Chapter 8: Trees – predicting ozone concentration

To get started with trees, we use a very small dataset: ozone measurements from New York City taken in 1973. You will need the package `partykit` to fit trees to the data.

1. Data preparation

- The ozone measurements are in the package `datasets` and can be loaded with `data("airquality")`.
- Convert `Month` and `Day` to day of year with `as.POSIXlt(paste(1973, airquality$Month, airquality$Day), format = "%Y %m %d")$yday`
- Explore the data with `str()`, `pairs()` and discover more about the dataset with the R help operator: `?airquality`.
- Convert the units to SI: Langley to Joule per square meter, miles per hour to meters per second, and Fahrenheit to Celsius.
- Subset to non-missing ozone data and drop `Month` and `Day`, using `subset()` and assign to `airq`. (Dropping works with the option `select = -c(Month, Day)`).

2. Fit tree Fit the tree with `partykit::ctree(formula, data = ...)` using all variables and assign to `airQTree`. The "partykit" package contains many functions, which can be called using the "::" syntax. For the usual formula syntax, remember the abbreviation ".", which uses all variables in the data set except for the response variable to the left of the "~" symbol. (for the curious: `?formula` gives you more special characters that you can use to specify your formula; https://thomasleeper.com/Rcourse/Tutorials/formulae.html gives examples).

Many tree-growing algorithms exist that partition the observations by univariate splits in a recursive way and then fit a constant model in each cell of the resulting partition. The most popular implementations perform an exhaustive search over all possible splits and maximize an information metric of node impurity so that the elements in one node are as homogeneous as possible and the elements between nodes as heterogeneous as possible. This approach has two fundamental problems: overfitting and a selection bias towards covariates with many possible splits. `ctree` addresses these problemes. It computes conditional inference trees and performs multiple statistical tests (remember the null hypothesis?) to achieve an unbiased selection and to determine when to stop growing a node further. From `?ctree`: "Roughly, the algorithm works as follows: 1) Test the global null hypothesis of independence between any of the input variables and the response (which may be multivariate as well). Stop if this hypothesis cannot be rejected. Otherwise select the input variable with strongest association to the response. This association is measured by a p-value corresponding to a test for the partial null hypothesis of a single input variable and the response. 2) Implement a binary split in the selected input variable. 3) Recursively repeate steps 1) and 2)."

3. Examine and plot the resulting tree model

- Examine a summary of the fitted tree by simply typing `airQTree` (or `print(airQTree)`). The number given after the cutpoint (splitting value) is the average value of the response variable (ozone, in our case) in this *terminal* node; the number given for `err` is the sum-of-squares error.
- Plot the tree `airQTree`. The figure shows the tree structure with split variables and cut points, including Bonferroni-corrected p-values. The terminal nodes are boxplots showing the response variable (ozone) on the y-axis with the number of cases that fall into this node. (The Bonferroni correction is a simple adjustment to p-values when many hypotheses need to be tested at the same time, as is the case with trees, where multiple predictors need to be evaluated to achieve the best split).