Task chapter 6: Model selection and regularization

This time we will use a data set for the temperature of Innsbruck and 36 predictor variables from the Global Ensemble Forecast System GEFS, which has a much coarser spatial resolution than our previous data set from ECMWF. The NWP forecasts are for a lead time of 24 hours. The data set also contains 5 time/season variables: a linear trend component `time`; annual harmonic waves `sin` and `cos`, and half-annual waves `sin2` and `cos2`.

First load the library `lmSubsets`. Then load the data set contained in that library with `data("IbkTemperature", package = "lmSubsets")` You will get more information about the data set with `?IbkTemperature`.

Your tasks are:

Build 4 reference forecast models:

- For a 24-h temperature forecast, the autocorrelation is still very high. We will therefore create a persistence reference forecasting model, which uses the *observed* temperature from the previous day as predictor. Create a new variable `IbkTemperature$lagTemp`. Since the data set contains some missing values, remove all of them after creating the lagged temperature variable but before proceeding further, using `na.omit()`.
- Build a persistence reference model with `lm` on the *whole* data set and assign it to `lmPers`. Then similarly build a one-variable reference model with the GEFS predictor `t2m` and assign it to `lmT2m`. Use `summary()` for both models to check which one is better and comment on the results. Also hypothesize on the reason for the magnitude of the intercept term of the lmT2m-model.
- Next fit the same models but with `glm()` and estimate the CV-errors (contained in `delta` where the `[1]` gives the standard k-fold CV error and `[2]` gives a bias-corrected version) with `cv.glm()` using 10-fold cross validation. Add "glm" to the model names. Is the ranking of the models still the same now that the models are also exposed to unseen data?
- The second set of references models adds the time/seasonal terms to the persistence model and the NWP-T2m model, respectively. For the persistence model this means, `temp ~ lagTemp + time + sin + cos + sin2 + cos2`. Use cross-validation (`cv.glm()`) to determine which of the 4 model has the lowest CV-error and is thus best.

In the second main task you are going to identify which subset of predictors gives the best model, using BIC as metric (for BIC see James2021, p234 and eq. 6.3. For simplicity, just fit the models to the complete data set. Normally you would need to use cross validation.

- First use the best subset model approach of the `lmSubsets` package, which uses several clever tricks to be computationally very efficient and still arrive at close to the truly best subset models. Use `lmSelect()` and assign it to `MOSPers_best` (MOS stands for "model output statistics"). R saves you a lot of typing with the "." shortcut notation to mean "all variables of the data set not already included on the left side of the ~ symbol: `temp ~ .` Force the model to always include `lagTemp` by specifying `include = "lagTemp"`. Use BIC as metric with `penalty = "BIC"` and be content with computing the 20 best subset models with `nbest = 20`. (Note that you could force the inclusion of more than 1 predictor in all subsets with `include = c("predictor1, predictor2, ...")` and that you could also exclude particular predictors with `exclude =`). What predictors are included in the best model? Do the selected predictors vary much among the 20 best subset models? Use `image()` which has been customized in the lmSubsets package to nicely visualize the model results order from best to worst. `image(MOSPers_best, hilite = 1, lab_hilite = "bold(lab)", pad_size = 2, pad_which = 2)`. Do `?image.lmSelect` to discover the meaning of the options. The number for `hilite` is usually 1, so that the best subset model is highlighted. Also the predictors for this highlighted subset are set apart in bold as specified with the `lab_hilite` option. Print the coefficients of the best subset model with `coef(MOSPers)`. If you do a `summary(IbkTemperature)` you will notice the vastly different magnitudes of the predictor variables. To handle that, glmnet first standardizes them (subtract mean and divide by standard variation). However, when showing `coef`, they are transformed back to original data magnitudes.
- Next, do an exhaustive search through all possible model configurations by using `lmSubsets()` including all predictor variables. Assign to `MOS_all`. What are the best subset models with respect to BIC as metric? Again, answer based on a visualization with `image()`. You will need two additional options: `size` to select models with a particular number of predictor variables (e.g. `3:27`); and `hilite_penalty = BIC`. Also print the coefficients of the best model.

In the third main task you fit a model with regularization using lasso.

The package glmnet performs that task. A quick and helpful introduction is available by the authors of the package at:

https://glmnet.stanford.edu/articles/glmnet.html. Unfortunately, you cannot use a formula to specify the model in that package but must split it into x (the predictor variables) and y (the response), see James2021 p274.

```
x <- model.matrix(temp ~ ., data = IbkTemperature)[, -1]
```

(the `[, -1]` excludes the first column, since this is `temp` - our response variable)

```
y <- IbkTemperature$temp
```

- Fit lasso using `glmnet()`; you only need to specify x and y; per default `alpha = 1`, i.e. lasso is selected as method.
- Perform cross validation using `cv.glmnet()` and assign it to `cvfit`
- Plot the result of the cross validation using `plot(cvfit)` to visualize the mean-squared error as a function of the logarithm of the penalty factor lamda. The first dashed vertical line marks the lambda for which the CV error is minimal. You can get its value with `cvfit$lambda.min`. The second dashed vertical line marks the lambda that gives the most regularized model such that the cross-validated error is within one standard error of the minimum `cvfit$lambda.1se`. This is often used to determine the final regularized model. Any value of lambda between lambda.min and lambda.1se, however, is fine. A smaller value means that more predictors will still be included. How many predictors are included for lambda.min, and how many for lambda.1se? You get their coefficients (and names), e.g. with `coef(cvfit, s = "lambda.min")`
- Compare the predictors included in the lasso with `lambda.min` and `lambda.1se`, respectively, with the ones obtained with best subset selection and exhaustive subset selection, respectively.