Assignment 2

Applied Forecasting in Complex Systems 2023

University of Amsterdam
November, 15, 2023

Student Name	Student_Id
Kyra Dresen	13241380
Mladen Mladenov	15269582
Martin Arnold	15202275

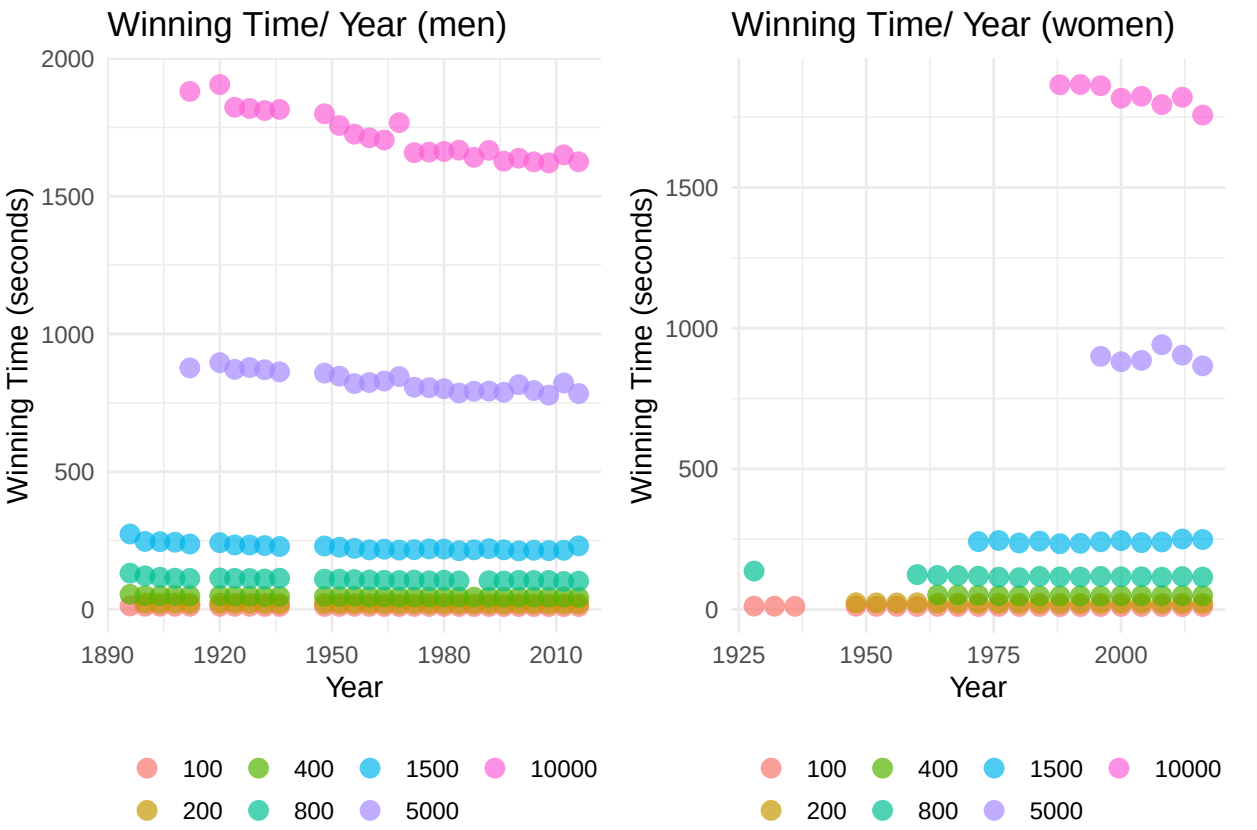
Exercise 1

In this task, we assess the `olympic_running` dataset.

> Data set olympic_running contains the winning times (in seconds) in each Olympic Games sprint, middle-distance and long-distance track events from 1896 to 2016.

1.1) We plot the winning time against the year and describe the main features of the plot

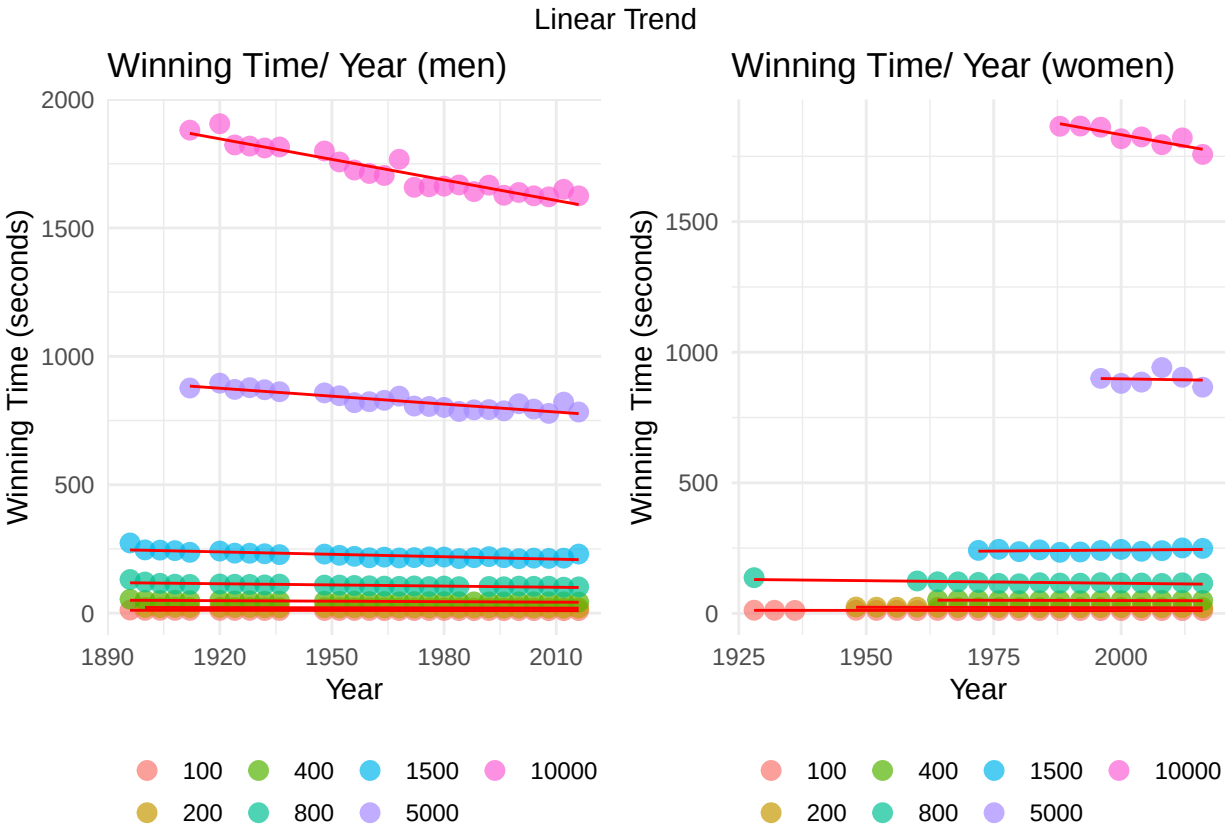
```
plot_running_times()
```



For both men and woman, we see a clear downward trend for most distances, most clearly seen in the 10 and 5 km.There does not appear to be seasonality or any cycles with the data. Additionally, the data contains missing values in 1916, 1940 and 1944 due to the World Wars. Furthermore, there are individual missing observations for several years: 1932, 1936, 1952, 1956, 1988. Women only started participating in 1928, while men competed since 1896. Competing distances appear to have been added at different times. For men, the majority of running distances were introduced in the period between 1896 and 1912. For women, the roll out was much slower, with the latest running distances being introduced in 1988 (10,000 meters) and 1996 (5,000 meters)

1.2) Next, we fit a regression line to the data for each event and assess at what average rate the winning times have been decreasing. We do so by setting `method = "lm"`, the extensive version can be found in the Appendix.

```
plot_linear_trend()
```



The estimated average decrease per year per level can be found in the table below:

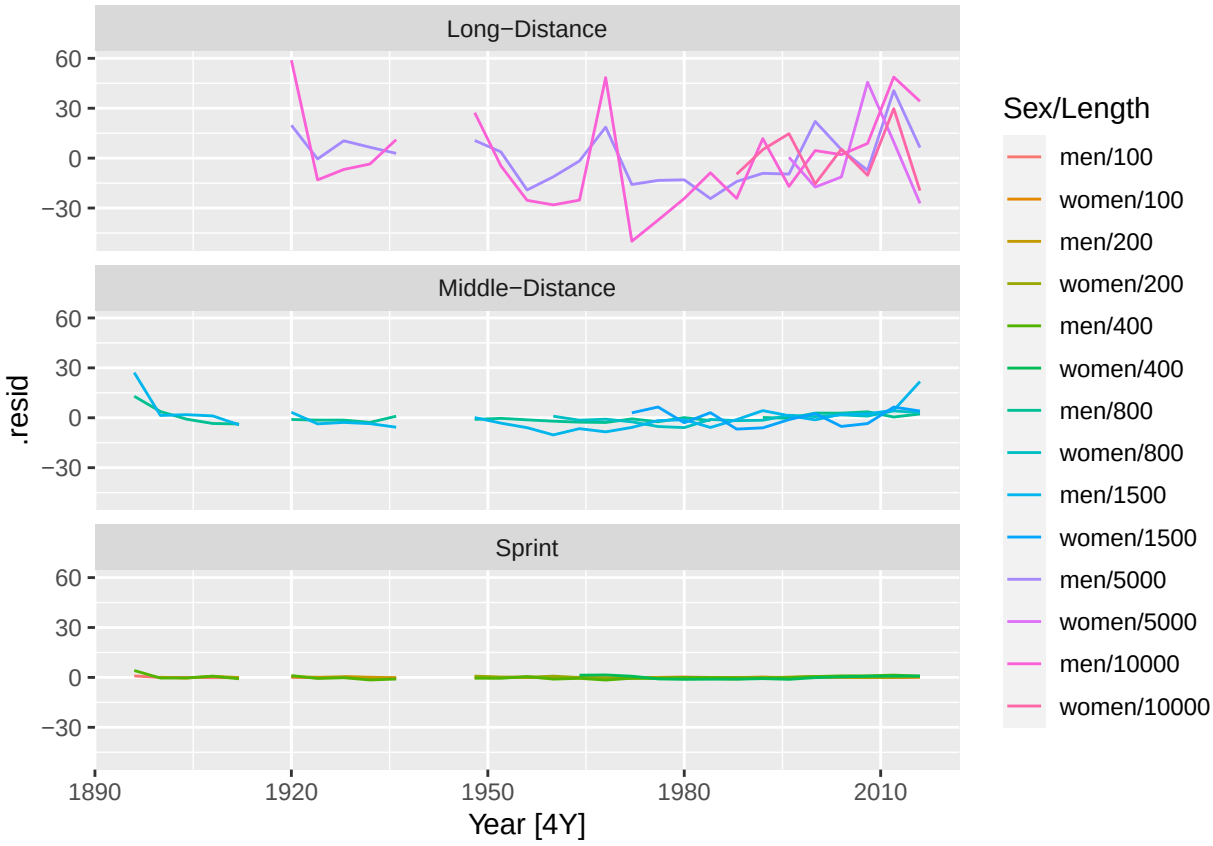
```
# Group by Sex and Length, then fit separate time series linear models
fit_run <- olympic_running %>%
    model(lm = TSLM(Time ~ trend()))

# Retrieve average decrease of slope
data = fit_run %>% coef() %>% filter(term=="trend()") %>% select(Length, Sex, estimate)


knitr::kable(xtabs(estimate ~ Sex + Length, data = data), "pipe")
```

	100	200	400	800	1500	5000	10000
men	-0.050	-0.100	-0.258	-0.607	-1.260	-4.12	-10.7
women	-0.057	-0.135	-0.160	-0.792	0.587	-1.21	-14.0

1.3) Following, we plot the residuals against the year and asses how well the fitted lines suit the given data. For clarity, we create three subplots based on `Length`.

```
# Assuming fit_run contains your fitted models
# Calculate residuals
residuals_data <- fit_run %>%
  augment() %>%
  select(Year, Sex, Length, .resid)
residuals_data <- residuals_data %>%
```

```
  mutate(discipline =
           ifelse(Length<800, "Sprint",
                  ifelse(Length<3000, "Middle-Distance", "Long-Distance")))
residuals_data %>% autoplot(.resid) + facet_wrap(~discipline, nrow=3)
```



While the residuals center around 0, there is increasing variation for some of them, most obvious in the long-distance disciplines. This might suggest heteroskedasticity and a model misspecification, and might affect the calculation of prediction intervals. Larger residuals (and variation over time) indicates that the fitted values does not fit the data very well, i.e. the fitted line does not capture the data very well. This might, however, also be a matter of scale here and just looking at the plots is not enough. To evaluate if the residuals are actually white noise (what you would want to achieve), further tests should be conducted.

1.4) Lastly, we predict the winning time for each race in the 2020 Olympics and give a prediction interval of 80 and 95%.

```
#forecast results 2020
forecast = forecast(fit_run)
data = forecast %>%
  filter(Year==2020) %>%
  hilo(level = c(80, 95)) %>% select(!.model)

knitr::kable(data, "pipe")
```

Length	Sex	Year	Time	.mean	80%	95%
100	men	2020	N(9.5, 0.061)	9.54	[9.22, 9.85]80	[9.05, 10.0]95
100	women	2020	N(11, 0.059)	10.53	[10.22, 10.85]80	[10.06, 11.0]95
200	men	2020	N(19, 0.13)	19.12	[18.66, 19.57]80	[18.42, 19.8]95
200	women	2020	N(21, 0.31)	21.20	[20.48, 21.91]80	[20.11, 22.3]95
400	men	2020	N(42, 1.5)	42.04	[40.49, 43.59]80	[39.67, 44.4]95
400	women	2020	N(48, 1.4)	48.44	[46.92, 49.95]80	[46.12, 50.7]95
800	men	2020	N(99, 13)	99.24	[94.60, 103.88]80	[92.14, 106.3]95
800	women	2020	N(111, 14)	111.48	[106.66, 116.31]80	[104.11, 118.9]95
1500	men	2020	N(207, 74)	207.00	[195.94, 218.05]80	[190.09, 223.9]95
1500	women	2020	N(245, 35)	245.46	[237.83, 253.09]80	[233.79, 257.1]95
5000	men	2020	N(773, 285)	772.83	[751.19, 794.46]80	[739.74, 805.9]95
5000	women	2020	N(892, 1562)	892.12	[841.47, 942.76]80	[814.66, 969.6]95
10000	men	2020	N(1580, 970)	1580.35	[1540.43, 1620.27]80	[1519.30, 1641.4]95
10000	women	2020	N(1763, 530)	1763.02	[1733.51, 1792.52]80	[1717.90, 1808.1]95

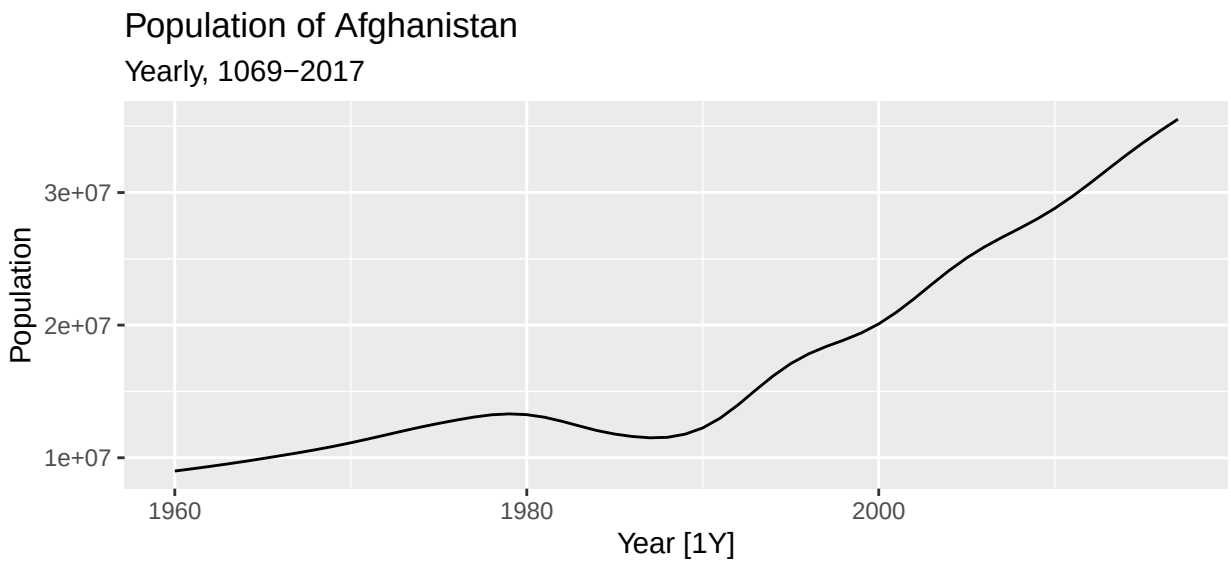
We made the following assumptions about the regression errors in our model: they have zero mean, they are not autocorrelated, they are unrelated to the predictor variables. For generating the prediction intervals we, furthermore, assumed that they are normally distributed with constant variance.

Exercise 2

2.1) Next, we assess the annual population of Afghanistan, which is available in the global_economy dataset. We plot the data and comment on its feature and assess whether the effect of the Soviet-Afghan war is visible.

```
pop_afg <- global_economy %>%
  filter(Code == 'AFG') %>%
    select(Population)
pop_plot <-  pop_afg %>% autoplot(Population)
```

```
pop_plot
```



Population has been steadily rising 1960 to 1980s, where a dip was experienced as a result of the war with the Soviet union. Following this dip, the population increase has been following a positive linear trend. There is no seasonality. Surprisingly, there was no secondary dip, after the invasion of Afghanistan in 2001 by the US.

2.2) We fit a linear trend model and compare this to a piece-wise linear trend model with knots at 1980 and 1989.

```
models <- pop_afg %>% model(
  linear_model = TSLM(Population ~ trend()),
  pw_model = TSLM(Population ~ trend(knots = c(1980, 1989)))
)

knitr::kable(
  report(models) %>%
    mutate(across(is.numeric, round, digits=3)) %>%
    select(!c(deviance, df.residual, rank, df, log_lik, statistic, sigma2))
  %>% t(), "pipe")
```

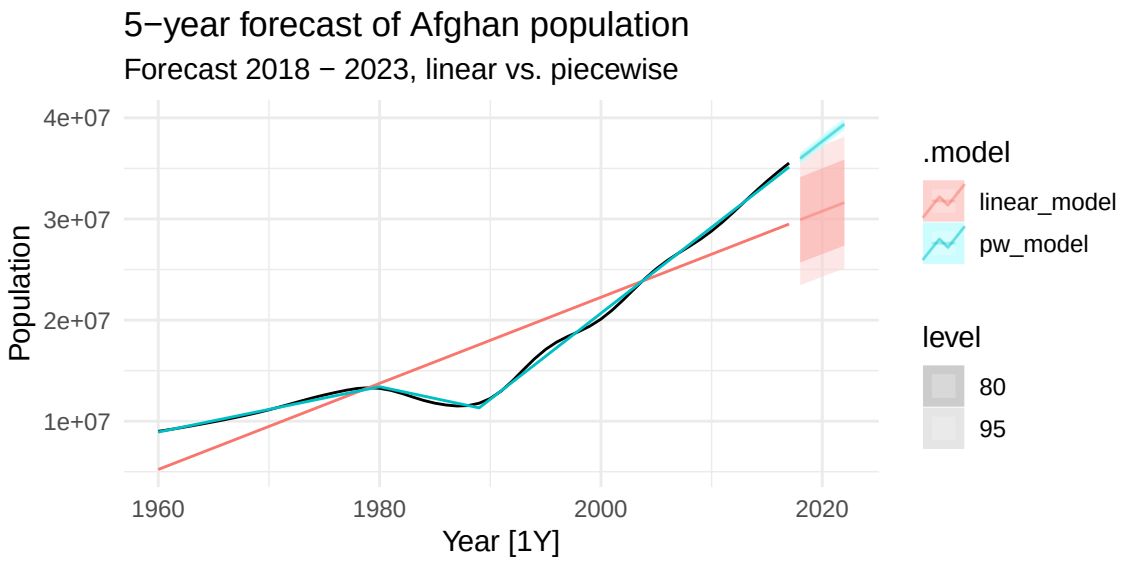
.model	linear_model	pw_model
r_squared	0.838	0.999
adj_r_squared	0.835	0.999

p_value	0	0
AIC	1741	1469
AICc	1742	1470
BIC	1747	1479
CV	1.06e+13	9.55e+10

For comparison, we can look at the $adj.R^2$, we can see that the piecewise model has a higher $adj.R^2$ and is in that regards the better model and captures the relationship between trend and population better. This is because the linear model is not able to capture the dip from the war.

2.3) Following, we generate forecasts from these two models for the five years after the end of the data, and reflect on the results.

```
forecasts <- models %>% forecast(h = 5)

plot_forecast_pop()
```



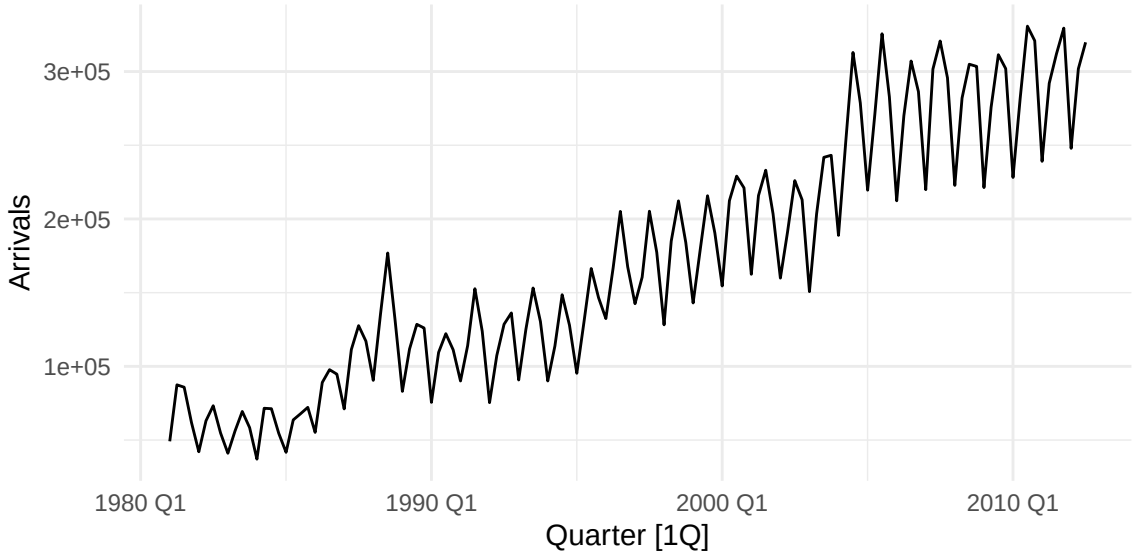
From the plot we can see that the piece-wise model performs considerably better. The piece-wise Linear model has tighter prediction intervals and follows the trend line remarkably well. The Linear model has greater prediction intervals and under-estimates the trend line.

Exercise 3

For this exercise we use the quarterly number of arrivals to Australia from New Zealand, 1981 Q1 - 2012 Q3, from data set `aus_arrivals`.

3.1) We split the data in train and test sets and make a time plot of the data:

```
nz_arrivals = aus_arrivals %>% filter((Origin == 'NZ'))
training_data = nz_arrivals %>% filter(year(Quarter) <=2010)
test_data = nz_arrivals %>% filter(year(Quarter) > 2010)
nz_arrivals %>% autoplot(Arrivals) + theme_minimal()
```

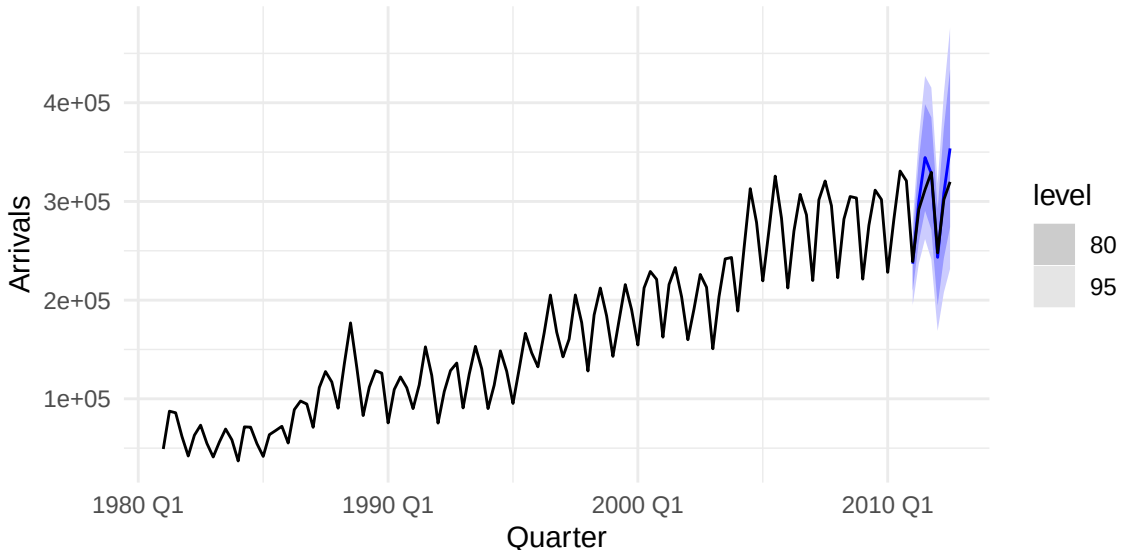


We can clearly see that the data shows an upward (positive) trend with a strong seasonality. The variance also seems to increase. We then fit a Holt-Winters' multiplicative model to the training data:

```
model = training_data %>% model(
  hw_smoothing = ETS(Arrivals ~ error("M") + trend("A") +season("M"))
)
```

And then forecast the test set:

```
forecasts = model %>% forecast(h = 7)
autoplot(forecasts, nz_arrivals) + theme_minimal()
```



The model slightly over-estimates the peaks in seasonal variation but appears to be fitting well enough. The multiplicative approach (for seasonality) is needed as it can be observed from the time plot, the Arrivals from New Zealand's seasonality has been increasing in variance - hence the assumption of constant seasonal variance of the additive approach does not hold. The data is also strictly positive, so we can also use the multiplicative method for the errors.

3.2) Next, we forecast the two-year test set with different models:

```
fs_models = training_data %>% model(
  ets_method = ETS(Arrivals ~ error("A") + trend("A") + season("N")),
  additive_transformed = ETS(log(Arrivals) ~ error("A") + trend("A") + season("A")),
  naive_method = SNAIVE(Arrivals),
  stl_ets_model = decomposition_model(
      STL(log(Arrivals)),
      ETS(season_adjust)))
```

```
forecasts = fs_models %>% forecast(h = 7)
plt1 = autoplot(forecasts, test_data) +
  theme_minimal()
plt1
```

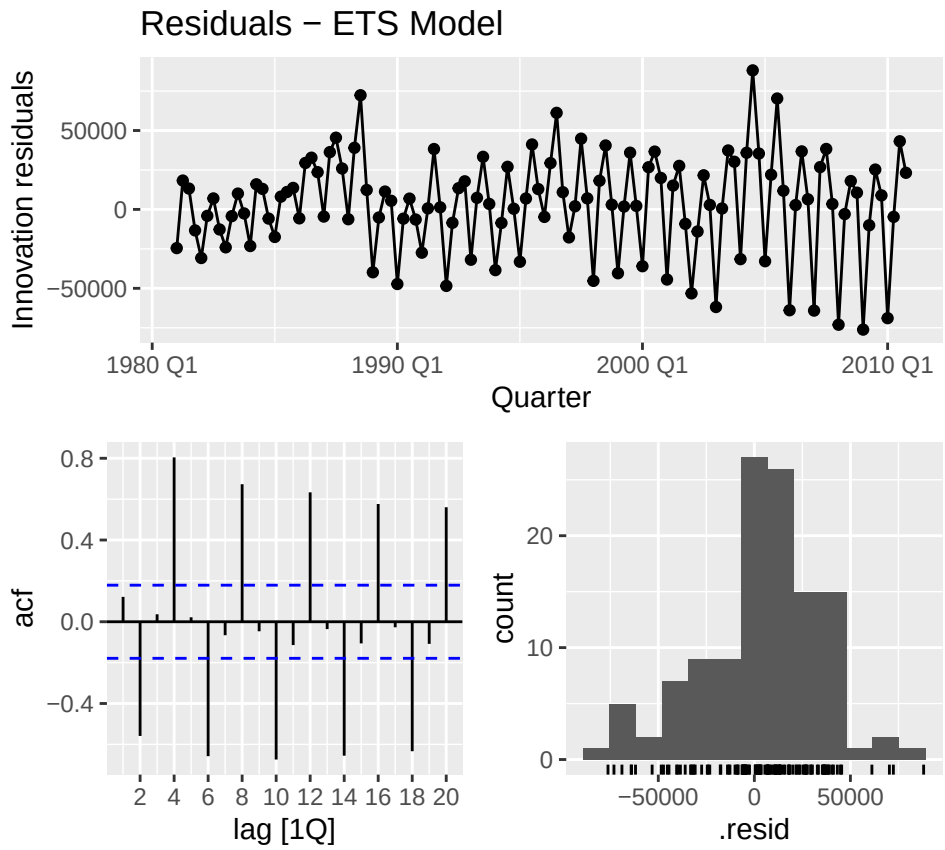
To evaluate the models, we calculate different accuracy scores:

```
acc = accuracy(forecasts, test_data) %>% select (!c(Origin,.type, MASE,RMSSE))
knitr::kable(acc, "pipe")
```

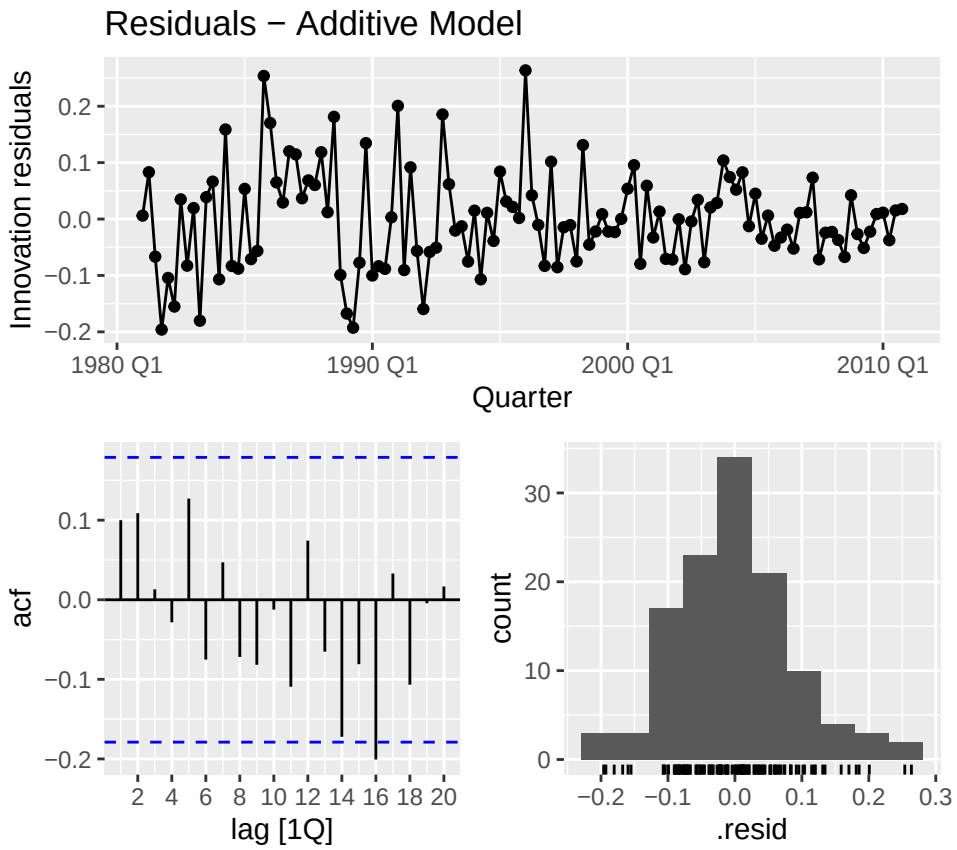
.model	ME	RMSE	MAE	MPE	MAPE	ACF1
additive_transformed	-20920	27624	20920	-6.78	6.78	-0.017
ets_method	-19498	36468	26038	-8.05	10.05	-0.172
naive_method	5692	14962	14203	2.27	4.97	-0.116
stl_ets_model	-22350	28963	22350	-7.28	7.28	-0.012

And look at the residuals diagnostics:

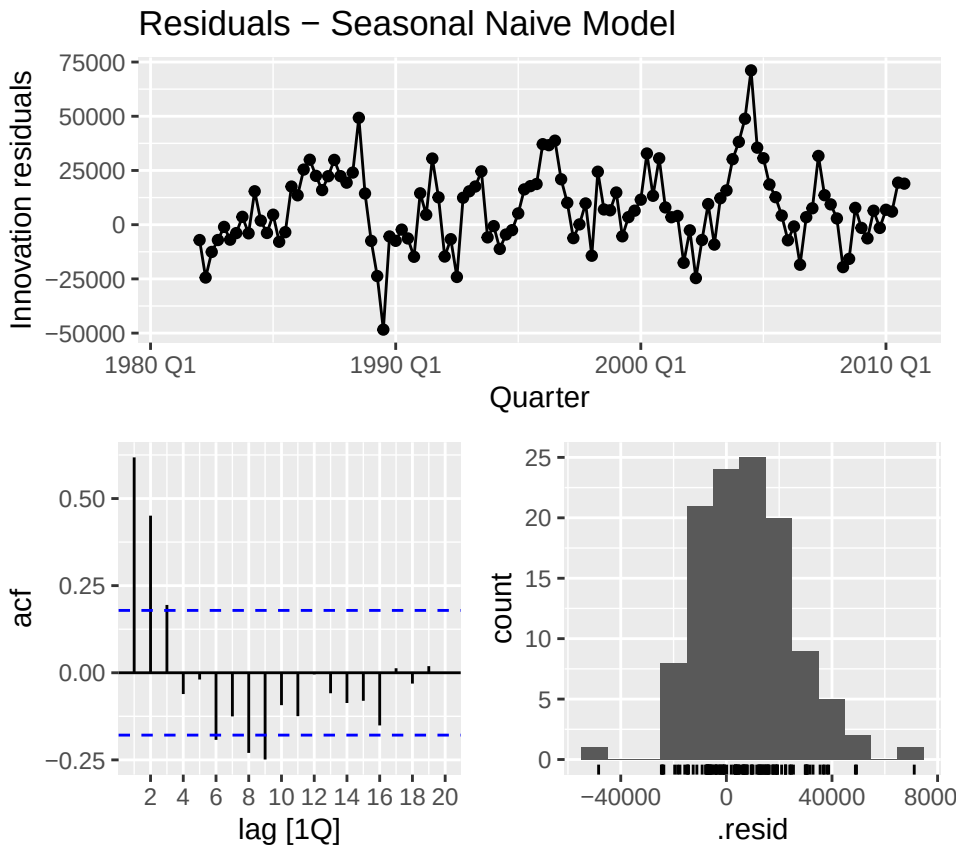
```
fs_models %>% select(ets_method) %>%  gg_tsresiduals() +
  labs(title="Residuals - ETS Model")
```



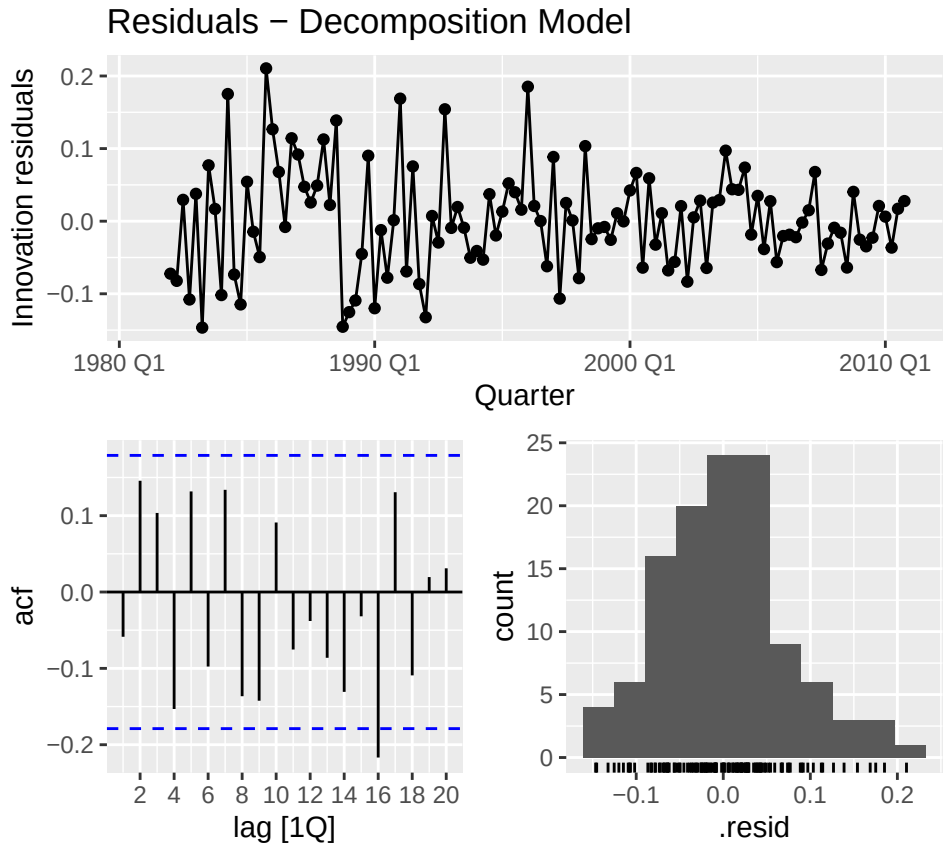

```
fs_models %>% select(additive_transformed) %>%  gg_tsresiduals() +
  labs(title="Residuals - Additive Model")
```




```
fs_models %>% select(naive_method) %>%  gg_tsresiduals() +
  labs(title="Residuals - Seasonal Naive Model")
```




```
fs_models %>% select(stl_ets_model) %>%  gg_tsresiduals() +
  labs(title="Residuals - Decomposition Model")
```



The best predicting model appears to be the seasonal naive model, as it has the lowest scores for both RMSE and MAE. However, looking at the autocorrelation of the residuals of the model, there appears to be significant correlation. Furthermore, the distribution of the residuals does not appear to be approximating a normal distribution either or it is a normal distribution with wide tails. Finally, there appears to be heteroskedasticity in the residuals, since there are quite a few spikes in the residuals.

3.3) Next, we compare the same four methods using time series cross-validation instead of using a training and test set.

```
knitr::kable(get_cv_accuracy_nz_arrivals(), "pipe")
```

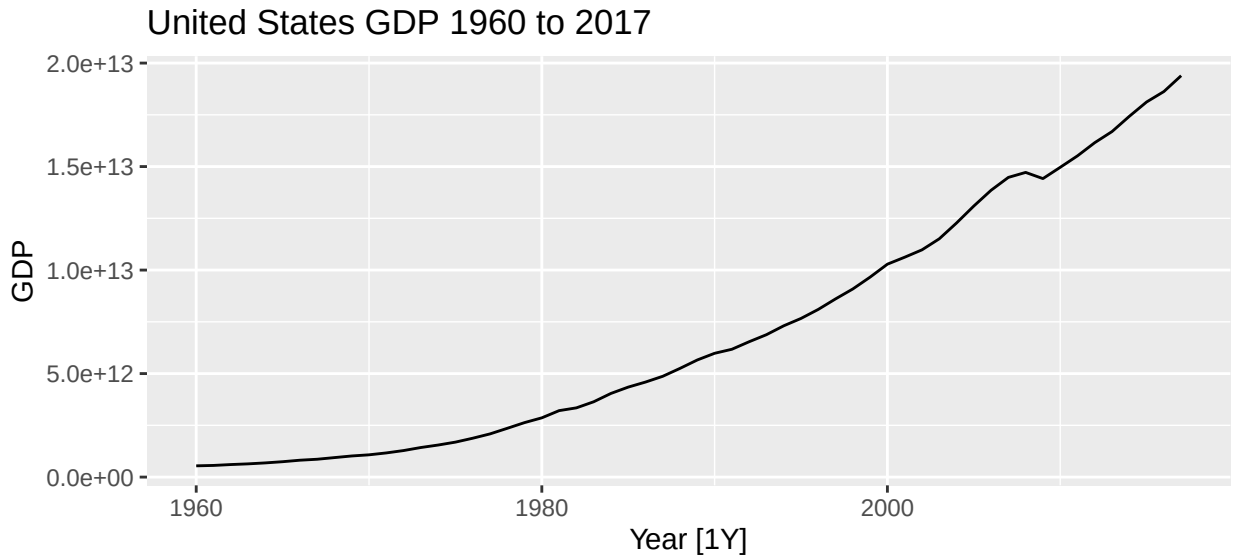
.model	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
additive_transformed	-12289	21392	15660	-3.90	5.12	1.77	2.42	0.182
ets_method	-10986	34303	26130	-5.14	9.83	2.96	3.87	-0.096
naive_method	8630	14508	13046	3.14	4.55	1.48	1.64	-0.137
stl_ets_model	-8468	22872	18152	-2.53	6.08	2.06	2.58	0.379

The seasonal naive model still performs best, but the error for all models has decreased, which shows the advantage of using cross-validation.

Exercise 4

4.1) We first plot the data:

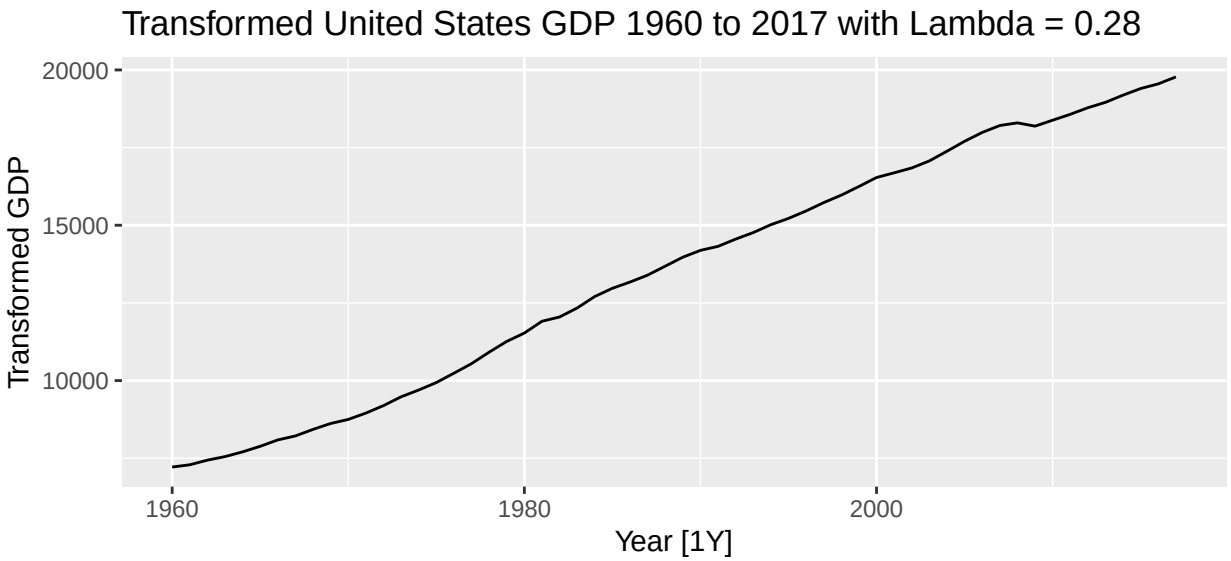
```
gdp <- global_economy %>%
  filter(Code == "USA") %>%
  select(Year, GDP) %>% as_tsibble(index=Year)
gdp %>% autoplot() + labs(title="United States GDP 1960 to 2017")
```



We see an increasing trend, with no clear seasonal pattern. It is difficult to see whether the variation increases as the level of the series increases. However, given that it concerns GDP data, you can argue that it is that case, as e.g. we always talk about % growth in GDP. We therefore apply the Box-Cox transformation, finding the optimal value for lambda using the Guerrero feature. We see that is stabilizes the variance and makes the graph more linear. The dip around 2008, most likely caused by the housing crisis, remains visible.

```
lambda <- gdp %>%
  features(GDP, features=guerrero) %>%
  pull(lambda_guerrero)
plot_gdp_transformed <- gdp %>% autoplot(box_cox(GDP, lambda))
```

```
plot_gdp_transformed
```



Next, we fit an ARIMA model, searching for the optimal model specification using a stepwise approach and another with a more exhaustive search.

```
gdp_fit <- gdp_transformed %>% model(stepwise = ARIMA(trans),
              search = ARIMA(trans, stepwise=FALSE,greedy = FALSE))

glance(gdp_fit) %>% arrange(AICc)
```

```
## # A tibble: 2 x 8
##   .model  sigma2 log_lik   AIC  AICc   BIC ar_roots ma_roots
##   <chr>    <dbl>   <dbl> <dbl> <dbl> <dbl> <list>   <list>
## 1 stepwi~  5479.   -325.  657.  657.  663. <cpl>    <cpl>
## 2 search   5479.   -325.  657.  657.  663. <cpl>    <cpl>
```

```
gdp_fit
```

```
## # A mable: 1 x 2
##                   stepwise                    search
##                    <model>                   <model>
## 1 <ARIMA(1,1,0) w/ drift> <ARIMA(1,1,0) w/ drift>
```

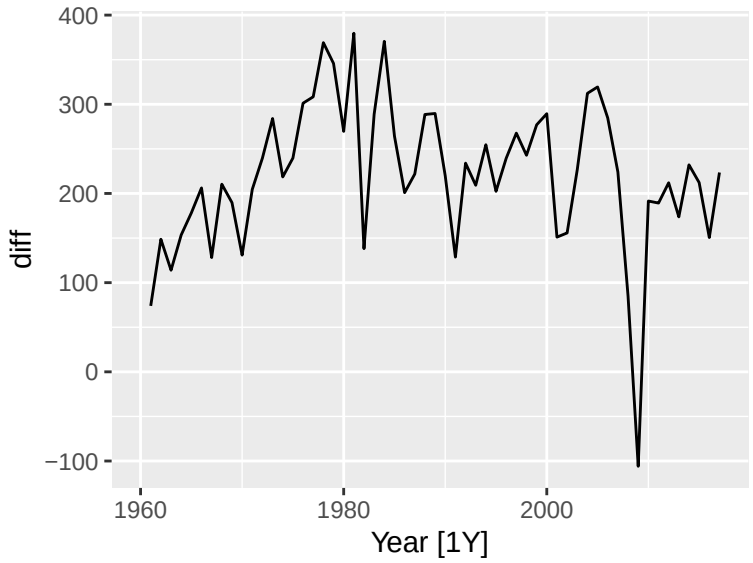
As one can see, both methods return the same model, an ARIMA(1,1,0) with drift.

4.2) The (transformed) data clearly is not stationary, given its increasing trend. We test this with the KPSS test, which gives a p-value of 0.01. We therefore difference the data, the KPSS test now gives a p value of 0.1, indicating stationarity. This suggest that we can take $d = 1$.

```
gdp_transformed %>% features(trans, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1      1.55        0.01
```

```
gdp_diff <- gdp_transformed %>% mutate(diff = difference(trans,differences = 1))
gdp_diff %>% autoplot(diff)
```

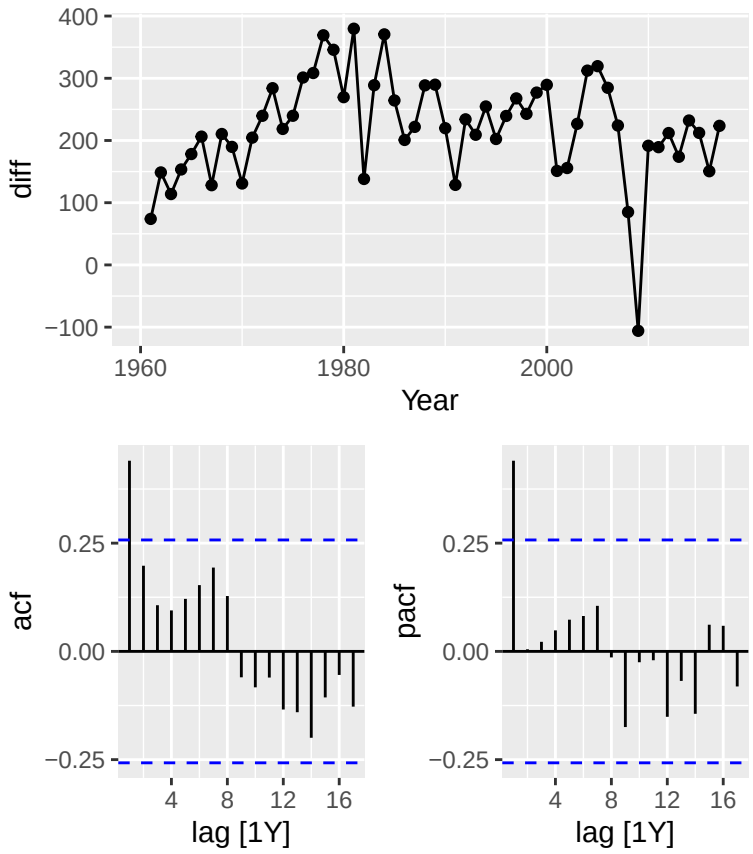



```
gdp_diff %>% features(diff, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1     0.208         0.1
```

We now look at the ACF and PACF plots of the differences data.

```
gdp_diff %>% gg_tsdisplay(diff, plot_type="partial")
```



Both show a peak for the first lag. The PACF plot does not seem sinusoidal or exponentially decaying. But the ACF plot does seem sinusoidal, hence an ARIMA(p,d,0) might be appropriate, as we found in the previous sub question. It might also be the case that it is an ARIMA (p,d,q) model, for which the plots do not help. We therefore also test some options for that specification. Note that the unit root test indicated that the differenced data is already stationary with d=1, and that we can only compare information criteria for ARIMA models with the same order of differencing. We therefore stick to d=1.

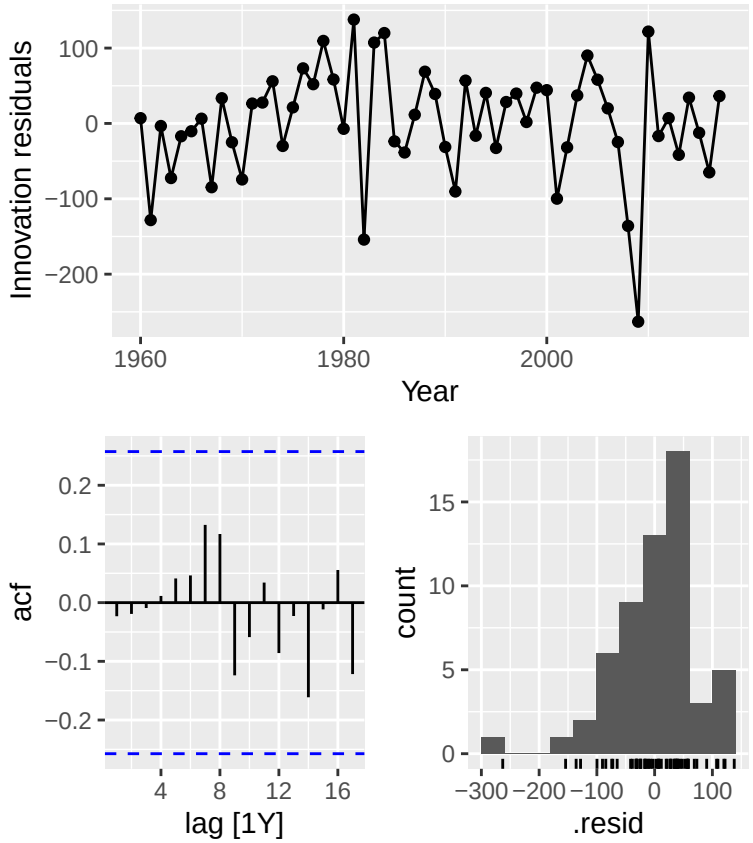
```
gdp_fit2 <- gdp_transformed %>% model(
  arima110 = ARIMA(trans ~ pdq(1,1,0)),
  arima210 = ARIMA(trans ~ pdq(2,1,0)),
  arima111 = ARIMA(trans ~ pdq(1,1,1)),
  arima211 = ARIMA(trans ~ pdq(2,1,1)),
  arima112 = ARIMA(trans ~ pdq(1,1,2)),
  )

glance(gdp_fit2) %>% arrange(AICc)
```

```
## # A tibble: 5 x 8
##   .model  sigma2 log_lik   AIC  AICc   BIC ar_roots ma_roots
##   <chr>    <dbl>   <dbl> <dbl> <dbl> <dbl> <list>   <list>
## 1 arima1~  5479.   -325.  657.  657.  663. <cpl>    <cpl>
## 2 arima1~  5580.   -325.  659.  659.  667. <cpl>    <cpl>
## 3 arima2~  5580.   -325.  659.  659.  667. <cpl>    <cpl>
```

```
## 4 arima1~  5630.   -325.  660.  661.  670. <cpl>    <cpl>
## 5 arima2~  5647.   -325.  660.  661.  671. <cpl>    <cpl>
```

We see that ARIMA(1,1,0) performs best, having the lowest AICc, equal to 657. We now check the residuals.

```
gdp_fit2 %>% select(arima110) %>% gg_tsresiduals()
```



The residuals and the ACF plot of the residuals look good, as they resemble white noise and do not show any autocorrelation. We also perform a portmanteau test with K= p+q = 1 to test for the resemblance with white noise.

```
augment(gdp_fit2) %>%
  filter(.model=="arima110") %>%
  features(.innov, ljung_box, lag = 10, dof=1)
```

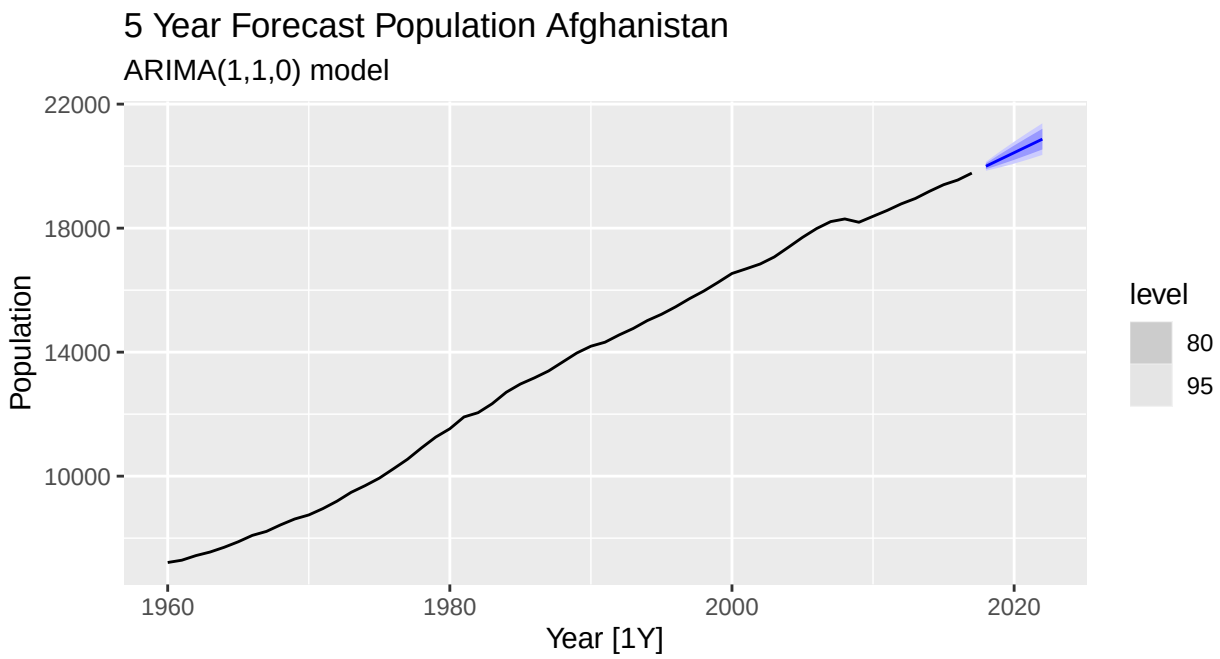
```
## # A tibble: 1 x 3
##   .model   lb_stat lb_pvalue
##   <chr>      <dbl>     <dbl>
## 1 arima110    3.81     0.923
```

The test returns a p value of 0.923, which clearly large enough, suggesting that the residuals are white noise.

4.3) We know produce forefcasts using the ARIMA(1,1,0) model.

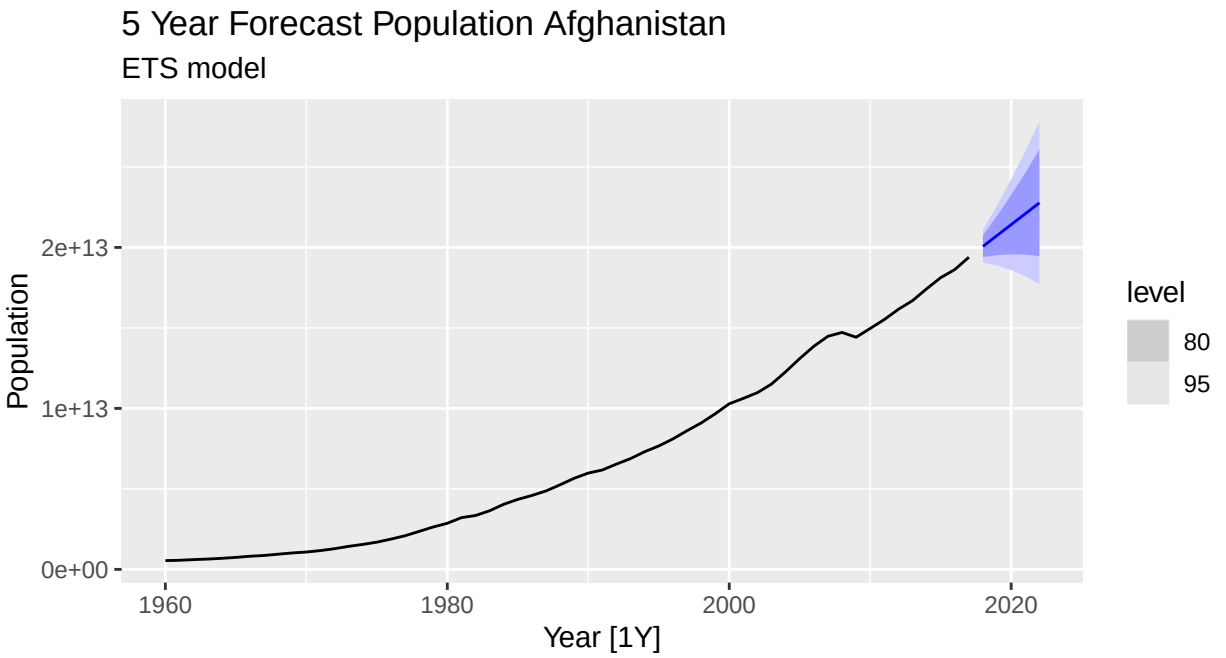
```
fc_gdp <- gdp_fit2 %>%
  forecast(h="5 years") %>%
  filter(.model=='arima110')

plot_fc1 <- gdp_transformed %>% autoplot(trans) + autolayer(fc_gdp)
```

```
plot_fc1
```




```
fc_ets_gdp <- gdp %>%
  model(
    ets = ETS(GDP))  %>% forecast(h="5 years")

plot_fc2 <- gdp %>% autoplot(GDP) + autolayer(fc_ets_gdp)
```

```
plot_fc2
```



We see that the forecasts are similar in terms of the increasing trend, but the ARIMA model has a much more narrow prediction interval. As we know, prediction intervals for ARIMA models tend to be too narrow, which could cause the difference and a prediction that is overconfident. However, it is difficult to make a fair comparison, between the ARIMA model and the ETS model, as we have also performed transformations.

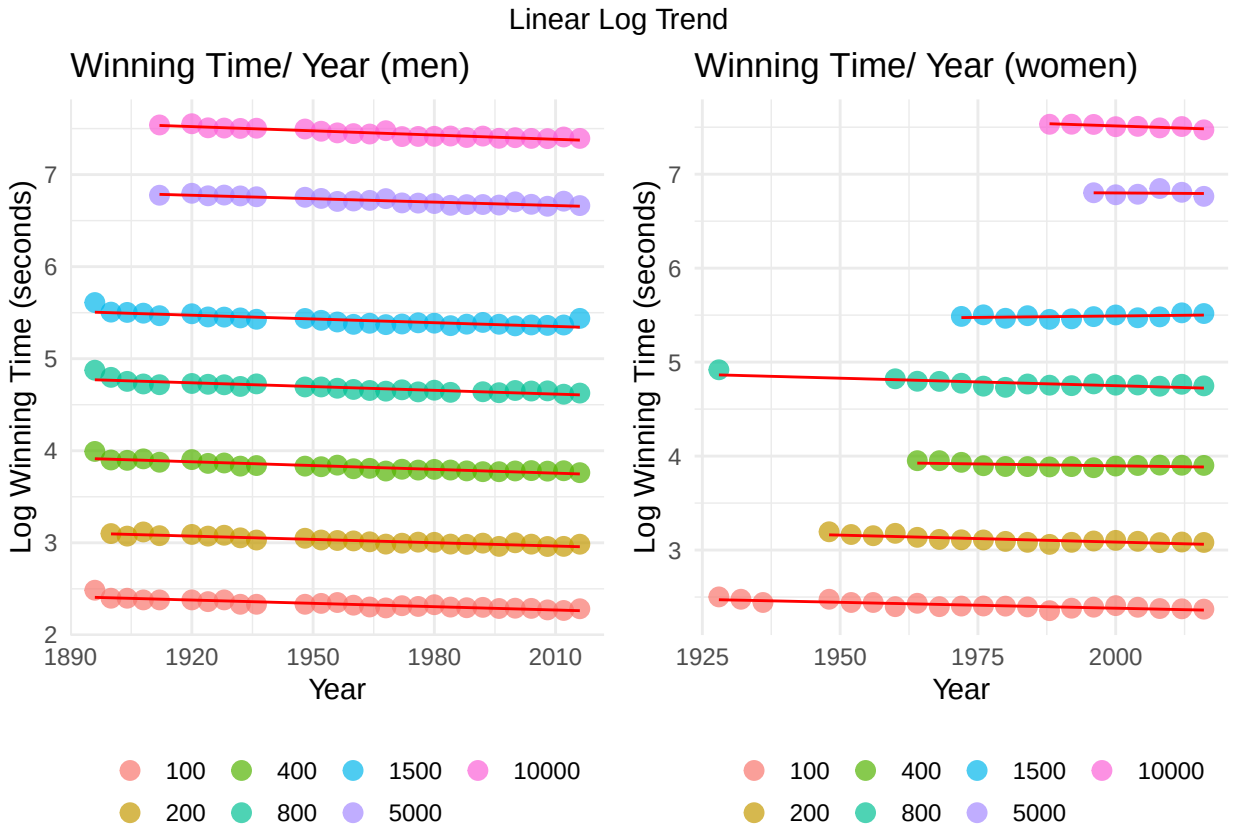
Appendix

```
plot_running_times =  function() {

  sexes = c("men", "women")
legend_labels = c(100, 200, 400, 800, 1500, 5000, 10000)
plots = list()

for (i in 1:2) {
  plot = ggplot(olympic_running %>% filter(Sex == sexes[i]),
       aes(x = Year, y = Time, color = as.factor(Length), size = Length)) +
  geom_point(size = 3, alpha = 0.7) +
  labs(
    title = paste0("Winning Time/ Year (", sexes[i], ")"),
    x = "Year",
    y = "Winning Time (seconds)") +
  theme_minimal() + theme(legend.position ="bottom") +
  scale_color_discrete(name = "", labels = )
  plots[[i]] = plot
}
grid.arrange(grobs = plots, ncol=2, common.legend = TRUE, legend="bottom")
}
```

```
plot_linear_trend = function() {
  sexes = c("men", "women")
legend_labels = c(100, 200, 400, 800, 1500, 5000, 10000)
plots = list()

for (i in 1:2) {
  plot = ggplot(olympic_running %>% filter(Sex == sexes[i]),
                aes(
                  x = Year, y = Time,
                  color = as.factor(Length),
                  size = Length)
                ) +
  geom_point(size = 3, alpha = 0.7) +
  geom_smooth(
    method = "lm", se = FALSE, aes(group = Length), color = "red", size = 0.5) +
  labs(
    title = paste0("Winning Time/ Year (", sexes[i], ")"),
    x = "Year",
    y = "Winning Time (seconds)") +
  theme_minimal() + theme(legend.position ="bottom") +
  scale_color_discrete(name = "", labels = )
  plots[[i]] = plot
}

grid.arrange(
```

```
  grobs = plots, ncol=2, common.legend = TRUE,
  legend="bottom", top=textGrob("Linear Trend")
  )
}
```

```
plot_log_linear_trend = function() {
  sexes = c("men", "women")
legend_labels = c(100, 200, 400, 800, 1500, 5000, 10000)
plots = list()

for (i in 1:2) {
  plot = ggplot(olympic_running %>% filter(Sex == sexes[i]),
                aes(
                  x = Year, y = log(Time),
                  color = as.factor(Length),
                  size = Length)
                ) +
  geom_point(size = 3, alpha = 0.7) +
  geom_smooth(
    method = "lm", se = FALSE, aes(group = Length), color = "red", size = 0.5) +
  labs(
    title = paste0("Winning Time/ Year (", sexes[i], ")"),
    x = "Year",
    y = "Log Winning Time (seconds)") +
  theme_minimal() + theme(legend.position ="bottom") +
  scale_color_discrete(name = "", labels = )
  plots[[i]] = plot
}

grid.arrange(
  grobs = plots, ncol=2, common.legend = TRUE,
  legend="bottom", top=textGrob("Linear Log Trend")
  )
}

plot_log_linear_trend()
```




```
plot_forecast_pop = function(model, forecast) {
  pop_afg %>%
  autoplot(Population) +
  geom_line(data = fitted(models),
            aes(y = .fitted, colour = .model)) +
  autolayer(forecasts, alpha = 0.5, level = c(80, 95)) +
  labs(title = "5-year forecast of Afghan population",
       subtitle = "Forecast 2018 - 2023, linear vs. piecewise",
       y = "Population")
}
```

```
get_cv_accuracy_nz_arrivals = function() {

   cross_val = nz_arrivals %>%
  stretch_tsibble(.init = 20, .step = 1) %>%
  relocate(Quarter, .id)
cv_model = cross_val %>% model(ets_method = ETS(
  Arrivals ~ error("A") + trend("A") + season("N")),
    additive_transformed = ETS(
      log(Arrivals) ~ error("A") + trend("A") + season("A")),
    naive_method = SNAIVE(Arrivals),
    stl_ets_model = decomposition_model(
        STL(log(Arrivals)),
        ETS(season_adjust)
```

```
        ))

output = cv_model %>%
  forecast(h=7) %>%
  accuracy(test_data) %>% select(!c(Origin, .type))
return (output)
}
```