Bern University
of Applied Sciences

ISRIC
World Soil Information

ISRIC spring school – Hands on Digital Soil Mapping

Machine learning 2:
Understanding methods, model selection and interpretation

Practical training

Madlene Nussbaum, 31 May 2018

Contents

Preparation

Load needed packages (maybe install first):

```
# install.packages(c("grpreg", "glmnet", "randomForest", "geoGAM"),
#                  dependencies = T)
library(grpreg) # for group lasso
library(glmnet) # for lasso
library(randomForest) # for random forest
library(geoGAM) # to get the Berne data set
```

As an example you can work with the Berne soil mapping study area: dataset `berne` in R package `geoGAM`, contains continuous, binary and a multinomial/ordered response and a spatial data `berne.grid` for prediction. This prediction grid is only a small subset of the agricultural area in the study area (CRAN does not allow larger files).

Load the data, select the calibration set and remove missing values in covariates:

```
dim(berne)

## [1] 1052  238

# continuous response, topsoil pH in 0-10 cm
d.ph10 <- berne[berne$dataset == "calibration" & !is.na(berne$ph.0.10), ]
d.ph10 <- d.ph10[complete.cases(d.ph10[13:ncol(d.ph10)]), ]

# ordered/multinomial tesponse, degree of waterlogging,
# 3 levels aggregated from subqualifiers of Swiss soil classification
d.drain <- berne[berne$dataset == "calibration" & !is.na(berne$dclass), ]
d.drain <- d.drain[complete.cases(d.drain[13:ncol(d.drain)]), ]

# covariates start at col 13 (see help page ?berne)
l.covar <- names(d.ph10[, 13:ncol(d.ph10)])
```

1 Lasso – linear shrinkage method

Lasso for continuous response

The `berne` dataset contains categorical covariates (factors, e.g. geological map with different substrate classes). The group lasso (R package `grpreg`) ensures that all dummy covariates of one factor are excluded (coefficients set to 0) together or remain in the model as a group.

The main tuning parameter λ is selected by cross validation. λ determines the degree of shrinkage that is applied to the coefficients.

HINT for R newbies: the `apply`-functions in R are replacements for loops (`sapply`: loop over a sequence of numbers, `lapply`: loop over a list). Compared to `for`, an `apply` is much faster and general coding style, though a bit more tricky to program.

Example, how to replace a `for` by a `sapply`:

```
# loop
# first create a vector to save the results
t.result <- c()
for( ii in 1:10){ t.result <- ii^2}
# the same as apply
t.result <- sapply(1:10, function(ii){ ii^2 })
# of course, this example is even shorter using:
t.result <- 1:10^2
```

Now we create the setup using `apply` and fit the grouped lasso:

```
# define groups: dummy coding of a factor is treated as group
# find factors
l.factors <- names(d.ph10[l.covar])[
  t.f <- unlist( lapply(d.ph10[l.covar], is.factor) ) ]
l.numeric <-  names(t.f[ !t.f ])
```

```
# create a vector that labels the groups with the same number
g.groups <- c( 1:length(l.numeric),
               unlist(
                 sapply(1:length(l.factors), function(n){
                   rep(n+length(l.numeric), nlevels(d.ph10[, l.factors[n]])-1)
                 })
               )
)
# grpreg needs model matrix as input
XX <- model.matrix( ~., d.ph10[, c(l.numeric, l.factors), F])[,-1]

# cross validation (CV) to find lambda
ph.cvfit <- cv.grpreg(X = XX, y = d.ph10$ph.0.10,
                      group = g.groups,
                      penalty = "grLasso",
                      returnY = T) # access CV results
```

Compute predictions for the validation set with optimal number of groups chosen by lasso:

```
# choose optimal lambda: CV minimum error + 1 SE (see glmnet)
l.se <- ph.cvfit$cvse[ ph.cvfit$min ] + ph.cvfit$cve[ ph.cvfit$min ]
idx.se <- min( which( ph.cvfit$cve < l.se ) ) - 1

# select validation data
d.ph10.val <- berne[berne$dataset == "validation" & !is.na(berne$ph.0.10), ]
d.ph10.val <- d.ph10.val[complete.cases(d.ph10.val[13:ncol(d.ph10)]), ]

# create model matrix for validation set
newXX <- model.matrix( ~., d.ph10.val[, c(l.factors, l.numeric), F])[,-1]
t.pred.val <-  predict(ph.cvfit, X = newXX,
                       type = "response",
                       lambda =  ph.cvfit$lambda[idx.se])
# get CV predictions, e.g. to compute R2
ph.lasso.cv.pred <- ph.cvfit$Y[,idx.se]
```

Get the lasso (non-zero) coefficients of the optimal model:

```
# get the non-zero coefficients:
t.coef <- ph.cvfit$fit$beta[, idx.se ]
t.coef[ t.coef > 0 ]

##                (Intercept)                  cl_mt_gh_4
##               2.6081447850               0.0061779207
##   tr_se_curvplan2m_std_10c   tr_se_curvplan2m_std_25c
##               0.0084819290               0.0177733319
##   tr_se_curvplan2m_std_50c                  tr_vdcn25
##               0.0024671794               0.0006273042
##         timesetd1979_2010    timesetd1968_1974_field
##               0.1146614892               0.2089107882
## ge_geo500h3idTorfbedeckung
##               0.0786699618
```

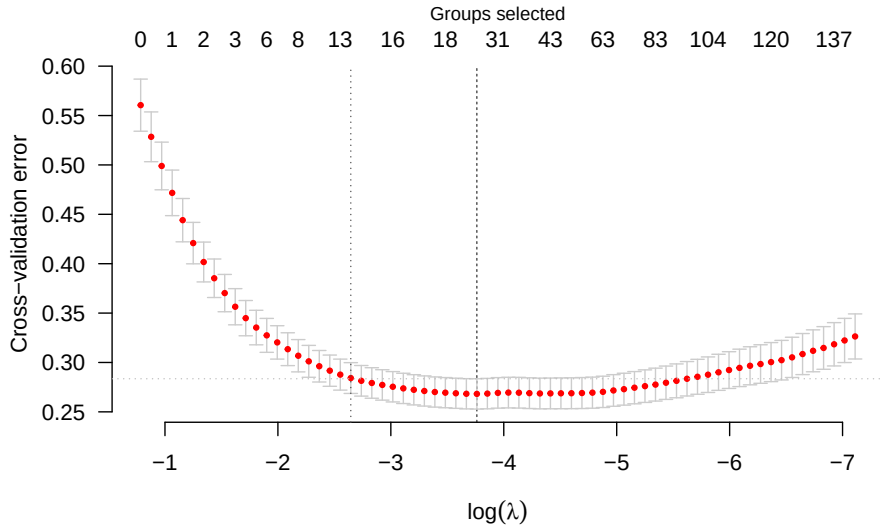

Figure 2: Cross validation error plotted against the tuning parameter lambda. The dashed line indicates lambda at minimal error, the dotted darkgrey line is the optimal lambda with minimal error + 1 SE.

Lasso for multinomial response

I am not aware of a lasso implementation for multinomial responses that can handle groups of factors. Therefore, we use "standard" lasso from R package `glmnet` (the option `type.multinomial = "grouped"` does only ensure all coefficients of the multinomial model for the same covariate are treated as groups).

```
# create model matrix for drainage classes
# use a subset of covariates only, because model optimization for
# multinomial takes long otherwise

set.seed(42) # makes sample() reproducible
XX <- model.matrix(~.,d.drain[, l.covar[sample(1:length(l.covar), 30)]])[,-1]

drain.cvfit <- cv.glmnet( XX, d.drain$dclass, nfold = 10,
                          keep = T, # access CV results
                          family = "multinomial",
                          type.multinomial = "grouped")
```

For getting the coefficients of the final model you run the `glmnet` function again with the selected λ. Please note: The multinomial fit results in a coefficient for each covariate and response level.

```
drain.fit <- glmnet( XX, d.drain$dclass,
                     family = "multinomial",
                     type.multinomial = "grouped",
                     lambda = drain.cvfit$lambda.min)
# The coeffs are here:
# drain.fit£beta£well
# drain.fit£beta£moderate
# drain.fit£beta£poor
```

Lasso - Questions:

1. How does the model change with λ? What does a small λ mean, what a large λ?

2. We applied above group lasso (`grpreg`) and "standard" lasso (`glmnet`). When would you use which? Why?

3. The function `model.matrix()` is called in the background by `lm` (ordinary least squares linear model). Here we used it explicitly. What does it do? (Hint: use `str()` and check e.g. for column `ge_caco3`)

4. Compared to a linear model (e.g. ordinary least squares by `lm()`): How do you interpret the coefficients you get from lasso?

2 Selection of covariates by random forest

For tree based ensemble methods like random forest or gradient boosting with trees covariate importance can be computed. Based on this measure non-relevant covariates can be excluded and model performance can possibly be increased.

Fit random forest model to topsoil pH of `berne` data:

```
rf.ph <- randomForest(x = d.ph10[, l.covar],
                      y = d.ph10$ph.0.10)
```

Create the importance plot:

```
varImpPlot(rf.ph, n.var = 20, main = "")
```

Then, reduce covariates by *recursive backward elimination* using the covariate importance as plotted above:

```
# we could remove one covariate after another, this would take long...
# speed up the process by removing 5-10 covariates at a time,
# create a sequence of covariate numbers for this
s.seq <- sort( c( seq(5, 95, by = 5),
                  seq(100, length(l.covar), by = 10) ),
               decreasing = T)

# collect results in list
qrf.elim <- oob.mse <- list()

# save model and OOB error of current fit
qrf.elim[[1]] <- rf.ph
oob.mse[[1]] <- tail(qrf.elim[[1]]$mse, n=1)
l.covar.sel <- l.covar
```

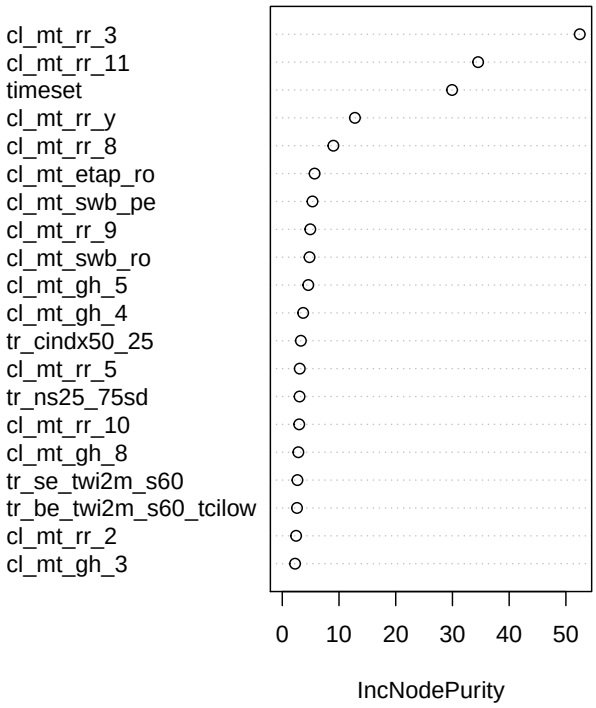

Figure 3: Covariate importance of 20 most important covariates for topsoil pH (before selection).

```
# iterate through number of retained covariates
for( ii in 1:length(s.seq) ){
  # compute importance and order it
  t.imp <- importance(qrf.elim[[ii]])
  t.imp <- t.imp[ order(t.imp[,1], decreasing = T),]

  # fit random forest (RF) to reduced covariate set
  # only select as many as in s.seq defined above
  qrf.elim[[ii+1]] <- randomForest(x = d.ph10[, names(t.imp[1:s.seq[ii]])],
                                   y = d.ph10$ph.0.10 )
  # save OOB error from RF fit
  oob.mse[[ii+1]] <- tail(qrf.elim[[ii+1]]$mse,n=1)
}
# prepare a data frame for plot
elim.oob <- data.frame(elim.n = c(length(l.covar), s.seq[1:length(s.seq)]),
```

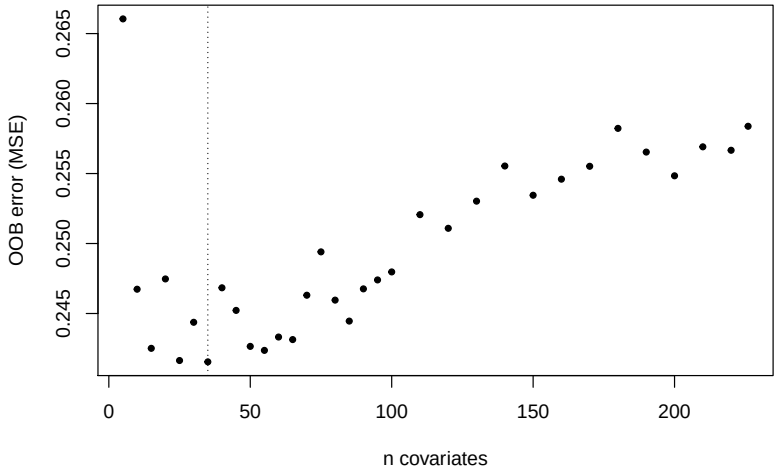

Figure 4: Path of out-of-bag mean squared error as covariates are removed.

Random forest model selection – Questions:

1. The covariate importance of the full model (Figure 3) attributed large importance to similarly named covariates (e.g. `cl_mt_rr..`: monthly and yearly rainfall pattern). Check if they are correlated (`cor()`). Why are they all ranked with large importance? Hint: think of the parameter `mtry` in random forest.

2. You must have waited quite a while for the results of the backward elimination of covariates. Was it worth it or not? Why?

3. Are the covariates selected at the minimal OOB error the best choice? Suggestions considering the plot 4?

4. Compared to the non-zero coefficients do you find the same covariates? Why are there differences? How would you explain these differences to a soil surveyor without in depth statistical knowledge?

3 Model interpretation

Partial resiudal plots

To demonstrate the principle we create a partial residual plot for an ordinary least squares fit. Then, we add the same plot for the lasso fitted on the topsoil pH data above:

```
# create a linear model (example, with covariates from lasso)
ols <- lm( ph.0.10 ~ timeset + ge_geo500h3id + cl_mt_gh_4 +
             tr_se_curvplan2m_std_25c, data = d.ph10 )
par(mfrow = c(1,2)) # two plots on same figure
```

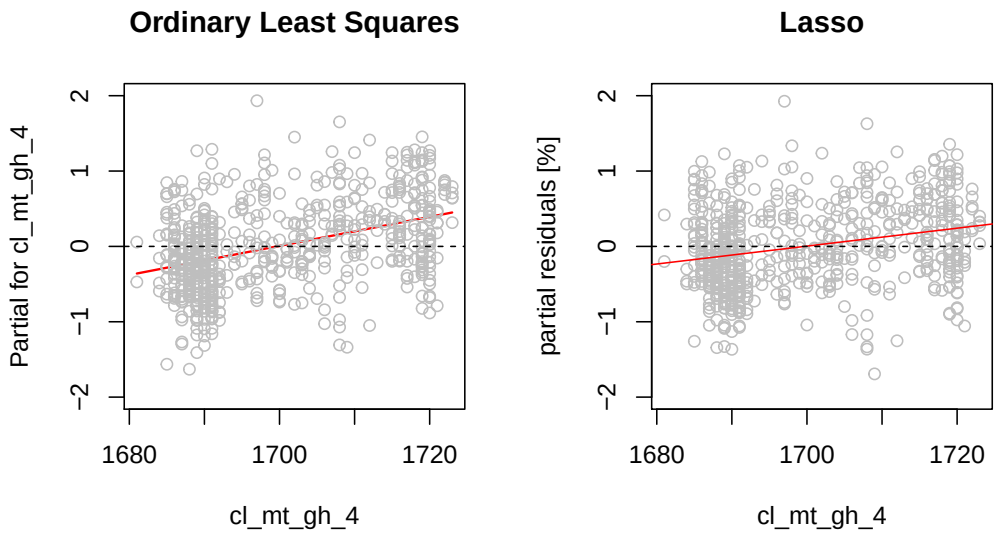

Figure 5: Partial residual plots for a climate covariate in the ordinary least squares fit and the lasso.

```
# residual plot for covariate cl_mt_gh_4
termplot(ols, partial.resid = T, terms = "cl_mt_gh_4",
         ylim = c(-2,2),
         main = "Ordinary Least Squares")
abline(h=0, lty = 2)

## Create partial residual plot for lasso
# there is no direct function available, but we can easily
# construct the plot with
# y-axis: residuals + effect of term (XBi), scaled
# x-axis: values covariate
# regression line: model fit of axis y~x

# get the index of the covariate
idx <- which( names(t.coef) == "cl_mt_gh_4" )

# residuals of lasso model chosen above
residuals <- d.ph10$ph.0.10 - ph.cvfit$Y[,idx.se]
# prediction for this covariate XBi
Xbeta <- ph.cvfit$fit$beta[idx, idx.se] * d.ph10$cl_mt_gh_4
# calculate partial residuals and center with mean
part.resid <- scale(residuals + Xbeta, scale = F)[,1]

# plot with similar settings
plot(d.ph10$cl_mt_gh_4,
     part.resid, pch = 1, col = "grey",
     ylim = c(-2,2),
     ylab = "partial residuals [%]", xlab = "cl_mt_gh_4",
     main = "Lasso")
abline(lm(part.resid ~ d.ph10$cl_mt_gh_4), col = "red")
abline(h=0, lty = 2)
```

Partial dependence plots

Interpretation of the most important covariates of a random forest model can be done by partial dependence plots. But keep in mind that the remaining covariates after model selection might be still multi-collinear, hence covariates might be exchangeable.

Partial dependence plots are a general method and can be also applied to other supervised machine learning methods.

```
# select the model with minimum OOB error
rf.selected <- qrf.elim[[ which.min(elim.oob$elim.OOBe)]]

t.imp <- importance(rf.selected, type = 2)
t.imp <- t.imp[ order(t.imp[,1], decreasing = T),]

# select 4 most important covariates for a plot
( t.3 <- names( t.imp[ 1:4 ] ) )

## [1] "cl_mt_rr_3"  "cl_mt_rr_11" "timeset"     "cl_mt_rr_y"

par( mfrow = c(2,2))
# Bug in partialPlot(): function does not allow a variable for the
# covariate name (e. g. x.var = name) in a loop, hence repeat the code
partialPlot(x = rf.selected,
            pred.data = d.ph10[, names(rf.selected$forest$xlevels)],
            x.var = "cl_mt_rr_3", ylab = "ph [-]", main = "")
partialPlot(x = rf.selected,
            pred.data = d.ph10[, names(rf.selected$forest$xlevels)],
            x.var = "cl_mt_rr_11", ylab = "ph [-]", main = "" )
partialPlot(x = rf.selected,
            pred.data = d.ph10[, names(rf.selected$forest$xlevels)],
            x.var = "timeset", ylab = "ph [-]", main = "" )
partialPlot(x = rf.selected,
            pred.data = d.ph10[, names(rf.selected$forest$xlevels)],
            x.var = "cl_mt_rr_y", ylab = "ph [-]", main = "" )
```

Model interpretation – Questions:

1. In Figure 5 the slope of the partial effect for `cl_mt_gh_4` is steeper for the ordinary least squares fit. Why?

2. The covariate `cl_mt_rr_3` is more important than `cl_mt_rr_y`. From what do you see that in partial dependence plots in Figure 6?

3. On which rainfall value [mm*10] of covariate `cl_mt_rr_11` do the trees in random forest split most often (Figure 6)?

4. What do you conclude from the partial residual and dependence plots? Are the plotted covariates good predictors? How would the plots look like if the covariates were very good or very bad predictors? Please draw.

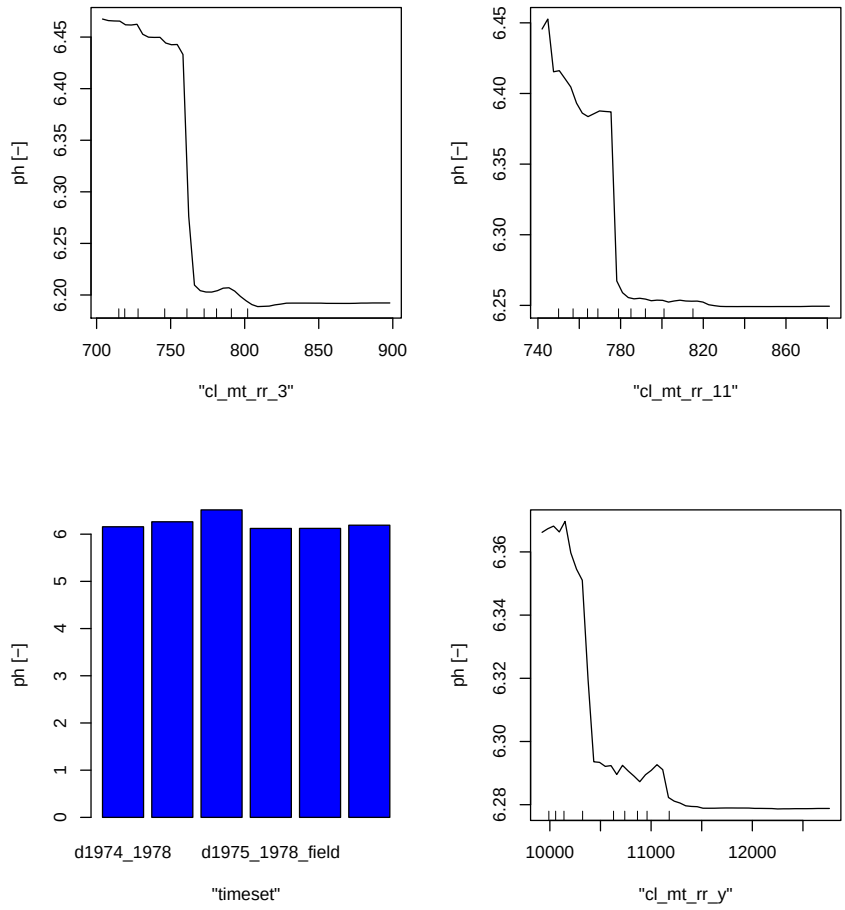

Figure 6: Partial dependence plots for the 4 most important covariates (cl_mt_rr_xx: monthly or yearly rainfall in mm*10).

4 Advanced tasks

If you have already finished the above, please choose a task among the following according to your interest:

Model averaging

So far we calibrated several models to predict topsoil pH. With model averaging we can combine these predictions computing a simple `mean`. Besides simple averaging, we could use weights like $\frac{1}{MSE}$ (make sure they sum up to 1).

Compute validation statistics (e.g. root mean squared error, R^2) on the validation set for the predictions of each model and the (weighted) averaged predictions.

Which model would you chose? Is model averaging a good strategy for the topsoil pH of the `berne` dataset?

Note: *Be aware not to select the final model based on the validation data!* If you start tuning your predictions on your validation data, you loose the independent estimate of prediction accuracy.

Better choose the best method for the final predictions based on cross validation errors (e.g. computed on the same cross validation subsets) or OOB errors.

Work with your own data

In case you already have a digital soil mapping project you are working on, you can apply the lasso, random forest including model selection or plotting for interpretation to this data set. In case you encounter problems related to your dataset, ask!

Fit lasso to binary data

Select the lasso for a binary response (e.g. presence/absence of waterlogging `waterlog.100`). Use `family = "binomial"` in `cv.grpreg` and make sure your response is coded as 0/1.

Better understand R coding

- Make sure you understand the `lapply` and `sapply` sections used in this document. Construct a `for` loop to replace the `sapply` and `lapply` functions used in this training.
- Is there a way you can replace the `for` loop in the recursive backward elimination? (Hint: check for the Fibonacci series with `apply`).

Optimize random forest before covariate slection

Optimize m_{try} before you start the covariate selection (function `train`, package `caret`). How much does the OOB error decrease? Are both steps (tuning, selection) worth the effort from a point of view of prediction performance?

Gradient boosting with trees

The gradient boosting algorithm can be used with trees as baselearners. From the trees variable importance can be derived as in random forest.

- Fit a gradient boosting model with trees (package `gbm` or `caret`) to the topsoil pH data.
- Create partial dependence plots for the boosted trees model (`?plot.gbm`, `plot(.., i.var = ..)`). Do you find the same relationships?

Automated documentation of R analysis

This document was generated with `KnitR`. This is a great way to document analysis done with R. It combines R with LaTeXand collects your output directly in a PDF. It is also possible to create Word documents using R markdown instead of LaTeX.

Checkout the `.Rnw`-file in the zip and try to understand which part built what in this PDF.

Try to create a basic example of a `KnitR` file e.g. with RStudio.

Knitr tutorial: http://kbroman.org/knitr_knutshell/

R session information

R session information (make transparent which R version was used and which packages were loaded):

```
toLatex(sessionInfo(), locale = FALSE)
```

- R version 3.5.0 (2018-04-23), `x86_64-pc-linux-gnu`

- Running under: `Progress Linux 4+ (dschinn-backports)`
- Matrix products: default
- BLAS: `/usr/lib/libblas/libblas.so.3.7.0`
- LAPACK: `/usr/lib/lapack/liblapack.so.3.7.0`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: foreach 1.4.3, geoGAM 0.1-2, glmnet 2.0-16, grpreg 3.1-3, knitr 1.20, Matrix 1.2-14, randomForest 4.6-14
- Loaded via a namespace (and not attached): codetools 0.2-15, coin 1.2-2, compiler 3.5.0, evaluate 0.10.1, grid 3.5.0, highr 0.6, iterators 1.0.8, lattice 0.20-35, magrittr 1.5, MASS 7.3-50, mboost 2.8-1, mgcv 1.8-23, modeltools 0.2-21, multcomp 1.4-8, mvtnorm 1.0-7, nlme 3.1-137, nnls 1.4, parallel 3.5.0, party 1.3-0, quadprog 1.5-5, sandwich 2.4-0, splines 3.5.0, stabs 0.6-3, stats4 3.5.0, stringi 1.2.2, stringr 1.2.0, strucchange 1.5-1, survival 2.42-3, TH.data 1.0-8, tools 3.5.0, zoo 1.8-1