Tutorial 07: Linear model and prediction assessment

Scottish weather data

In this tutorial, you will use the same data as last week to perform cross validation. Cross-validation is a technique used to assess how well a model is able to predict unseen data.

We consider data from eight weather stations in Scotland, covering the time period from 1 January 1960 to 31 December 2018. Some of the measurements are missing, either due to instrument problems or data collection issues. Load the data from the previous Lab's Learn page (`ghcnd_stations.Rdata` and `ghcnd_values.Rdata`).

The `ghcnd_stations` data frame has 5 variables:

- `ID`: The identifier code for each station
- `Name`: The humanly readable station name
- `Latitude`: The latitude of the station location, in degrees
- `Longitude`: The longitude of the station location, in degrees
- `Elevation`: The station elevation, in metres above sea level The `ghcnd_stations` data frame has 5 variables:
- `ID`: The identifier code for each station
- `Name`: The humanly readable station name
- `Latitude`: The latitude of the station location, in degrees
- `Longitude`: The longitude of the station location, in degrees
- `Elevation`: The station elevation, in metres above sea level

The `ghcnd_values` data frame has 7 variables:

- `ID`: The station identifier code for each observation
- `Year`: The year the value was measured
- `Month`: The month the value was measured
- `Day`: The day of the month the value was measured
- `DecYear`: "Decimal year", the measurement date converted to a fractional value, where whole numbers correspond to 1 January, and fractional values correspond to later dates within the year. This is useful for both plotting and modelling.
- `Element`: One of "TMIN" (minimum temperature), "TMAX" (maximum temperature), or "PRCP" (precipitation), indicating what each value in the `Value` variable represents
- `Value`: Daily measured temperature (in degrees Celsius) or precipitation (in mm)

The `left_join()` function adds copies of the rows from one data frame to another, where one or more columns match. Consider the `ghcnd` data set which for each observation contain both the measurements and the station data:

```r
ghcnd <- left_join(ghcnd_values, ghcnd_stations, by = "ID")
head(ghcnd)
```

```
## # A tibble: 6 x 11
##   ID           Year Month   Day DecYear Element Value Name    Latitude Longitude
##   <chr>       <int> <int> <int>   <dbl> <chr>   <dbl> <chr>      <dbl>     <dbl>
## 1 UKE00105874  1960     1     1   1960  TMAX      3.9 BRAEMAR     57.0     -3.40
## 2 UKE00105874  1960     1     1   1960  TMIN      3.3 BRAEMAR     57.0     -3.40
## 3 UKE00105874  1960     1     2   1960. TMAX      7.2 BRAEMAR     57.0     -3.40
## 4 UKE00105874  1960     1     2   1960. TMIN     -8.3 BRAEMAR     57.0     -3.40
## 5 UKE00105874  1960     1     3   1960. TMAX      6.7 BRAEMAR     57.0     -3.40
## 6 UKE00105874  1960     1     3   1960. TMIN     -6.7 BRAEMAR     57.0     -3.40
## # i 1 more variable: Elevation <dbl>
```

Choose one of the stations, and create a new data variable `data` from `ghcnd` with the yearly averages of TMIN as a column with missing values removed with `filter()`.

Solution:

```
data <- ghcnd %>%
  filter(ID == "UKE00105875") %>%
  pivot_wider(names_from = Element, values_from = Value) %>%
  filter(!is.na(TMIN)) %>%
  group_by(ID, Name, Year) %>%
  summarise(TMIN = mean(TMIN), .groups = "drop")
```

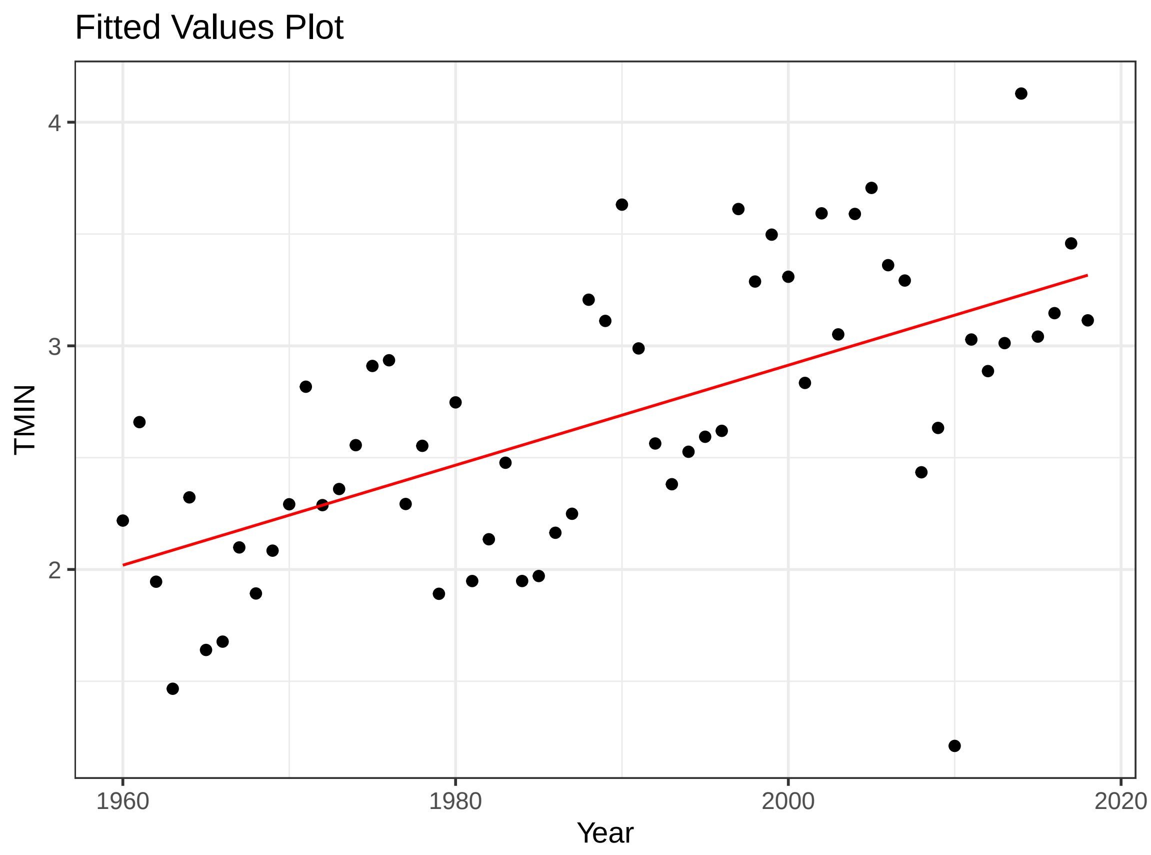
Using `data` estimate a linear model for TMIN using `Year` as a covariate. Plot the observed values of `TMIN` against `Year` using ggplot2 together with a line for the fitted model.

The first thing to do with a linear model is to check its assumptions: 1. the errors are independent with zero mean and constant variance, and less importantly, 2. that they are normally distributed. The usual way to check this is with residual plots. Plot the result from the `lm` function and analyse each plot.

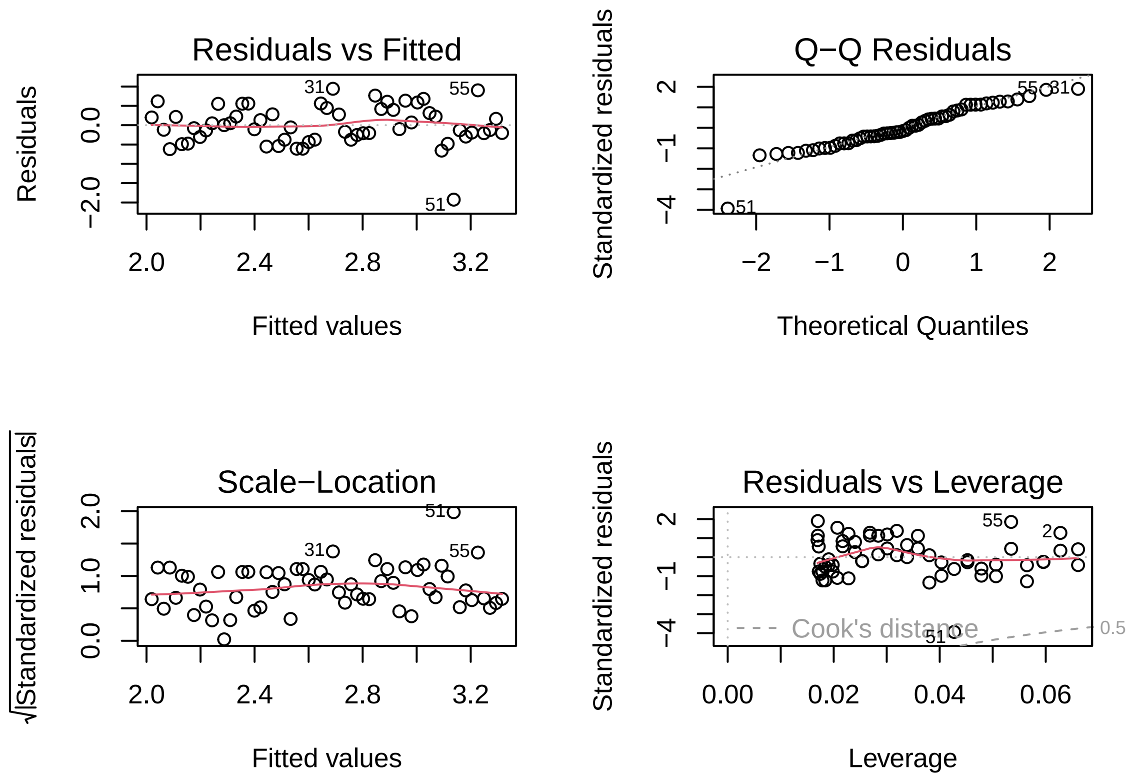
```
fit0 <- lm(TMIN ~ 1 + Year, data = data)

# Extract the fitted values
fitted_values <- fitted(fit0)

# Plot the fitted values against the observed values
data %>%
ggplot() +
     geom_point(aes(x = Year, y = TMIN)) +
     geom_line(aes(x = Year, y = fitted_values), color = "red") +
     labs(title = "Fitted Values Plot", x = "Year",y = "TMIN")
```



```
par(mfrow=c(2,2))
plot(fit0)
```



Compute the squared error and the 80% Interval score based on the whole data set.

Solution:

```
  pred0 <- predict(fit0, newdata = data,
                interval = "prediction", level = 0.8)

  score.se  <- mean((data$TMIN - pred0[, "fit"])^2)

  score.int <- mean(pred0[,"upr"] -  pred0[, "lwr"] + 2/0.2 * (( pred0[, "lwr"] - data$TMIN) *
            (data$TMIN <  pred0[, "lwr"]) + (data$TMIN - pred0[,"upr"]) * (data$TMIN > pred0[,"upr"])))

  scores.all = c(score.se, score.int)
```

Cross validation

We now want to compute the scores from cross validation based on a random sample of the data. Leave 20% of the data for prediction and use 80% for estimation.

Solution:

```
  n_train <- floor(nrow(data)*0.8)
  data_train <- data[1:n_train,]
  data_valid <- data[-(1:n_train),]

  fit1 <- lm(TMIN ~ 1 + Year, data = data_train)

  pred1 <- predict(fit1, newdata = data_valid,
                interval = "prediction", level = 0.8)

  score1.se  <- mean((data_valid$TMIN - pred1[, "fit"])^2)

  score1.int <- mean(pred1[,"upr"] -  pred1[, "lwr"] + 2/0.2 * (( pred1[, "lwr"] - data_valid$TMIN) *
            (data_valid$TMIN <  pred1[, "lwr"]) + (data_valid$TMIN - pred1[,"upr"]) * (data_valid$TMIN
```

k-fold cross validation

We now want to compute the 5 average 80% Interval scores from 5-fold cross validation based on a random partition of the data into 5 approximately equal parts.

First add a new column Group to data defining the partitioning, using `mutate()`. One approach is to compute a random permutation index vector, and then use the modulus operator `%%` to reduce it to 5 values, or `ceiling()` on scaled indices.

Solution:

```
data <-
  data %>%
  mutate(Group = sample(seq_len(nrow(data)), size = nrow(data), replace = FALSE),
         Group = (Group %% 5) + 1)
# Alternative:
# data <-
#  data %>%
#  mutate(Group = sample(seq_len(nrow(data)), size = nrow(data), replace = FALSE),
#         Group = ceiling(Group / nrow(data) * 5))
```

Loop over the partition groups, estimating the model leaving the group out, and then predicting and scoring predictions for the group.

Solution:

```r
scores.int.k <- numeric(5)
scores.se.k <- numeric(5)
for (grp in seq_len(5)) {
  fit <- lm(TMIN ~ 1 + Year, data = data %>% filter(Group != grp))

  pred <- predict(fit, newdata = data %>% filter(Group == grp),
                     interval = "prediction", level = 0.8)

  obs <- (data %>% filter(Group == grp)) %>% pull("TMIN")

  scores.se.k[grp]  <- mean((obs - pred[, "fit"])^2)

  scores.int.k[grp] <- mean(pred[,"upr"] -  pred[, "lwr"] + 2/0.2 * (( pred[, "lwr"] - obs) *
            (obs <  pred[, "lwr"]) + (obs - pred[,"upr"]) * (obs > pred[,"upr"])))
}
```

Compare the resulting scores from using the whole data set with the simple and k-fold cross validation scores. Use `knitr::kable` to visualize both scores.

The average cross validation scores for this problem are larger than the one for the whole data set. It is intended to reduce the risk of underestimating the prediction error, so this is what we would expect. You can repeat the random group allocation to see how much it influences the cross validation score average.

```r
knitr::kable(data.frame("Whole data SE" = score.se,
                        "Cross validation SE" = score1.se,
                        "K-fold SE" = mean(scores.se.k)))
```

Whole.data.SE	Cross.validation.SE	K.fold.SE
0.2416402	0.7150074	0.2664038

```r
knitr::kable(data.frame("Whole data Int" = score.int,
                        "Cross validation Int" = score1.int,
                        "K-fold Int" = mean(scores.int.k)))
```

Whole.data.Int	Cross.validation.Int	K.fold.Int
1.644106	3.059963	1.719205