introduction

A Floristic Quality Assessment Calculator for R

This package provides functions for calculating Floristic Quality Assessment (FQA) metrics using regional FQA databases that have been approved or approved with reservations as ecological planning models by the U.S. Army Corps of Engineers (USACE). These databases are stored in a sister R package, fqadata. Both packages were developed for the USACE by the U.S. Army Engineer Research and Development Center's Environmental Laboratory.

To complete this tutorial interactively, follow along in R studio.

```
#attach packages required for this tutorial
library(fqacalc) #for FQA calculations
library(dplyr) #for data manipulation
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
```

Package Data

`fqacalc` contains all regional FQA databases that have been either fully approved or approved with reservations for use by the U.S. Army Corps of Engineers. By referencing these databases, the package can assign a Coefficient of Conservatism (or C Value) to each plant species that the user inputs. A list of regional FQA databases can be viewed using the `db_names()` function, and specific FQA databases can be accessed using the `view_db()` function. Below is an example of how to view one of the regional databases.

```
#view a list of all available databases
head(db_names()$fqa_db)
#> [1] "atlantic_coastal_pine_barrens_2018"
#> [2] "chicago_region_2017"
#> [3] "colorado_2020"
#> [4] "dakotas_excluding_black_hills_2017"
#> [5] "delaware_2013"
#> [6] "eastern_great_lakes_hudson_lowlands_2018"

#NOTE citations for lists can be viewed using db_names()$citation

#store the Colorado database as an object
colorado <- view_db("colorado_2020")

#view it
```

```
head(colorado)
#> # A tibble: 6 × 13
#>   name    name_origin acronym accepted_scientific_…¹ family nativity     c     w
#>   <chr>   <chr>       <chr>   <chr>                  <chr>  <chr>    <dbl> <dbl>
#> 1 ABIES … accepted_s… ABBI3   Abies bifolia          Pinac… native       5     1
#> 2 ABIES … synonym     <NA>    Abies bifolia          Pinac… native       5     1
#> 3 ABIES … accepted_s… ABCO    Abies concolor         Pinac… native       5    NA
#> 4 ABRONI… accepted_s… ABEL    Abronia elliptica      Nycta… native       4    NA
#> 5 ABRONI… accepted_s… ABFR2   Abronia fragrans       Nycta… native       6    NA
#> 6 ABRONI… accepted_s… ABAR    Abronia glabrifolia    Nycta… native       9    NA
#> # ℹ abbreviated name: ¹accepted_scientific_name
#> # ℹ 5 more variables: wetland_indicator <chr>, physiognomy <chr>,
#> #   duration <chr>, common_name <chr>, fqa_db <chr>
```

`fqacalc` also comes with sample inventory data from Crooked Island, Michigan, downloaded from the Universal FQA Calculator. The data set is called `crooked_island` and is used in this tutorial to demonstrate how the package works. When calculating metrics for crooked_island, use the 'michigan_2014' regional database.

```
#view the data
head(crooked_island)
#>   acronym  common_name                    name
#> 1  ABIBAL   balsam fir          Abies balsamea
#> 2  AMMBRE marram grass Ammophila breviligulata
#> 3  ANTELE  white camas        Anticlea elegans
#> 4  ARCUVA    bearberry Arctostaphylos uva-ursi
#> 5  ARTCAM     wormwood    Artemisia campestris
#> 6  CALEPI    reedgrass  Calamagrostis epigeios

#print the dimensions (35 rows and 3 columns)
dim(crooked_island)
#> [1] 35  3

#view the documentation for the data set (bottom right pane of R studio)
?crooked_island

#load the data set into local environment
crooked_island <- crooked_island
```

Reading Data into R

Data (inventory or transect) can be read into R for analysis using base R or the `readxl` package (for .xls or .xlsx files).

If the data is a csv file, it can be read in using `read.csv()`. For example, code to read in data might look like `my_data <- read.csv("path/to/my/data.csv")`. If the data is in an Excel file, it can be read in with the same code, but replace `read.csv()` with `read_excel()`.

In order to calculate FQA metrics using `fqacalc`, the data must be in the following format:

1. The data must have either a column named `name` containing scientific names of plant species, or a column named `acronym` containing acronyms of plant species. Different regional FQA databases use

different naming conventions and have different ways of creating acronyms (and some don't have acronyms!) so be sure to look at the relevant regional database to check that the site assessment is using the same conventions. Names/acronyms do not have to be in the same case, but otherwise must exactly match their counterpart in the regional FQA database in order to be recognized by `fqacalc` functions.

2. If the user is calculating cover-weighted metrics, the data must have another column containing cover values and it must be called `cover`. If the cover values are in percent cover, they must be between 0-100. If they are in a cover class, such as the Braun-Blanquet classification system, they must be correct for that class or else they won't be recognized. See the section on cover-weighted functions to learn more about cover classes.
3. If the user is calculating cover-weighted metrics for a transect containing multiple plots, the data should also have a column containing the plot ID. The plot ID column can have any name, and it can contain numbers or characters, as long as the IDs are exactly the same within plots but distinct between plots.

In this case, each observation is one row, containing the species name or acronym, the cover value, and the plot ID. It might look something like this:

plot_id	name	cover
1	Plant A	20
1	Plant B	50
2	Plant C	35
2	Plant D	45

Functions that Match Plant Species from Data to Regional FQA Databases

`fqacalc` contains two functions that help the user understand how the data they input matches up to the regional database: `accepted_entries()` and `unassigned_plants()`. `accepted_entries()` is a function that shows which plant species in the input data frame are successfully matched to species in the regional database, and `unassigned_plants()` shows which species are matched but don't have a C value stored in the regional database.

What happens when a plant species is not in the regional FQA database?

`accepted_enteries` shows which species are recognized, but it also provides warnings when a species is not recognized. To demonstrate this we can add a mistake to the `crooked_island` data set.

```
#introduce a typo
mistake_island <- crooked_island %>%
  mutate(name = sub("Abies balsamea", "Abies blahblah", name))

#store accepted entries
accepted_entries <- accepted_entries(#this is the data
                                     mistake_island,
                                     #'key' to join data to regional database
                                     key = "name",
```

```
                                    #this is the regional database
                                    db = "michigan_2014",
                                    #include native AND introduced entries
                                    native = FALSE)
#> Species ABIES BLAHBLAH not listed in database. It will be discarded.
```

Now, when we use `accepted_entries()` to see which species were matched to the regional data set, we can see that we received a message about the species 'ABIES BLAHBLAH' being discarded and we can also see that the accepted entries data set we created only has 34 entries instead of the expected 35 entries.

What happens when plant species don't have C values?

In some cases, a plant species can be matched to the regional database, but the species is not associated with any C Value. Plant species that are matched but have no C Value will be excluded from FQA metric calculation but they can *optionally* be included in other metrics like species richness, relative cover, relative frequency, relative importance, and mean wetness, as well as any summarizing functions containing these metrics. This option is denoted with the `allow_no_c` argument.

`unassigned_plants()` is a function that shows the user which plant species have not been assigned a C Value.

```
#To see unassigned_plants in action we're going to Montana!

#first create a df of plants to input
no_c_plants<- data.frame(name = c("ABRONIA FRAGRANS",
                                  "ACER GLABRUM",
                                  "ACER GRANDIDENTATUM",
                                  "ACER PLATANOIDES"))

#then create a df of unassigned plants
unassigned_plants(no_c_plants, key = "name", db = "montana_2017")
#> montana_2017 does not have wetness coefficients, wetland metrics cannot be calculated.
#>                   name              name_origin acronym accepted_scientific_name
#> 1    ABRONIA FRAGRANS accepted_scientific_name    <NA>         Abronia fragrans
#> 2 ACER GRANDIDENTATUM accepted_scientific_name    <NA>      Acer grandidentatum
#>          family     nativity  c  w wetland_indicator physiognomy duration
#> 1 Nyctaginaceae       native NA NA              <NA>        <NA>     <NA>
#> 2     Aceraceae undetermined NA NA              <NA>        <NA>     <NA>
#>                   common_name       fqa_db
#> 1 Fragrant White Sand-Verbena montana_2017
#> 2              Bigtooth Maple montana_2017
```

The function returns two species that are in the 'montana_2017' databases but aren't assigned a C Value.

How will duplicates be treated?

If the data contains duplicate species, they will be excluded from certain FQA metrics. For example, species richness counts the number of unique species, so duplicates are not allowed. Generally, duplicates are excluded for all unweighted (inventory) metrics but can optionally be included in cover-weighted metrics and are always included in relative metrics.

Duplicate behavior in cover-weighted functions is controlled by the `allow_duplicates` argument and the `plot_id` argument. If `allow_duplicates = FALSE`, no duplicate species will be allowed at all, no matter how `plot_id` is set. If `allow_duplicates = TRUE` and the `plot_id` argument is set, duplicate species will be allowed *if they are in different plots*.

If there are duplicates, and the user is attempting to perform a cover-weighted calculation where duplicates are not allowed, the duplicated species will be condensed into one entry with an aggregate cover value. A message will notify the user if this occurs. See this example.

```
#write a dataframe with duplicates
transect <- data.frame(acronym  = c("ABEESC", "ABIBAL", "AMMBRE",
                                    "AMMBRE", "ANTELE", "ABEESC",
                                    "ABIBAL", "AMMBRE"),
                       cover = c(50, 4, 20, 30, 30, 40, 7, 60),
                       plot_id = c(1, 1, 1, 1, 2, 2, 2, 2))

#set allow_duplicates to FALSE
cover_FQI(transect, key = "acronym", db = "michigan_2014",
          native = FALSE, allow_duplicates = FALSE)
#> Duplicate entries detected. Duplicates will only be counted once. Cover values of duplicate
         species will be added together.
#> [1] 11.89212

#set allow_duplicates to TRUE
#but set plot_id so duplicates will not be allowed within the same plot
cover_FQI(transect, key = "acronym", db = "michigan_2014",
          native = FALSE, allow_duplicates = FALSE, plot_id = "plot_id")
#> Duplicate entries detected. Duplicates will only be counted once. Cover values of duplicate
         species will be added together.
#> [1] 11.35398
```

Will synonyms be recognized?

Some regional FQA databases include accepted scientific names as well as commonly used synonyms. As long as these synonyms are in the regional database, they will be recognized by `fqacalc` functions. There are a few important rules regarding synonyms.

1. If both the synonym and the accepted name are used in the data, the synonym will be converted to the accepted name and both observations will only count as *one* species.
2. If the data contains a name that is listed as a synonym to one species and an accepted name to a different species, it will default to the species with the matching accepted name.
3. If the data contains a species that is listed as a synonym to multiple species in the regional FQA database, this entry will *not* be included! To include the species, enter the accepted scientific name instead of the synonym.

In all of these cases, `fqacalc` functions will print messages to warn the user about synonym issues. See this example:

```
#df where some entries are listed as accepted name and synonym of other species
synonyms <- data.frame(name = c("CAREX FOENEA", "ABIES BIFOLIA"),
                       cover = c(60, 10))
```

```
mean_c(synonyms, key = "name", db = "wyoming_2017", native = F)
#> CAREX FOENEA is an accepted scientific name and a synonym. It will default to accepted scientific
         name.
#> [1] 6
```

Unweighted (Inventory) FQI Metrics

`fqacalc` contains a variety of functions that calculate Total Species Richness, Native Species Richness, Mean C, Native Mean C, Total FQI, Native FQI, and Adjusted FQI. All of these functions eliminate duplicate species and species that cannot be found in the regional database. All but Total Species Richness and Native Species Richness automatically eliminate species that are not associated with a C Value.

Function Arguments

In general, all of these metric functions have the same arguments.

- **x**: A data frame containing a list of plant species. This data frame *must* have one of the following columns: `name` or `acronym`.
- **key**: A character string representing the column that will be used to join the input `x` with the regional FQA database. If a value is not specified the default is `name`. `name` and `acronym` are the only acceptable values for key.
- **db**: A character string representing the regional FQA database to use. See `db_names()` for a list of potential values.
- **native**: native Boolean (TRUE or FALSE). If TRUE, calculate metrics using only native species.

Additionally, `species_richness()` and `all_metrics()` have an argument called `allow_no_c`. If `allow_no_c = TRUE` than species that are in the regional FQA database but don't have C Values will be included. If `allow_no_c` is FALSE, then these species will be omitted. This argument is also found in `mean_w()` and all of the relative functions.

Functions

```
#total mean c
mean_c(crooked_island, key = "acronym", db = "michigan_2014", native = FALSE)
#> [1] 5.371429

#native mean C
mean_c(crooked_island, key = "acronym", db = "michigan_2014", native = TRUE)
#> [1] 6.714286

#total FQI
FQI(crooked_island, key = "acronym", db = "michigan_2014", native = FALSE)
#> [1] 31.7778

#native FQI
FQI(crooked_island, key = "acronym", db = "michigan_2014", native = TRUE)
#> [1] 35.52866
```

```r
#adjusted FQI (always includes both native and introduced species)
adjusted_FQI(crooked_island, key = "acronym", db = "michigan_2014")
#> [1] 60.0544
```

And finally, `all_metrics()` prints all of the metrics in a data frame format.

```r
#a summary of all metrics (always includes both native and introduced)
#can optionally include species with no C value
#--if TRUE, this species will count in species richness and mean wetness metrics
all_metrics(crooked_island, key = "acronym", db = "michigan_2014", allow_no_c = TRUE)
#>                          metrics     values
#> 1          Total Species Richness 35.0000000
#> 2         Native Species Richness 28.0000000
#> 3     Introduced Species Richness  7.0000000
#> 4    % of Species with no C Value  0.0000000
#> 5     % of Species with 0 C Value 20.0000000
#> 6   % of Species with 1-3 C Value  8.5714286
#> 7   % of Species with 4-6 C Value 34.2857143
#> 8  % of Species with 7-10 C Value 37.1428571
#> 9                         Mean C  5.3714286
#> 10                Native Mean C  6.7142857
#> 11                    Total FQI 31.7778000
#> 12                   Native FQI 35.5286605
#> 13                 Adjusted FQI 60.0543971
#> 14                 Mean Wetness  0.7142857
#> 15           Native Mean Wetness  0.8571429
#> 16                 % Hydrophytes 37.1428571
```

All of the functions are documented with help pages.

```r
#In R studio, this line of code will bring up documentation in bottom right pane
?all_metrics
```

Cover-Weighted Functions

Cover-Weighted Functions calculate the same metrics but they are weighted by species abundance. Therefore, the input data frame must also have a column named `cover` containing cover values. Cover values can be continuous (i.e. percent cover) or classed (e.g. using the Braun-Blanquet method).

The following tables describe how cover classes are converted to percent cover. Internally, cover-weighted functions convert cover classes to the percent cover midpoint. For this reason, using percent cover is recommended over using cover classes.

Braun-Blanquet Classes	% Cover Range	Midpoint
+	<1%	0.1
1	<5%	2.5
2	5-25%	15

Braun-Blanquet Classes	% Cover Range	Midpoint
3	25-50%	37.5
4	50-75%	62.5
4	75-100%	87.5

Carolina Veg Survey Classes	% Cover Range	Midpoint
1	<0.1	0.1
2	0-1%	0.5
3	1-2%	1.5
4	2-5%	3.5
5	5-10%	7.5
6	10-25%	17.5
7	25-50%	37.5
8	50-75%	62.5
9	75-95%	85
10	95-100%	97.5

Daubenmire Classes	% Cover Range	Midpoint
1	0-5%	2.5
2	5-25%	15
3	25-50%	37.5
4	50-75%	62.5
5	75-95%	85
6	95-100%	97.5

USFS Ecodata Classes	% Cover Range	Midpoint
1	<1%	0.5
3	1.1-5%	3
10	5.1-15%	10
20	15.1-25%	20
30	25.1-35%	30
40	35.1-45%	40
50	45.1-55%	50
60	55.1-65%	60
70	65.1-75%	70

USFS Ecodata Classes	% Cover Range	Midpoint
80	75.1-85%	80
90	85.1-95%	90
98	95.1-100%	98

Cover-Weighted functions come in two flavors: Transect-level and plot-level. Transect-level metrics are those that calculate a metric for an entire transect, which typically includes multiple plots. `transect_summary` and `plot_summary` are both always calculated at the transect-level. Plot-level metrics calculate a metric for a single plot. `cover_mean_c` and `cover_FQI` can be transect-level or plot-level. It is up to the user to decide if they are calculating a transect-level or a plot-level metric.

To calculate `cover_mean_c` and `cover_FQI` at the transect-level, set `allow_duplicate = TRUE`, because different plots along the transect may contain the same species. It is also highly recommended to include a plot ID column and set the `plot_id` argument to be equal to that column name. This will allow duplicate species between plots but not allow duplication within plots.

To calculate `cover_mean_c` and `cover_FQI` at the plot-level, set `allow_duplicate = FALSE`. There is no need to set the `plot_id` argument because duplicate species will not be allowed under any circumstance.

If duplicated species are found where they are not supposed to be, the duplicated entries will only be counted once and their cover values will be added together. The user will also receive a message stating duplicates have been removed.

Function Arguments

Cover-Weighted Functions have a few additional arguments:

- **cover_class**: A character string representing the cover method used. Acceptable cover methods are: `"percent_cover"`, `"carolina_veg_survey"`, `"braun-blanquet"`, `"doubinmire"`, and `"usfs_ecodata"`. `"percent_cover"` is the default and is recommended.
- **allow_duplicates**: Boolean (TRUE or FALSE). If TRUE, allow duplicate entries of the same species. If FALSE, do not allow species duplication. See cover-weighted function description. Setting `allow_duplicates` to TRUE is best for calculating metrics for multiple plots which potentially contain the same species. Setting `allow_duplicates` to FALSE is best for calculating metrics for a single plot, where each species is entered once along with its total cover value.
- **plot_id**: A character string representing the column in `x` that contains plot identification values. `plot_id` is a required argument in `plot_summary`, where it acts as a grouping variable. `plot_id` is optional but recommended for cover-weighted functions and frequency functions. If `plot_id` is set in a cover-weighted function or a frequency function, it only prevents duplicates from occurring in the same plot. It does not act as a grouping variable.

Functions

```
#first make a hypothetical plot with cover values
plot <- data.frame(acronym  = c("ABEESC", "ABIBAL", "AMMBRE", "ANTELE"),
                   name = c("Abelmoschus esculentus",
                            "Abies balsamea", "Ammophila breviligulata",
                            "Anticlea elegans; zigadenus glaucus"),
                   cover = c(50, 4, 20, 30))
```

```r
#now make up a transect
transect <- data.frame(acronym  = c("ABEESC", "ABIBAL", "AMMBRE",
                                    "AMMBRE", "ANTELE", "ABEESC",
                                    "ABIBAL", "AMMBRE"),
                       cover = c(50, 4, 20, 30, 30, 40, 7, 60),
                       plot_id = c(1, 1, 1, 1, 2, 2, 2, 2))

#plot cover mean c (no duplicates allowed)
cover_mean_c(plot, key = "acronym", db = "michigan_2014",
             native = FALSE, cover_class = "percent_cover",
             allow_duplicates = FALSE)
#> [1] 4.923077

#transect cover mean c (duplicates allowed along unless in the same plot)
cover_mean_c(transect, key = "acronym", db = "michigan_2014",
             native = FALSE, cover_class = "percent_cover",
             allow_duplicates = TRUE, plot_id = "plot_id")
#> Duplicate entries detected in the same plot. Duplicates in the same plot will be counted once.
         Cover values of duplicate species will be added together.
#> [1] 6.394834

#cover-weighted FQI
#you can choose to allow duplicates depending on if species are in a single plot
cover_FQI(transect, key = "acronym", db = "michigan_2014", native = FALSE,
          cover_class = "percent_cover",
          allow_duplicates = TRUE)
#> [1] 11.66145

#transect summary function (always allows duplicates)
transect_summary(transect, key = "acronym", db = "michigan_2014")
#>                            metrics     values
#> 1          Total Species Richness  4.0000000
#> 2         Native Species Richness  3.0000000
#> 3     Introduced Species Richness  1.0000000
#> 4    % of Species with no C Value  0.0000000
#> 5     % of Species with 0 C Value 25.0000000
#> 6   % of Species with 1-3 C Value 25.0000000
#> 7   % of Species with 4-6 C Value  0.0000000
#> 8  % of Species with 7-10 C Value 50.0000000
#> 9                          Mean C  5.7500000
#> 10                 Native Mean C  7.6666667
#> 11          Cover-Weighted Mean C  5.8307255
#> 12   Cover-Weighted Native Mean C  9.4665127
#> 13                      Total FQI 11.5000000
#> 14                     Native FQI 13.2790562
#> 15             Cover-Weighted FQI 11.6614509
#> 16      Cover-Weighted Native FQI 16.3964810
#> 17                   Adjusted FQI 66.3952810
#> 18                   Mean Wetness  1.7500000
#> 19            Native Mean Wetness  0.6666667
#> 20                  % Hydrophytes 12.5000000
```

There is also a plot summary function that summarizes plots along a transect. Data is input as a single data frame containing species per plot. This data frame must also have a column representing the plot that the species was observed in.

Because it is sometimes useful to calculate the total amount of bare ground or un-vegetated water in a plot, the user can also choose to include bare ground or water. To get this feature to work, the user must set another argument:

- **allow_non_veg**: Boolean (TRUE or FALSE). If TRUE, allow input to contain un-vegetated ground and un-vegetated water.

If `allow_non_veg` is true, the user can include “UNVEGETATED GROUND” or “UNVEGETATED WATER” along with plant species. They can also use acronyms “GROUND” or “WATER”.

```
#print transect to view structure of data
transect_unveg <- data.frame(acronym = c("GROUND", "ABEESC", "ABIBAL", "AMMBRE",
                                         "ANTELE", "WATER", "GROUND", "ABEESC",
                                         "ABIBAL", "AMMBRE"),
                             cover = c(60, 50, 4, 20, 30, 20, 20, 40, 7, 60),
                             quad_id = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2))

#plot summary of a transect
#duplicates are allowed, unless they are in the same plot
plot_summary(x = transect_unveg, key = "acronym", db = "michigan_2014",
             cover_class = "percent_cover",
             plot_id = "quad_id")
#> Species c("GROUND", "WATER", "GROUND") does not have a wetness coefficient. It will be omitted
         from wetness metric calculations.
#>   quad_id species_richness native_species_richness mean_wetness   mean_c
#> 1       1               4                       3     1.750000 5.750000
#> 2       2               3                       2     3.333333 4.333333
#>   native_mean_c cover_mean_c       FQI native_FQI cover_FQI native_cover_FQI
#> 1      7.666667     4.923077 11.500000  13.279056  9.846154         16.42241
#> 2      6.500000     5.803738  7.505553   9.192388 10.052370         13.10786
#>   adjusted_FQI percent_ground_cover percent_water_cover
#> 1     66.39528                   60                  NA
#> 2     53.07228                   20                  20
```

Relative Functions

Relative functions calculate relative frequency, relative coverage, and relative importance for each species, physiognomic group, or family. `fqacalc` also contains a species summary function that produces a summary of each species’ relative metrics in a data frame. Relative functions always allow duplicate species observations. If a plot ID column is indicated using the `plot_id` argument, duplicates will not be allowed if they occur in the same plot. Relative functions also always allow “ground” and “water” to be included.

Relative functions have one additional argument which tells the functions what to calculate the relative value of:

- **col**: A character string equal to ‘species’, ‘family’, or ‘physiog’.

Relative functions do not distinguish between native and introduced.

```r
#To calculate the relative value of a tree

#relative frequency
relative_frequency(transect, key = "acronym", db = "michigan_2014",
              col = "physiog")
#>   physiognomy relative_frequency
#> 1        forb              37.5
#> 2       grass              37.5
#> 3        tree              25.0

#can also include bare ground and water in the data
#here transect_unveg is data containing ground and water defined previously
relative_frequency(transect_unveg, key = "acronym", db = "michigan_2014",
              col = "physiog")
#>          physiognomy relative_frequency
#> 1 Unvegetated Ground                 20
#> 2  Unvegetated Water                 10
#> 3               forb                 30
#> 4              grass                 20
#> 5               tree                 20

#relative cover
relative_cover(transect, key = "acronym", db = "michigan_2014",
               col = "family", cover_class = "percent_cover")
#>          family relative_cover
#> 1     Malvaceae      37.344398
#> 2 Melanthiaceae      12.448133
#> 3      Pinaceae       4.564315
#> 4       Poaceae      45.643154

#relative importance
relative_importance(transect, key = "acronym", db = "michigan_2014",
                    col = "species", cover_class = "percent_cover")
#>                     name relative_importance
#> 1  ABELMOSCHUS ESCULENTUS            31.17220
#> 2          ABIES BALSAMEA            14.78216
#> 3 AMMOPHILA BREVILIGULATA            41.57158
#> 4        ANTICLEA ELEGANS            12.47407

#species summary (including ground and water)
species_summary(transect_unveg, key = "acronym", db = "michigan_2014",
                cover_class = "percent_cover")
#>   acronym                    name   nativity  c  w frequency coverage
#> 1  ABEESC  ABELMOSCHUS ESCULENTUS introduced  0  5         2       90
#> 2  ABIBAL           ABIES BALSAMEA     native  3  0         2       11
#> 3  AMMBRE AMMOPHILA BREVILIGULATA     native 10  5         2       80
#> 4  ANTELE         ANTICLEA ELEGANS     native 10 -3         1       30
#> 5  GROUND       UNVEGETATED GROUND       <NA>  0 NA         2       80
#> 6   WATER        UNVEGETATED WATER       <NA>  0 NA         1       20
#>   relative_frequency relative_cover relative_importance
#> 1                 20      28.938907           24.469453
```

```
#> 2                 20       3.536977           11.768489
#> 3                 20      25.723473           22.861736
#> 4                 10       9.646302            9.823151
#> 5                 20      25.723473           22.861736
#> 6                 10       6.430868            8.215434
```

```
#physiognomy summary (including ground and water)
physiog_summary(transect_unveg, key = "acronym", db = "michigan_2014",
                cover_class = "percent_cover")
#>          physiognomy frequency coverage relative_frequency relative_cover
#> 1 Unvegetated Ground         2       80                 20      25.723473
#> 2  Unvegetated Water         1       20                 10       6.430868
#> 3               forb         3      120                 30      38.585209
#> 4              grass         2       80                 20      25.723473
#> 5               tree         2       11                 20       3.536977
#>   relative_importance
#> 1           22.861736
#> 2            8.215434
#> 3           34.292605
#> 4           22.861736
#> 5           11.768489
```

Wetness metric

`fqacalc` has one wetness metric function called `mean_w`, which calculates the mean wetness coefficient. The wetness coefficient is based off of the wetland indicator status. Negative wetness coefficients indicate a stronger affinity for wetlands, while positive wetness coefficients indicate an affinity for uplands.

`mean_w` can optionally include species without a C value, as long as they do have a wetness coefficient.

```
#mean wetness
mean_w(crooked_island, key = "acronym", db = "michigan_2014", allow_no_c = FALSE)
#> [1] 0.7142857
```

The End