PCRedux package - Case Study

The PCRedux package authors

2020-10-01

Case study for the application of the PCRedux package

In this case study we used 400 amplification curves from the *kbqPCR* dataset from and assign them to the object *curves*. A comprehensive PDF version (including domain knowledge about qPCRs and machine learning) of this document is available **online supplement**. This online document contains additional code (incl. installation of the software, data import, preprocessing)) that helps to understand how this introductionary case study was created. Before we start we need to load additional libraries (some from CRAN, some from github.com).

```
library(qpcR)
library(PCRedux)

library(dplyr)
library(forcats)
library(reshape2)

library(mlr)

library(ggplot2)

if("devtools" %in% rownames(installed.packages()) == FALSE) {
  library(devtools)
}

if("gbm" %in% rownames(installed.packages()) == FALSE) {
  install.packages("gbm")
}

if("patchwork" %in% rownames(installed.packages())) {
  library(patchwork)
} else {
  devtools::install_github("thomasp85/patchwork")
  "patchwork" %in% rownames(installed.packages())
  library(patchwork)
}
```

For the example qPCR data from a real experiment with a CFX96 (Bio-Rad) were used. It is an allelic specific PCR with TaqMan probes (FAM, HEX, ROX, Cy5, Cy5-5) (experimental details are described in Romaniuk et al. (2019). Original pcrd and rdml files are here: https://github.com/kablag/AScall/tree/master/examples

```
# Determine the Number of amplification curves
curves <- ncol(kbqPCR)

paste("Number of amplification curves:", curves - 1)
```

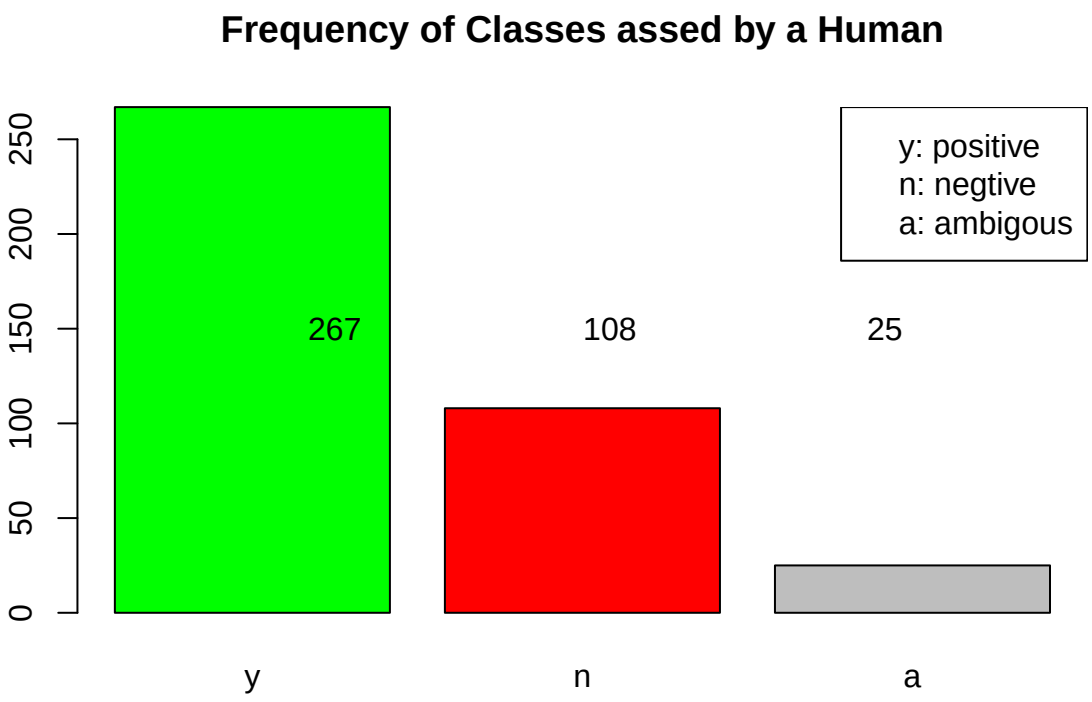
```
## [1] "Number of amplification curves: 400"
```

The decision of the classes (negative, ambiguous, positive) were taken from the `decision_res_kbqPCR.rda` of the *PCRedux* package.

```
dec <- unlist(lapply(1L:(curves -1), function(i) {
  decision_modus(decision_res_kbqPCR[i, 2:8])
}))

class <- c(y = sum(dec == "y"), a = sum(dec == "a"), n = sum(dec == "n"))

# Plot the classes
barplot(table(dec), col = c("green", "red", "grey"),
        main = "Frequency of Classes assed by a Human")
legend("topright", c("y: positive", "n: negtive", "a: ambigous"))
text(c(1:3), rep(150,3), class[c(1,3,2)])
```

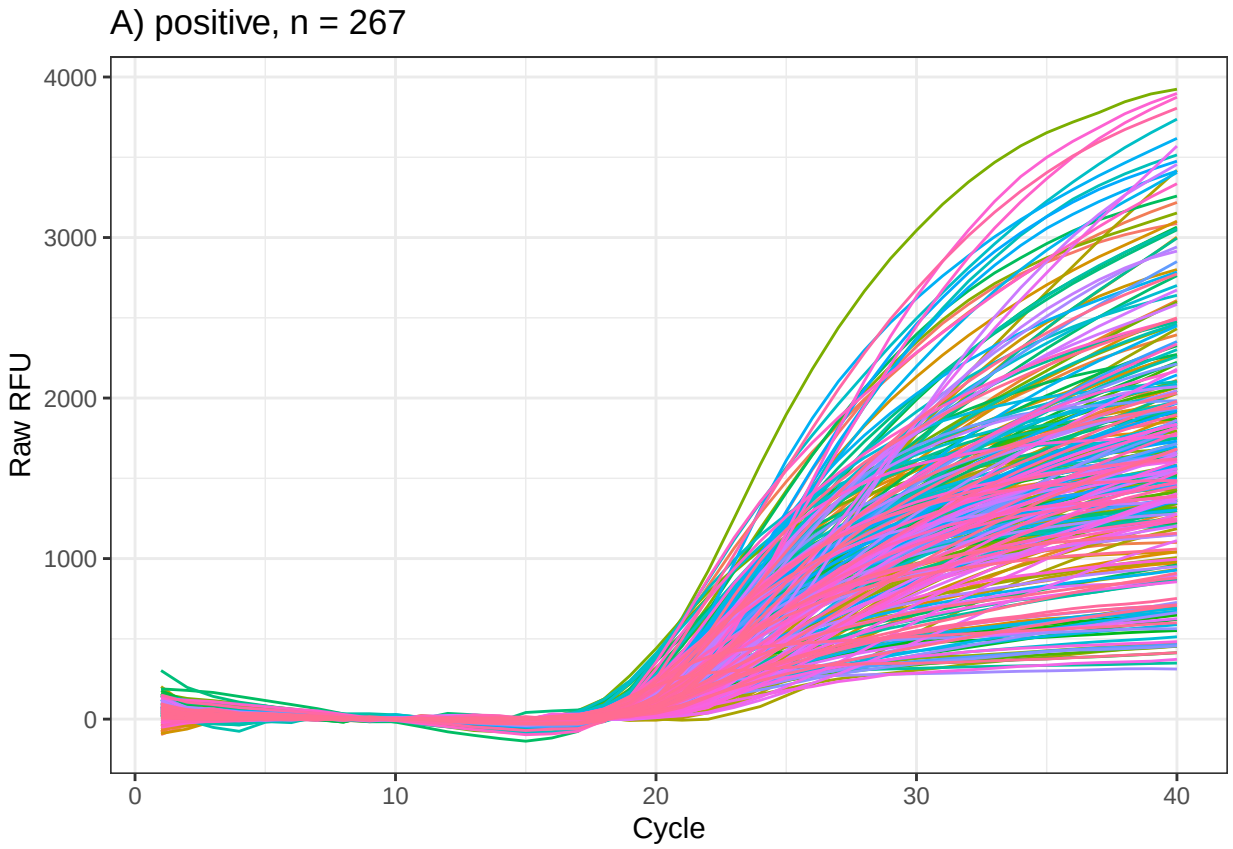


Visualizng qPCR curves separated as positive, negative and ambigous Amplification Curves

As a definition for a positive amplification curve, it is assumed that it has a sigmoid shape and a good signal-to-noise ratio. An ambiguous amplification curve has a signal that is above the noise level but has no sigmoid form. In this example, the ambiguous amplification curves are almost linear and do not show a plateau. Negative amplification curves do not achieve a high signal and do not show a sigmoid curve.

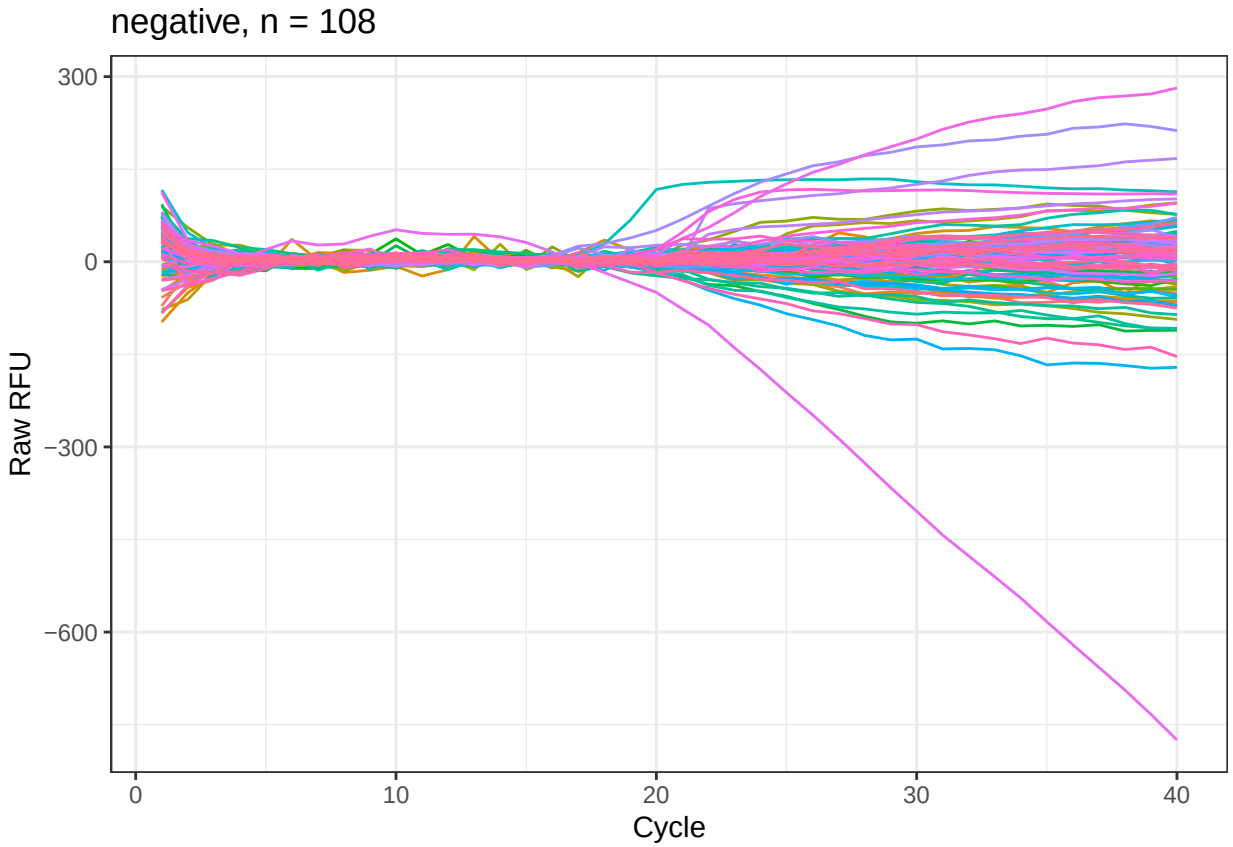
```
p1_pos <- data.frame(kbqPCR[, c(1L, which(dec == "y") + 1)]) %>%
  melt(id.vars = "cyc") %>%
  ggplot(aes(x = cyc, y = value, color = variable)) +
  geom_line() +
  theme_bw() +
  xlab("Cycle") +
  ylab("Raw RFU") +
  ggtitle(paste0("A) positive, ", "n = ", class[1])) + #qPCR curves
  theme(legend.position = "none")

# Positive Curves
p1_pos
```



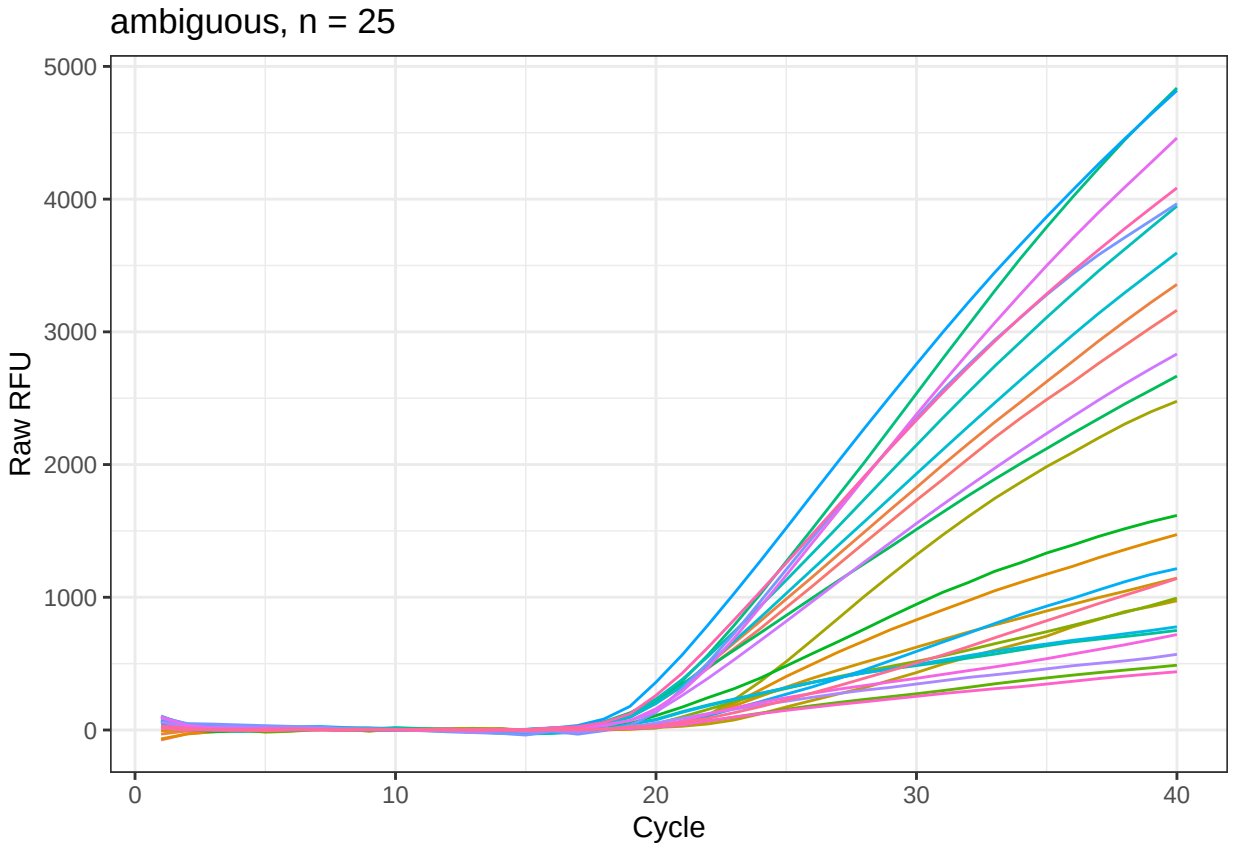

```
p1_neg <- data.frame(kbqPCR[, c(1L, which(dec == "n") + 1)]) %>%
  melt(id.vars = "cyc") %>%
  ggplot(aes(x = cyc, y = value, color = variable)) +
  geom_line() +
  theme_bw() +
  xlab("Cycle") +
  ylab("Raw RFU") +
  ggtitle(paste0("negative, ", "n = ", class[3])) + #qPCR curves
  theme(legend.position = "none")

# Negative Curves
p1_neg
```




```
p1_amb <- data.frame(kbqPCR[, c(1L, which(dec == "a") + 1)]) %>%
  melt(id.vars = "cyc") %>%
  ggplot(aes(x = cyc, y = value, color = variable)) +
  geom_line() +
  theme_bw() +
  xlab("Cycle") +
  ylab("Raw RFU") +
  ggtitle(paste0("ambiguous, ", "n = ", class[2])) + #qPCR curves
  theme(legend.position = "none")

# Ambiguous Curves
p1_amb
```



Calculating some parameters with the encu() function

The *encu()* function was used to calculate some parameters that are later used for the machine learning. Please be patient, this step will take some time.

```
res <- encu(kbqPCR[, 1L:curves])


head(res)
```

```
##                   runs  cpD1  cpD2 cpD2_approx cpD2_ratio      eff   sliwin
## 1    A01_p949_unkn_B2m 22.51 19.21    19.48311  0.9859824 0.000000 1.411886
## 2  A01_p949_unkn_HA1_G 24.07 20.27    20.68362  0.9800028 0.000000 1.836779
## 3  A01_p949_unkn_HA2_C 23.11 19.76    19.94307  0.9908204 2.022231 1.876352
## 4 A01_p949_unkn_HER2_C 24.81 18.94    19.27053  0.9828481 2.161240 1.107599
## 5 A01_p949_unkn_UTA2_T  1.00 40.00    28.18102  1.4193952 1.042248 0.000000
## 6    A02_p949_unkn_B2m 21.31 18.36    18.48386  0.9932989 0.000000 1.872527
##   cpDdiff loglin_slope cpD2_range top       f.top tdp       f.tdp bg.stop
## 1    3.30   0.08374959   6.564496  20 0.107835818  30   0.6115121       9
## 2    3.80   0.07573348   8.559779  18 0.008510364  31   0.6380311      11
## 3    3.35   0.07531076   7.122306  18 0.018989921  29   0.4184203      10
## 4    5.87   0.00000000  17.369064  20 0.081338666  26  -0.1060916      11
## 5   39.00   0.00000000  40.000000   0 1.000000000  10 -12.2032249       5
## 6    2.95   0.09382633   6.926792  17 0.016462960  33   0.9474524       9
##   amp.stop    b_slope b_model_param c_model_param d_model_param e_model_param
## 1       35 -6.2672612     -5.574343 -5.160598e-03      1.043315      4.450226
## 2       37 -5.6178620     -5.545142 -3.799486e-05      1.077896     14.926224
```

```
## 3       36 -6.1402785     -4.646686 -1.795466e-03      1.092918      3.386890
## 4       32 -3.2344943     -3.413703 -4.997303e-03      1.274865      1.749620
## 5       40 -0.3314856 -10000.000000 -1.000000e+04  10000.000000  10000.000000
## 6       34 -6.7551477     -7.298111  7.300215e-04      1.017968     17.572983
##   f_model_param  f_intercept convInfo_iteratons qPCRmodel qPCRmodelRF
## 1   9670.332949 11685.782255               374        l6          l7
## 2     17.083710    35.122803                89        l6          l7
## 3   9731.939666 10376.291031               338        l7          l7
## 4  11035.891815  7535.028549               429        l5          l7
## 5 -10000.000000     7.862683               812        l6          l7
## 6      4.920904     8.452846                28        l7          l7
##          minRFU maxRFU         init2         fluo    slope_bg  intercept_bg
## 1  -0.049089650      1  8.113553e-09   0.06249944 0.015426011  -0.065894593
## 2  -0.008059149      1  4.682824e-09   0.05994716 0.002540320  -0.010702667
## 3  -0.011755639      1  5.852458e-08   0.06464790 0.004594483  -0.016999183
## 4  -0.032615086      1  2.050097e-08   0.04477632 0.011242504  -0.039607588
## 5 -12.115585390      1 -2.496381e+00 -13.06626420 2.737521490 -11.499490546
## 6  -0.006769370      1  7.648998e-10   0.06887139 0.003133627  -0.009447139
##      sigma_bg     sd_bg head2tail_ratio mblrr_slope_pt mblrr_intercept_bg
## 1 0.013782560 0.2913815   -0.0053706413    0.016834754      -4.381488e-03
## 2 0.002282872 0.2721438   -0.0004508547    0.021759464       5.887798e-04
## 3 0.003932412 0.2311414    0.0005485522    0.026090767      -1.976695e-03
## 4 0.012991522 0.1336255    0.0054758405    0.037356896       1.796142e-03
## 5 3.383538916 1.0049851    0.0725048152    0.004578025      -1.229363e+01
## 6 0.004105282 0.3901553    0.0024836420    0.003956219      -3.114974e-03
##   mblrr_slope_bg mblrr_cor_bg mblrr_intercept_pt mblrr_cor_pt    polyarea
## 1   0.0003590634            0         0.34502518    0.9836875  0.09319854
## 2  -0.0001321112            0         0.15205901    0.9816597  0.15322259
## 3   0.0002110333            0        -0.02431304    0.9949512  0.15061648
## 4  -0.0006885145            0        -0.46845515    0.9990859  0.17097500
## 5   0.0353965240            0         0.25334499    0.0000000 -7.37441784
## 6   0.0002761692            0         0.84676537    0.9085009  0.06665375
##   peaks_ratio autocorrelation cp_e.agglo cp_bcp amptester_shapiro amptester_lrt
## 1 0.006925488       0.7682340      0.075  0.125             FALSE          TRUE
## 2 0.002466177       0.7759635      0.075  0.125             FALSE          TRUE
## 3 0.004649833       0.7961352      0.075  0.150             FALSE          TRUE
## 4 0.597343671       0.8306873      0.075  0.150             FALSE          TRUE
## 5 1.260531708       0.6296774      0.050  0.100             FALSE          TRUE
## 6 0.000000000       0.7244684      0.050  0.100             FALSE          TRUE
##   amptester_rgt amptester_tht amptester_slt amptester_polygon
## 1         FALSE          TRUE          TRUE          9.062471
## 2         FALSE          TRUE          TRUE          9.868007
## 3         FALSE          TRUE          TRUE          9.913802
## 4          TRUE          TRUE          TRUE         10.750512
## 5         FALSE         FALSE         FALSE         20.475000
## 6          TRUE          TRUE          TRUE          7.843177
##   amptester_slope.ratio hookreg_hook hookreg_hook_slope hookreg_hook_delta
## 1            0.14452245            0          0.0000000                  0
## 2            0.16218446            0          0.0000000                  0
## 3            0.06708085            0          0.0000000                  0
## 4            0.09532211            0          0.0000000                  0
## 5           -0.02626256            1         -0.4387811                 27
## 6            0.13258664            0          0.0000000                  0
##   central_angle number_of_cycles detection_chemistry device
```

```
## 1    -0.9984119              40                  NA     NA
## 2    -0.9992332              40                  NA     NA
## 3    -0.9990252              40                  NA     NA
## 4    -0.9997855              40                  NA     NA
## 5    -0.8480318              40                  NA     NA
## 6    -0.9982053              40                  NA     NA
```

Merging decisions and features into one dataset

Since the parameters from the calculations with the *encu()* function and the decisions are known by now, we can merge them into a single dataframe.

```
dat <- cbind(res, decision = factor(c("ambiguous", "negative", "positive")[dec],
                                    levels = c("positive", "ambiguous", "negative"))) %>%
  select(cpD2, amptester_polygon, cpDdiff, decision) %>%
  filter(!is.na(dec))

head(dat)
```

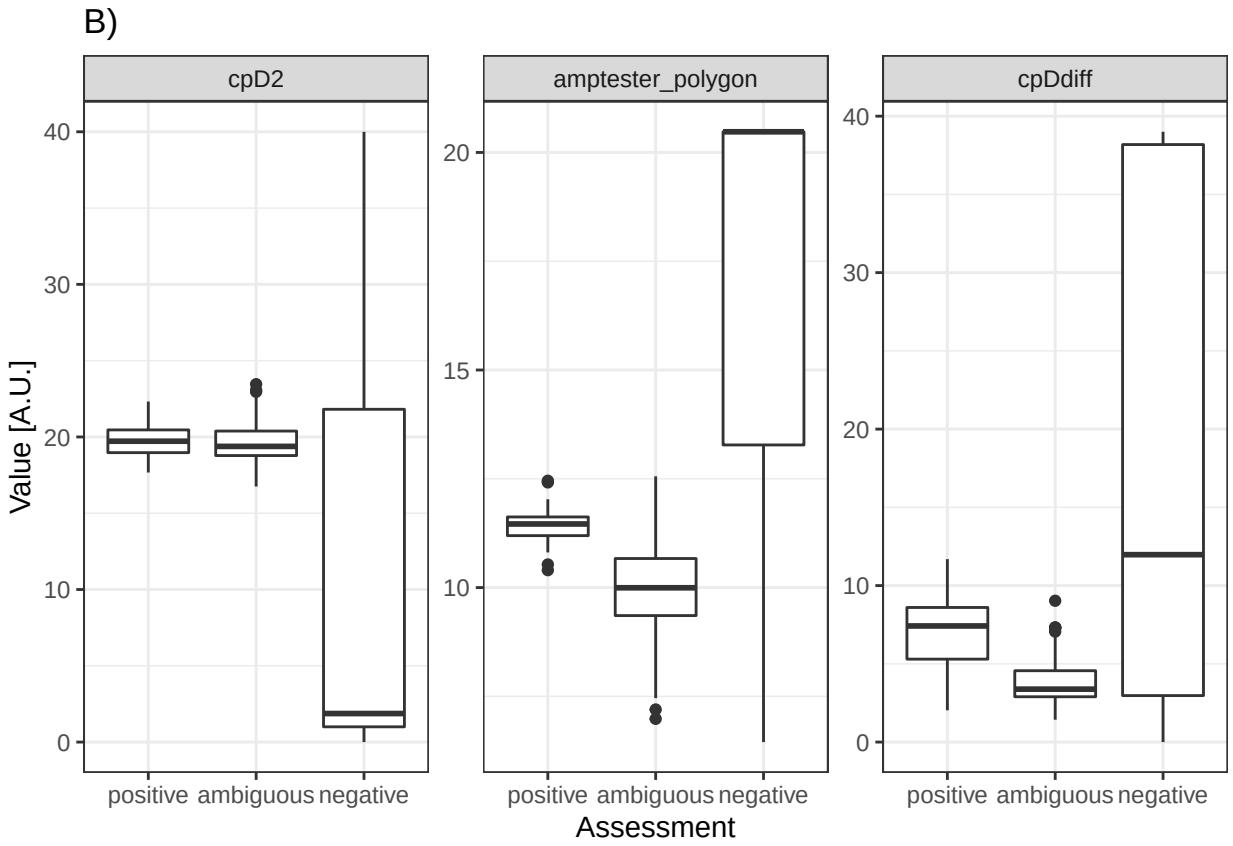
```
##    cpD2 amptester_polygon cpDdiff  decision
## 1 19.21          9.062471    3.30 ambiguous
## 2 20.27          9.868007    3.80 ambiguous
## 3 19.76          9.913802    3.35 ambiguous
## 4 18.94         10.750512    5.87 ambiguous
## 5 40.00         20.475000   39.00  negative
## 6 18.36          7.843177    2.95 ambiguous
```

Visualizing the parameter calculations

Next we visualize the results of the parameter calculations.

```
p2 <- ggplot(data = dat %>%
               mutate(id = rownames(dat)) %>%
               melt(id.vars = c("id", "decision")),
             aes(x = decision, y = value)) +
  geom_boxplot() +
  theme_bw() +
  facet_wrap(~ variable, scales = "free_y") +
  scale_y_continuous("Value [A.U.]") +
  scale_x_discrete("Assessment") +
  ggtitle("B)") # Separation of types of curves by encu() parameters

p2
```



Modeling with different classifiers

Building the models follows the typical steps involving the definition of the splits, classifiers and their tasks. Here we use the *mlr* package. We want to classify only! As classifiers we use:

- Random Forest (classif.ranger),
- Support Vector Machines (classif.ksvm),
- linear discriminant analysis (classif.lda),
- Generalized Boosted Regression Models (classif.gbm),
- Multinomial Regression (classif.multinom) and
- Generalized Linear Regression with Lasso or Elasticnet Regularization (classif.glmnet).

```
tsk <- makeClassifTask("pcr_classif", data = dat, target = "decision")

mdls <- list()
mdls[[1]] <- makeLearner("classif.ranger", predict.type = "prob")
mdls[[2]] <- makeLearner("classif.ksvm", predict.type = "prob")
mdls[[3]] <- makeLearner("classif.lda", predict.type = "prob")
mdls[[4]] <- makeLearner("classif.gbm", predict.type = "prob")
mdls[[5]] <- makeLearner("classif.multinom", predict.type = "prob")
mdls[[6]] <- makeLearner("classif.glmnet", predict.type = "prob")

set.seed(4732)
results <- do.call(rbind, lapply(mdls, function(mdl) {
  res <- resample(mdl, tsk, cv10, measures = list(mmce, multiclass.au1u))
  cbind(model = res[["learner.id"]], res[["measures.test"]])
}))
```

```
## Distribution not specified, assuming multinomial ...
## Distribution not specified, assuming multinomial ...
## Distribution not specified, assuming multinomial ...
## Distribution not specified, assuming multinomial ...
## Distribution not specified, assuming multinomial ...
## Distribution not specified, assuming multinomial ...
## Distribution not specified, assuming multinomial ...
## Distribution not specified, assuming multinomial ...
## Distribution not specified, assuming multinomial ...
## Distribution not specified, assuming multinomial ...
## # weights:  15 (8 variable)
## initial  value 395.500424
## iter  10 value 123.613433
## iter  20 value 114.385660
## iter  30 value 114.321001
## iter  30 value 114.321001
## iter  30 value 114.321001
## final  value 114.321001
## converged
## # weights:  15 (8 variable)
## initial  value 395.500424
## iter  10 value 120.584262
## iter  20 value 111.082477
## final  value 111.080855
## converged
## # weights:  15 (8 variable)
## initial  value 395.500424
## iter  10 value 151.633277
## iter  20 value 121.549732
## iter  30 value 121.192513
## final  value 121.192509
## converged
## # weights:  15 (8 variable)
## initial  value 395.500424
## iter  10 value 126.187887
## iter  20 value 117.526771
## final  value 117.428040
## converged
## # weights:  15 (8 variable)
## initial  value 395.500424
## iter  10 value 150.704661
## iter  20 value 114.404400
## iter  30 value 114.129806
## final  value 114.129803
## converged
## # weights:  15 (8 variable)
## initial  value 395.500424
## iter  10 value 136.411088
## iter  20 value 122.556785
## final  value 122.326389
## converged
## # weights:  15 (8 variable)
## initial  value 395.500424
## iter  10 value 127.347729
```

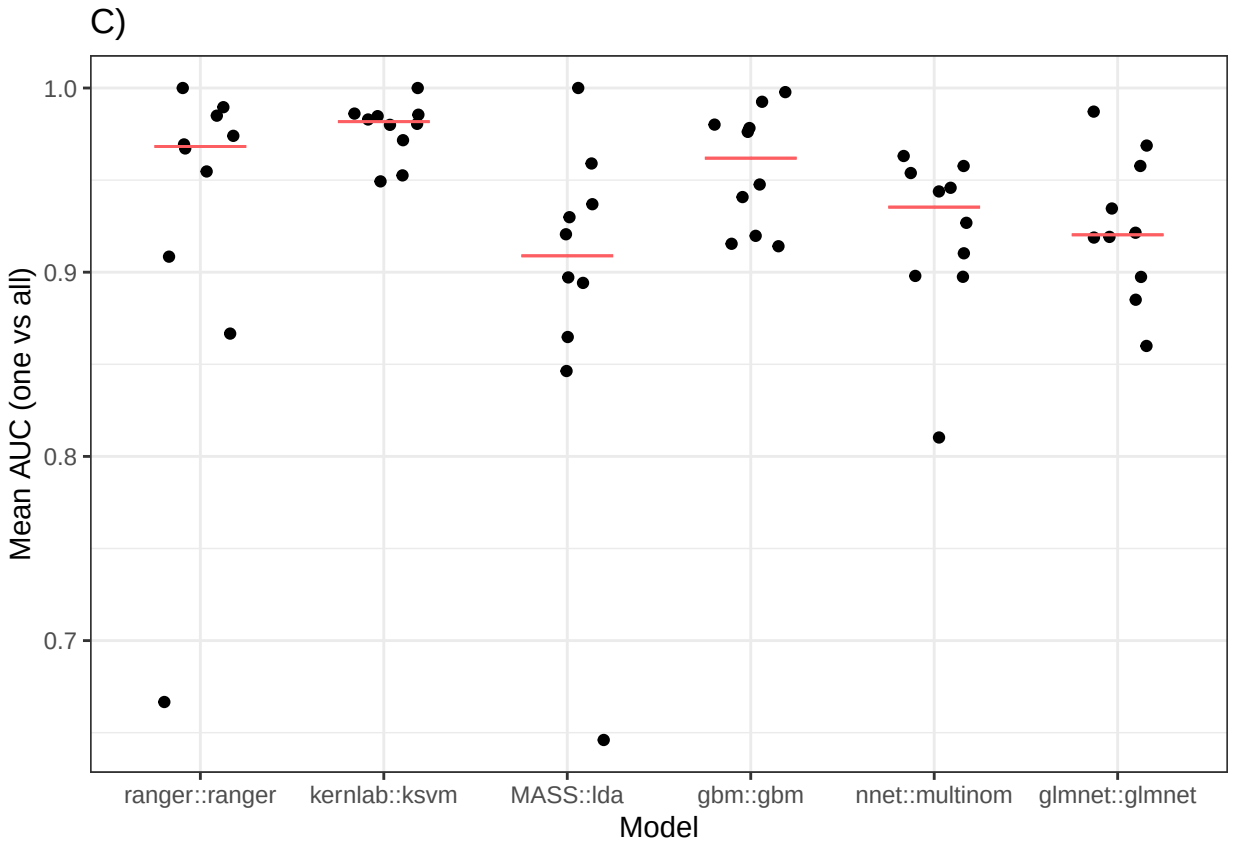
```
## iter  20 value 121.576311
## final  value 121.569286
## converged
## # weights:  15 (8 variable)
## initial  value 395.500424
## iter  10 value 134.178967
## iter  20 value 116.814489
## iter  30 value 116.448420
## iter  30 value 116.448420
## iter  30 value 116.448420
## final  value 116.448420
## converged
## # weights:  15 (8 variable)
## initial  value 395.500424
## iter  10 value 139.303418
## iter  20 value 124.071741
## final  value 123.983227
## converged
## # weights:  15 (8 variable)
## initial  value 395.500424
## iter  10 value 125.254120
## iter  20 value 115.861306
## iter  30 value 115.855795
## iter  30 value 115.855795
## iter  30 value 115.855795
## final  value 115.855795
## converged
```

```
results[["model"]] <- fct_recode(results[["model"]],
                                 `ranger::ranger` = "classif.ranger",
                                 `kernlab::ksvm` = "classif.ksvm",
                                 `MASS::lda` = "classif.lda",
                                 `gbm::gbm` = "classif.gbm",
                                 `nnet::multinom` = "classif.multinom",
                                 `glmnet::glmnet` = "classif.glmnet")
```

Visualizing the model results

```
p3 <- ggplot(data = results, aes(x = model, y = multiclass.au1u)) +
  geom_point(position = position_jitter(width = 0.2, seed = 4)) +
  geom_errorbar(data = results %>%
                  group_by(model) %>%
                  summarise(auc = median(multiclass.au1u)),
                aes(x = model, ymin = auc, ymax = auc),
                inherit.aes = FALSE, color = "#FC5E61",
                width = 0.5) +
  theme_bw() +
  xlab("Model") +
  ylab("Mean AUC (one vs all)") +
  ggtitle("C)") # Results of crossvalidating models trained on encu() parameters

p3
```



Final plot

Finally we plot all findings in a summary graphic.

```
cairo_ps("figure1.eps", width = 11, height = 3.1)
p1_pos + {
    p1_neg +
      p1_amb +
      plot_layout(ncol = 1)
  } + p2 + plot_layout(ncol = 3, widths = c(1, 1, 4))
dev.off()
```

```
## pdf
##   2
```

References

Romaniuk, Dmitrii S., Anna M. Postovskaya, Alexandra A. Khmelevskaya, Dmitry B. Malko, and Grigory A. Efimov. 2019. "Rapid Multiplex Genotyping of 20 HLA-A*02:01 Restricted Minor Histocompatibility Antigens." *Frontiers in Immunology* 10. https://doi.org/10.3389/fimmu.2019.01226.