Tasks chapter6:

David Kurz, Malte Hildebrandt
2023S707737 VU Geostatistik
Universität Innsbruck

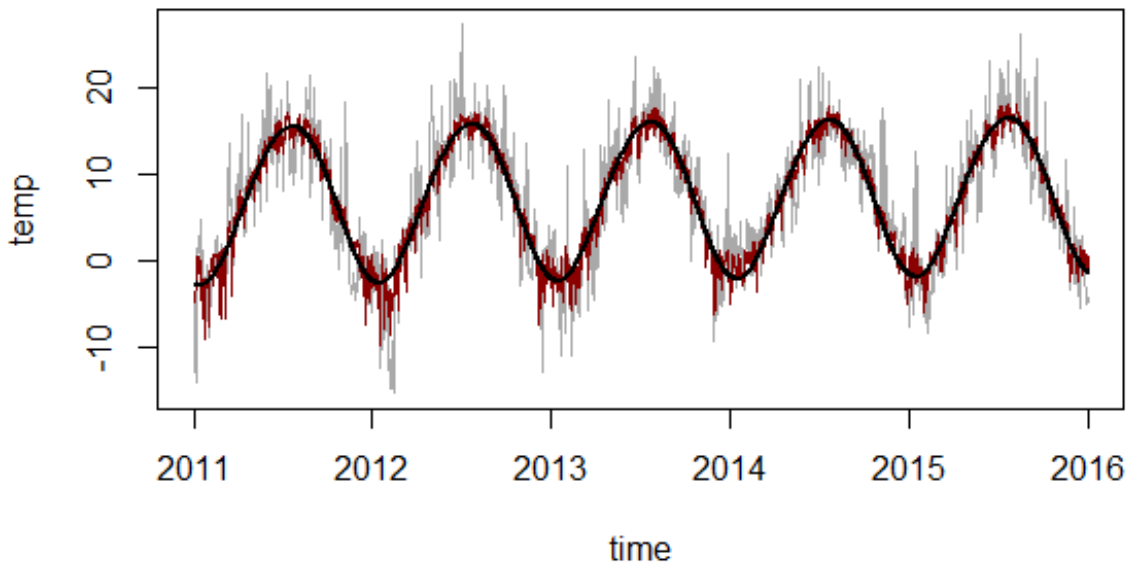
May 2, 2023

1 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.

	Min	**1Q**	**Median**	**3Q**	**Max**
lmPers	-4.504e-13	-3.000e-17	2.500e-16	5.400e-16	1.083e-14
lmT2m	-15.8198	-2.8877	0.1267	2.8957	16.0661

	Estimate	**Std. Error**	**t value**	**Pr(>|t|)**
lmPers (interc.)	9.330e-15	3.472e-16	-2.687e+01	<2e-16
lmT2m (interc.)	-201.5491	3.7175	-54.22	<2e-16
lmPers lagTemp	1	3.367e-17	2.970e+16	<2e-16
t2m	0.7675	0.1366	56.18	<2e-16

	res. std. error	**mult. R^2**	**adj. R^2**	**F-stat.**
lmPers	1.059e-14 on 1817 d.o.f.	1	1	8.823e+32 on 1 and 1817 DF
lmT2m	4.46 on 1817 d.o.f.	0.6346	0.6344	3156 on 1 and 1817 DF

⇒ In the next part we compare the coefficients, R-squared, and p-values of both models to assess their performance. The model with higher R-squared and lower p-values is considered better. In this case it's clearly the Model for persistence forecast. Additionally we can check the magnitude of the intercept term for lmT2m.

- 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?

	Std. k-fold CV-Error	**its bias corr.**
glmPers	3.998e-28	1.08e-28
glmT2m	19.90106	19.89940

⇒ Now we compare the CV-errors for both models. A lower CV-error indicates a better performance of the model. If we check the ranking of the models compared to the lm() models, we see that the persistance model is still a lot better than the other t2m.

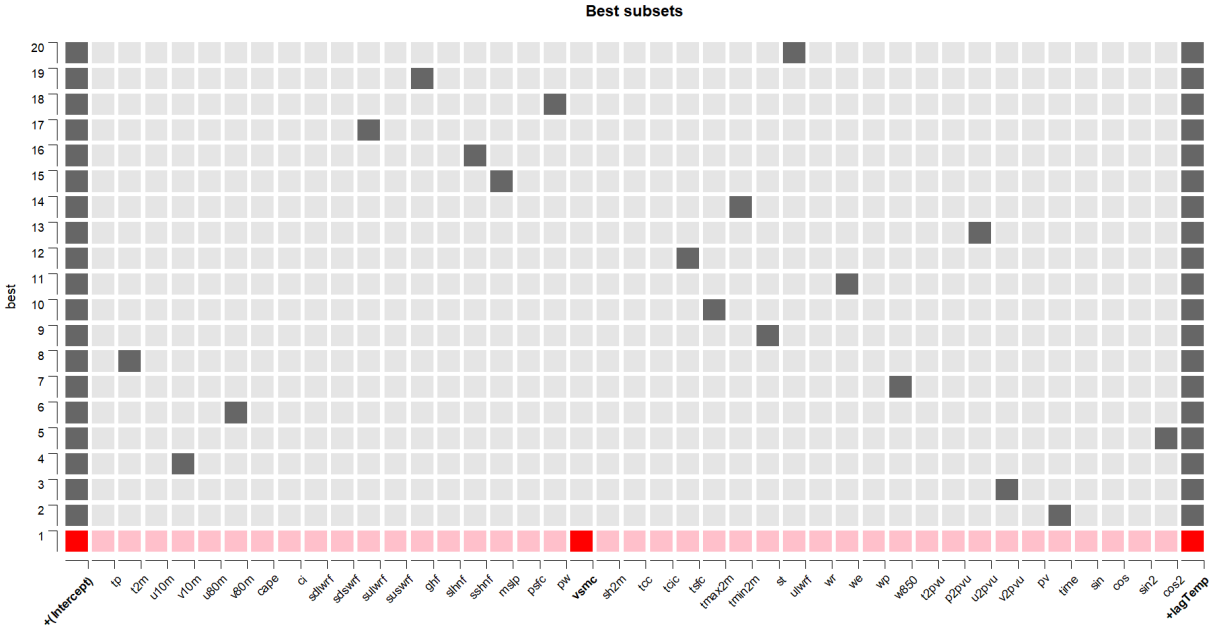
- The second set of references models adds the time/seasonal terms to the persistence model and the NWP-T2m model, respectively. For the persistence model this means, temp $\sim$ lagTemp + time + sin + cos + sin2 + cos2. Use cross-validation (cv.glm()) to determine which of the 4 model has the lowest CV-error and is thus best.

	Std. k-fold CV-Error	**its bias corr.**
glmPers time	4.536820e-28	1.246159e-28
glmT2m time	10.81991	10.81484

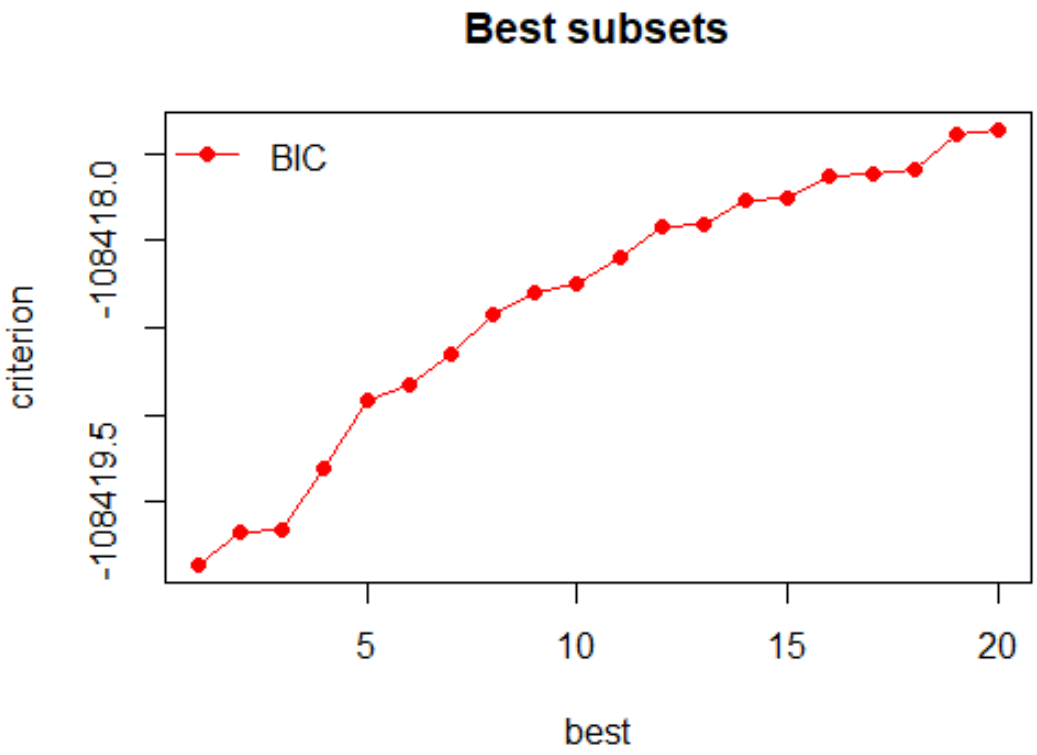
⇒ If we consider the time/seasonal terms in our persistance and T2m model, we observe an increasing performance in the T2m model but a slight decrease in the persistance model, which is still the better model.

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

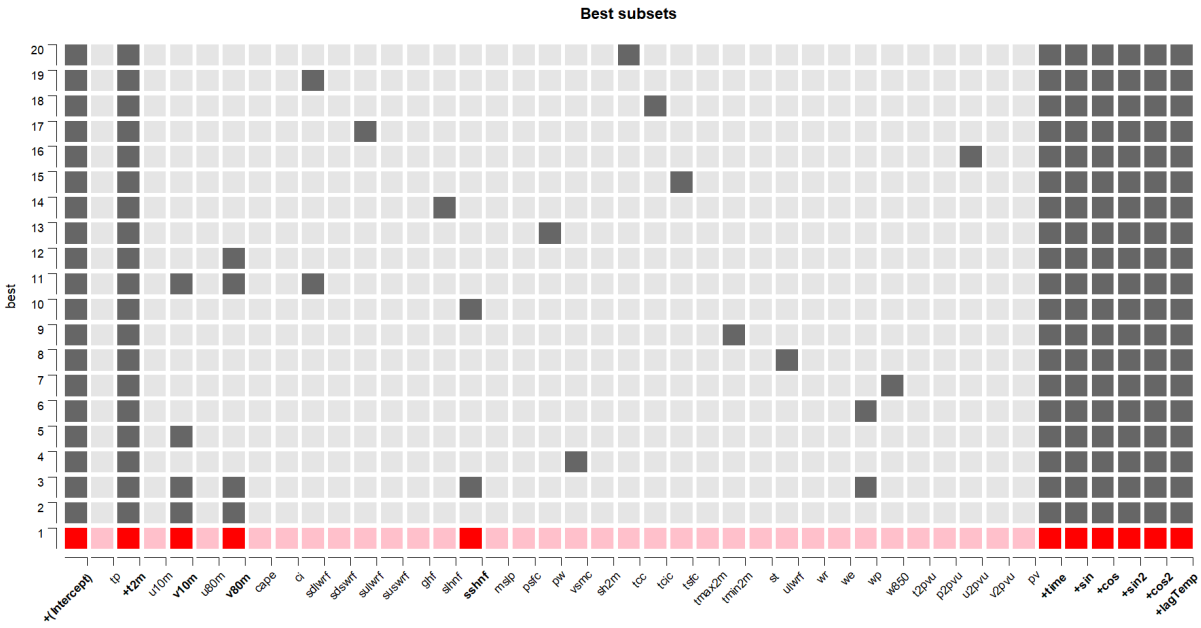
- First use the best subset model approach of the lmSubsets package, which uses several clever tricks to be computationally very efficient and still arrive at close to the truly best subset models. Use lmSelect() and assign it to MOSPers_best (MOS stands for "model output statistics"). R saves you a lot of typing with the "." shortcut notation to mean "all variables of the data set not already included on the left side of the $\sim$ symbol: temp $\sim$. 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.

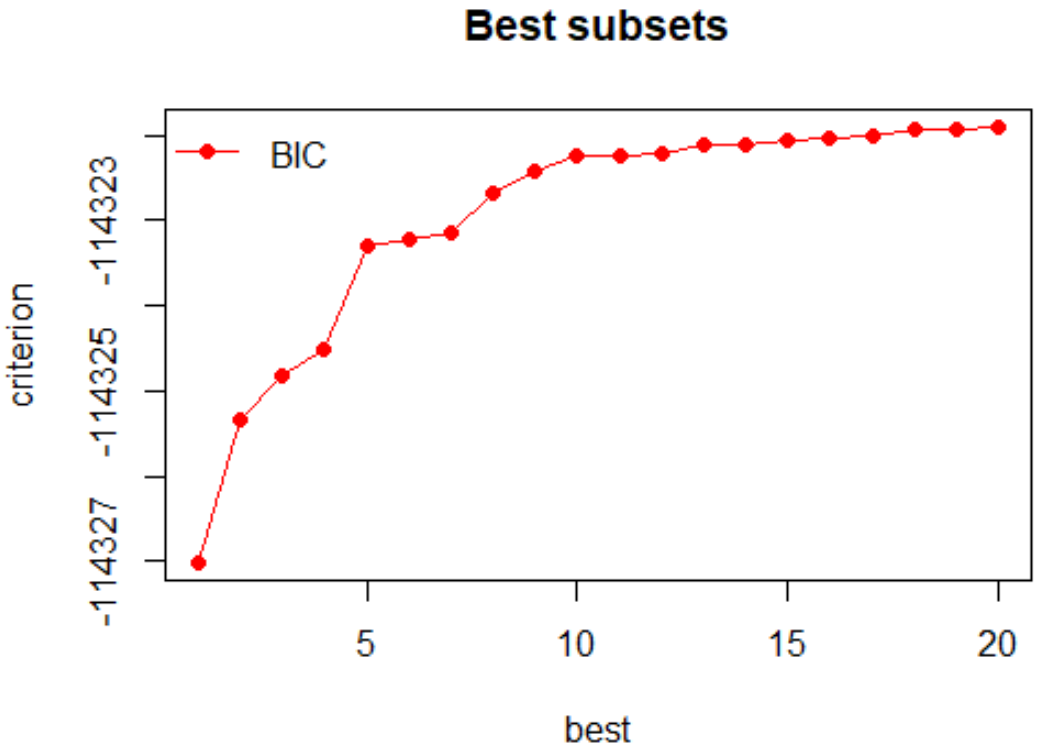


(Intercept): -1.831461e-13
vsmc: 3.974066e-13
lagTemp: 1.000000e+00



⇒ The best component of the MOS_best object contains only the lagTime as a predictor included for finding the best model. In this case the best model coefficient is vscm with it's values above (inclusivlely intercept and lagTime). In the first best subset plot, the best model is highlighted and its predictors set apart in bold.

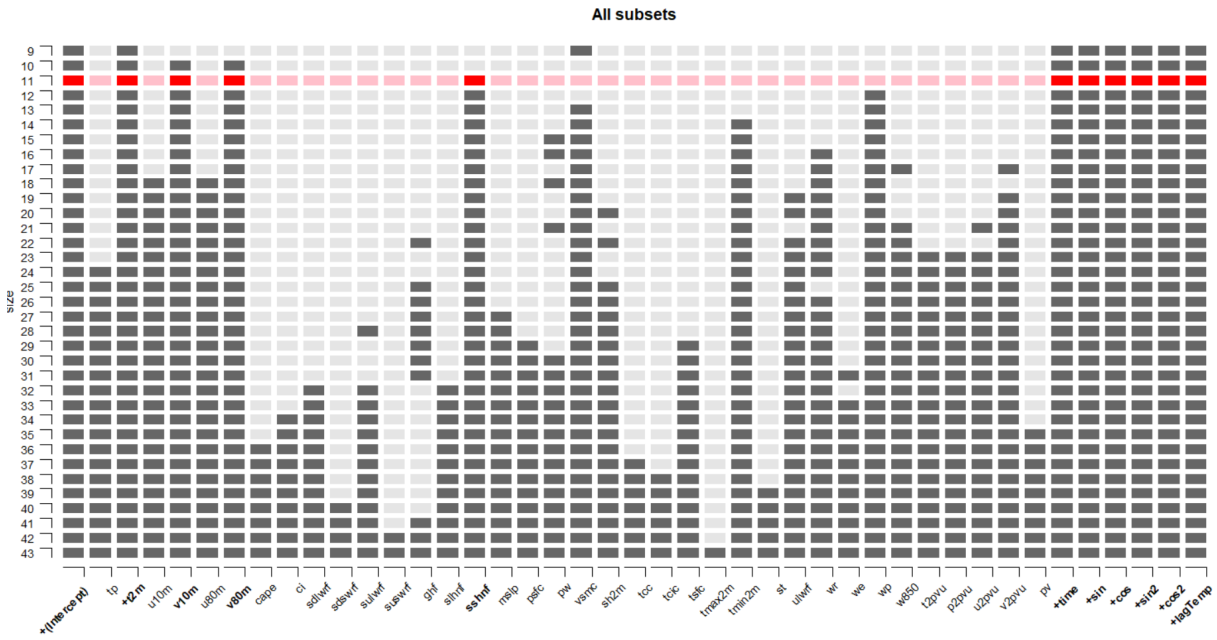

Best subsets
best
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
+(Intercept)
tp
+t2m
u10m
v10m
u80m
v80m
cape
ci
sdlwrf
sdswrf
sulwrf
suswrf
ghf
shnf
sshnf
msp
pstc
pw
vsmc
sh2m
tcc
toc
tslc
tmax2m
tmin2m
st
ulwrf
wr
we
wp
w850
t2pvu
p2pvu
u2pvu
v2pvu
pv
+time
+sin
+cos
+sin2
+cos2
+lagTemp

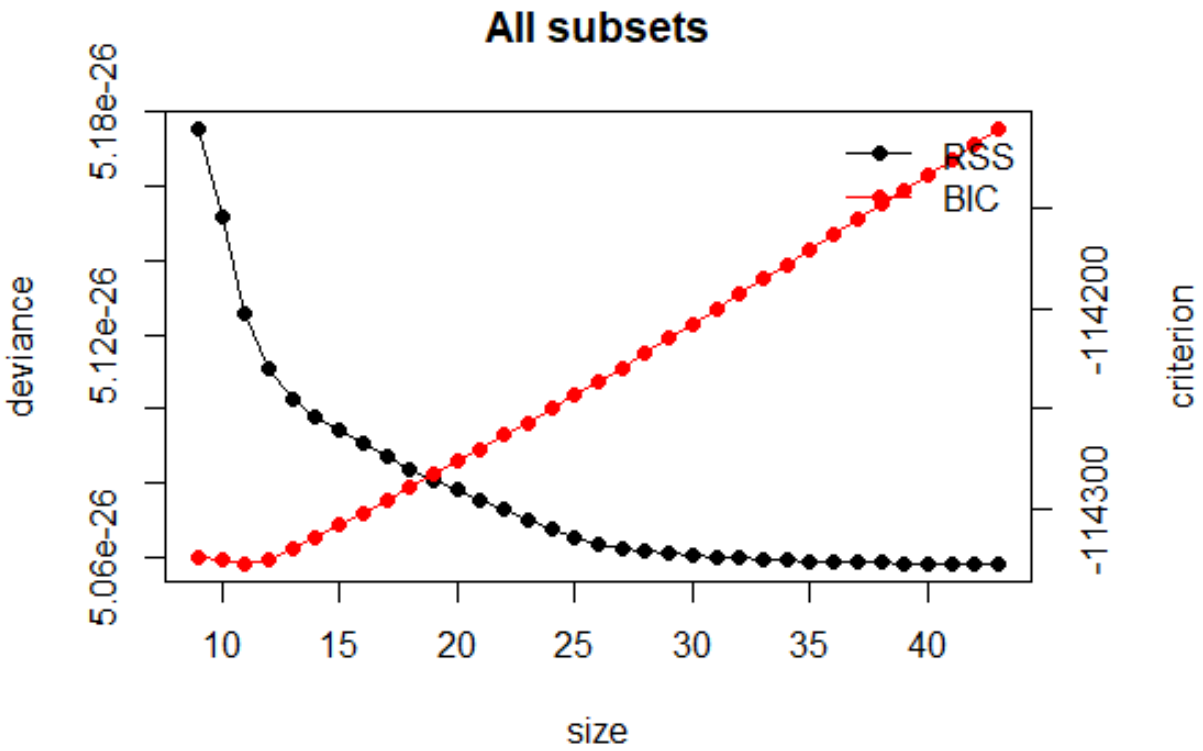

Best subsets
BIC
criterion
-114323
-114325
-114327
5
10
15
20
best

(Intercept)	t2m	v10m	v80m	sshnf	time
4.080416e-13	-3.735221e-16	-2.473317e-18	-1.679755e-17	-1.907575e-17	-1.564069e-16
sin	**cos**	**sin2**	**cos2**	**lagTemp**	
1.388696e-15	6.946478e-15	-2.395113e-16	2.988505e-16	1.000000e+00	

⇒ The best component of the MOS_best_all object contains lagTime, t2m, time, sin, cos, sin2, cos2 as a predictors.

- 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.


All subsets
size
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
+(tmx tcc pl)
tp
+2m
u10m
v10m
u80m
v80m
cape
ci
sdlwrf
sdswrf
sulwrf
suswrf
ghf
slhrf
ssfrnf
mslp
psfc
pw
vsmc
sh2m
tcc
tcic
tslc
tmax2m
tmin2m
st
ulwrf
wr
we
wp
w850
t2pvu
p2pvu
u2pvu
v2pvu
pv
+time
+sin
+cos
+sin2
+cos2
+lagTemp

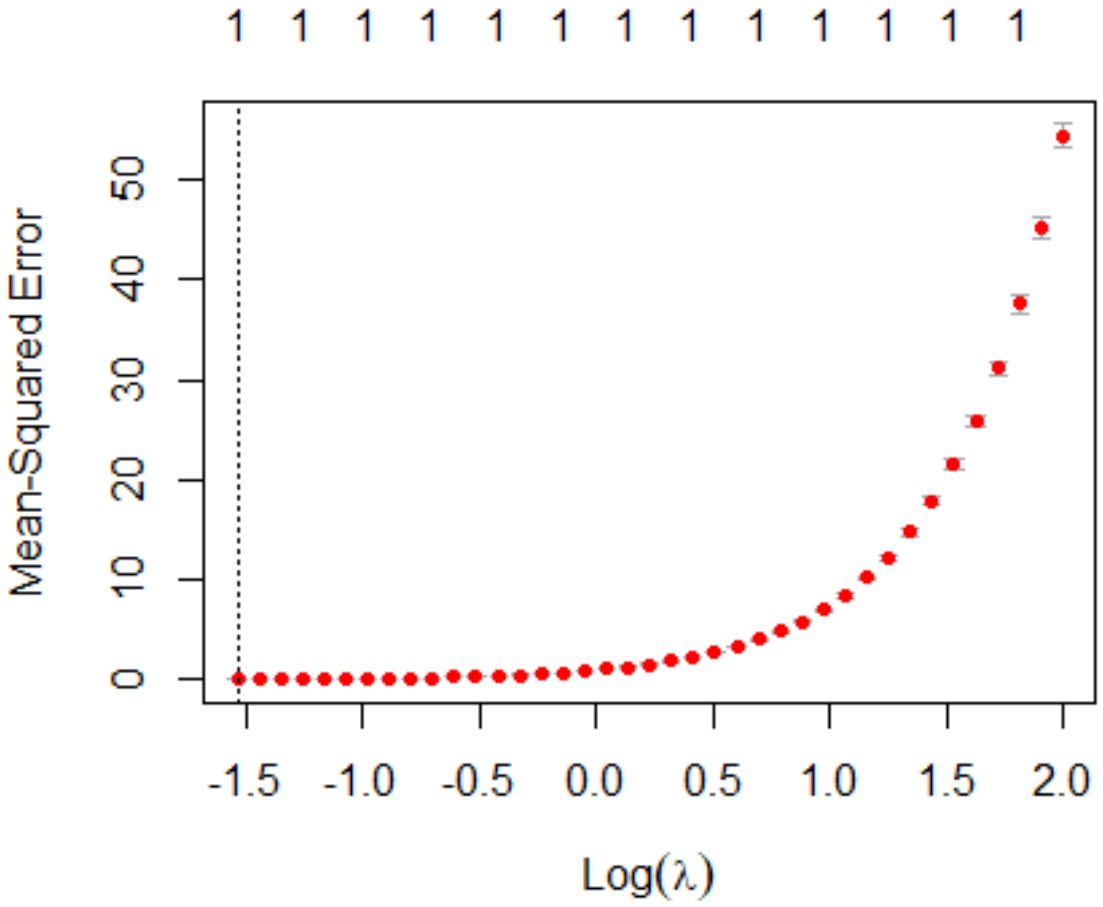

All subsets
deviance
5.06e-26
5.12e-26
5.18e-26
criterion
-114300
-114200
RSS
BIC
10
15
20
25
30
35
40
size

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.


1 1 1 1 1 1 1 1 1 1 1 1 1
Mean-Squared Error
0
10
20
30
40
50
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
Log(λ)