coinflibs: Conditional Inference after Likelihood-based Selection

David Rügamer

The package contains functions to calculate limits and conduct inference in a selective manner for linear models after likelihood- or test-based model selection.

Example: Combining AIC search and significance hunting

```
# install and load package
library("devtools")
install_github("davidruegamer/coinflibs")
library("coinflibs")

library(MASS)
# use the cpus data
data("cpus")
# Fit initial model
cpus$perf <- log10(cpus$perf)
cpus$cach <- as.factor(cpus$cach)
mod <- lm(perf ~ .-name, data = cpus)

# use the stepAIC function to find the best model in a backward
# stepwise search
cpus.lm <- stepAIC(mod, trace = FALSE, direction = "backward", steps = 3)
# check model selection
cpus.lm$anova$Step

# recalculate all visited models in the first step
allvisited <- lapply(attr(mod$terms, "term.labels"),
                     function(x) update(mod, as.formula(paste0("perf ~ .-", x))))
# combine the visited models and the final model
lom1 <- c(allvisited, list(mod))

# perform F-test at level
alpha = 0.001
# and check for non-significant variables
coefTable <- anova(cpus.lm)
drop <- rownames(coefTable)[alpha < coefTable[-nrow(coefTable),5]]

# drop non-significant variable
cpus.lm2 <- update(cpus.lm, as.formula(paste0(".~.-",drop)))

# create list of models, which are examined during the significance search
lom2 <- list(cpus.lm, cpus.lm2)
```

```
# now compute selective inference, which adjust for the AIC-based search as
# well as significance hunting
selinf( # supply all lists of visited models, where the best model in the
    # first list is interpreted as the final model
    lom2, # list given by signficance hunting
    lom1, # list given by AIC-based search
    response = cpus$perf,
    what = c("Ftest", "aic"), # specify what type of selection was done
    # for each supplied list
    sd = summary(cpus.lm2)$sigma
)
```

Example: Combining models visited during stepwise AIC search

```
# install and load package
library("devtools")
install_github("davidruegamer/coinflibs")
library("coinflibs")

library(MASS)
# use the cpus data
data("cpus")
# Fit initial model
cpus$perf <- log10(cpus$perf)
cpus$cach <- as.factor(cpus$cach)
cpus$name <- NULL
currentmod <- lm(perf ~ 1, data = cpus)

# make a stepwise AIC-based forward search
# for all variables in the pool of possible covariates
varsInPool <- colnames(cpus)[-7]

# since the stepAIC function does not provide the models
# fitted in each step, we have to do the search 'manually'
improvement <- TRUE
listOfModelComps <- list()

# do the forward stepwise AIC search...
while(improvement & length(varsInPool)>0){

    # compute all other models
    allOtherMods <- lapply(varsInPool, function(thisvar)
      update(currentmod,
             as.formula(paste0(". ~ . + ",
                               thisvar))))

    # store all models that were examined in this step
    listOfModels <- append(allOtherMods, list(currentmod))
    # save this list for later
    listOfModelComps <- append(listOfModelComps, list(listOfModels))

    # check the AIC of all models
```

```
    aics <- sapply(listOfModels, AIC)
    # what is the best model?
    (wmaic <- which.min(aics))
    # is there any improvement?
    if(wmaic == length(listOfModels)) improvement <- FALSE
    # redefine the current (best) model
    currentmod <- listOfModels[[wmaic]]
    # and update the variables available
    varsInPool <- varsInPool[-wmaic]

}

# variables left, which did not improve the model
varsInPool
# the final model call
currentmod$call

# get the test vector from the current model
vTs <- extract_testvec(limo = currentmod)

# extract list of model components in each step when comparisons
# are done based on the AIC
listOfComps <- lapply(listOfModelComps, function(lom)
    extract_components(listOfModels = lom, response = cpus$perf, what = "aic"))

# calculate the truncation limits for each of the comparisons in each iteration
listOfLimits <- lapply(listOfComps, function(lom)
    calculate_limits(comps = lom, vTs = vTs))

# now compute selective inference, which adjust for the forward stepwise AIC search
# by supplying the lists of limits
calculate_selinf(limitObject = listOfLimits,
                 y = cpus$perf,
                 sd = sigma(currentmod))

############################################################
# now do that with the function provided in the package
############################################################

currentmod <- lm(perf ~ 1, data = cpus)

res <- forwardAIC_adjustedInference(yname = "perf",
                                    data = cpus,
                                    mod = currentmod,
                                    var = NULL)

res$inf
```

References

Rügamer, D., Greven, S. Selective inference after likelihood- or test-based model selection in linear models, Statistics & Probability Letters 140, 7-12 (2018). https://doi.org/10.1016/j.spl.2018.04.010.

Rügamer, D., Greven, S. Inference for L_2-Boosting. Stat Comput 30, 279–289 (2020). https://doi.org/10.1007/s11222-019-09882-0.

Rügamer, D., Baumann, P., Greven, S. Selective inference for additive and linear mixed models, Computational Statistics & Data Analysis 167, 107350 (2022). https://doi.org/10.1016/j.csda.2021.107350.