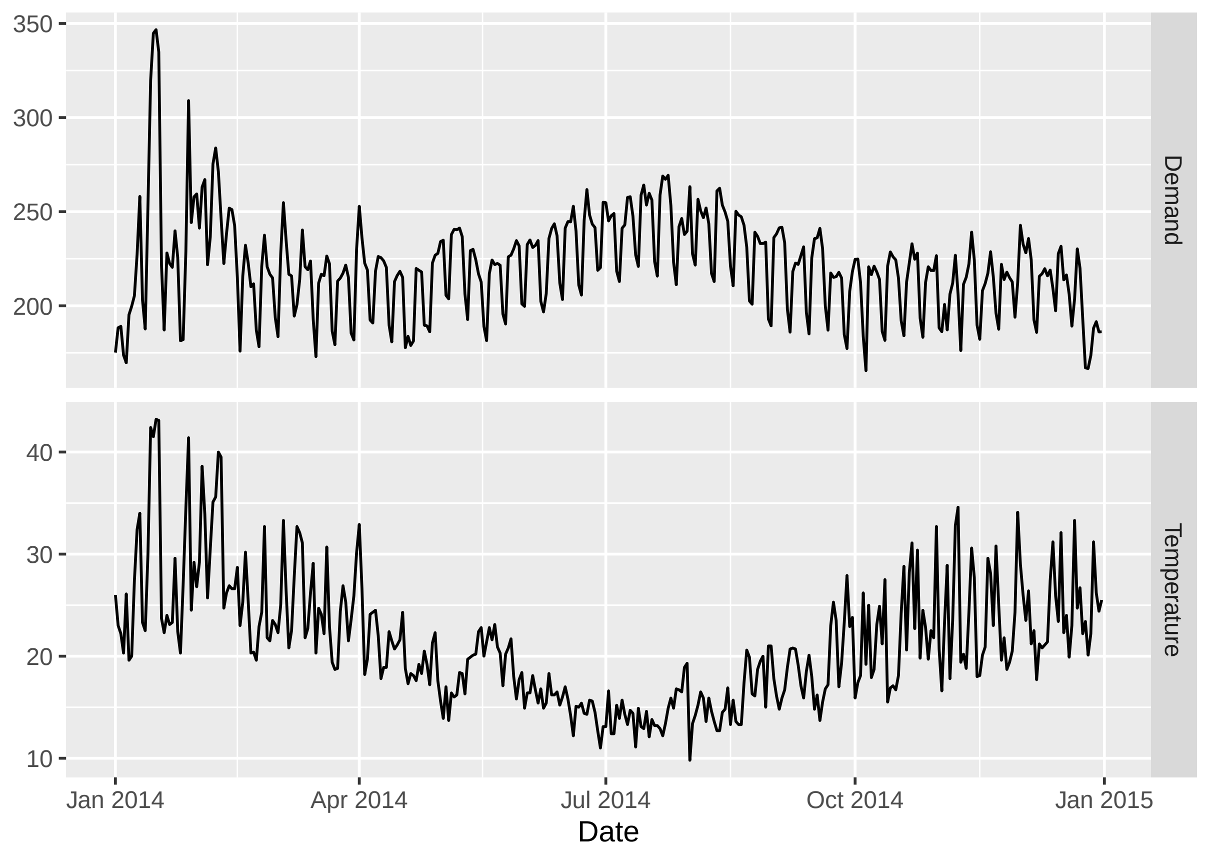
```
vic_elec_daily <- vic_elec |>
  filter(year(Time) == 2014) |>
  index_by(Date = date(Time)) |>
  summarise(
    Demand = sum(Demand) / 1e3,
    Temperature = max(Temperature),
    Holiday = any(Holiday)
  ) |>
  mutate(Day_Type = case_when(
    Holiday ~ "Holiday",
    wday(Date) %in% 2:6 ~ "Weekday",
    TRUE ~ "Weekend"
  ))
```

```r
vic_elec_daily |>
  ggplot(aes(x = Temperature, y = Demand, colour = Day_Type)) +
  geom_point() +
  labs(y = "Electricity demand (GW)",
       x = "Maximum daily temperature")
```



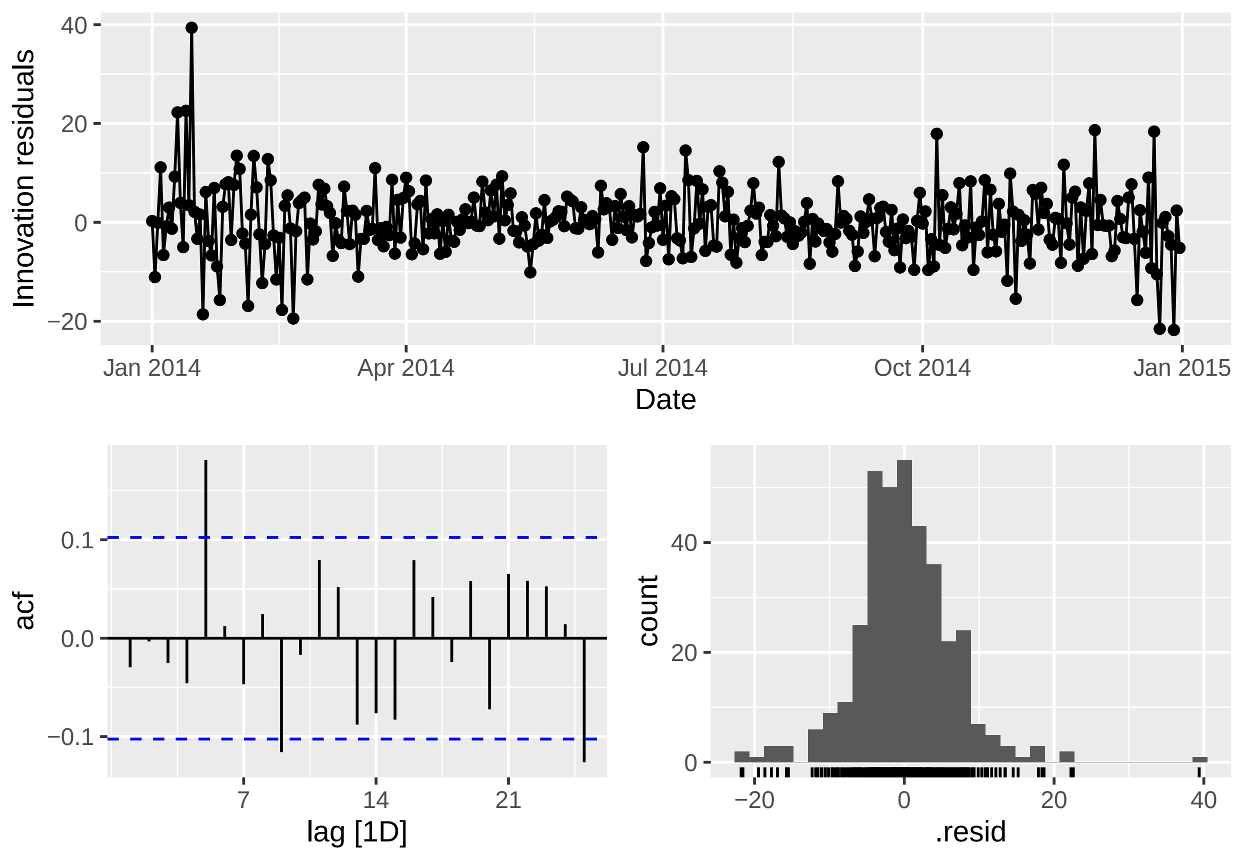
The data stored as vic_elec_daily includes total daily demand, daily maximum temperatures, and an indicator variable for if that day is a public holiday. Figure 10.6 shows the time series of both daily demand and daily maximum temperatures. The plots highlight the need for both a non-linear and a dynamic model.

```r
vic_elec_daily |>
  pivot_longer(c(Demand, Temperature)) |>
  ggplot(aes(x = Date, y = value)) +
  geom_line() +
  facet_grid(name ~ ., scales = "free_y") + ylab("")
```



In this example, we fit a quadratic regression model with ARMA errors using the ARIMA() function. The model also includes an indicator variable for if the day was a working day or not.

```
fit <- vic_elec_daily |>
  model(ARIMA(Demand ~ Temperature + I(Temperature^2) +
                (Day_Type == "Weekday")))
fit |> gg_tsresiduals()
```



The fitted model has an ARIMA(2,1,2)(2,0,0)[7] error, so there are 6 AR and MA coefficients.

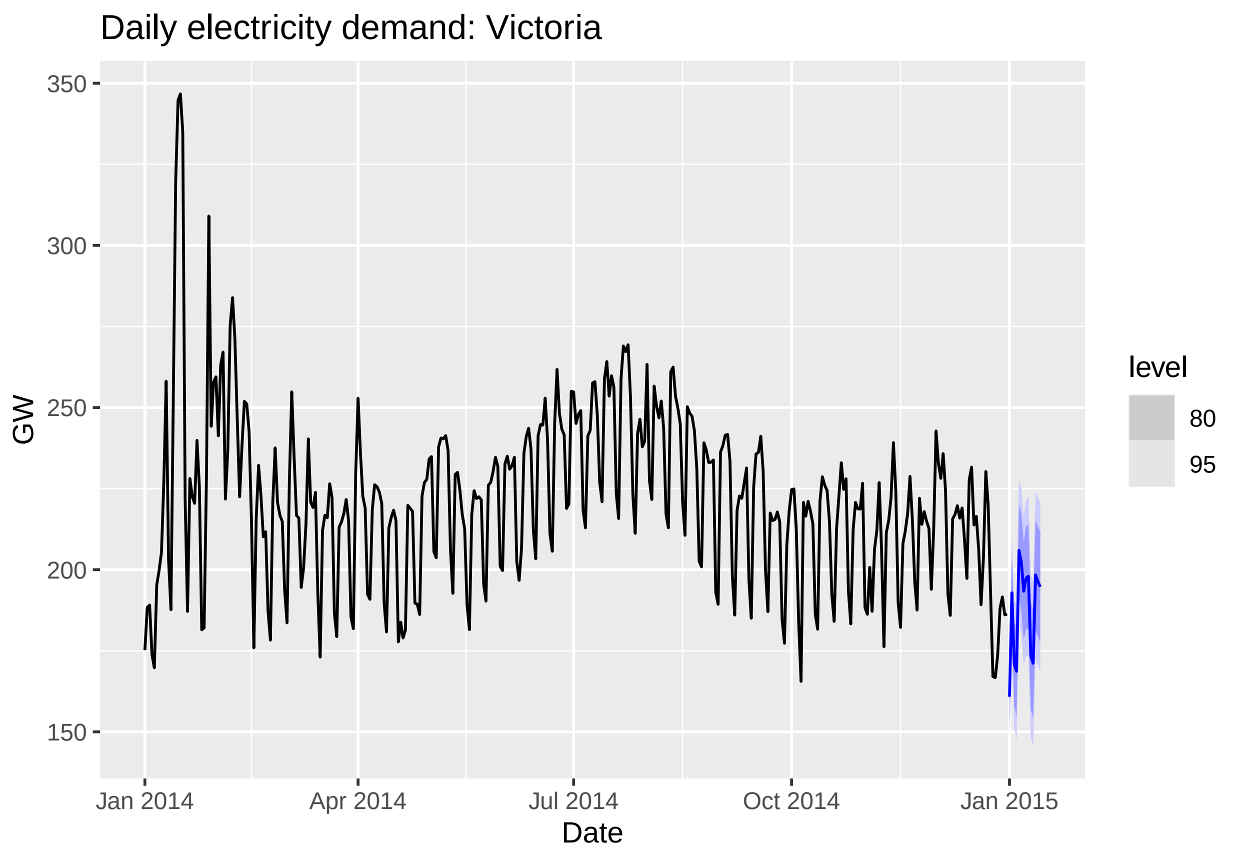
```
augment(fit) |>
  features(.innov, ljung_box, dof = 6, lag = 14)
```

```
## # A tibble: 1 x 3
##   .model                                                            lb_stat lb_pvalue
##   <chr>                                                               <dbl>     <dbl>
## 1 "ARIMA(Demand ~ Temperature + I(Temperature^2) + (Day_Type ~    28.4  0.000404
```

There is clear heteroscedasticity in the residuals, with higher variance in January and February, and lower variance in May. The model also has some significant autocorrelation in the residuals, and the histogram of the residuals shows long tails. All of these issues with the residuals may affect the coverage of the prediction intervals, but the point forecasts should still be ok.

Using the estimated model we forecast 14 days ahead starting from Thursday 1 January 2015 (a non-work-day being a public holiday for New Years Day). In this case, we could obtain weather forecasts from the weather bureau for the next 14 days. But for the sake of illustration, we will use scenario based forecasting (as introduced in Section 7.6) where we set the temperature for the next 14 days to a constant 26 degrees.

```
vic_elec_future <- new_data(vic_elec_daily, 14) |>
  mutate(
    Temperature = 26,
    Holiday = c(TRUE, rep(FALSE, 13)),
    Day_Type = case_when(
      Holiday ~ "Holiday",
      wday(Date) %in% 2:6 ~ "Weekday",
      TRUE ~ "Weekend"
    )
  )
forecast(fit, vic_elec_future) |>
  autoplot(vic_elec_daily) +
  labs(title="Daily electricity demand: Victoria",
       y="GW")
```



The point forecasts look reasonable for the first two weeks of 2015. The slow down in electricity demand at the end of 2014 (due to many people taking summer vacations) has caused the forecasts for the next two weeks to show similarly low demand values.

10.4 Stochastic and deterministic trends

Sometimes in time series modelling, a distinction is made between a stochastic trend and a deterministic trend.

Stochastic and Deterministic Trends

Deterministic Trend It is simply a linear function of time where the error is a stationary process (so, no differencing i.e. d=0).

$y_t = \beta_0 + \beta_1 t + \eta_t$ where $\eta_t \sim ARIMA(p, 0, q)$

Stochastic Trend It is also a linear function of time but error terms have the difference of 1.

$y_t = \beta_0 + \beta_1 t + \eta_t$ where $\eta_t \sim ARIMA(p, 1, q)$

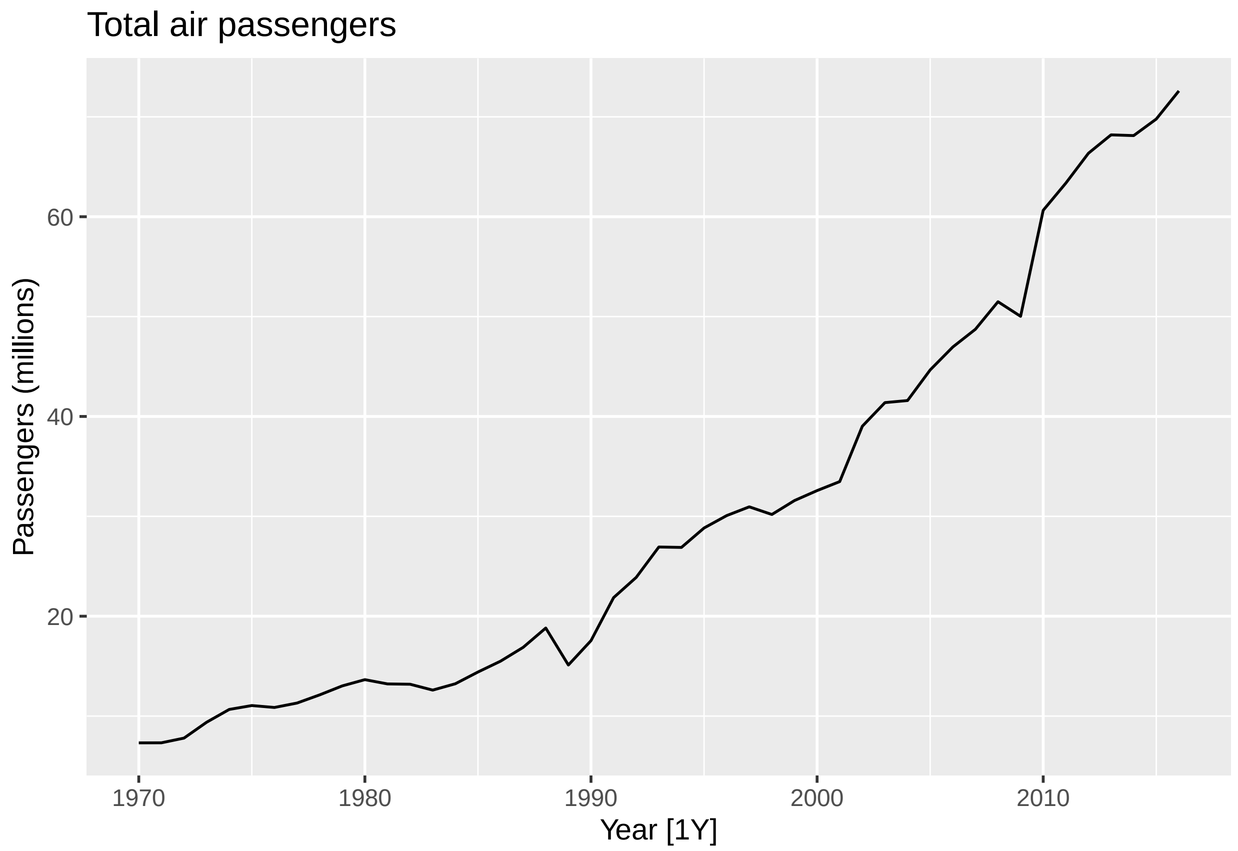
Now, if we take difference both sides of the second equation, we get

$y_t = \beta_1 + \eta'_t$ where $\eta'_t \sim ARIMA(p, 0, q)$

Here, β_0 got disappeared and slope of a line becomes a constant and η'_t is a different of η_t and that is the stationary process.

Example: Air transport passengers Australia

```r
aus_airpassengers |>
  autoplot(Passengers) +
  labs(y="Passengers (millions)",title = "Total air passengers")
```



It's an annual data, so we don't have to thinnk about seasonality and trend is not-quite linear but as increasing.

Deterministic Trend

If we fit a linear trend to it which is deterministic, so, we do:

```r
fit_deterministic <- aus_airpassengers |>
  model(ARIMA(Passengers ~ 1+ trend() + pdq(d=0)))
report(fit_deterministic)
```

```
## Series: Passengers
## Model: LM w/ ARIMA(1,0,0) errors
##
## Coefficients:
```

```
##          ar1  trend()  intercept
##       0.9564   1.4151     0.9014
## s.e.  0.0362   0.1972     7.0751
##
## sigma^2 estimated as 4.343:  log likelihood=-100.88
## AIC=209.77   AICc=210.72   BIC=217.17
```

Here, we can write the equations as:

$y_t = 0.9014 + 1.4151t + \eta_t$ and $\eta_t = 0.9564\eta_{t-1} + \epsilon_t$ and

$\epsilon_t = NID(0, 4.343)$

Stochastic Trend

```r
fit_stochastic <- aus_airpassengers |>
  model(ARIMA(Passengers ~ 1 + pdq(d=1)))
report(fit_stochastic)
```

```
## Series: Passengers
## Model: ARIMA(0,1,0) w/ drift
##
## Coefficients:
##       constant
##         1.4191
## s.e.    0.3014
##
## sigma^2 estimated as 4.271:  log likelihood=-98.16
## AIC=200.31   AICc=200.59   BIC=203.97
```

Here, we can write the equations as:

$y_t - y_{t-1} = 1.4191 + \epsilon_t$

After doing iteration,

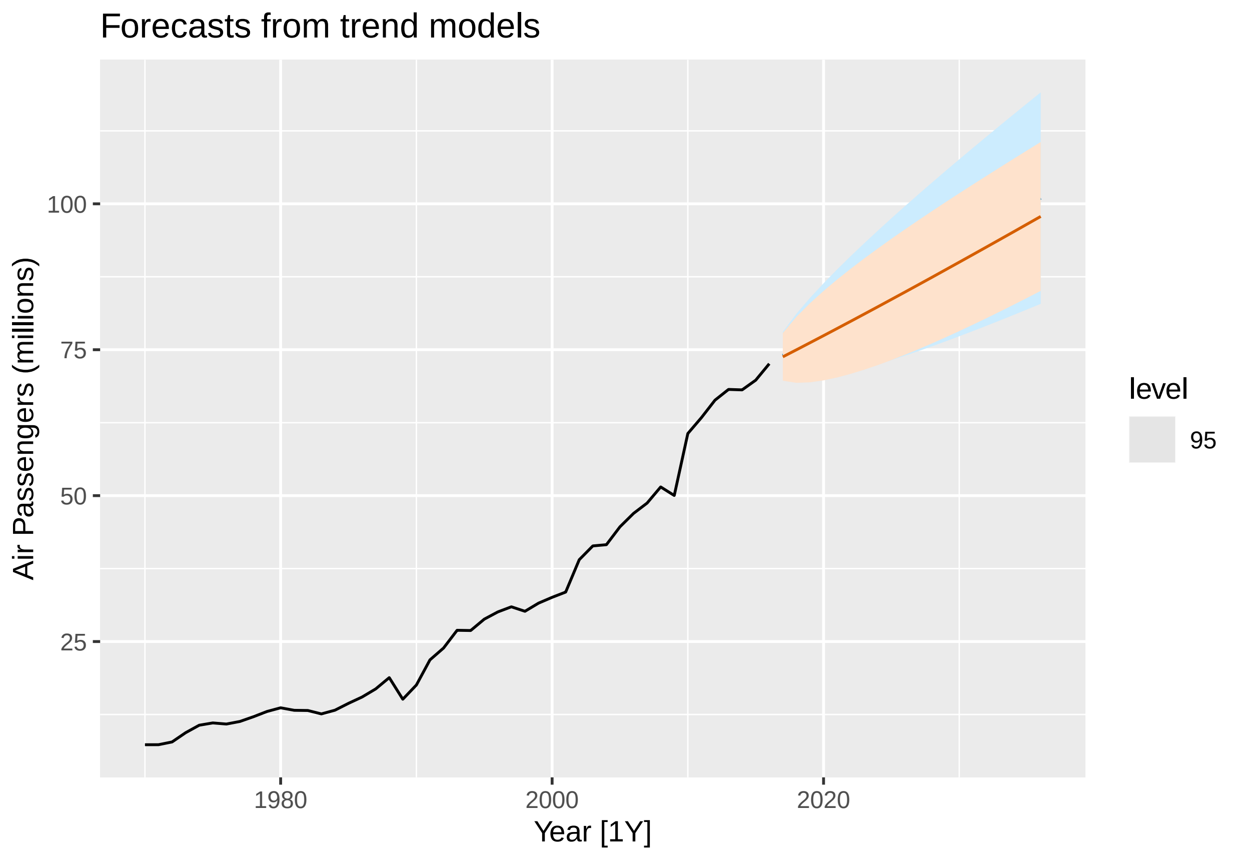
$y_t = y_0 + 1.4191t + \eta_t$

$\eta_t = \eta_{t-1} + \epsilon_t$

$\epsilon_t \sim NID(0, 4.271)$

Now, when we come to forecast:

```r
aus_airpassengers |>
  autoplot(Passengers) +
  autolayer(fit_stochastic |> forecast(h=20),
            colour="#0072B2",level = 95) +
  autolayer(fit_deterministic |> forecast(h=20),
            colour="#D55E00",level = 95) +
  labs(y="Air Passengers (millions)",title = "Forecasts from trend models")
```



Here, slopes are pretty similar and so point forecast are not very different but prediction intervals are quite different, deterministic trend is much narrower and reason is that we are making much stronger assumption that what we see is going to continue in future whereas with the stochastic trend, because of the differencing that's allowing to move around a lot more. So,

- Point forecasts are almost identical, but prediction intervals differ.
- Stochastic trends have much wider prediction intervals because the errors are non-stationary.
- Be careful of forecasting with deterministic trends too far ahead. In general, we don't want to be using forecasting with deterministic trends very far ahead into the future because it's making a very strong assumption that's unlikely to hold for a long period ahead but stochastic trend is a safer trend here.

Theory

There are two different ways of modelling a linear trend. A deterministic trend is obtained using the regression model

$y_t = \beta_0 + \beta_1 t + \eta_t$,

where η_t is an ARMA process. A stochastic trend is obtained using the model $y_t = \beta_0 + \beta_1 t + \eta_t$,

where η_t is an ARIMA process with $d = 1$. In the latter case, we can difference both sides so that $y'_t = \beta_1 + \eta'_t$, where η'_t is an ARMA process. In other words,$y_t = y_{t-1} + \beta_1 + \eta'_t$.

This is similar to a random walk with drift (introduced in Section 9.1), but here the error term is an ARMA process rather than simply white noise.

Although these models appear quite similar (they only differ in the number of differences that need to be applied to η_t), their forecasting characteristics are quite different.

Example: Air transport passengers Australia

```
aus_airpassengers |>
  autoplot(Passengers) +
  labs(y = "Passengers (millions)",
       title = "Total annual air passengers")
```

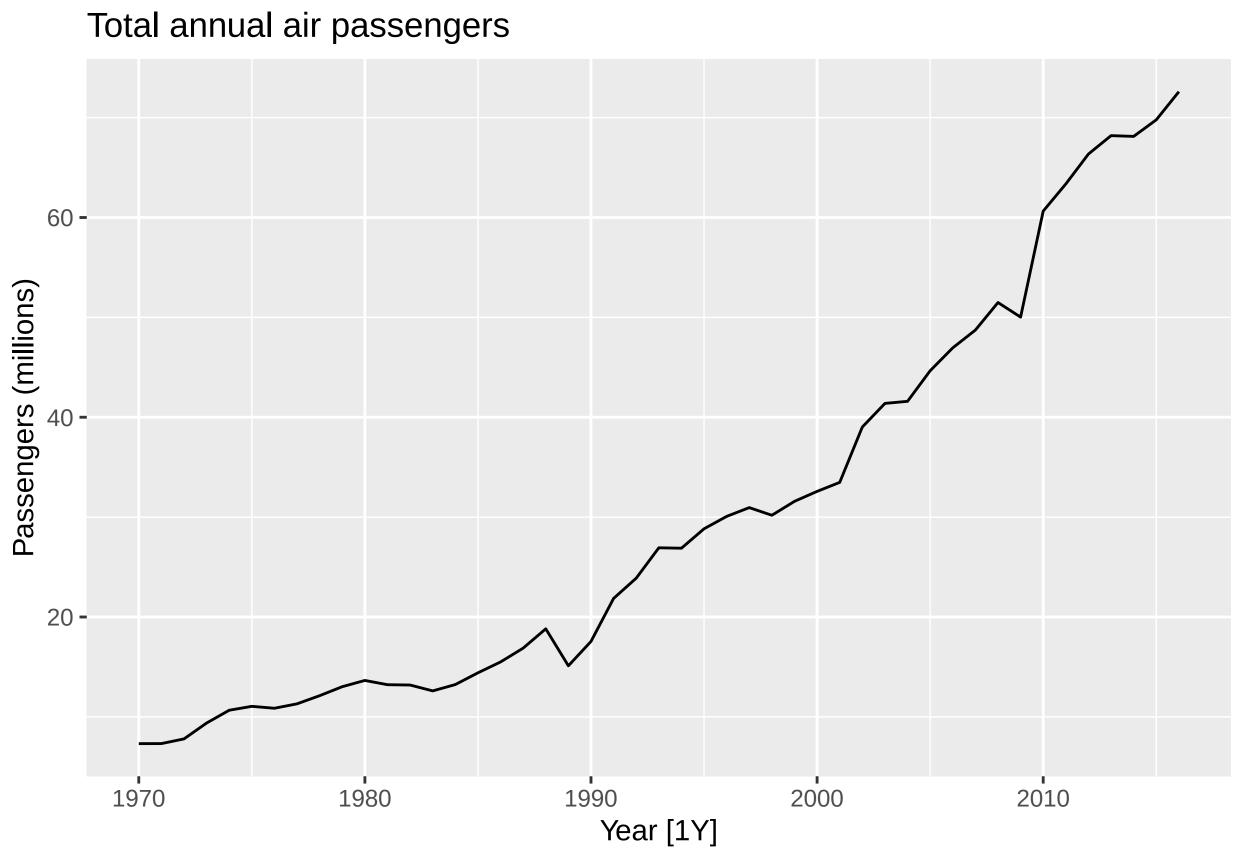


Figure 10.9 shows the total number of passengers for Australian air carriers each year from 1970 to 2016. We will fit both a deterministic and a stochastic trend model to these data.

The deterministic trend model is obtained as follows:

```
fit_deterministic <- aus_airpassengers |>
  model(deterministic = ARIMA(Passengers ~ 1 + trend() +
                                pdq(d = 0)))
report(fit_deterministic)
```

```
## Series: Passengers
## Model: LM w/ ARIMA(1,0,0) errors
##
## Coefficients:
##          ar1  trend()  intercept
##       0.9564   1.4151     0.9014
## s.e.  0.0362   0.1972     7.0751
##
```

```
## sigma^2 estimated as 4.343:  log likelihood=-100.88
## AIC=209.77   AICc=210.72   BIC=217.17
```

The estimated growth in visitor numbers is 1.42 million people per year.

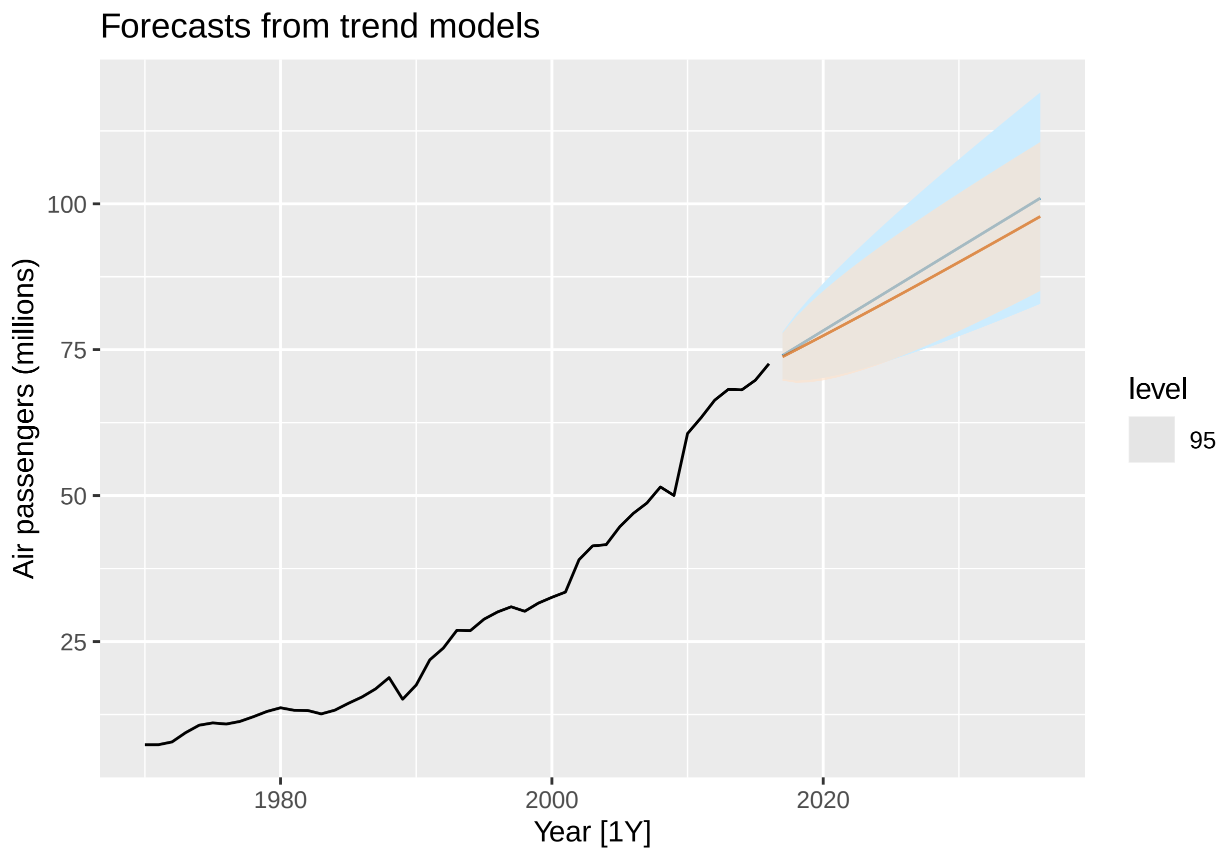
Alternatively, the stochastic trend model can be estimated.

```
fit_stochastic <- aus_airpassengers |>
  model(stochastic = ARIMA(Passengers ~ pdq(d = 1)))
report(fit_stochastic)
```

```
## Series: Passengers
## Model: ARIMA(0,1,0) w/ drift
##
## Coefficients:
##       constant
##         1.4191
## s.e.    0.3014
##
## sigma^2 estimated as 4.271:  log likelihood=-98.16
## AIC=200.31   AICc=200.59   BIC=203.97
```

In this case, the estimated growth in visitor numbers is also 1.42 million people per year. Although the growth estimates are similar, the prediction intervals are not, as Figure 10.10 shows. In particular, stochastic trends have much wider prediction intervals because the errors are non-stationary.

```
aus_airpassengers |>
  autoplot(Passengers) +
  autolayer(fit_stochastic |> forecast(h = 20),
    colour = "#0072B2", level = 95) +
  autolayer(fit_deterministic |> forecast(h = 20),
    colour = "#D55E00", alpha = 0.65, level = 95) +
  labs(y = "Air passengers (millions)",
       title = "Forecasts from trend models")
```



There is an implicit assumption with deterministic trends that the slope of the trend is not going to change over time. On the other hand, stochastic trends can change, and the estimated growth is only assumed to be the average growth over the historical period, not necessarily the rate of growth that will be observed into the future. Consequently, it is safer to forecast with stochastic trends, especially for longer forecast horizons, as the prediction intervals allow for greater uncertainty in future growth.

10.5 Dynamic harmonic regression

Combine Fourier terms with ARIMA errors

Advantages

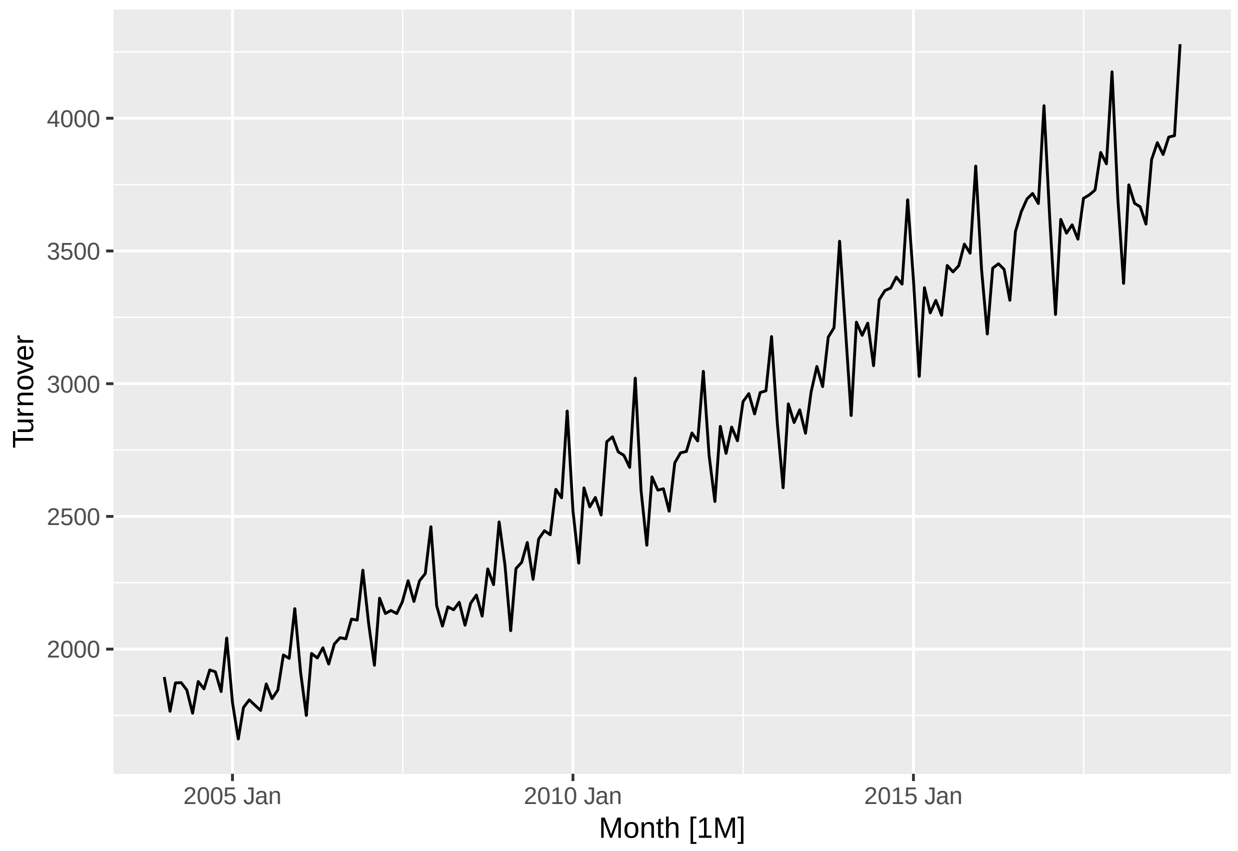
- It allows any length seasonality;
- For data with more than one seasonal period, you can include Fourier terms of different frequencies.
- the seasonal pattern is smooth for small values of K (but more wiggly seasonality can be handled by increasing K);
- the short-term dynamics are easily handled with a simple ARMA error.

Disadvantages

- Seasonality is assumed to be fixed.

Example: Eating-out expenditure

```
aus_cafe <- aus_retail |>
  filter(Industry == "Cafes, restaurants and takeaway food services",
         year(Month) %in% 2004:2018) |>
  summarise(Turnover = sum(Turnover))
aus_cafe |> autoplot(Turnover)
```



It is monthly data and normally we don't use Fourier terms for monthly data because there is only 12 periods and we can handle with other ways but just how it works we have given the example.

Here, trend is close to linear and there is a strong seasonality. We are going to deal with log(Turnover) because in above plot, size of the fluctuations increases as level of the series increases which is typical log behaviour.

```
fit <- aus_cafe |> model(
  `K=1` = ARIMA(log(Turnover) ~ fourier(K=1) + PDQ(0,0,0)),
  `K=2` = ARIMA(log(Turnover) ~ fourier(K=2) + PDQ(0,0,0)),
  `K=3` = ARIMA(log(Turnover) ~ fourier(K=3) + PDQ(0,0,0)),
  `K=4` = ARIMA(log(Turnover) ~ fourier(K=4) + PDQ(0,0,0)),
  `K=5` = ARIMA(log(Turnover) ~ fourier(K=5) + PDQ(0,0,0)),
  `K=6` = ARIMA(log(Turnover) ~ fourier(K=6) + PDQ(0,0,0))
)
glance(fit)
```

```
## # A tibble: 6 x 8
##   .model    sigma2 log_lik   AIC  AICc   BIC ar_roots   ma_roots
##   <chr>      <dbl>   <dbl> <dbl> <dbl> <dbl> <list>     <list>
## 1 K=1    0.00175     317. -616. -615. -588. <cpl [2]> <cpl [3]>
## 2 K=2    0.00107     362. -700. -698. -661. <cpl [5]> <cpl [1]>
## 3 K=3    0.000761    394. -763. -761. -725. <cpl [3]> <cpl [1]>
## 4 K=4    0.000539    427. -822. -818. -771. <cpl [1]> <cpl [5]>
## 5 K=5    0.000317    474. -919. -917. -875. <cpl [2]> <cpl [0]>
## 6 K=6    0.000316    474. -920. -918. -875. <cpl [0]> <cpl [1]>
```

$K = 1$ means seasonality is modeled as sine wave and then add two harmonics with $K = 2$ and so on. With $K = 6$, it got fully saturated.

Here, model with $K = 6$ gives smallest AICc statistics.

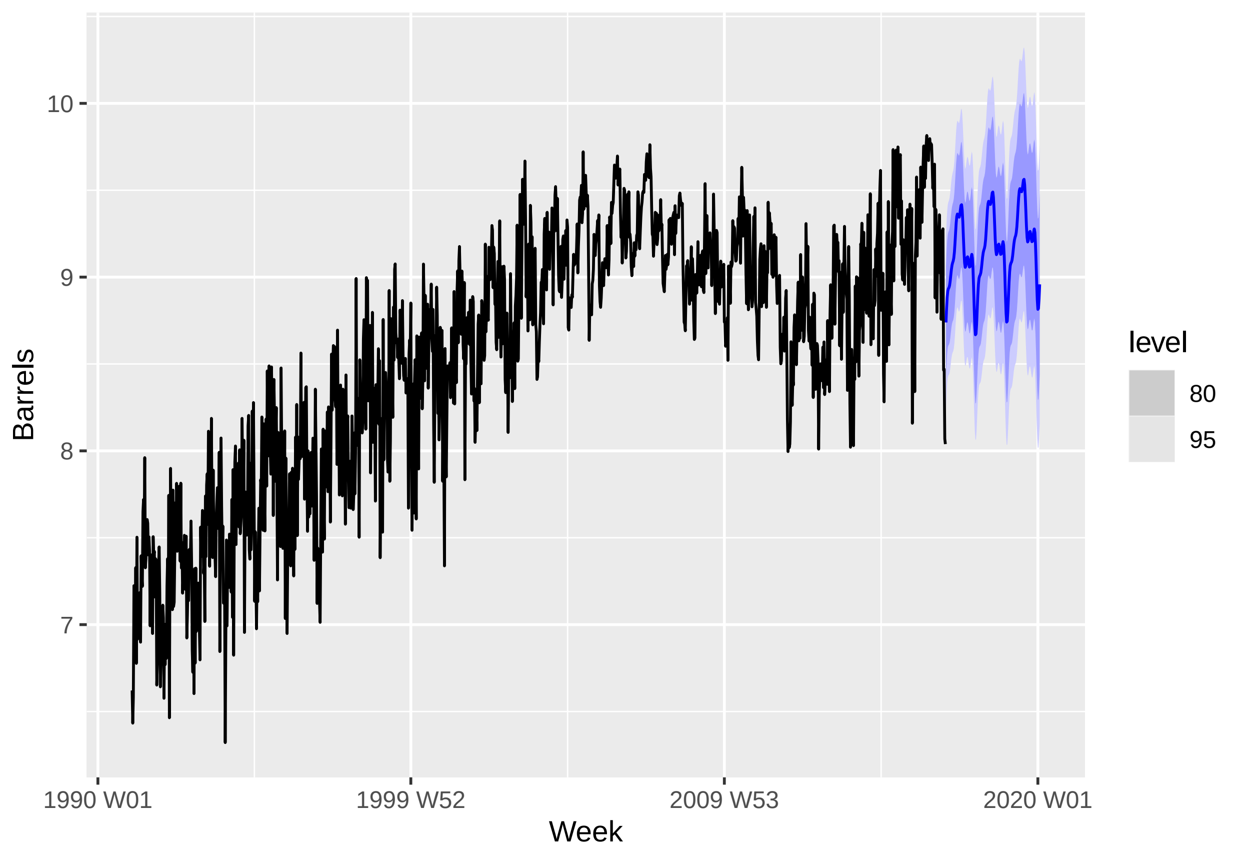
Now, what we would normally use fourier terms for weekly data because period of seasonality is very large like 52.

```
fit <- us_gasoline |>
  model(K06=ARIMA(Barrels ~ fourier(K=6)+PDQ(0,0,0)))
report(fit)
```

```
## Series: Barrels
## Model: LM w/ ARIMA(0,1,1) errors
##
## Coefficients:
##           ma1  fourier(K = 6)C1_52  fourier(K = 6)S1_52  fourier(K = 6)C2_52
##       -0.8956              -0.1122              -0.2300               0.0419
## s.e.   0.0130               0.0123               0.0122               0.0099
##       fourier(K = 6)S2_52  fourier(K = 6)C3_52  fourier(K = 6)S3_52
##                    0.0316               0.0832               0.0345
## s.e.               0.0099               0.0094               0.0094
##       fourier(K = 6)C4_52  fourier(K = 6)S4_52  fourier(K = 6)C5_52
##                    0.0186               0.0397              -0.0314
## s.e.               0.0093               0.0092               0.0092
##       fourier(K = 6)S5_52  fourier(K = 6)C6_52  fourier(K = 6)S6_52  intercept
##                    0.0010              -0.0522               0.0002     0.0014
## s.e.               0.0092               0.0091               0.0091     0.0007
##
## sigma^2 estimated as 0.06205:  log likelihood=-33.11
## AIC=96.23   AICc=96.59   BIC=174.39
```

We are using K=6 because it gives the smallest AICc statistics. Here, d=1 so drift term is there and there will be some trend in the forecast.

```
forecast(fit, h="3 years") |>
  autoplot(us_gasoline)
```



Here, we have quite big prediction interval because there is lots of noise in the dataset.

Theory

When there are long seasonal periods, a dynamic regression with Fourier terms is often better than other models we have considered in this book.22

For example, daily data can have annual seasonality of length 365, weekly data has seasonal period of approximately 52, while half-hourly data can have several seasonal periods, the shortest of which is the daily pattern of period 48.

Seasonal versions of ARIMA and ETS models are designed for shorter periods such as 12 for monthly data or 4 for quarterly data. The ETS() model restricts seasonality to be a maximum period of 24 to allow hourly data but not data with a larger seasonal period. The problem is that there are m−1 parameters to be estimated for the initial seasonal states where m is the seasonal period. So for large m, the estimation becomes almost impossible.

The ARIMA() function will allow a seasonal period up to m=350, but in practice will usually run out of memory whenever the seasonal period is more than about 200. In any case, seasonal differencing of high order does not make a lot of sense — for daily data it involves comparing what happened today with what happened exactly a year ago and there is no constraint that the seasonal pattern is smooth.

So for such time series, we prefer a harmonic regression approach where the seasonal pattern is modelled using Fourier terms with short-term time series dynamics handled by an ARMA error.

The advantages of this approach are:

- it allows any length seasonality;

- for data with more than one seasonal period, Fourier terms of different frequencies can be included;
- the smoothness of the seasonal pattern can be controlled by K, the number of Fourier sin and cos pairs – the seasonal pattern is smoother for smaller values of K;
- the short-term dynamics are easily handled with a simple ARMA error.

The only real disadvantage (compared to a seasonal ARIMA model) is that the seasonality is assumed to be fixed — the seasonal pattern is not allowed to change over time. But in practice, seasonality is usually remarkably constant so this is not a big disadvantage except for long time series.

###Example: Australian eating out expenditure

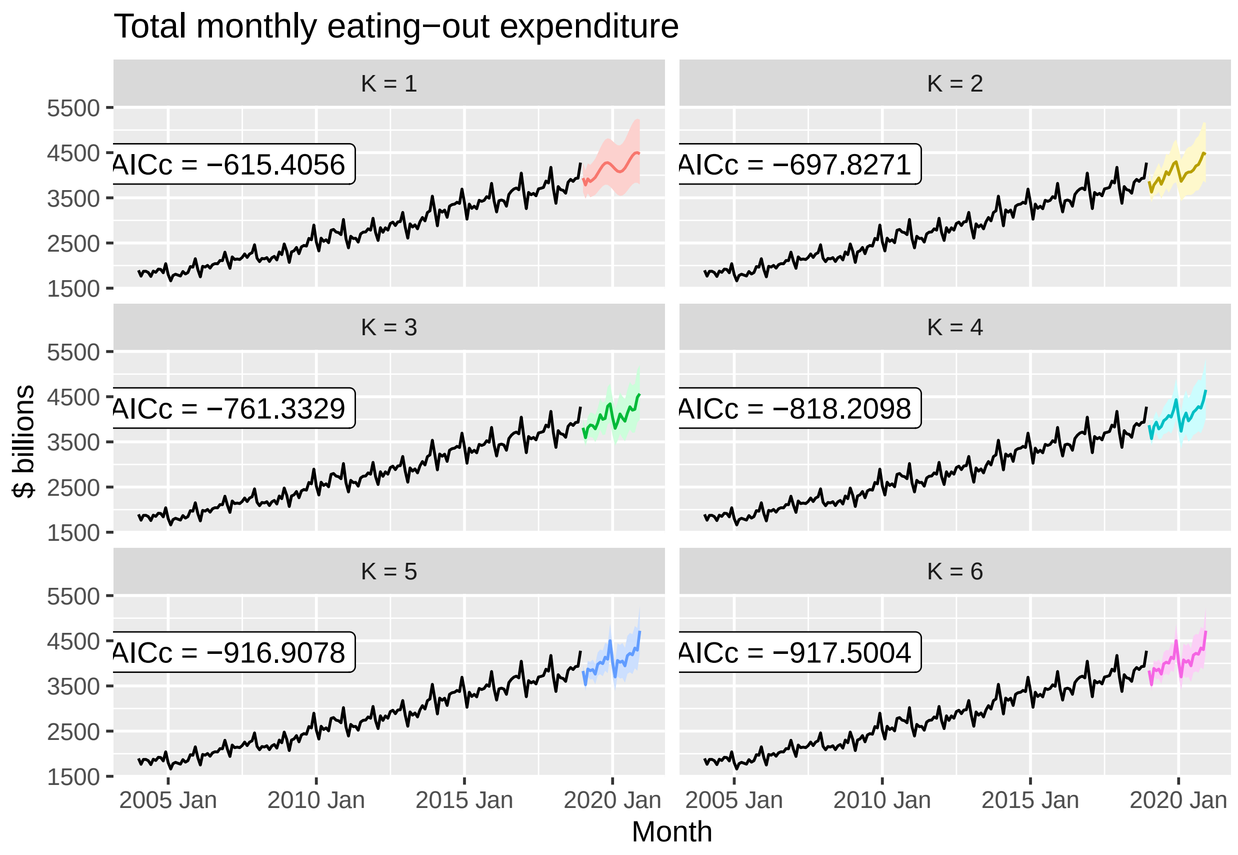
In this example we demonstrate combining Fourier terms for capturing seasonality with ARIMA errors capturing other dynamics in the data. For simplicity, we will use an example with monthly data. The same modelling approach using weekly data is discussed in Section 13.1.

We use the total monthly expenditure on cafes, restaurants and takeaway food services in Australia ($billion) from 2004 up to 2018 and forecast 24 months ahead. We vary K, the number of Fourier sin and cos pairs, from K=1 to K=6(which is equivalent to including seasonal dummies). Figure 10.11 shows the seasonal pattern projected forward as K increases. Notice that as K increases the Fourier terms capture and project a more "wiggly" seasonal pattern and simpler ARIMA models are required to capture other dynamics. The AICc value is minimised for K=6, with a significant jump going from K=4 to K=5, hence the forecasts generated from this model would be the ones used.

```r
aus_cafe <- aus_retail |>
  filter(
    Industry == "Cafes, restaurants and takeaway food services",
    year(Month) %in% 2004:2018
  ) |>
  summarise(Turnover = sum(Turnover))

fit <- model(aus_cafe,
  `K = 1` = ARIMA(log(Turnover) ~ fourier(K=1) + PDQ(0,0,0)),
  `K = 2` = ARIMA(log(Turnover) ~ fourier(K=2) + PDQ(0,0,0)),
  `K = 3` = ARIMA(log(Turnover) ~ fourier(K=3) + PDQ(0,0,0)),
  `K = 4` = ARIMA(log(Turnover) ~ fourier(K=4) + PDQ(0,0,0)),
  `K = 5` = ARIMA(log(Turnover) ~ fourier(K=5) + PDQ(0,0,0)),
  `K = 6` = ARIMA(log(Turnover) ~ fourier(K=6) + PDQ(0,0,0))
)

fit |>
  forecast(h = "2 years") |>
  autoplot(aus_cafe, level = 95) +
  facet_wrap(vars(.model), ncol = 2) +
  guides(colour = "none", fill = "none", level = "none") +
  geom_label(
    aes(x = yearmonth("2007 Jan"), y = 4250,
        label = paste0("AICc = ", format(AICc))),
    data = glance(fit)
  ) +
  labs(title= "Total monthly eating-out expenditure",
       y="$ billions")
```



10.6 Lagged predictors

lagged predictors is used when we change in the x variable doesn't necessarily affect the y variable i.e. sometimes a change in x_t does not affect y_t instantaneously.

- Examples:

1) y_t = sales and x_t = advertising
2) y_t = stream flow and x_t = rainfall
3) y_t = size of herd and x_t = breeding stock

Whenever we get these lagged effect of predictors, we can put lagged values of x_t into the dynamic regression model.

- These are dynamic systems with input x_t and output y_t
- x_t is often a leading indicator
- There can be multiple predictors.

The model include present and past values of the predictor:

$y_t = a + \gamma_0 x_t + \gamma_1 x_{t-1} + ... + \gamma_k x_{t-k} + \eta_t$ where η_t is an ARIMA process.

We can rewrite this model using the backshift operation:

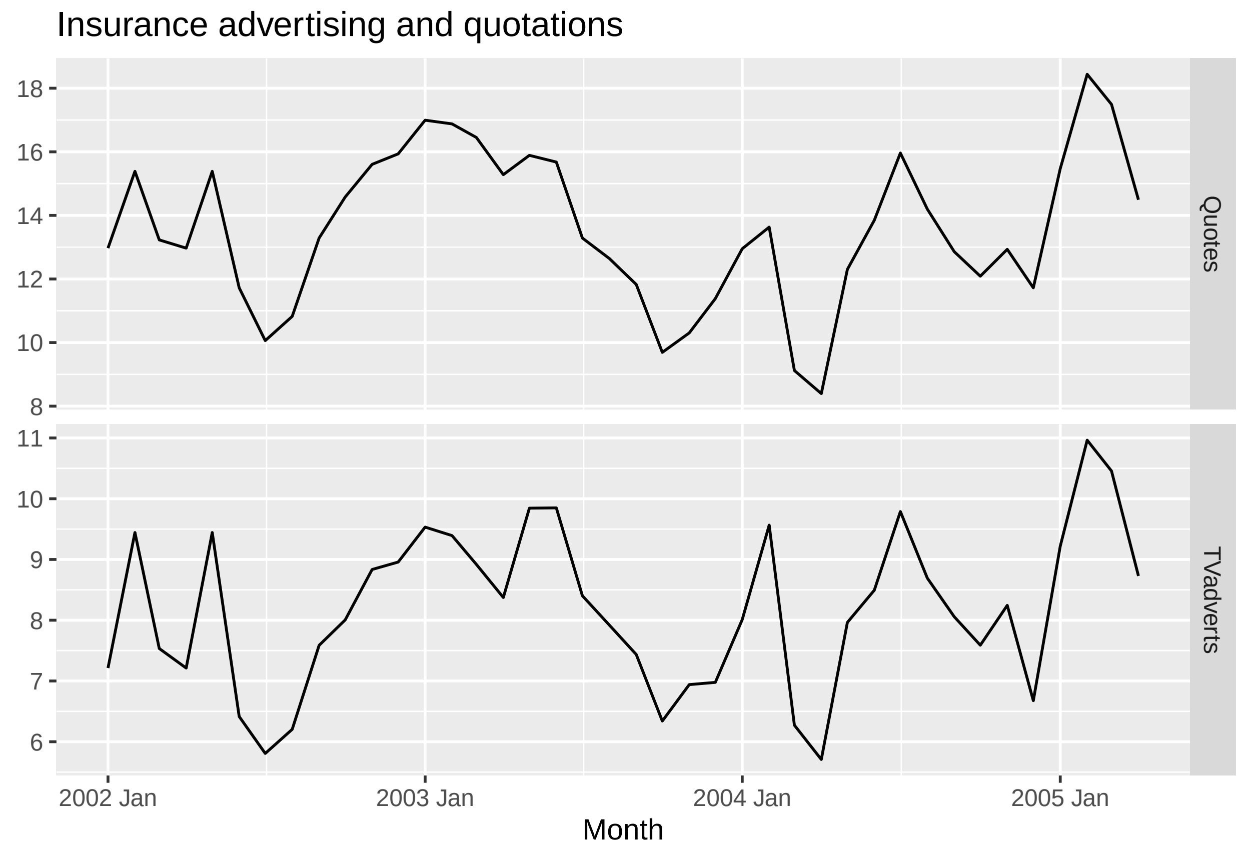
$y_t = a + (\gamma_0 + \gamma_1 B + \gamma_2 B^2 + ... + \gamma_k B^k)x_t + \eta_t = a + \gamma(B)x_t + \eta_t$

- $\gamma(B)$ is called the transfer function since it describes how change in x_t is transferred to y_t

- x can influence y but y is not allowed to influence x (x is exogenous variable). If we go in both direction i.e. x affects y and y affects x then we need a different model which is called multivariate model.

Example: Insurance quotes and TV adverts

```
insurance |>
  pivot_longer(Quotes:TVadverts) |>
  ggplot(aes(x=Month, y=value)) +
  geom_line() +
  facet_grid(vars(name), scales = "free_y") +
  labs(y=NULL, title = "Insurance advertising and quotations")
```



Here, we can see both series moves together. Now, we are going to fit a range of models with different number of lagged advertising in them. Starting with no lag, one lag, two lags, three lags and so on and we compare the following three models. We use NA so that we can use lag=3 in last model.

```
fit <- insurance |>
  # Restrict data so models use some fitting period
  mutate(Quotes=c(NA,NA,NA,Quotes[4:40])) |>
  # Estimate models
  model(
    ARIMA(Quotes~pdq(d=0)+TVadverts),
    ARIMA(Quotes~pdq(d=0)+TVadverts+lag(TVadverts)),
    ARIMA(Quotes~pdq(d=0)+TVadverts+lag(TVadverts)+lag(TVadverts,2)),
    ARIMA(Quotes~pdq(d=0)+TVadverts+lag(TVadverts)+lag(TVadverts,2)+lag(TVadverts,3))
  )
```

```r
glance(fit)
```

```
## # A tibble: 4 x 8
##   .model                        sigma2 log_lik   AIC  AICc   BIC ar_roots ma_roots
##   <chr>                          <dbl>   <dbl> <dbl> <dbl> <dbl> <list>   <list>
## 1 "ARIMA(Quotes ~ pdq(d = 0)~  0.265   -28.3  66.6  68.3  75.0 <cpl>    <cpl>
## 2 "ARIMA(Quotes ~ pdq(d = 0)~  0.209   -24.0  58.1  59.9  66.5 <cpl>    <cpl>
## 3 "ARIMA(Quotes ~ pdq(d = 0)~  0.215   -24.0  60.0  62.6  70.2 <cpl>    <cpl>
## 4 "ARIMA(Quotes ~ pdq(d = 0)~  0.206   -22.2  60.3  65.0  73.8 <cpl>    <cpl>
```

Now, from the AICc statistic, we can see the second model is best model with lag 1.

Now, we are going to refit the model from all the available data.

```r
fit_best <- insurance |>
  model(ARIMA(Quotes ~ pdq(d=0)+TVadverts+lag(TVadverts)))
report(fit_best)
```

```
## Series: Quotes
## Model: LM w/ ARIMA(1,0,2) errors
##
## Coefficients:
##          ar1     ma1     ma2  TVadverts  lag(TVadverts)  intercept
##       0.5123  0.9169  0.4591     1.2527          0.1464     2.1554
## s.e.  0.1849  0.2051  0.1895     0.0588          0.0531     0.8595
##
## sigma^2 estimated as 0.2166:  log likelihood=-23.94
## AIC=61.88   AICc=65.38   BIC=73.7
```

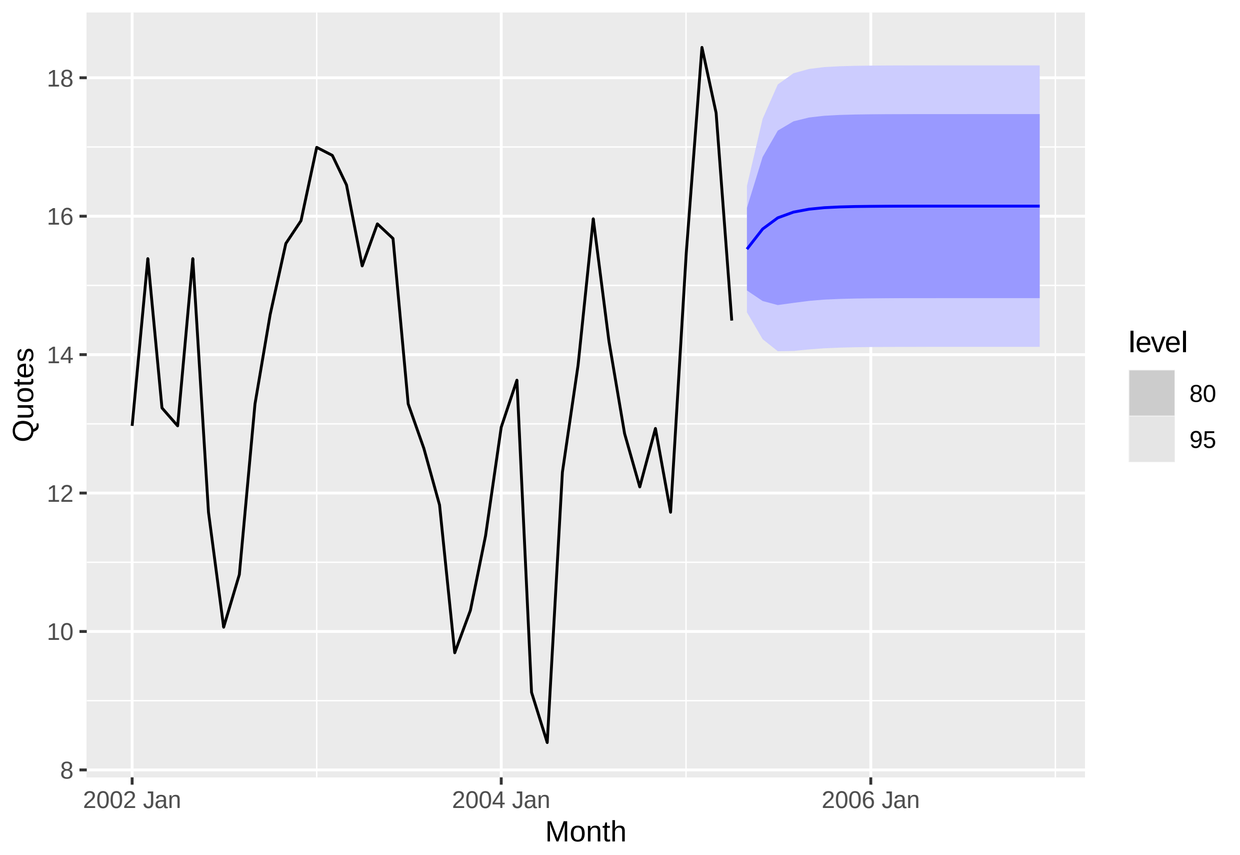
So, we write the equation as:

$y_t = 2.1554 + 1.2527x_t + 0.1464x_{t-1} + \eta_t$

$\eta_t = 0.5123\eta_{t-1} + \epsilon_t + 0.9169\epsilon_{t-1} + 0.4591\epsilon_{t-2}$

Now, consider a scenario where we spend 10 unit on TV advertising and we go 20 steps ahead. So,

```r
advert_a <- new_data(insurance,20) |>
  mutate(TVadverts=10)
forecast(fit_best,advert_a) |> autoplot(insurance)
```



It tells how much we are going to spend in each of the subsequent steps.

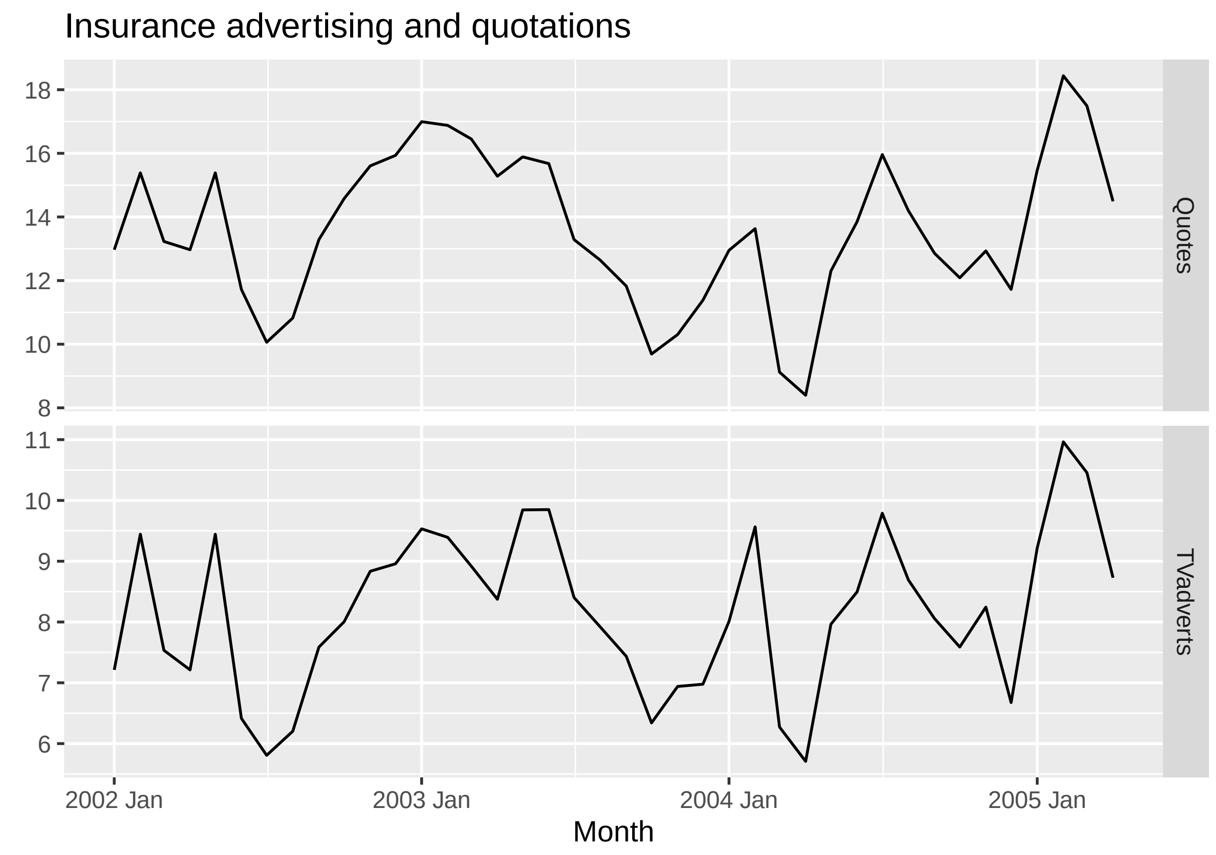
Sometimes, the impact of a predictor that is included in a regression model will not be simple and immediate. For example, an advertising campaign may impact sales for some time beyond the end of the campaign, and sales in one month will depend on the advertising expenditure in each of the past few months. Similarly, a change in a company's safety policy may reduce accidents immediately, but have a diminishing effect over time as employees take less care when they become familiar with the new working conditions.

In these situations, we need to allow for lagged effects of the predictor. Suppose that we have only one predictor in our model. Then a model which allows for lagged effects can be written as $y_t = \beta_0 + \gamma_0 x_t + \gamma_1 x_{t-1} + \cdots + \gamma_k x_{t-k} + \eta_t$, where η_t is an ARIMA process. The value of k can be selected using the AICc, along with the values of p and q for the ARIMA error.

Example: TV advertising and insurance quotations

A US insurance company advertises on national television in an attempt to increase the number of insurance quotations provided (and consequently the number of new policies). Figure 10.12 shows the number of quotations and the expenditure on television advertising for the company each month from January 2002 to April 2005.

```
insurance |>
  pivot_longer(Quotes:TVadverts) |>
  ggplot(aes(x = Month, y = value)) +
  geom_line() +
  facet_grid(vars(name), scales = "free_y") +
  labs(y = "", title = "Insurance advertising and quotations")
```



We will consider including advertising expenditure for up to four months; that is, the model may include advertising expenditure in the current month, and the three months before that. When comparing models, it is important that they all use the same training set. In the following code, we exclude the first three months in order to make fair comparisons.

```
fit <- insurance |>
  # Restrict data so models use same fitting period
  mutate(Quotes = c(NA, NA, NA, Quotes[4:40])) |>
  # Estimate models
  model(
    lag0 = ARIMA(Quotes ~ pdq(d = 0) + TVadverts),
    lag1 = ARIMA(Quotes ~ pdq(d = 0) +
                 TVadverts + lag(TVadverts)),
    lag2 = ARIMA(Quotes ~ pdq(d = 0) +
                 TVadverts + lag(TVadverts) +
                 lag(TVadverts, 2)),
    lag3 = ARIMA(Quotes ~ pdq(d = 0) +
                 TVadverts + lag(TVadverts) +
                 lag(TVadverts, 2) + lag(TVadverts, 3))
  )
```

Next we choose the optimal lag length for advertising based on the AICc.

```
glance(fit)
```

```
## # A tibble: 4 x 8
##   .model sigma2 log_lik   AIC  AICc   BIC ar_roots  ma_roots
##   <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>    <list>
## 1 lag0    0.265   -28.3  66.6  68.3  75.0 <cpl [2]> <cpl [0]>
## 2 lag1    0.209   -24.0  58.1  59.9  66.5 <cpl [1]> <cpl [1]>
## 3 lag2    0.215   -24.0  60.0  62.6  70.2 <cpl [1]> <cpl [1]>
## 4 lag3    0.206   -22.2  60.3  65.0  73.8 <cpl [1]> <cpl [1]>
```

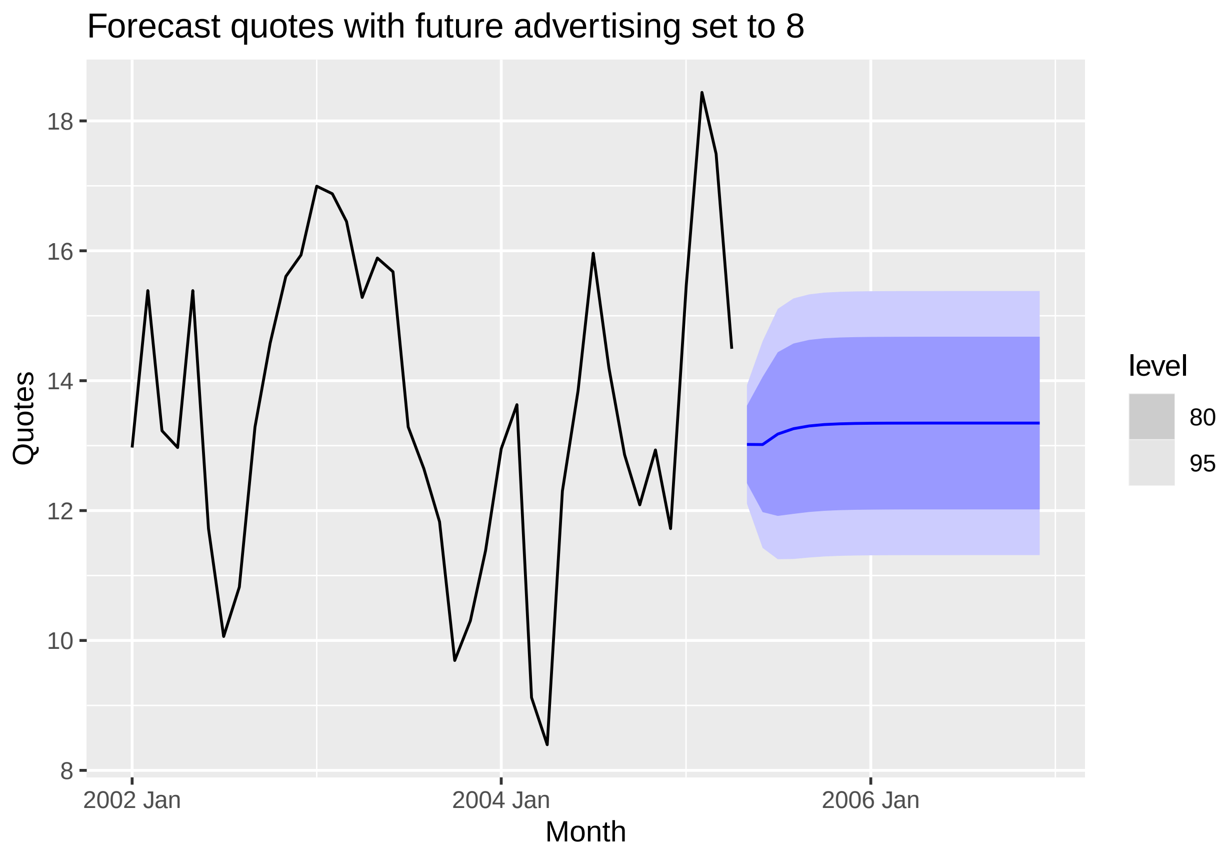
The best model (with the smallest AICc value) is lag1 with two predictors; that is, it includes advertising only in the current month and the previous month. So we now re-estimate that model, but using all the available data.

```
fit_best <- insurance |>
  model(ARIMA(Quotes ~ pdq(d = 0) +
              TVadverts + lag(TVadverts)))
report(fit_best)
```

```
## Series: Quotes
## Model: LM w/ ARIMA(1,0,2) errors
##
## Coefficients:
##          ar1     ma1     ma2  TVadverts  lag(TVadverts)  intercept
##       0.5123  0.9169  0.4591     1.2527          0.1464     2.1554
## s.e.  0.1849  0.2051  0.1895     0.0588          0.0531     0.8595
##
## sigma^2 estimated as 0.2166:  log likelihood=-23.94
## AIC=61.88   AICc=65.38   BIC=73.7
```

We can calculate forecasts using this model if we assume future values for the advertising variable. If we set the future monthly advertising to 8 units, we get the forecasts in Figure 10.13.

```
insurance_future <- new_data(insurance, 20) |>
  mutate(TVadverts = 8)
fit_best |>
  forecast(insurance_future) |>
  autoplot(insurance) +
  labs(
    y = "Quotes",
    title = "Forecast quotes with future advertising set to 8"
  )
```


Forecast quotes with future advertising set to 8
18
16
14
12
10
8
Quotes
2002 Jan
2004 Jan
2006 Jan
Month
level
80
95