Tasks chapter8: Trees – predicting ozone concentration

David Kurz, Malte Hildebrandt
2023S707737 VU Geostatistik
Universität Innsbruck

May 24, 2023

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)).

   → Ozone_SI is the response variable, which is used in the formula to indicate that all other variables in the airq dataset should be included as predictors. The resulting tree is stored in the airQTree object.

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).

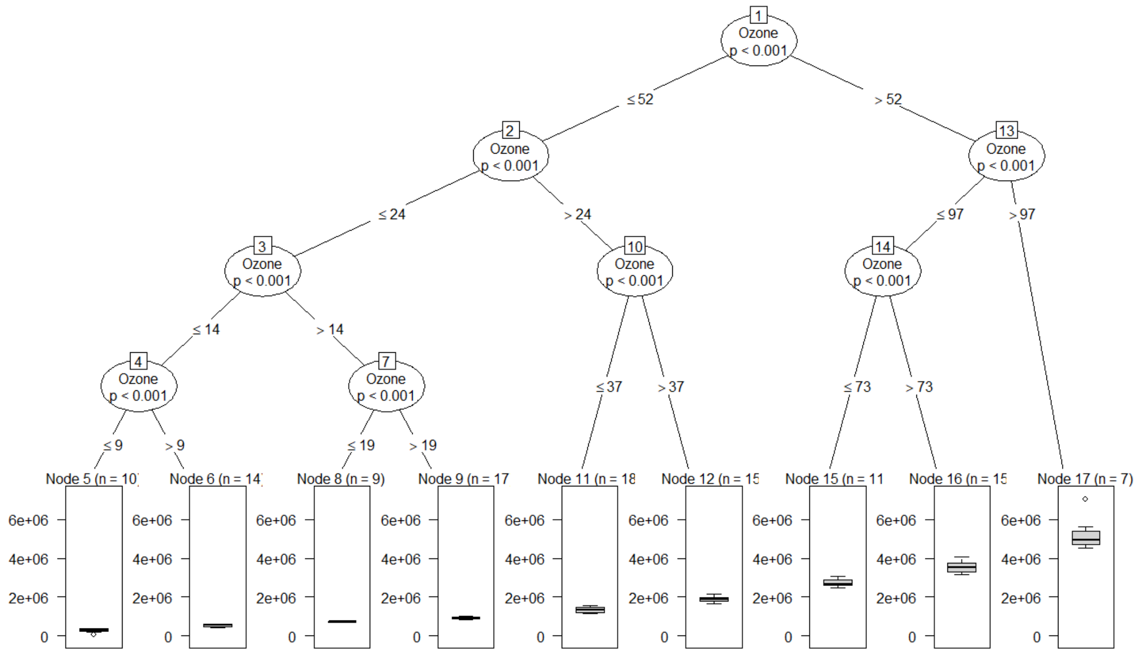
   Many tree-growing algorithms exist that partition the observations by univariate splits in a recursive way and then fit a constant model in each cell of the resulting partition. The most popular implementations perform an exhaustive search over all possible splits and maximize an information metric of 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 "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)."

The tree algorithm works as follows:

- It tests the global null hypothesis of independence between any of the input variables and the response.
- If this hypothesis cannot be rejected, the algorithm stops growing the tree.
- If the global null hypothesis is rejected, the algorithm selects the input variable with the strongest association to the response.
- The association is measured by a p-value corresponding to a test for the partial null hypothesis of a single input variable and the response.
- It implements a binary split in the selected input variable.
- Steps 1-3 are repeated recursively to grow the tree further.
- By performing statistical tests and making unbiased variable selections, the ctree algorithm helps address overfitting and selection bias commonly associated with other tree-growing algorithms.

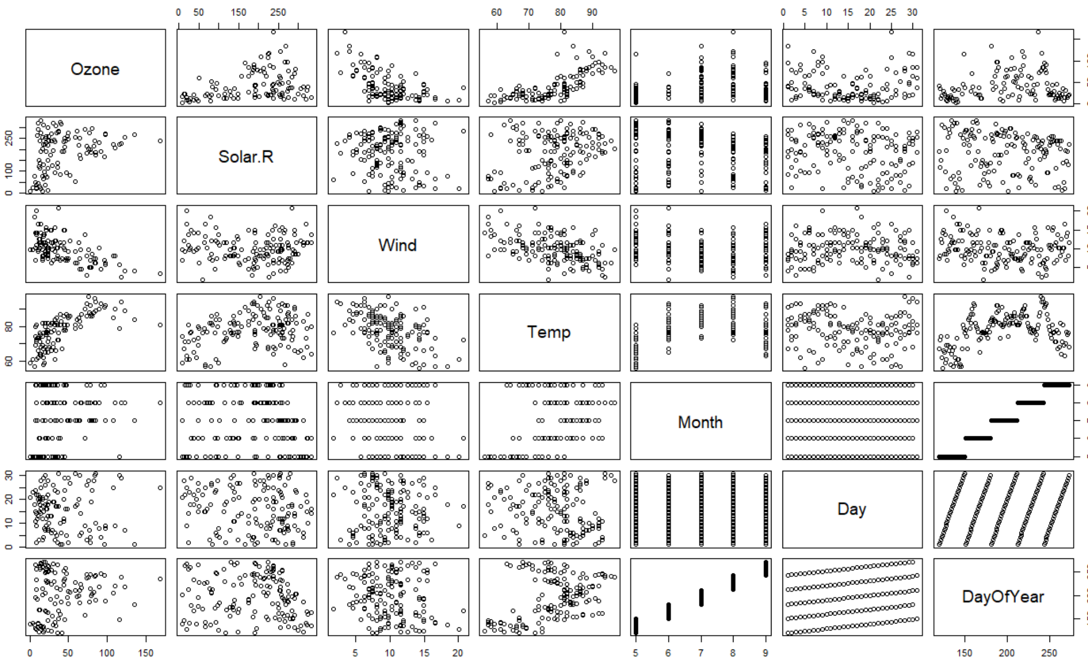
3. **Examine and plot the resulting tree model**

   - 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).



```
Fitted party:
| [1] root
| [2] Ozone <= 52
| | [3] Ozone <= 24
| | | [4] Ozone <= 14
| | | | [5] Ozone <= 9: 280328.000 (n = 10, err = 1.01709e+11)
| | | | [6] Ozone > 9: 523000.000 (n = 14, err = 41138761600.0)
| | | [7] Ozone > 14
| | | | [8] Ozone <= 19: 720577.778 (n = 9, err = 20228989155.6)
| | | | [9] Ozone > 19: 915557.647 (n = 17, err = 59108007905.9)
| | [10] Ozone > 24
| | | [11] Ozone <= 37: 1341204.444 (n = 18, err = 341266937244.4)
| | | [12] Ozone > 37: 1877221.333 (n = 15, err = 3.84662e+11)
| [13] Ozone > 52
| | [14] Ozone <= 97
| | | [15] Ozone <= 73: 2731010.909 (n = 11, err = 451969373090.9)
| | | [16] Ozone > 73: 3553610.667 (n = 15, err = 1.356587e+12)
| | [17] Ozone > 97: 5235977.143 (n = 7, err = 4.588035e+12)

Number of inner nodes: 8
Number of terminal nodes: 9
```


Ozone
Solar.R
Wind
Temp
Month
Day
DayOfYear