levelSets Example: Confidence Regions

Richard Raubertas

17 February 2026

Contents

1 Introduction

Some types of confidence regions for parameters of statistical models are in fact level sets: they are regions of parameter space where an *inference function* (Kim and Lindsay, 2011) is at or above a threshold value. This vignette illustrates the use of the `levelSets` package to map and visualize such confidence regions.

Confidence regions can be obtained by inverting hypothesis tests: a $100(1-\alpha)\%$ confidence region for d-dimensional parameter θ consists of all points θ_0 such that the null hypothesis $H_0 : \theta = \theta_0$ is not rejected at significance level α. A common type of test is the likelihood ratio (LR) test, which uses the log-likelihood as the inference function. Let $l(\theta; y)$ denote the log-likelihood of the model given the data y, evaluated at parameter value θ, and let $\hat{\theta}$ denote the maximum likelihood estimate (MLE) of θ. The LR test rejects H_0 if test statistic $T = 2(l(\hat{\theta}; y) - l(\theta_0; y))$ is greater than the critical value $\chi^2(1-\alpha, d)$, the $1-\alpha$ quantile of a chi-square distribution with d degrees of freedom. Therefore an LR confidence region for the parameter consists of all θ for which $l(\theta; y) \geq l(\hat{\theta}; y) - \chi^2(1-\alpha, d)/2$. In other words, it is a level set of the log-likelihood function.[1]

Kim and Lindsay (2011) proposed two methods to visualize such confidence regions. Both involve generating a set of rays in the d-dimensional parameter space that start from a single point (such as the MLE) and have random directions. The first method adjusts the rays so that their endpoints[2] follow a so-called *confidence distribution*, defined so that for any α the probability is $1-\alpha$ that an endpoint lies within the $1-\alpha$ confidence region. Their second method, called boundary sampling, finds the intersections of rays with the *boundary* of the $1-\alpha$ confidence region. The `bdryFromRays()` function of this package provides an implementation of this second method.

[1] Replacing the log-likelihood function with a posterior density shows that one form of Bayesian credible region can also be viewed as a level set.

[2] Points with position = 1 in the terminology of this package.

2 Example: Circuit failure data

As an example, consider the limited failure population (LFP) model of failure times for integrated circuits, discussed in section 18.4 of Meeker, Hahn, and Escobar (2017). The model assumes that a fraction `dfrac` of circuits are defective, with failure times having a Weibull distribution with parameters `shape` and `scale`. The remaining fraction `1 - dfrac` are assumed to have essentially infinite failure times. (This is a form of "cure model" sometimes applied to patient survival times in biomedical settings.) LFP is thus a parametric model with a 3-dimensional parameter space.

Meeker, Hahn, and Escobar provide data from a study that measured failure times for 4156 circuits that were tested for 1370 hours at elevated temperature and humidity. 28 of the circuits failed at various times during the test; the failure time of the remaining circuits was censored at 1370 hours.

Here we examine the 3-dimensional LR-based 95% confidence region for the full set of model parameters (`dfrac`, `shape`, `scale`). Therefore the response function is the log-likelihood, and the confidence region is the level set consisting of parameter vectors whose log-likelihood is no more than `qchisq(0.95, df=3) / 2 = 3.907` below its maximum.

The log-likelihood response function, maximum likelihood estimate (MLE) of the parameters, and variance-covariance matrix of the MLE are already defined in list `circuitFailure_3dEx`:

```
library(levelSets)
Ex <- circuitFailure_3dEx
fobj <- Ex$fobj  # 'fnObj' object with the response function
theta_mle <- Ex$theta_mle
theta_vcov <- Ex$theta_vcov
thresh <- Ex$thresh  # threshold value of log-likelihood that defines an
                     # asymptotic 95% conf region
```

Note that the confidence region is defined in terms of simple differences between the maximum log-likelihood and the log-likelihood at other points in input space, independent of their magnitudes. Therefore, set the numerical tolerance for response function values to use only an absolute tolerance, not a relative tolerance:

```
fobj <- update(fobj, resptol=c(0, 1e-3))
```

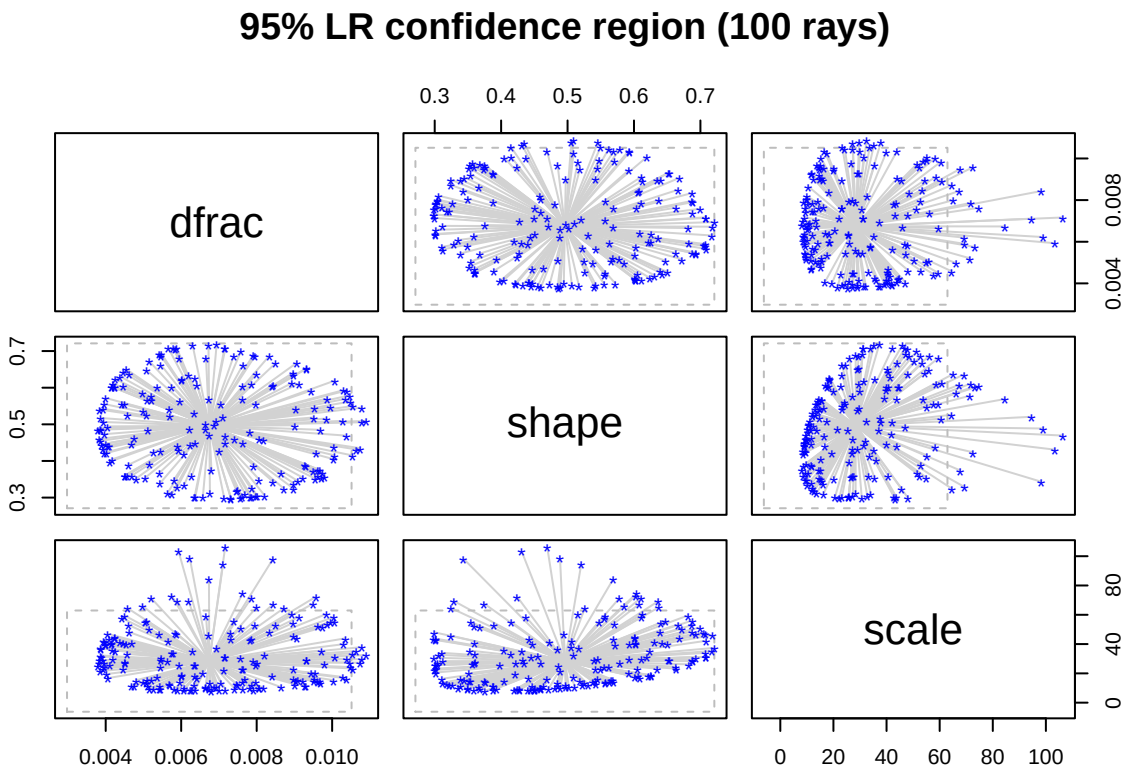
A natural choice for a ray origin is the MLE. The initial search region is set to be ± 3 standard errors around the MLE. Input space dimensions are scaled by the asymptotic variance-covariance matrix from the MLE fit:

```
rect1 <- inpRect(rbind(theta_mle - 3*sqrt(diag(theta_vcov)),
                       theta_mle + 3*sqrt(diag(theta_vcov))),
                 spec=fobj)
scl1 <- inpScale(theta_vcov, spec=fobj)
set.seed(1)
rays1 <- randomRays(100, rect=rect1, scale=scl1)
segs1 <- bdryFromRays(fobj, rays=rays1, lsetthresh=thresh)
summary(segs1)
```

```
## Level set segments along 100 rays in a 3-dimensional input space
## Level set threshold: -296.9403
## Number of segments over all rays: 100
##   Segment endpoints by type:
##     nfeasbdry nlsetbdry     ncens
##             0       200         0
##   Number of rays with 0, 1, >1 segments: 0, 100, 0
## Bounding box for level set segments:
##              dfrac     shape      scale
##   [1,] 0.003795821 0.2982295   8.176735
##   [2,] 0.010900705 0.7213245 106.437257
```

```
##   Box volume= 0.2953751, # of intersecting rays= 100
```

```
pairs(segs1, rect=rect1, main="95% LR confidence region (100 rays)")
```



The dashed gray box in each panel is the projection of the search region `rect1` to each 2D plane. The confidence region is clearly not centered at the MLE, is not ellipsoidal, and extends beyond three (asymptotic) standard errors from the MLE. This is especially true for the `scale` parameter, for which the upper extent of the confidence region is not well determined by these rays. The next sections consider two approaches to improving resolution of the confidence region.

3 Adaptive selection of rays

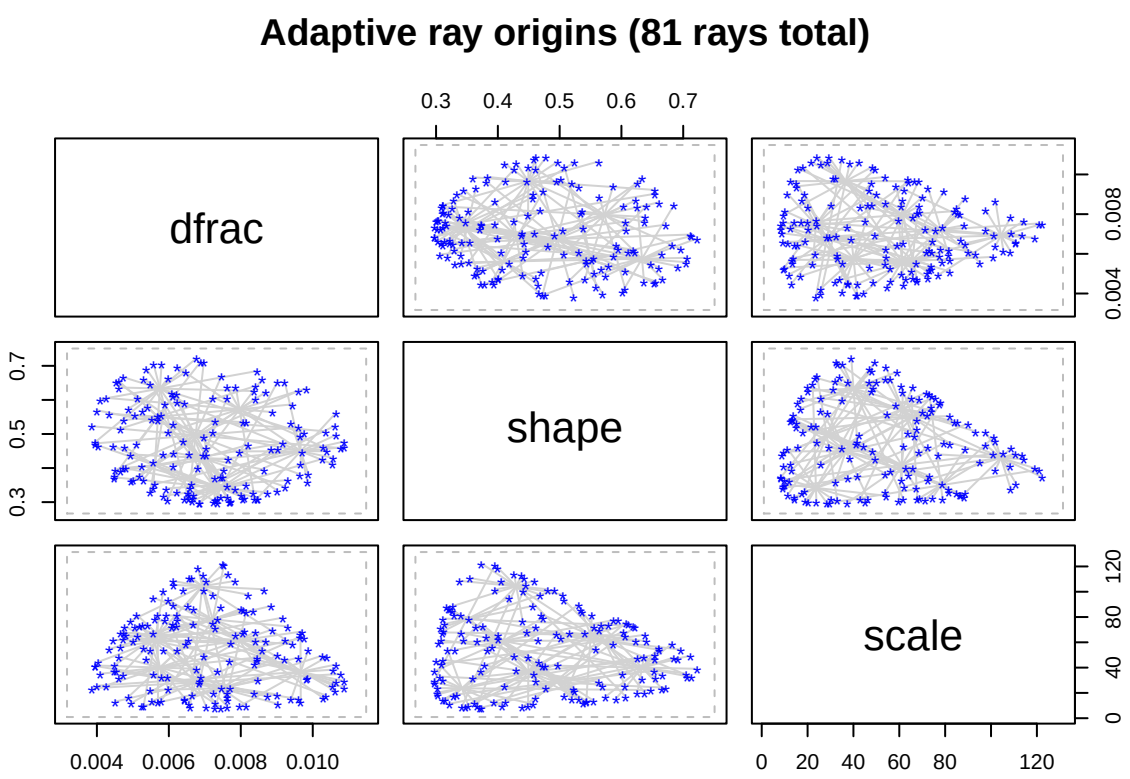
Function `bdrySearch()` searches for a level set boundary in a series of steps, where each step adaptively selects a new ray origin in an attempt to explore new portions of the boundary. The search region and input dimension scaling can also be updated at each step. (See `?srchControl` for the full set of control options.)

```
srch2 <- bdrySearch(fobj, lsetthresh=thresh, rect=rect1, scale=scl1,
                    initOrigin=theta_mle,
                    control=list(axisDegree=2, updateScale=2))
segs2 <- lsetSegs(srch2)
summary(segs2)
```

```
## Level set segments along 81 lines in a 3-dimensional input space
## Level set threshold: -296.9403
## Number of segments over all lines: 81
##   Segment endpoints by type:
##     nfeasbdry nlsetbdry     ncens
```

```
##              0       162         0
##   Number of lines with 0, 1, >1 segments: 0, 81, 0
## Bounding box for level set segments:
##               dfrac     shape      scale
##   [1,] 0.003861998 0.2975039   8.376975
##   [2,] 0.010881049 0.7226697 122.363945
##   Box volume= 0.3401667, # of intersecting lines= 81
```

```
pairs(segs2, rect=srch2$rect, main="Adaptive ray origins (81 rays total)")
```



Even with fewer rays (81 versus 100) we get a clearer picture of the size and shape of the confidence region.

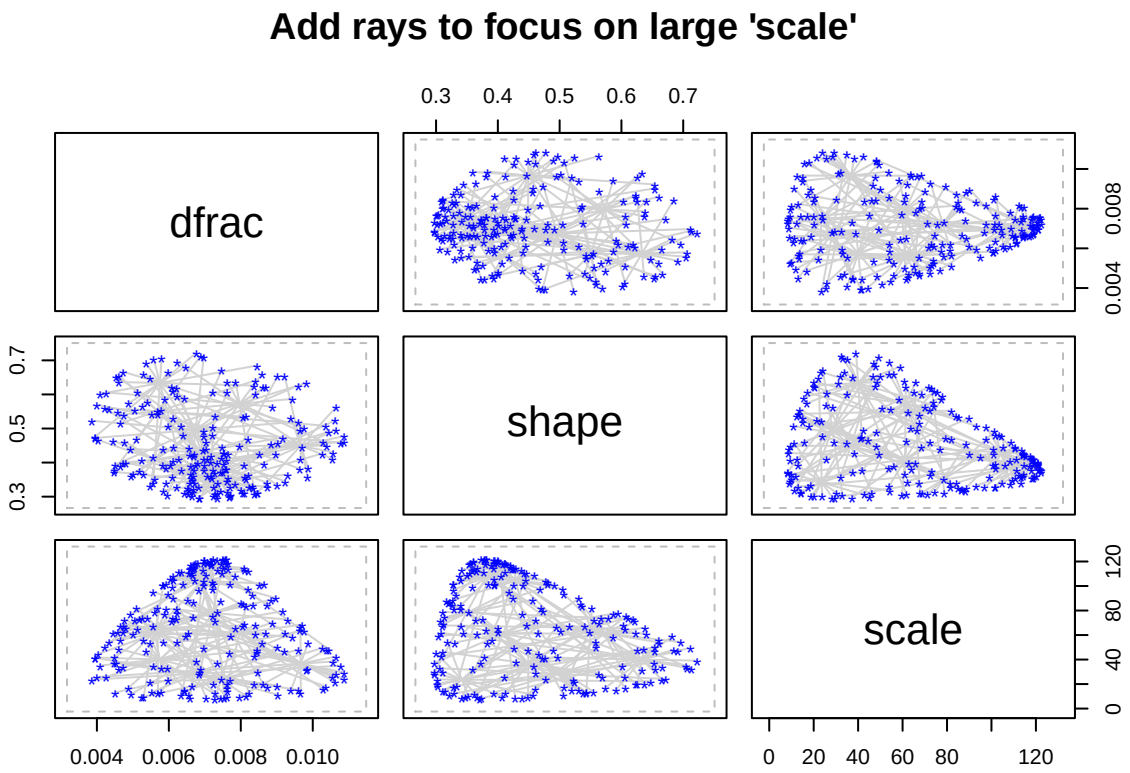
The `currentBdry` argument of `bdrySearch()` allows the boundary search to build on previous searches. For example, we can extend `srch2` to focus on boundary points with large values of `scale` by using control options `originCrit` and `srchDirection`:

```
srch2b <- bdrySearch(fobj, lsetthresh=thresh,
                     # Build on previous search results, including updated
                     # search region and scaling:
                     currentBdry=srch2, rect=srch2$rect, scale=srch2$scale,
                     control=list(axisDegree=2,
                                  originCrit="maxproj",
                                  srchDirection=c(0, 0, 1),
                                  maxSteps=3))
segs2b <- lsetSegs(srch2b)
summary(segs2b)  # Added 3 new origins, 27 new rays
```

```
## Level set segments along 108 lines in a 3-dimensional input space
```

```
## Level set threshold: -296.9403
## Number of segments over all lines: 108
##   Segment endpoints by type:
##     nfeasbdry nlsetbdry     ncens
##             0       216         0
##   Number of lines with 0, 1, >1 segments: 0, 108, 0
## Bounding box for level set segments:
##               dfrac     shape      scale
##   [1,] 0.003861998 0.2975039   8.376975
##   [2,] 0.010881049 0.7226697 122.619794
##   Box volume= 0.3409303, # of intersecting lines= 108
```

```
pairs(segs2b, rect=srch2b$rect, main="Add rays to focus on large 'scale'")
```



4 Slicing input space

A slice of input space is just a subset where specified inputs are held at fixed values. This reduces the number of dimensions that need to be explored for level set boundary points within a slice, simplifying visualization and interpretation. Functions `slicedBdryFromRays()` and `slicedBdrySearch()` apply the algorithms of `bdryFromRays()` and `bdrySearch()`, respectively, to find level set boundary points within slices.

For the circuit failure example, we consider six slices defined by six values of `scale`:

```
slices <- sliceMat(list(scale=c(10, 30, 50, 70, 90, 110)), spec=fobj)
```

To apply `slicedBdryFromRays()` we need to specify a search region that intersects all the slices, such as

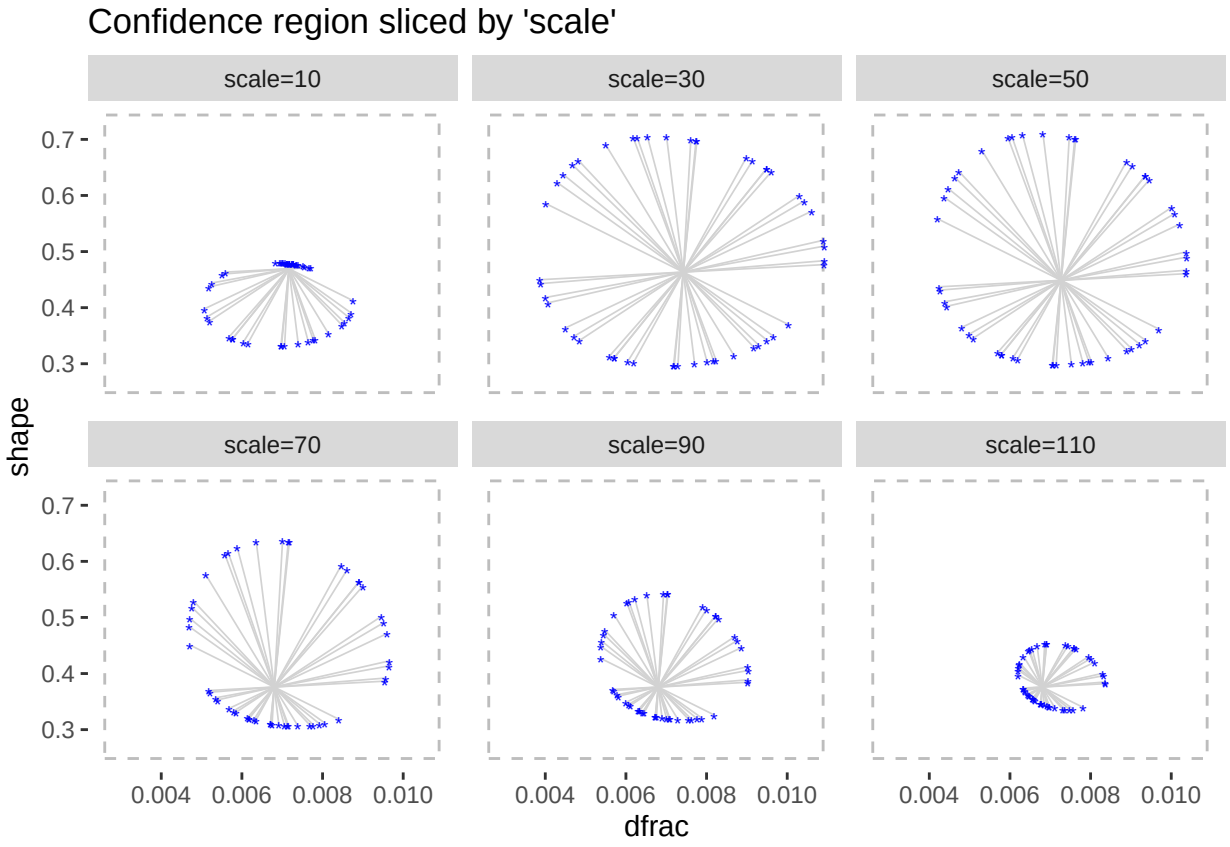
```
rect3 <- boundingRect(rect1, slices, expand=1.1)
```

and ray origins that lie within the slices. As always, it is also desirable that the origins lie within the level set, although such points may be difficult to find in general. One option is to ask the function to attempt to find origins based on an initial, unsliced search, via the `currentBdry` argument:

```
srch3 <- slicedBdryFromRays(fobj, lsetthresh=thresh, slices=slices,
                            rect=rect3, scale=scl1,
                            currentBdry=segs1)  # list with one component per slice
slsegs3 <- lapply(srch3, lsetSegs)  # list of 'lsetSegs' objects
# Number of level set segments found per slice:
sapply(slsegs3, length)
```

```
##  scale=10  scale=30  scale=50  scale=70  scale=90 scale=110
##        25        25        25        25        25        25
```

```
if (requireNamespace("ggplot2", quietly=TRUE)) {
  plt <- plotSegsList(slsegs3, dims=c("dfrac", "shape"), rect=rect3)
  print(plt + ggplot2::theme(panel.background=ggplot2::element_blank()) +
        ggplot2::labs(title="Confidence region sliced by 'scale'"))
}
```



In this case the function was able to find origins and level set segments for all the slices, although that is not guaranteed. The plots show that the confidence region is more egg-shaped than elliptical for any given `scale` value.

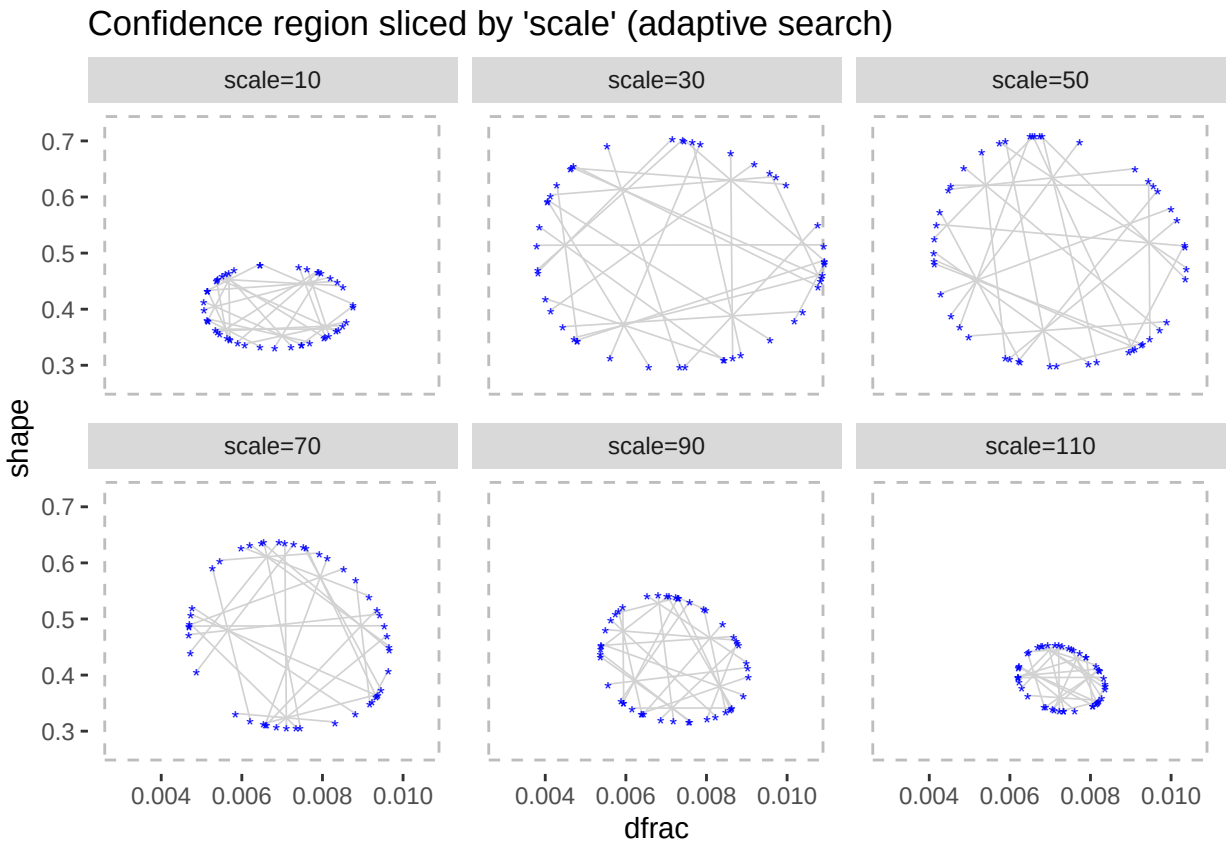
Slicing can also be combined with the adaptive ray origin approach:

```
rect4 <- boundingRect(srch2$rect, slices, expand=1.1)
srch4 <- slicedBdrySearch(fobj, lsetthresh=thresh, slices=slices,
                          currentBdry=srch2, rect=rect4,
                          scale=srch2$scale,
                          control=list(axisDegree=2,
                                       maxSteps=6))
slsegs4 <- lapply(srch4, lsetSegs)  # list of 'lsetSegs' objects
# Number of level set segments found per slice:
sapply(slsegs4, length)
```

```
##  scale=10  scale=30  scale=50  scale=70  scale=90 scale=110
##        24        24        24        24        24        24
```

Here `control` is used make the number of rays per slice (24) similar to that with `slicedBdryFromRays()` (25), to equalize computational effort.

```
if (requireNamespace("ggplot2", quietly=TRUE)) {
  plt <- plotSegsList(slsegs4, dims=c("dfrac", "shape"), rect=rect3)
  print(plt + ggplot2::theme(panel.background=ggplot2::element_blank()) +
        ggplot2::labs(title="Confidence region sliced by 'scale' (adaptive search)"))
}
```



With adaptively selected origins the sizes and shapes of confidence region slices are a little clearer than with a single, fixed origin per slice.

References

Kim, Daeyoung and Lindsay, Bruce G. 2011. Using confidence distribution sampling to visualize confidence sets. *Statistica Sinica* 21:923–948.

Meeker, William Q., Hahn, Gerald J., Escobar, Luis A. *Statistical Intervals: A Guide for Practitioners and Researchers*, 2nd ed. John Wiley & Sons, 2017.