Machine learning improves error rates in quality control of mass spectrometry-based proteomics

Eralp DOGU

May 18, 2018

Contents

1 Introduction **1**

2 Methods **2**

3 Preliminary data **2**

4 Novel aspect **2**

5 Installing required packages **2**

6 Load the Data **2**

7 Simulate data: One peptide (LVNELTEFAK) four metrics **2**

8 Preprocess the data: Normalize the data **3**

9 Building Random Forest with a slinding baseline **4**

10 Compute threshold **5**

11 Create a control chart **5**

12 Model intrepretation: partial dependence plots and maps **7**

1 Introduction

Advances in mass spectrometry-based proteomics allow the quantitative profiling of an increasingly large number of biological samples and analytes. This introduces new challenges for fully automated, longitudinal monitoring of these experiments for system suitability and quality control (QC). On one hand, automated systems must detect different types and patterns of deviations from optimal performance. On the other hand, they must integrate these patterns across multiple metrics (such as intensity, retention time etc.) of multiple analytes. The integration of univariate per-metric and per-analyte statistical quality assessment methods into QC systems is now common across proteomics laboratories. However, the increase in the number of samples, metrics and analytes limits the performance of these univariate statistical summaries, complicates the interpretation, and inflates the rate of false alarms (1). This work presents an alternative machine learning-based monitoring approach to monitor longitudinal quality control performance.

2 Methods

A state-of-the art approach for longitudinal profiling for QC is Statistical Process Control (SPC). SPC is a collection of statistical methods and graphical summaries that detect deviations from normal operating conditions. Typically, multivariate control charts such as Hotelling's , multivariate cumulative sum (MCUSUM) and multivariate exponentially weighted moving averages (MEWMA) are used for QC based on multi-dimensional measurements (2). However, the application of these methods to mass spectrometry-based proteomics is challenged by factors such as the complexity of data structures, missing values, and complex relationships between the quantified analytes. Recently, an approach called real time contrasts (RTC) control chart was proposed to overcome these difficulties (3, 4). We generalize this approach to targeted and untargeted proteomics workflows such as data dependent acquisition (DDA), selected reaction monitoring (SRM) and data independent acquisition (DIA), by transforming the longitudinal profiling problem into a series of classification problems. RTC control chart assigns one class label to QC data obtained under optimal performance conditions (guide set) and another class label to a window of real-time QC data. Next, class probability estimates and error rates are monitored and compared to control limits that are computed from bootstrapped guide set to detect suboptimal performance.

The method is implemented in an open-source R-based software package MSstatsQC and its web-based graphical user interface, and is available for use stand-alone, or for integration with automated pipelines such as QCloud (5), a cloud-based automated QC system. Thus, it can support proteomics laboratories in daily quality assessment and unbiased instrument evaluation.

3 Preliminary data

The performance of the method is tested on QC1 datasets from QCloud system for both data dependent acquisition (DDA) and selected reaction monitoring (SRM). QCloud supports bovine serum albumin as QC1 samples tested in Thermo instruments (LTQ-Orbitrap Velos Pro and LTQ-q-Orbitrap Fusion Lumos) corresponding to a low complexity quality control sample that is analyzed several times per day in order to establish the performance of the instrument before and after running each real sample. Year-long datasets are used to showcase the importance and potential contribution of advanced multivariate SPC methods for improving error rates in QC analysis. Evaluations on datasets from the QCloud system demonstrated that the proposed method substantially improved our ability of real time monitoring, early detection and prevention of chromatographic and instrumental problems with mass spectrometric assays.

4 Novel aspect

We demonstrated the importance of modern multivariate monitoring for reproducible and reliable results and integrated the method to the context of monitoring longitudinal QC performance of mass spectrometry-based proteomics experiments.

5 Installing required packages

6 Load the Data

7 Simulate data: One peptide (LVNELTEFAK) four metrics

Simulation steps: (a) Extract mean vector and covariance matrix from the training set which is all GO observations for four metrics-retention time, total area, mass accuracy, FWHM (b) Generate n in-control observations (c) Introduce a step change (here 3sigma increase in retention time and 1.5sigma increase in peak area) and generate m out-of-control observations (d) Merge in-control and out-of-control datasets and create DATA with labels

```
#generate multivariate normal data
#parameters from a training sample
n<-100 #incontrol observations
m<-100 #ooc observations
Data<-c()
Data1<-c()
S0<-c()
mean <-c(with(data=Train,tapply(RT,INDEX=PepSeq,FUN=mean))[4],
         with(data=Train,tapply(TotalArea,INDEX=PepSeq,FUN=mean)) [4],
         with(data=Train,tapply(MassAccu,INDEX=PepSeq,FUN=mean)) [4],
         with(data=Train,tapply(FWHM,INDEX=PepSeq,FUN=mean)) [4]
         )
covar<-cov(Train[Train$PepSeq=="LVNELTEFAK",c(3,6,7,8)])
S0<-data.frame(idfile=1:n,PepSeq=rep("LVNELTEFAK",n),mvrnorm(n, mean, covar))
colnames(S0)<-c("idfile","PepSeq","RT","TotalArea","MassAccu","FWHM")
S0 <- reshape(S0, idvar = "idfile", timevar = "PepSeq", direction = "wide")
RESPONSE<-c("GO")
S0 <- cbind(S0,RESPONSE)

Data1<-data.frame(idfile=(n+1):(n+m),PepSeq=rep("LVNELTEFAK",m),
                  mvrnorm(m,
                  mean+(sqrt(3*c(covar[1,1],1.0*covar[2,2],0.0*covar[3,3],0.0*covar[4,4]))),
                  covar))
colnames(Data1)<-c("idfile","PepSeq","RT","TotalArea","MassAccu","FWHM")
Data1 <- reshape(Data1, idvar = "idfile", timevar = "PepSeq", direction = "wide")
RESPONSE<-c("NOGO")
Data <- cbind(Data1,RESPONSE)
```

8 Preprocess the data: Normalize the data

Caret package has an automated preprocess argument that can handle centering, scaling, transforming etc.

```
#preProcess = c("center", "scale", "nzv")
###
###
# new_data <- rbind(S0,Data)
# maxs <- apply(new_data %>% select(-c(idfile,RESPONSE)), 2, max)
# mins <- apply(new_data %>% select(-c(idfile,RESPONSE)), 2, min)
#
# scaled_data <- as.data.frame(scale(new_data %>% select(-c(idfile,RESPONSE)),
# center = mins, scale = maxs - mins))
# #scaled_data$RESPONSE <- ifelse(new_data$RESPONSE =="GO",1,0)
# scaled_data$RESPONSE <- as.factor(new_data$RESPONSE)
# #scaled_data$RESPONSE <- as.factor(scaled_data$RESPONSE)
# scaled_data$idfile <- new_data$idfile
#
# #select random ind for train and test
# set.seed(123)
```

```r
#
# ## 75% of the sample size
# smp_size <- floor(0.75 * nrow(scaled_data))
#
# ## set the seed to make your partition reproducible
# set.seed(123)
# train_ind <- sample(seq_len(nrow(scaled_data)), size = smp_size)
#
#
# train <- scaled_data[train_ind,]
# test <- scaled_data[-train_ind,]
```

9 Building Random Forest with a slinding baseline

(a) Sliding baseline classifies and compares each new observation with the in-control class
(b) Then predicts class probabilities from out-of-bag observations
(c) p1 is the weighted class probability of the ith observation for a sliding baseline
(d) p1 exceeds a given threshold, we can conclude there actually is a change

```r
N0<-length(S0[,1])
Nw<-10 #Sliding baseline size
p0<-c()
p1<-c()
imp<-list()
SW<-c()
Data.model<-c()
Predict<-c()
nTrees<-c()
for (i in Nw:length(Data[,1])) {
  SW<-Data[(i-Nw+1):i,]
  Data.model<-rbind(S0,SW)
  rownames(Data.model)<-1:(N0+Nw)
  Data.model<-Data.model[,-1]
  fitControl <- trainControl(classProbs = T, method = "oob")

  fit <- train(as.factor(RESPONSE) ~ .,
               data = Data.model,
               method="rf",
               preProcess = c("center", "scale", "nzv"),
               trainControl=fitControl,
               tuneGrid = data.frame(mtry = 6))

  Predict<-predict(fit, type='prob', OOB=TRUE)
  nTrees[i]<-fit$finalModel$ntree
  p1[i]<-sum(Predict[(N0+1):(N0+Nw),2])/Nw
  importance<-varImp(fit)$importance
  imp[[i]]<-t(importance)

}
```

10 Compute threshold

(a) Threshold is based on in-control observations and sets a certain false alarm rate
(b) Generate a DATA that has no change
(c) Compute p1
(d) Bootstrap p1 to compute the pth percentile and use it as a threshold (CL)

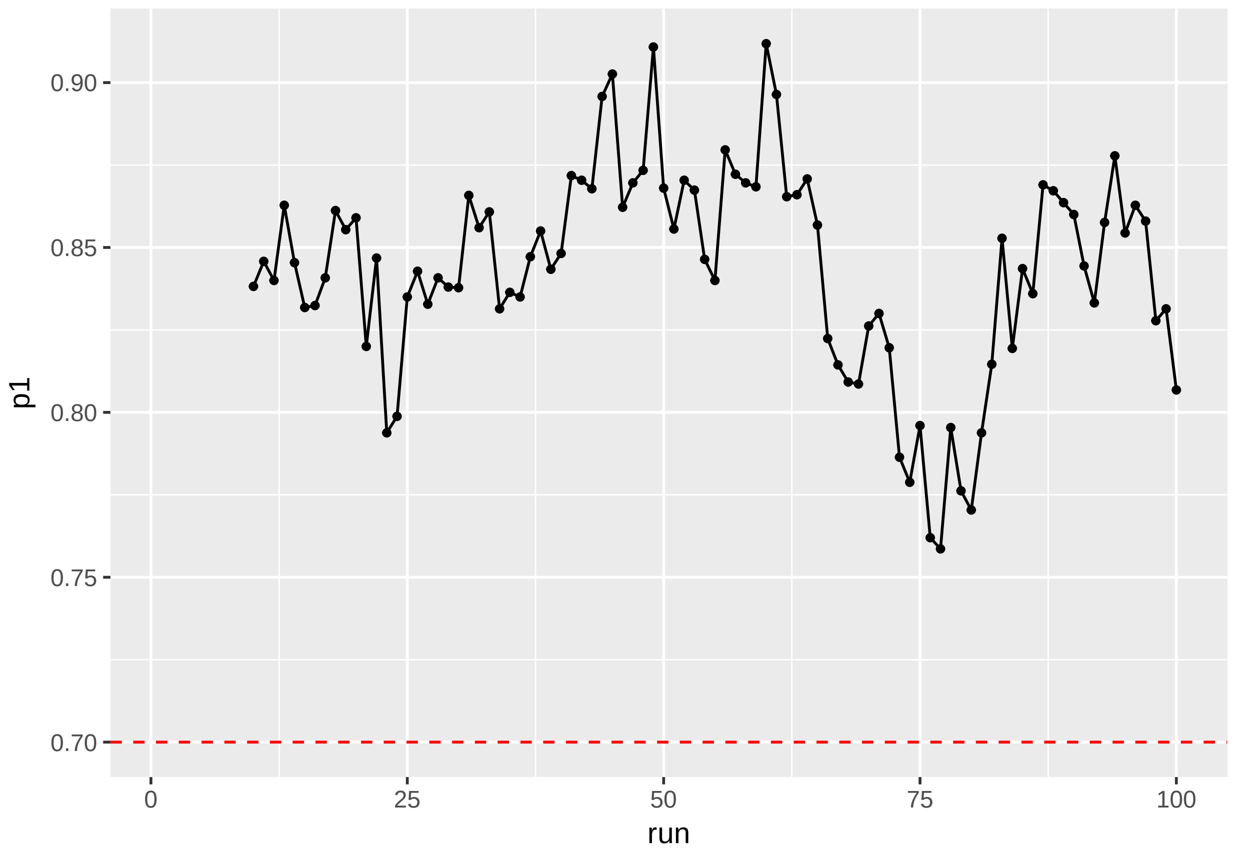
```
n = length(p1)
B = 1000
result = rep(NA, B)
for (i in 1:B) {
  boot.sample = sample(n, replace = TRUE)
  result[i] = quantile(p1[boot.sample],0.9973,na.rm = T)
}
CL<-mean(result)
```

11 Create a control chart

```
chart.stat<-as.data.frame(cbind(p1,CL=0.7,run=1:length(p1)))
plot1<-ggplot(data=chart.stat,aes(y=p1, x=run))+
  geom_line(size=0.5)+
  geom_point(size=1)+
  geom_hline(yintercept=chart.stat$CL, linetype = "dashed", color="red")
plot1
```

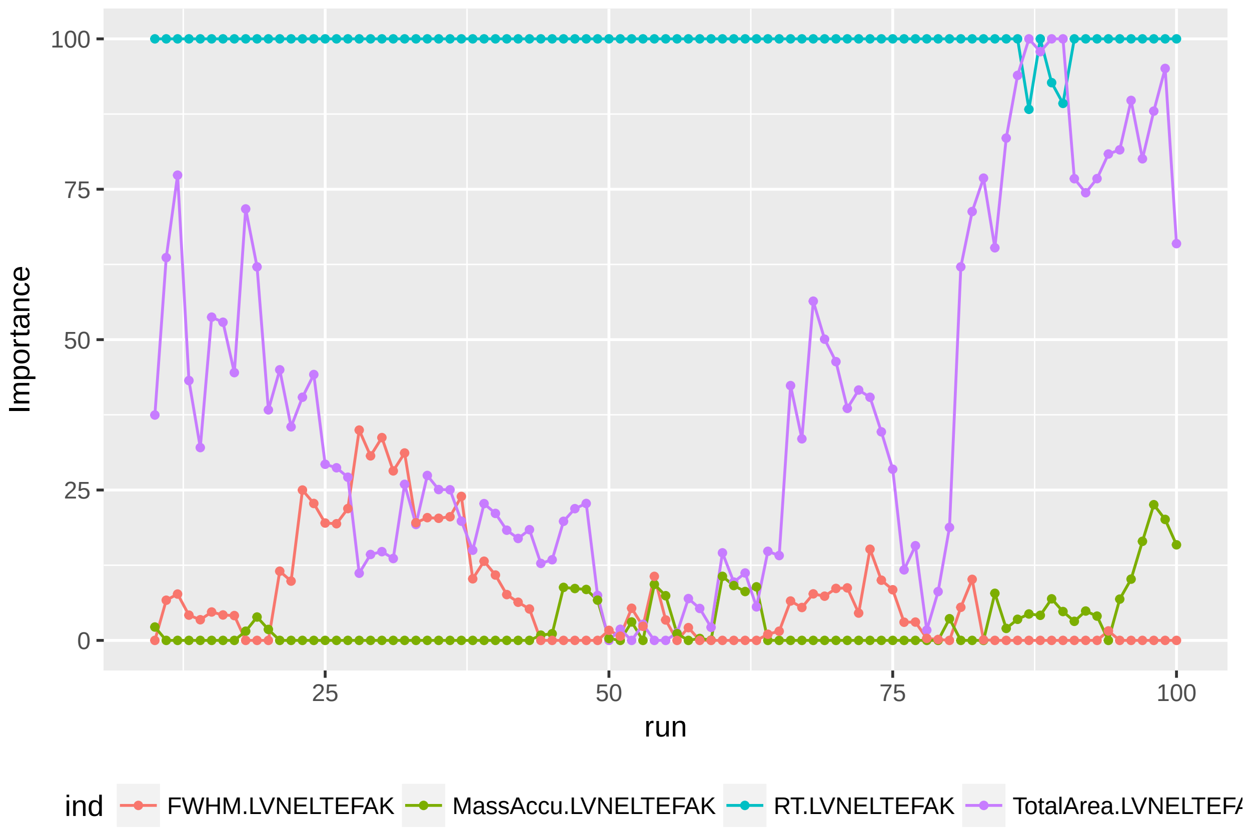
```
## Warning: Removed 9 rows containing missing values (geom_path).
```

```
## Warning: Removed 9 rows containing missing values (geom_point).
```



#Variable importance

```
#Change in variable importance over time
IMP<-as.data.frame(do.call(rbind, imp))
IMP<-stack(data.frame((sapply(IMP,c))))
IMP<-cbind(IMP, run=rep(Nw:length(Data[,1]),4))
plot2<-ggplot(data=IMP,aes(x=run, y=values, color=ind))+
  geom_line(size=0.5)+
  geom_point(size=1)+
  ylab("Importance")+
  xlab("run")+
  scale_fill_continuous(guide = guide_legend()) +
  theme(legend.position="bottom")
plot2
```



```
#cowplot::plot_grid(plot1, plot2, labels = c("RF based control chart", "Importance"), align = "v")
```

12 Model intrepretation: partial dependence plots and maps

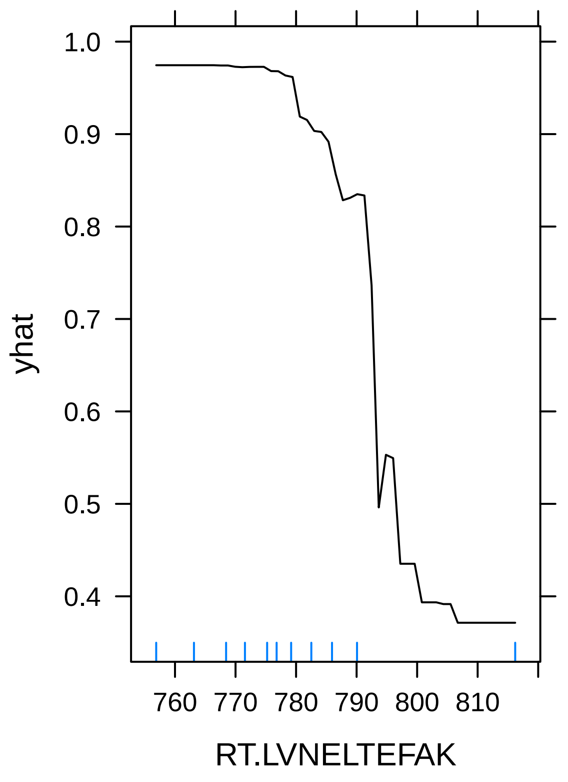
The partial dependence plot shows the marginal effect of a feature on the predicted outcome of a previously fit model. The prediction function is fixed at a few values of the chosen features and averaged over the other features.

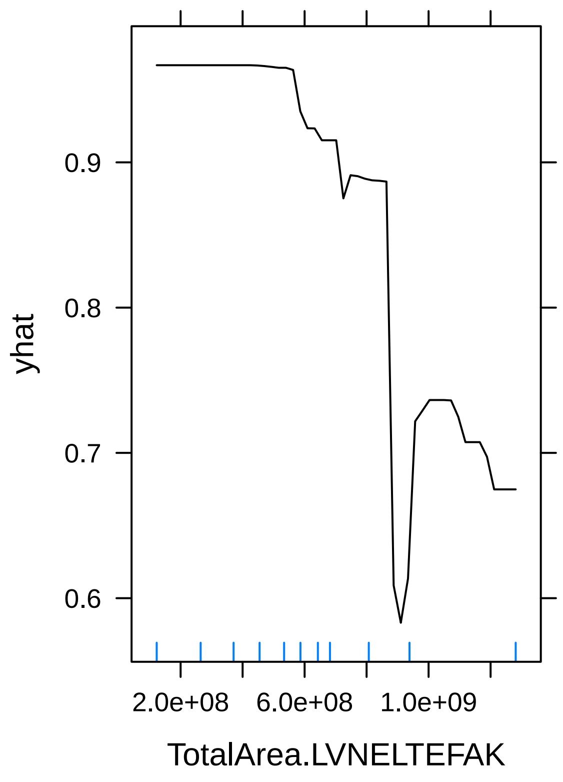
Other names: marginal means, predictive margins, marginal effects. https://christophm.github.io/interpretable-ml-book/agnostic.html

A partial dependence plot can show if the relationship between the target and a feature is linear, monotonic or more complex.

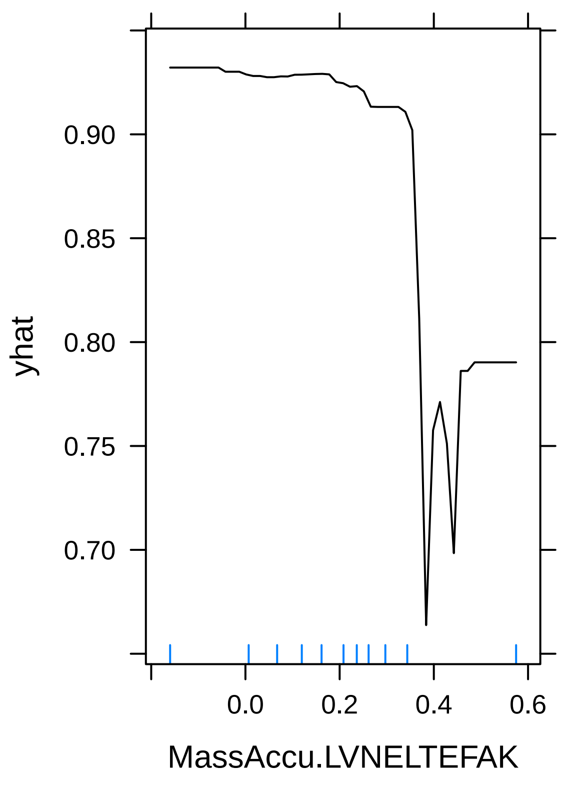
It is not wise to draw conclusions from PDPs in regions outside the area of the training data. Here we describe two ways to mitigate the risk of extrapolation in PDPs: rug plots and convex hulls.

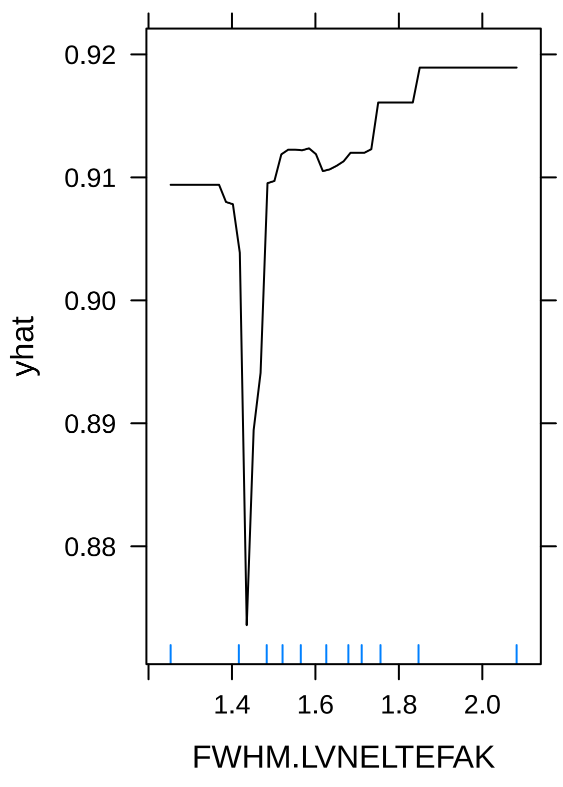
```
pdpTA <- partial(fit, pred.var = "TotalArea.LVNELTEFAK",plot = TRUE, prob=TRUE, rug = TRUE)
pdpRT <- partial(fit, pred.var = "RT.LVNELTEFAK", plot = TRUE, prob=TRUE, rug=TRUE)
pdpMA <- partial(fit, pred.var = "MassAccu.LVNELTEFAK", plot = TRUE, prob=TRUE, rug = TRUE)
pdpFWHM <- partial(fit, pred.var = "FWHM.LVNELTEFAK", plot = TRUE, prob=TRUE, rug = TRUE)
grid.arrange(pdpRT, pdpTA, ncol = 2)
```



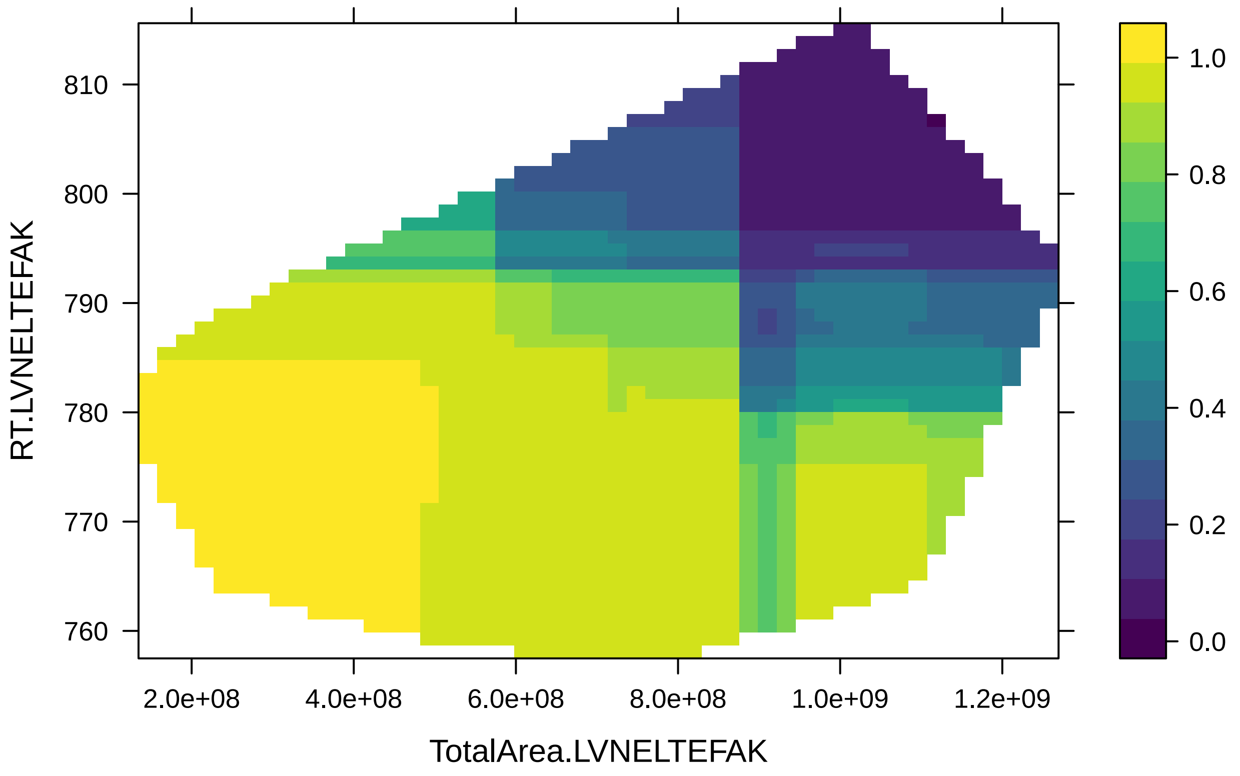


```r
grid.arrange(pdpMA, pdpFWHM, ncol = 2)
```

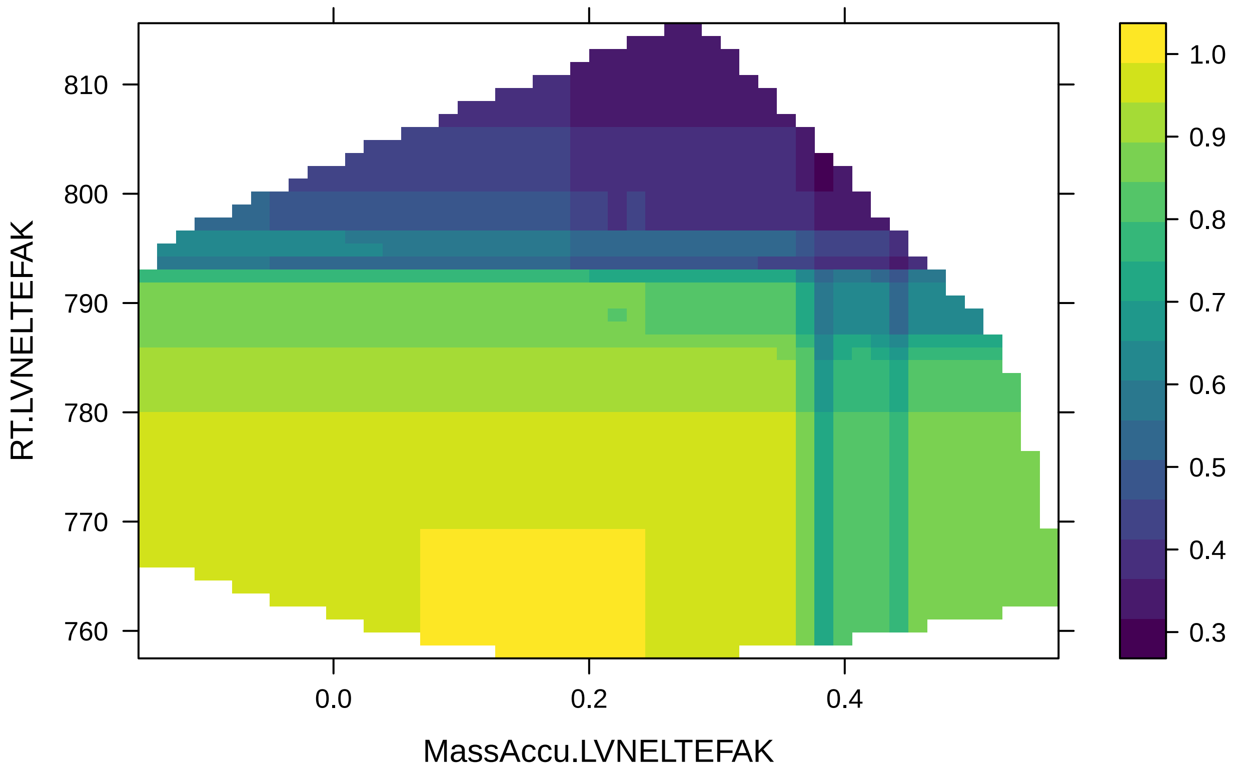




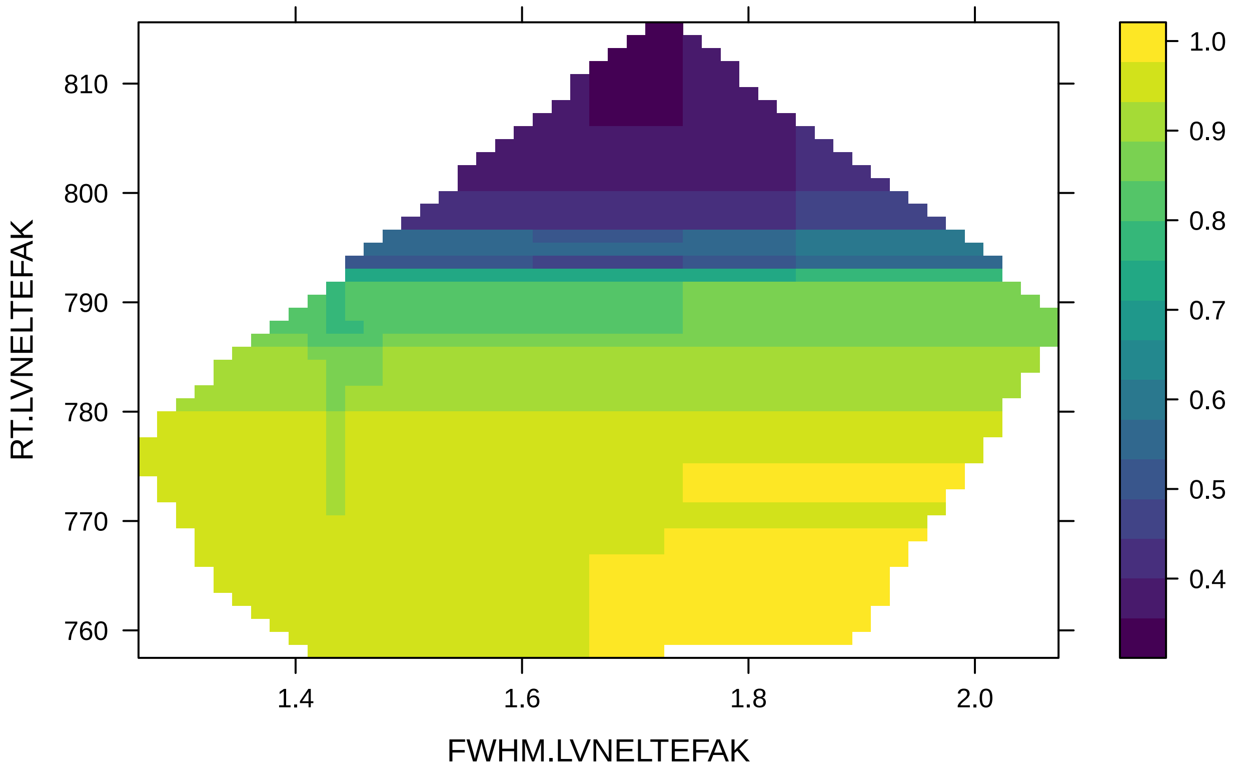
```r
pdp1<- partial(fit, pred.var = c("TotalArea.LVNELTEFAK","RT.LVNELTEFAK"), plot = TRUE,
               prob=TRUE, chull=TRUE)
pdp1
```



```r
pdp2<- partial(fit, pred.var = c("MassAccu.LVNELTEFAK","RT.LVNELTEFAK"), plot = TRUE,
               prob=TRUE, chull=TRUE)
pdp2
```



```
pdp3<- partial(fit, pred.var = c("FWHM.LVNELTEFAK", "RT.LVNELTEFAK"), plot = TRUE,
               prob=TRUE, chull=TRUE)
pdp3
```



```
#grid.arrange(pdp1, pdp2, pdp3,ncol = 3)
```