# Lab session 2

**Aim of this practical session:**

In this practical session we are going

- To continue the work from Lab 1 in interpreting statistical plots and model output.
- To provide seasonal and trend analysis for environmental time series (Section 1).
- To examine and interpret some models for extremes (Section 2).

# 1 Part 1: Seasonal and trend analysis

## Sulphur dioxide in the air

Sulphur dioxide is an atmospheric pollutant that is monitored in most cities. The data here are weekly concentrations from 1989, in the file `SO2.csv`.

The dataset contains (amongst others) the following variables of interest:

| Variable | Meaning |
|---|---|
| `Years` | Year of observation |
| `Weeks` | Week of observation (within the year) |
| `ln.SO2.` | Log($SO_2$) |

A natural logarithm transformation has been applied to the $SO_2$ data.

```
# Load required R packages:

library(ggplot2)    # For visualizing our data
library(broom)      # For model predictions


# Read in data (making sure to set the working directory to the
# appropriate location):
SO2 <- read.csv("datasets/SO2.csv", header = TRUE)

# Examine structure of the dataset:
str(SO2)
```

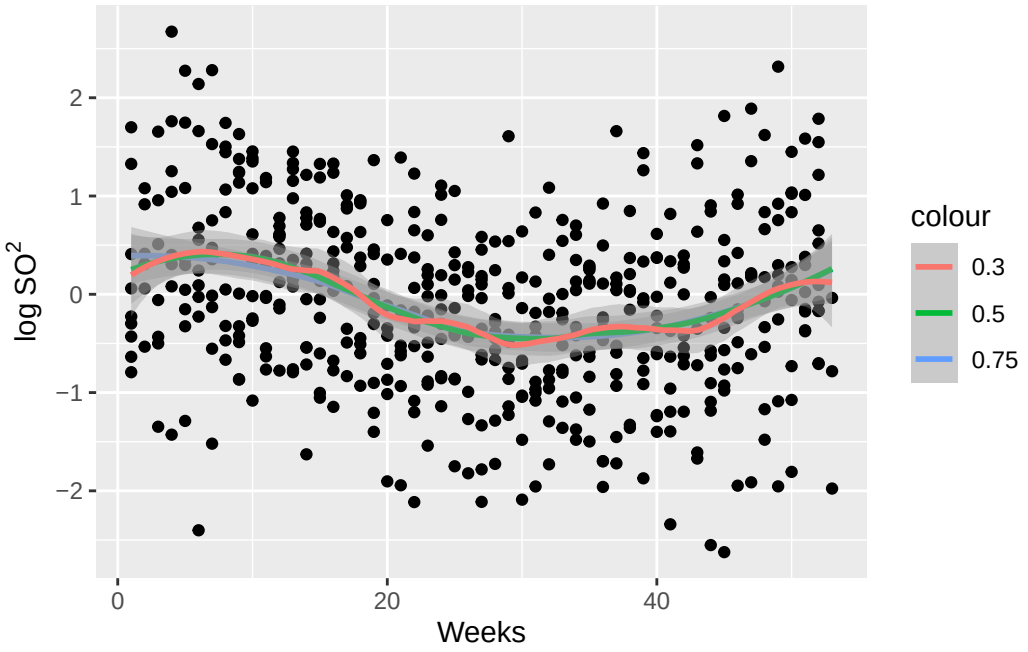
```
'data.frame':   516 obs. of  7 variables:
 $ Years         : int  1989 1989 1989 1989 1989 1989 1989 1989 1989 1989 ...
 $ Weeks         : int  2 5 7 8 9 10 11 12 13 14 ...
 $ Log.Rain      : num  2.001 -0.916 3.19 2.51 2.398 ...
 $ Mean.Temp     : num  2.157 -0.929 0.986 5.271 3.243 ...
 $ Humidity      : num  87.9 92.9 84.4 78 76.9 ...
 $ Wind.Dir.Speed: num  -111 -168 -117 -171 -134 ...
 $ ln.SO2.       : num  0.3086 0.3552 0.4732 0.0487 -0.4865 ...
```

```
# Add year as a decimal (continuous variable):
SO2$year_num <- SO2$Years + SO2$Weeks / 53
```

## 1.1 Visualizing our data

Lets create some exploratory plots to visualize the relationship between log $SO^2$ and the week while adding some loess curves with different spans (to examine possible seasonal patterns):

```
ggplot(data = SO2,
       aes(y = ln.SO2., x = Weeks ))+
  geom_point()+
  geom_smooth(method = "loess",span=0.75,aes(color="0.75"))+
  geom_smooth(method = "loess",span=0.5,aes(color="0.5"))+
    geom_smooth(method = "loess",span=0.3,aes(color="0.3"))+
  labs(y=expression(log~SO^2))
```




There seems to be some evidence of a seasonal pattern in log($SO_2$), with higher values towards the starts and ends of the week within years.

Task 1

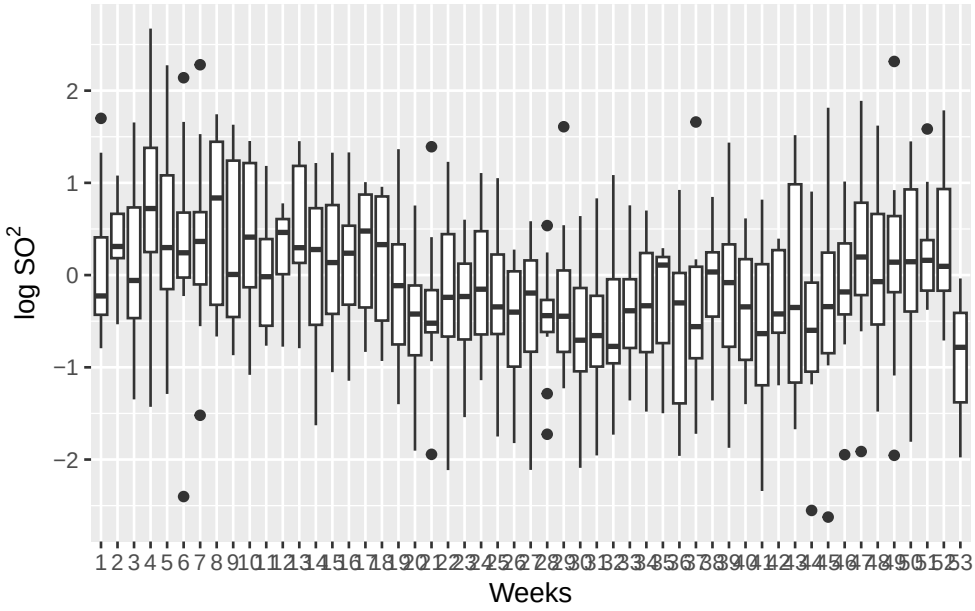
Another way of exploring potential seasonality is by using boxplots. Try to produce some boxplots that show the Log($SO_2$) against the weeks. Recall that boxplots are meant to summarise the distribution of a continuous variable against categorical ones -thus, you need to declare the `Weeks` variable as a factor (this can be done directly on `ggplot`)

Take hint

We can achieve this by adding a `geom_boxplot()` layer to a `ggplot` object.

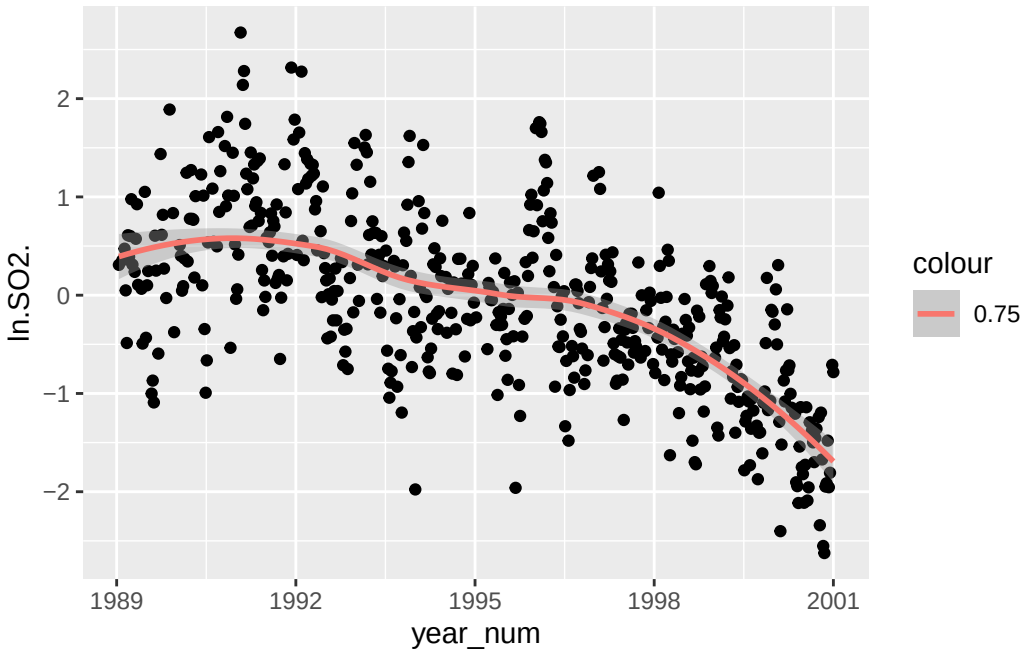
Click here to see the solution

```
ggplot(data = SO2,
       aes(y = ln.SO2., x = as.factor(Weeks) ))+
  geom_boxplot()+
   labs(y=expression(log~SO^2),x="Weeks")
```




Now to examine the trend we can plot our data over years as follows:

```
ggplot(data = SO2,
       aes(y = ln.SO2., x = year_num ))+
  geom_point()+
  geom_smooth(method = "loess",aes(color="0.75"))
```




There is an overall decreasing (not necessarily linear) trend in log($SO_2$) over time. However, we need to account for the seasonal pattern when trying to assess the trend, so we fit a

harmonic model to the data, with a long-term trend also in the model:

```
# Create a model with seasonal pattern (harmonic model) plus trend:
harmonic_model <- lm(ln.SO2. ~ sin(2 * pi * (Weeks - 53) / 53) +
              cos(2 * pi * (Weeks - 53) / 53) + year_num,
            data = SO2)
```

Lets look at the summary of the model:

```
summary(harmonic_model)
```

```
Call:
lm(formula = ln.SO2. ~ sin(2 * pi * (Weeks - 53)/53) + cos(2 *
    pi * (Weeks - 53)/53) + year_num, data = SO2)

Residuals:
     Min       1Q   Median       3Q      Max
-2.44795 -0.42321  0.03846  0.42933  1.64858

Coefficients:
                               Estimate Std. Error t value Pr(>|t|)
(Intercept)                  331.667008  16.390999  20.235  < 2e-16 ***
sin(2 * pi * (Weeks - 53)/53)   0.246362   0.039360   6.259 8.19e-10 ***
cos(2 * pi * (Weeks - 53)/53)   0.326653   0.040119   8.142 2.98e-15 ***
year_num                       -0.166260   0.008215 -20.239  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6365 on 512 degrees of freedom
Multiple R-squared:  0.5082,    Adjusted R-squared:  0.5053
F-statistic: 176.4 on 3 and 512 DF,  p-value: < 2.2e-16
```

We can also compute the predicted values by first generating a grid of values where we want to predict:

```
# Create a prediction grid over the weeks of the years 1989 to 2000:
pred_data <- expand.grid(Weeks = 1:53, year = 1989:2000)
pred_data$year_num <- pred_data$year + pred_data$Weeks / 53
```

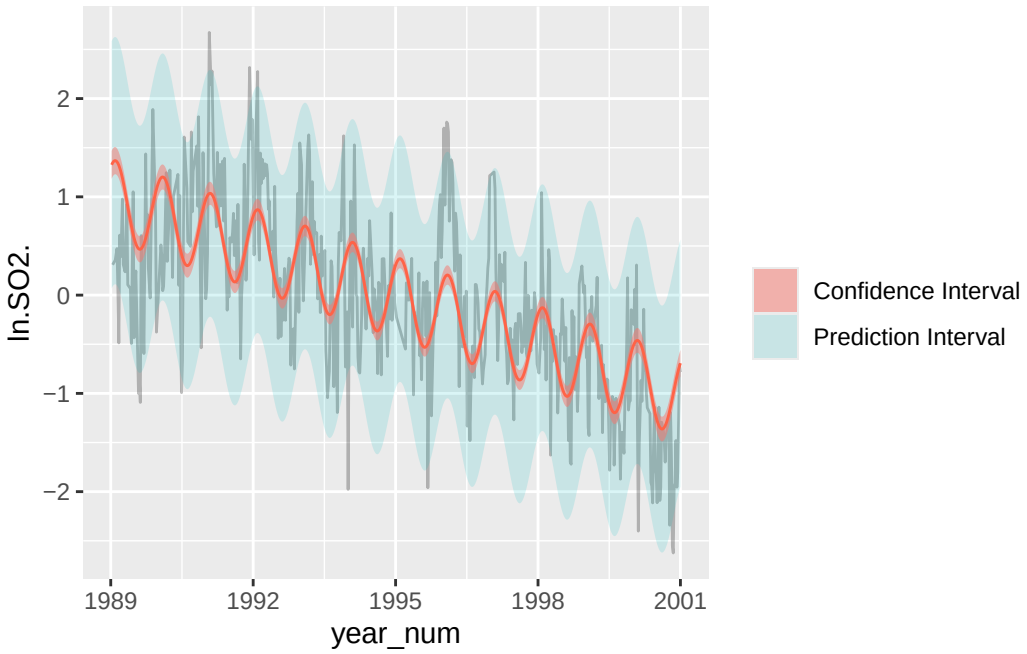
Then we can use the `augment` function from the `broom` library to compute the model predictions for (i) the mean (i.e., $E(\mu_i|Y_i) = \beta_0 + \beta_1 \text{year} + \gamma_1 \sin\left(\frac{2\pi \text{week}}{p}\right) + \gamma_2 \cos\left(\frac{2\pi \text{week}}{p}\right)$ ) or (ii) for new observations (i.e., including the uncertainty associated with the observational error $\epsilon_i$) as follows:

```
# Model predictions with confidence interval for the mean

pred_mean = broom::augment(x=harmonic_model,
                     newdata = pred_data,
                     interval = "confidence",
                     type.predict = "response")

# Model predictions with prediction interval for the new observations
```

```
pred_obs = broom::augment(x=harmonic_model,
                      newdata = pred_data,
                      interval = "prediction",
                      type.predict = "response")
```

We can visualize the model predictions as follows:

```
ggplot() +
  geom_line(data=SO2,aes(y=ln.SO2.,x=year_num),alpha=0.25)+
  geom_ribbon(data= pred_obs,
                aes(x=year_num,
                    ymin = .lower,
                    ymax = .upper,
                    fill="Prediction Interval"),
                alpha = 0.15)+
    geom_ribbon(data= pred_mean,
                aes(x=year_num,
                    ymin = .lower,
                    ymax = .upper,
                    fill="Confidence Interval"),
                alpha = 0.5)+
  geom_line(data=pred_mean,
            aes(y=.fitted,x=year_num),
            color="tomato")+
  scale_fill_discrete(name="")
```



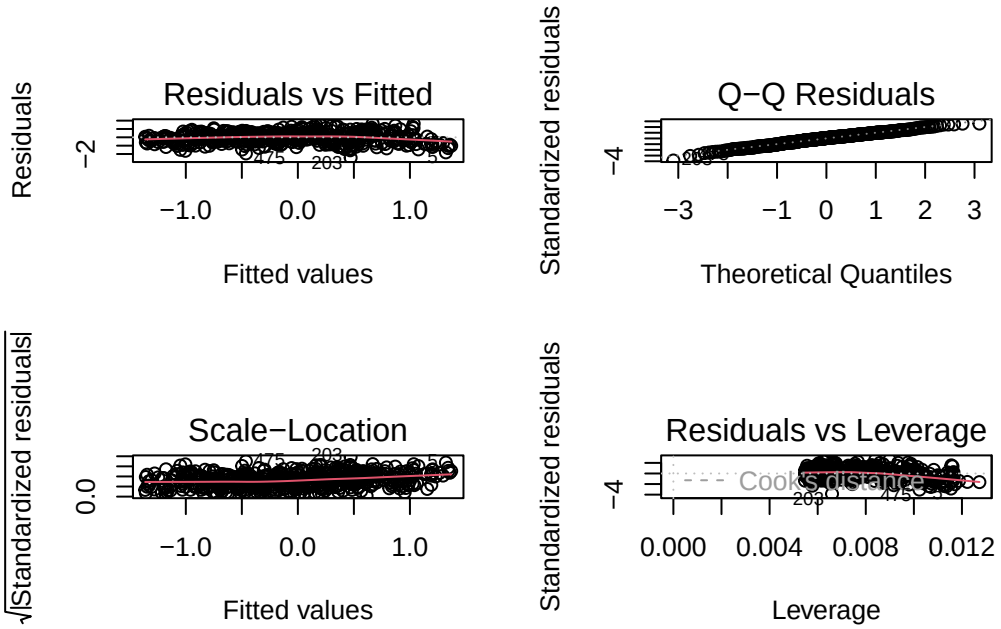

The plot shows the data (grey lines) and the fitted model predictions (as red lines). Then, the blue and red shaded regions indicate the prediction and confidence intervals respectively (notice how the prediction intervals is wider because it takes the observational error into account).

**How well does this model fit the data?** It seems to capture the trend well (following the general pattern in the data) and it seems to fit the seasonal pattern fairly well (having peaks and troughs at the same locations as in the data). However, we note that the model is not

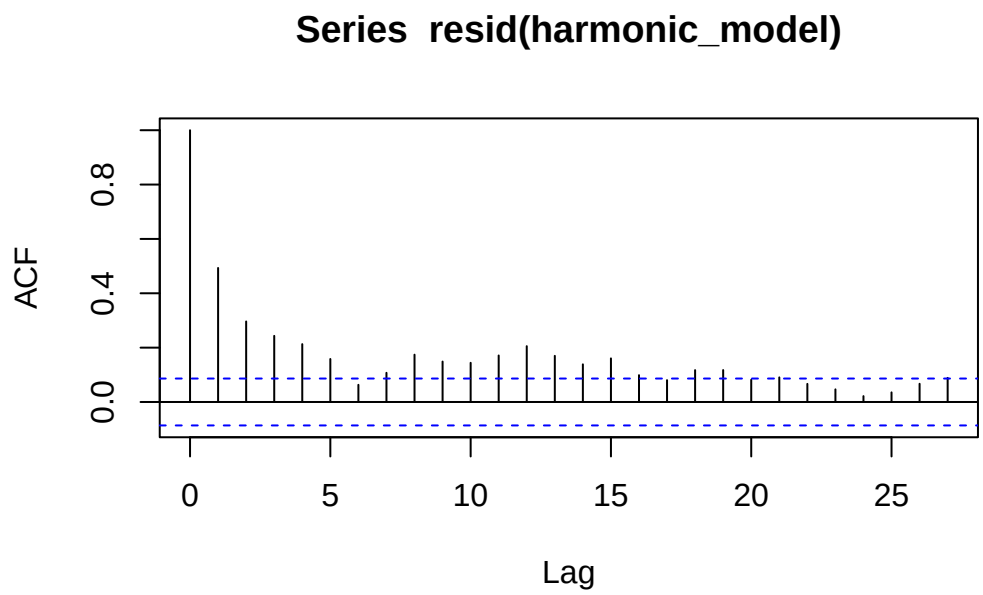
perfect: it does not, e.g., capture the higher peak in year 1996. Perhaps, a more flexible model like an additive model (a type of GAM) might be more appropriate?

We should always check our diagnostic plots when assessing a model:

```
par(mfrow=c(2,2));plot(harmonic_model);par(mfrow=c(1,1))
```




```
acf(resid(harmonic_model))
```




- The plot of residuals versus fitted values (top left) looks OK, since we don't see any clear patterns here.
- The Normal quantile-quantile (Q-Q) plot (top right) also looks OK, since most of the data points follow the line.

- The plot of the autocorrelation function (ACF; bottom) shows that many of the lower lags have values lying outwith the intervals (i.e. lying outwith the blue dashed lines). Therefore, there is autocorrelation in the residuals, and the assumptions of the model are not satisfied. Therefore, this model is not appropriate for the data. We could consider fitted an AR(1) model here to try to address this. (Our more flexible additive model proposed above may also be worth trying.)

## 1.2 Trend detection in Haddock stocks

We will study data relating to the annual estimates of biomass for North Sea Haddock from 1963–2000.

The data are stored in `haddock.dat`, which contains the columns `Year` and `Biomass`.

```
# Read in data:
haddock.all <- read.table("datasets/haddock.dat",header=TRUE)
# Remove the data from year 2001 onwards:
haddock.data <- haddock.all[haddock.all$Year <= 2000,]
```

Task 2

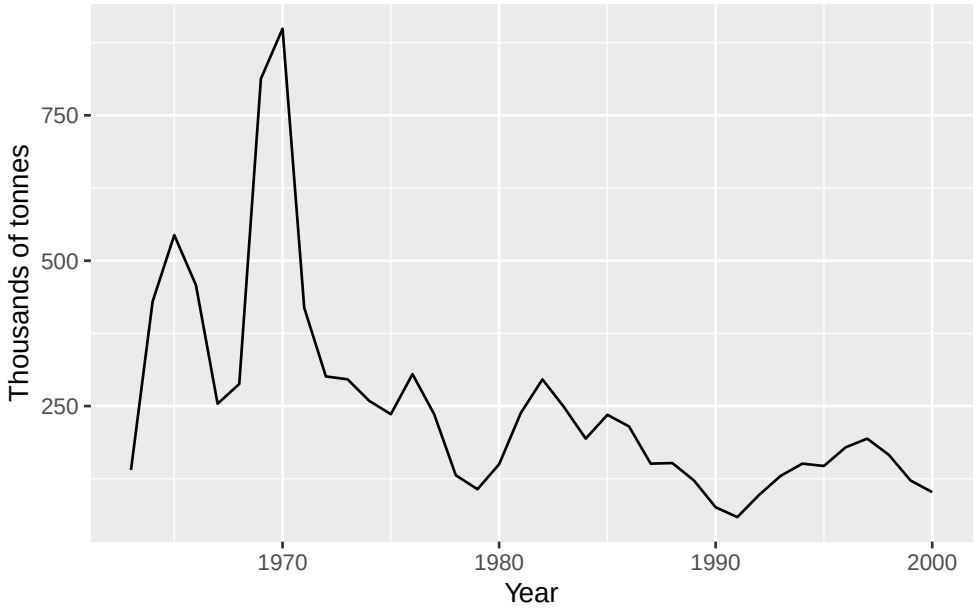
Using `ggplot` produce a line plot that shows biomass (measured in thousands of tonnes) over the years

Take hint

We can add a `geom_line()` layer to produce a line graph in ggplot.

Click here to see the solution

```
ggplot(data=haddock.data,aes(x=Year,y=Biomass))+
  geom_line()+
  labs(y="Thousands of tonnes")
```



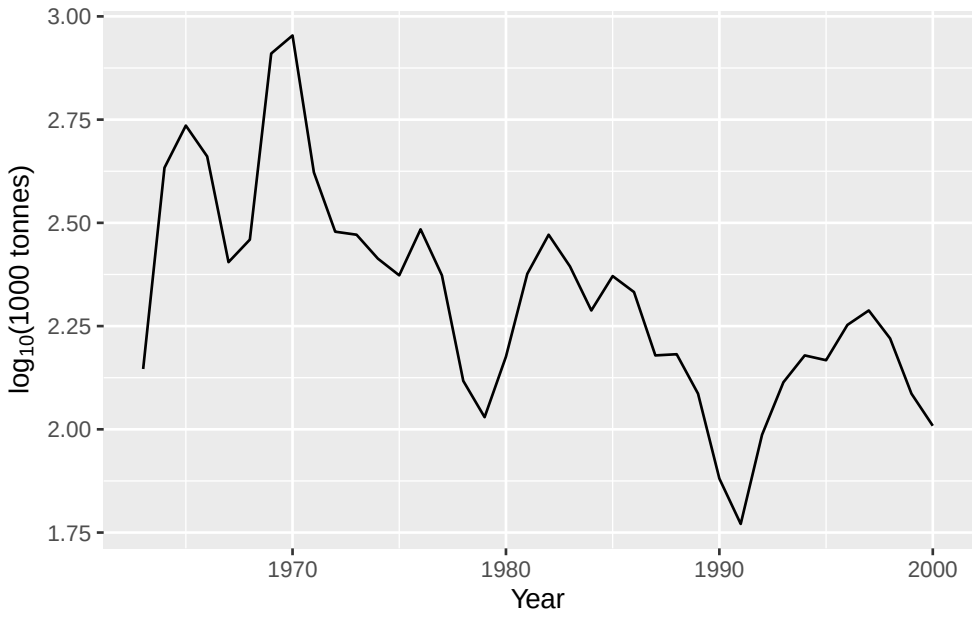
Question

What does the plot produced in the previous task tells you?
See Solution
This plot shows that there is some kind of nonlinear trend in the data, with some higher values earlier on (around 1970) and then generally lower values later on. There seems to be higher variability for the time when there were some higher values.

Task 3

Notice that a log-transformation may be appropriate to address the large variability observed in the data. Plot the $\log_{10}$ of the biomass agianst years using `ggplot`, what do you notice?
Take hint
We can use apply the `log10` function directly to the variable(s) in the ggplot `aes` argument.
Click here to see the solution

```
ggplot(data=haddock.data,aes(x=Year,y=log10(Biomass)))+
  geom_line()+
  labs(y=expression(paste(log[10], "(1000 tonnes)")))
```




We can fit a simple model to the log-transformed biomass (see previous task) below:

```
# Linear trend model:

## Set up data by creating a Time index 1:38 years
haddock.data$Time <- haddock.data$Year-1962
## Fit model:
trend.model0 <- lm(log10(Biomass) ~ Time, data = haddock.data)
summary(trend.model0)
```

```
Call:
lm(formula = log10(Biomass) ~ Time, data = haddock.data)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.46891 -0.06912  0.01352  0.09904  0.45118

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.631107   0.063576  41.385  < 2e-16 ***
Time        -0.016065   0.002842  -5.653 2.02e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1921 on 36 degrees of freedom
Multiple R-squared:  0.4703,    Adjusted R-squared:  0.4556
F-statistic: 31.96 on 1 and 36 DF,  p-value: 2.021e-06
```

The p-value for Time is $< 0.05$, so there is a statistically significant trend. Since the coefficient of this is $< 0$ (and the p-value is $< 0.05$), this is a statistically significant decreasing trend.

Let's predict into the future:

```
## Predict for years 2001 to 2010, i.e., time index 39:48
pred.t <- data.frame(Time = 39:48)
```

Now we use the `augment` function from `broom` to get the model predictions:

```
trend_preds = broom::augment(x=trend.model0,
                     newdata = pred.t,
                     interval = "prediction",
                     type.predict = "response")

# This will allow a plot labelled with "Year"
trend_preds$Year <- 1962 + trend_preds$Time
```

Lastly, we use `ggplot` to plot our predictions:

```
ggplot()+
  geom_line(data=haddock.data,
            aes(x=Year,y=log10(Biomass)),
            alpha=0.25)+
   geom_ribbon(data= trend_preds,
                aes(x=Year,
                    ymin = .lower,
                    ymax = .upper),alpha=0.25)+
  geom_line(data=trend_preds,aes(x=Year,y=.fitted),color="tomato")
```


Since we're predicting for 1 to 10 years ahead, we can extract the 6th and 10th values:

```
trend_preds[c(6,10),]
```

```
# A tibble: 2 x 5
   Time .fitted .lower .upper  Year
  <int>   <dbl>  <dbl>  <dbl> <dbl>
1    44    1.92   1.51   2.34  2006
2    48    1.86   1.43   2.29  2010
```

so we have:

- The predicted log(biomass) in 2006 is 1.92, with a 95% prediction interval of (1.51, 2.34).
- The predicted log(biomass) in 2010 is 1.86, with a 95% prediction interval of (1.43, 2.29).

Since we are on the log scale, we can exponentiate to get predictions on the original scale:

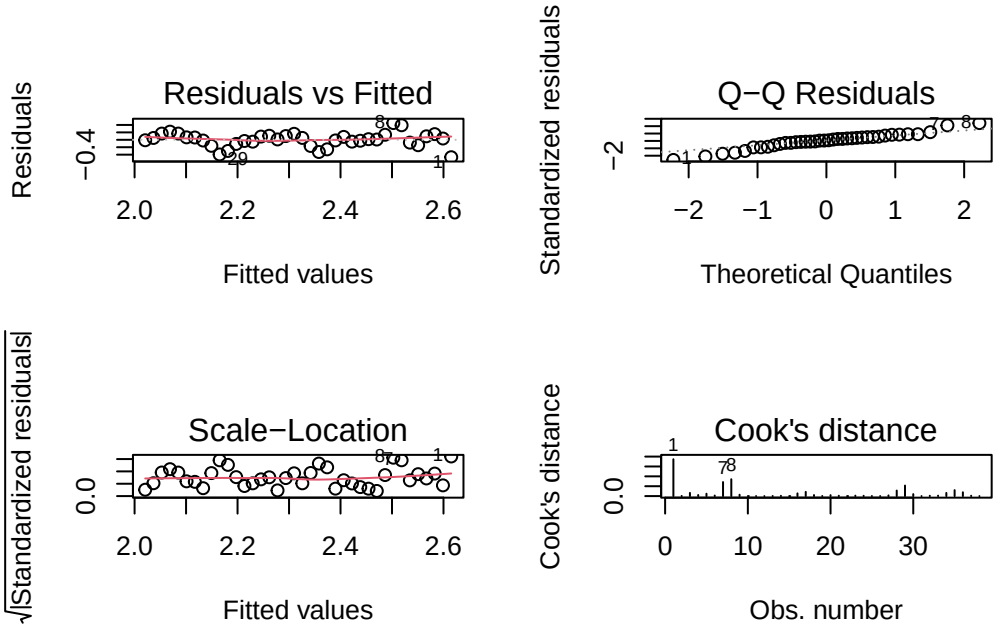
```
exp(trend_preds[c(6,10),2:4])
```

```
# A tibble: 2 x 3
  .fitted .lower .upper
    <dbl>  <dbl>  <dbl>
1    6.85   4.50  10.4
2    6.42   4.19   9.85
```

- The predicted biomass in 2006 is 6.85, with a 95% prediction interval of (4.50, 10.42).
- The predicted biomass in 2010 is 6.42, with a 95% prediction interval of (4.19, 9.85).

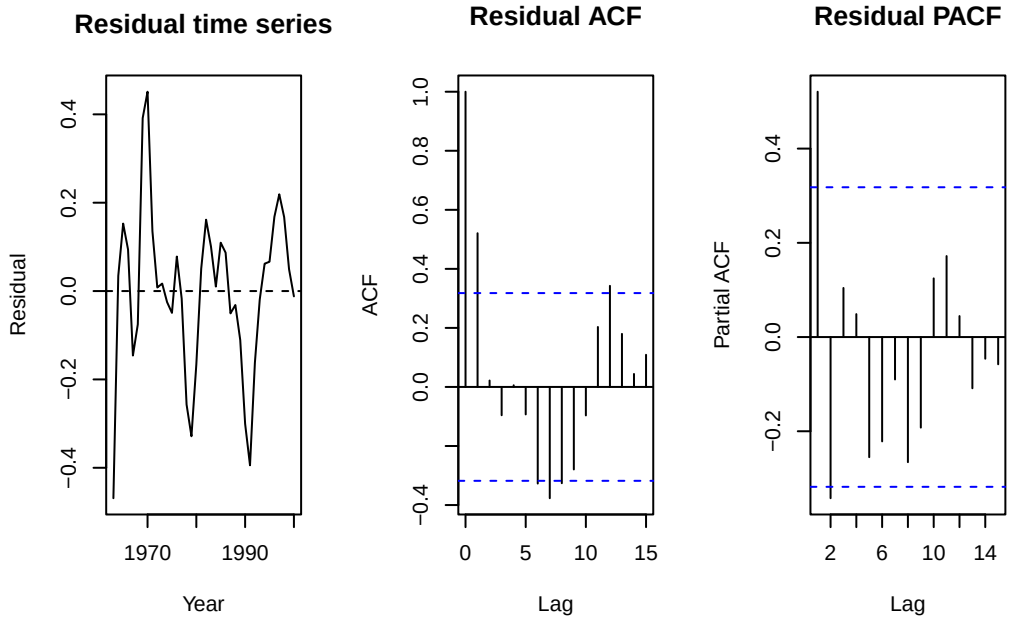
```
layout(matrix(1:4, nrow = 2, byrow = TRUE))
# Check model diagnostic plots:
```

```
plot(trend.model0, which = 1:4)
```




The plot of residuals versus fitted values (top left) shows a clear pattern. This means that the deterministic part of the model does not capture the patterns in the data (a problem!).

```
layout(matrix(1:3, nrow = 1, byrow = TRUE))
e0 <- resid(trend.model0)

plot(haddock.data$Year, e0, type = "l", xlab = "Year",
     ylab = "Residual", main = "Residual time series")
abline(h = 0, lty=2)
acf(e0,main="Residual ACF")
pacf(e0,main="Residual PACF")
```



The lag 1 coefficient is outwith the intervals, so we have statistically significant evidence of residual autocorrelation in the data. This shows that the model is not appropriate for the data. Let's try a time series regression approach.

**Time series model**

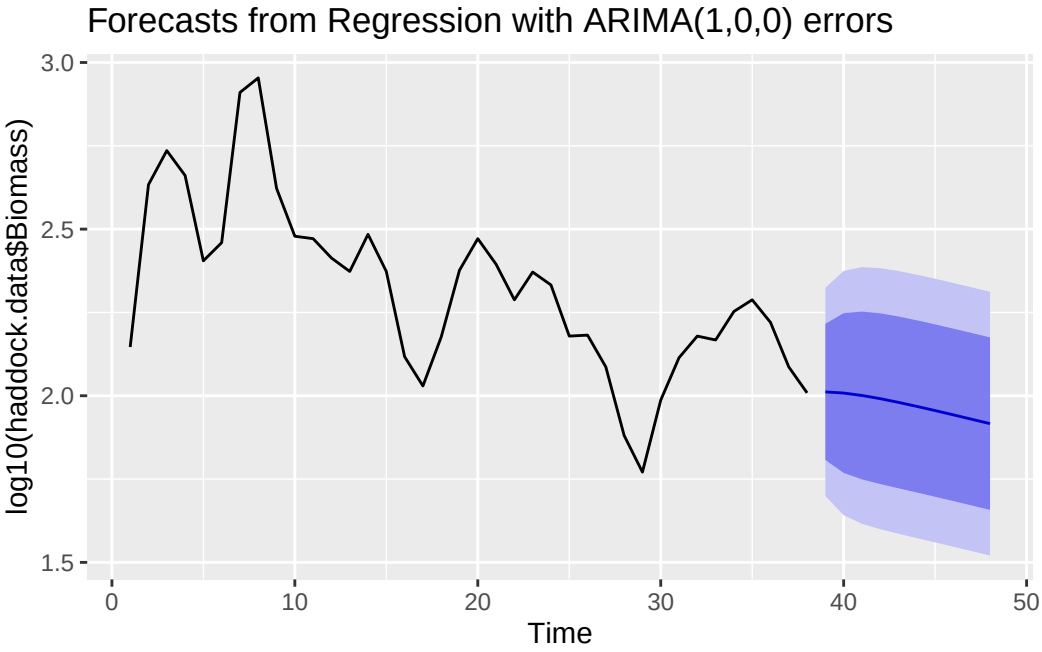
Lets begin with a simple AR(1) model:

```
library(forecast)

ar1_model <- Arima(log10(haddock.data$Biomass), order=c(1,0,0),
                   xreg = haddock.data$Time,
                   include.constant = TRUE) # Include intercept

summary(ar1_model)
```

```
Series: log10(haddock.data$Biomass)
Regression with ARIMA(1,0,0) errors

Coefficients:
         ar1  intercept     xreg
      0.6141     2.5611  -0.0134
s.e.  0.1405     0.1240   0.0054

sigma^2 = 0.02539:  log likelihood = 17.2
AIC=-26.4   AICc=-25.19   BIC=-19.85

Training set error measures:
                      ME      RMSE       MAE          MPE    MAPE      MASE
Training set 0.008715057 0.1529264 0.1141291 -0.07817945 4.91856 0.8889311
                   ACF1
Training set 0.1968254
```

Now we can forecast into the future as follows:

```
forecast_result <- forecast::forecast(
  ar1_model,
  xreg = trend_preds$Time,
  h = 10  # forecast horizon
)
autoplot(forecast_result)
```

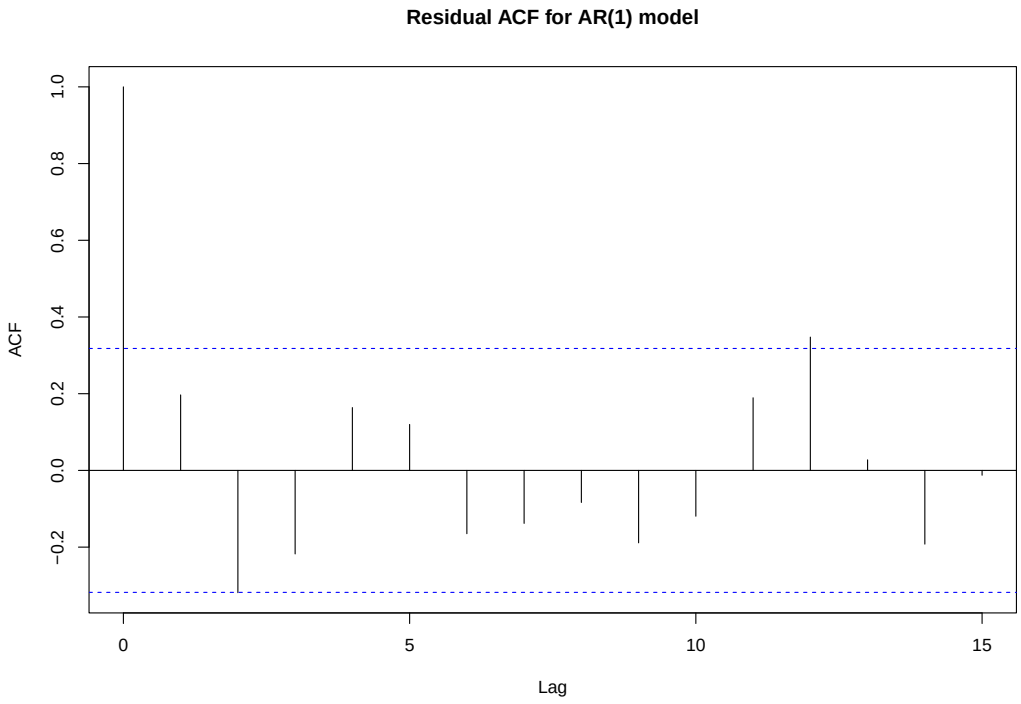



Notice that our predictions are added as a blue solid line, with

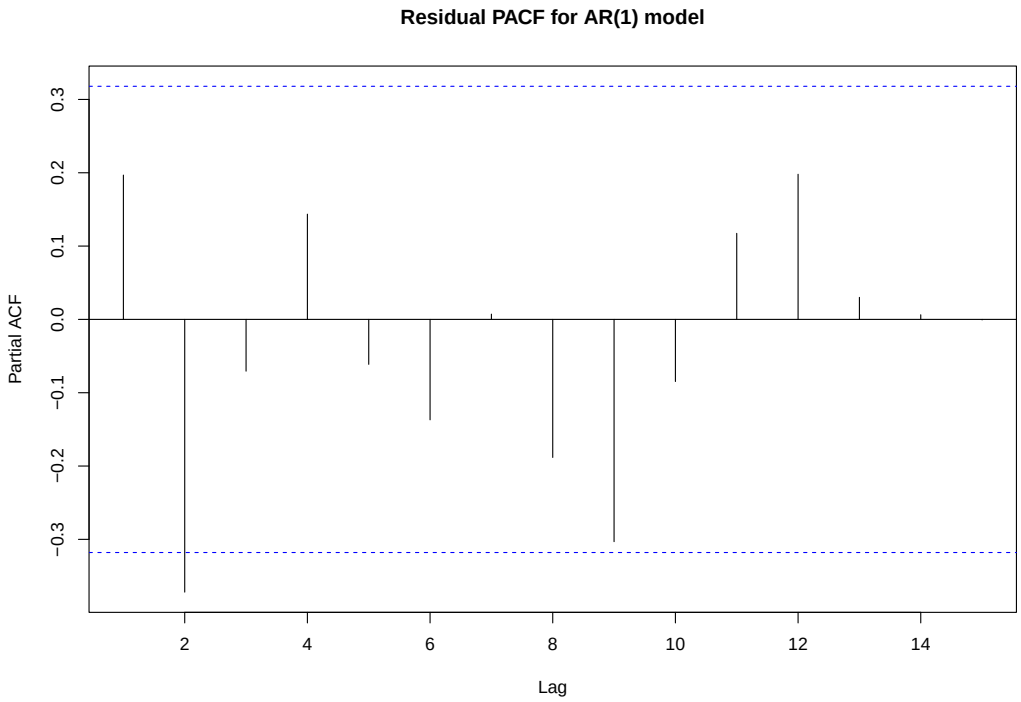
- Darker/lighter inner region indicating 80% prediction interval
- Lighter/outer region indicating 95% prediction interval

We check the time series diagnostic plots:

```
# Check model diagnostic plots:
e1 <- resid(ar1_model)
acf(e1, main = "Residual ACF for AR(1) model")
```



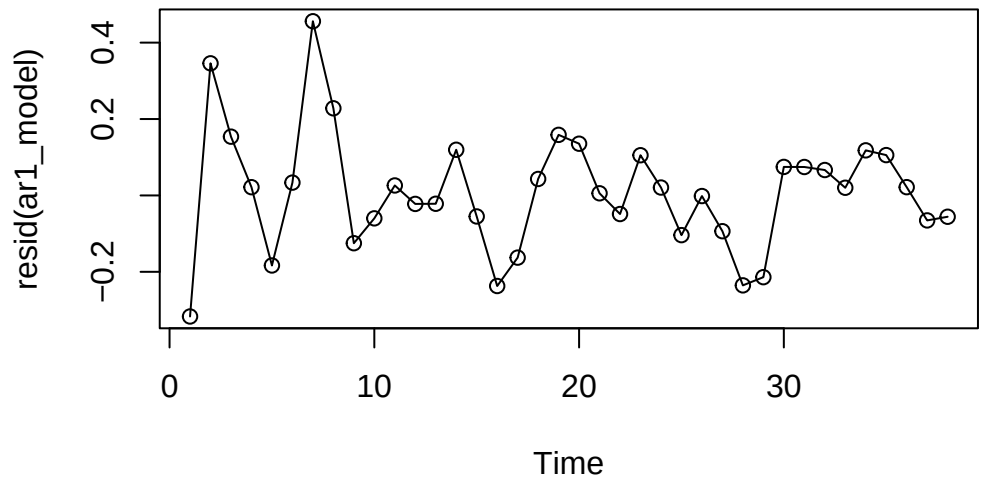

```
pacf(e1, main = "Residual PACF for AR(1) model")
```



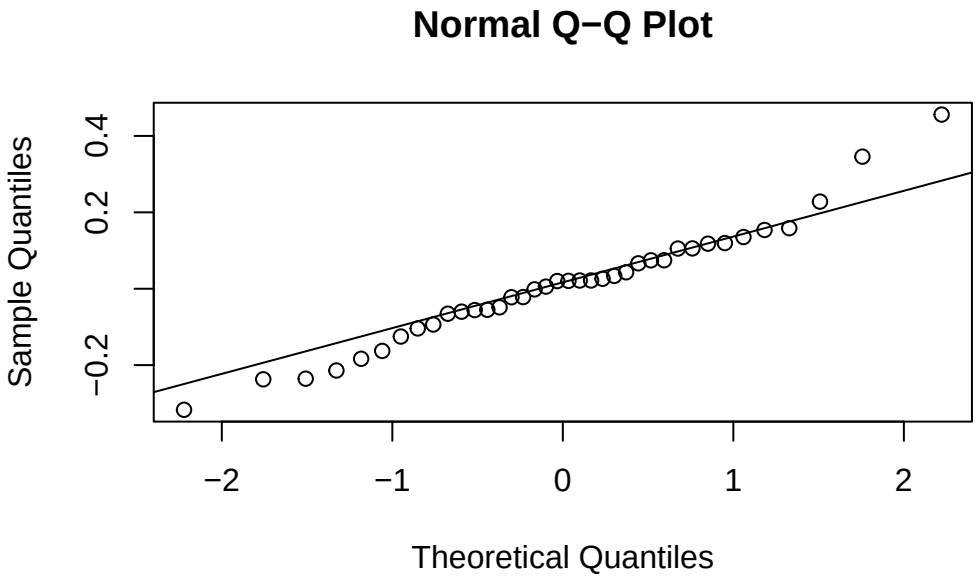

This looks much better, since the lag 1 coefficient is now within the confidence interval bounds, so we no longer have any statistically significant evidence of residual autocorrelation. Therefore, the AR(1) model appears to be appropriate for the data.

We should also check the other diagnostic plots:

```
plot(resid(ar1_model), type = "o")
```




```
qqnorm(resid(ar1_model))
qqline(resid(ar1_model))
```




These do not look too bad — there is little remaining pattern in the plot of residuals versus time (left) and the points seem to follow the line fairly well on the Q-Q plot (right). therefore, this model appears to be appropriate for the data.

# 2 Part 2: Models for extremes

Lets look at extreme data, and makes use of the `extRemes` R package.

The data set `ewe_dailyflow_noleap.txt` provides daily river flows from the River Ewe.

Here we will use some tidy data manipulation & wrangling via the `tidyr` and `dplyr` R packages/ we will also make use of the `lubridate` library to handle date and time variables. Firs lets load the libraries and data:

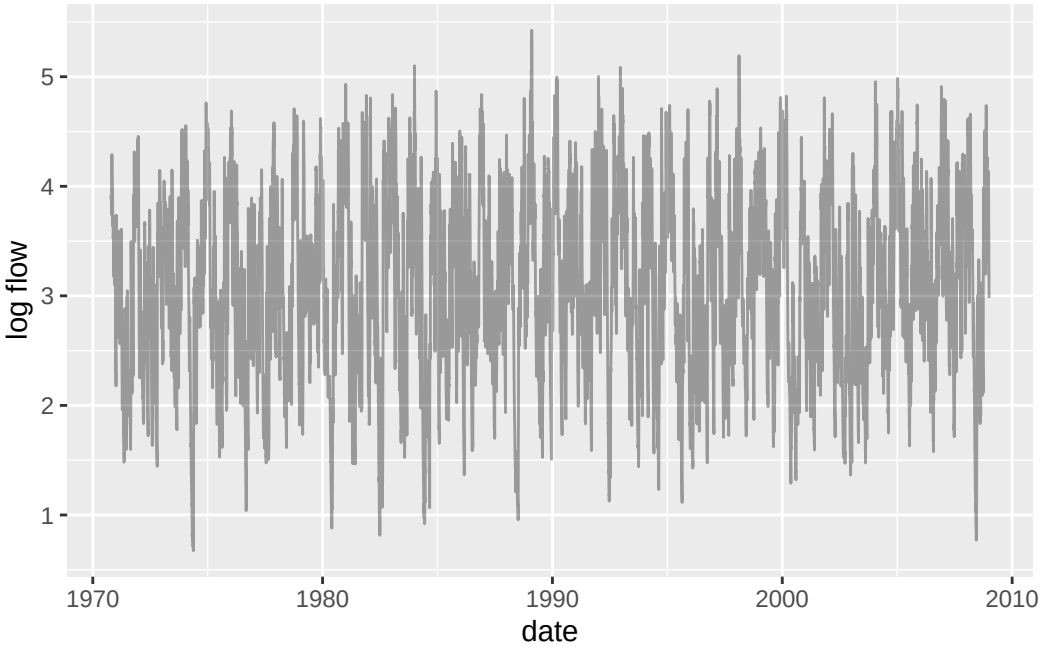
```
library(extRemes)  # library for GEV models
library(lubridate) # library to work with dates and time
library(tidyr)     # library for data tidying
library(dplyr)     # library for data manipulation

#Read the data
ewe <- read.table("datasets/ewe_dailyflow_noleap.txt",header=T)
```

We will work with the log-transformed flow. Here, we will use the `mutate` function fro `dplyr` to create our log-transformed variable named `log_flow`. We will also use this function to create a date variable by combining the information on the year, month and day:

```
ewe <- ewe %>% mutate(
   log_flow = log(flow),  # we'll work with logged data
   date = make_date(year, month, day))
```

Lets visualise our daily time series data:

```
ggplot(data=ewe,aes(y=log_flow,x=date))+
  geom_line(alpha=0.35)+
labs(y="log flow")
```




Suppouse now we want to compute the annual maximum flows for the river Ewe. We can achieve this using the `sumarise` function from `dplyr`:

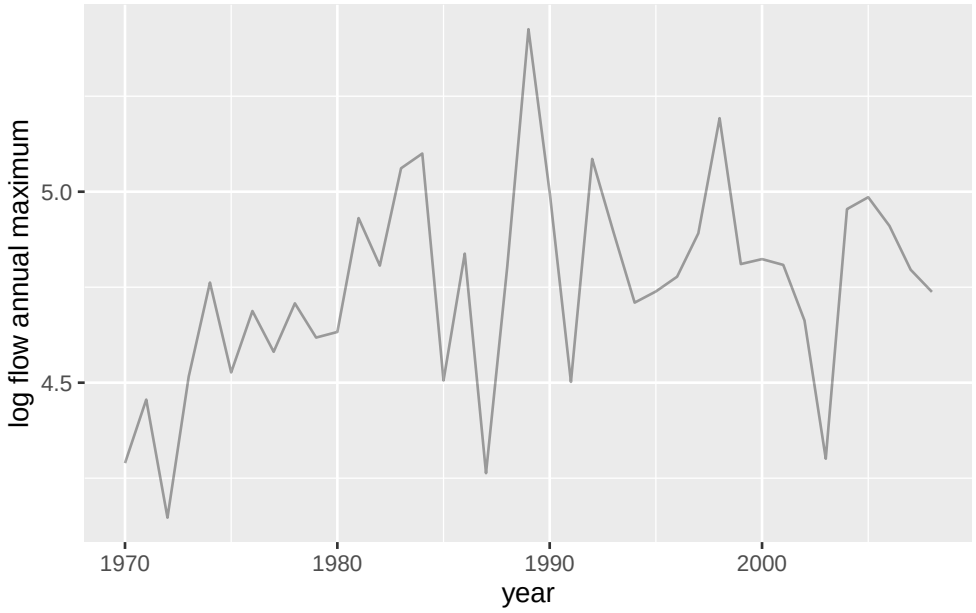
```
ewe_max <- ewe %>% summarise(flow_max= max(log_flow),.by=year)
```

Here, the `.by` argument allows us to specify the grouping variable for which the max function `log_flow` of will be summarised.

Task 4

Plot the annual maximum flows for the river Ewe and comment.
Click here to see the solution

```
ggplot(data=ewe_max,aes(y=flow_max,x=year))+
  geom_line(alpha=0.35)+
labs(y="log flow annual maximum")
```




Question

What does the plot produced in the previous task tells you?
See Solution
From this, we can see that there is a potential increasing trend in the annual maximum flows at the start of the time period, but this does not continue after around 1974. In fact, this apparent trend is only due to the data for a few timepoints, and there is high variability, so that we cannot say that there is clear evidence for a trend here.

Now, we will use the `fevd` function from the `extRemes` library to estimate the GEV parameters using maximum likelihood. This function receives as input a numeric vector.

```
# GEV model fitting
fit_gev <- fevd(ewe_max$flow_max,method="MLE")
results <- summary(fit_gev)
```

```
fevd(x = ewe_max$flow_max, method = "MLE")

[1] "Estimation Method used: MLE"


 Negative Log-Likelihood Value:  3.380743


 Estimated parameters:
  location      scale      shape
 4.6548447  0.2650445 -0.2652807

 Standard Error Estimates:
  location      scale      shape
0.04605764 0.03128801 0.07928741

 Estimated parameter covariance matrix.
              location         scale         shape
```

```
location  2.121306e-03 -1.166577e-05 -0.001252898
scale    -1.166577e-05  9.789396e-04 -0.001335417
shape    -1.252898e-03 -1.335417e-03  0.006286493

 AIC = 12.76149

 BIC = 17.75217
```

**Question**

What are the standard errors for the estimated scale parameter, $\sigma$, in your model (rounded to two decimal places)?

_____

**Question**

What is your estimate of the GEV parameter for location, $\mu$ (rounded to 2 decimal places)?

_____

Running the following code provides us with confidence intervals for the GEV model parameters:

```r
ci(fit_gev, alpha = 0.05, type = c("parameter"))
```

```
fevd(x = ewe_max$flow_max, method = "MLE")

[1] "Normal Approx."

         95% lower CI   Estimate 95% upper CI
location    4.5645734  4.6548447    4.7451160
scale       0.2037211  0.2650445    0.3263678
shape      -0.4206812 -0.2652807   -0.1098802
```

**Question**

Based on the output of the GEV model, which of the three families of the GEV distribution best describes extreme river flow events?
Take hint
Looking at our course notes, we have the following statements:
"The Gumbel, Frechet and Weibull distributions are all special cases (of the GEv distribution) depending on the value of $\xi$ (the shape parameter).

- If $\xi < 0$, then we have the Weibull distribution.
- If $\xi > 0$, then we have the Frechet distribution.
- If $\xi \to 0$, then we have the Gumbel distribution."

- (A) Gumbel
- (B) Frechet
- (C) Weibull

Running the following code provides us with estimates for the 10, 50 and 100 year return levels:

```
return.level(fit_gev,return.period = c(10,50,100))
```

```
fevd(x = ewe_max$flow_max, method = "MLE")
get(paste("return.level.fevd.", newcl, sep = ""))(x = x, return.period = return.period)

 GEV model fitted to  ewe_max$flow_max
Data are assumed to be  stationary
[1] "Return Levels for period units in years"
 10-year level  50-year level 100-year level
      5.103972       5.299083       5.359084
```

We can generate 95% confidence interval estimates for these return levels using the following code:

```
ci(fit_gev, alpha = 0.05, type = c("return.level"),return.period = c(10,50,100))
```

```
fevd(x = ewe_max$flow_max, method = "MLE")

[1] "Normal Approx."

                     95% lower CI Estimate 95% upper CI
10-year return level     4.996722 5.103972     5.211223
50-year return level     5.156316 5.299083     5.441850
100-year return level    5.193936 5.359084     5.524233
```

Since the 100-year return level is 5.36, we can say that a (log) flow of 5.36 is expected to be exceeded once on average, every 100 years.

Question

Based on the output above, state what the approximate return level would be for a 100-year return level - how would you communicate this statement to the a non-expert (e.g., the general public)?
Solution
Since the 100-year return level is 5.36, we can say that a (log) flow of 5.36 is expected to be exceeded once on average, every 100 years.
The interval bounds for the 10-year return level are (5.19, 5.52). A 100-year return level does **not** guarantee that a maximum flow will fall in a fixed range. Instead, it estimates a **threshold that is exceeded** on average once every 100 years.