An Introduction to TITAN2

Matthew Baker (UMBC), Ryan King (Baylor), and David Kahle (Baylor)

Version 2.4.900

Introduction to Threshold Indicator Taxa Analysis with TITAN2

The purpose of this vignette is to walk users through the basic functionality of the package **TITAN2** for analysis of taxon-specific contributions to community change along an environmental gradient. The Everglades data for this vignette was described in Baker and King (2010) first published by King and Richardson (2003).

For users familiar with the original version of `titan()` and associated functions, released as a text file for R 2.9.2 in Appendix 3 of [Baker and King (2010)] (http://onlinelibrary.wiley.com/doi/10.1111/j.2041-210X.2009.00007.x/full), there are a number of updates included in this R package **TITAN2**, some of which may break your code. First, the original scripts have been divided into modular component functions to facilitate dissemination through CRAN. Second, a number of functions have been added to reduce run time and allow for the handling of large datasets. Third, several new analyses and plotting functions have been included, and others dropped. Fourth, several minor bugs have been fixed; these include screens for inappropriate data.

First Steps

For users new to **TITAN2**, the package can be loaded like any other R package downloaded from CRAN with `install.packages("TITAN2")`; this only needs to be done once. The package can then be loaded into an active R session with `library()`:

```
library("TITAN2")
```

Note that the package needs to be loaded once per R session.

As with the first version of the project, `titan()` requires 1) a site by taxa matrix and 2) an environmental gradient. To facilitate this discussion, we have included in **TITAN2** examples of both: the datasets `glades.taxa` and `glades.env`. This is the same data included with the original version in Baker and King (2010). Users can load the taxa data with the `data()` function:

```
data(glades.taxa)
str(glades.taxa, list.len = 5)
# 'data.frame': 126 obs. of  164 variables:
#  $ ABLARHAM: num  0 0 0 0 0 0 0 0 0 0 ...
#  $ ACENTRIA: num  0 0 0 0 0 0 0 0 0 0 ...
#  $ ANOPHELE: num  0 0 0 0 0.802 ...
#  $ APEDELAS: num  0 0 0 0 0 0 0 0 0 0 ...
#  $ APHAOPAC: num  0.382 0.802 1.294 0 0.564 ...
#   [list output truncated]
```

See below for a discussion of how you can get your data into R.

The `data()` function loads the site by taxa matrix of 126 rows and 164 columns `glades.taxa`, and the `str()` ("structure") function gives a summary of what the data object is. The columns of `glades.taxa` contain $log_{10}(x+1)$ transformed densities of macroinvertebrate taxa (though transformation is usually unnecessary in **TITAN2**). The rows are sample units. Taxa names are 8-character abbreviations of scientific names. Only taxa with at least 5 occurrences are included. Taxa in your file must occur at least 3 times. Missing values are not allowed.

Loading the sample environmental gradient data is done similarly:

```
data(glades.env)
str(glades.env)
# 'data.frame': 126 obs. of  1 variable:
#  $ TP.ugL: num  41.2 41.8 43.6 40.1 43.6 43 43.6 36.5 34.1 56.4 ...
```

This dataset has 126 rows and 1 column, surface-water total phosphorus (ug/L). As with the taxa data, missing values are not allowed in the environmental gradient data.

We are now ready to run the `titan()` function. (Note: this takes a while to run, so beware.)

```
glades.titan <- titan(glades.env, glades.taxa)
```

In the above use of `titan()` many parameters are set by default, but they can be customized to the user's circumstance by direct specification. The default values assumed by the above call are

```
glades.titan <- titan(glades.env, glades.taxa,
  minSplt = 5, numPerm = 250, boot = TRUE, nBoot = 500, imax = FALSE,
  ivTot = FALSE, pur.cut = 0.95, rel.cut = 0.95, ncpus = 1, memory = FALSE
)
```

`titan()`'s extensive computations often require long run times, typically minutes or hours depending on the specifications. To save you some time from running the above function yourself, we've included the results in the built-in dataset `glades.titan`, accessible with the `data()` function as before:

```
data(glades.titan)
str(glades.titan, 1)
# List of 13
#  $ sppmax     : num [1:164, 1:16] 5.05 14.1 109.4 18.35 39.55 ...
#   ..- attr(*, "dimnames")=List of 2
#  $ sumz.cp    : num [1:4, 1:6] 14.8 32.5 14.8 29.9 13.2 ...
#   ..- attr(*, "dimnames")=List of 2
#  $ env        : num [1:126, 1] 41.2 41.8 43.6 40.1 43.6 43 43.6 36.5 34.1 56.4 ...
#  $ taxa       : num [1:126, 1:164] 0 0 0 0 0 0 0 0 0 0 ...
#   ..- attr(*, "dimnames")=List of 2
#  $ envcls     : num [1:117] 3.5 4.8 5.3 6.1 6.5 6.5 6.5 6.8 7.2 7.3 ...
#  $ srtEnv     : num [1:126] 2.5 3.3 3.5 3.5 3.5 4.8 5.3 6.1 6.5 6.5 ...
#  $ ivzScores  : num [1:656, 1:117] 1 2 2 2 2 2 2 1 2 2 ...
#  $ ivz        : num [1:117, 1:2] 99.3 121.4 133.8 131.8 123.8 ...
#   ..- attr(*, "dimnames")=List of 2
#  $ ivz.f      : num [1:117, 1:2] 88.8 113.4 123.3 125.2 113.2 ...
#   ..- attr(*, "dimnames")=List of 2
#  $ maxSumz    : num [1:500, 1:2] 15.1 15.4 16.9 15.4 14.6 ...
#  $ maxFsumz   : num [1:500, 1:2] 14.3 15.4 16.9 15.4 16.4 ...
#  $ metricArray: num [1:164, 1:4, 1:500] 1 1 2 1 1 2 2 1 2 2 ...
#  $ arguments  : Named num [1:10] 5 250 1 500 0 0 0.95 0.95 2 0
#   ..- attr(*, "names")= chr [1:10] "minSplt" "numPerm" "boot" "nBoot" ...
```

In `titan()` the primary arguments are the environmental gradient (here `glades.env`) and the taxa matrix (here `glades.taxa`). When you use the function, be sure to save the output to another variable (this is the `glades.titan <- titan(gla...` part above). Note that the deviance argument in `titan()` has been dropped from previous versions. Other arguments include the following; you can learn more about them in the documentation of `titan()` obtained by typing `?titan`:

- `minSplt` - the minimum number of observations on either side of a change point required to compute IndVal, z scores, and associated statistics. There are few reasons to make this value larger unless one is investigating sample bias. Although small data sets may require reducing `minSplt`, it should never be smaller than 3.

- `numPerm` - the number of random permutations of the taxa data used to compare observed IndVal scores to random ordering of observations along the environmental gradient, compute IndVal p-values (the probability that the magnitude of the observed value is no different from a distribution of those obtained via random permutation), and to calculate the mean and standard deviation necessary for computing z scores. We recommend a minimum of 250 for this argument during data exploration, and higher numbers (e.g., 500 or 1000) for more formal analysis because larger numbers will result in convergence on a more precise z score that will show less variation between runs. Larger values will increase computation time.
- `boot` - a logical `TRUE` or `FALSE` controlling whether the function will employ a bootstrap resampling procedure to estimate uncertainty associated with the sample. This is strongly recommended as the absence of the bootstrap results and associated diagnostics will almost certainly result in interpretive errors, and should not be called **TITAN2** (Baker and King 2013).
- `nBoot` - controls the number of new datasets created by resampling the observed data with replacement. Values <500 can be used for exploration of large datasets to reduce computation time, but we recommend 500 or 1000 for formal analysis.
- `imax` - controls whether the IndVal maximum or the z-score maximum (default and v1.0) is used to determine change points for each taxon. This argument is provided as an option for those users concerned about the effect of bias introduced by the permutation used to calculate z scores (see Baker and King 2013 for explanation). In practice, we have found that z-score bias is rarely a concern with empirical biological survey data. Nevertheless, we provide the option because we are unable to anticipate the full range of applications users may require.
- `ivTot`- controls whether IndVal scores are computed by mean relative abundance across partitions (the default; after Dufrene and Legendre 1997) or relative abundance obtained by the ratio of summed abundance in each partition to the total. We have found that the latter is helpful for exploring the effects of extreme skew in samples along the environmental gradient, but do not generally recommend it.
- `pur.cut`- the cut off value that defines what is considered a pure response direction. The default (0.95) requires that 95% of the results from bootstrap replicates agree with the observed response direction. NOTE: it is possible for strong unimodal responses to produce high reliability and low purity.
- `rel.cut`- the cut off value that defines what is considered a reliable response magnitude. The default (0.95) requires that 95% of the results from bootstrap replicates have a IndVal p-value less than or equal to 0.05, indicating a response magnitude at a given change point location that differs significantly from what would expect from random permutation. Low reliability often indicates that resampling can produce sample distributions without a strong response signal, but this is not always the case.
- `ncpus`- controls the number of processing cores to which the bootstrap procedure is allocated. The default (1) processes each bootstrap replicate in sequence. If ncpus>1 the replicates will be allocated to different cores and processed in parallel using the R package **parallel** (which comes with R). Parallel processing dramatically shortens run time and is recommended for most desktop applications as modern personal computers typically have 2-8 processing cores.
- `memory`- a logical indicating whether temporary files should be written to disk during processing of bootstrap replicates. Although this IO procedure can extend sequential processing time, its effects is negligible relative to parallelization, and we recommend it for data matrices larger than 250,000 elements (e.g., 500 sites by 500 taxa). Note that use of `memory = TRUE` results in the creation of a scratch directory within the local workspace called "temp.dir"" containing the excess results from each bootstrap replicate necessary for accurate summary. These files may be saved or discarded following each program run.

During normal program operation, the taxa dataset is screened to assess whether it is appropriate for `titan()` and the following messages are written to the screen:

```
# 100% occurrence detected 1 times (0.8% of taxa), use of TITAN less than ideal for this data type
```

```
# Taxa frequency screen complete
```

The first screening checks for excessively rare or common taxa occurrences and returns a warning if 100% or below minimum number of occurrences are detected. In some cases where analysis would be nonsensical, `titan()` will end its run. The second screen checks to make sure the environmental gradient and the taxa matrix have the same number of records.

The next set of messages detail progress of the change point analysis computations on the observed data:

```
# Determining partitions along gradient
# Calculating observed IndVal maxima and class values
# Calculating IndVals using mean relative abundance
# Permuting IndVal scores
# IndVal $z$ score calculation complete
# Summarizing Observed Results
# Estimating taxa change points using z-score maxima
```

The final message issued in a normal execution presents the progress of the bootstrap routine by counting off replicates (if `ncpus = 1`):

```
# Bootstrap resampling in sequence...
# 1
# 2
# 3
```

or informing the user that the bootstrap is running in parallel (if `ncpus = 2` or greater):

```
# Bootstrap resampling in parallel using 2 CPUs...no index will be printed to screen
```

This is the most computationally intensive step in the analysis. The Everglades dataset takes ~6-7 seconds per bootstrap replicate, depending on processor speed. Thus, 500 replicates take 3500 seconds or 58 minutes and ~60 minutes for the entire run. However, `ncpus = 2` cuts this time in half to about 32 min, and `ncpus = 4` lowers the time to about 15 min, etc. For initial explorations, users may consider setting `boot = FALSE`. However, we do not recommend any interpretation of graphical outputs or many tabular outputs until filtered by bootstrap diagnostics.

Note on Loading Your Own Data As noted in the previous section, `titan()` requires two datasets: a site by taxa matrix and a environmental gradient dataset. The site by taxa matrix should ideally be of class `data.frame`, the standard data structure in R. These can be read into R in several ways, for example the `read.table()` or `read.csv()` functions. The **readr** package has other similar functions to help read data into R. The environmental gradient dataset can either be a `data.frame` object with one column or simply an atomic vector of class `numeric`. In addition to the the data-reading functions listed above, `scan()` can be helpful for reading data of this type into R. In addition to these, the `file.choose()` function allows users to select files interactively, which can help avoid pathing issues.

Tabular Results

Community change points are returned automatically at the end of each `titan()` call, but these tabular sum(z) change point results are part of the titan object and may be called as follows:

```
glades.titan$sumz.cp
#           cp    0.05  0.10  0.50   0.90  0.95
# sumz-  14.75 13.2000 13.20 14.60 18.355 20.10
# sumz+  32.45 21.6500 21.85 30.35 33.150 37.85
# fsumz- 14.75 13.1500 13.20 14.70 18.550 19.90
# fsumz+ 29.85 21.8975 28.10 30.10 32.750 33.70
```

The columns include the observed change point (`cp`) defined by the sum(z) maximum and selected quantiles (0.05-0.95) of the change points determined by resampling the observed data.

Rows include the change points corresponding to declining or negatively responding taxa (sumz-), those corresponding to the increasing or positively responding taxa (sumz+), and the corresponding scores using filtered versions of both sums. Filtering is achieved by computing the sum(z) scores using only those taxa that are determined to be pure and reliable indicators. FIltered scores are sensitive to `pur.cut` and `rel.cut` arguments at the `titan()` function call. Filtered scores may be used as a check to indicate whether impure or unreliable indicator taxa are contributing substantially to the pattern of sum(z) scores (in this case they do not).

To obtain tabular results for individual taxa we can simply type `glades.titan$sppmax`. Since this is a 164×16 matrix, we use the `head()` function to just display the top to get a feel for the results:

```
head(glades.titan$sppmax)
#          ienv.cp zenv.cp freq maxgrp IndVal obsiv.prob zscore      5%    10%
# ABLARHAM    5.05   15.60   15      1  31.83      0.004   9.77  7.3475 10.235
# ACENTRIA   14.10   14.60    7      1  14.24      0.004   4.99  9.3550 10.900
# ANOPHELE  109.40   43.15   15      2  27.42      0.004   7.81 29.1175 32.200
# APEDELAS   18.35   20.10    5      1   6.69      0.132   1.94 10.9000 13.060
# APHAOPAC   39.55   39.55  104      1  66.05      0.004   8.80 32.2000 33.000
# ARRENSP1   86.95   85.55    9      2  21.97      0.020   3.91 13.1500 18.170
#             50%    90%      95% purity reliability  z.median filter
# ABLARHAM 15.80  20.10  21.6000  1.000       1.000 10.604653      1
# ACENTRIA 14.55  28.05  28.5000  0.998       0.956  5.915082      1
# ANOPHELE 43.45  69.70  83.8625  1.000       1.000  9.336369      2
# APEDELAS 18.40  43.76  55.0000  0.844       0.520  2.565831      0
# APHAOPAC 37.85  39.55  40.1000  0.998       1.000  9.246758      1
# ARRENSP1 85.80 109.40 126.2000  0.960       0.812  4.716420      0
```

The resulting table lists all taxa as rows and the columns are as follows:

- `ienv.cp`- environmental change point for each taxon based on IndVal maximum (used if `imax = TRUE`)
- `zenv.cp`- environmental change point for each taxon based on z maximum (default, `imax = FALSE`)
- `freq`- number of non-zero abundance values per taxon
- `maxgrp`- 1 if z- (negative response); 2 if z+ (positive response)
- `IndVal`-Dufrene and Legendre 1997 IndVal statistic, scaled 0-100%
- `obsiv.prob`- probability of an equal or larger IndVal from random permutation. This is an uncorrected value that doesn't account for the analysis of multiple taxa in the same data set nor repeated estimates of across n partitions of the predictor (x value). To reiterated, this is *not* a valid test of indicator "significance", nor have the authors ever advocated its use in this manner (see Baker and King 2010). The authors include it here only because it is used in the calculation of reliability, which assesses the effect of nBoot replicates on the `obsiv.prob`; reliability, when coupled with purity, is a defensible metric for identifying robust indicator taxa.
- `zscore`- IndVal z score
- `5%`, `10%`, `50%`, `90%`, `95%`- change point quantiles among bootstrap replicates
- `purity` - proportion of replicates matching observed `maxgrp` assignment
- `reliability` - proportion of replicate `obsiv.prob` values $< = 0.05$
- `z.median`- median score magnitude across all bootstrap replicates
- `filter`- logical (if >0) indicating whether each taxa met purity and reliability criteria, value indicates `maxgrp` assignment.

Remember that each `titan` object (created with the `titan()` function) contains a number of additional elements for post hoc exploration:

```
str(glades.titan, max.level = 1, give.attr = FALSE)
# List of 13
#  $ sppmax     : num [1:164, 1:16] 5.05 14.1 109.4 18.35 39.55 ...
#  $ sumz.cp    : num [1:4, 1:6] 14.8 32.5 14.8 29.9 13.2 ...
#  $ env        : num [1:126, 1] 41.2 41.8 43.6 40.1 43.6 43 43.6 36.5 34.1 56.4 ...
#  $ taxa       : num [1:126, 1:164] 0 0 0 0 0 0 0 0 0 0 ...
#  $ envcls     : num [1:117] 3.5 4.8 5.3 6.1 6.5 6.5 6.5 6.8 7.2 7.3 ...
#  $ srtEnv     : num [1:126] 2.5 3.3 3.5 3.5 3.5 4.8 5.3 6.1 6.5 6.5 ...
#  $ ivzScores  : num [1:656, 1:117] 1 2 2 2 2 2 2 1 2 2 ...
#  $ ivz        : num [1:117, 1:2] 99.3 121.4 133.8 131.8 123.8 ...
#  $ ivz.f      : num [1:117, 1:2] 88.8 113.4 123.3 125.2 113.2 ...
#  $ maxSumz    : num [1:500, 1:2] 15.1 15.4 16.9 15.4 14.6 ...
#  $ maxFsumz   : num [1:500, 1:2] 14.3 15.4 16.9 15.4 16.4 ...
#  $ metricArray: num [1:164, 1:4, 1:500] 1 1 2 1 1 2 2 1 2 2 ...
#  $ arguments  : Named num [1:10] 5 250 1 500 0 0 0.95 0.95 2 0
```

`summary(glades.titan)` provides similar output. The items within each `titan` object include:

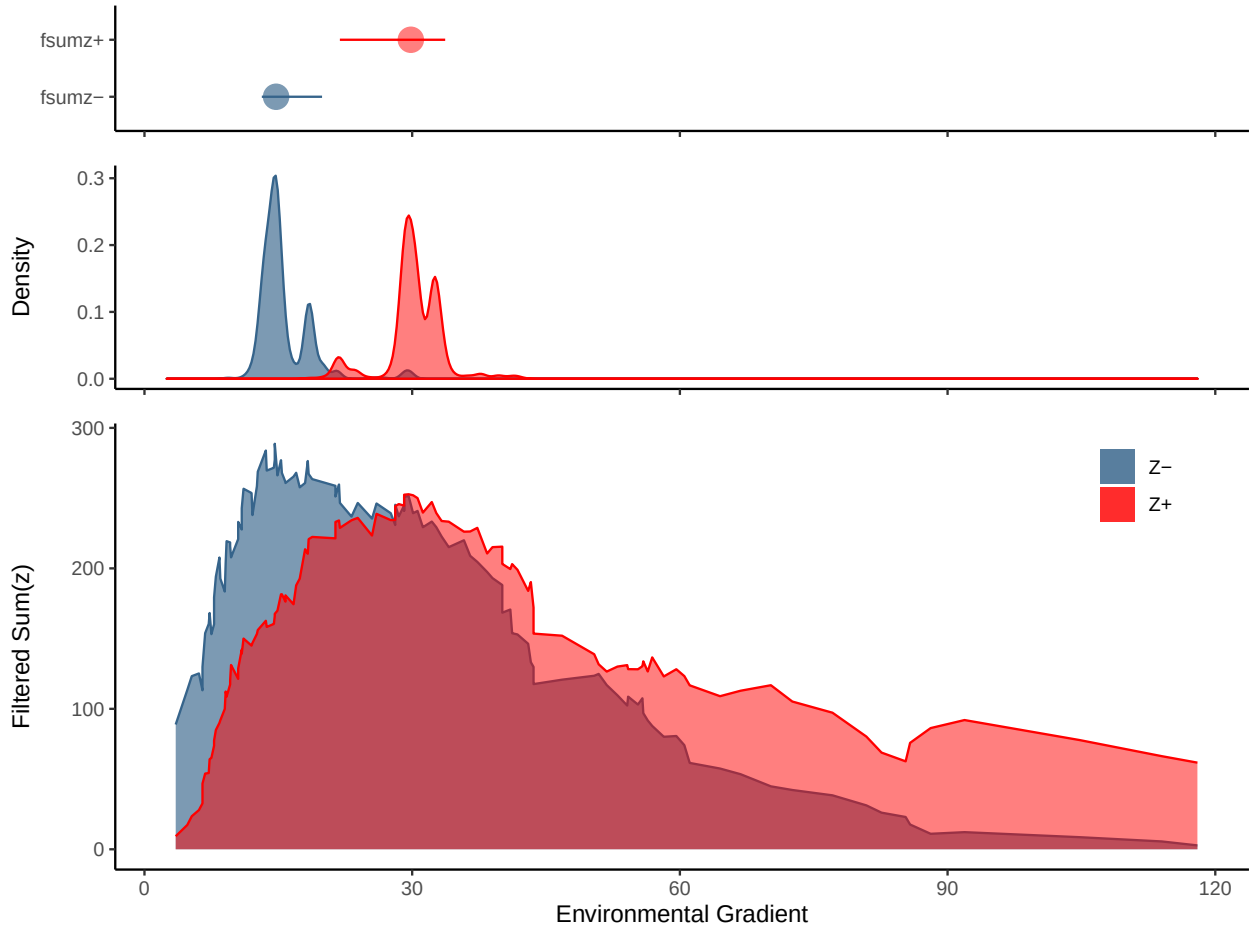
- `sppmax`- the taxon-specific result table described above
- `sumz.cp`- the community result table described above
- `env`- the environmental gradient, i.e. `glades.env`
- `taxa`- the taxa matrix, i.e. `glades.taxa`
- `envcls`- the environmental values used as candidate split points
- `srtEnv`- a sorted version of the environmental gradient
- `ivzScores` - a matrix containing `maxgrp`, z scores, indVals, and `obsiv.prob` for each taxon
- `ivz` - sums of all taxa `z` scores belonging to `maxgrp = 1` or `maxgrp = 2` at each candidate split point along the environmental gradient
- `ivz.f` - sums of all pure and reliable taxa z scores belonging to `maxgrp = 1` or `maxgrp = 2` at each candidate split point along the environmental gradient
- `maxSumz` - the environmental value corresponding to the unfiltered sum(z) maximum for each maxgrp and each bootstrap replicate
- `maxFsumz` - the environmental value corresponding to the filtered sum(z) maximum for each maxgrp and each bootstrap replicate
- `metricArray` - an array of summary metrics from the bootstrap based on ivzScores
- `arguments` - a list of arguments used in `titan()` function call

Visualizing Results

`plot_sumz()` and `plot_sumz_density()` The community-level output may be viewed using the `plot_sumz()` or `plot_sumz_density()` functions. The `plot_sumz_density()` function is new to this version of **TITAN2**. We recommend its use because we think it will be both easier to understand than previous output and encourage appropriate interpretation of results.

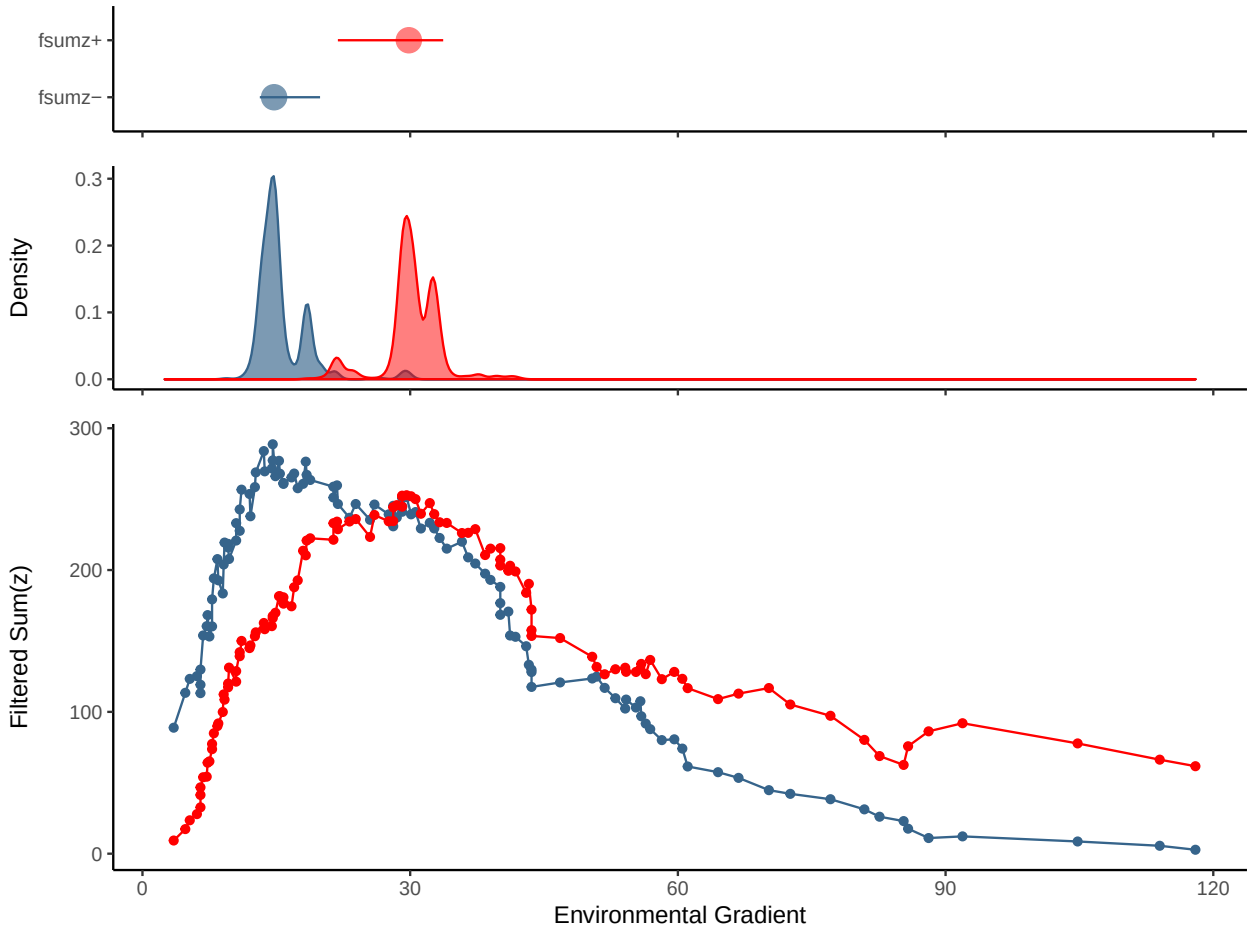
There are many options for changing the look of the graphs; see their full documentations with `?plot_sumz` or `?plot_sumz_density`. An abbreviated version is presented below. To begin, here's how you use `plot_sumz_density()`:

```
plot_sumz_density(glades.titan)
```



`plot_sumz_density()` plots community level change identified with **TITAN2**. There are three component panels in the default output and we will consider them from the bottom to the top. The first plot is a slight modification of the original sum(z) plot first presented by Baker and King (2010). It shows the magnitude of change among taxa declining along the gradient (z-) and those increasing along the gradient. Peaks in the values indicate points along the environmental gradient that produce large amounts of change in community composition and/or structure. These are the nominal community change points. Plateaus denote regions of similar change. The sums are converted to areas using the logical argument `ribbon`, a line plot can be obtained by setting `ribbon` to FALSE, and the original point plot can be obtained by setting the logical argument `points` to TRUE.

```
plot_sumz_density(glades.titan, ribbon = FALSE, points = TRUE)
```



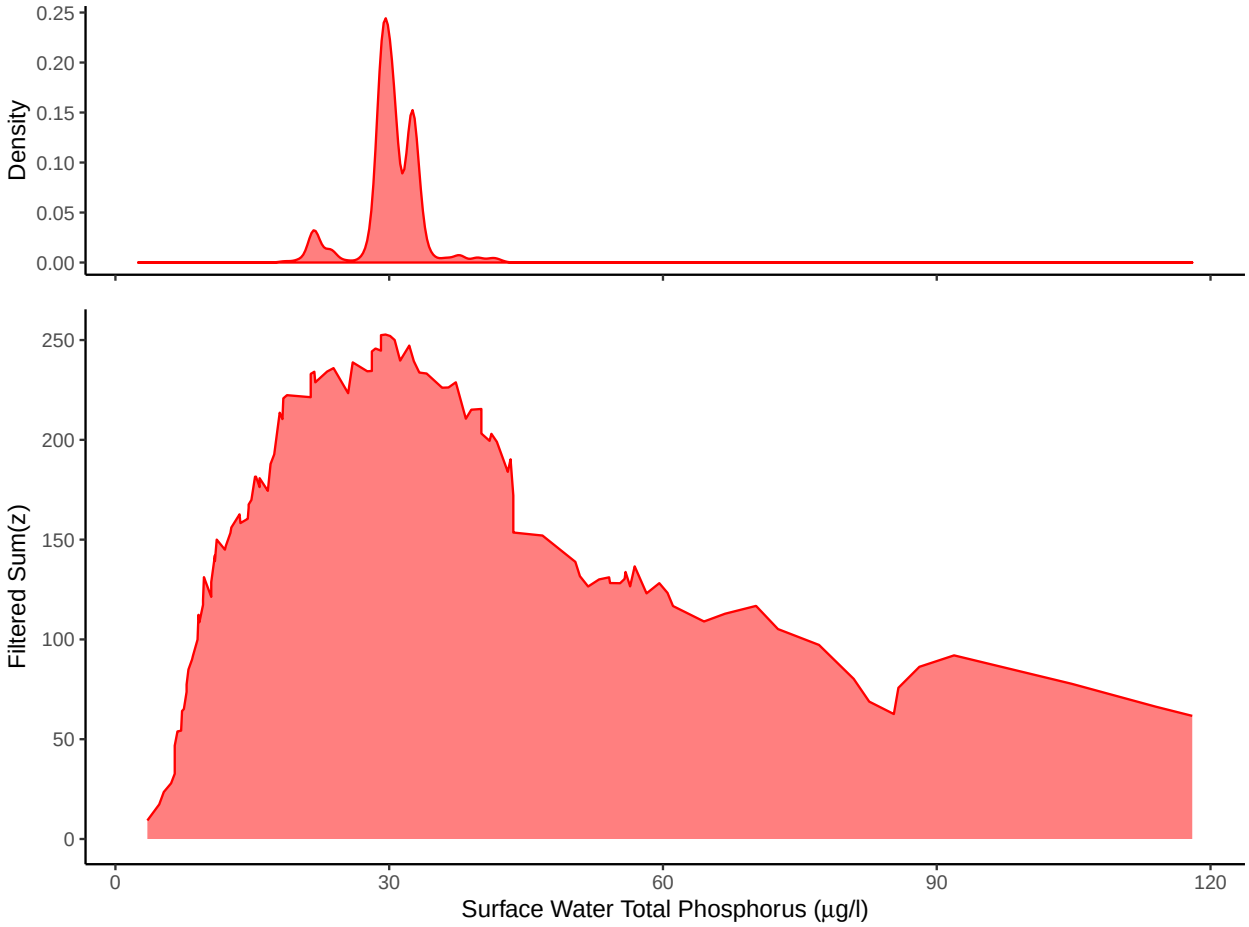
The second change from earlier versions of **TITAN2** is that the logical argument `filter` defaults to TRUE. This argument ensures that the only taxa contributing to patterns in the sum(z) are pure and reliable taxa identified by the bootstrap results rather than all taxa. Setting `filter` to FALSE allows impure or unreliable taxa to contribute to apparent patterns of commuity change. In general, the filtered scores should show a similar pattern but be somewhat lower in magnitude depending on contributions from impure or unreliable taxa.

The middle panel shows the estimated probability density of the sum(z) across all bootstrap replicates. Previously these distributions were represented by cumulative frequency distributions of the sum(z) maxima. A narrow spread indicates a high degree of precison regarding the location of the community change point.

The top panel shows the observed sum(z-) and sum(z+) maxima as circles with the 95th percentile of their distributions as horzontal lines. The primary reason for this plot is to illustrate how much information is hidden by this technique relative to the probability density functions.

There are several additional plotting options in `plot_sumz_density()`. First, any of the three panels can be dropped by setting logical agruments `sumz`, `density`, or `change_points` to FALSE. Second, it is possible to display only the z- or z+ taxa through the use of the logical arguments `sumz1` and `sumz2`. Finally, the x axis label can be edited using the `xlabel` aregument.

```
plot_sumz_density(glades.titan,
  ribbon = TRUE, points = FALSE, sumz1 = FALSE, change_points = FALSE,
  xlabel = expression(paste("Surface Water Total Phosphorus ("*mu*"g/l)"))
)
```



Besides the mandatory titan object, arguments include:

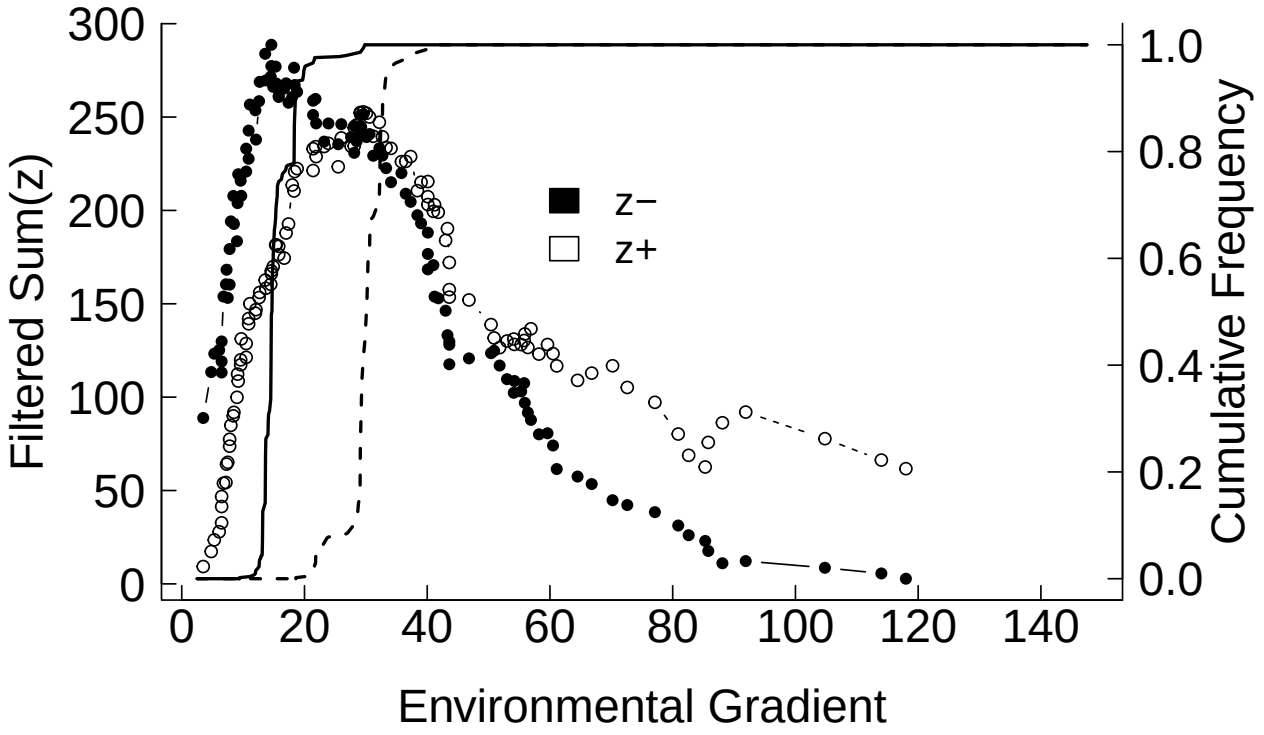
- `filter` - controls whether filtered (default) or unfiltered sum(z) scores are plotted
- `sumz` - logical whether sum(z) values are plotted
- `points` - logical whether point values of the sum(z) should be plotted
- `ribbon` - logical whether values of sum(z) should be plotted as a ribbon
- `density` - logical whether estimated probability densities should be plotted
- `change_points` - logical whether community change points should be plotted
- `sumz1` - logical whether sum(z) for `maxgrp = 1` should be plotted
- `sumz2` - logical whether sum(z) for `maxgrp = 2` should be plotted
- `xmin` - controls the x-axis minimum
- `xmax` - controls the x-axis maximum
- `xlabel` - text string for x-axis label
- `y1label` - text string for sum(z) y-axis, default labels are provided
- `y2label` - text string for density plot
- `alpha1` - controls the transparancy of sum(z-)
- `alpha2` - controls the transparancy of sum(z+)

- `col1` - color for 'maxgrp = 1
- `col2` - color for 'maxgrp = 2
- `axis_lab_size` - controls size of axis labels
- `legend.position` - controls location of legend

For `plot_sumz`, the older plotting function in **TITAN2**, filled and hollow symbols denote the magnitude of summed z scores of increasing (z+) or decreasing (z-) taxa. Peaks in the values indicate points along the environmental gradient that produce large amounts of change in community composition and/or structure. Solid and dashed lines are cumulative frequency distributions of sum(z-) and sum(z+) maxima (respectively) across bootstrap replicates. Vertical CFDs indicate narrow uncertainty about where the maximum chnage occurs, sloping or stair-step CFDs suggest broad uncertainty regarding the location of maximum change.

Note that users can adjust colors (e.g. col1 and fil1) and labels for the left hand y-axis using the argument `y1label` or the x-axis with `xlabel`. Setting `filter = TRUE` will display the filtered sum(z) scores. In general, the filtered scores should show a similar pattern but be somewhat lower in magnitude depending on contributions from impure or unreliable taxa:

```
plot_sumz(glades.titan, filter = TRUE)
```



`plot_taxa_ridges()` Pure (purity>=0.95 or higher) and reliable (reliability>=0.95 or higher) indicator taxa that contribute to the overall sum(z) scores can be plotted using the function `plot_taxa_ridges()`.

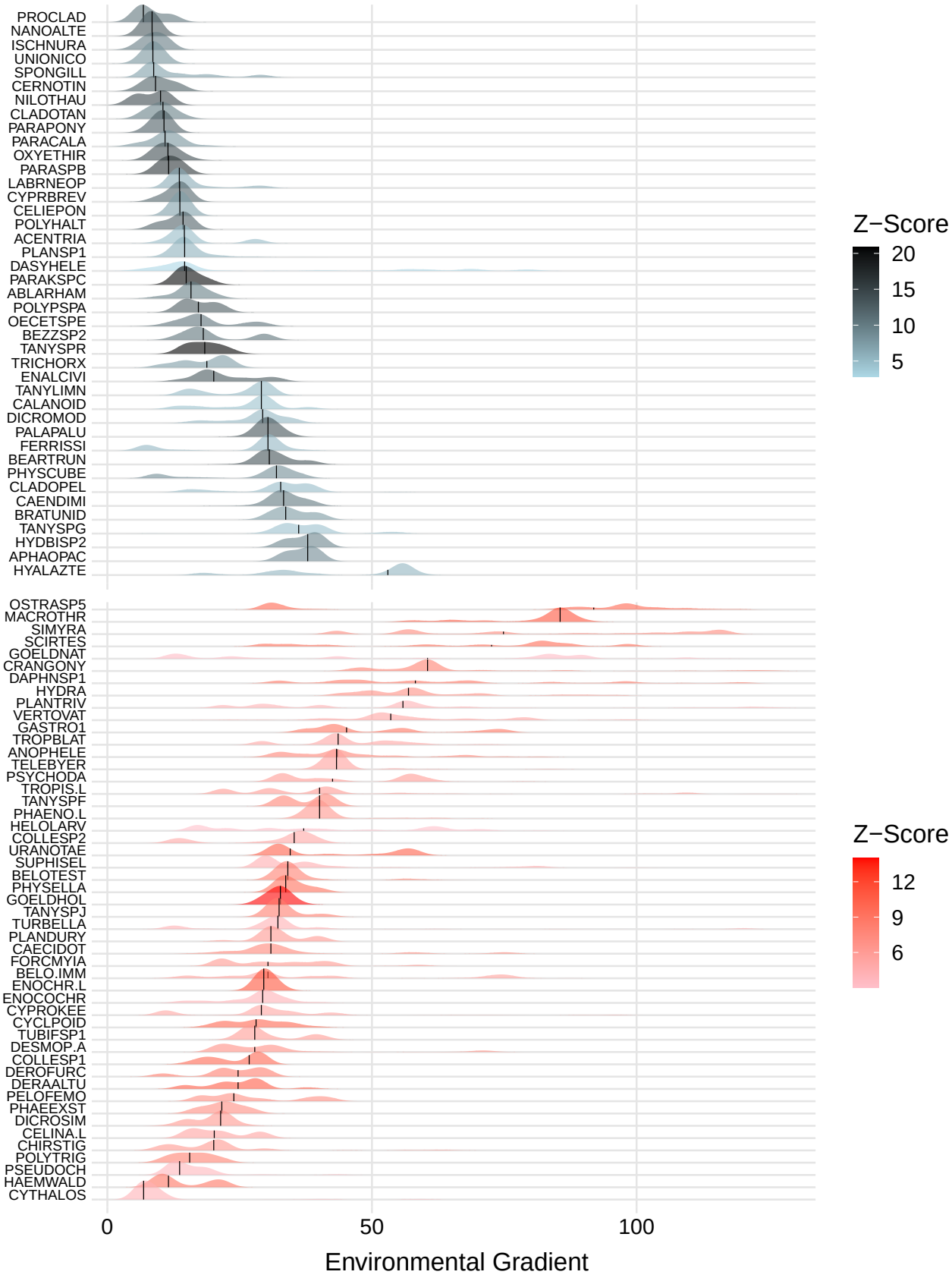
Taxa that meet purity (pur.cut) and reliability (rel.cut) criteria are extracted from the full taxa output object `titan.out$sppmax`, separated into two groups: z- (negative responders, or decliners) and z+ (positive responders, or increasers).

These two groups of negative and positive responders are arranged into separate, stacked panels for plotting. Within each panel, taxa change points (values of x resulting in the largest indicator z value) are visualized as a probability density function of the nBoot bootstrapped replicates (n=500 or however many nBoot replicates specified by the user).

Taxa names are arranged discretely along the y-axis based on the rank order of the central tendency of the probability density function for z- (negative, decliners) and z+ (positive, increasers), respectively.

There are many options for changing the look of the graph, so see the full documentation with `?plot_taxa_ridges`, but you can get started by simply calling `plot_taxa_ridges()` on the output of `titan()`:

```
plot_taxa_ridges(glades.titan, axis.text.y = 8)
# There are 90 indicator taxa of 90 possible for plotting
# Number of Decreasers = 41
# Number of Increasers = 49
```

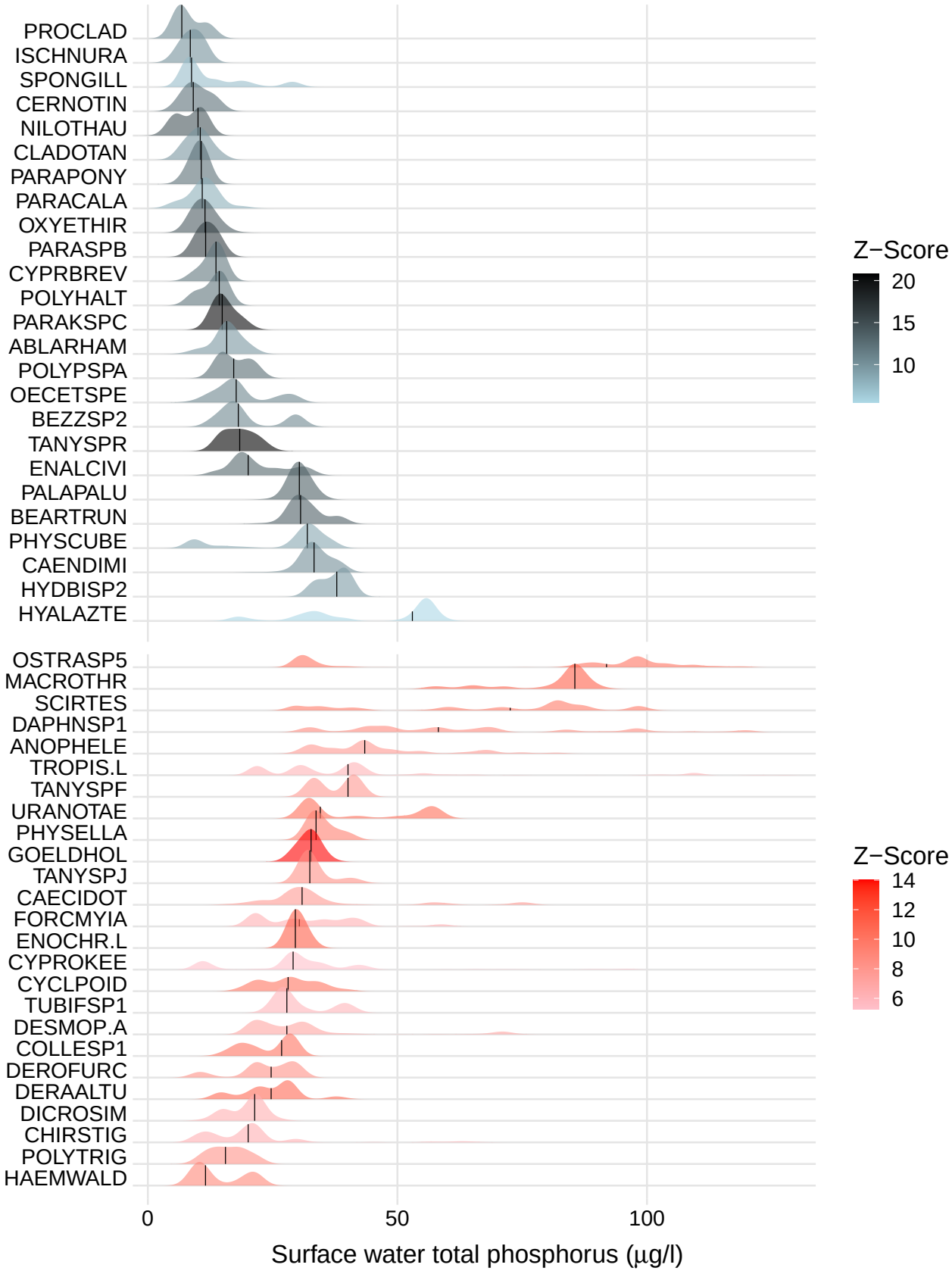


Optional arguments include:

- `z1` - logical whether `maxgrp = 1` taxa (z-, negative, decliners) should be plotted
- `z2` - logical whether `maxgrp = 2` taxa (z-, negative, decliners) should be plotted
- `z1_fill_low` - low value of the color fill gradient for z- (negative) taxa
- `z1_fill_high` - high value of the color fill gradient for z- (negative) taxa
- `z2_fill_low` - low value of the color fill gradient for z+ (positive) taxa
- `z2_fill_high` - high value of the color fill gradient for z+ (positive) taxa
- `pur.cut` - the purity value cutoff for selecting taxa for plotting (defaults to value selected in titan.out). We do not recommend selecting values less than 0.95 for purity; higher values up to 1.0 may be used to refine your selection to the purest responders in terms of direction.
- `rel.cut` - the reliability value cutoff for selecting taxa for plotting (defaults to value selected in titan.out). We do not recommend selecting values less than 0.95 for purity; higher values up to 1.0 may be used to refine your selection to the purest responders in terms of direction.
- `xlabel` - the x axis label; the default is "Environmental Gradient"
- `n_ytaxa` - the maximum number of taxa that can be plotted on the y-axes (combined between the two panels if both z1 and z2 = TRUE, or just for one panel if z1 or z2 = FALSE). If the total number of pure and reliable indicator taxa *exceeds* the value of n_ytaxa, then the function will select only the most pure and reliable taxa to match the value of n_ytaxa based on rank order of purity, reliability, and finally z.med (sequential sorting, respectively). The selection of n_ytaxa is independent of group membership, so that only the best indicators are displayed, regardless of z- or z+ grouping.
- `ytxt.sz` - size of the taxa labels on the y-axis. Defaults to ytxt.sz=10. Depending upon the height and width of the plot, labels may need to set to a smaller size to avoid overlap (e.g., see the plot above; here, ytxt.sz = 8).
- `printspp` - a logical determining whether the sppmax output for the plotted taxa is printed to the console; default = FALSE.
- `grid` - a logical determining whether gridlines are plotted; default=TRUE
- `xlim` - the upper and lower limits of the x-axis, e.g., xlim=c(0,180). The default is computed internally based on the lower and upper limits of the x variable.
- `bw` - short for "bandwidth", it controls the degree of smoothing of the probability density plots. This is computed internally but can be adjusted manually using this argument. We strongly advise against changing this parameter from the default.

Below is the same function with a user-specified x-label and n_ytaxa set to 50

```
plot_taxa_ridges(glades.titan,
  xlabel = expression(paste("Surface water total phosphorus ("*mu*"g/l)")),
  n_ytaxa = 50
)
# There are 90 indicator taxa of 50 possible for plotting
# Number of Decreasers = 41
# Number of Increasers = 49
```

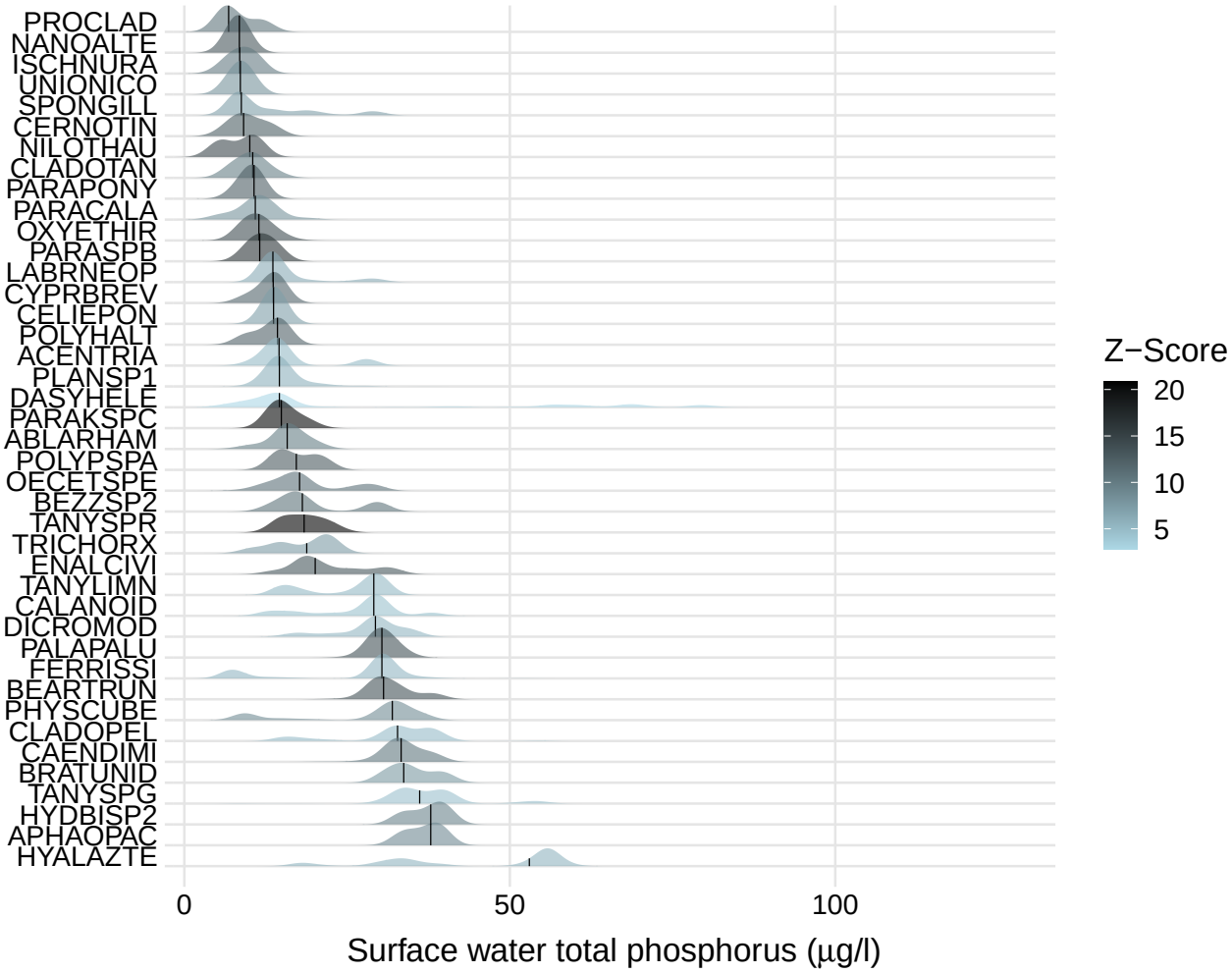


There are a total of 90 pure and reliable indicator taxa in the `glades.titan` output, thus 40 taxa were

excluded from this plot because we set `n_ytaxa = 50`. The 50 taxa plotted are the top 50 out of 90 taxa based on descending rank order of `purity`, `reliability`, and `z.median`, respectively.This argument can be useful when there are too many pure and reliable taxa to effectively visualize in one plot.

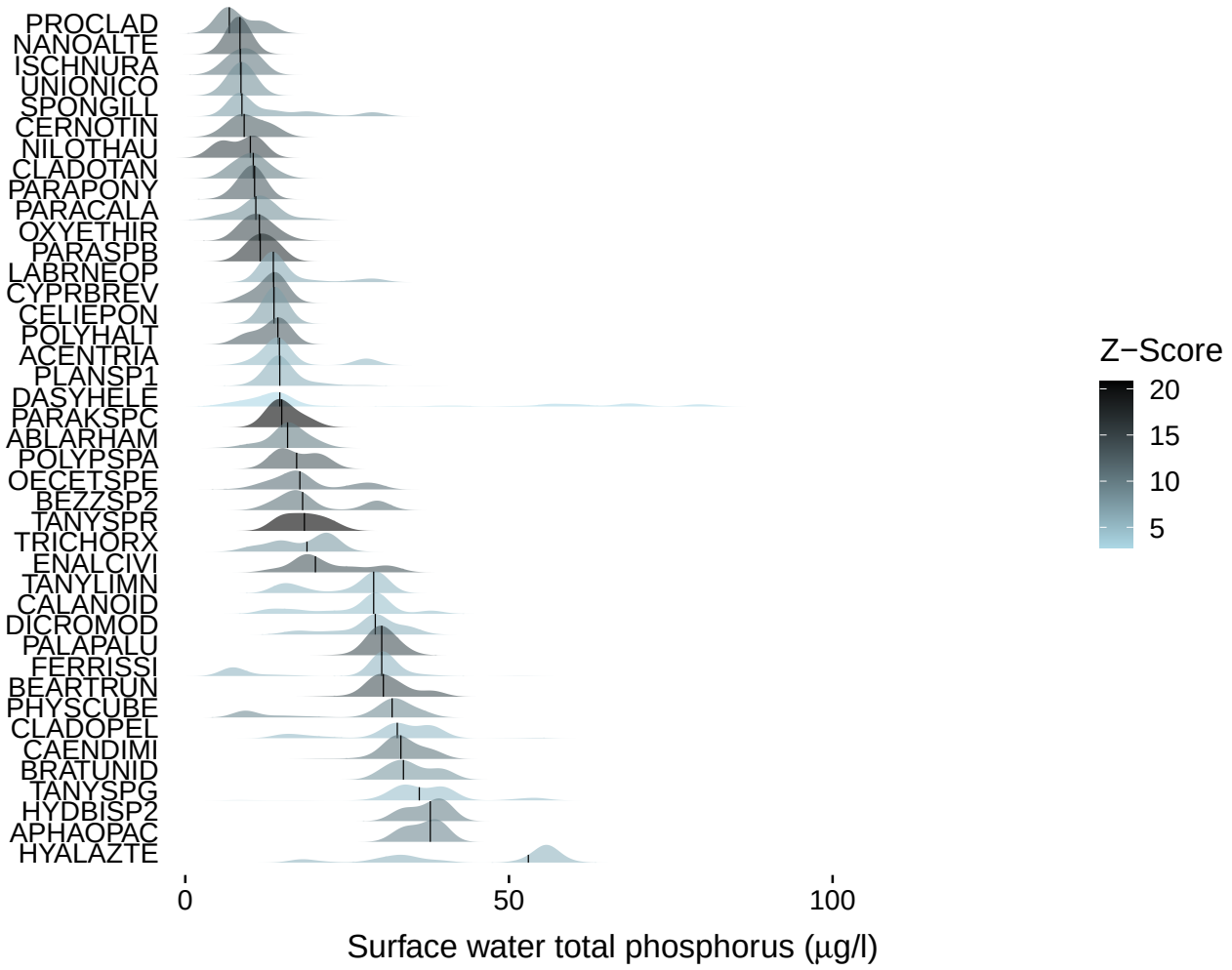
Alternatively, set `z1` or `z2 = FALSE`.

```
plot_taxa_ridges(glades.titan,
  xlabel = expression(paste("Surface water total phosphorus ("*mu*"g/l)")),
  z2 = FALSE
)
# There are 90 indicator taxa of 90 possible for plotting
# Number of Decreasers = 41
# Number of Increasers = 49
```



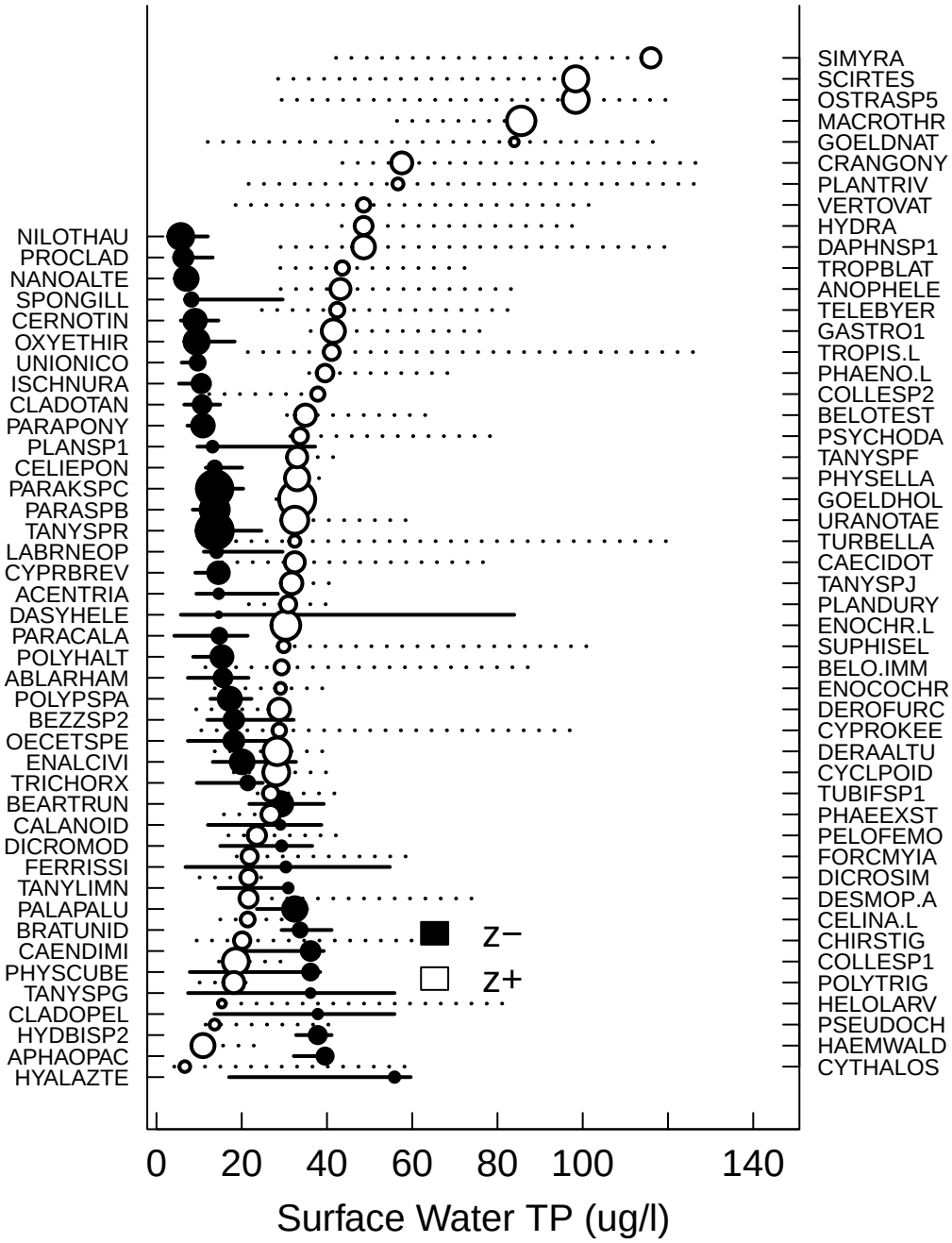
Gridlines are set to `TRUE` as the default, but can be removed if desired:

```
plot_taxa_ridges(glades.titan,
  xlabel = expression(paste("Surface water total phosphorus ("*mu*"g/l)")),
  z2 = FALSE, grid = FALSE
)
# There are 90 indicator taxa of 90 possible for plotting
# Number of Decreasers = 41
# Number of Increasers = 49
```



`plot_taxa()` Pure and reliable indicator taxa change points can be plotted using an older plotting function that was integral to the original TITAN2 release, `plot_taxa()`. We have replaced this plot with the distributions in `plot_taxa_ridges()` because it tended to empahsize the observed change points over the bootrap replicates. Again, there are many options for changing the look of the graph, so see the full documentation with `?plot_taxa`, but you can get started by simply calling `plot_taxa()` on the output of `titan()`:

```
plot_taxa(glades.titan, xlabel = "Surface Water TP (ug/l)")
```



Each plot includes robust declining taxa on the left axis, increasers on the right. Each observed change point (or median of bootstrap replicates if `z.med = TRUE`) is indicated by the circular symbol; the horizontal lines suggest 5-95% quantiles from the bootstrapped change point distribution.

Optional arguments include:

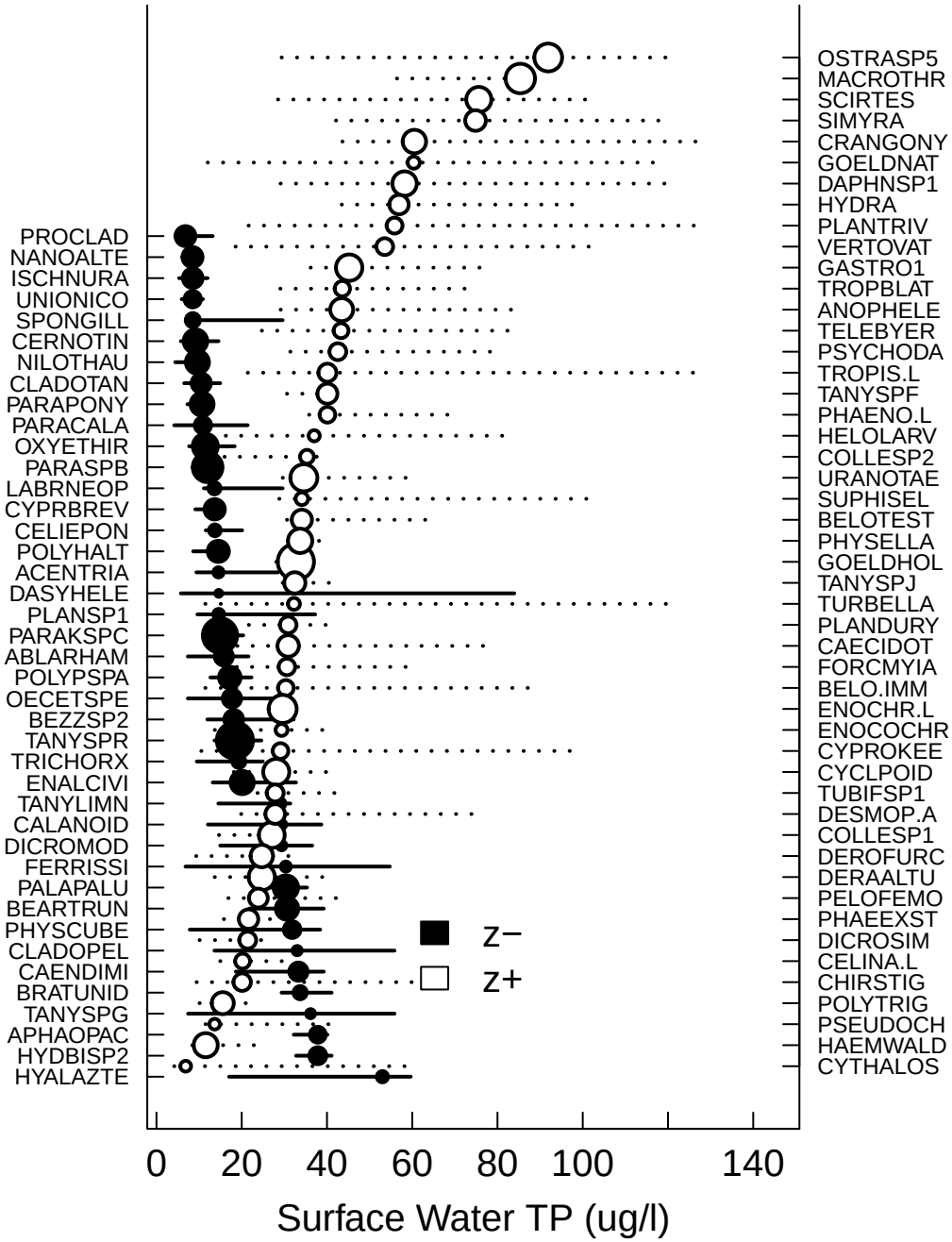
- `z1` - logical whether `maxgrp = 1` taxa change points should be plotted
- `z2` - logical whether `maxgrp = 2` taxa change points should be plotted
- `interval` - logical whether 5%-95% quantile intervals should be plotted
- `prob95` - controls whether change point symbols are plotted at observed value or at quantile extremes, the extremes are useful for thinking about change from a risk perspective

- `z.med` - controls whether change point symbols are plotted at median of bootstrap replicates and using median z score instead of observed value
- `cex.taxa` - controls size of taxon names in graph.

`log`, `at`, `xmin`, `xmax`, `xlabel`, `tck`, `bty`, `ntick`, `prtty`, `dig`, `leg.x`, `leg.y`, `cex`, `cex.axis`, `cex.leg`, `cex.lab`, `legend`, `col1`, `fil1`, `col2`, and `fil2` are the same as in `plot_sumz()` above.

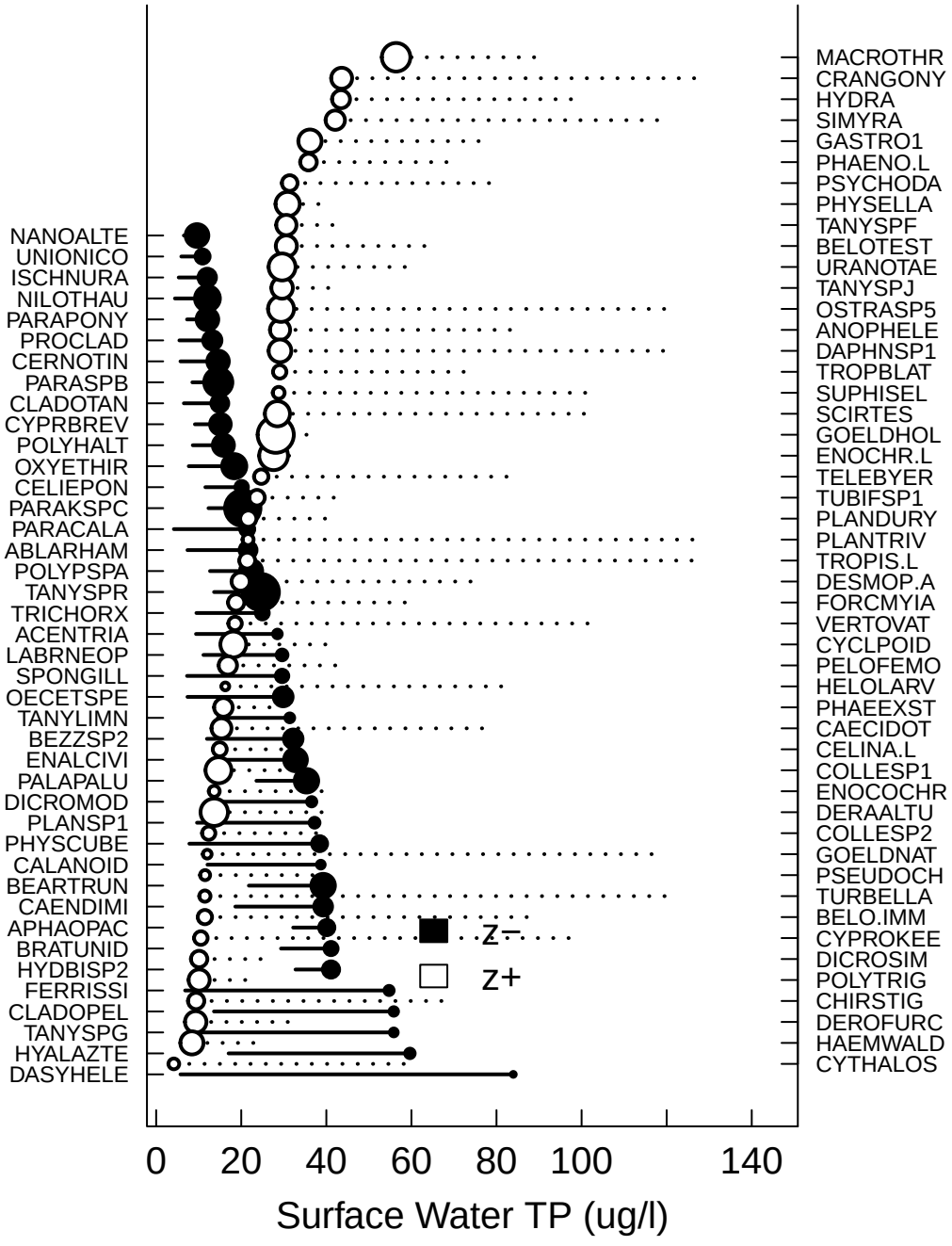
Here is the same function with `z.med = TRUE`:

```
plot_taxa(glades.titan, xlabel = "Surface Water TP (ug/l)", z.med = TRUE)
```



Alternatively, with `prob95 = TRUE`:

```
plot_taxa(glades.titan, xlabel = "Surface Water TP (ug/l)", z.med = FALSE, prob95 = TRUE)
```



Importantly, some of the control over the plotting that was optional in the original version has been internalized and warnings are issued when users plot results without using the bootstrap results. Tables of significant z- and z+ taxa used to make graph are automatically returned but no longer automatically written as text files (these can be easily extracted from the `glades.titan$sppmax` table).

`plot_cps()` In **TITAN2** 2.0 and later, there are several plotting options; all contained within a single function centered on interpreting taxon-specific change points in the context of the results from the bootstrap resampling procedure.

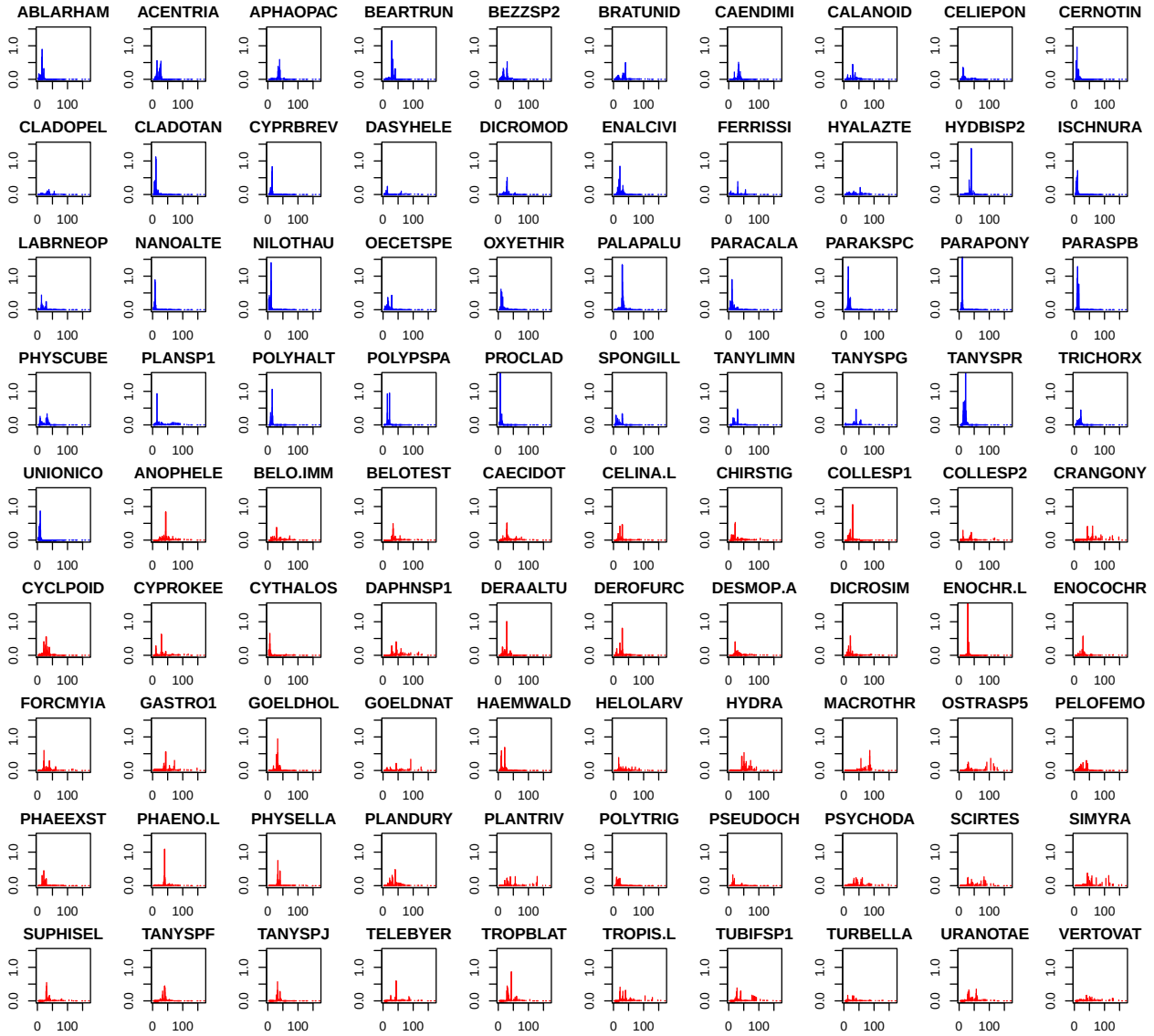
- `taxa.dist` - logical controls if taxon-specific change point distributions are plotted

- `z.weights` - logical controls if the median z-sore magnitude is used to weight the PDF from the distribution of resampled change points
- `taxaID` - if a number or name, creates a single taxon-specific abundance plot
- `cp.med` - logical controls if change point locations should be plotted using the median value across all bootstrap replicates instead of the observed value
- `cp.trace` - logical controls if IndVal and zscores are plotted for all candidate change points
- `cp.hist` - logical controls if histogram of change point PDF is plotted
- `stacked` - logical controls if change point PDFs for increasing and decreasing taxa are stacked or plotted separately

The remaining arguments: `xlabel`, `xmin`, `xmax`, `tck`, `bty`, `ntick`, `cex`, `cex.axis`, `cex.leg`, `cex.lab`, `leg.x`, `leg.y` and `legend` are the same as in the previous plotting functions.

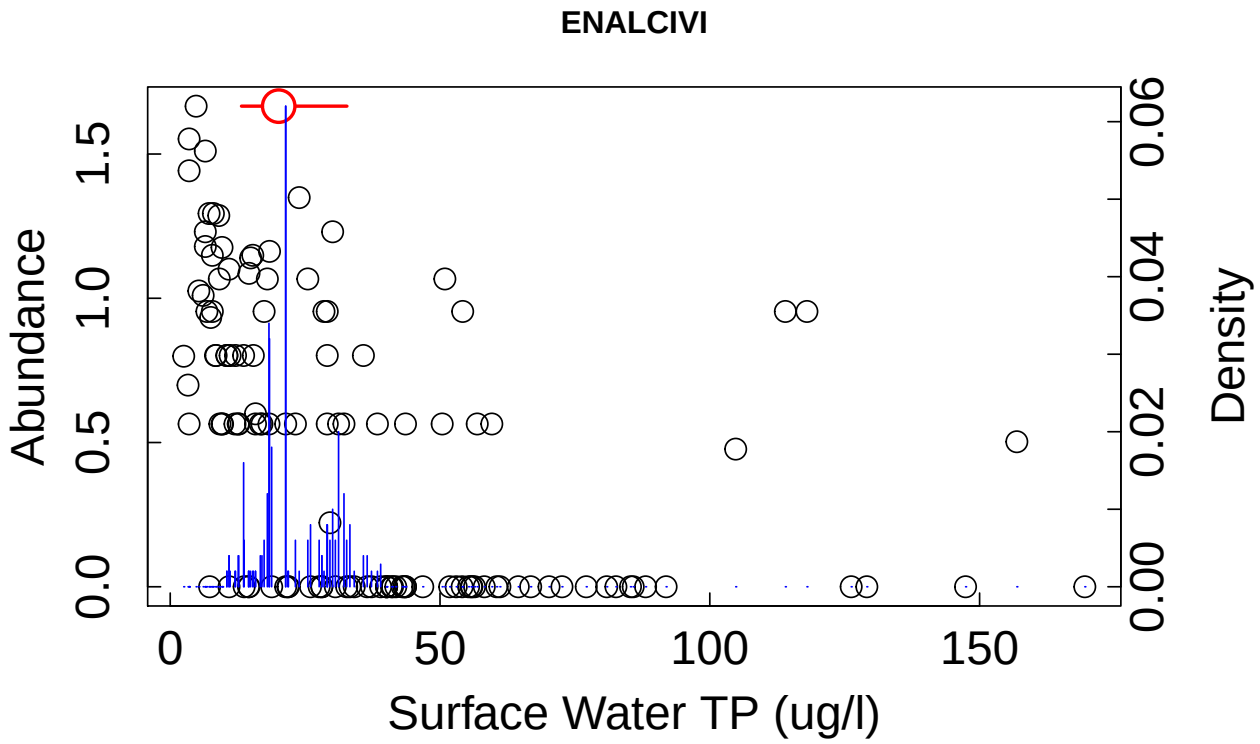
The default graphic from `plot_cps()` is a z-score weighted probability density function derived from the empirical distribution of change points across all bootstrap replicates plotted as a histogram for each pure and reliable taxon (blue for decreasers and red for increasers). NOTE: When you try to plot the matrix of plots consecutively you can generate an error about "Figure margins too large". This is easily addressed by closing the plot window, or turning the graphics device off.

```
plot_cps(glades.titan)
```



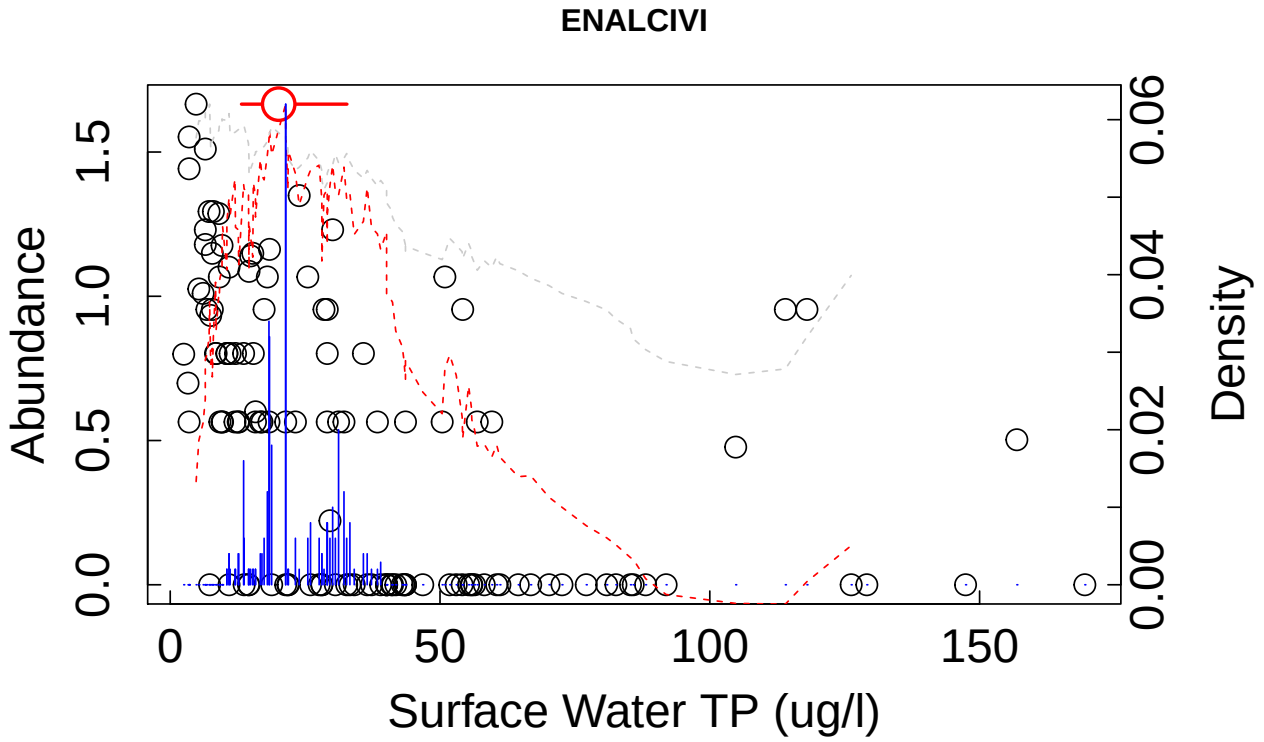
Typically, these histograms are visually scanned to distinguish those taxa whose bootstrapped change point distributions are clearly unimodal from those that are multimodal or more uniform. For any taxon of interest, a closer inspection can be achieved by setting the argument "taxaID" to a taxon ID number or label name. For example, in the plot above, if the declining (i.e., `maxgrp = 1`) taxon `"ENALCIVI"` is of interest, the user can set `taxaID = 57` or `taxaID = "ENALCIVI"` and either will produce the plot below. Again, you may need to close the plotting window or turn graphics off because we are switching plot margins.

```
plot_cps(glades.titan, taxaID = "ENALCIVI", xlabel = "Surface Water TP (ug/l)")
```



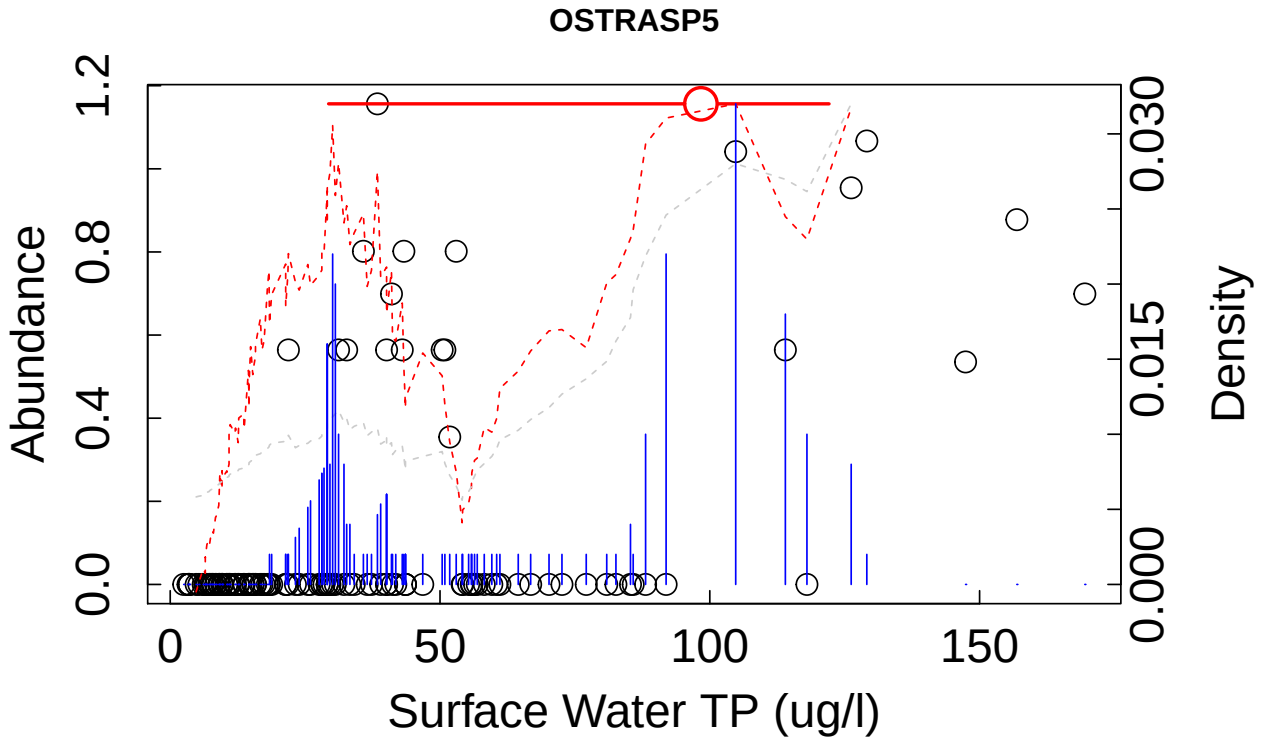
The base plot is simply taxon-specific abundance (black circles) along the environmental gradient. Overlaid near the top of the plot in red is the observed change point, or if `cp.med = TRUE`, the median change point across all bootstrap replicates. Overlaid on the abundance plot is the z-weighted (if `z.weights = TRUE`) probability density function of change point locations across all bootstrap replicates (blue histogram). Setting `cp.hist = FALSE` will remove the histogram. On the other hand, setting `cp.trace = TRUE` will plot the observed IndVal z scores (in red; what **TITAN2** uses by default if `imax = FALSE`) and the observed IndVal scores (in grey) for every candidate change point along the environmental gradient (see Baker and King 2013 for examples):

```
plot_cps(glades.titan, taxaID = "ENALCIVI", cp.trace = TRUE, xlabel = "Surface Water TP (ug/l)")
```



Below is an increasing taxon from the same data set with a conspicuously bimodal distribution of replicate change points:

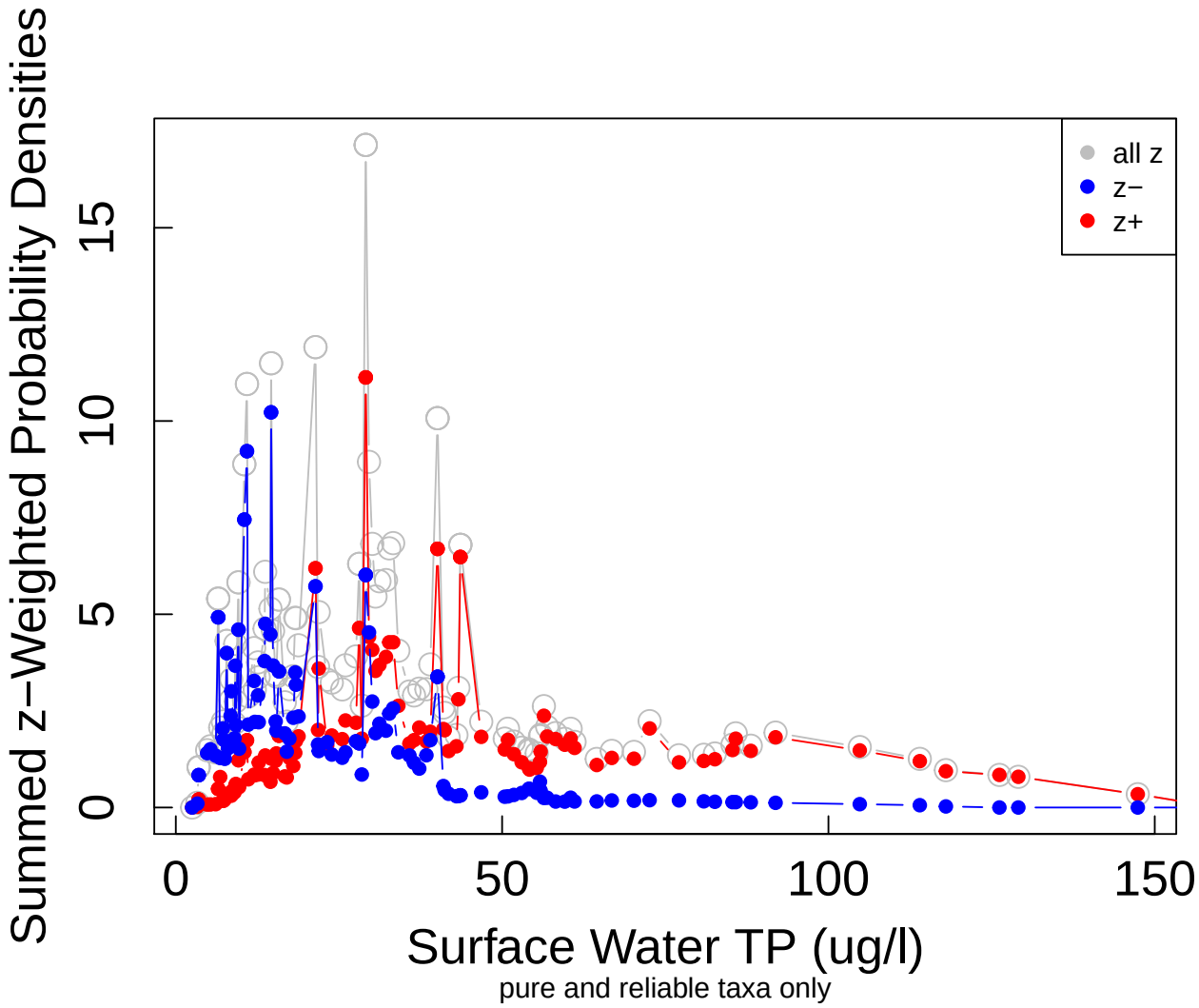
```
plot_cps(glades.titan, taxaID = "OSTRASP5", cp.trace = TRUE, xlabel = "Surface Water TP (ug/l)")
```



Here it appears as if two potential change points were predominantly identified: one that kept nearly all of the abundance to the right, and another (the observed) than kept nearly all the absences to the left.

Finally, if `taxa.dist = FALSE`, instead of a taxon-specific plot, `plot_cps()` will produce a plot of the summed z-weighted (if `z.weights = TRUE`) probability densities across increasing (`maxgrp = 1`; blue) decreasing (`maxgrp = 2`; red) and all pure and reliable taxa (filter>0; grey):

```
plot_cps(glades.titan, taxa.dist = FALSE, xlabel = "Surface Water TP (ug/l)")
# [1] "Summary of Summed Density Functions"
#           uw.env uw.max zw.env zw.max
# Group 1    14.55   1.00  14.55  10.22
# Group 2    28.80   1.34  28.80  11.13
# P&R Taxa   28.80   2.01  28.80  17.14
```

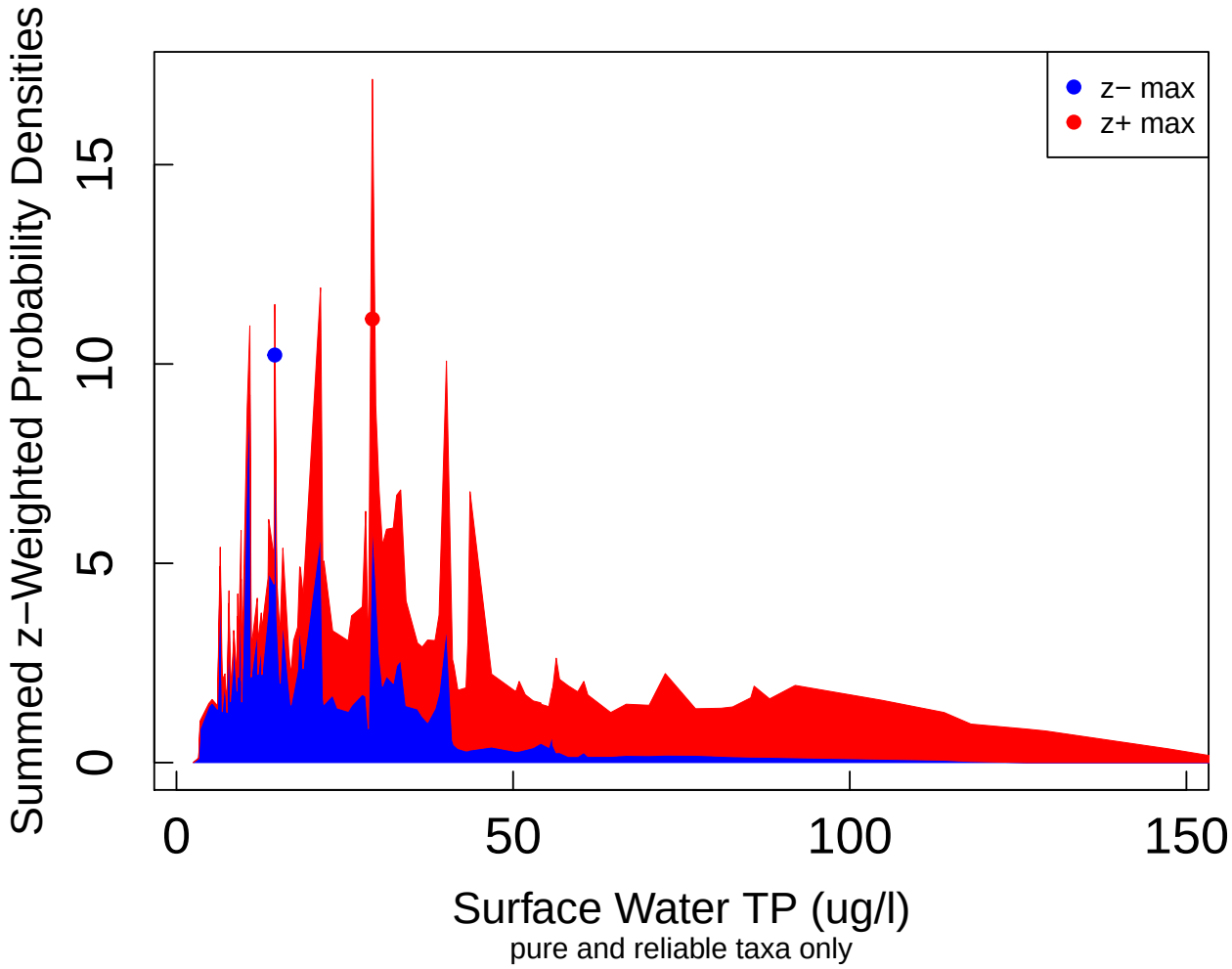


The accompanying table summarizes the location of both the z-weighted and unweighted maxima along the environmental gradient, as well as the magnitude of those maxima for decreasing taxa (`maxgrp = 1`), increasing taxa (`maxgrp = 2`), and all pure and reliable taxa combined. In this case there appears to be good correspondence between the sum(z) maxima and the summed z-weighted PDF maxima. In other words, the points at which community change appears to be greatest corresponds to where the sum of individual taxa changes tend to occur, even when different sets of observations are used. In practice, this is a desirable result we would expect to occur most of the time, yet exceptions to this pattern can also be informative. For example, in the plot above, there appears to be another candidate change point closer to ~10 ug/l where declining taxa tend to change frequently, and at both ~25 ug/l and ~40 ug/l there are large numbers of frequently occurring change points across both decliners and increasers even though they are not particularly

large for either group by itself.

Alternatively, these different sums can be represented as a stacked bar chart by setting `stacked = TRUE`:

```
plot_cps(glades.titan, taxa.dist = FALSE, xlabel = "Surface Water TP (ug/l)", stacked = TRUE)
# [1] "Summary of Summed Density Functions"
#           uw.env uw.max zw.env zw.max
# Group 1    14.55   1.00  14.55  10.22
# Group 2    28.80   1.34  28.80  11.13
# P&R Taxa   28.80   2.01  28.80  17.14
```



Here the blue and red circles correspond to the location of the respective maxima for decreasing and increasing taxa change points. In either case, the summed PDFs are meant to provide complementarity to the `plot_sumz()` graphic about where there are key values along the environmental gradient where many coincident change points occur regardless of the sample.

Citations:

Baker, M.E. and R. S. King. 2010. A new method for detecting and interpreting biodiversity and ecological community thresholds. Methods in Ecology and Evolution 1: 25-43.Z

Baker, M.E. and R. S. King. 2013. Of TITAN and strawmen: an appeal for greater understanding of community data. Freshwater Science 32(2):489-506.

King, R.S. and C. J. Richardson. 2003. Integrating bioassessment and ecological risk assessment: an approach to developing numerical water-quality criteria. Environmental Management 31(6):795-809.

Dufrene, M. and P. Legendre. 1997. Species assemblages and indicator species: the need for a flexible asymmetrical approach. Ecological Monographs. 67:345-366.