Applied Economic Forecasting

4. Time Series Regressions

Section 1

The linear model with time series

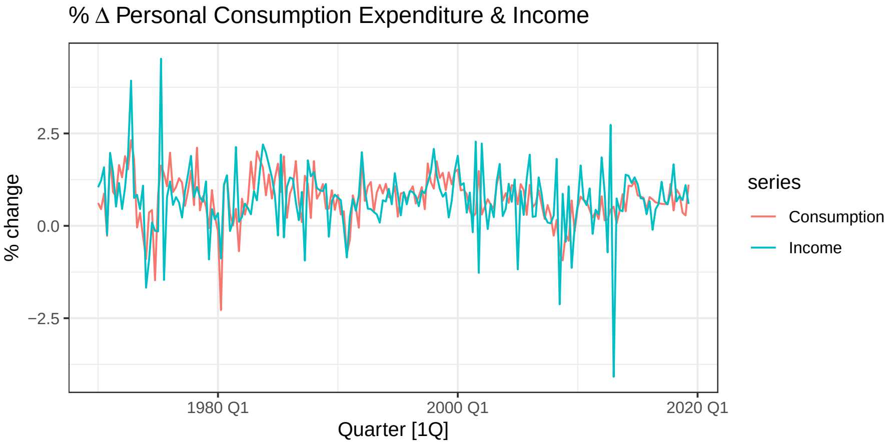
Multiple regression and forecasting

$$y_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \cdots + \beta_k x_{k,t} + \varepsilon_t.$$

- y_t is the variable we want to predict:
  - the "response" variable, the regressand, the explained variable, the dependent variable
- Each $x_{j,t}$ is numerical and is called a "predictor". Assumed to be known for all past and future times.
- The coefficients $\beta_1, \ldots, \beta_k$ measure the **marginal effects** of each predictor holding all other predictors in the model unchanged (constant).
- ε_t is a white noise error term and $\varepsilon \sim \mathbf{N}(0, \sigma^2)$.
- Even if $x_{j,t}$ does not **cause** y_t, the correlation between the two is useful for forecasting.

Example: US consumption expenditure

```
us_change %>% select(Consumption, Income) %>%
  pivot_longer(-Quarter, names_to = "series") %>%
  autoplot(value) +
  labs(y = "% change",
      title = bquote("%"~Delta~"Personal Consumption Expenditure & Income"))
```



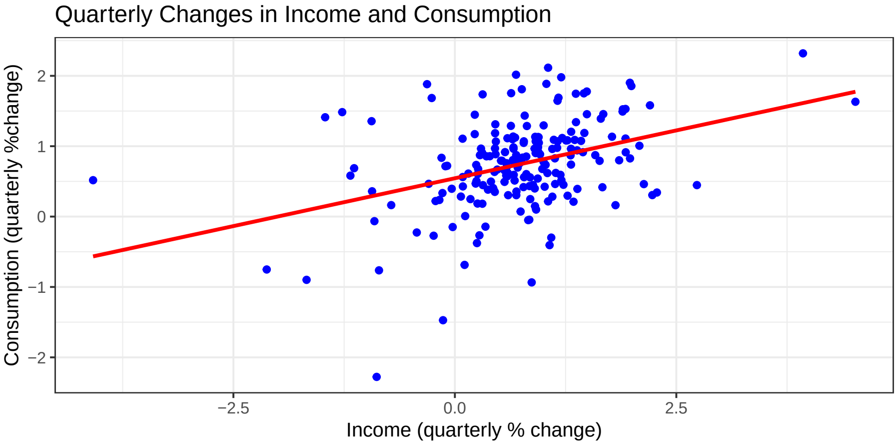
Example: US consumption expenditure

```
fit.cons <- us_change %>%
  model(TSLM(Consumption ~ Income))
fit.cons%>% report()
```

```
## Series: Consumption
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.58236 -0.27777  0.01862  0.32330  1.42229
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.54454    0.05403  10.079  < 2e-16 ***
## Income       0.27183    0.04673   5.817  2.4e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5905 on 196 degrees of freedom
## Multiple R-squared: 0.1472,  Adjusted R-squared: 0.1429
## F-statistic: 33.84 on 1 and 196 DF, p-value: 2.4022e-08
```

Example: US consumption expenditure

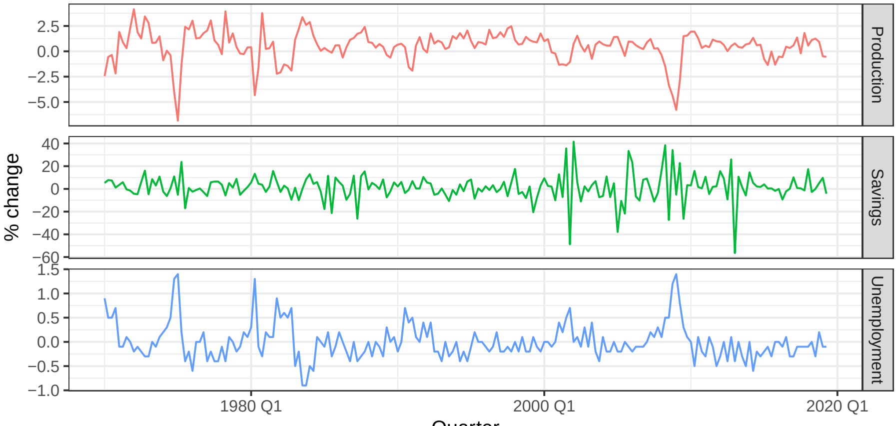
```
us_change %>% ggplot(aes(x = Income, y = Consumption)) +
  geom_point(col = "blue") +
  geom_smooth(method = "lm", se = FALSE, col = "red") +
  labs(title = "Quarterly Changes in Income and Consumption",
       y="Consumption (quarterly %change)",
       x = "Income (quarterly % change)")
```



Example: US consumption expenditure

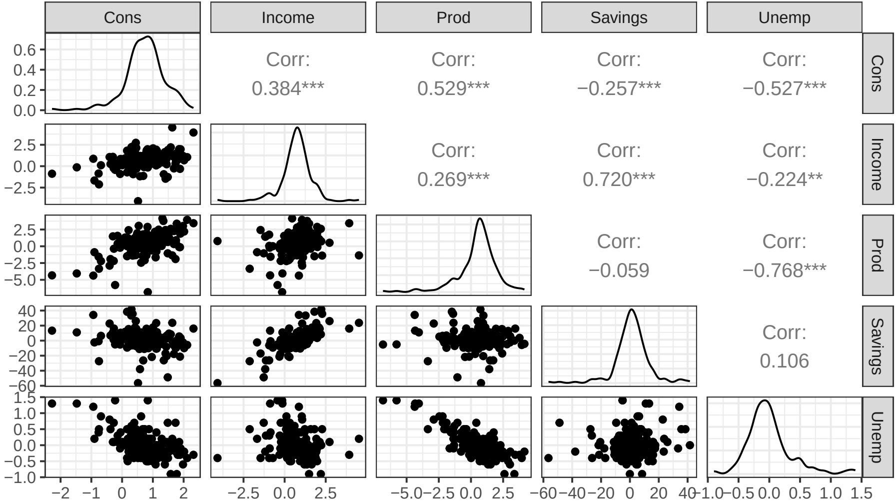
How about additional predictors?

```
us_change %>% select(Production:Unemployment) %>%
  pivot_longer(-Quarter, names_to = "series") %>%
  ggplot(aes(x = Quarter, y = value, colour = series)) +
  geom_line() + facet_grid(series ~., scales = "free_y") +
  theme(legend.position = "none") +
  labs(y = "% change")
```



Example: US consumption expenditure

```
us_change %>% GGally::ggpairs(columns = 2:ncol(us_change),
  columnLabels = c("Cons", "Income", "Prod", "Savings", "Unemp"))
```



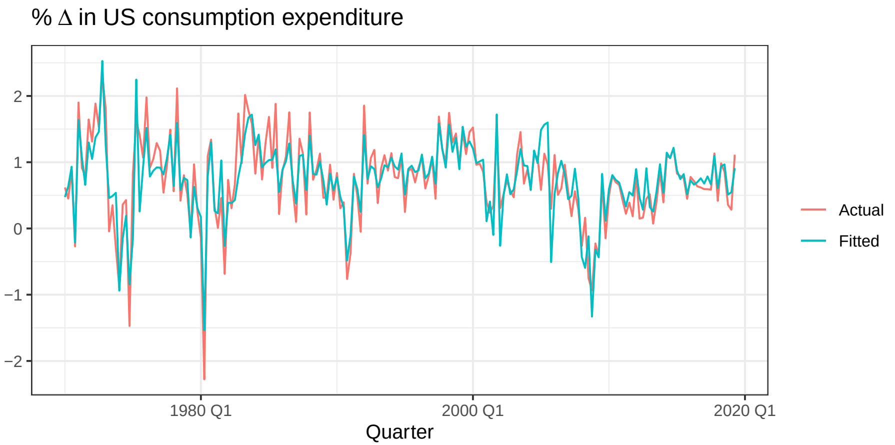
Assumptions

- The relationship between our variables (in reality) is truly linear.
- The errors are mean zero (otherwise we have a systematic bias in our forecasts)
- The errors are not autocorrelated (otherwise our forecasts are inefficient– there is more useful information remaining in residuals that should have been included in the model.)
- The errors are uncorrelated with the regressors (otherwise, there is information in the error (such as an omitted variable) that should have been included in the model.)
- **For our prediction Intervals and hypothesis testing:** the errors are normally distributed with a constant variance, σ^2.

```r
fit.consMR <- us_change %>% model(tslm = TSLM(Consumption ~
              Income + Production + Unemployment + Savings))
fit.consMR %>% report()
```

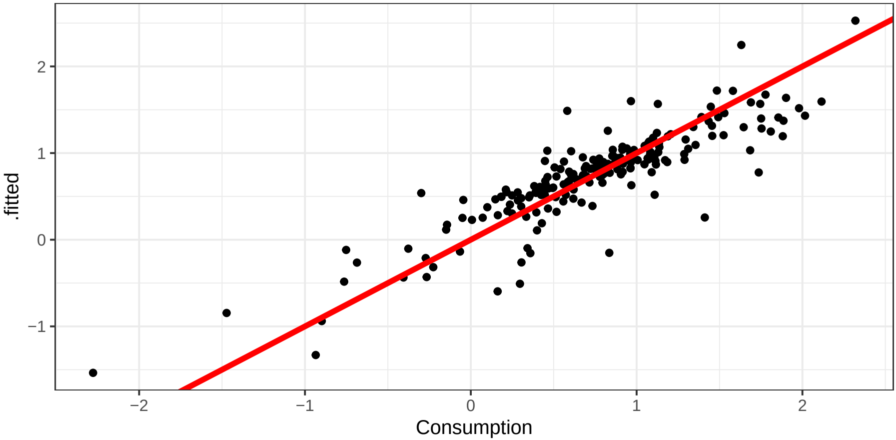
```
## Series: Consumption
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.90555 -0.15821 -0.03608  0.13618  1.15471
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.253105   0.034470   7.343 5.71e-12 ***
## Income        0.740583   0.040115  18.461  < 2e-16 ***
## Production    0.047173   0.023142   2.038   0.0429 *
## Unemployment -0.174685   0.095511  -1.829   0.0689 .
## Savings      -0.052890   0.002924 -18.088  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3102 on 193 degrees of freedom
## Multiple R-squared: 0.7683,	Adjusted R-squared: 0.7635
## F-statistic:   160 on 4 and 193 DF, p-value: < 2.22e-16
```

```r
fit.consMR %>% augment() %>% ggplot(aes(x = Quarter)) +
  geom_line(aes(y = Consumption, col = "Actual")) +
  geom_line(aes(y = .fitted, col = "Fitted")) +
  labs(y = NULL,
       title = bquote('%'~Delta~ "in US consumption expenditure")) +
  guides(colour=guide_legend(title=NULL))
```



Actual vs Predicted

```
fit.consMR %>% augment() %>%
  ggplot(aes(x = Consumption, y = .fitted)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, col = "red", size = 1.5)
```



Goodness of Fit: R^2

A common way to summarize how well a linear regression model fits the data is via the coefficient of determination, or R^2.

R^2 tells us how much of the variation in our dependent variable ($\mathtt{y}$) is explained by the regressors (x_js).

$$R^2 = \frac{\sum_{t=1}^{T} (\hat{y}_t - \bar{y})^2}{\sum_{t=1}^{T} (y_t - \bar{y})^2} = \frac{\text{Explained (Regression) Sum of Squares}}{\text{Total Sum of Squares}}$$

Goodness of Fit: R^2

Assuming that the model has an intercept:

- If the predictions are close to the actual values, we would expect R^2 to be close to 1.
- If the predictions are unrelated to the actual values, then $R^2 = 0$

In all cases, $0 \leq R^2 \leq 1$.

We must be careful when comparing models on the basis of R^2!

- The value of R^2 will **never decrease** when adding an extra predictor to the model. This can lead to over-fitting.
- We could be dealing with a "spurious" regression problem.

Section 2

Diagnostic Tests

Evaluating Regression Models

The differences between the observed y values and the corresponding fitted $\hat{y}$ values are the training-set errors or "residuals" defined as,

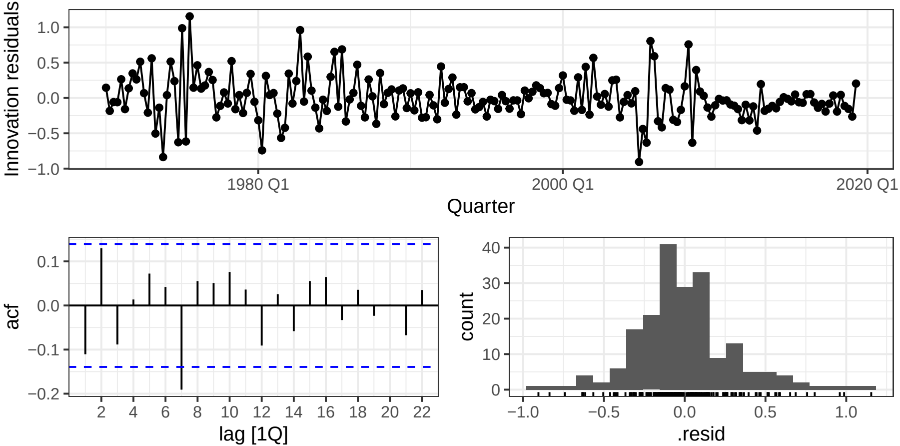
$$\begin{aligned} e_t &= y_t - \hat{y}_t \\ &= y_t - \hat{\beta}_0 - \hat{\beta}_1 x_{1,t} - \hat{\beta}_2 x_{2,t} - \cdots - \hat{\beta}_k x_{k,t} \end{aligned}$$

These residuals have the following properties:

$$\sum_{t=1}^{T} e_t = 0 \quad \text{and} \quad \sum_{t=1}^{T} x_{k,t} e_t = 0 \quad \text{for all } k.$$

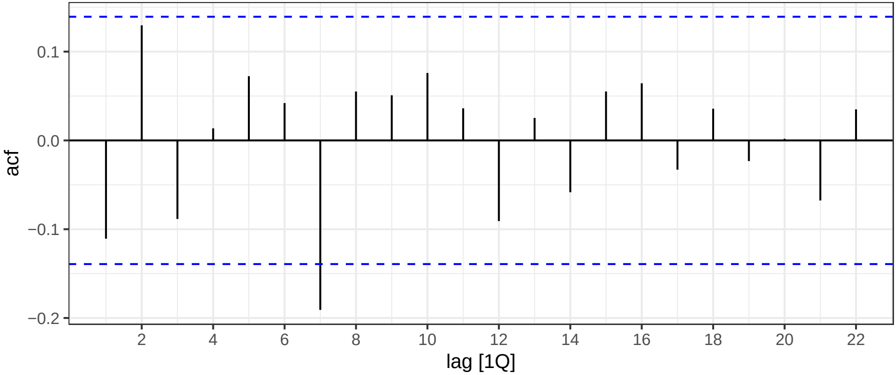
gg_tsresiduals

```
fit.consMR %>% gg_tsresiduals()
```



ACF of residuals

Recall that we would like for there to be no autocorrelation in the residuals. If there is, our model forecasts are inefficient and there is some useful information (that could improve our forecasts) remaining in the residuals.



Lag 7 is potentially problematic. Conclusion from L-B Test??

```r
fit.consMR %>% augment() %>%
  features(.innov, ljung_box, lag = 8)
```

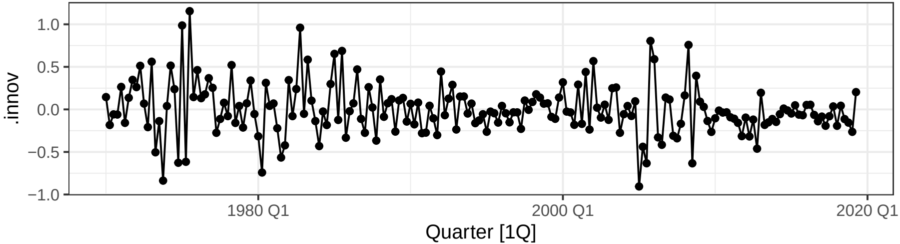
```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 tslm      17.1    0.0290
```

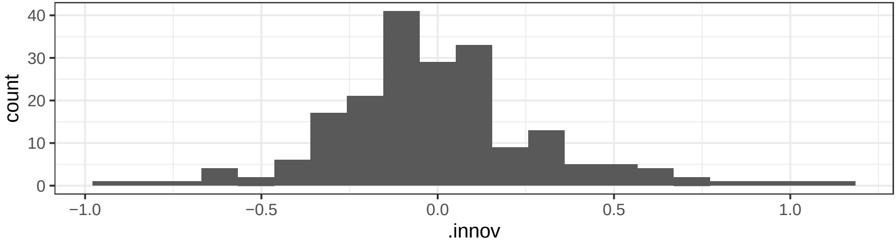
Quick Notes:

Forecasts from a model with autocorrelated errors are still unbiased, and so are not "wrong", but they will usually have larger prediction intervals than they need to.

Therefore we should always look at an ACF plot of the residuals.

Histogram & Time Plot





Histogram & Time Plot

Timeplot: shows changing variability across time » Heteroskedasticity » affects prediction intervals.

Histogram: shows skewness/non-normality » affects the prediction intervals.

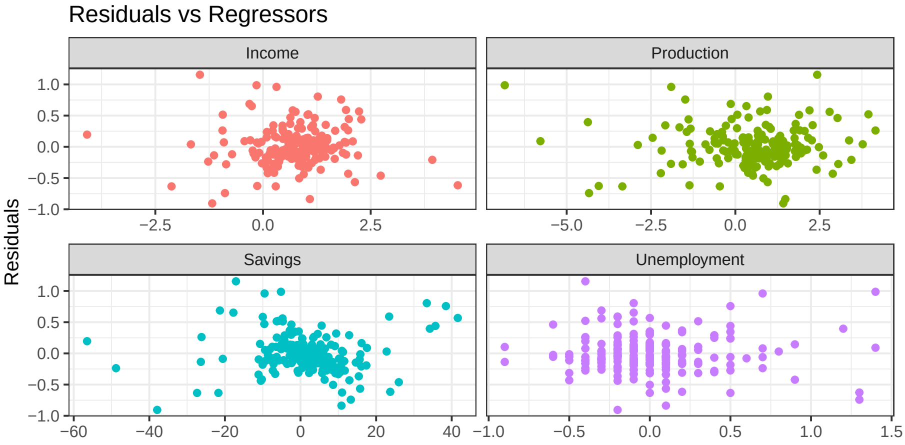
Residual plots

- Useful for spotting outliers and whether the linear model was appropriate.
- Scatterplot of residuals ε_t against each predictor $x_{j,t}$.
- Scatterplot residuals against the fitted values $\hat{y}_t$
- Expect to see scatterplots resembling a horizontal band with no values too far from the band and no patterns such as curvature or increasing/decreased spread.

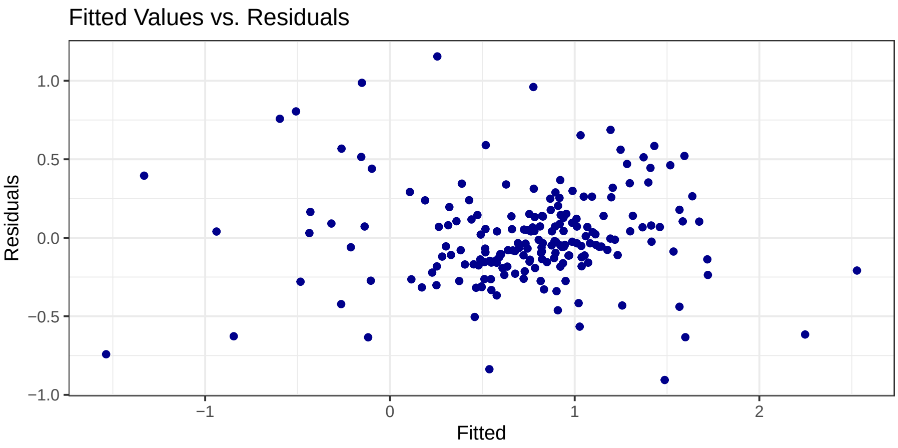
Residual patterns

1. If a plot of the residuals vs any predictor **in** the model shows a pattern, then the relationship is nonlinear.
2. If a plot of the residuals vs any predictor **not** in the model shows a pattern, then the predictor should be added to the model.
3. If a plot of the residuals vs fitted values shows a pattern, then there is *heteroskedasticity* in the errors. (Could try a transformation.)

```r
us_change %>% left_join(residuals(fit.consMR), by = "Quarter") %>%
  pivot_longer(Income:Unemployment, names_to = "regressor",
               values_to = "x") %>%
  ggplot(aes(x = x, y = .resid)) + geom_point(aes(colour=regressor)) +
  facet_wrap(regressor~., scales= "free_x") + theme(legend.pos = "none") +
  labs(x = NULL, y = "Residuals", title = "Residuals vs Regressors")
```



```
fit.consMR %>% augment() %>%
  ggplot(aes(x = .fitted, y = .innov)) +
  geom_point(col = "darkblue") +
  labs(x = "Fitted", y = "Residuals",
       title = "Fitted Values vs. Residuals")
```



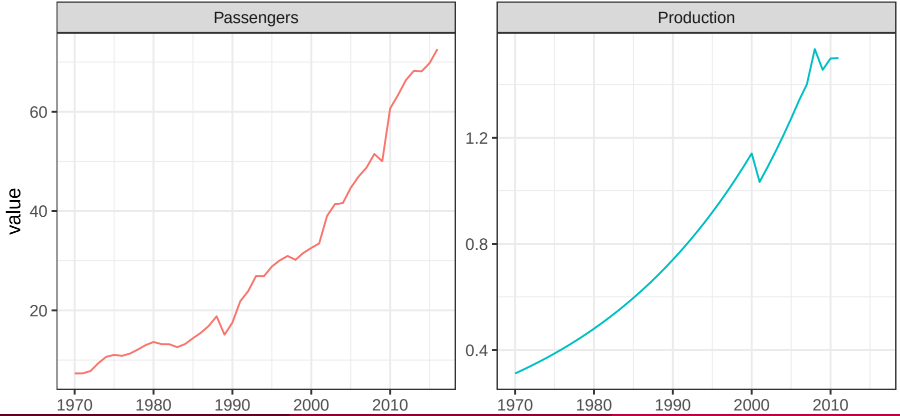
Spurious Regressions

In time series, we often have data that is "non-stationary"– the series does not fluctuate around a constant mean or with a constant variance.

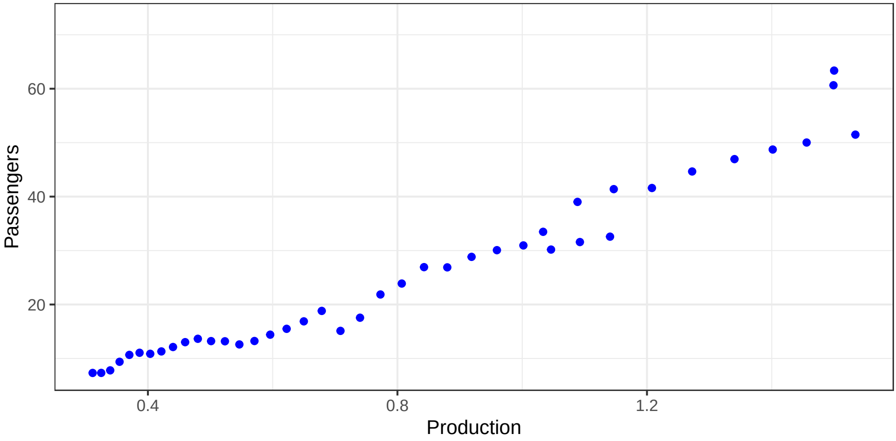
Because of this, we can have two variables that appear to be highly correlated simply because they have the same patterns (say both are trending). Regressing non-stationary time series can lead to "spurious regressions" (think high R^2 but an otherwise nonsensical relationship and results).

Spurious Regressions

```
spur <- aus_airpassengers %>% left_join(guinea_rice, by = "Year")
spur %>% pivot_longer(-Year, names_to = "vars",
                      values_to="value") %>%
  ggplot() + geom_line(aes(x = Year, y = value, colour = vars)) +
  facet_wrap(.~vars, scales = "free_y") +
  theme(legend.pos = "none")
```



```r
spur %>%
  ggplot() +
  geom_point(aes(y = Passengers, x = Production),
             col = "blue")
```

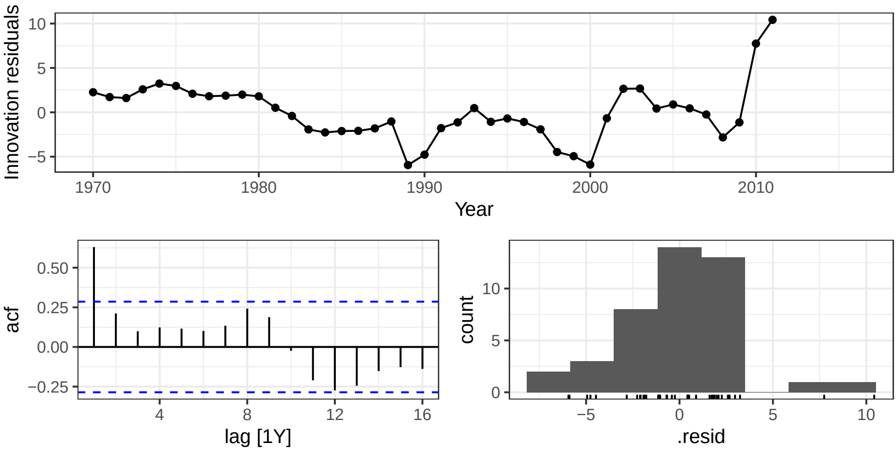


Spurious Regressions

```
spur.fit <- spur %>% model(spur = TSLM(Passengers ~ Production))
spur.fit %>% report()
```

```
## Series: Passengers
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9448 -1.8917 -0.3272  1.8620 10.4210
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -7.493      1.203  -6.229 2.25e-07 ***
## Production    40.288      1.337  30.135  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.239 on 40 degrees of freedom
## Multiple R-squared: 0.9578,  Adjusted R-squared: 0.9568
## F-statistic: 908.1 on 1 and 40 DF, p-value: < 2.22e-16
```

```
spur.fit %>% gg_tsresiduals()
```



Key Takeaway: just because two series move together does not mean they are related!

Section 3

Useful predictors for linear models

Linear trend

Given the general form:

$$y_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \cdots + \beta_k x_{k,t} + \varepsilon_t.$$

We can introduce a trend to the model by including $x_{j,t} = t$ as a regressor,

$$y_t = \beta_0 + \beta_1 t + \varepsilon$$

where $t = 1, 2, \ldots, T$

A trend variable can be specified in the `TSLM()` function using `trend()` as a predictor.

Why would you want to include a trend?

Dummy variables

If a categorical variable takes only two values (e.g., 'Yes' or 'No'), then an equivalent numerical variable can be constructed taking value `1` if `yes` and `0` if `No`. This is called a **dummy variable**.

- A dummy variable can also be used to account for an outlier in the data. Rather than omit the outlier, a dummy variable removes its effect.
- `TSLM()` will automatically handle creating dummies if we were to specify a factor variable as a predictor.

Seasonal Dummy variables

Suppose we have quarterly retail sales data and suspect that there might be seasonality in our data (e.g. Q4 might have unusually high sales figures since we have Thanksgiving, Black Friday, Cyber Monday, and Christmas in Nov. & Dec.)

We could create our quarterly dummy variables as follows:

Quarter	$Q_{1,t}$	$Q_{2,t}$	$Q_{3,t}$
2000 Q1	1	0	0
2000 Q2	0	1	0
2000 Q3	0	0	1
2000 Q4	0	0	0
2001 Q1	1	0	0
2001 Q2	0	1	0
2001 Q3	0	0	1
2001 Q4	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Dummy variables

Beware of the dummy variable trap!

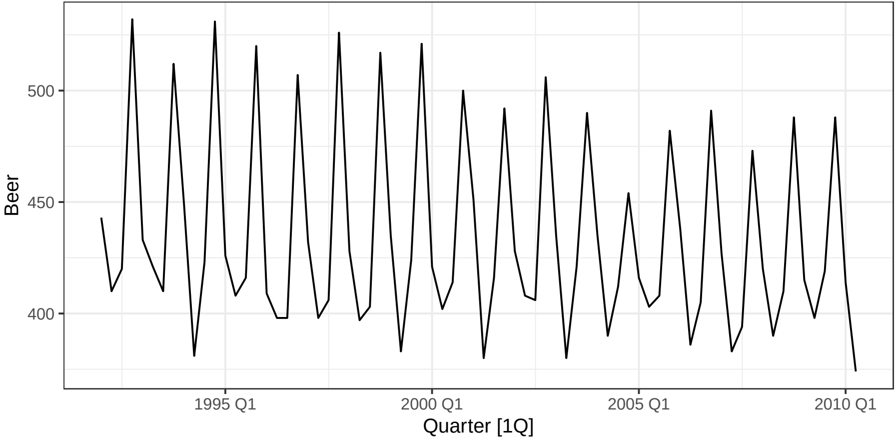
If there are more than two categories, then the variable can be coded using several dummy variables (one fewer than the total number of categories).

- Using one dummy for each category gives too many dummy variables!
- The regression will then be singular and inestimable.
- Either omit the constant, or omit the dummy for one category.
- If we omit one category, the **coefficients of the remaining dummies are relative to that omitted category**.

Australian Beer Production

```
beer <- aus_production %>% select(Beer) %>% filter_index("1992 Q1"~.)

beer %>% autoplot(Beer)
```



Manual Approach

```r
beer.full <- beer %>% mutate(t = row_number(),
         Q2 = ifelse(quarter(Quarter)==2,1,0),
         Q3 = ifelse(quarter(Quarter)==3,1,0),
         Q4 = ifelse(quarter(Quarter)==4,1,0))

beer.full %>% lm(Beer ~ t + Q2 + Q3 + Q4, data = .) %>%
  summary()
```

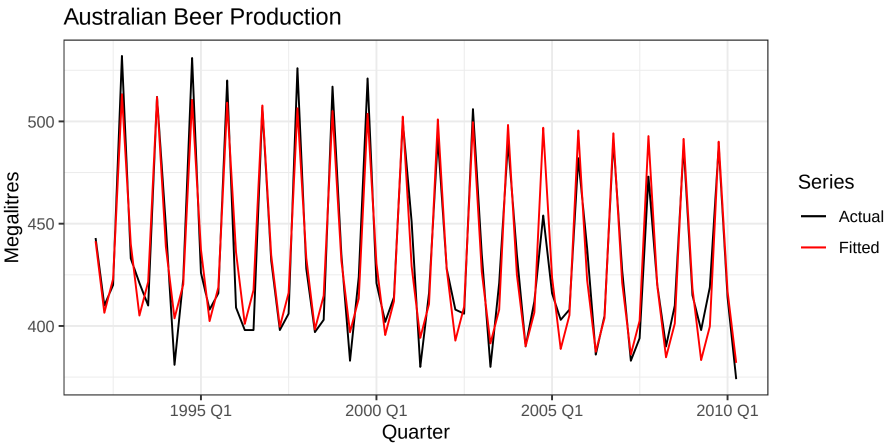
```
##
## Call:
## lm(formula = Beer ~ t + Q2 + Q3 + Q4, data = .)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -42.903  -7.599  -0.459   7.991  21.789
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 441.80044    3.73353 118.333  < 2e-16 ***
## t            -0.34027    0.06657  -5.111 2.73e-06 ***
## Q2          -34.65973    3.96832  -8.734 9.10e-13 ***
## Q3          -17.82164    4.02249  -4.430 3.45e-05 ***
## Q4           72.79641    4.02305  18.095  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.23 on 69 degrees of freedom
## Multiple R-squared:  0.9243, Adjusted R-squared:  0.9199
## F-statistic: 210.7 on 4 and 69 DF,  p-value: < 2.2e-16
```

Using the TSLM() function

```
fit.beer <- beer %>% model(TSLM(Beer~ trend() + season()))
fit.beer %>% report()
```

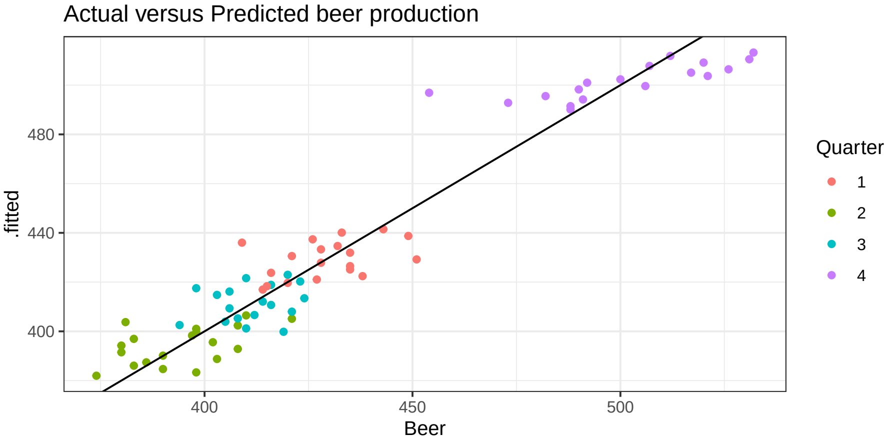
```
## Series: Beer
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -42.9029  -7.5995  -0.4594   7.9908  21.7895
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    441.80044    3.73353 118.333  < 2e-16 ***
## trend()         -0.34027    0.06657  -5.111 2.73e-06 ***
## season()year2  -34.65973    3.96832  -8.734 9.10e-13 ***
## season()year3  -17.82164    4.02249  -4.430 3.45e-05 ***
## season()year4   72.79641    4.02305  18.095  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.23 on 69 degrees of freedom
## Multiple R-squared: 0.9243,  Adjusted R-squared: 0.9199
## F-statistic: 210.7 on 4 and 69 DF, p-value: < 2.22e-16
```

```r
fit.beer %>% augment() %>% ggplot(aes(x = Quarter)) +
  geom_line(aes(y = Beer, colour = "Actual")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  scale_color_manual(breaks = c("Actual", "Fitted"),
                     values = c("black", "red")) +
  labs(title = "Australian Beer Production", y = "Megalitres") +
  guides(colour = guide_legend(title = "Series"))
```



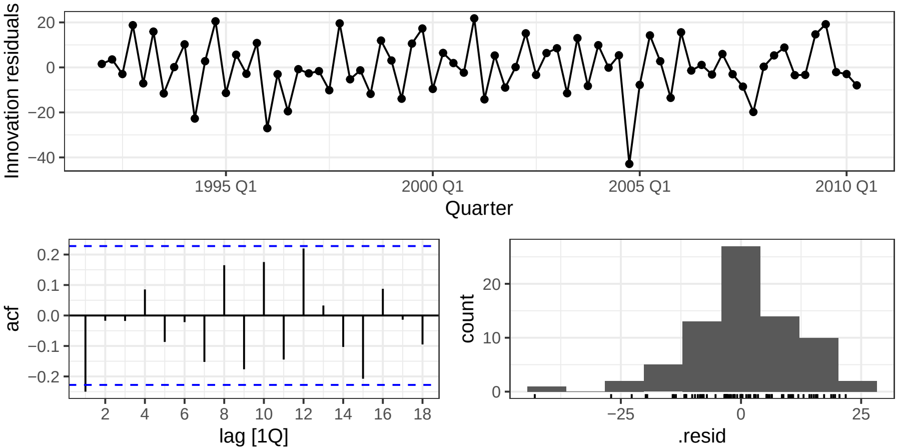
Australian Beer Production

```r
fit.beer %>% augment() %>%
  ggplot(aes(y = .fitted, x = Beer)) +
  geom_point(aes(colour = factor(quarter(Quarter)))) +
  guides(colour = guide_legend(title = "Quarter")) +
  geom_abline(slope = 1, intercept = 0, col = "black") +
  labs(title = "Actual versus Predicted beer production")
```



Australian Beer Production

```
fit.beer %>% gg_tsresiduals()
```



Intervention Variables

Other uses of dummy variables in regression models are as follows:

Spikes: An event affects y_t only at time τ . Set $D_t = 1$ if $t = \tau$, otherwise $D_t = 0$. This is usually a one time (or an outlier) event.

```
D <- ifesle(t==tau, 1, 0)
```

Steps: An event has an immediate and permanent effect on the series at and after time τ. Set $D_t = 1$ if $t \geq \tau$, otherwise $D_t = 0$. For example, use of Zoom due to COVID-19?

```
D <- ifesle(t>=tau, 1, 0)
```

Ramp: An event changes the trend of a series linearly at and after time τ. This means that our trend is piecewise linear. Set $D_t = t - \tau$ if $t \geq \tau$, otherwise $D_t = 0$.

```
D <- pmax(0, t - tau)
```

Fourier Series

An alternative to using dummy variables to capture seasonality is using pairs of Fourier terms:

$$s_k(t) = \sin\left(\frac{2\pi kt}{m}\right) \qquad c_k(t) = \cos\left(\frac{2\pi kt}{m}\right)$$

$$y_t = a + bt + \sum_{k=1}^{K}\left[\alpha_k s_k(t) + \beta_k c_k(t)\right] + \varepsilon_t \quad K \leq \frac{m}{2}$$

- where `m` is the seasonal period.
- Every periodic function can be approximated by sums of `sin` and `cos` terms for large enough K.
- Choose K by minimizing AICc.
- Called "harmonic regression"

```
fit <- tslm(y ~ trend + fourier(K))
```

Manual

```
beer.fourier <- beer %>% mutate(
  t  = row_number(),
  cos1 = cos((2*pi*1*t)/4),
  sin1 = sin((2*pi*1*t)/4),
  cos2 = cos((2*pi*2*t)/4),
  sin2 = sin((2*pi*2*t)/4)
)

fits1 <- beer.fourier %>% lm(Beer~ t + cos1 + cos2 + sin1 + si
                    data = .)
fits1 %>% summary()
```

```
##
## Call:
## lm(formula = Beer ~ t + cos1 + cos2 + sin1 + sin2, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -44.023  -7.328  -0.939   8.086  22.181
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.469e+02  2.892e+00 154.542  < 2e-16 ***
## t           -3.416e-01  6.705e-02  -5.095 2.98e-06 ***
## cos1         5.373e+01  2.023e+00  26.554  < 2e-16 ***
## cos2         1.443e+01  1.767e+00   8.163 1.11e-11 ***
## sin1         8.877e+00  2.025e+00   4.384 4.15e-05 ***
## sin2         9.109e+13  2.166e+14   0.420    0.675
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.3 on 68 degrees of freedom
## Multiple R-squared:  0.9245, Adjusted R-squared:  0.919
## F-statistic: 166.6 on 5 and 68 DF,  p-value: < 2.2e-16
```

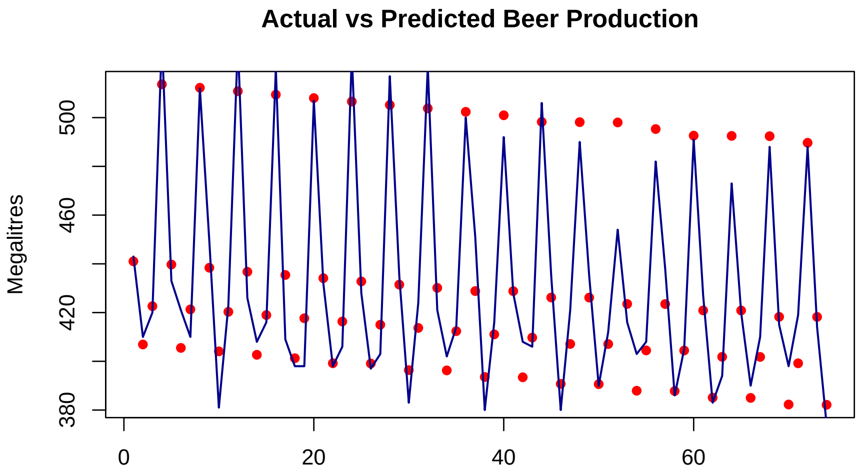
via TSLM()

```
beer %>% model(TSLM(Beer ~ trend() + fourier(K = 2))) %>%
  report()
```

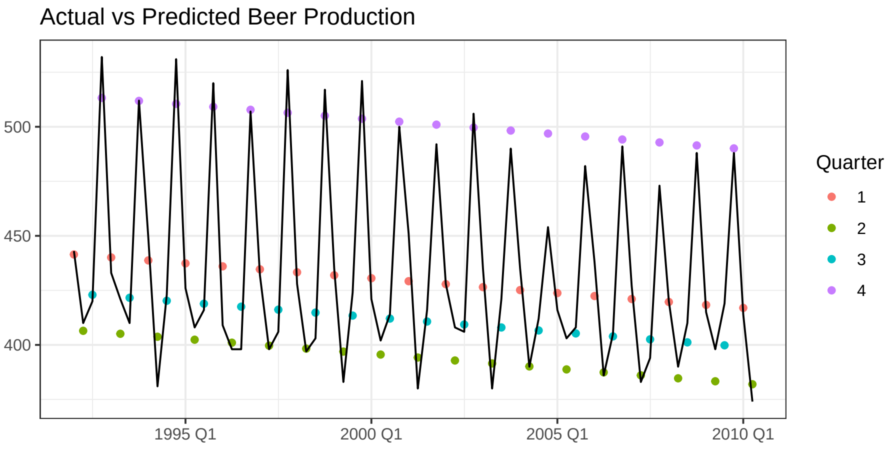
```
## Series: Beer
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -42.9029  -7.5995  -0.4594   7.9908  21.7895
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        446.87920    2.87321 155.533  < 2e-16 ***
## trend()             -0.34027    0.06657  -5.111 2.73e-06 ***
## fourier(K = 2)C1_4   8.91082    2.01125   4.430 3.45e-05 ***
## fourier(K = 2)S1_4 -53.72807    2.01125 -26.714  < 2e-16 ***
## fourier(K = 2)C2_4 -13.98958    1.42256  -9.834 9.26e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.23 on 69 degrees of freedom
## Multiple R-squared: 0.9243,  Adjusted R-squared: 0.9199
## F-statistic: 210.7 on 4 and 69 DF, p-value: < 2.22e-16
```

Notice the changing Intercepts? Which line corresponds to which quarter?

```
plot(fits1$fitted.values,
     col = "red", type = "p", cex = 1, pch = 16,
     main = "Actual vs Predicted Beer Production", ylab = "Megalitres", xla
lines(beer.fourier$Beer, col = "darkblue", lwd = 1.5)
```



```r
beer %>% model(TSLM(Beer ~ trend() + fourier(K = 2))) %>%
  augment() %>% ggplot(aes(x = Quarter)) +
  geom_point(aes(y = .fitted, col = factor(quarter(Quarter)))) +
  geom_line(aes(y = Beer)) +
  guides(colour = guide_legend(title = "Quarter")) +
  labs(title = "Actual vs Predicted Beer Production", y = NULL, x = NULL)
```



Section 4

Selecting predictors and forecast evaluation

Selecting predictors

- When there are many predictors, how should we choose which ones to use?
- We need a way of comparing two competing models.

What not to do!

- Plot y against a particular predictor (x_j) and if it shows no noticeable relationship, drop it.
- Do a multiple linear regression on all the predictors and drop all variables whose p values are greater than 0.05. This ignores potential correlation between our predictors.
- Maximize R^2 or minimize MSE.

The glance() function

```
fit.consMR %>% glance() %>% t() %>% knitr::kable(digits = 3)
```

.model	tslm
r_squared	0.7682829
adj_r_squared	0.7634805
sigma2	0.0962325
statistic	159.9781
p_value	3.92929e-60
df	5
log_lik	-46.6599
AIC	-456.5799
AICc	-456.1401
BIC	-436.8503
CV	0.1038972
deviance	18.57287
df.residual	193
rank	5

Comparing regression models

Computer output for regression will always give the R^2 value. This is a useful summary of the model. However ...

- R^2 does not allow for "degrees of freedom''.
- Adding *any* variable tends to increase the value of R^2, even if that variable is irrelevant.
- using R^2 to determine whether a model will give good predictions will lead to overfitting.

To overcome this problem, we can use *adjusted* R^2:

Adjusted R^2

$$\bar{R}^2 = 1 - (1 - R^2)\frac{T-1}{T-k-1}$$

where $k =$ no. predictors and $T =$ no. observations.

Maximizing $\bar{R}^2$ is equivalent to minimizing $\hat{\sigma}_e$.

$$\hat{\sigma}_e = \sqrt{\frac{1}{T-k-1}\sum_{t=1}^{T} e_t^2}$$

Akaike's Information Criterion

$$\text{AIC} = -2\log(\mathcal{L}) + 2(k+2)$$

where $\mathcal{L}$ is the likelihood and k is the number of predictors in the model.

Alternatively,

$$\text{AIC} = T\log\left(\frac{\text{SSE}}{T}\right) + 2(k+2),$$

- This is a *penalized likelihood* approach.
- *Minimizing* the AIC gives the best model for prediction.
- AIC penalizes terms more heavily than $\bar{R}^2$.
- Minimizing the AIC is asymptotically (when $T \to \infty$) equivalent to minimizing MSE via leave-one-out cross-validation.

Corrected AIC

For small values of T, the AIC tends to select too many predictors, and so a bias-corrected version of the AIC has been developed.

$$\text{AIC}_{\text{C}} = \text{AIC} + \frac{2(k+2)(k+3)}{T-k-3}$$

As with the AIC, the AIC_{C} should be **minimized**.

Schwarz's Bayesian Information Criterion

$$\text{BIC} = -2\log(\mathcal{L}) + (k+2)\log(T)$$

where $\mathcal{L}$ is the likelihood and k is the number of predictors in the model.

Alternatively,

$$\text{BIC} = T\log\left(\frac{\text{SSE}}{T}\right) + (k+2)\log(T)$$

- BIC penalizes terms more heavily than AIC
- Also called SBIC and SC.
- Minimizing BIC is asymptotically equivalent to leave-v-out cross-validation when $v = T\left[1 - \frac{1}{(\log(T)-1)}\right]$.

Cross-validation

Leave-one-out

1. Remove observation t from the data set, and fit the model using the remaining data. Then compute the error ($e_t^* = y_t - \hat{y}_t$) for the omitted observation.
   - Not the same as the residual because the t^{th} observation was not used in estimating the $\hat{y}_t$.
2. Repeat step 1 for $t = 1, \dots, T$
3. Compute the MSE from $e_1^*, \dots e_T^*$.

```r
selects <- c("adj_r_squared","CV","AIC","AICc", "BIC")
m <- matrix(NA,nrow = 5, ncol = 5)
rownames(m) <- paste0("model", 1:5)
colnames(m) <- selects
model <- m %>% as_tibble(rownames = "rowname")

model[1,2:6] <- us_change %>% model(TSLM(Consumption ~ Income + Production +
  Unemployment + Savings )) %>% glance() %>% select_at(selects)
model[2,2:6] <- us_change %>% model(TSLM(Consumption ~ Income + Production +
  Unemployment)) %>% glance() %>% select_at(selects)
model[3,2:6] <- us_change %>% model(TSLM(Consumption ~ Income + Production +
  Savings)) %>% glance() %>% select_at(selects)
model[4,2:6] <- us_change %>% model(TSLM(Consumption ~ Income + Unemployment +
  Savings)) %>% glance() %>% select_at(selects)
model[5,2:6] <- us_change %>% model(TSLM(Consumption ~ Production + Unemployment +
  Savings)) %>% glance() %>% select_at(selects)

knitr::kable(model,digits = 3)
```

rowname	adj_r_squared	CV	AIC	AICc	BIC
model1	0.763	0.104	-456.580	-456.140	-436.850
model2	0.366	0.271	-262.274	-261.961	-245.833
model3	0.761	0.105	-455.178	-454.865	-438.736
model4	0.760	0.104	-454.362	-454.050	-437.921
model5	0.349	0.279	-257.138	-256.826	-240.697

Section 5

Forecasting with regression

Ex-ante versus ex-post forecasts

- *Ex ante forecasts* are made using only information available in advance.
  - require forecasts of predictors
- *Ex post forecasts* are made using later information on the predictors.
  - useful for studying behaviour of forecasting models.
- trend, seasonal and calendar variables are all known in advance, so these don't need to be forecasted.

Australian Beer production

```
beer.fore <- beer %>%
  model(TSLM(Beer ~ trend() + season())) %>% forecast()

beer.fore %>% autoplot(beer) +
  labs(x = NULL, y = "megalitres",
      title = "Forecasts of beer production using regression")
```

Scenario based forecasting

- Assumes possible scenarios for the predictor variables
- Prediction intervals for scenario based forecasts do not include the uncertainty associated with the future values of the predictor variables.

Example

A US policy maker is interested in comparing the predicted change in consumption when there is a *constant growth of 1% and 0.5% respectively for income and savings with no change in the employment rate*, versus a *respective decline of 1% and 0.5%*, for **each of the next four quarters (out-of-sample here)**.

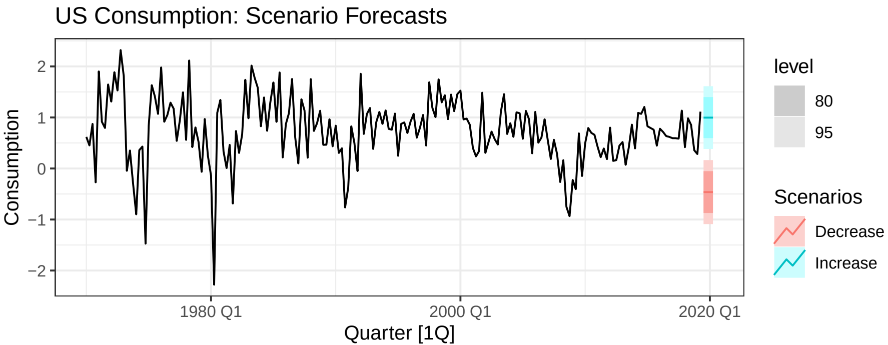
```
fit.consBest <- us_change %>%
  model(TSLM(Consumption ~ Income + Savings + Unemployment))

h <- 4

future <- scenarios(
  Increase = new_data(us_change, h) %>%
    mutate(Income = 1, Savings = 0.5, Unemployment = 0),
  Decrease = new_data(us_change, h) %>%
    mutate(Income = -1, Savings = -0.5, Unemployment = 0),
  names_to = "Scenarios")

fc <- forecast(fit.consBest, new_data = future)
```

```r
us_change %>% autoplot(Consumption) + autolayer(fc) +
  labs(title = "US Consumption: Scenario Forecasts")
```



Note

- The prediction intervals for scenario based forecasts do not include the uncertainty associated with the future values of the predictor variables.
- They assume that the values of the predictors are known in advance.

Section 6

Nonlinear Regressions

Log-Log Model

$$\log y_t = \beta_0 + \beta_1 \log x_t + \varepsilon_t$$

While this provides a non-linear functional form, the model is still **linear in the parameters**.

In this model, the slope, β_1 can be interpreted as an elasticity. In fact, β_1 is the anticipated percentage change in y resulting from a 1% increase in the x variable.

How?

$$\frac{d \log y}{dx} = \beta_1 \frac{\log(x)}{dx}$$
$$\Rightarrow \frac{1}{y} \cdot \frac{dy}{dx} = \beta_1 \cdot \frac{1}{x}$$
$$\Rightarrow \beta_1 = \frac{x}{y} \cdot \frac{dy}{dx}$$

Other log tranformations

Log-linear form is specified by only transforming the forecast variable.

$$\log y_t = \beta_0 + \beta_1 x_t + \varepsilon_t$$

Linear-log form is obtained by transforming the predictor.

$$y_t = \beta_0 + \beta_1 \log x_t + \varepsilon_t$$

Working with zeros

- Recall that in order to perform a logarithmic transformation to a variable, all of its observed values must be greater than zero.
- In the event that variable x contains zeros, we use the transformation

$$\log(x + 1)$$

Nonlinear trend

Piecewise linear trend with bend at $t = \tau$

$$
\begin{aligned}
x_{1,t} &= t \\
x_{2,t} &= \begin{cases} 0 & t < \tau \\ (t - \tau) & t \geq \tau \end{cases}
\end{aligned}
$$

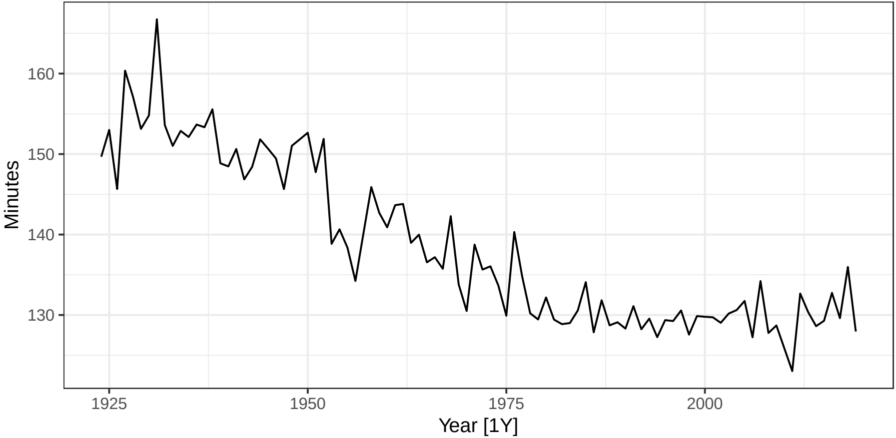
Quadratic or higher order trend

$$x_{1,t} = t, \quad x_{2,t} = t^2, \quad \ldots$$

NOT RECOMMENDED!: to use quadratic or higher order trends in forecasting. When they are extrapolated, the resulting forecasts are often unrealistic.

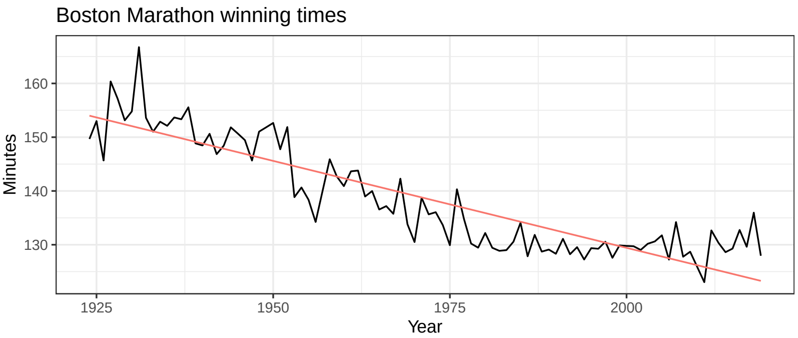
Nonlinear Trend

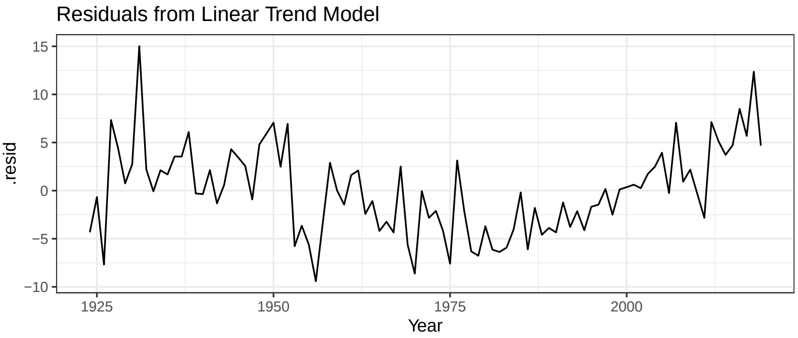
```r
boston_men <- boston_marathon  %>% filter(Year >= 1924)  %>%
  filter(Event == "Men's open division") %>%
  mutate(Minutes = as.numeric(Time)/60)
boston_men %>% autoplot(Minutes)
```



Manually

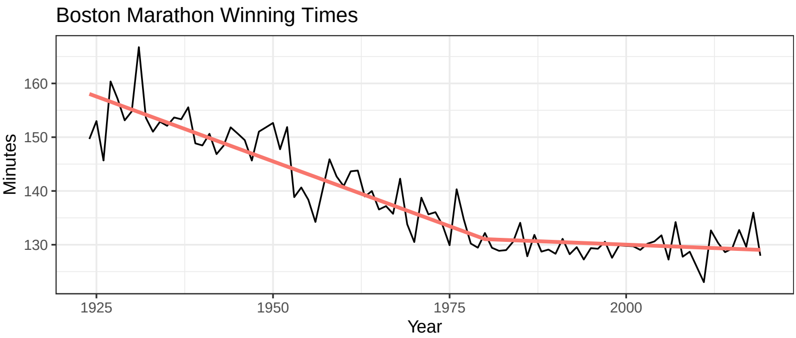
```
b.times <- boston_men %>% mutate(t = row_number(),
                      Dt = pmax(0, Year - 1980))
mod1 <- b.times %>% model(TSLM(Minutes ~ t))
aug.mod1 <- mod1 %>% augment()
gg1 <- aug.mod1 %>% ggplot(aes(x = Year)) +
  geom_line(aes(y = Minutes)) +
  geom_line(aes(y = .fitted, col = "red")) +
  theme(legend.pos = "none") + ggtitle("Boston Marathon winning times")
gg2 <- aug.mod1 %>% ggplot(aes(x = Year, y = .resid)) +
  geom_line() + ggtitle("Residuals from Linear Trend Model")
gridExtra::grid.arrange(gg1,gg2, ncol = 1)
```

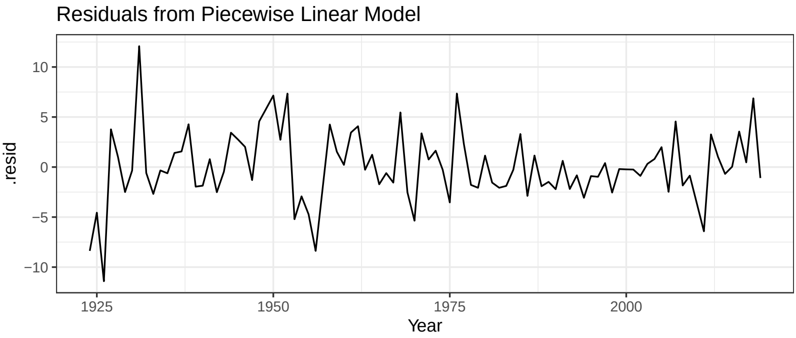

Boston Marathon winning times
Minutes
160
150
140
130
1925
1950
1975
2000
Year


Residuals from Linear Trend Model
.resid
15
10
5
0
−5
−10
1925
1950
1975
2000
Year

Piecewise Linear Model

```
mod2 <- b.times %>% model(TSLM(Minutes ~ t + Dt))
aug.mod2 <- mod2 %>% augment()
gg1 <- aug.mod2 %>% ggplot(aes(x = Year)) +
  geom_line(aes(y = Minutes)) +
  geom_line(aes(y = .fitted, col = "blue"), size = 1.1) +
  theme(legend.pos = "none") + ggtitle("Boston Marathon Winning Times")
gg2 <- aug.mod2 %>% ggplot(aes(x = Year, y = .resid)) +
  geom_line() + ggtitle("Residuals from Piecewise Linear Model")
gridExtra::grid.arrange(gg1,gg2, ncol = 1)
```


Boston Marathon Winning Times
Minutes
160
150
140
130
1925
1950
1975
2000
Year

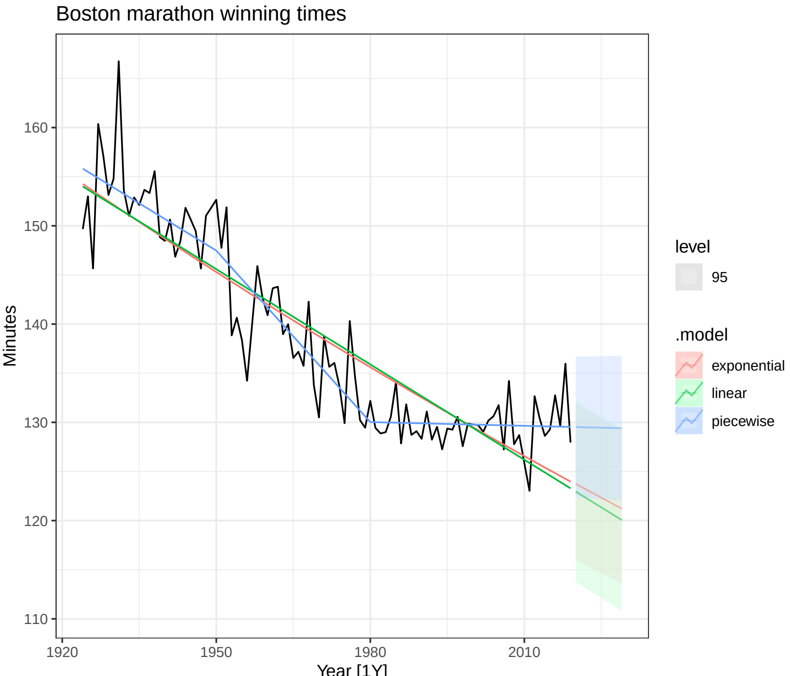

Residuals from Piecewise Linear Model
.resid
10
5
0
−5
−10
1925
1950
1975
2000
Year

Using `TSLM()` function

```
trend_fits <- boston_men %>%
  model(linear = TSLM(Minutes ~ trend()),
        exponential = TSLM(log(Minutes) ~ trend()),
        piecewise = TSLM(Minutes ~ trend(knots = c(1950, 1980))))
trend.fore <- trend_fits %>% forecast(h = 10)

boston_men %>%
  autoplot(Minutes) +
  geom_line(data = fitted(trend_fits),
            aes(y = .fitted, colour = .model)) +
  autolayer(trend.fore, alpha = 0.5, level = 95) +
  labs(y = "Minutes",
       title = "Boston marathon winning times")
```

Using `TSLM()` function



Section 7

Correlation, causation and forecasting

Correlation is not causation

- When x is useful for predicting y, it is not necessarily causing y.
- e.g., predict number of drownings y using number of ice-creams sold x.
- Correlations are useful for forecasting, even when there is no causality.
- Better models usually involve causal relationships (e.g., temperature x and people z to predict drownings y).

Multicollinearity

In regression analysis, multicollinearity occurs when:

- Two predictors are highly correlated (i.e., the correlation between them is close to ± 1).
- A linear combination of some of the predictors is highly correlated with another predictor.
- A linear combination of one subset of predictors is highly correlated with a linear combination of another subset of predictors.

Thinking back to the Dummy variable trap

Suppose you have quarterly data and use four dummy variables, d_1, d_2, d_3, d_4. Then $d_4 = 1 - d_1 - d_2 - d_3$ so there is perfect correlation between d_4 and $d_1 + d_2 + d_3$. Knowing d_1, d_2, d_3 will help us to pefectly predict d_4

Multicollinearity

If multicollinearity exists . . .

- the numerical estimates of coefficients may be wrong (worse in Excel than in a statistics package)
- don't rely on the p-values to determine significance.
- the uncertainty associated with individual regression coefficients will be large. That is the variance is inflated.
- there is no problem with model *predictions* provided the predictors used for forecasting are within the range used for fitting.
- omitting variables can help.
- combining variables can help.

Section 8

Decided to put Dynamic Regressions as seperate topic