

levelSets Example: Classifiers

Richard Raubertas

03 February 2026

Contents

1	Introduction	1
2	Problem setup	2
2.1	Develop the random forest classifier	2
2.2	Specify the response function and input space	3
2.3	Search region and input scaling	4
2.4	Define the level set of interest	4
3	Profiling the response function along some rays	4
4	Identifying the high-probability Adelie region of feature space	5
5	Slicing the level set	7
	References	10

1 Introduction

A classifier is a rule that uses one or more features of an item to assign the item to one of a set of K possible classes or groups. A classifier effectively partitions the space of possible feature vectors x into regions, where each region is associated with one class: an item whose feature vector belongs to a region associated with class k is assigned to that class. We may want to understand and visualize those regions of feature space.

The output of many classifiers takes the form of a vector $p = (p_1, p_2, \dots, p_K)$ where p_k is the classifier's reported probability that the item belongs to class k . (Less refined classifiers may discretize those probabilities to 1 for a single class and 0 for all the others.) If we focus on a single class k of interest, its regions can be interpreted as a level set, namely the set of x vectors in d -dimensional feature space for which p_k is greater than or equal to some threshold probability.

To illustrate this application of level sets, we use a random forest classifier [Breiman 2001, Liaw and Wiener 2002] to classify penguins into one of three species based on physical characteristics. The data set is from the `palmerpenguins` package [Horst et al 2020] and is described in `?palmerpenguins::penguins`. There are 333 penguins with complete data, from three species: 146 Adelie (44%), 68 Chinstrap (20%), and 119 Gentoo (36%). For each animal there are seven features, of which two (`island` and `year`) pertain to where and when the animal was measured, and the other five are attributes of the animal itself. Here the modeling focuses on those five, and on discriminating Adelie penguins from the other two species.

2 Problem setup

2.1 Develop the random forest classifier

```
penguin_data <- stats::na.omit(as.data.frame(palmerpenguins::penguins))
# Shorter names for feature (input) space dimensions:
names(penguin_data) <- sub("_length_mm", "_len", names(penguin_data))
names(penguin_data) <- sub("_depth_mm", "_dep", names(penguin_data))
names(penguin_data) <- sub("body_mass_g", "weight", names(penguin_data))
penguin_data$sex <- ifelse(penguin_data$sex == "female", 1, 2) # now numeric
# Features to use for modeling:
pvars <- c("bill_len", "bill_dep", "flipper_len", "weight", "sex")

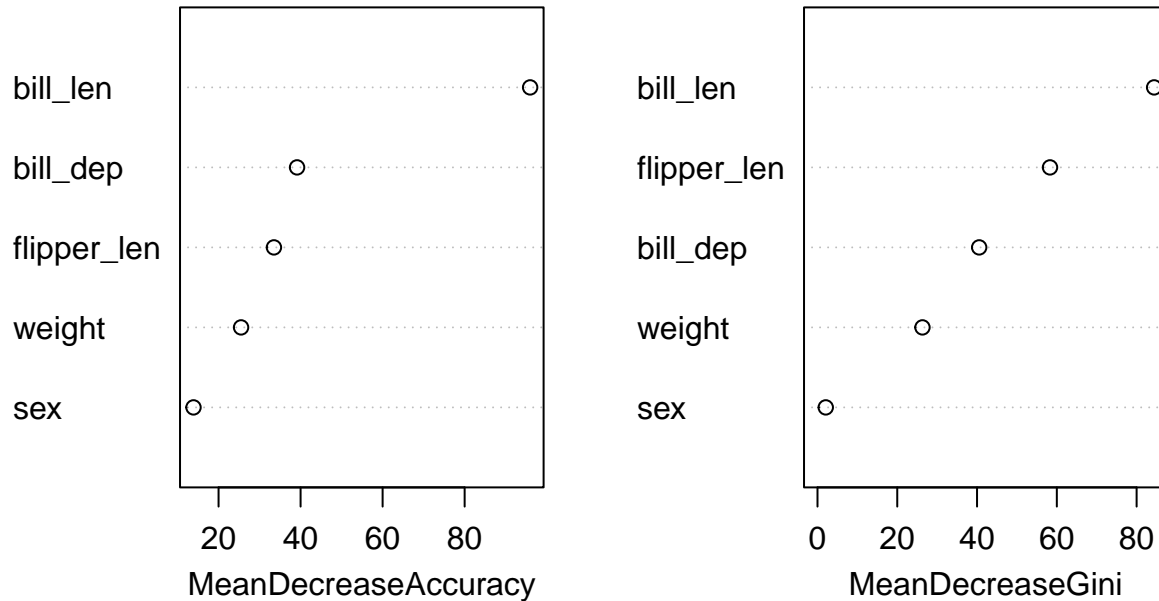
set.seed(1)
rf1 <- randomForest::randomForest(penguin_data[, pvars],
                                   y=penguin_data[, "species"],
                                   importance=TRUE)

print(rf1)

##
## Call:
## randomForest(x = penguin_data[, pvars], y = penguin_data[, "species"],      importance = TRUE)
##               Type of random forest: classification
##               Number of trees: 500
## No. of variables tried at each split: 2
##
##               OOB estimate of  error rate: 1.8%
## Confusion matrix:
##               Adelie Chinstrap Gentoo class.error
## Adelie         143          3      0 0.02054795
## Chinstrap        3         65      0 0.04411765
## Gentoo           0          0     119 0.00000000

randomForest::varImpPlot(rf1)
```

rf1



The classifier appears to be very accurate, since the estimated (OOB, “out-of-bag”) error rate is only 1.8%. `bill_len` is the most important predictor, `sex` the least.

2.2 Specify the response function and input space

The response function in this case is essentially a wrapper for the `predict` method for `randomForest` objects. Since the class (species) of interest is Adelie, we are interested in the level set where the reported probability of being that species is high.

```
respfn <- function(x, model) {
  if (nrow(x) > 0) {
    predprobs <- predict(model, newdata=x, type="prob") # 3-column matrix
                                                         # of class probabilities
    unname(predprobs[, "Adelie"])
  } else numeric(0) # 'predict.randomForest' does not like empty 'newdata'
}
```

We don't want to extrapolate the classifier model beyond the range of the data. So set the feasible region of input space to be the bounding box of the input data. Also, random forest models are inherently discontinuous functions of the inputs, so turn off attempts to use derivatives.

```
feasbnds <- apply(data.matrix(penguin_data[, pvars]), 2, range)
library(levelSets)
fobj <- fnObj(pvars, respfn=respfn, feasbnds=feasbnds, model=rf1,
             derivmethod="none")
d <- inpDim(fobj) # 5
```

2.3 Search region and input scaling

The level set search region is set to be a slightly expanded version of the feasible region. (Expansion is recommended because it allows the search to clearly identify points where the level set touches the boundary of the feasible region. This avoids ambiguous “censored” boundary points.) Also set the scaling of inputs to reflect the differing ranges of the features.

```
rect0 <- boundingRect(feasbnds, expand=1.1, spec=fobj)
sc10 <- inpScale(feasbnds[2, ] - feasbnds[1, ], spec=fobj)
```

2.4 Define the level set of interest

We want to identify regions of input space where the reported probability of being an Adelie penguin is high. (Note that their proportion in the training data is 0.44.)

```
thresh <- 0.75
```

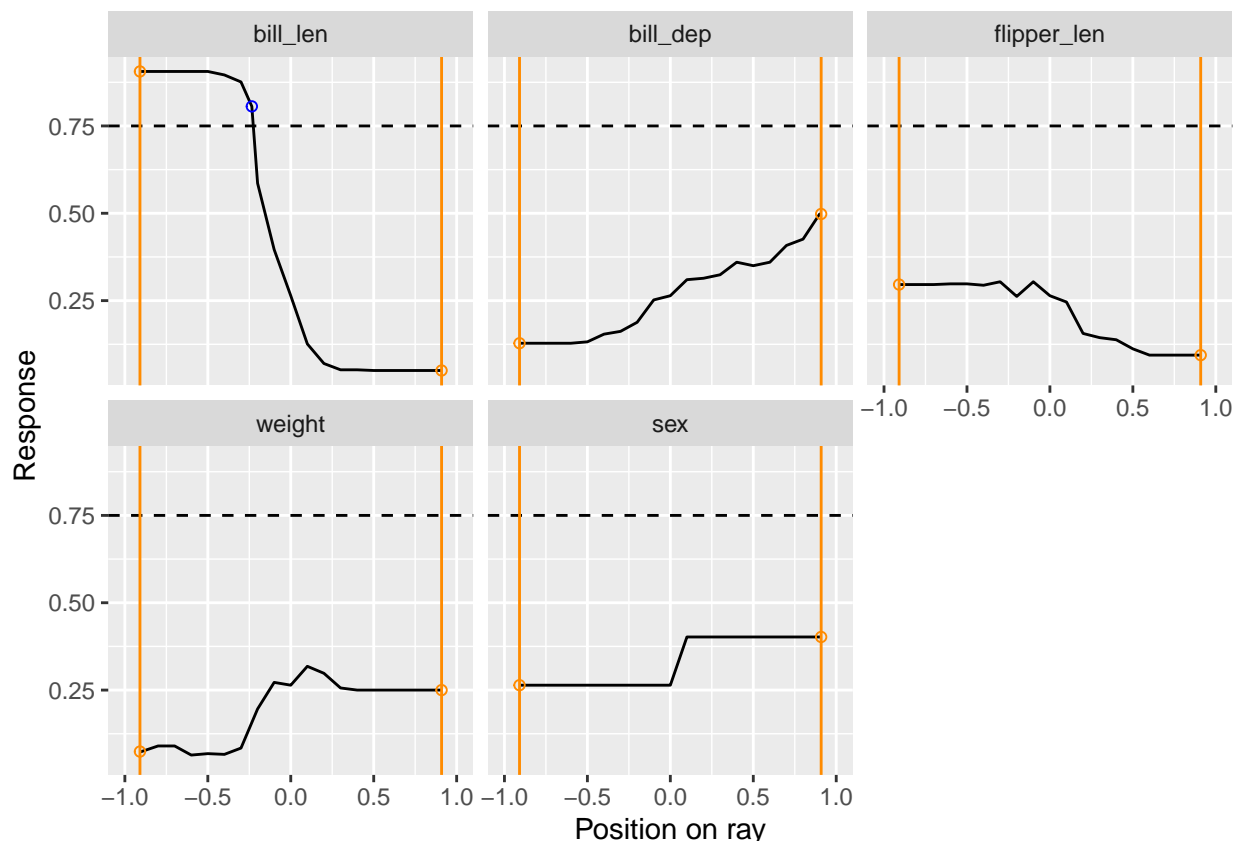
3 Profiling the response function along some rays

First we examine profiles of the reported Adelie probability along rays parallel to each coordinate axis. The ray origin (position 0) is the reference point of ‘rect0’, namely its center. Position 1 along each ray is its intersection with the boundary of ‘rect0’.

```
rays0 <- axisRays(fobj, rect=rect0, scale=sc10)
profs <- respProfiles(fobj, rays=rays0, posns=c(-10:10)/10,
                     lsetthresh=thresh)
summary(profs)
```

```
## Response function profiled along 5 rays in a 5-dimensional input space
## Level set threshold: 0.75
## Number of profile points over all rays: 116
##   In feasible region= 106 (on its bdry= 10)
##   In level set= 9 (on its bdry= 1)
## Range of response values over all rays: 0.05, 0.906
```

```
plot(profs, groupBy=inpNames(fobj)) # 'groupBy' puts each ray in a separate panel
```



Short `bill_len` is strongly associated with a high probability of species being Adelie.

4 Identifying the high-probability Adelie region of feature space

We use the `bdrySearch()` function, and select the initial ray origin from among the profile points in `profs` that belong to the level set.

```
profinfo <- respInfo(profs) # data frame, including column 'lset'
print(subset(profinfo, lset))
```

##	resp	feas	lset	line	posn	feasbdry	lsetbdry	deriv	trend
## 2	0.906	TRUE	TRUE	1	-0.9090897	TRUE	FALSE	NA	<NA>
## 3	0.906	TRUE	TRUE	1	-0.9000000	FALSE	FALSE	NA	<NA>
## 4	0.906	TRUE	TRUE	1	-0.8000000	FALSE	FALSE	NA	<NA>
## 5	0.906	TRUE	TRUE	1	-0.7000000	FALSE	FALSE	NA	<NA>
## 6	0.906	TRUE	TRUE	1	-0.6000000	FALSE	FALSE	NA	<NA>
## 7	0.906	TRUE	TRUE	1	-0.5000000	FALSE	FALSE	NA	<NA>
## 8	0.896	TRUE	TRUE	1	-0.4000000	FALSE	FALSE	NA	<NA>
## 9	0.876	TRUE	TRUE	1	-0.3000000	FALSE	FALSE	NA	<NA>
## 10	0.806	TRUE	TRUE	1	-0.2347122	FALSE	TRUE	NA	<NA>

All nine level set points are from the `bill_len` profile (`line` = 1). We choose point 8 in `profs` because it is reasonably close to the center of `rect` (`posn` not too far from the ray origin at position 0) but not actually on the boundary of the level set.

```
srch1 <- bdrySearch(fobj, lsetthresh=thresh, rect=rect0, scale=sc10,
  initOrigin=ptCoord(profs)[8, ])
```

```
segs1 <- lsetSegs(srch1)
```

To gain some confidence that the line segments in `segs1` do in fact lie entirely in the level set, and that the level set does not have multiple parts, evaluate the response function at a number of points within each segment:

```
chk <- lsetSegsCheck(segs1, fnobj=fobj, posns=(1:4)/5)
table(chk$validIn) # all TRUE
```

```
##
## TRUE
## 125
```

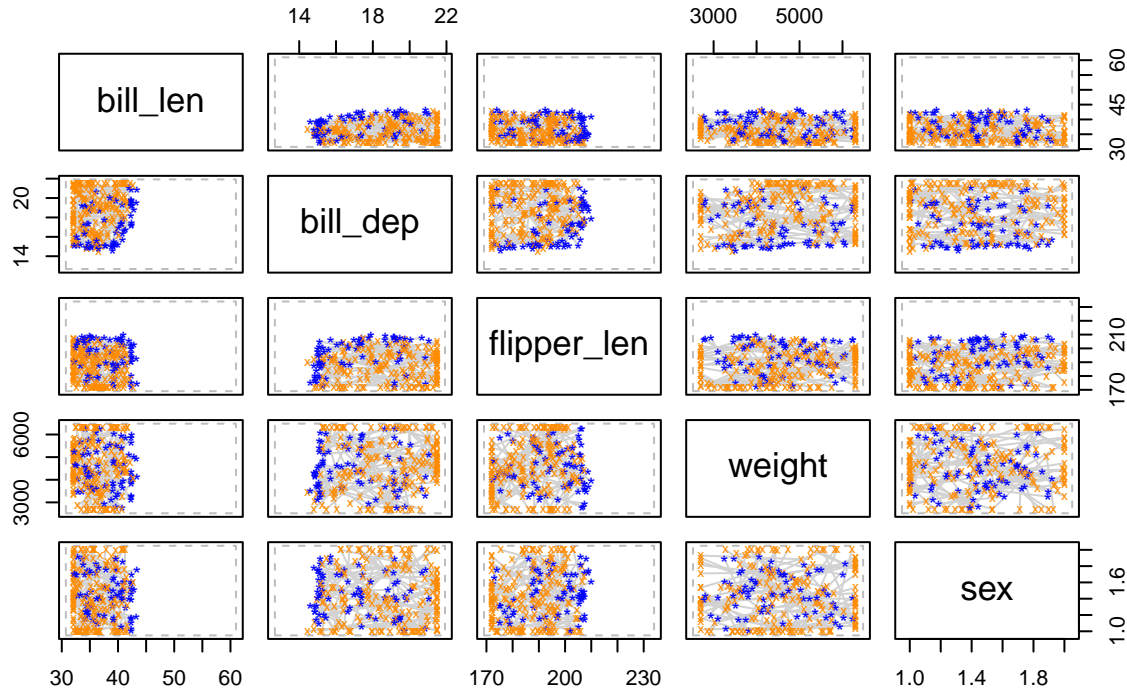
Summarize and plot the segments:

```
summary(segs1)
```

```
## Level set segments along 125 lines in a 5-dimensional input space
## Level set threshold: 0.75
## Number of segments over all lines: 125
## Segment endpoints by type:
##      nfeasbdry nlsetbdry      ncens
##           179          71          0
## Number of lines with 0, 1, >1 segments: 0, 125, 0
## Bounding box for level set segments:
##      bill_len bill_dep flipper_len weight sex
## [1,] 32.10000 14.44655   172.0000   2700   1
## [2,] 43.38199 21.50000   209.9684   6300   2
## Box volume= 10877075, # of intersecting lines= 125
```

```
pairs(segs1, rect=rect0, main=paste0("Segments in feature space with ",
                                     "Pr(Adelie) >= ", thresh))
```

Segments in feature space with $\text{Pr}(\text{Adelie}) \geq 0.75$



A level set segment was found on each of the 125 rays. Some things to note:

- Most segment endpoints (179 of 250) are on the boundary of the feasible region.
- The level set is approximately rectangular when projected onto any pair of `bill_len`, `bill_dep`, and `flipper_len`.
- There are no obvious patterns in the level set with respect to `weight`.
- This package requires all inputs to the response function to be numeric, so the feature `sex` was coded as 1 (female) or 2 (male) when developing the classifier `rf1`. There are segments that end at values between 1 and 2, which doesn't make sense for classifying penguins. A way to address this is to slice the level set analysis, as shown in the next section.

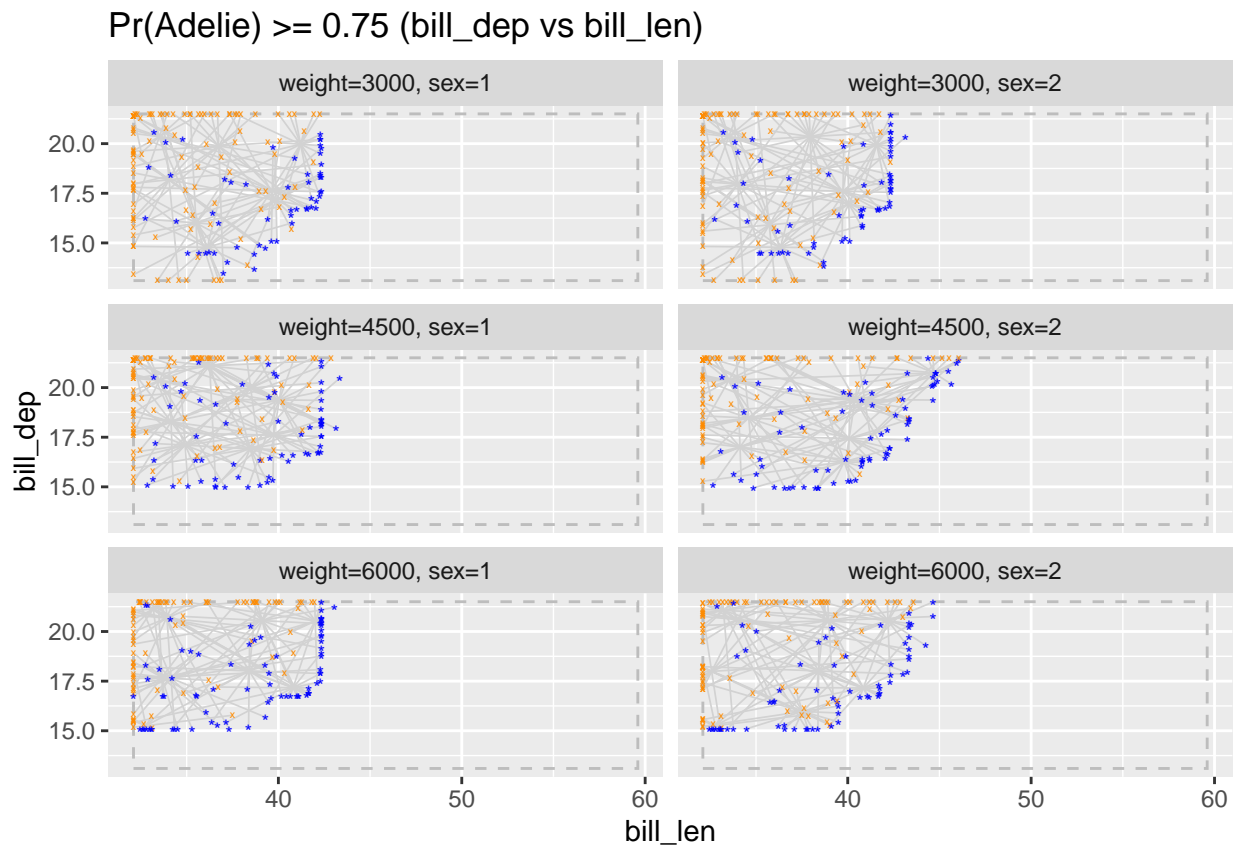
5 Slicing the level set

Since `sex` is a categorical feature it is logical to examine the level set separately for females and males. We will also slice `weight` at three values (low/medium/high = 3000/4500/6000 grams). And the boundary search will build on the results already found so far (`currentBdry=srch1`). Control option `axisDegree` is set to 2, rather than the default of 1, in order to increase the number of rays per origin at each search step. (See `?srchControl`.)

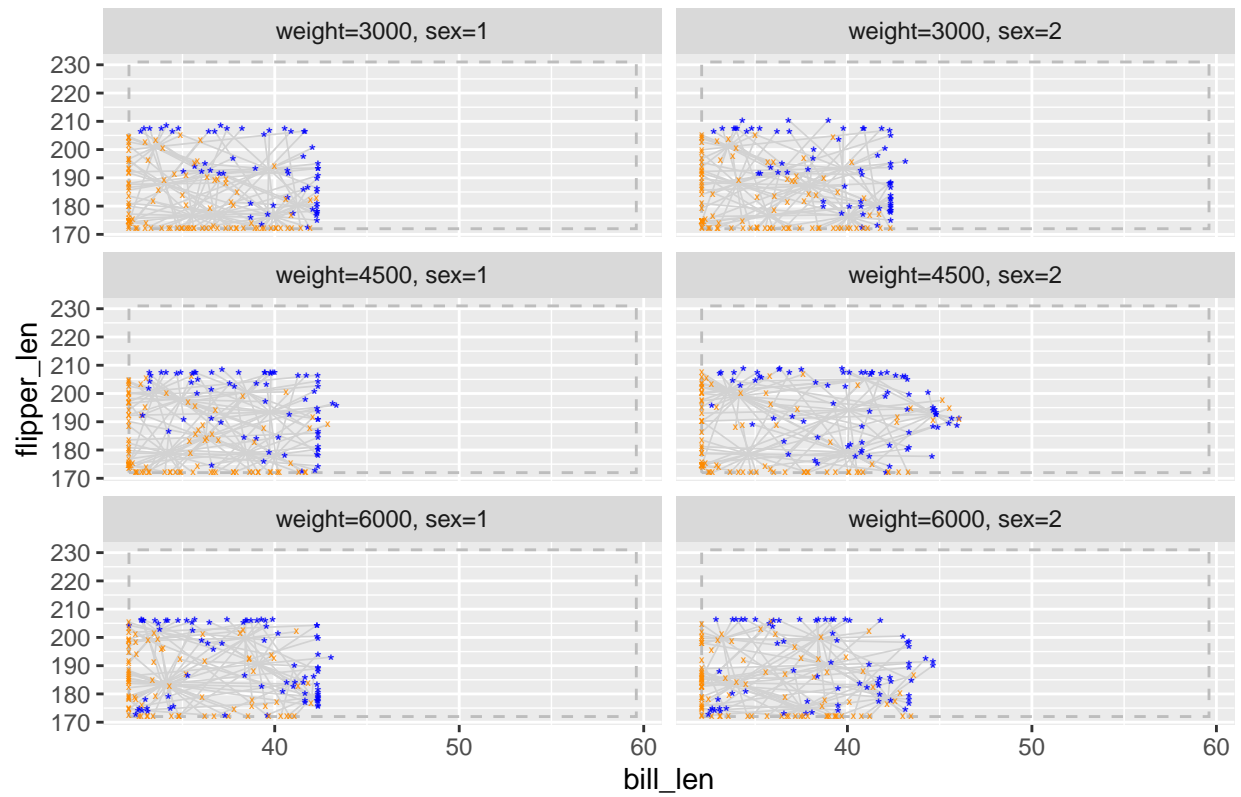
```
slices2 <- sliceMat(list(sex=c(1, 2), weight=c(3000, 4500, 6000)),
                    spec=fobj) # 2 * 3 = 6 slices
srch2 <- slicedBdrySearch(fobj, lsetthresh=thresh, rect=rect0, scale=scl0,
                        slices=slices2, currentBdry=srch1,
                        control=list(axisDegree=2))
```

The result `srch2` is a list with one component per slice. Each component is itself a list like the one returned by `bdrySearch()`. The function `plotSegsList()` can plot the results, putting each slice into a separate panel (package `ggplot2` is required).

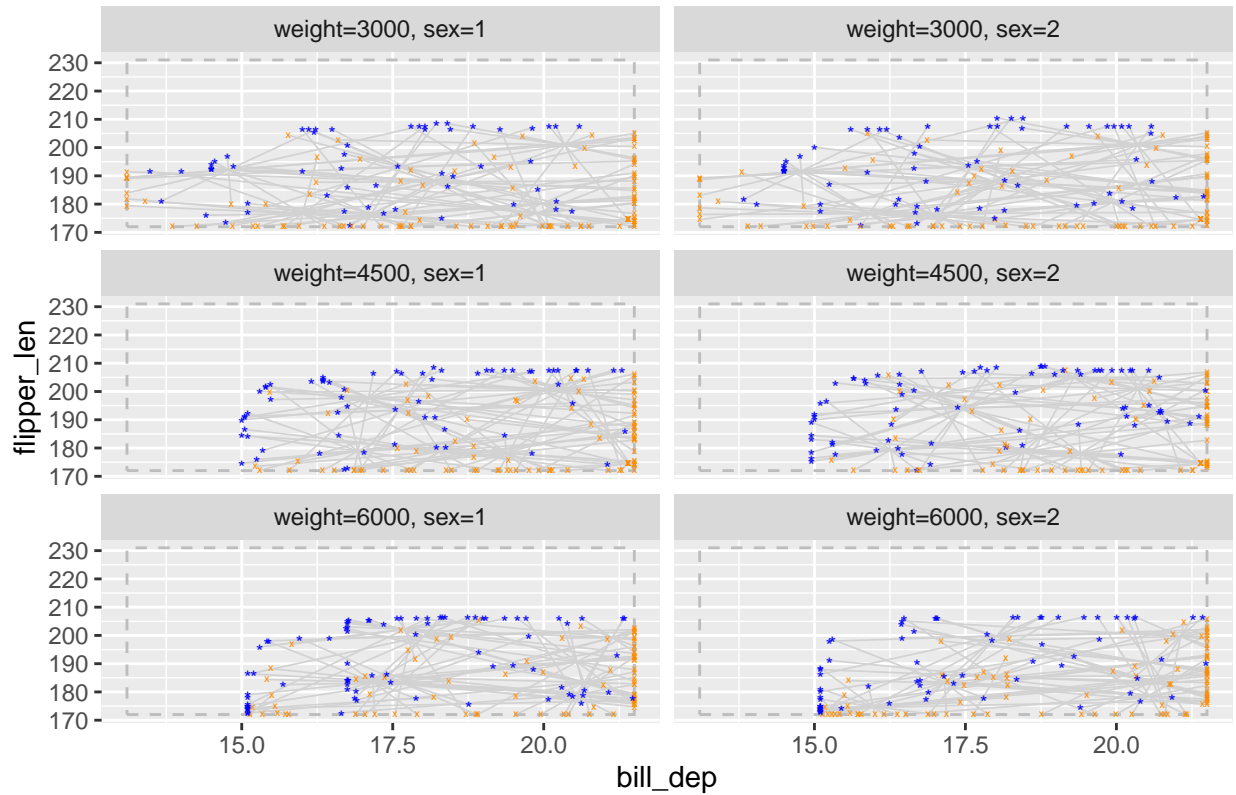
```
if (requireNamespace("ggplot2", quietly=TRUE)) {
  plt <- plotSegsList(srch2, dims=c("bill_len", "bill_dep"), rect=feasbnds,
                    wrap=list(ncol=2))
  print(plt + ggplot2::labs(title=paste0("Pr(Adelie) >= ", thresh,
                                         " (bill_dep vs bill_len)")))
  plt <- plotSegsList(srch2, dims=c("bill_len", "flipper_len"), rect=feasbnds,
                    wrap=list(ncol=2))
  print(plt + ggplot2::labs(title=paste0("Pr(Adelie) >= ", thresh,
                                         " (flipper_len vs bill_len)")))
  plt <- plotSegsList(srch2, dims=c("bill_dep", "flipper_len"), rect=feasbnds,
                    wrap=list(ncol=2))
  print(plt + ggplot2::labs(title=paste0("Pr(Adelie) >= ", thresh,
                                         " (flipper_len vs bill_dep)")))
}
```



Pr(Adelie) ≥ 0.75 (flipper_len vs bill_len)



Pr(Adelie) ≥ 0.75 (flipper_len vs bill_dep)



The level set has corners, which is not surprising for a tree-based classifier. Sex has only small effect on the level set, as expected from the variable importance measures shown earlier. Weight has a noticeable effect, with the level set extending to small values of `bill_dep` for animals with the smallest weights.

References

Breiman, L (2001). Random forests. *Machine Learning* 45(1), 5-32.

Horst AM, Hill AP, Gorman KB (2020). palmerpenguins: Palmer Archipelago (Antarctica) penguin data. R package version 0.1.0. <https://allisonhorst.github.io/palmerpenguins/>. doi:10.5281/zenodo.3960218.

Liaw A, Wiener M (2002). Classification and regression by randomForest. *R News*, 2(3), 18-22. <https://CRAN.R-project.org/doc/Rnews/>.