mlr3中的可视化

概述

我们展示了mlr3生态系统的可视化功能。mlr3viz 包为几乎所有的mlr3对象创建了一个绘图。这篇文章展示了所有可用的绘图及其可重复的代码。我们从基础mlr3对象的绘图开始。这包括任务的boxplots，聚类学习器的dendrograms 和预测的 ROC 曲线。之后，我们对分类树进行调参，并将结果可视化。最后，我们展示了过滤器的可视化。

包

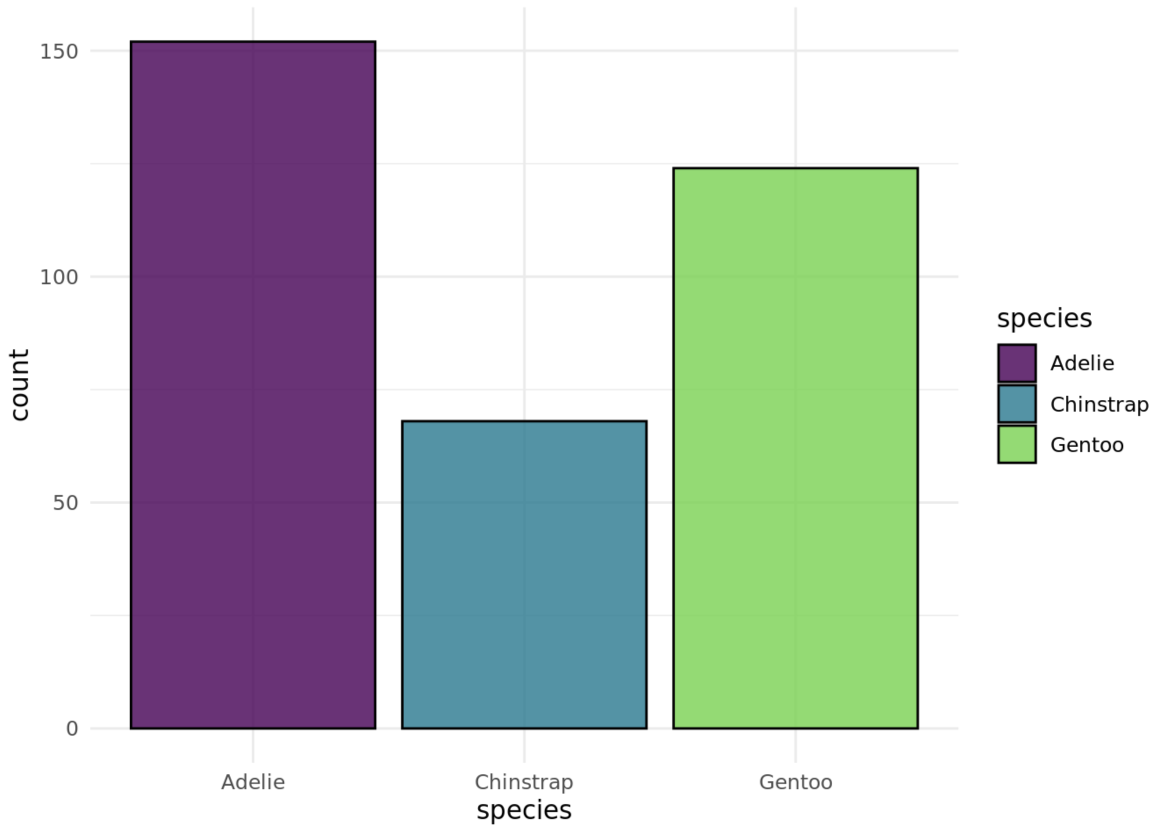
mlr3viz包定义了`autoplot()`函数，用ggplot2画图。通常一个对象有不止一种类型的图。你可以通过`type`参数来改变绘图。帮助页面列出了所有可能的选择。访问帮助页面的最简单方法是通过pkgdown网站。图形使用viridis的调色板，外观由`theme`参数控制。默认情况下，使用的是`minimal theme`。

任务

分类

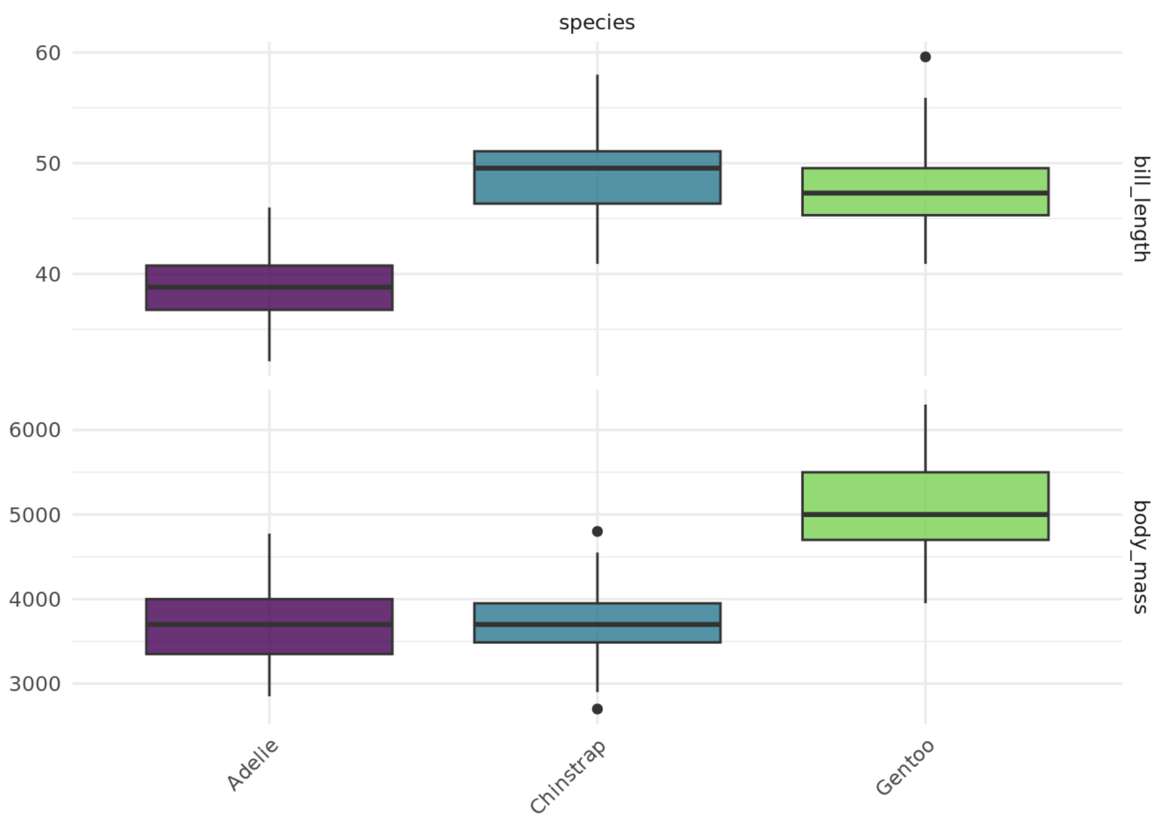
我们先来看看分类任务`Palmer Penguins`的图。我们绘制目标变量的类别频率。

```
library(mlr3verse)
library(mlr3viz)
task = tsk("penguins")
task$select(c("body_mass", "bill_length"))
autoplot(task, type = "target")
```



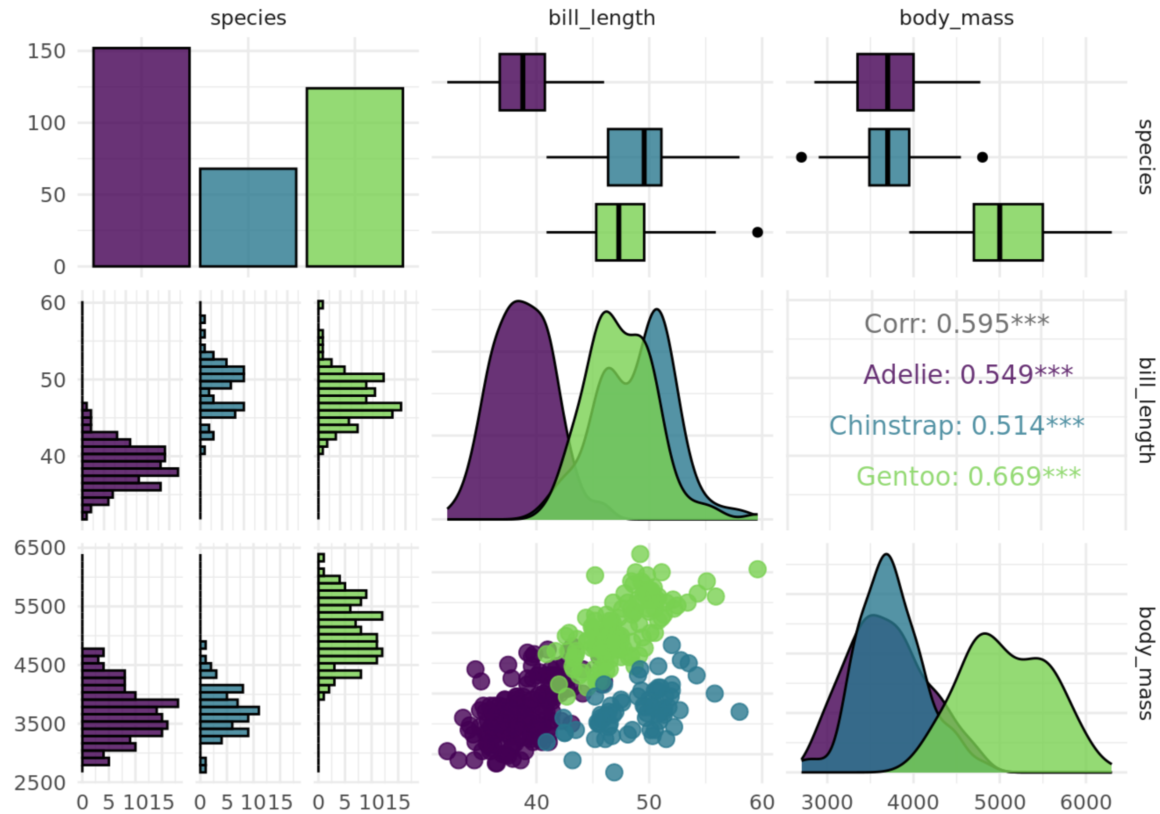
`"duo"` 图展示多个特征的分布：

```
autoplot(task, type = "duo")
```



`"pairs"` 图展示了多个特征的配对比较。目标变量的类别以不同颜色显示。

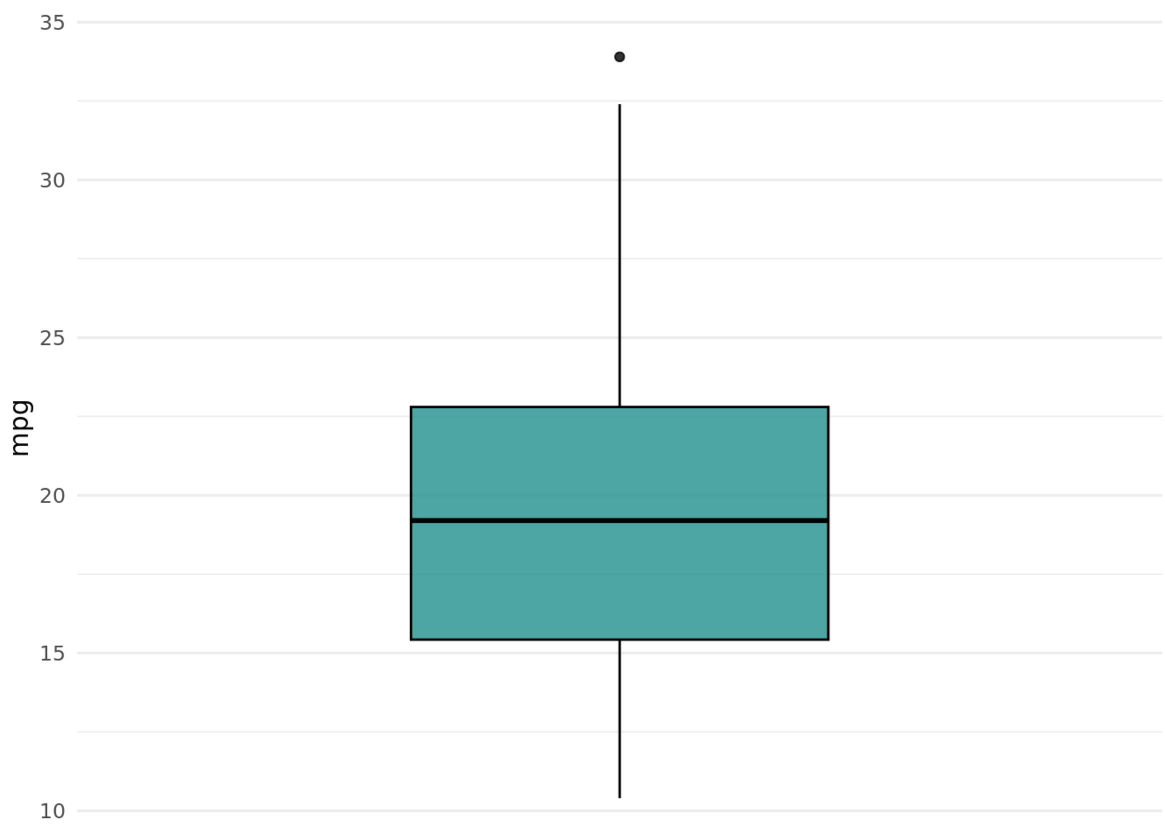
```
autoplot(task, type = "pairs")
```



回归

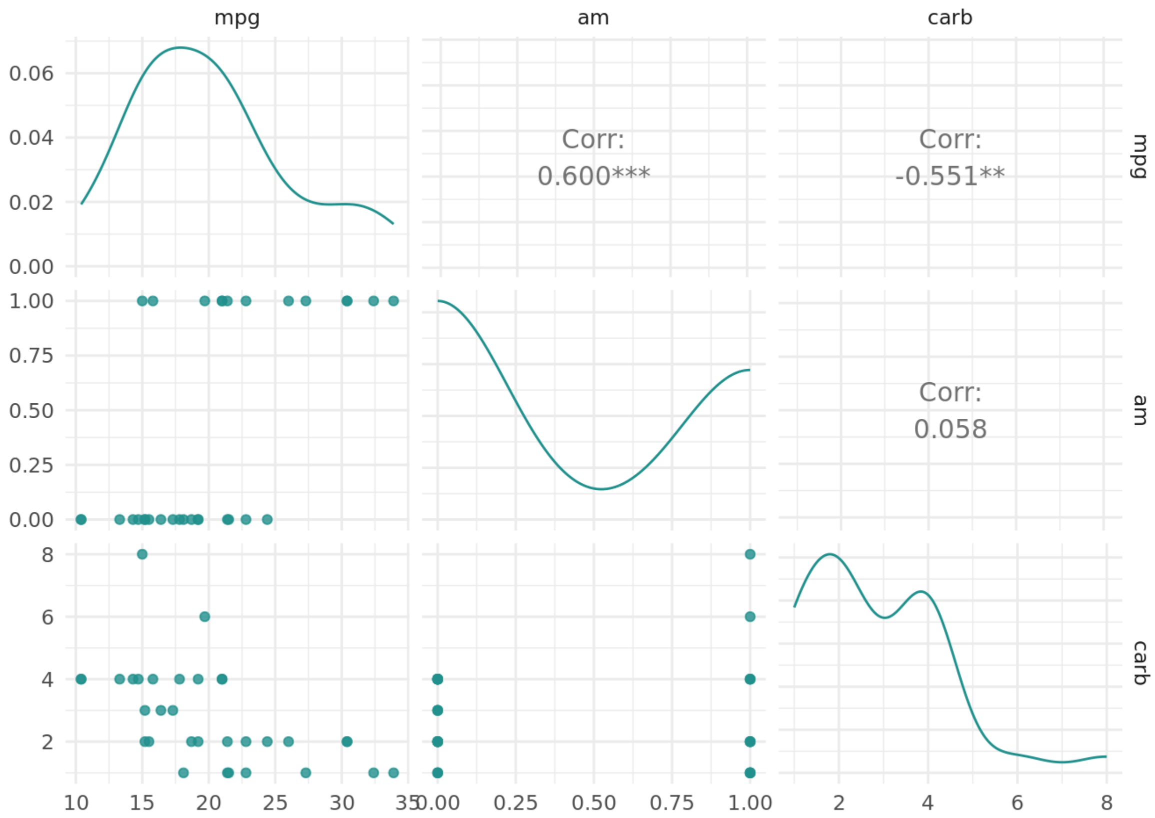
接下来，我们绘制回归任务`mtcars`。我们创建一个目标变量的boxplot。

```
task = tsk("mtcars")
task$select(c("am", "carb"))
autoplot(task, type = "target")
```



`"pairs"` 图显示了多个特征和目标变量的配对比较。

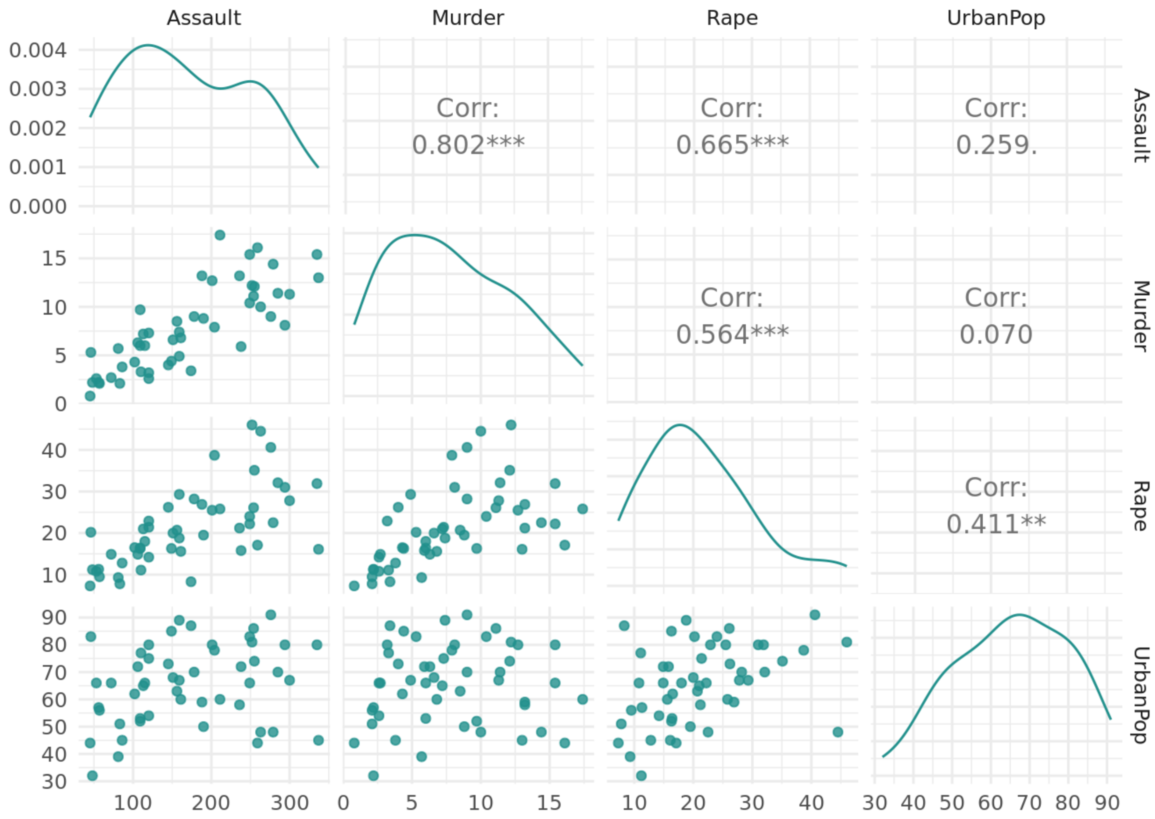
```
autoplot(task, type = "pairs")
```



聚类

最后，我们绘制了聚类任务["美国逮捕案"]。`"pairs"` 图显示了多个特征的配对比较。

```
library(mlr3cluster)
task = mlr_tasks$get("usarrests")
autoplot(task, type = "pairs")
```

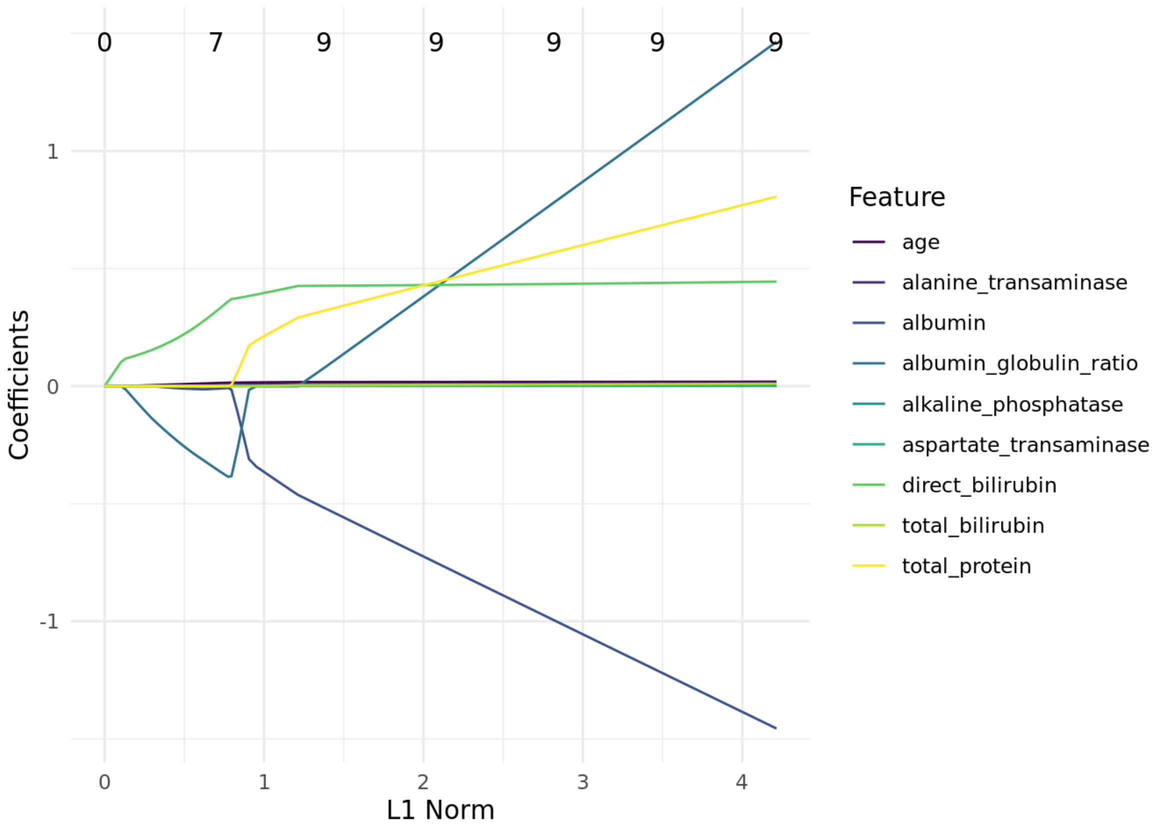


学习器

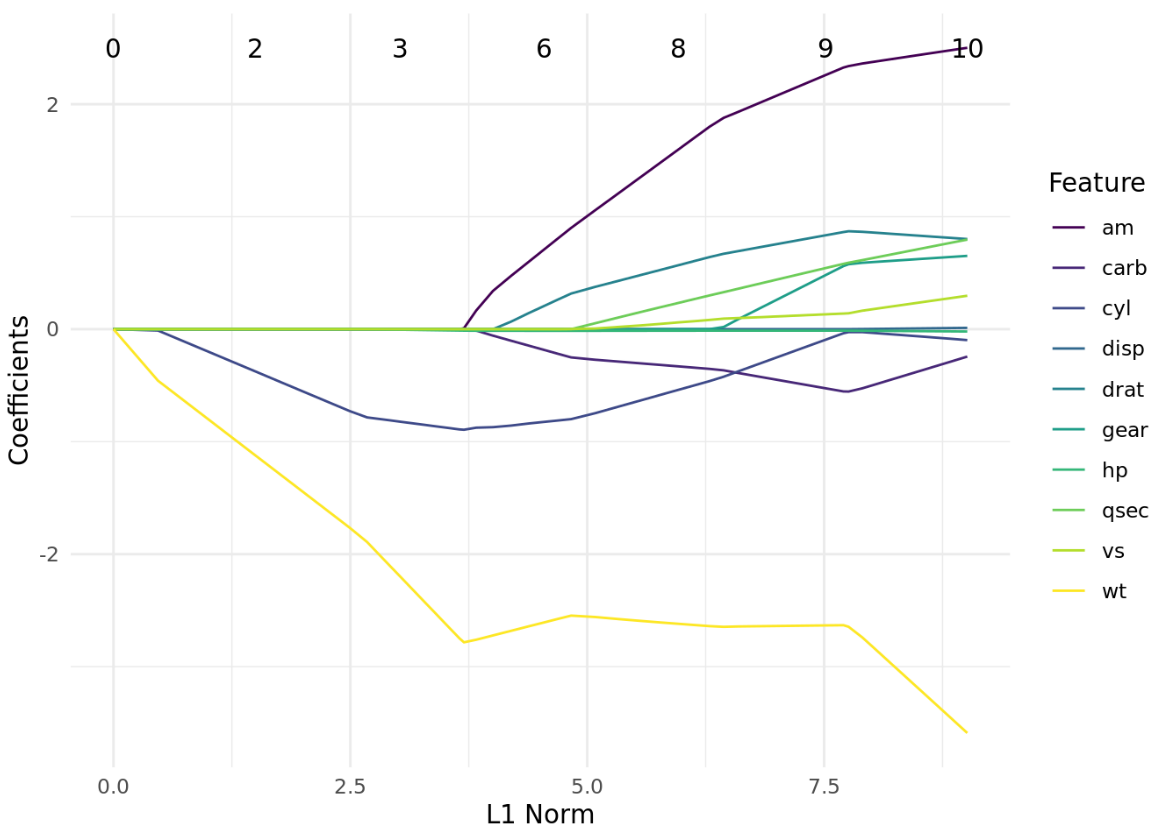
GLMNet

分类 和 回归 GLMNet 学习器配备了一个绘图函数。

```
library(mlr3data)
task = tsk("ilpd")
task$select(setdiff(task$feature_names, "gender"))
learner = lrn("classif.glmnet")
learner$train(task)
autoplot(learner)
```



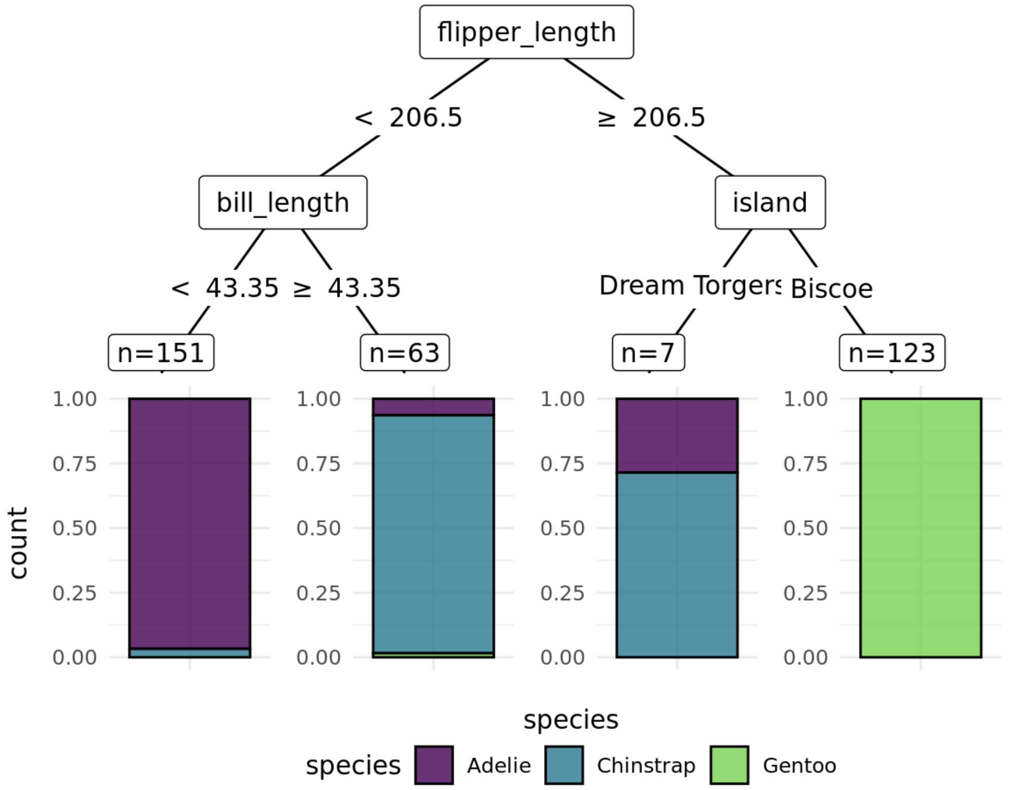

```r
task = tsk("mtcars")
learner = lrn("regr.glmnet")
learner$train(task)
autoplot(learner)
```



Rpart

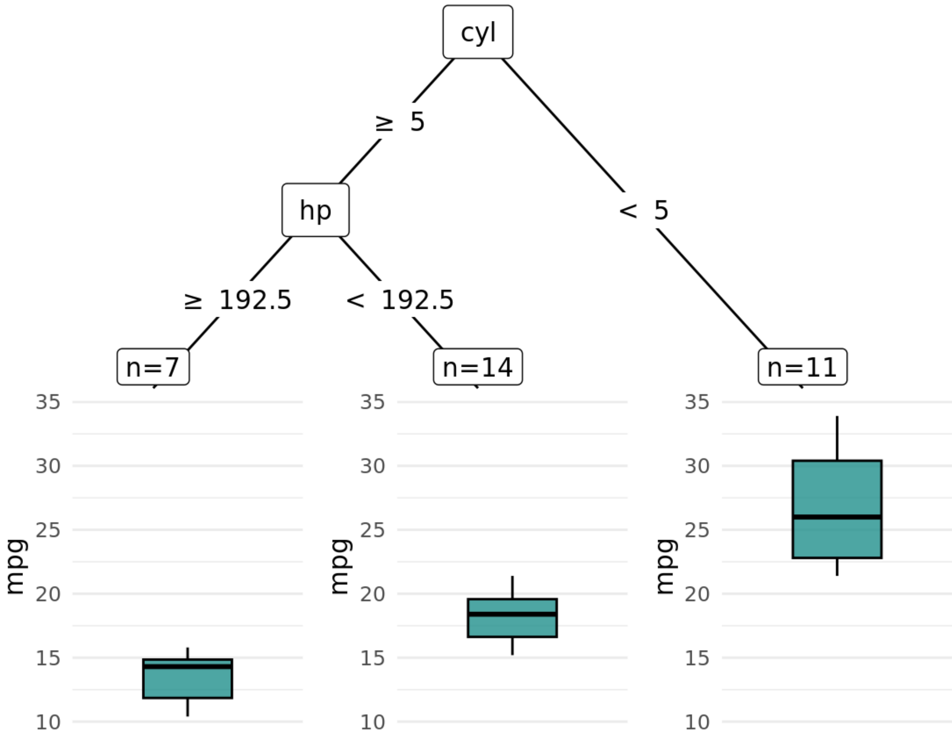
我们绘制了一个rpart包的分类树。我们必须用`keep_model = TRUE`来拟合学习器以保存模型对象。

```
task = tsk("penguins")
learner = lrn("classif.rpart", keep_model = TRUE)
learner$train(task)
autoplot(learner)
```



也可以绘制回归树：

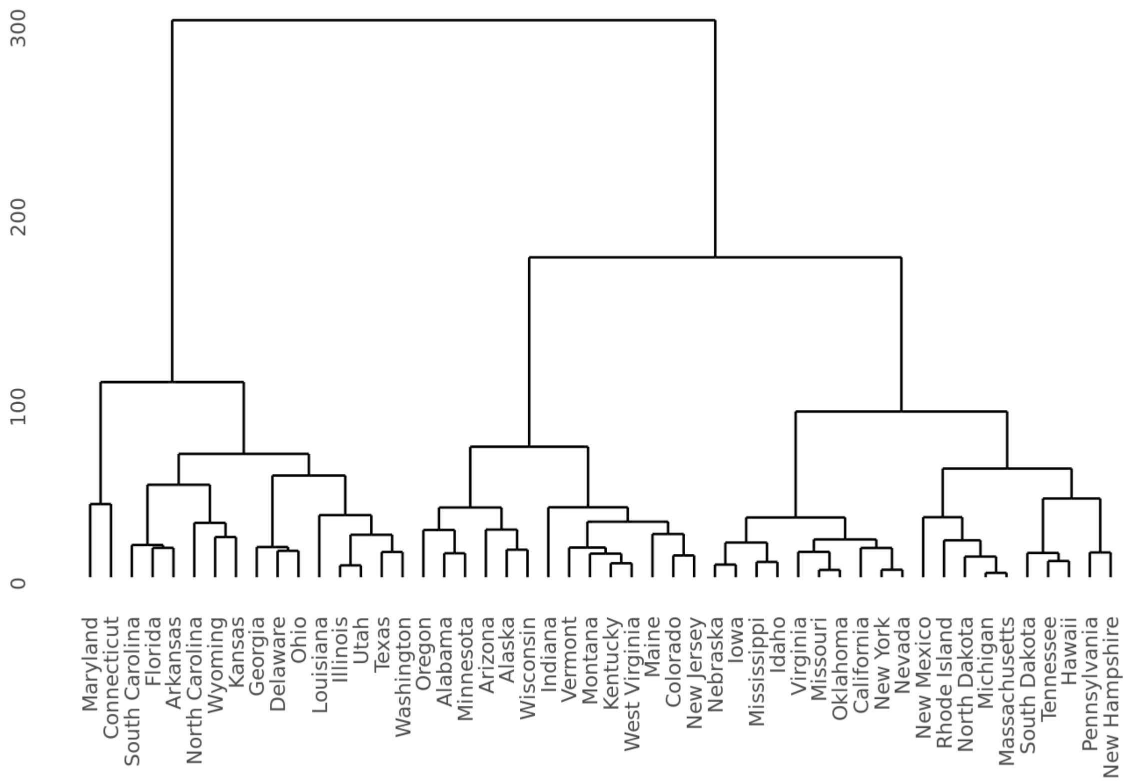
```
task = tsk("mtcars")
learner = lrn("regr.rpart", keep_model = TRUE)
learner$train(task)
autoplot(learner)
```



层次聚类

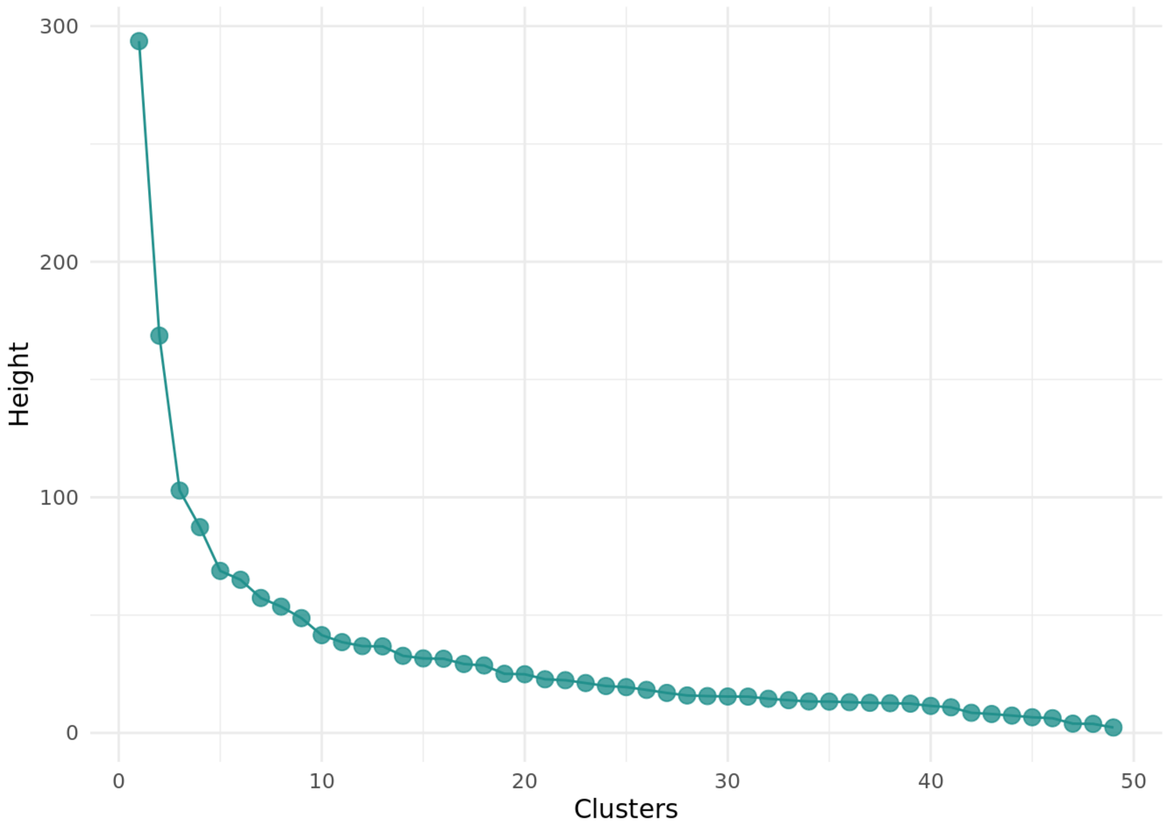
`"dend"` 图显示了数据层次聚类的结果。

```
library(mlr3cluster)
task = tsk("usarrests")
learner = lrn("clust.hclust")
learner$train(task)
autoplot(learner, type = "dend", task = task)
```



`"scree"`类型绘制了聚类的数量和高度（碎石图）：

```
autoplot(learner, type = "scree")
```

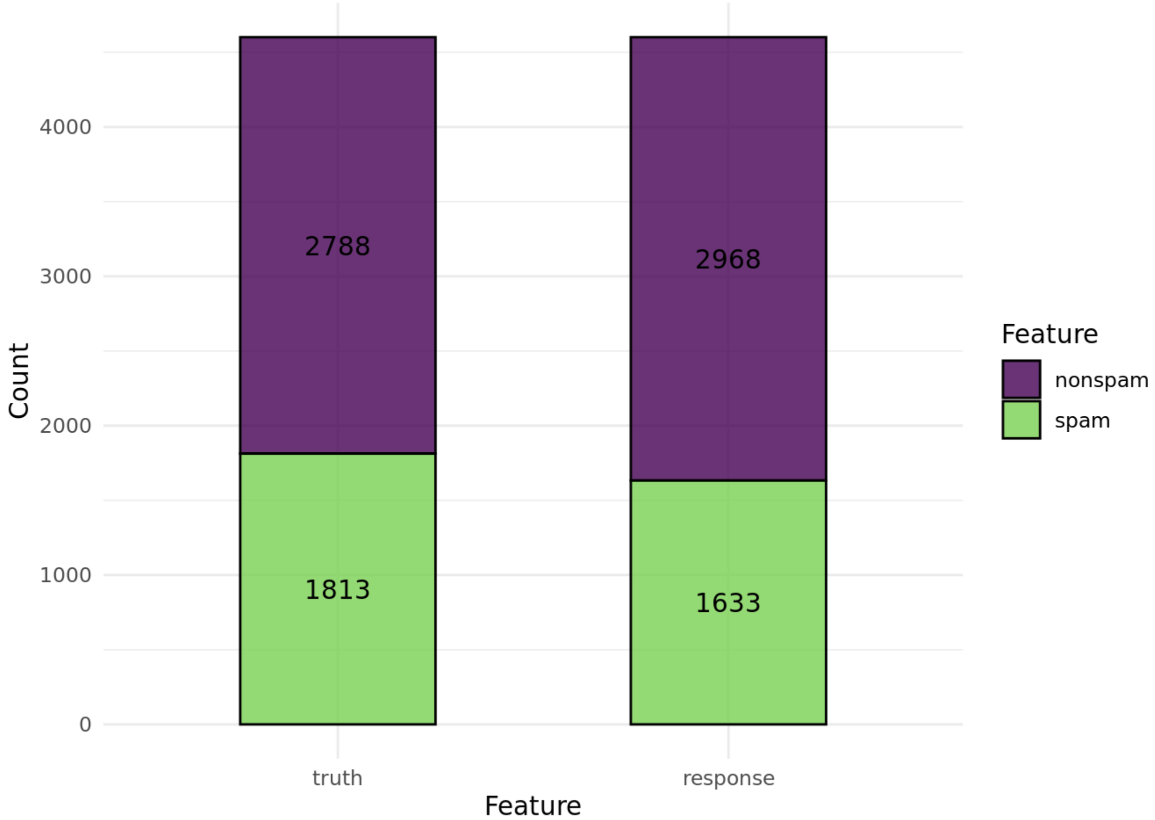


预测对象

分类

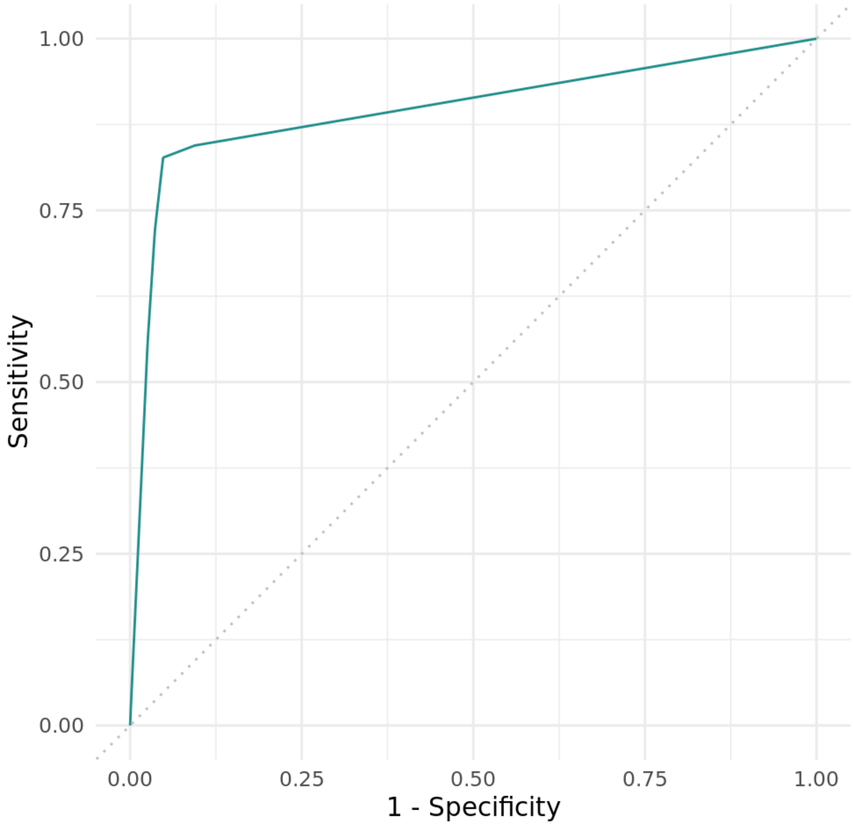
我们绘制了一个分类学习器的预测图。`"stacked"` 图显示了预测的和真实的类别标签。

```
task = tsk("spam")
learner = lrn("classif.rpart", predict_type = "prob")
pred = learner$train(task)$predict(task)
autoplot(pred, type = "stacked")
```



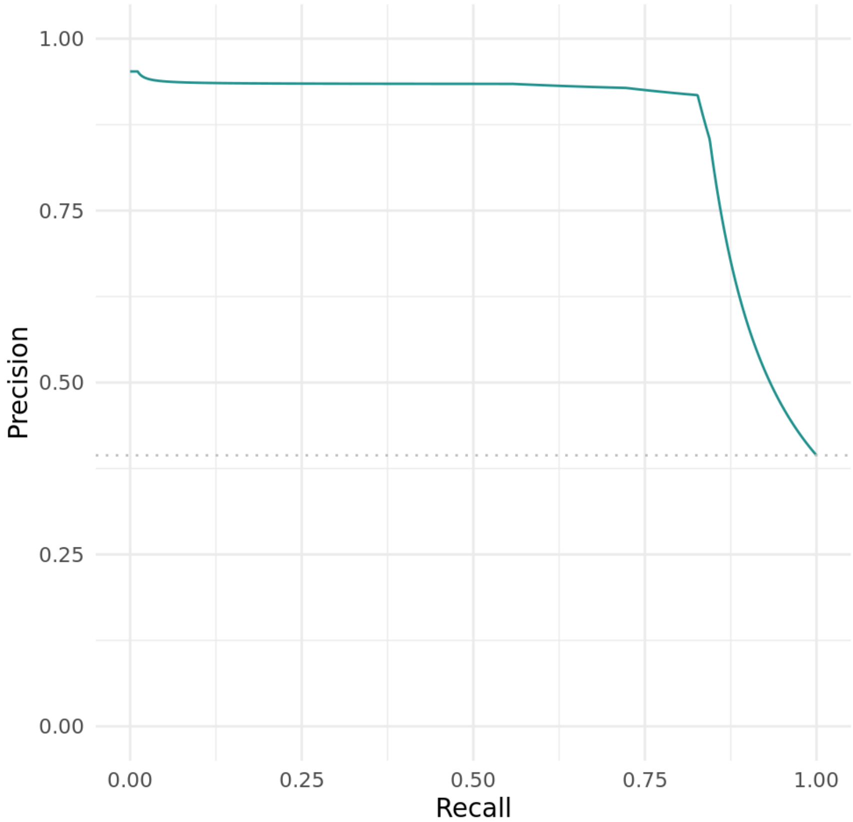
ROC 曲线描绘了不同阈值下的真阳性率与假阳性率。

```
autoplot(pred, type = "roc")
```



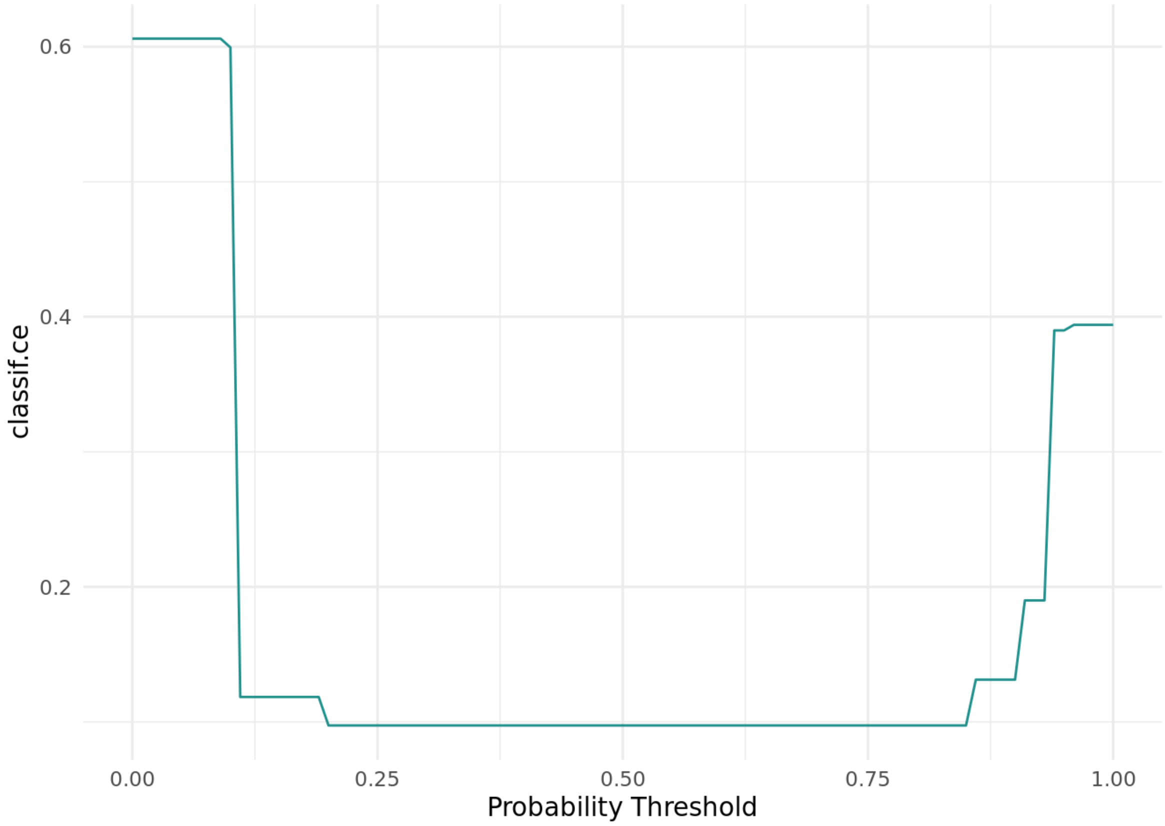
PR曲线（precision-recall）描绘了不同阈值下的查准率与召回率。

```
autoplot(pred, type = "prc")
```



`"threshold"` 图改变了二元分类的阈值，并对由此产生的性能进行了绘制。

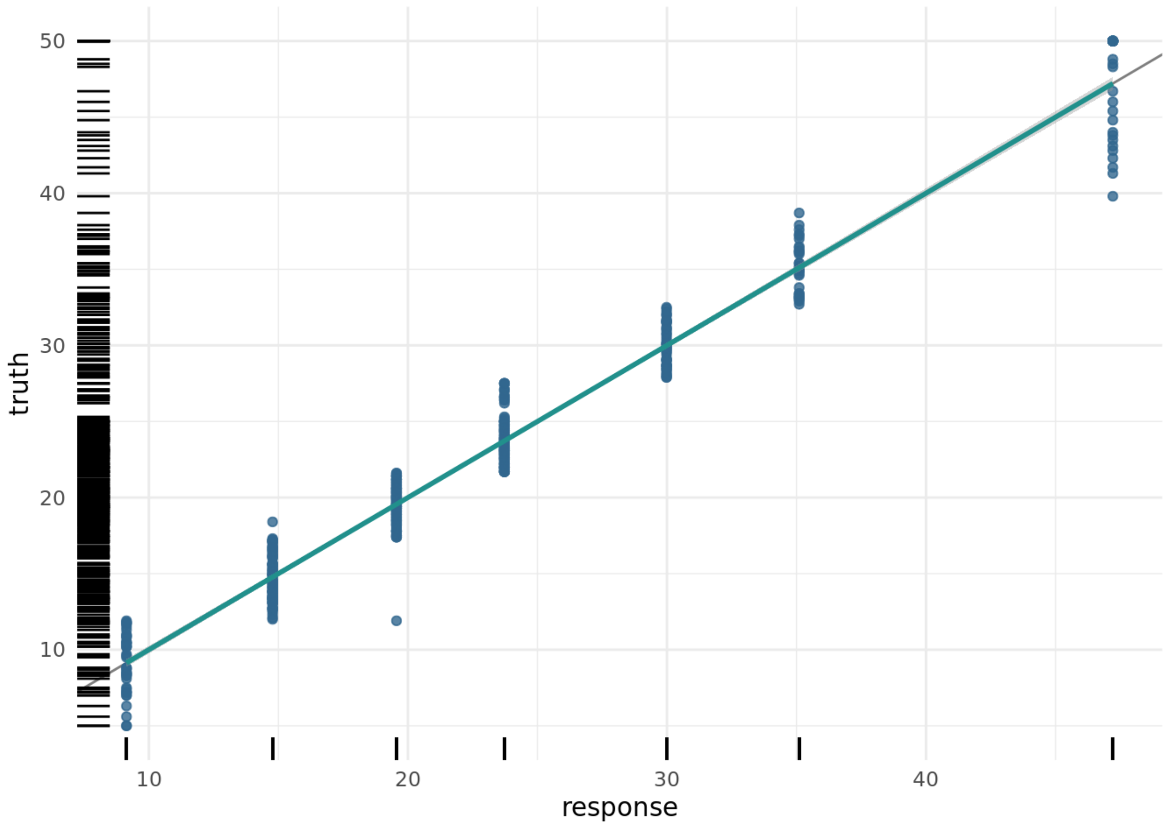
```
autoplot(pred, type = "threshold")
```



回归

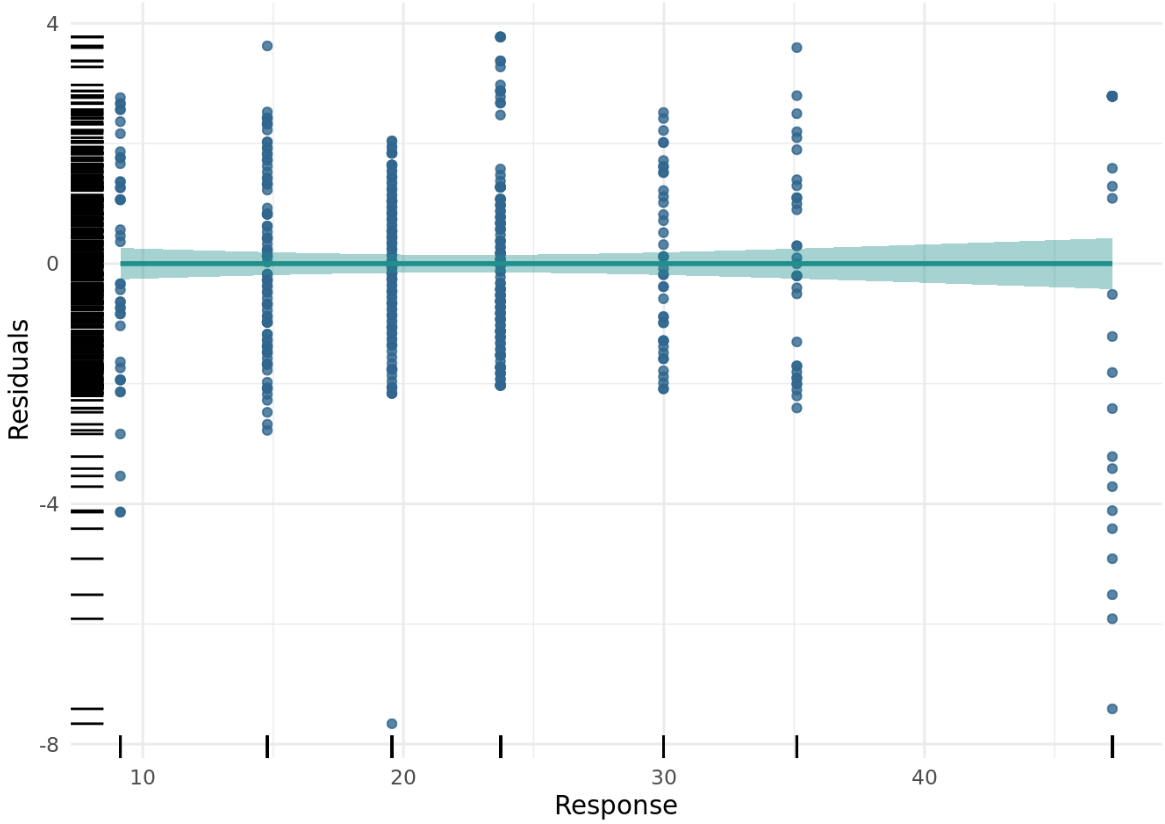
回归学习器的预测通常表现为真实值和预测值的散点图。

```
task = tsk("boston_housing")
learner = lrn("regr.rpart")
pred = learner$train(task)$predict(task)
autoplot(pred, type = "xy")
```



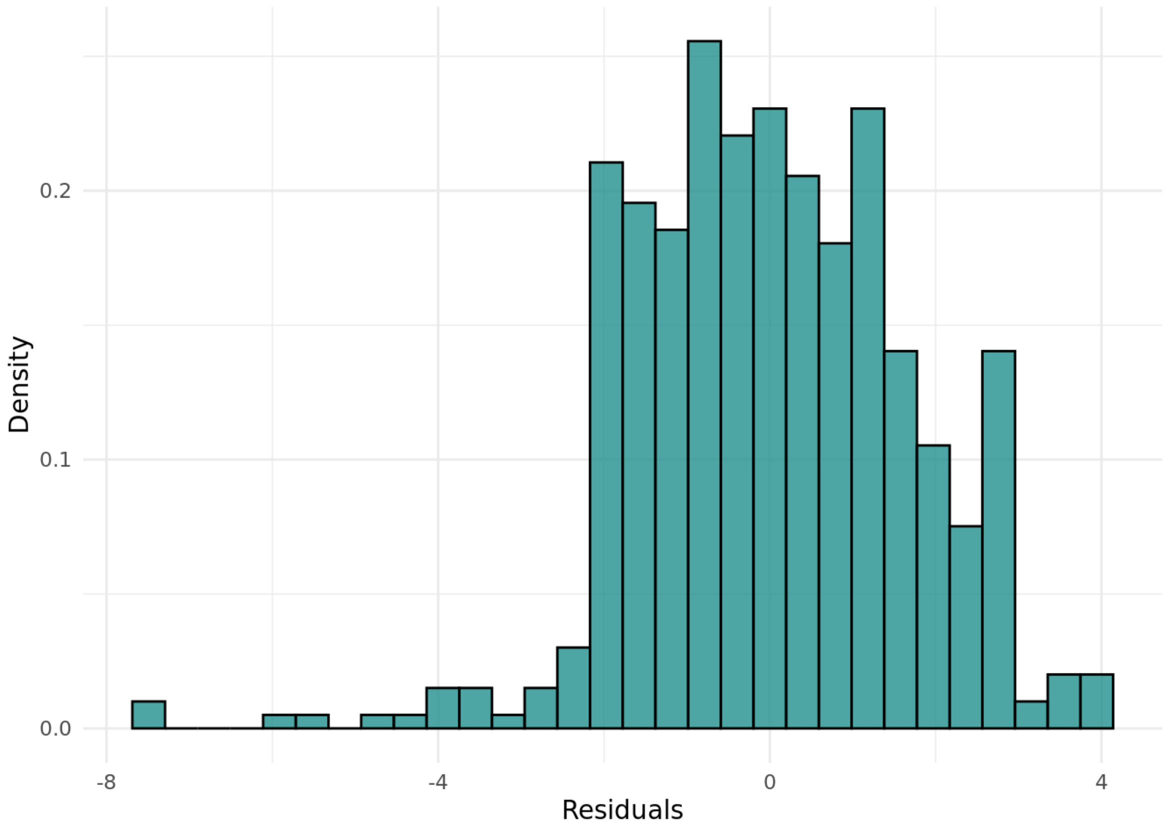
此外，我们还绘制了响应与残差。

```
autoplot(pred, type = "residual")
```



我们还可以绘制残差的分布图。

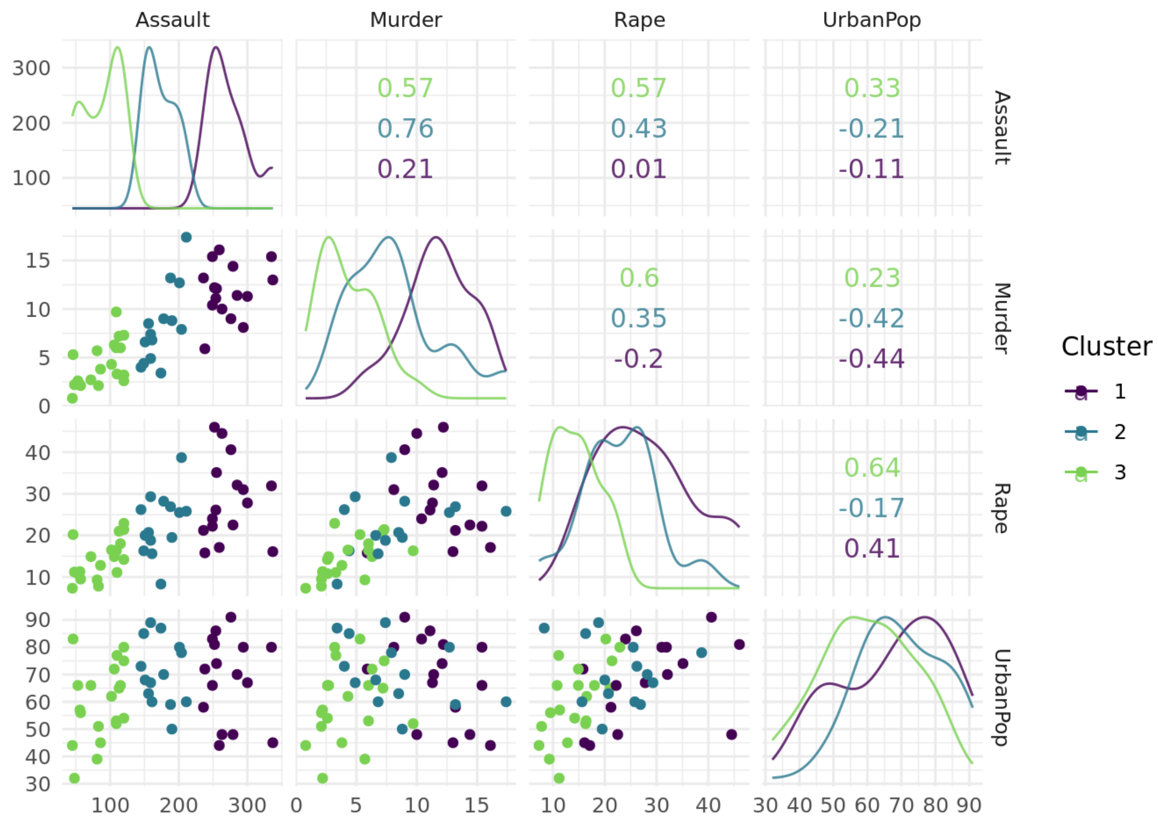
```
autoplot(pred, type = "histogram")
```



聚类

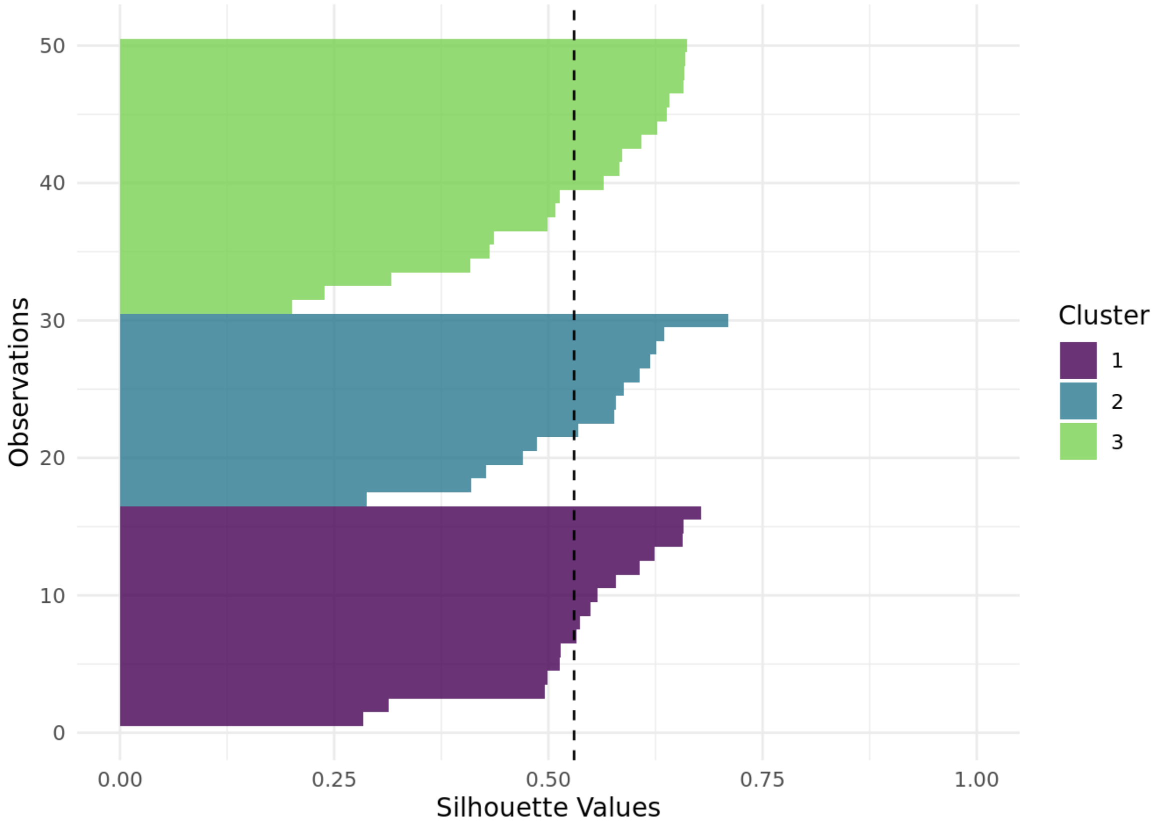
聚类学习器的预测通常以按聚类着色的数据点的散点图来表示。

```
library(mlr3cluster)
task = tsk("usarrests")
learner = lrn("clust.kmeans", centers = 3)
pred = learner$train(task)$predict(task)
autoplot(pred, task, type = "scatter")
```



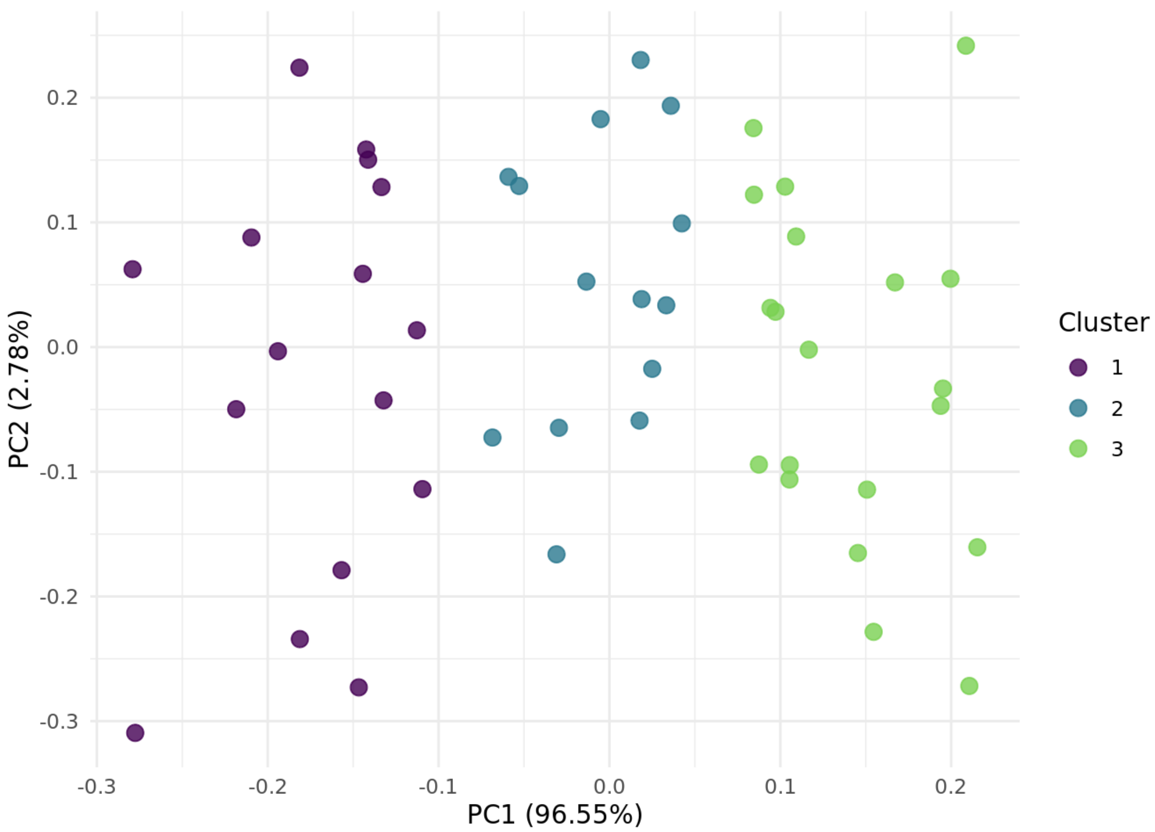
`"sil"` 图显示了聚类的 silhouette 宽度。虚线是平均 silhouette 宽度。

```
autoplot(pred, task, type = "sil")
```



`"pca"` 图中显示了按聚类着色的数据的前两个主成分：

```
autoplot(pred, task, type = "pca")
```



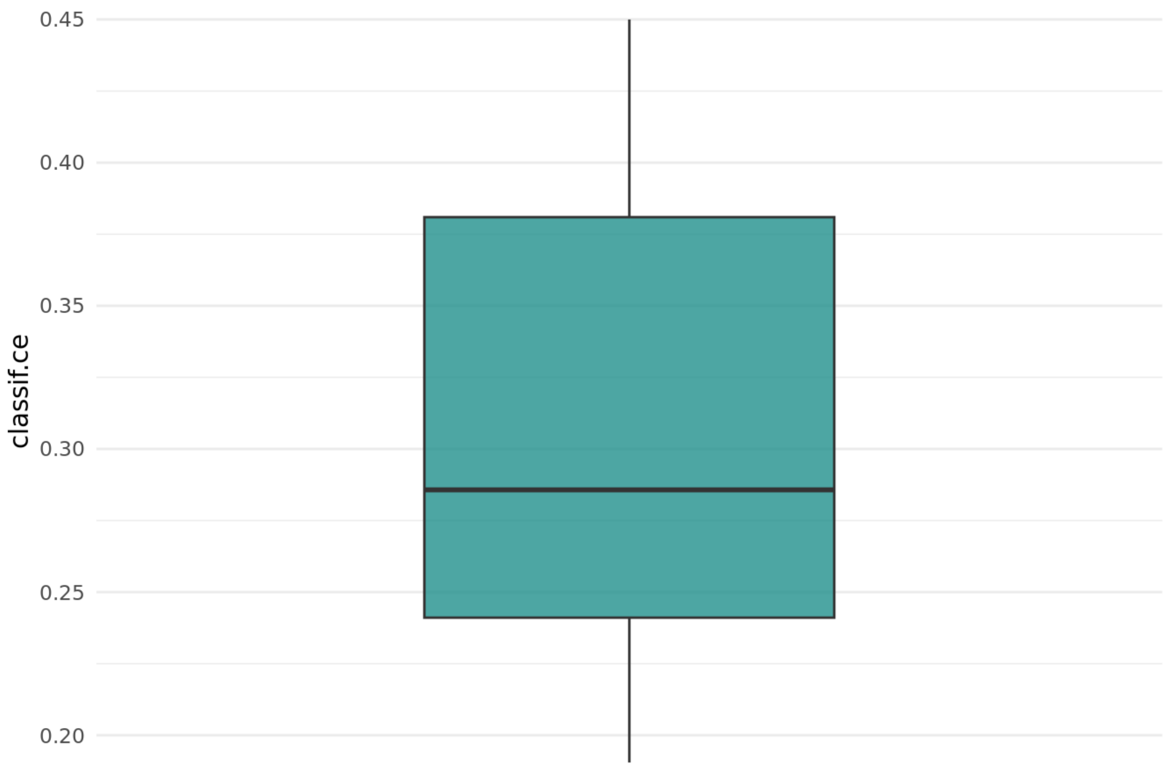
重抽样结果

分类

`"boxplot"` 展示性能指标的分布：

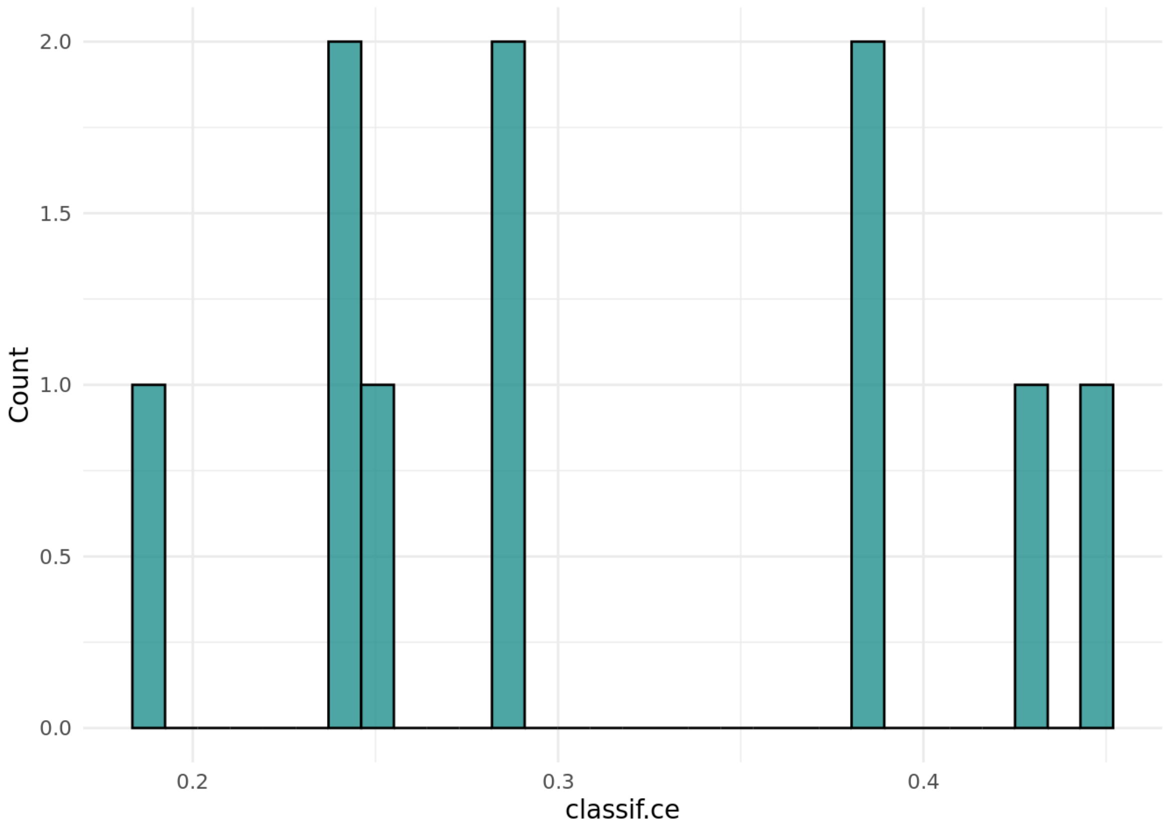
```
task = tsk("sonar")
learner = lrn("classif.rpart", predict_type = "prob")
resampling = rsmp("cv")
rr = resample(task, learner, resampling)

autoplot(rr, type = "boxplot")
```



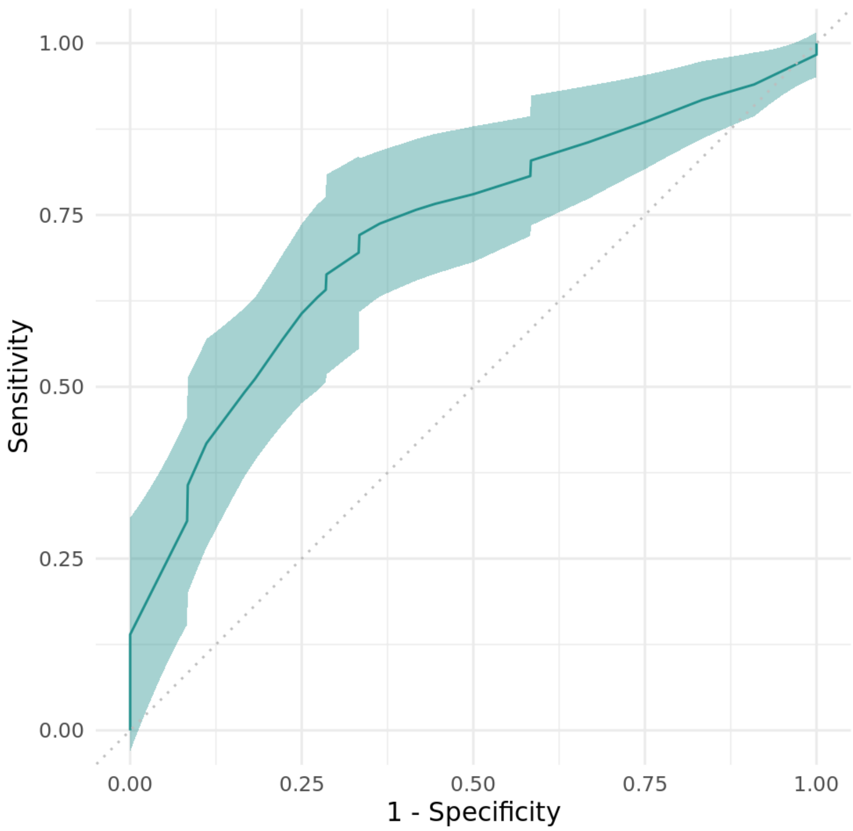
我们还可以将性能指标的分布绘制成“`histogram`”。

```
autoplot(rr, type = "histogram")
```



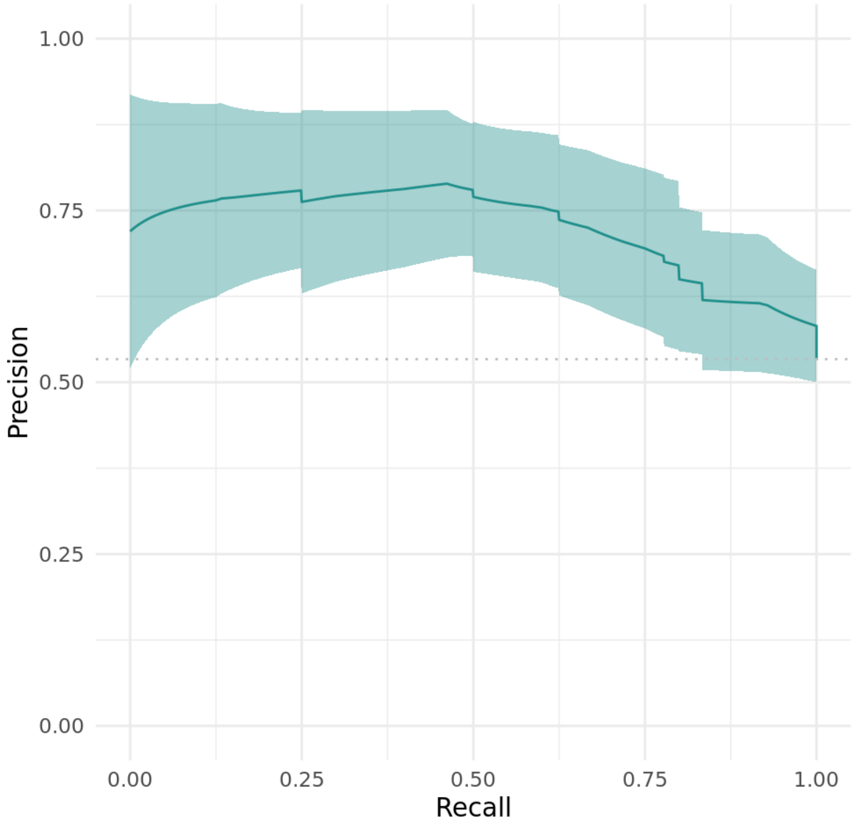
ROC 曲线描绘了不同阈值下的真阳性率与假阳性率。

```
autoplot(rr, type = "roc")
```



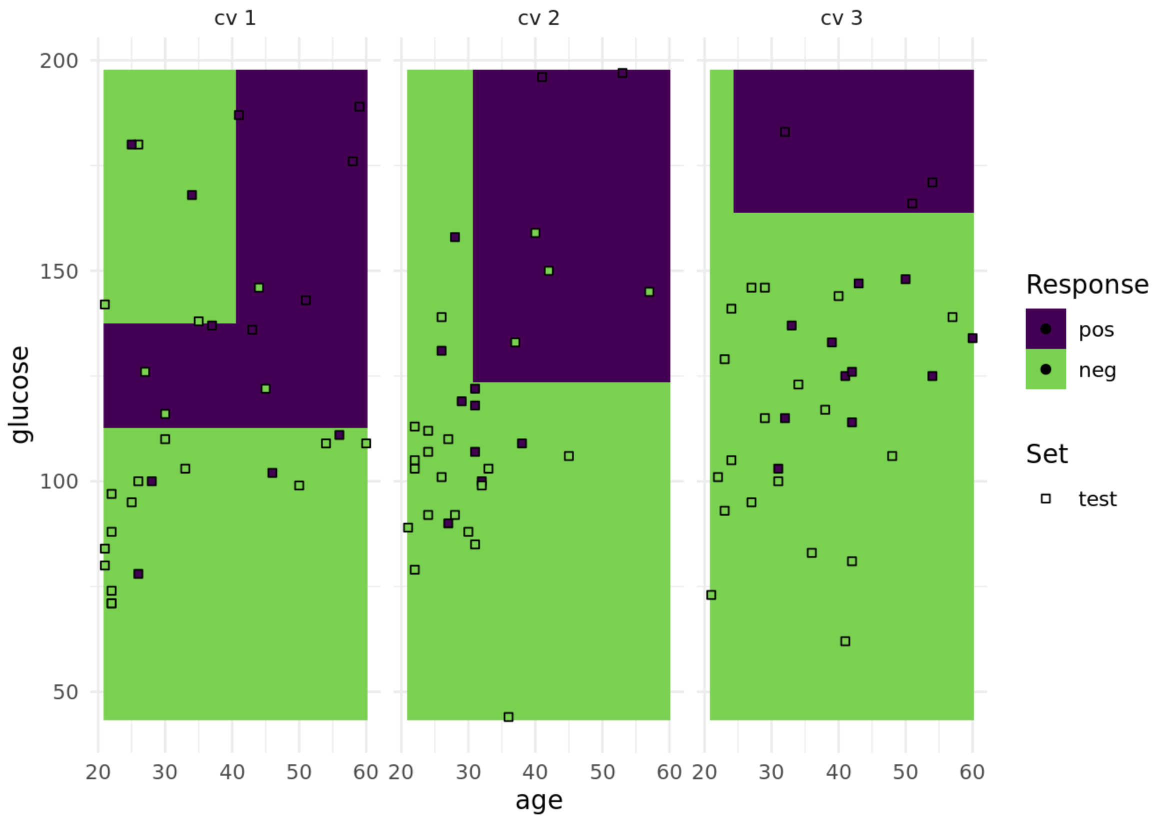
PR曲线（precision-recall）描绘了不同阈值下的查准率与召回率。

```
autoplot(rr, type = "prc")
```



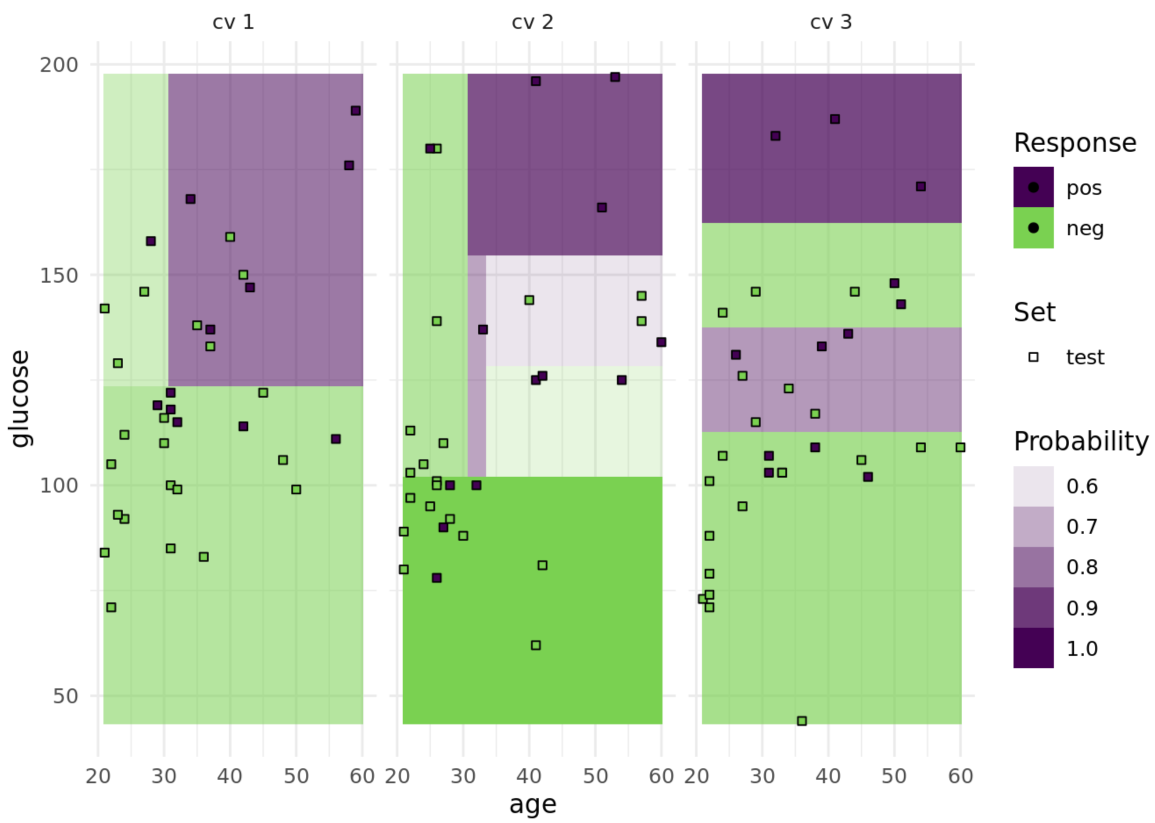
`"prediction"`图显示了两个特征和背景色表示的预测类别。点标志着测试集的观察结果，颜色呈现出真实值。

```
task = tsk("pima")
task$filter(seq(100))
task$select(c("age", "glucose"))
learner = lrn("classif.rpart")
resampling = rsmp("cv", folds = 3)
rr = resample(task, learner, resampling, store_models = TRUE)
autoplot(rr, type = "prediction")
```



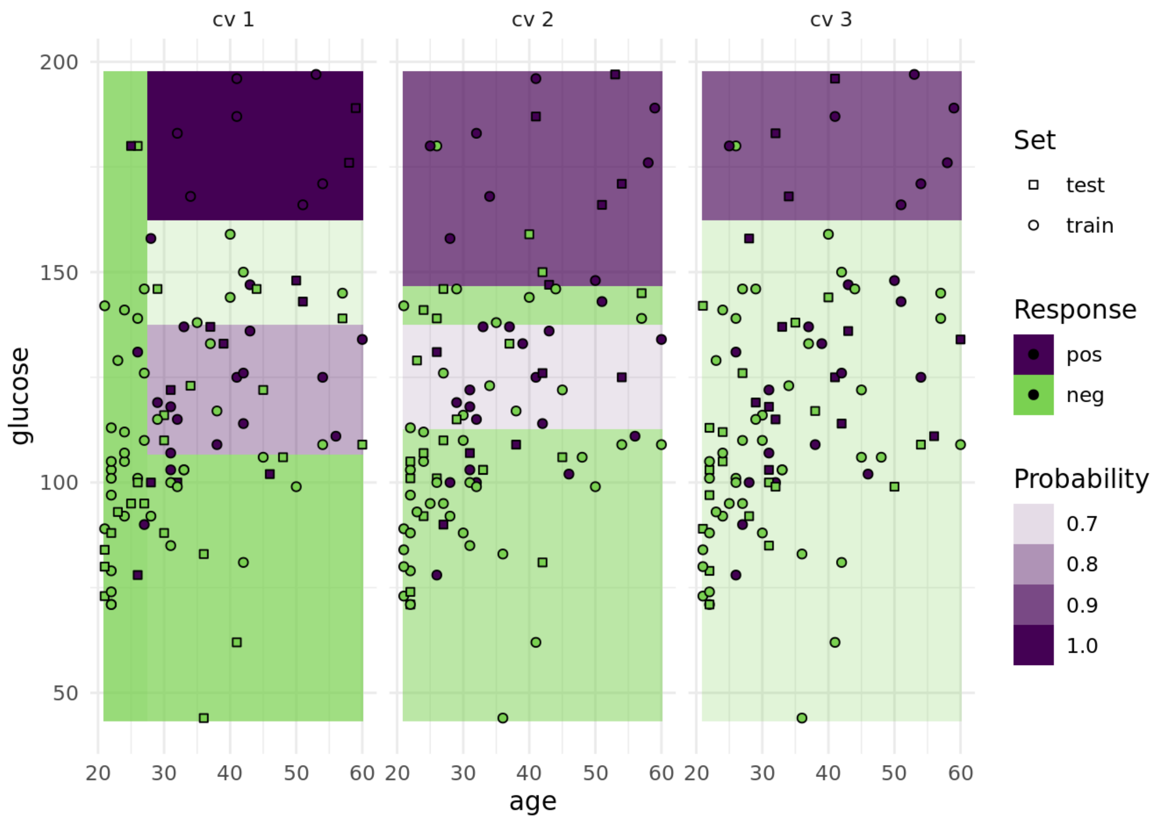
或者，可以绘制类概率：

```
task = tsk("pima")
task$filter(seq(100))
task$select(c("age", "glucose"))
learner = lrn("classif.rpart", predict_type = "prob")
resampling = rsmp("cv", folds = 3)
rr = resample(task, learner, resampling, store_models = TRUE)
autoplot(rr, type = "prediction")
```



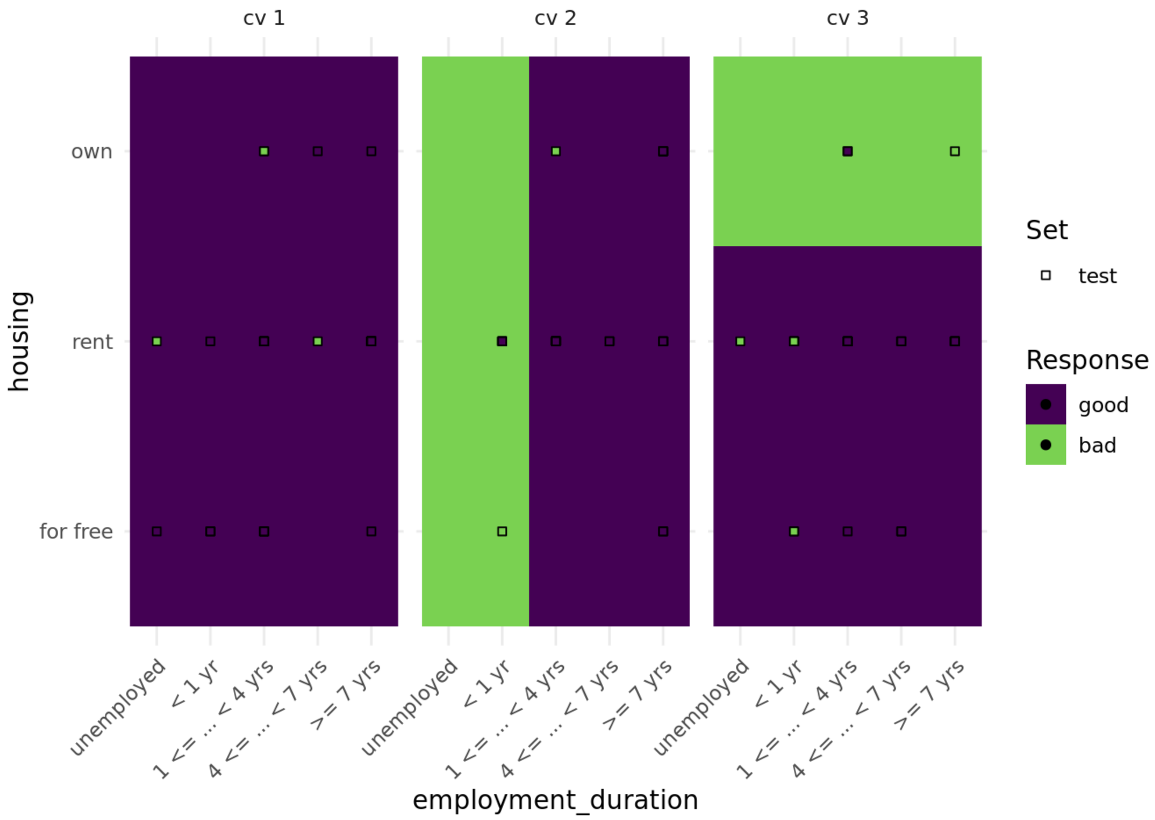
除了测试集，还可以绘制训练集：

```
task = tsk("pima")
task$filter(seq(100))
task$select(c("age", "glucose"))
learner = lrn("classif.rpart", predict_type = "prob", predict_sets = c("train",
"test"))
resampling = rsmp("cv", folds = 3)
rr = resample(task, learner, resampling, store_models = TRUE)
autoplot(rr, type = "prediction", predict_sets = c("train", "test"))
```



`"prediction"` 图也可以展示类别特征：

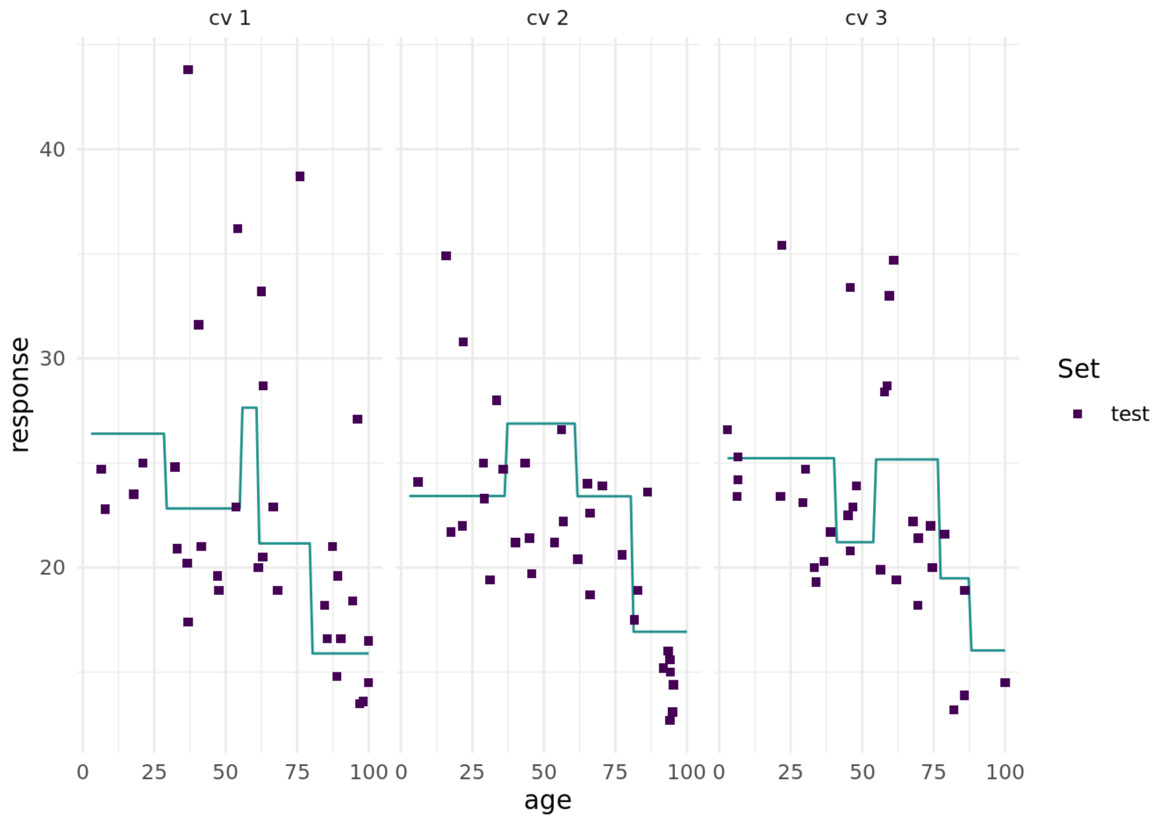
```
task = tsk("german_credit")
task$filter(seq(100))
task$select(c("housing", "employment_duration"))
learner = lrn("classif.rpart")
resampling = rsmp("cv", folds = 3)
rr = resample(task, learner, resampling, store_models = TRUE)
autoplot(rr, type = "prediction")
```



回归

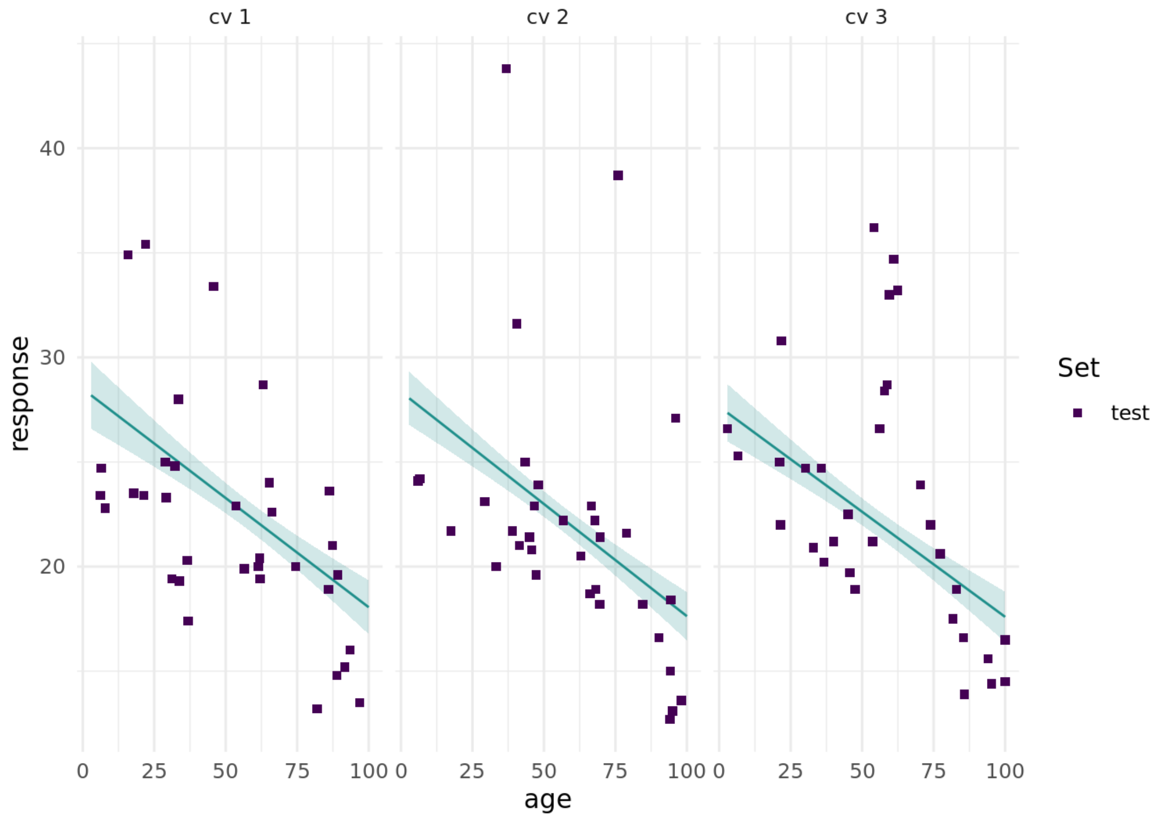
“`prediction`”图显示了一个特征和响应。点标志着测试集的观察结果。

```
task = tsk("boston_housing")
task$select("age")
task$filter(seq(100))
learner = lrn("regr.rpart")
resampling = rsmp("cv", folds  = 3)
rr = resample(task, learner, resampling, store_models = TRUE)
autoplot(rr, type = "prediction")
```



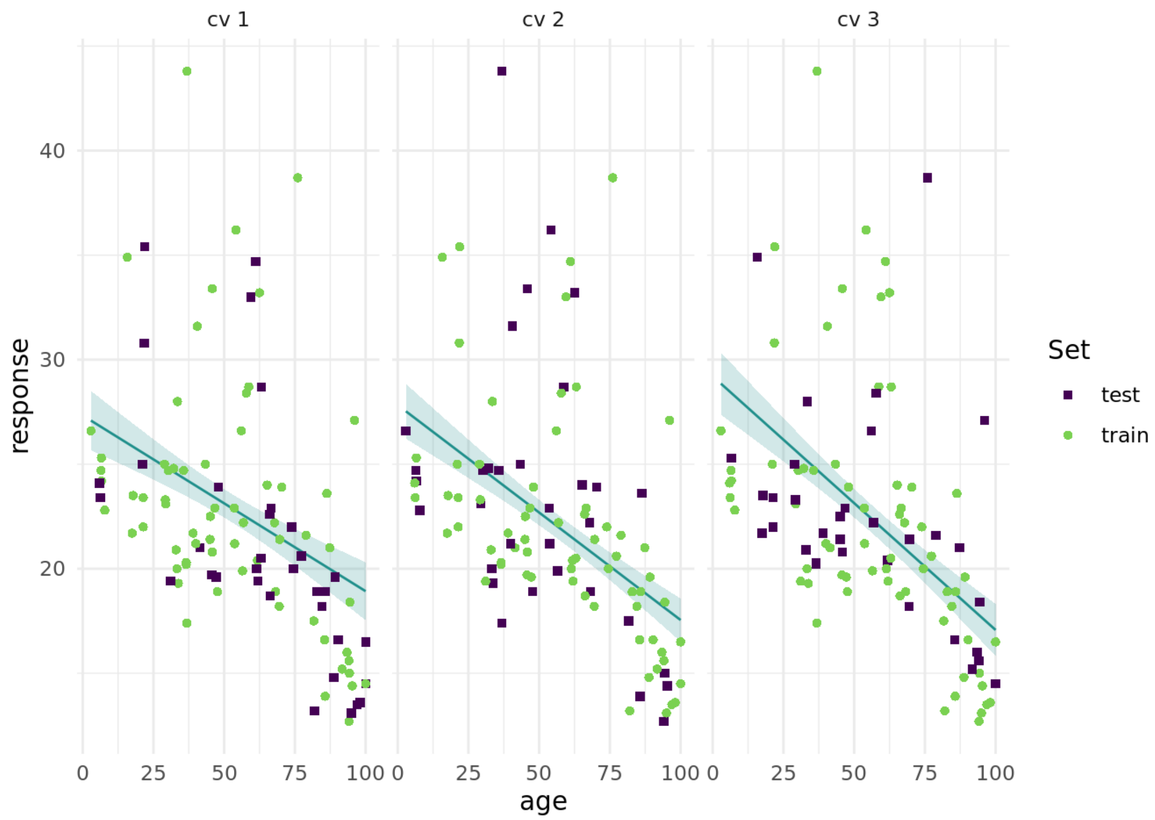
另外，还可以绘制置信带：

```
task = tsk("boston_housing")
task$select("age")
task$filter(seq(100))
learner = lrn("regr.lm", predict_type = "se")
resampling = rsmp("cv", folds  = 3)
rr = resample(task, learner, resampling, store_models = TRUE)
autoplot(rr, type = "prediction")
```



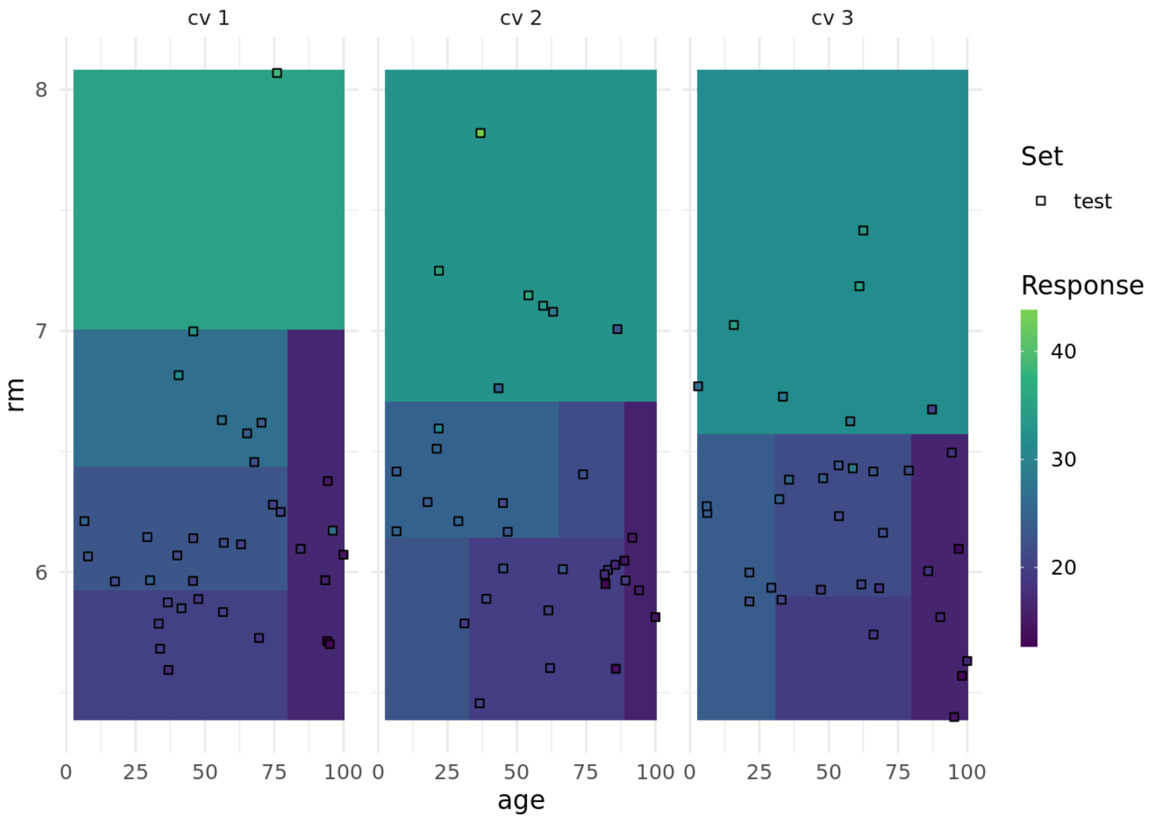
再增加训练集：

```
task = tsk("boston_housing")
task$select("age")
task$filter(seq(100))
learner = lrn("regr.lm", predict_type = "se", predict_sets = c("train", "test"))
resampling = rsmp("cv", folds = 3)
rr = resample(task, learner, resampling, store_models = TRUE)
autoplot(rr, type = "prediction", predict_sets = c("train", "test"))
```



我们还可以将预测面添加到背景图中：

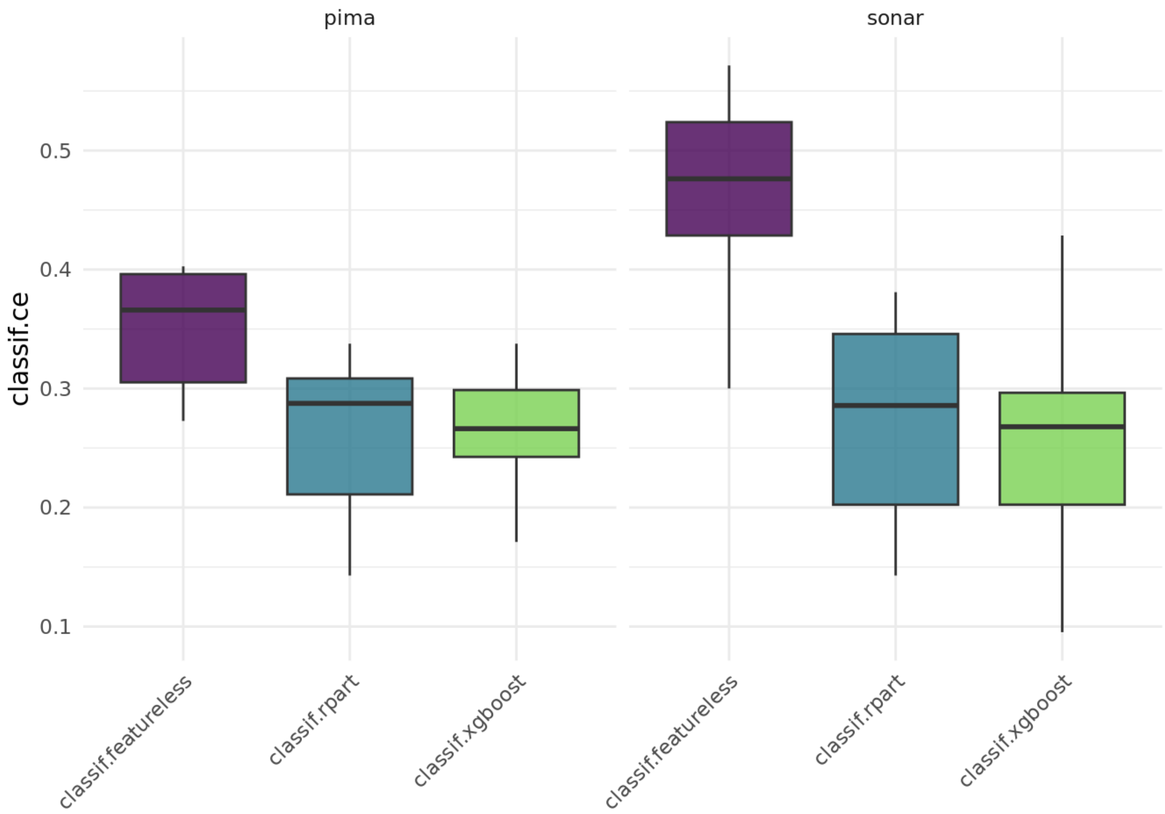
```
task = tsk("boston_housing")
task$select(c("age", "rm"))
task$filter(seq(100))
learner = lrn("regr.rpart")
resampling = rsmp("cv", folds = 3)
rr = resample(task, learner, resampling, store_models = TRUE)
autoplot(rr, type = "prediction")
```



基准测试结果

我们展示了一个有多个任务基准测试的性能分布。

```
tasks = tsks(c("pima", "sonar"))
learner = lrns(c("classif.featureless", "classif.rpart", "classif.xgboost"),
predict_type = "prob")
resampling = rsmps("cv")
bmr = benchmark(benchmark_grid(tasks, learner, resampling))
autoplot(bmr, type = "boxplot")
```

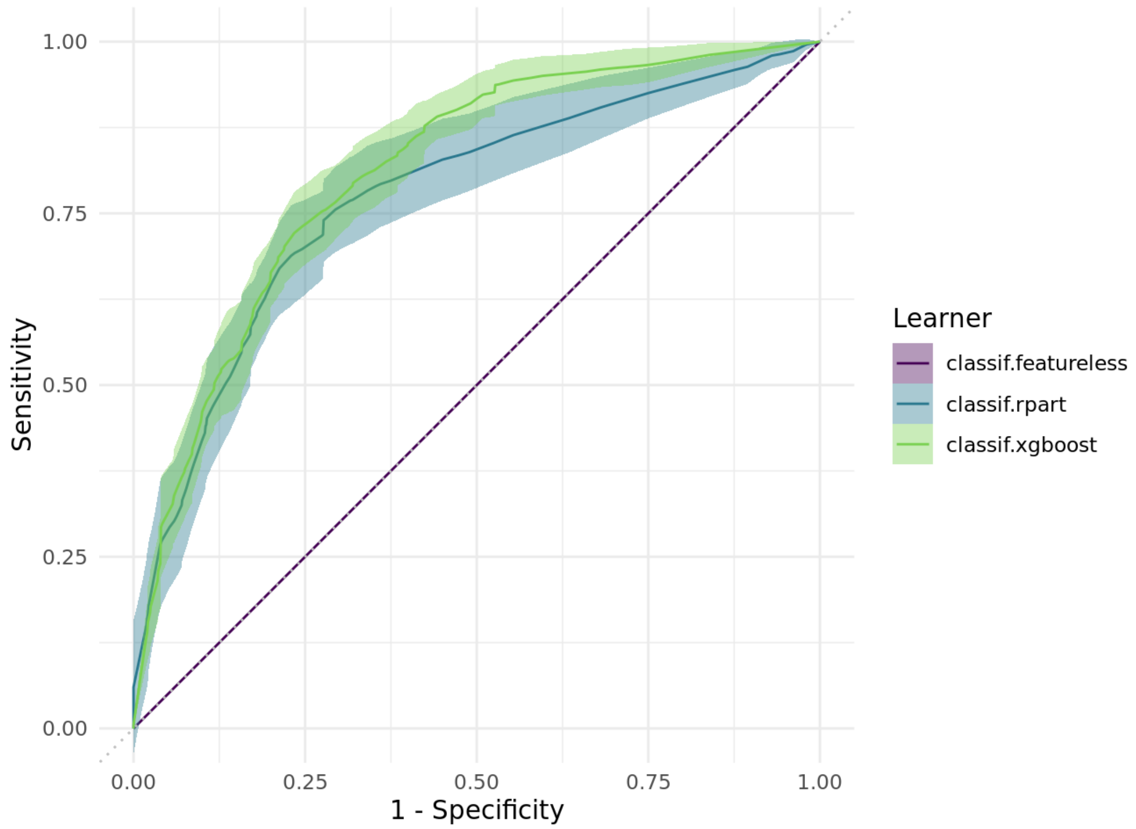


我们绘制了一个有一个任务和多个学习器的基准测试结果：

```
tasks = tsk("pima")
learner = lrns(c("classif.featureless", "classif.rpart", "classif.xgboost"),
predict_type = "prob")
resampling = rsmps("cv")
bmr = benchmark(benchmark_grid(tasks, learner, resampling))
```

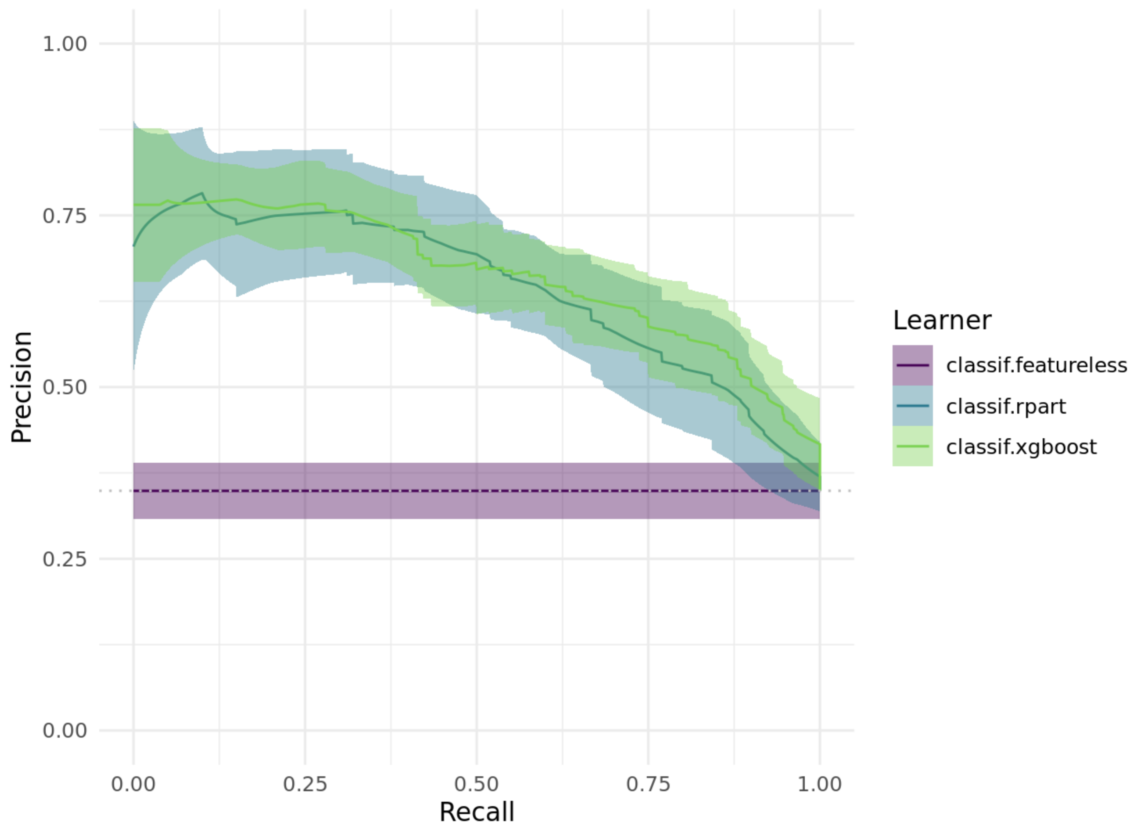
为每个学习器绘制ROC曲线：

```
autoplot(bmr, type = "roc")
```



或者为每个学习器绘制PR曲线：

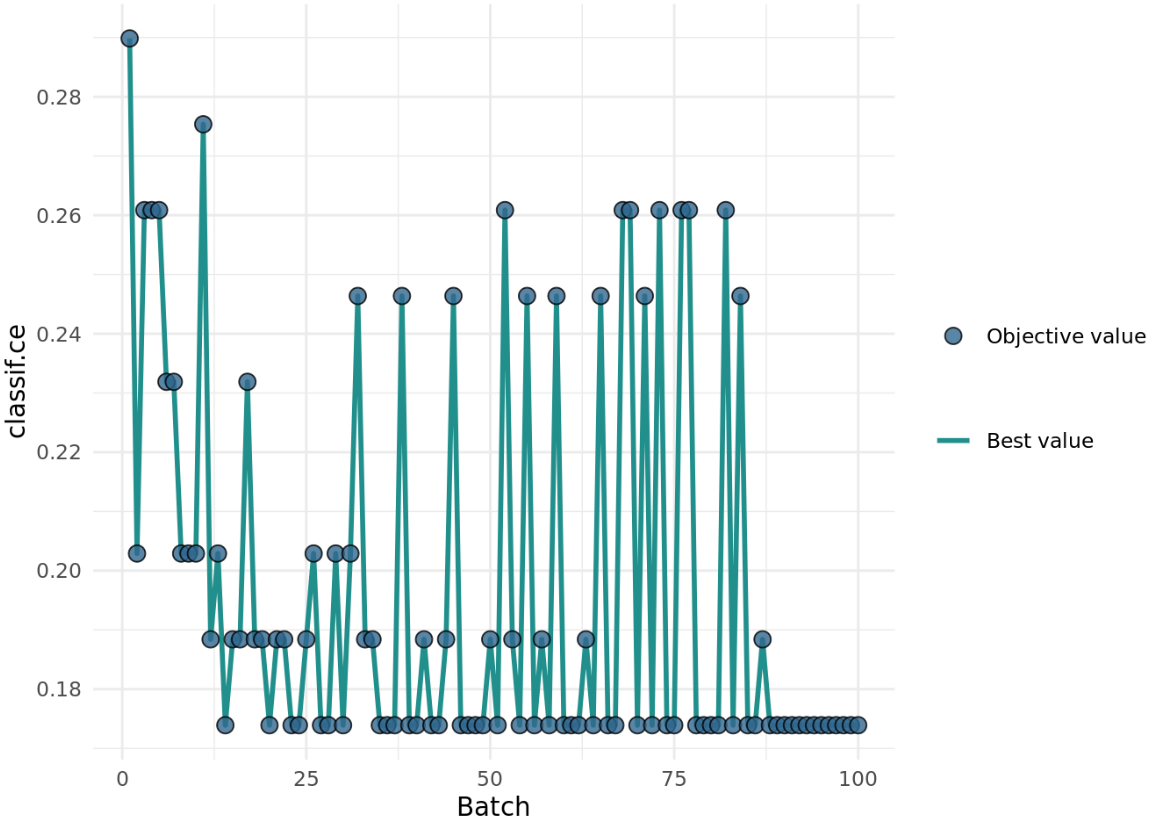
```
autoplot(bmr, type = "prc")
```



调参实例

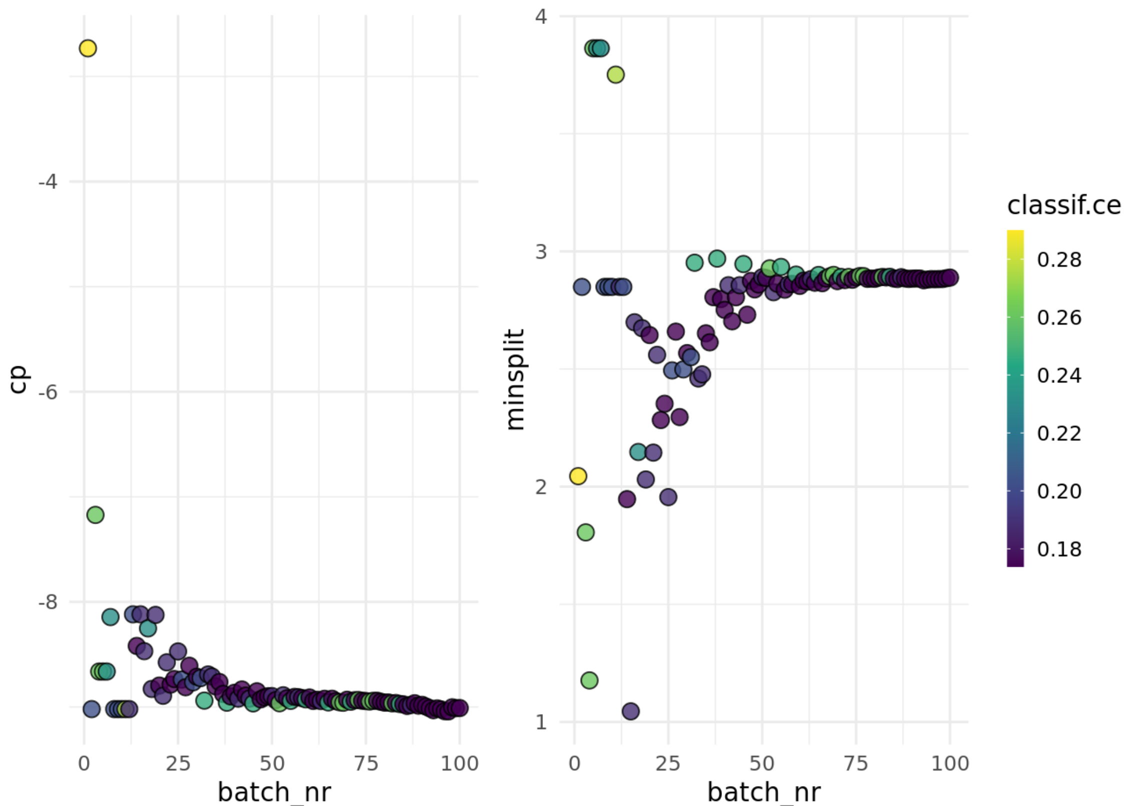
我们对`"sonar"`任务的决策树的超参数进行调参，"性能 "图显示了各批次的性能。

```r
library(mlr3tuning)
library(mlr3tuningspaces)
library(mlr3learners)
instance = tune(
  method = tnr("gensa"),
  task = tsk("sonar"),
  learner = lts(lrn("classif.rpart")),
  resampling = rsmp("holdout"),
  measures = msr("classif.ce"),
  term_evals = 100
)
autoplot(instance, type = "performance")
```



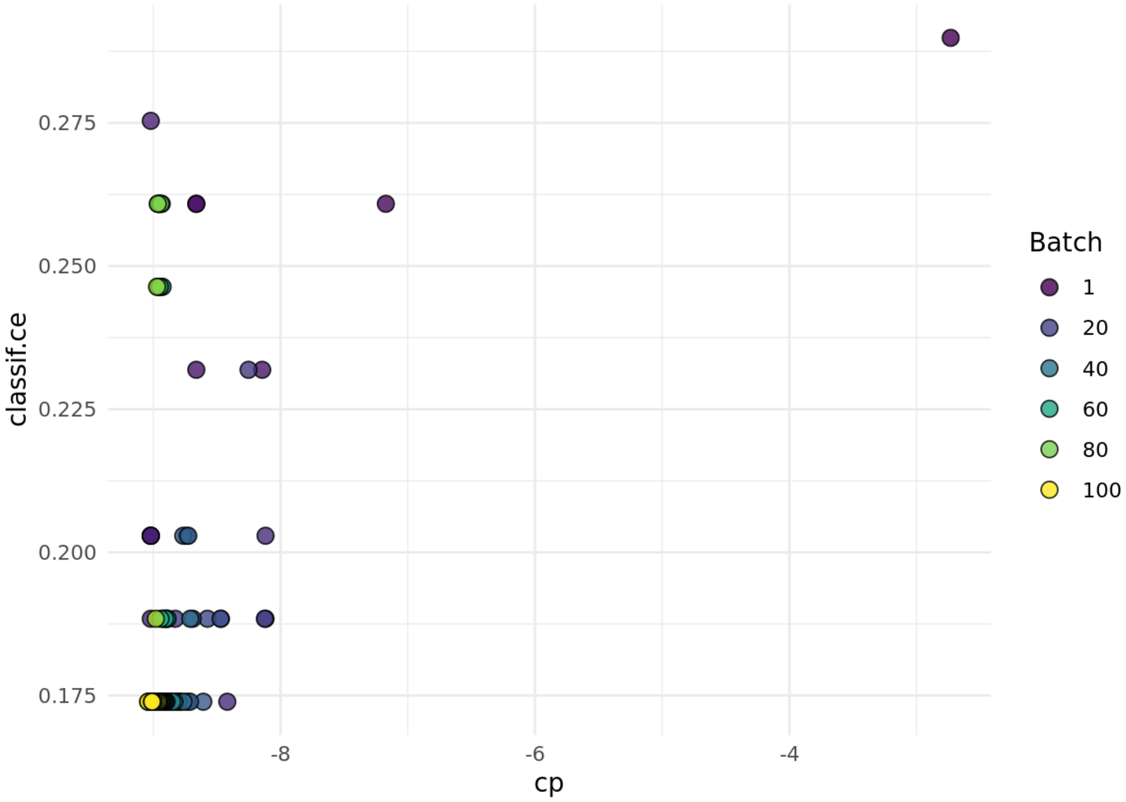
`"parameter"` 图展示了每组超参数设置的性能：

```r
autoplot(instance, type = "parameter", cols_x = c("cp", "minsplit"))
```



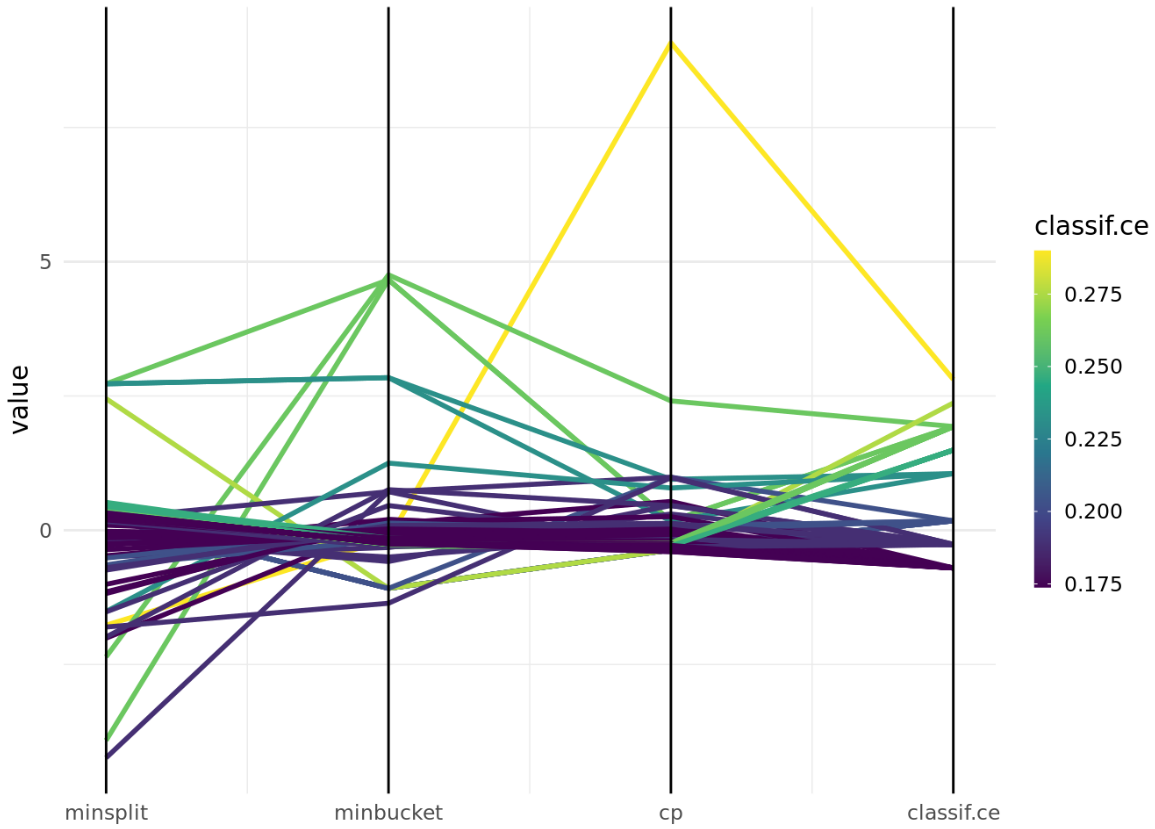
`"marginal"`图展示了不同超参数值的性能，颜色表示批次：

```
autoplot(instance, type = "marginal", cols_x = "cp")
```



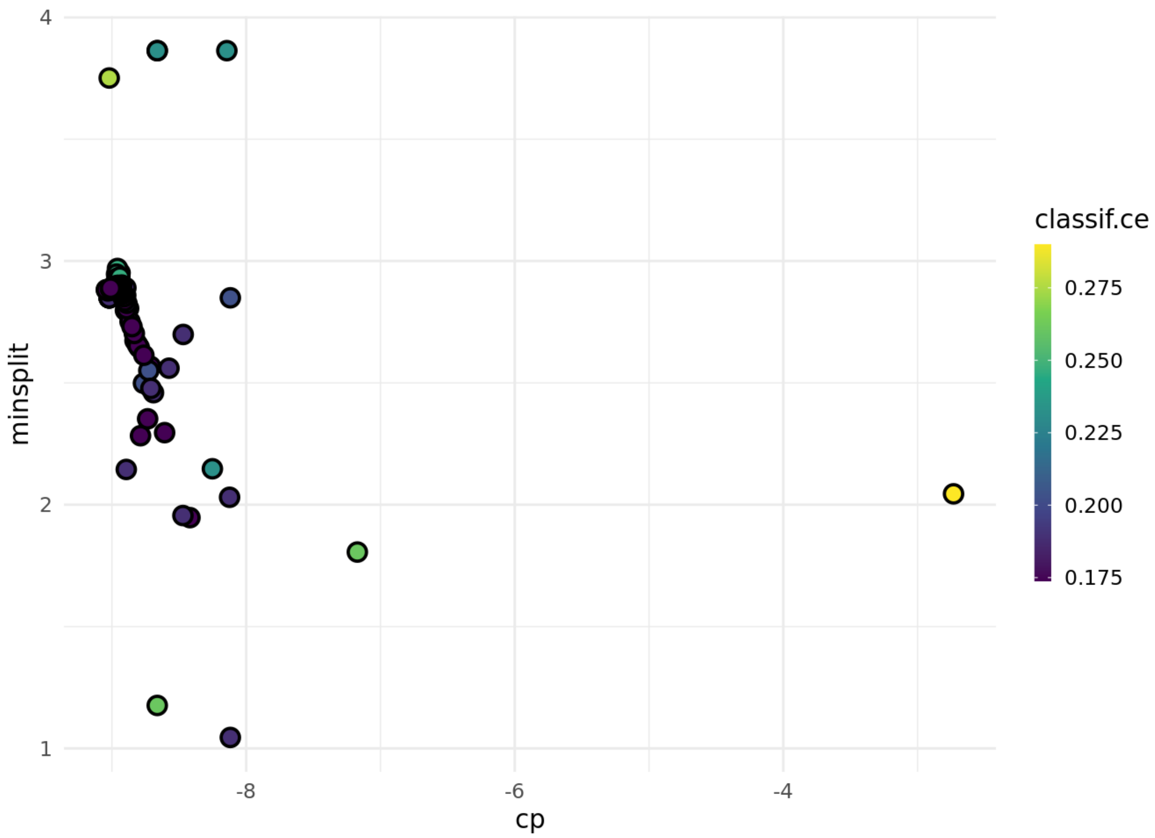
`"parallel"`图可视化超参数关系图。

```
autoplot(instance, type = "parallel")
```



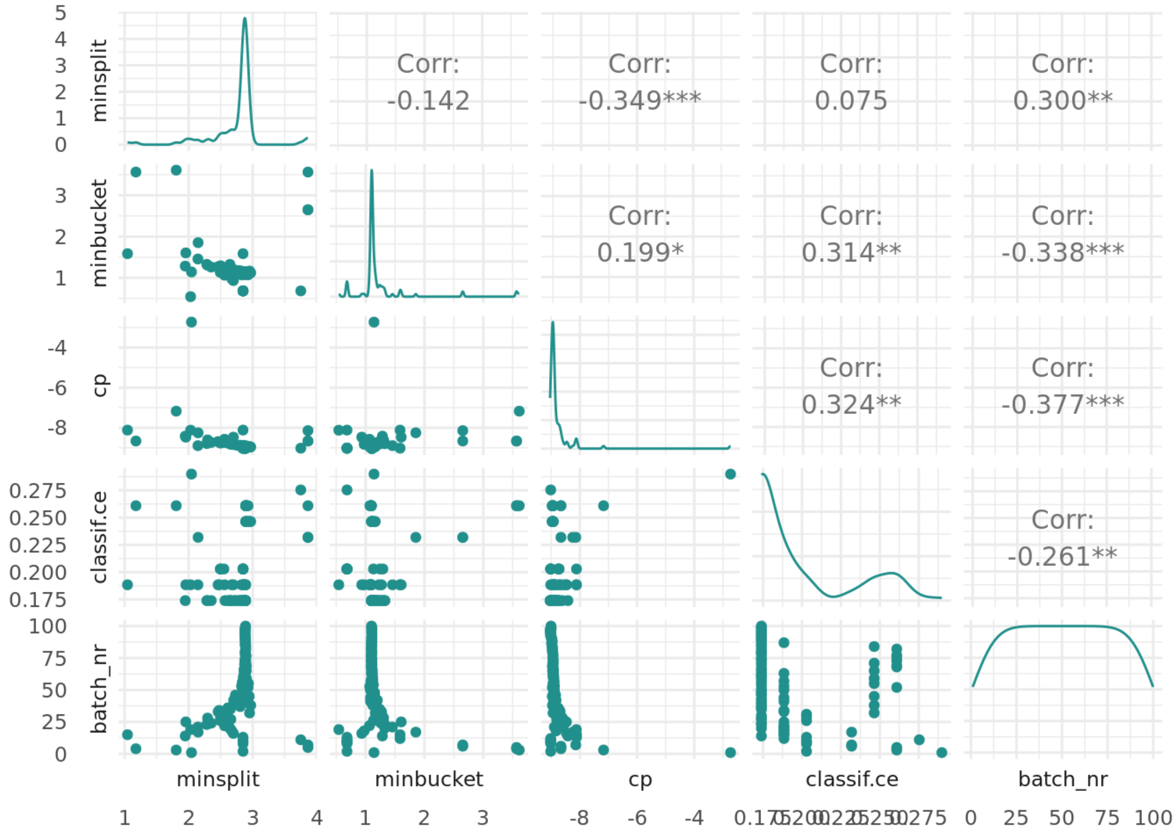
我们将 `"cp"` 与 `"minsplit"` 作对比，并根据性能给各点着色：

```
autoplot(instance, type = "points", cols_x = c("cp", "minsplit"))
```



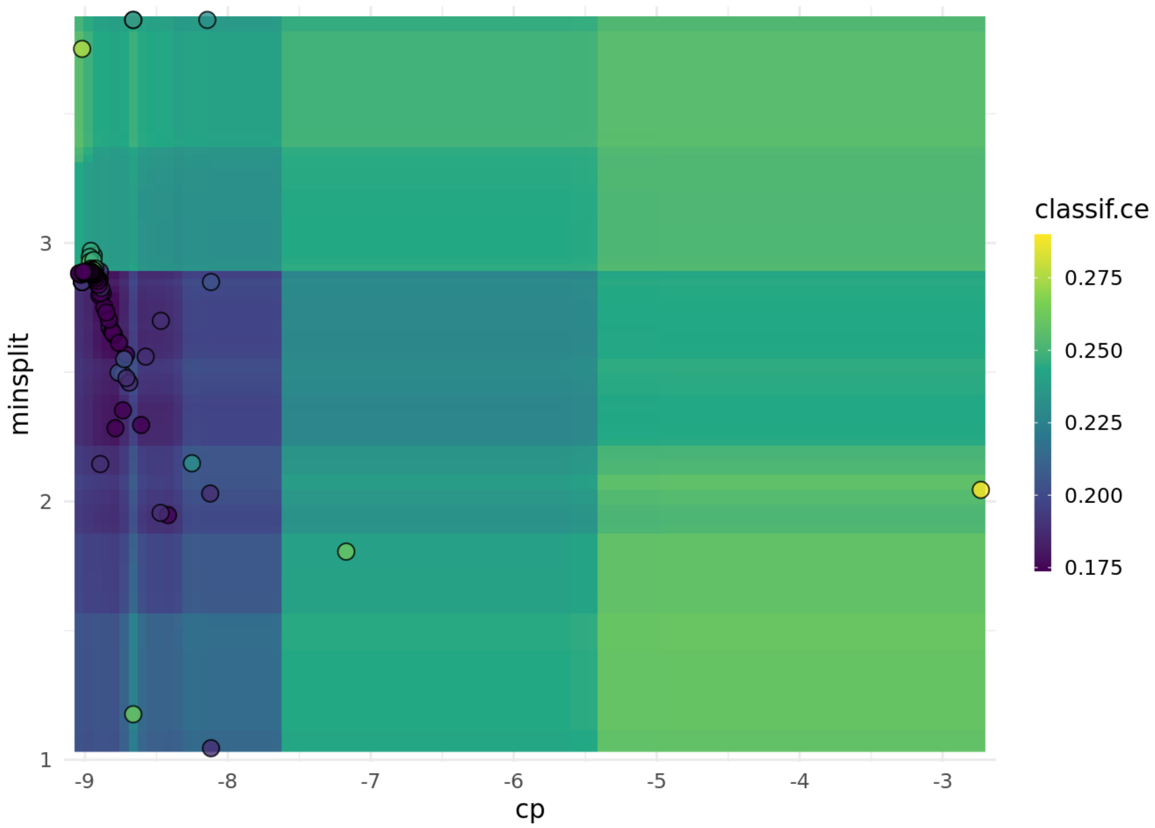
接下来，我们把所有的超参数都画在一起。

```
autoplot(instance, type = "pairs")
```



我们绘制了两个超参数的性能曲面。该表面是用一个学习器插值的。

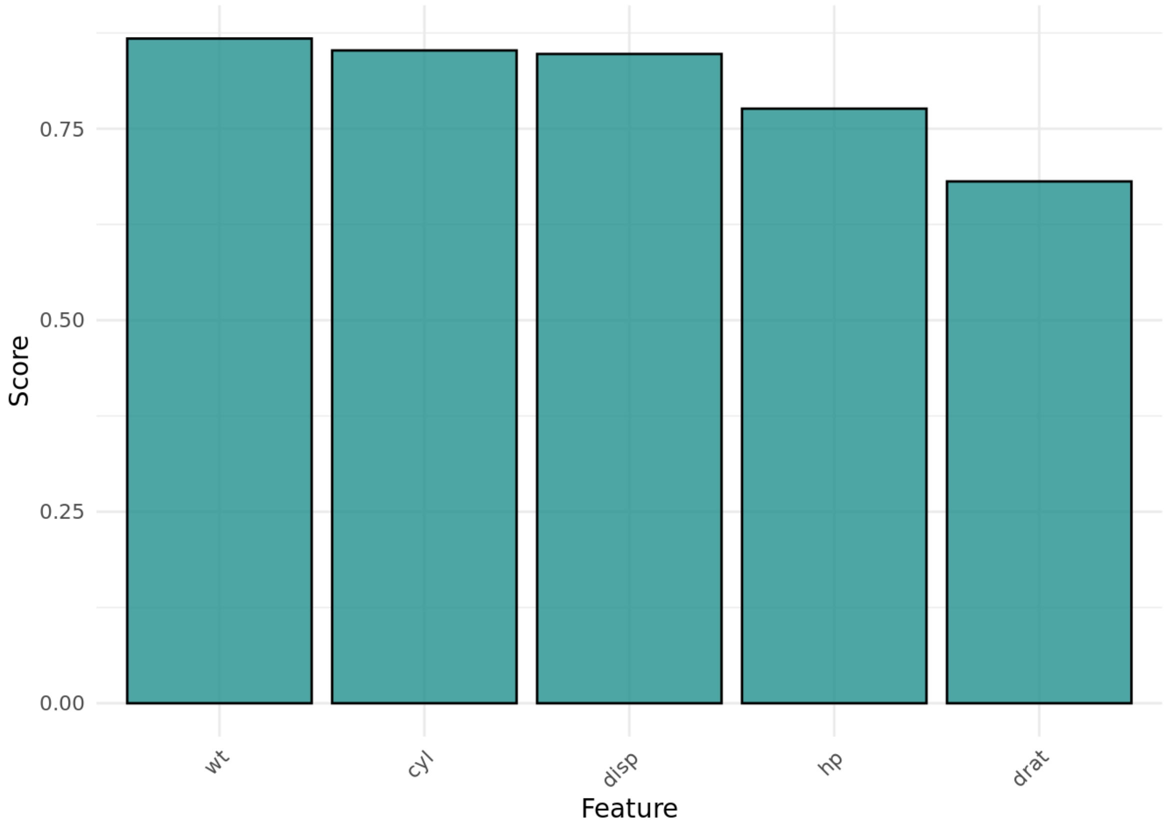
```
autoplot(instance, type = "surface", cols_x = c("cp", "minsplit"),
         learner = mlr3::lrn("regr.ranger"))
```



过滤器

我们绘制了MTCars任务的过滤分数。

```
library(mlr3filters)
task = tsk("mtcars")
f = flt("correlation")
f$calculate(task)
autoplot(f, n = 5)
```



结论

mlr3viz包汇集了 mlr3 生态系统的可视化功能。所有的图都是用 `autoplot()` 函数绘制的，外观可以用 `theme` 参数来定制。如果你需要高度定制一个图，例如用于出版，我们鼓励你查看我们在GitHub上的代码。该代码应该很容易适应你的需要。我们也期待着新的可视化方案。你可以在GitHub上的问题中提出新的图形。