EDA and Hypothesis Testing

Gillian Tatreau

November 2022

General Set up

In [1]:
```
# import packages
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from distfit import distfit
import scipy.stats as ss
import math
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

In [2]:
```
# create dataframe and view data
df = pd.read_csv("Food_Production.csv")
df.head
```

Out[2]:
```
<bound method NDFrame.head of              Food product  Land use change  Animal
Feed  Farm  Processing  \
0   Wheat & Rye (Bread)              0.1            0.0   0.8          0.2
1          Maize (Meal)              0.3            0.0   0.5          0.1
2         Barley (Beer)              0.0            0.0   0.2          0.1
3               Oatmeal              0.0            0.0   1.4          0.0
4                  Rice              0.0            0.0   3.6          0.1
5              Potatoes              0.0            0.0   0.2          0.0
6               Cassava              0.6            0.0   0.2          0.0
7            Cane Sugar              1.2            0.0   0.5          0.0
8            Beet Sugar              0.0            0.0   0.5          0.2
9          Other Pulses              0.0            0.0   1.1          0.0
10                 Peas              0.0            0.0   0.7          0.0
11                 Nuts             -2.1            0.0   2.1          0.0
12           Groundnuts              0.4            0.0   1.4          0.4
13              Soymilk              0.2            0.0   0.1          0.2
14                 Tofu              1.0            0.0   0.5          0.8
15          Soybean Oil              3.1            0.0   1.5          0.3
16             Palm Oil              3.1            0.0   2.1          1.3
17        Sunflower Oil              0.1            0.0   2.1          0.2
18         Rapeseed Oil              0.2            0.0   2.3          0.2
19            Olive Oil             -0.4            0.0   4.3          0.7
20             Tomatoes              0.4            0.0   0.7          0.0
21       Onions & Leeks              0.0            0.0   0.2          0.0
22      Root Vegetables              0.0            0.0   0.2          0.0
23            Brassicas              0.0            0.0   0.3          0.0
24     Other Vegetables              0.0            0.0   0.2          0.1
25         Citrus Fruit             -0.1            0.0   0.3          0.0
26              Bananas              0.0            0.0   0.3          0.1
27               Apples              0.0            0.0   0.2          0.0
28     Berries & Grapes              0.0            0.0   0.7          0.0
29                 Wine             -0.1            0.0   0.6          0.1
30          Other Fruit              0.1            0.0   0.4          0.0
31               Coffee              3.7            0.0  10.4          0.6
32       Dark Chocolate             14.3            0.0   3.7          0.2
33     Beef (beef herd)             16.3            1.9  39.4          1.3
34    Beef (dairy herd)              0.9            2.5  15.7          1.1
35        Lamb & Mutton              0.5            2.4  19.5          1.1
36             Pig Meat              1.5            2.9   1.7          0.3
37         Poultry Meat              2.5            1.8   0.7          0.4
38                 Milk              0.5            0.2   1.5          0.1
39               Cheese              4.5            2.3  13.1          0.7
40                 Eggs              0.7            2.2   1.3          0.0
41        Fish (farmed)              0.5            0.8   3.6          0.0
42     Shrimps (farmed)              0.2            2.5   8.4          0.0

    Transport  Packging  Retail  Total_emissions  \
0         0.1       0.1     0.1              1.4
1         0.1       0.1     0.0              1.1
2         0.0       0.5     0.3              1.1
3         0.1       0.1     0.0              1.6
4         0.1       0.1     0.1              4.0
5         0.1       0.0     0.0              0.3
6         0.1       0.0     0.0              0.9
7         0.8       0.1     0.0              2.6
8         0.6       0.1     0.0              1.4
9         0.1       0.4     0.0              1.6
10        0.1       0.0     0.0              0.8
11        0.1       0.1     0.0              0.2
12        0.1       0.1     0.0              2.4
```

```
13        0.1       0.1     0.3               1.0
14        0.2       0.2     0.3               3.0
15        0.3       0.8     0.0               6.0
16        0.2       0.9     0.0               7.6
17        0.2       0.9     0.0               3.5
18        0.2       0.8     0.0               3.7
19        0.5       0.9     0.0               6.0
20        0.2       0.1     0.0               1.4
21        0.1       0.0     0.0               0.3
22        0.1       0.0     0.0               0.3
23        0.1       0.0     0.0               0.4
24        0.2       0.0     0.0               0.5
25        0.1       0.0     0.0               0.3
26        0.3       0.1     0.0               0.8
27        0.1       0.0     0.0               0.3
28        0.2       0.2     0.0               1.1
29        0.1       0.7     0.0               1.4
30        0.2       0.0     0.0               0.7
31        0.1       1.6     0.1              16.5
32        0.1       0.4     0.0              18.7
33        0.3       0.2     0.2              59.6
34        0.4       0.3     0.2              21.1
35        0.5       0.3     0.2              24.5
36        0.3       0.3     0.2               7.2
37        0.3       0.2     0.2               6.1
38        0.1       0.1     0.3               2.8
39        0.1       0.2     0.3              21.2
40        0.1       0.2     0.0               4.5
41        0.1       0.1     0.0               5.1
42        0.2       0.3     0.2              11.8

    Eutrophying emissions per 1000kcal (gPO₄eq per 1000kcal)  ...  \
0                                                        NaN  ...
1                                                        NaN  ...
2                                                        NaN  ...
3                                                   4.281357  ...
4                                                   9.514379  ...
5                                                   4.754098  ...
6                                                   0.708419  ...
7                                                   4.820513  ...
8                                                   1.541311  ...
9                                                   5.008798  ...
10                                                  2.173410  ...
11                                                  3.113821  ...
12                                                  2.437931  ...
13                                                       NaN  ...
14                                                       NaN  ...
15                                                       NaN  ...
16                                                  1.207014  ...
17                                                  5.730769  ...
18                                                  2.170814  ...
19                                                  4.214932  ...
20                                                 39.526316  ...
21                                                  8.756757  ...
22                                                  4.351351  ...
23                                                 29.470588  ...
24                                                       NaN  ...
25                                                  7.000000  ...
26                                                  5.483333  ...
27                                                  3.020833  ...
```

```
28                                            10.736842        ...
29                                                  NaN        ...
30                                                  NaN        ...
31                                           197.357143        ...
32                                            16.843327        ...
33                                           110.406593        ...
34                                           133.805861        ...
35                                            30.640379        ...
36                                            31.958159        ...
37                                            26.324324        ...
38                                            17.750000        ...
39                                            25.418605        ...
40                                            15.111111        ...
41                                           131.351955        ...
42                                                  NaN        ...

    Freshwater withdrawals per 100g protein (liters per 100g protein)  \
0                                                 NaN
1                                                 NaN
2                                                 NaN
3                                          371.076923
4                                         3166.760563
5                                          347.647059
6                                                 NaN
7                                                 NaN
8                                                 NaN
9                                          203.503036
10                                         178.487849
11                                        2531.414574
12                                         707.524828
13                                                NaN
14                                                NaN
15                                                NaN
16                                                NaN
17                                                NaN
18                                                NaN
19                                                NaN
20                                        3361.818182
21                                         110.000000
22                                         284.000000
23                                        1085.454545
24                                                NaN
25                                        1378.333333
26                                        1272.222222
27                                        6003.333333
28                                        4196.000000
29                                                NaN
30                                                NaN
31                                          32.375000
32                                        1081.200000
33                                         727.783350
34                                        1375.025329
35                                         900.949525
36                                        1109.888752
37                                         381.062356
38                                        1903.636364
39                                        2538.586957
40                                         520.638068
41                                        1618.636264
42                                                NaN
```

```
    Freshwater withdrawals per kilogram (liters per kilogram)  \
0                                                 NaN
1                                                 NaN
2                                                 NaN
3                                               482.4
4                                              2248.4
5                                                59.1
6                                                 0.0
7                                               620.1
8                                               217.7
9                                               435.7
10                                              396.6
11                                             4133.8
12                                             1852.3
13                                               27.8
14                                                NaN
15                                              414.6
16                                                6.4
17                                             1007.9
18                                              237.7
19                                             2141.8
20                                              369.8
21                                               14.3
22                                               28.4
23                                              119.4
24                                              102.5
25                                               82.7
26                                              114.5
27                                              180.1
28                                              419.6
29                                               78.9
30                                              153.5
31                                               25.9
32                                              540.6
33                                             1451.2
34                                             2714.3
35                                             1802.8
36                                             1795.8
37                                              660.0
38                                              628.2
39                                             5605.2
40                                              577.7
41                                             3691.3
42                                                NaN

    Greenhouse gas emissions per 1000kcal (kgCO₂eq per 1000kcal)  \
0                                                 NaN
1                                                 NaN
2                                                 NaN
3                                            0.945482
4                                            1.207271
5                                            0.628415
6                                            1.355236
7                                            0.911681
8                                            0.515670
9                                            0.524927
10                                           0.283237
11                                           0.069919
12                                           0.556897
```

```
13          NaN
14          NaN
15          NaN
16     0.828054
17     0.407240
18     0.426471
19     0.613122
20    11.000000
21     1.351351
22     1.162162
23     3.000000
24          NaN
25     1.218750
26     1.433333
27     0.895833
28     2.684211
29          NaN
30          NaN
31    50.946429
32     9.023211
33    36.439560
34    12.197802
35    12.529968
36     5.150628
37     5.335135
38     5.250000
39     6.170543
40     3.243056
41     7.614525
42          NaN

    Greenhouse gas emissions per 100g protein (kgCO₂eq per 100g protein)  \
0           NaN
1           NaN
2           NaN
3      1.907692
4      6.267606
5      2.705882
6     14.666667
7           NaN
8           NaN
9      0.836058
10     0.441044
11     0.263319
12     1.233766
13          NaN
14          NaN
15          NaN
16          NaN
17          NaN
18          NaN
19          NaN
20    19.000000
21     3.846154
22     4.300000
23     4.636364
24          NaN
25     6.500000
26     9.555556
27    14.333333
```

```
28                                                    15.300000
29                                                          NaN
30                                                          NaN
31                                                    35.662500
32                                                    93.300000
33                                                    49.889669
34                                                    16.869301
35                                                    19.850075
36                                                     7.608158
37                                                     5.698614
38                                                     9.500000
39                                                    10.815217
40                                                     4.208724
41                                                     5.976759
42                                                          NaN

    Land use per 1000kcal (m² per 1000kcal)  \
0                                       NaN
1                                       NaN
2                                       NaN
3                                  2.897446
4                                  0.759631
5                                  1.202186
6                                  1.858316
7                                  0.581197
8                                  0.521368
9                                  4.565982
10                                 2.156069
11                                 2.107317
12                                 1.570690
13                                      NaN
14                                      NaN
15                                      NaN
16                                 0.273756
17                                 1.997738
18                                 1.202489
19                                 2.976244
20                                 4.210526
21                                 1.054054
22                                 0.891892
23                                 3.235294
24                                      NaN
25                                 2.687500
26                                 3.216667
27                                 1.312500
28                                 4.228070
29                                      NaN
30                                      NaN
31                                38.607143
32                                13.338491
33                               119.490842
34                                15.838828
35                               116.659306
36                                 7.263598
37                                 6.605405
38                                14.916667
39                                22.684755
40                                 4.354167
41                                 4.698324
42                                      NaN
```

```
    Land use per kilogram (m² per kilogram)  \
0                                        NaN
1                                        NaN
2                                        NaN
3                                       7.60
4                                       2.80
5                                       0.88
6                                       1.81
7                                       2.04
8                                       1.83
9                                      15.57
10                                      7.46
11                                     12.96
12                                      9.11
13                                      0.66
14                                       NaN
15                                     10.52
16                                      2.42
17                                     17.66
18                                     10.63
19                                     26.31
20                                      0.80
21                                      0.39
22                                      0.33
23                                      0.55
24                                      0.38
25                                      0.86
26                                      1.93
27                                      0.63
28                                      2.41
29                                      1.78
30                                      0.89
31                                     21.62
32                                     68.96
33                                    326.21
34                                     43.24
35                                    369.81
36                                     17.36
37                                     12.22
38                                      8.95
39                                     87.79
40                                      6.27
41                                      8.41
42                                       NaN

    Land use per 100g protein (m² per 100g protein)  \
0                                                NaN
1                                                NaN
2                                                NaN
3                                           5.846154
4                                           3.943662
5                                           5.176471
6                                          20.111111
7                                                NaN
8                                                NaN
9                                           7.272303
10                                          3.357336
11                                          7.936314
12                                          3.479756
```

```
13                                                NaN
14                                                NaN
15                                                NaN
16                                                NaN
17                                                NaN
18                                                NaN
19                                                NaN
20                                           7.272727
21                                           3.000000
22                                           3.300000
23                                           5.000000
24                                                NaN
25                                          14.333333
26                                          21.444444
27                                          21.000000
28                                          24.100000
29                                                NaN
30                                                NaN
31                                          27.025000
32                                         137.920000
33                                         163.595787
34                                          21.904762
35                                         184.812594
36                                          10.729295
37                                           7.055427
38                                          27.121212
39                                          39.759964
40                                           5.650685
41                                           3.687788
42                                                NaN

    Scarcity-weighted water use per kilogram (liters per kilogram)  \
0                                                 NaN
1                                                 NaN
2                                                 NaN
3                                             18786.2
4                                             49576.3
5                                              2754.2
6                                                 0.0
7                                             16438.6
8                                              9493.3
9                                             22477.4
10                                            27948.2
11                                           229889.8
12                                            61797.9
13                                              955.6
14                                                NaN
15                                            14888.2
16                                               36.2
17                                            36369.4
18                                            10593.7
19                                           177480.2
20                                             5335.7
21                                              932.0
22                                              929.2
23                                             8455.1
24                                             4911.4
25                                             4662.7
26                                              661.9
27                                            12948.6
```

```
28                                               21162.1
29                                                1149.3
30                                                9533.1
31                                                 337.0
32                                                2879.2
33                                               34732.5
34                                              119805.2
35                                              141925.0
36                                               66867.4
37                                               14177.9
38                                               19786.3
39                                              180850.6
40                                               17982.7
41                                               41572.2
42                                                   NaN

    Scarcity-weighted water use per 100g protein (liters per 100g protein)  \
0                                                 NaN
1                                                 NaN
2                                                 NaN
3                                        14450.923080
4                                        69825.774650
5                                        16201.176470
6                                                 NaN
7                                                 NaN
8                                                 NaN
9                                        10498.552080
10                                       12577.947790
11                                      140777.587300
12                                       23605.003820
13                                                NaN
14                                                NaN
15                                                NaN
16                                                NaN
17                                                NaN
18                                                NaN
19                                                NaN
20                                       48506.363640
21                                        7169.230769
22                                        9292.000000
23                                       76864.545450
24                                                NaN
25                                       77711.666670
26                                        7354.444444
27                                      431620.000000
28                                      211621.000000
29                                                NaN
30                                                NaN
31                                         421.250000
32                                        5758.400000
33                                       17418.505520
34                                       60691.590680
35                                       70927.036480
36                                       41327.194070
37                                        8185.854503
38                                       59958.484850
39                                       81906.974640
40                                       16206.470800
41                                       18229.423370
42                                                NaN
```

```
    Scarcity-weighted water use per 1000kcal (liters per 1000 kilocalories)
0                                                 NaN
1                                                 NaN
2                                                 NaN
3                                         7162.104461
4                                        13449.891480
5                                         3762.568306
6                                                 NaN
7                                         4683.361823
8                                         2704.643875
9                                                 NaN
10                                                NaN
11                                       37380.455280
12                                       10654.810340
13                                                NaN
14                                                NaN
15                                                NaN
16                                           4.095023
17                                        4114.185520
18                                        1198.382353
19                                       20076.945700
20                                       28082.631580
21                                        2518.918919
22                                        2511.351351
23                                       49735.882350
24                                                NaN
25                                       14570.937500
26                                        1103.166667
27                                       26976.250000
28                                       37126.491230
29                                                NaN
30                                                NaN
31                                         601.785714
32                                         556.905222
33                                       12722.527470
34                                       43884.688640
35                                       44771.293380
36                                       27977.991630
37                                        7663.729730
38                                       32977.166670
39                                       46731.421190
40                                       12487.986110
41                                       23224.692740
42                                                NaN

[43 rows x 23 columns]>
```

In [3]:
```
# list of column names, to choose variables to focus on
df.columns
```

Out[3]:
```
Index(['Food product', 'Land use change', 'Animal Feed', 'Farm', 'Processing',
       'Transport', 'Packging', 'Retail', 'Total_emissions',
       'Eutrophying emissions per 1000kcal (gPO₄eq per 1000kcal)',
       'Eutrophying emissions per kilogram (gPO₄eq per kilogram)',
       'Eutrophying emissions per 100g protein (gPO₄eq per 100 grams protei
n)',
       'Freshwater withdrawals per 1000kcal (liters per 1000kcal)',
       'Freshwater withdrawals per 100g protein (liters per 100g protein)',
       'Freshwater withdrawals per kilogram (liters per kilogram)',
       'Greenhouse gas emissions per 1000kcal (kgCO₂eq per 1000kcal)',
       'Greenhouse gas emissions per 100g protein (kgCO₂eq per 100g protein)',
       'Land use per 1000kcal (m² per 1000kcal)',
       'Land use per kilogram (m² per kilogram)',
       'Land use per 100g protein (m² per 100g protein)',
       'Scarcity-weighted water use per kilogram (liters per kilogram)',
       'Scarcity-weighted water use per 100g protein (liters per 100g protei
n)',
       'Scarcity-weighted water use per 1000kcal (liters per 1000 kilocalorie
s)'],
      dtype='object')
```

In [4]:
```
# new data frame that renames chosen variables to be easier to use
df2 = df.rename(columns={"Food product" : "food", "Total_emissions" : "total_er
                         "Eutrophying emissions per 1000kcal (gPO₄eq per 1000k
                         "Eutrophying emissions per 100g protein (gPO₄eq per 1(
                         "Greenhouse gas emissions per 1000kcal (kgCO₂eq per 1(
                         "Greenhouse gas emissions per 100g protein (kgCO₂eq pe
                         "Land use per 1000kcal (m² per 1000kcal)" : "land_kca
                         "Land use per 100g protein (m² per 100g protein)" : "
                         "Scarcity-weighted water use per 1000kcal (liters per
                         "Scarcity-weighted water use per 100g protein (liters
```

In [5]:
```
# name of columns, with new names of chosen variables
df2.columns
```

Out[5]:
```
Index(['food', 'Land use change', 'Animal Feed', 'Farm', 'Processing',
       'Transport', 'Packging', 'Retail', 'total_em', 'eutro_kcal',
       'Eutrophying emissions per kilogram (gPO₄eq per kilogram)',
       'eutro_protein',
       'Freshwater withdrawals per 1000kcal (liters per 1000kcal)',
       'Freshwater withdrawals per 100g protein (liters per 100g protein)',
       'Freshwater withdrawals per kilogram (liters per kilogram)',
       'greenhouse_kcal', 'greenhouse_protein', 'land_kcal',
       'Land use per kilogram (m² per kilogram)', 'land_protein',
       'Scarcity-weighted water use per kilogram (liters per kilogram)',
       'water_protein', 'water_kcal'],
      dtype='object')
```

Creating Scenarios: all food products, vegetarian food products, vegan food products

In [6]:
```
# creating variables to use: df2 will have all food products,
# veg will have vegetarian food products (including eggs etc that are animal p
# and vegan which excludes all animal products entirely
all_animal = df2.index.isin([33, 34, 35, 36, 37, 38, 39, 40, 41, 42])
animal_veg = df2.index.isin([33, 35, 36, 37, 41, 42])
```

```
veg = df2[~animal_veg]
vegan = df2[~all_animal]
```

Explanation of variables

- **food** is the food product
- **total_em** is the total emissions of greenhouse gas per kg of food product(Kg CO2) totaled over every aspect of food production (total emissions of greenhouse gas omitted by producing the food item)
- **eutro_kcal** is the measure of eutrophication (which is caused by land runoff during production) measured in g PO equivalent per 1000 Calories for each food product
- **eutro_protein** is the measure of eutrophication measured in g PO equivalent per 100 g of protein for each food product
- **greenhouse_kcal** is the greenhouse gas emissions (kg CO2 equivalent) per 1000 Calories for each food product
- **greenhouse_protein** is the greenhouse gas emissions (kg CO2 equivalent) per 100 g of protein for each food product
- **land_kcal** is the land used in production of each food product, measured in m2 per 1000 Calories of food
- **land_protein** is the land used in production of each food product, measured in m2 per 100 g of protein in each food product
- **water_kcal** is the amount of water used in production of each food product, weighted for water scarcity, measured in liters per 1000 Calories of food
- **water_protein** is the amount of water used in production of each food product, weighted for water scarcity, measured in liters per 100 g of protein in each food product

Descriptive stats and histograms of chosen variables

In [7]:
```
# function to calculate optimal bin width and number of bins for each variable
def bin_count(column, data):
    q1 = column.quantile(0.25)
    q3 = column.quantile(0.75)
    iqr = q3 - q1
    bin_width = (2 * iqr) / (len(column) ** (1 / 3))
    binnum = int(np.ceil((column.max() - column.min()) / bin_width))
    print(binnum, bin_width)
```

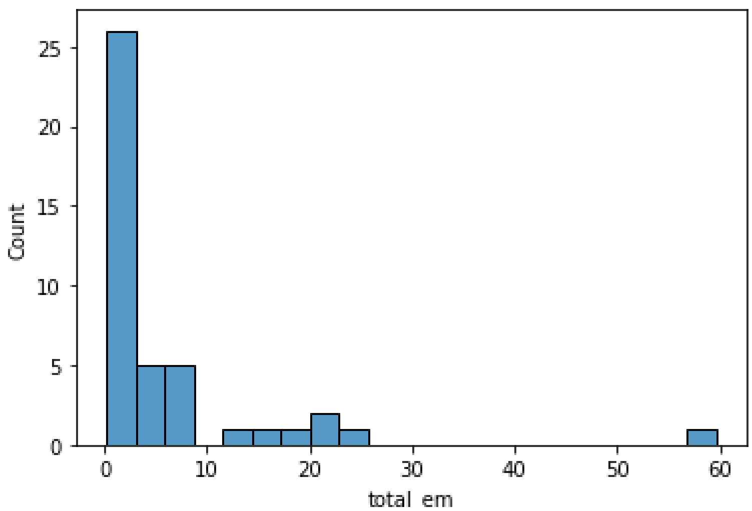
In [8]:
```
# outlier function
def outlier(column, data):
    f = data.food
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)

    IQR = Q3 - Q1

    outlier_name = f[((column<(Q1-1.5*IQR))|(column>(Q3+1.5*IQR)))]

    outlier_val = column[((column<(Q1-1.5*IQR))|(column>(Q3+1.5*IQR)))]
```

```
    outliers = pd.concat([outlier_name, outlier_val], axis=1)

    print(outliers)
```

total_em

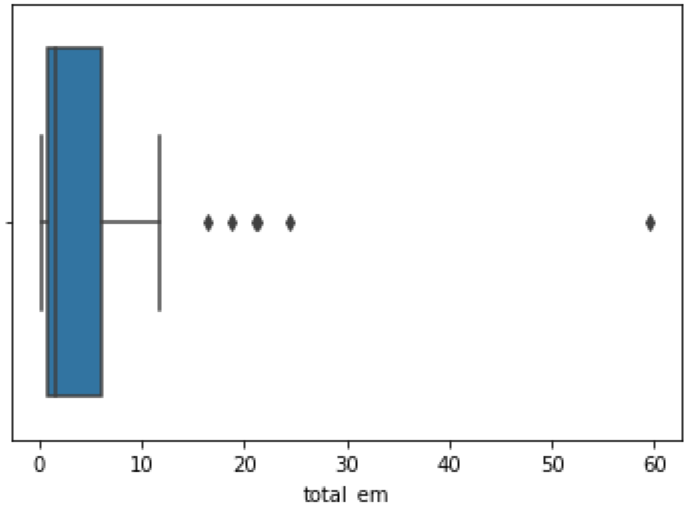
In [9]:
```
bin_count(df2.total_em, df2)
sns.histplot(data=df2, x="total_em", bins = 21)
print(df2.total_em.describe(), "\nNumber of Na's: ", df2.total_em.isna().sum()
```

```
21 2.940002769443513
count    43.000000
mean      5.972093
std      10.501753
min       0.200000
25%       0.850000
50%       1.600000
75%       6.000000
max      59.600000
Name: total_em, dtype: float64
Number of Na's:  0
```



In [10]:
```
sns.boxplot(data=df2, x="total_em")
outlier(df2.total_em, df2)
```

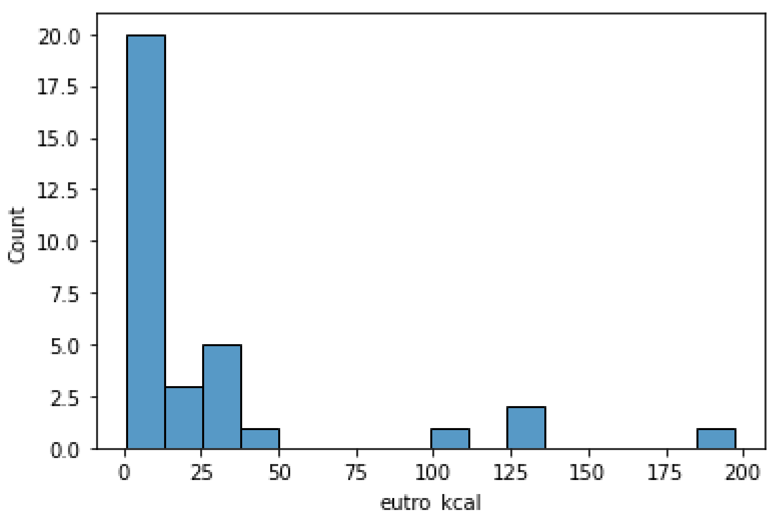
```
                  food  total_em
31              Coffee      16.5
32      Dark Chocolate      18.7
33    Beef (beef herd)      59.6
34   Beef (dairy herd)      21.1
35       Lamb & Mutton      24.5
39              Cheese      21.2
```



eutro_kcal

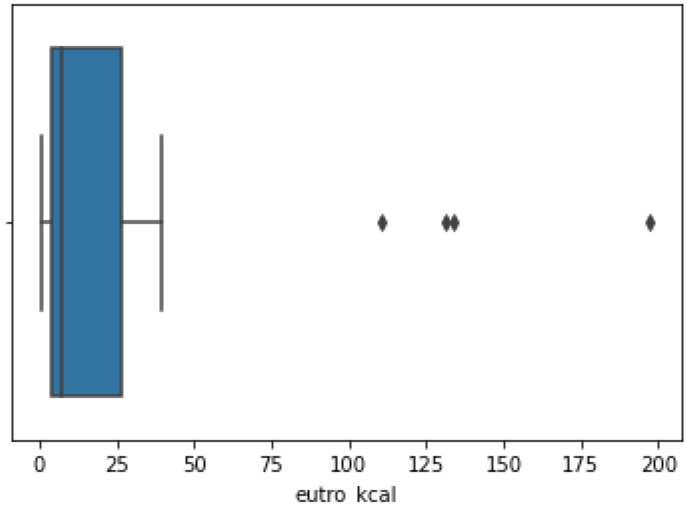
In [11]:
```
bin_count(df2.eutro_kcal, df2)
sns.histplot(data=df2, x="eutro_kcal", bins = 16)
print(df2.eutro_kcal.describe(), "\nNumber of Na's: ", df2.eutro_kcal.isna().su
```

```
16 12.621684325851025
count     33.000000
mean      27.181547
std       46.445959
min        0.708419
25%        4.214932
50%        7.000000
75%       26.324324
max      197.357143
Name: eutro_kcal, dtype: float64
Number of Na's:  10
```



In [12]:
```
sns.boxplot(data=df2, x="eutro_kcal")
outlier(df2.eutro_kcal, df2)
```

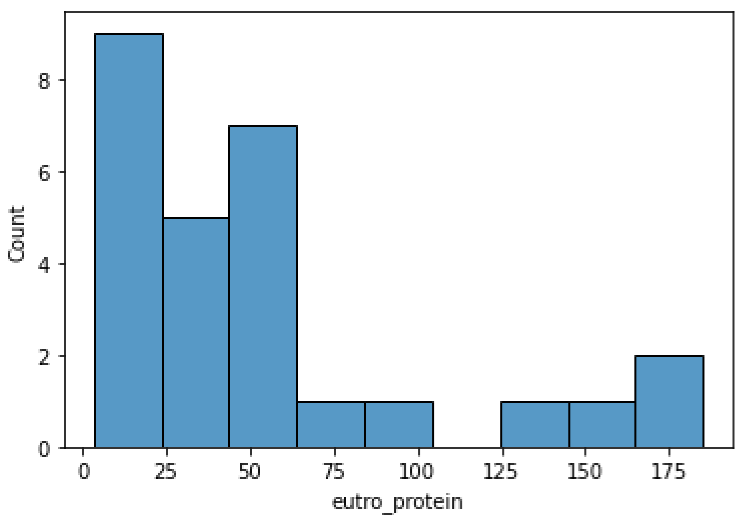
```
                  food  eutro_kcal
31              Coffee  197.357143
33    Beef (beef herd)  110.406593
34   Beef (dairy herd)  133.805861
41       Fish (farmed)  131.351955
```



eutro_protein

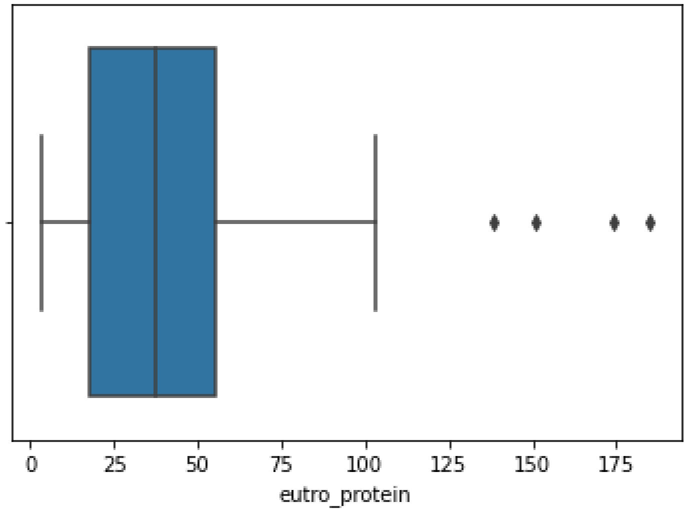
In [13]:
```
bin_count(df2.eutro_protein, df2)
sns.histplot(data=df2, x="eutro_protein", bins = 9)
print(df2.eutro_protein.describe(), "\nNumber of Na's: ", df2.eutro_protein.is
```

```
9 21.374589583957793
count     27.000000
mean      52.771953
std       52.033823
min        3.384338
25%       17.855335
50%       37.333333
75%       55.297183
max      185.050659
Name: eutro_protein, dtype: float64
Number of Na's:  16
```



In [14]:
```
sns.boxplot(data=df2, x="eutro_protein")
outlier(df2.eutro_protein, df2)
```

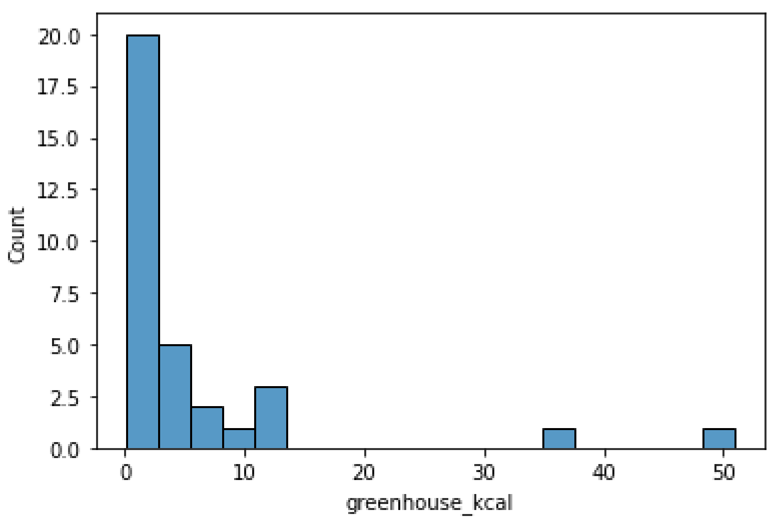
```
                  food  eutro_protein
31              Coffee     138.150000
32      Dark Chocolate     174.160000
33    Beef (beef herd)     151.158475
34   Beef (dairy herd)     185.050659
```



greenhouse_kcal

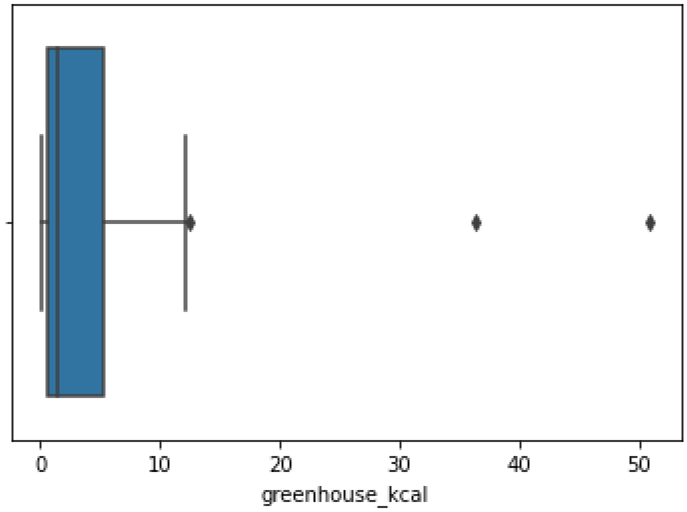
In [15]:
```
bin_count(df2.greenhouse_kcal, df2)
sns.histplot(data=df2, x="greenhouse_kcal", bins = 19)
print(df2.greenhouse_kcal.describe(), "\nNumber of Na's: ", df2.greenhouse_kca
```

```
19 2.6869455042630506
count    33.000000
mean      5.633943
std      10.613575
min       0.069919
25%       0.628415
50%       1.351351
75%       5.335135
max      50.946429
Name: greenhouse_kcal, dtype: float64
Number of Na's:  10
```



In [16]:
```
sns.boxplot(data=df2, x="greenhouse_kcal")
outlier(df2.greenhouse_kcal, df2)
```

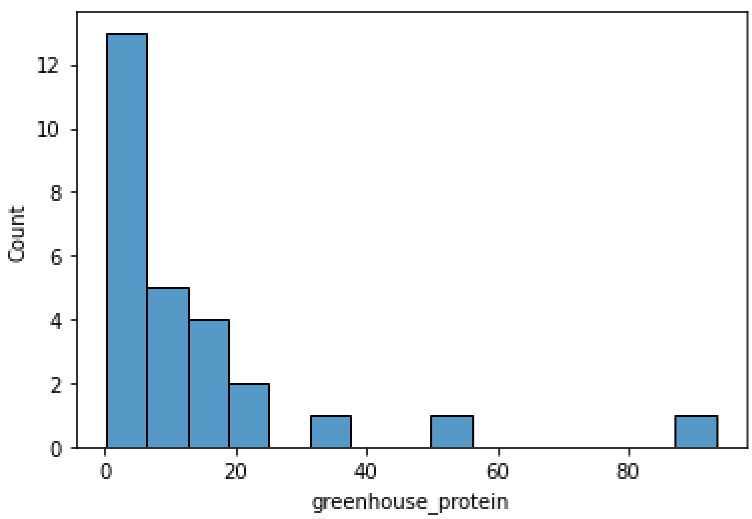
```
                 food  greenhouse_kcal
31             Coffee        50.946429
33  Beef (beef herd)        36.439560
35      Lamb & Mutton        12.529968
```



greenhouse_protein

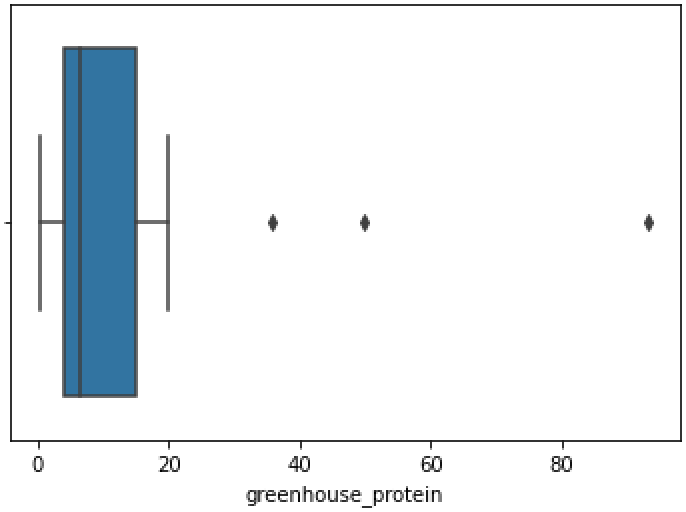
In [17]:
```
bin_count(df2.greenhouse_protein, df2)
sns.histplot(data=df2, x="greenhouse_protein", bins = 15)
print(df2.greenhouse_protein.describe(), "\nNumber of Na's: ", df2.greenhouse_
```

```
15 6.254438856879786
count    27.000000
mean     13.524906
std      19.427462
min       0.263319
25%       4.027439
50%       6.500000
75%      14.983333
max      93.300000
Name: greenhouse_protein, dtype: float64
Number of Na's:  16
```



In [18]:
```
sns.boxplot(data=df2, x="greenhouse_protein")
outlier(df2.greenhouse_protein, df2)
```

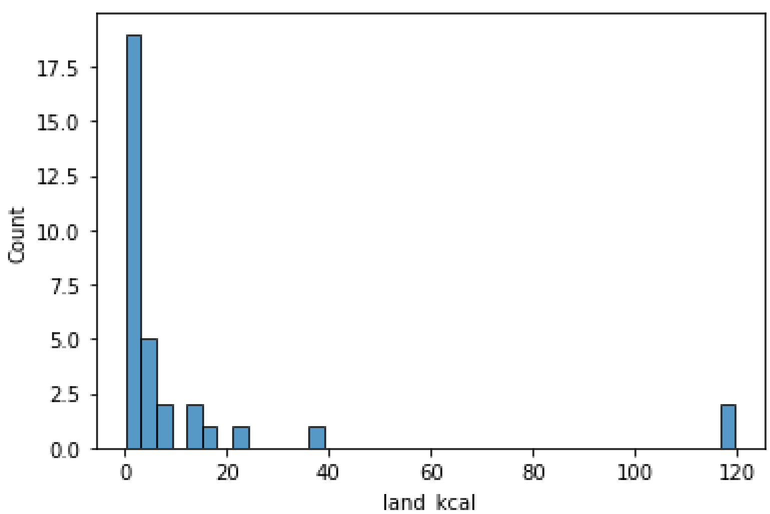
```
                 food  greenhouse_protein
31             Coffee           35.662500
32     Dark Chocolate           93.300000
33  Beef (beef herd)           49.889669
```



land_kcal

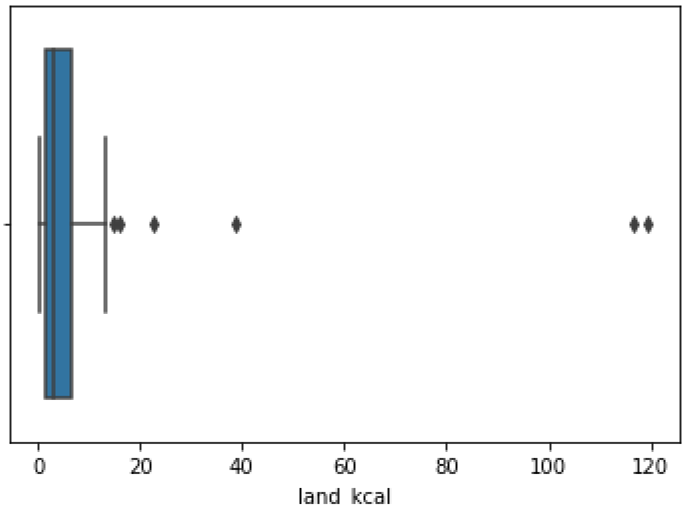
In [19]:
```
bin_count(df2.land_kcal, df2)
sns.histplot(data=df2, x="land_kcal", bins = 40)
print(df2.land_kcal.describe(), "\nNumber of Na's: ", df2.land_kcal.isna().sum
```

```
40 3.0215837959422407
count     33.000000
mean      12.423165
std       28.348693
min        0.273756
25%        1.312500
50%        2.976244
75%        6.605405
max      119.490842
Name: land_kcal, dtype: float64
Number of Na's:  10
```



In [20]:
```
sns.boxplot(data=df2, x="land_kcal")
outlier(df2.land_kcal, df2)
```

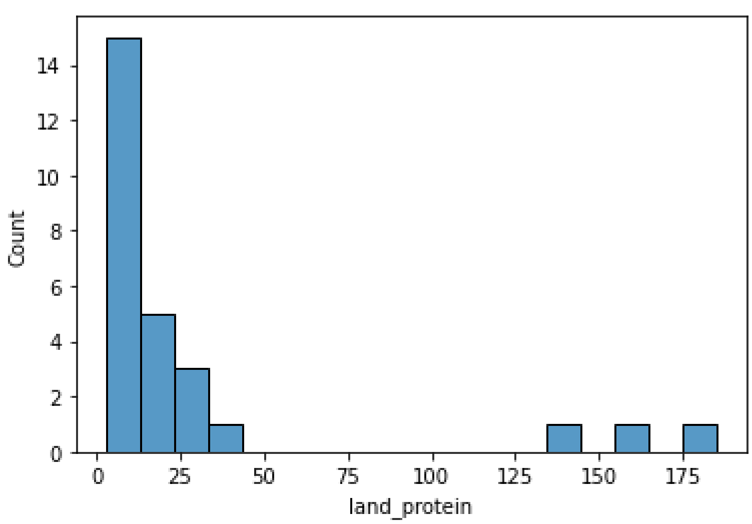
```
                 food   land_kcal
31             Coffee   38.607143
33   Beef (beef herd)  119.490842
34  Beef (dairy herd)   15.838828
35      Lamb & Mutton  116.659306
38               Milk   14.916667
39             Cheese   22.684755
```



land_protein

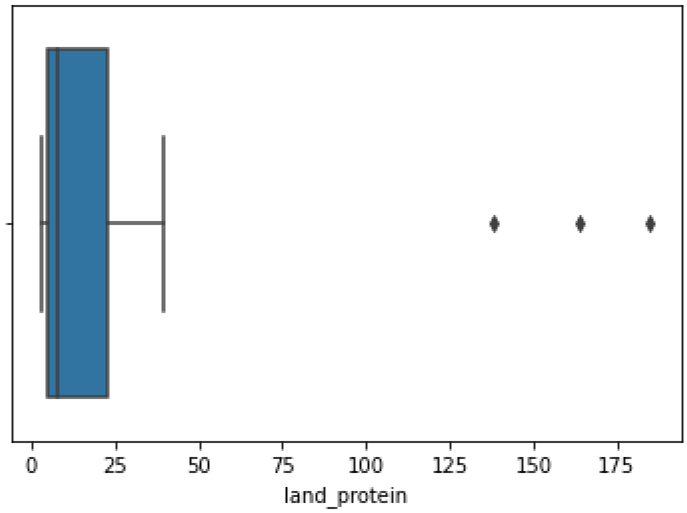
In [21]:
```
bin_count(df2.land_protein, df2)
sns.histplot(data=df2, x="land_protein", bins = 18)
print(df2.land_protein.describe(), "\nNumber of Na's: ", df2.land_protein.isna
```

```
18 10.226725794340677
count     27.000000
mean      29.105042
std       49.307339
min        3.000000
25%        5.088235
50%        7.936314
75%       23.002381
max      184.812594
Name: land_protein, dtype: float64
Number of Na's:  16
```



In [22]:
```
sns.boxplot(data=df2, x="land_protein")
outlier(df2.land_protein, df2)
```

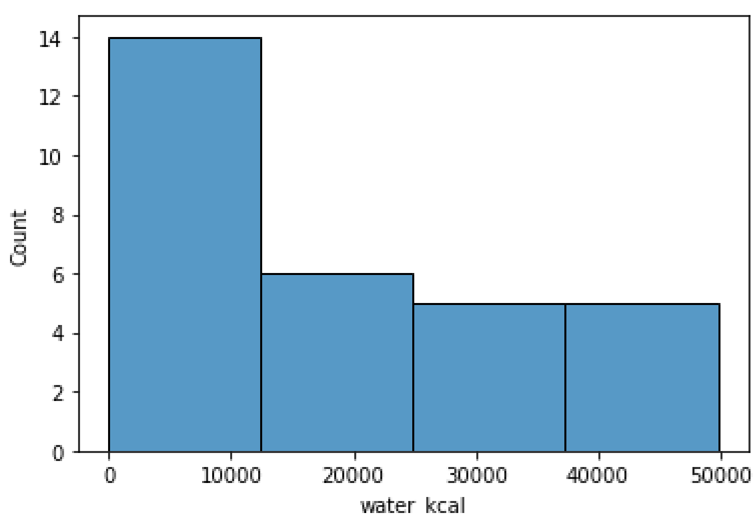
```
                food  land_protein
32    Dark Chocolate    137.920000
33  Beef (beef herd)    163.595787
35      Lamb & Mutton    184.812594
```



water_kcal

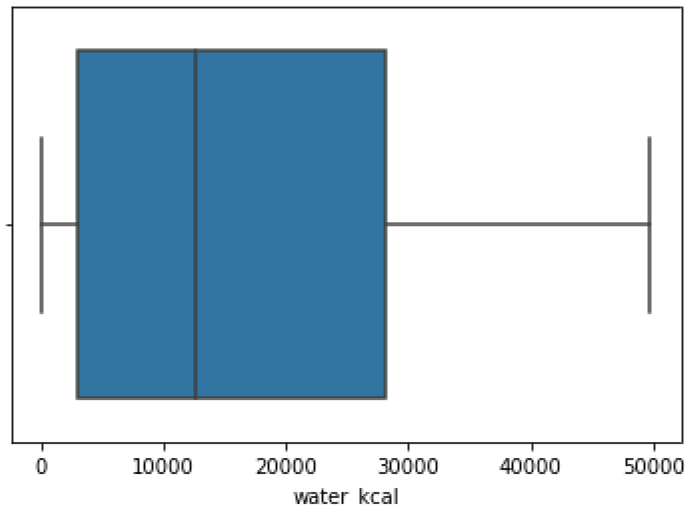
In [23]:
```
bin_count(df2.water_kcal, df2)
sns.histplot(data=df2, x="water_kcal", bins = 4)
print(df2.water_kcal.describe(), "\nNumber of Na's: ", df2.water_kcal.isna().su
```

```
4 14321.722040903755
count       30.000000
mean     17380.575408
std      16232.080209
min          4.095023
25%       2969.124983
50%      12605.256790
75%      28056.471593
max      49735.882350
Name: water_kcal, dtype: float64
Number of Na's:  13
```



In [24]:
```
sns.boxplot(data=df2, x="water_kcal")
outlier(df2.water_kcal, df2)
```

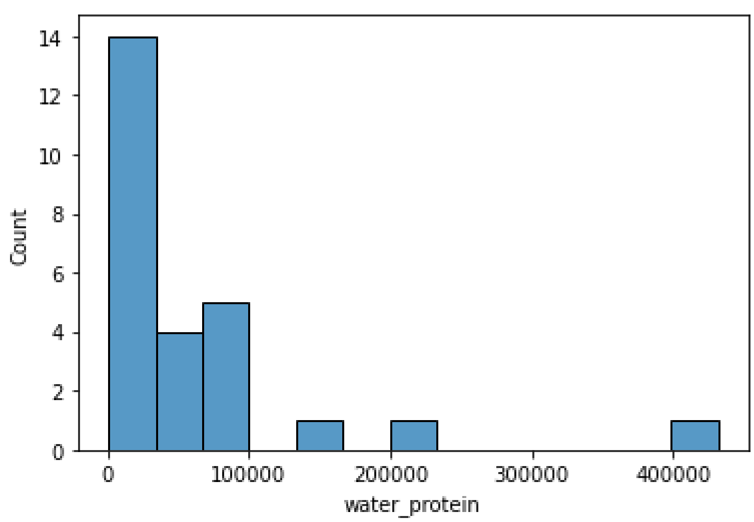
```
Empty DataFrame
Columns: [food, water_kcal]
Index: []
```



water_protein

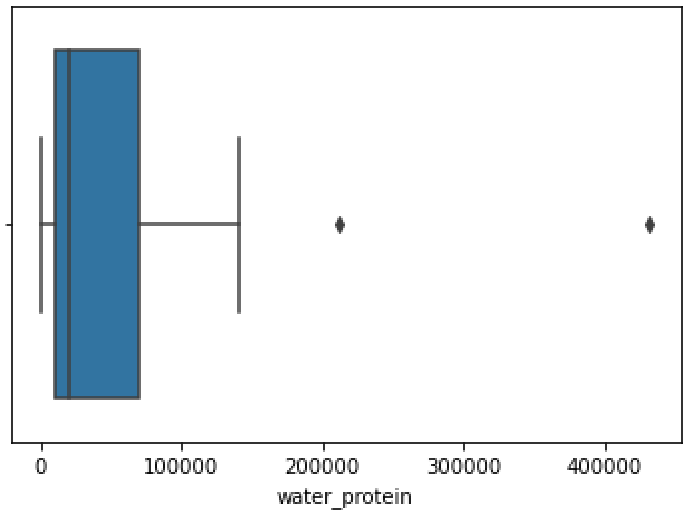
In [25]:
```
bin_count(df2.water_protein, df2)
sns.histplot(data=df2, x="water_protein", bins = 13)
print(df2.water_protein.describe(), "\nNumber of Na's: ", df2.water_protein.isı
```

```
13 34043.13126120608
count        26.000000
mean      59196.438503
std       89928.189299
min         421.250000
25%       11018.401008
50%       20917.213595
75%       70651.721023
max      431620.000000
Name: water_protein, dtype: float64
Number of Na's:  17
```



In [26]:
```
sns.boxplot(data=df2, x="water_protein")
outlier(df2.water_protein, df2)
```

```
              food  water_protein
27          Apples       431620.0
28  Berries & Grapes       211621.0
```

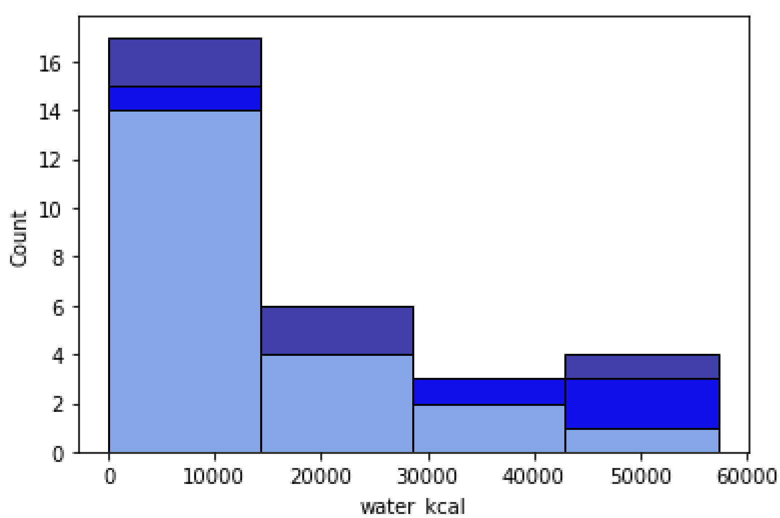


PMFs and CDFs

water_kcal

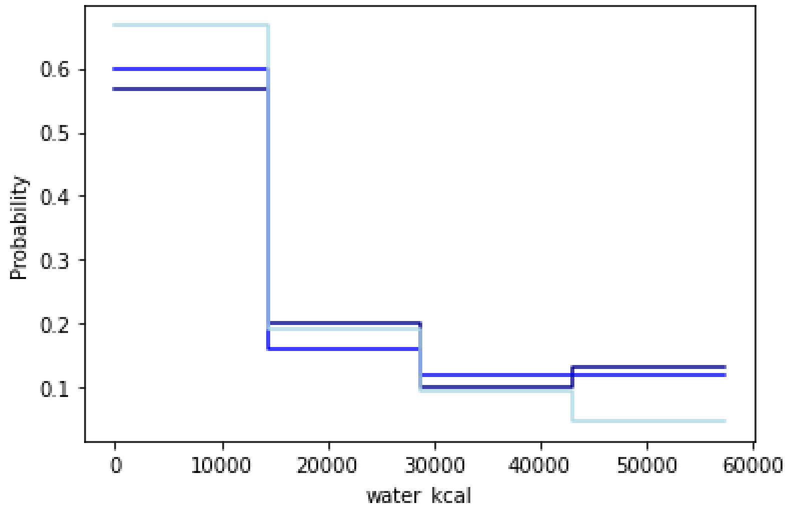
In [27]:
```
sns.histplot(data=df2, x="water_kcal", color = "darkblue", binwidth=14321.7)
sns.histplot(data=veg, x="water_kcal", color = "blue", binwidth=14321.7)
sns.histplot(data=vegan, x="water_kcal", color = "lightblue", binwidth=14321.7
```

Out[27]: <AxesSubplot:xlabel='water_kcal', ylabel='Count'>



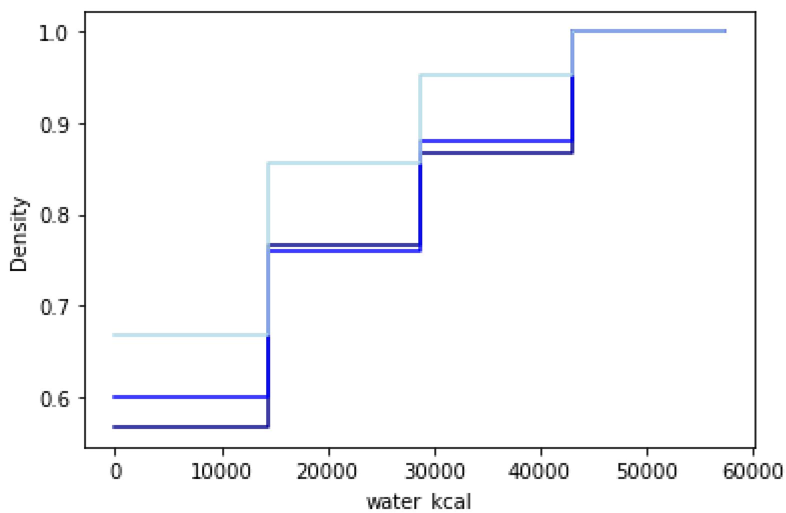
In [28]:
```
# pmf
sns.histplot(data=df2, x="water_kcal", color = "darkblue", stat = "probability"
sns.histplot(data=veg, x="water_kcal", color = "blue", stat = "probability", e
sns.histplot(data=vegan, x="water_kcal", color = "lightblue", stat = "probabil
```

Out[28]: <AxesSubplot:xlabel='water_kcal', ylabel='Probability'>



In [29]:
```
# cdf
sns.histplot(data=df2, x="water_kcal", element="step", fill=False, cumulative=T
sns.histplot(data=veg, x="water_kcal", element="step", fill=False, cumulative=T
sns.histplot(data=vegan, x="water_kcal", element="step", fill=False, cumulative
```

Out[29]: <AxesSubplot:xlabel='water_kcal', ylabel='Density'>

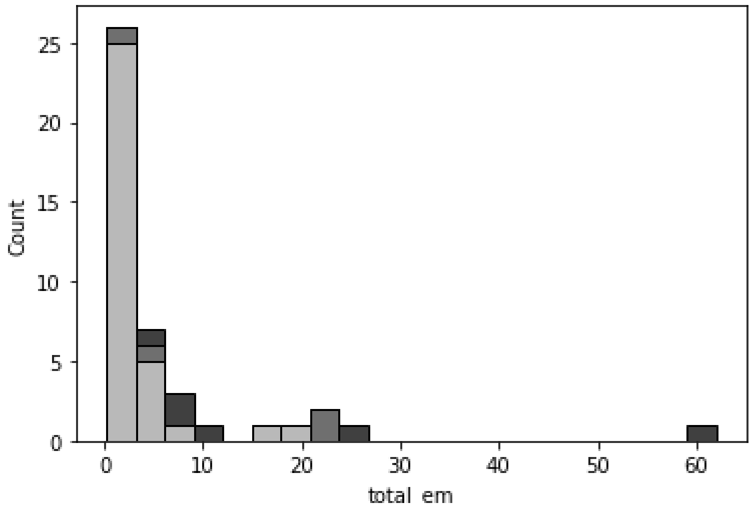


It appear that vegan food products use less water in general than the other scenarios, with vegetarian products using less water in the first and third bins and more in the second.

total_em

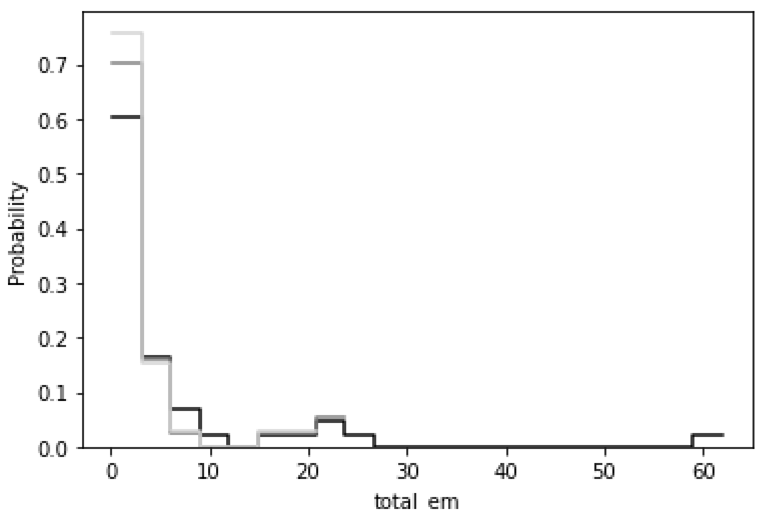
In [30]:
```
sns.histplot(data=df2, x="total_em", color = "black", binwidth = 2.94)
sns.histplot(data=veg, x="total_em", color = "grey", binwidth = 2.94)
sns.histplot(data=vegan, x="total_em", color = "lightgrey", binwidth = 2.94)
```

Out[30]: <AxesSubplot:xlabel='total_em', ylabel='Count'>



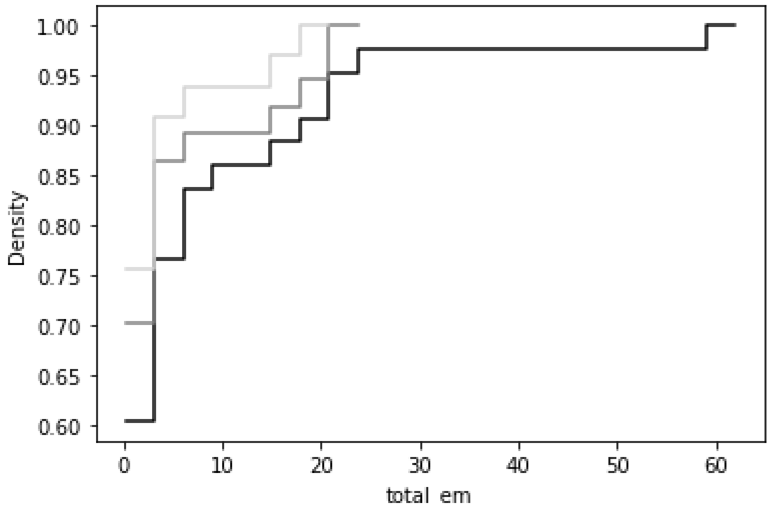
In [31]:
```
# pmf
sns.histplot(data=df2, x="total_em", color = "black", stat = "probability", ele
sns.histplot(data=veg, x="total_em", color = "grey", stat = "probability", elem
sns.histplot(data=vegan, x="total_em", color = "lightgrey", stat = "probability
```

Out[31]: <AxesSubplot:xlabel='total_em', ylabel='Probability'>



In [32]:
```
# cdf
sns.histplot(data=df2, x="total_em", element="step", fill=False, cumulative=Tru
sns.histplot(data=veg, x="total_em", element="step", fill=False, cumulative=Tru
sns.histplot(data=vegan, x="total_em", element="step", fill=False, cumulative=T
```

Out[32]: <AxesSubplot:xlabel='total_em', ylabel='Density'>

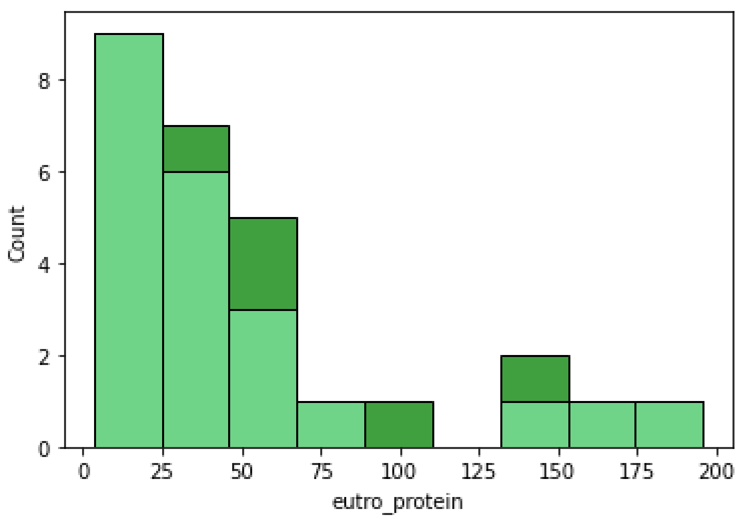


It appears that vegan food products produce fewer emissions during their production, followed by vegetarian products, then all food products.

eutro_protein

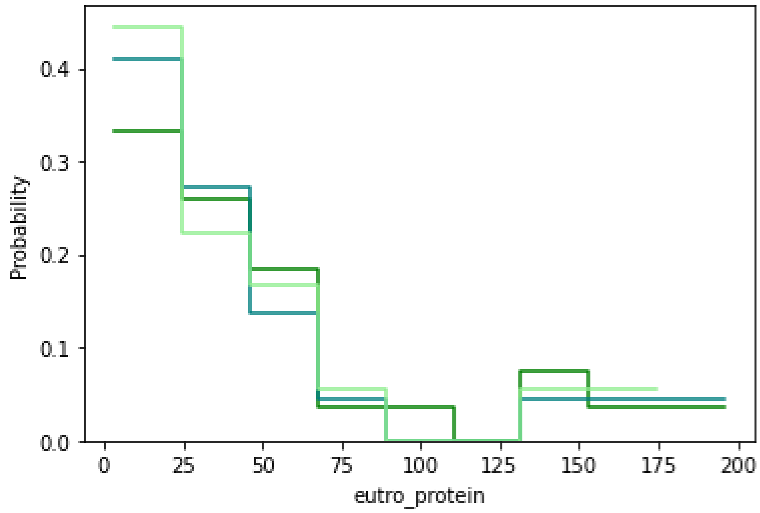
In [33]:
```
sns.histplot(data=df2, x="eutro_protein", binwidth = 21.37, color = "green")
sns.histplot(data=veg, x="eutro_protein", binwidth = 21.37, color = "teal")
sns.histplot(data=veg, x="eutro_protein", binwidth = 21.37, color = "lightgreen
```

Out[33]:
```
<AxesSubplot:xlabel='eutro_protein', ylabel='Count'>
```



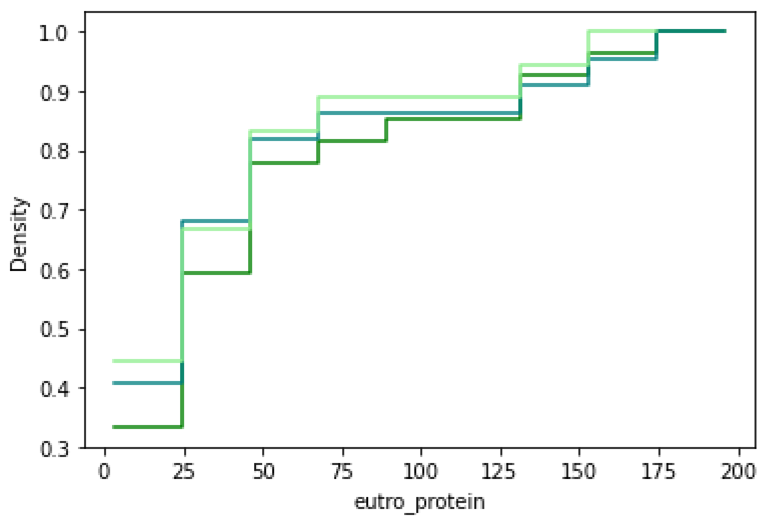
In [34]:
```
# pmf
sns.histplot(data=df2, x="eutro_protein", color = "green", stat = "probability"
sns.histplot(data=veg, x="eutro_protein", color = "teal", stat = "probability"
sns.histplot(data=vegan, x="eutro_protein", color = "lightgreen", stat = "proba
```

Out[34]:
```
<AxesSubplot:xlabel='eutro_protein', ylabel='Probability'>
```



In [35]:
```
# cdf
sns.histplot(data=df2, x="eutro_protein", element="step", fill=False, cumulati
sns.histplot(data=veg, x="eutro_protein", element="step", fill=False, cumulati
sns.histplot(data=vegan, x="eutro_protein", element="step", fill=False, cumula
```

Out[35]: <AxesSubplot:xlabel='eutro_protein', ylabel='Density'>

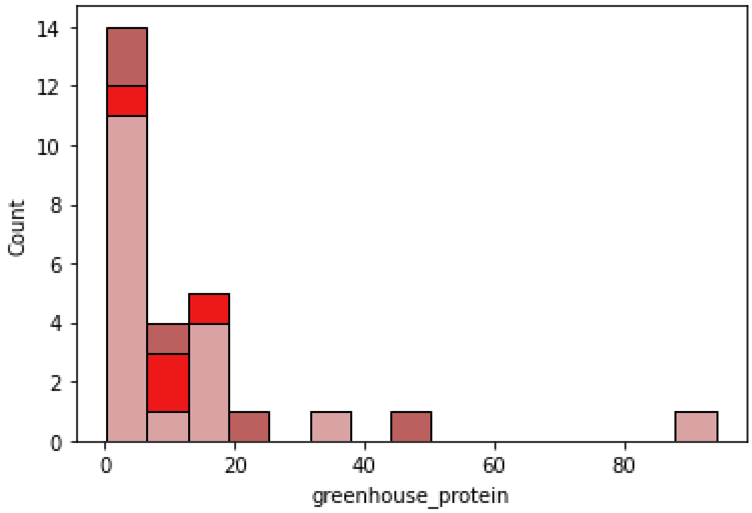


In general, vegan food products produce less eutrophying emissions than the other scenarios; however, the difference betweenbetween all three (and especially between vegan and vegetarian) is much less noticable than in other variables.

greenhouse_protein

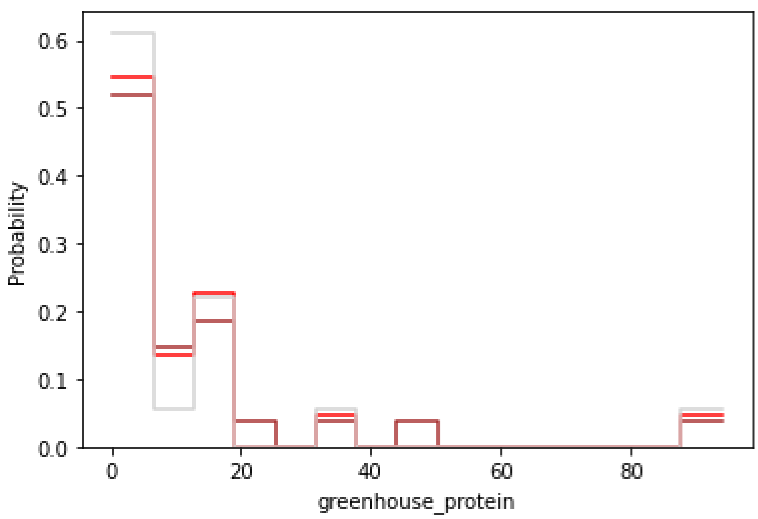
In [36]:
```
sns.histplot(data=df2, x="greenhouse_protein", color = "brown", binwidth = 6.2
sns.histplot(data=veg, x="greenhouse_protein", color = "red", binwidth = 6.25)
sns.histplot(data=vegan, x="greenhouse_protein", color = "lightgrey", binwidth
```

Out[36]: <AxesSubplot:xlabel='greenhouse_protein', ylabel='Count'>



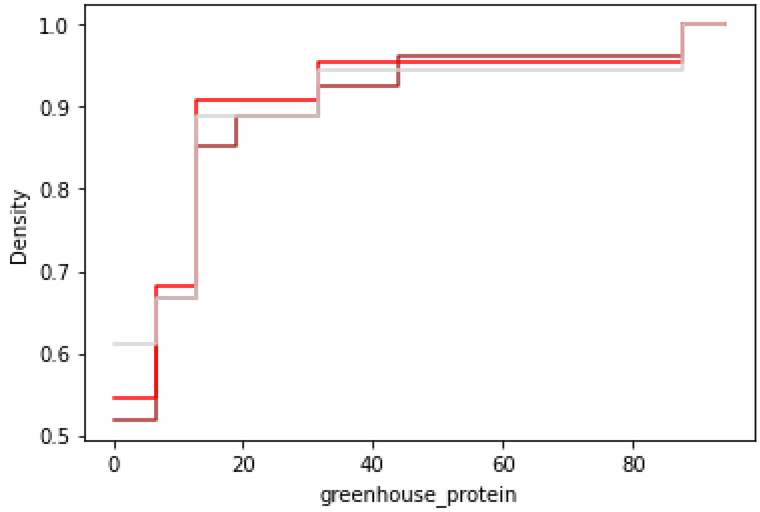
In [37]:
```
# pmf
sns.histplot(data=df2, x="greenhouse_protein", color = "brown", stat = "probab
sns.histplot(data=veg, x="greenhouse_protein", color = "red", stat = "probabil
sns.histplot(data=vegan, x="greenhouse_protein", color = "lightgrey", stat = "
```

Out[37]: <AxesSubplot:xlabel='greenhouse_protein', ylabel='Probability'>



In [38]:
```
# cdf
sns.histplot(data=df2, x="greenhouse_protein", element="step", fill=False, cumu
sns.histplot(data=veg, x="greenhouse_protein", element="step", fill=False, cumu
sns.histplot(data=vegan, x="greenhouse_protein", element="step", fill=False, cu
```

Out[38]: <AxesSubplot:xlabel='greenhouse_protein', ylabel='Density'>



For this variable, while the vegan food products start out with lower greenhouse gas emissions per 100g protein, the vegetarian food products are lower over the entire range, except for the very smallest emissions.

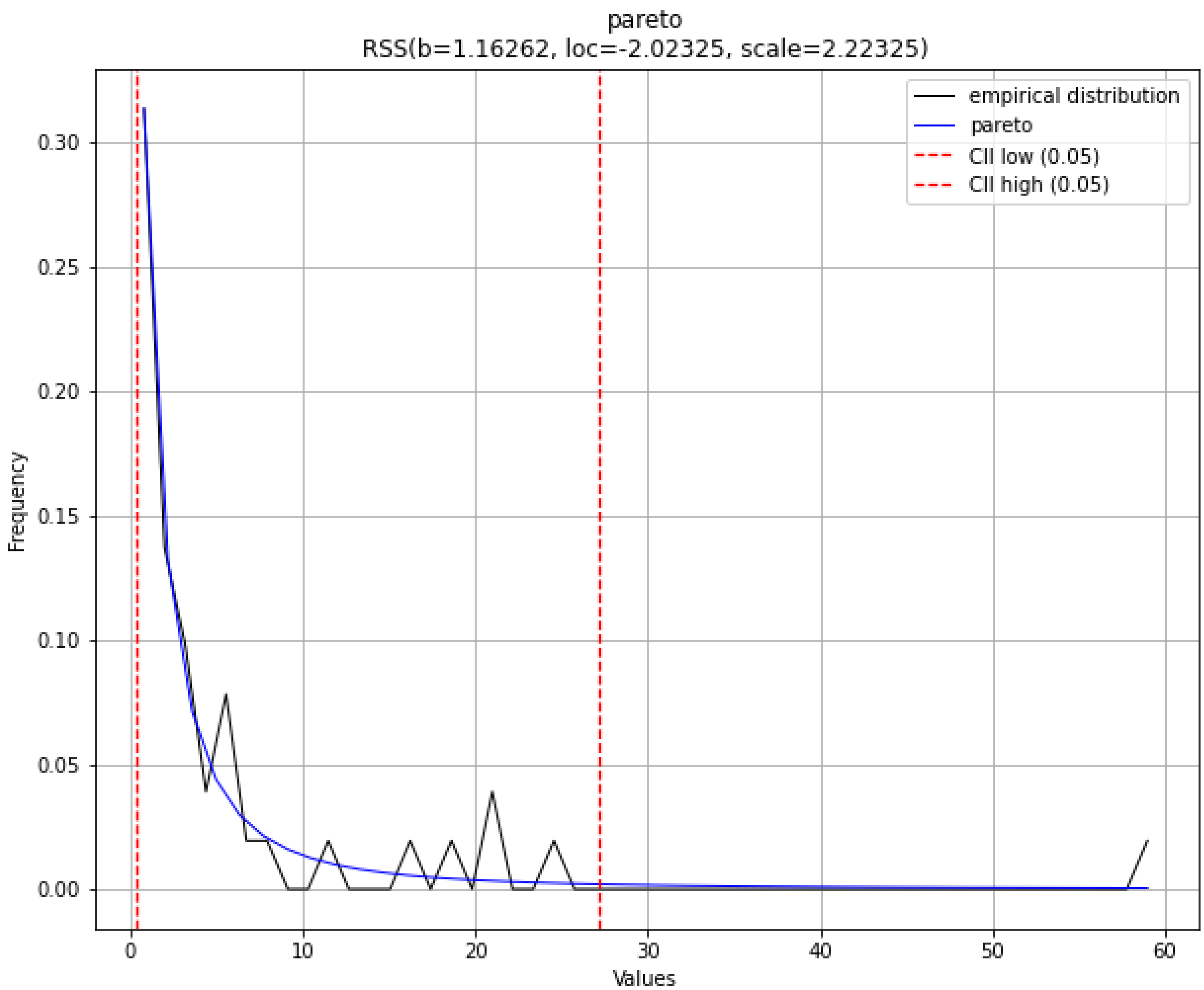
Plot of analytic distributions for chosen variables

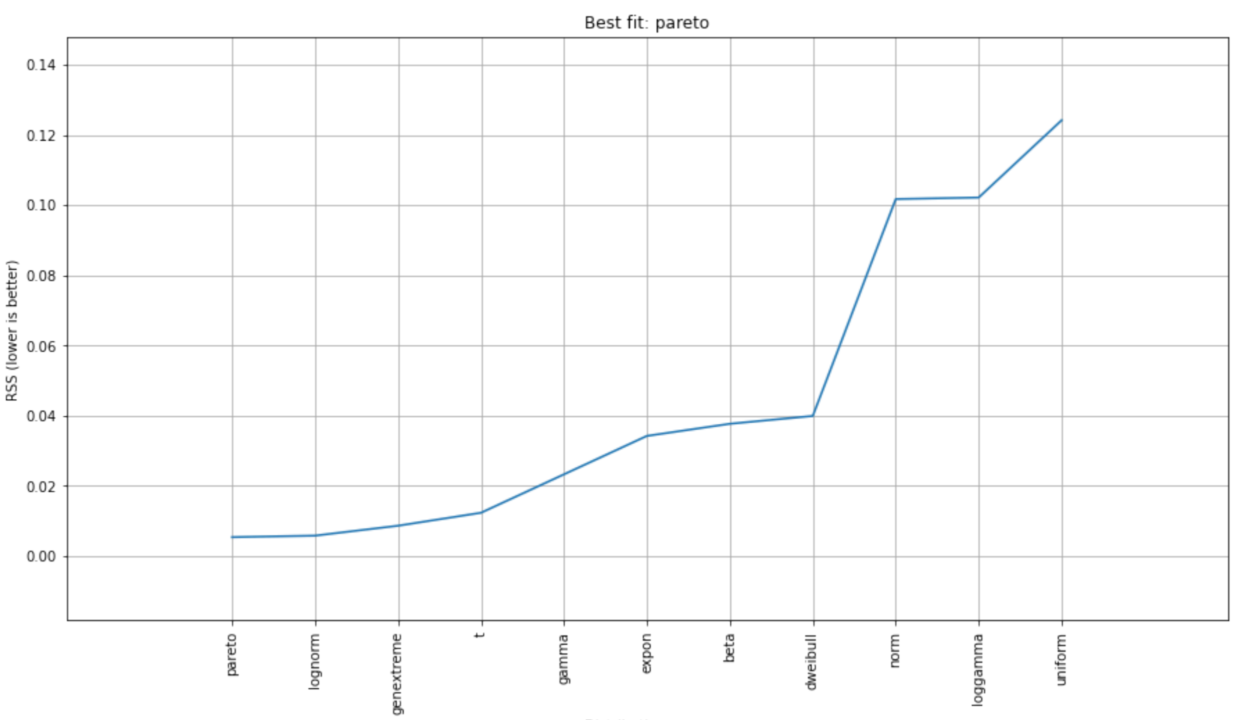
In [39]:
```
# function to plot analytic distribution
def define_analytic(variable):
    x = variable.dropna()

    dist = distfit()

    dist.fit_transform(x)

    dist.plot()

    dist.plot_summary()

    print(dist.summary)
```

total_em

In [40]:
```
define_analytic(df2.total_em)
```

```
[distfit] >fit..
[distfit] >transform..
[distfit] >[norm      ] [0.00 sec] [RSS: 0.101711] [loc=5.972 scale=10.379]
[distfit] >[expon     ] [0.00 sec] [RSS: 0.0342422] [loc=0.200 scale=5.772]
[distfit] >[pareto    ] [0.08 sec] [RSS: 0.00540812] [loc=-2.023 scale=2.223]
[distfit] >[dweibull  ] [0.04 sec] [RSS: 0.0399319] [loc=1.100 scale=5.452]
[distfit] >[t         ] [0.03 sec] [RSS: 0.0123526] [loc=1.158 scale=0.947]
[distfit] >[genextreme] [0.07 sec] [RSS: 0.008662] [loc=1.110 scale=1.369]
[distfit] >[gamma     ] [0.08 sec] [RSS: 0.0232953] [loc=0.200 scale=12.639]
[distfit] >[lognorm   ] [0.13 sec] [RSS: 0.00584022] [loc=0.180 scale=1.719]
[distfit] >[beta      ] [0.13 sec] [RSS: 0.0376993] [loc=0.200 scale=95.288]
[distfit] >[uniform   ] [0.00 sec] [RSS: 0.124166] [loc=0.200 scale=59.400]
[distfit] >[loggamma  ] [0.08 sec] [RSS: 0.102152] [loc=-3784.087 scale=492.98
7]
[distfit] >Compute confidence interval [parametric]
[distfit] >plot..
[distfit] >plot summary..
         distr     score  LLE         loc       scale  \
0       pareto  0.005408  NaN   -2.023254    2.223254   
1      lognorm   0.00584  NaN    0.179722    1.718868   
2   genextreme  0.008662  NaN    1.110065    1.369342   
3            t  0.012353  NaN    1.158025    0.947051   
4        gamma  0.023295  NaN         0.2   12.639399   
5        expon  0.034242  NaN         0.2    5.772093   
6         beta  0.037699  NaN         0.2   95.287846   
7     dweibull  0.039932  NaN         1.1    5.451763   
8         norm  0.101711  NaN    5.972093   10.378922   
9     loggamma  0.102152  NaN -3784.08745  492.987498   
10     uniform  0.124166  NaN         0.2        59.4   

                                       arg  
0                     (1.1626182631995419,)  
1                     (1.7244549781410417,)  
2                    (-1.2577950034553629,)  
3                     (0.6891032604746714,)  
4                     (0.5340272357887086,)  
5                                        ()  
6   (0.22194965537755468, 2.7405919322437646)  
7                     (0.6109681984353121,)  
8                                        ()  
9                      (2182.236110880629,)  
10                                       ()  
```

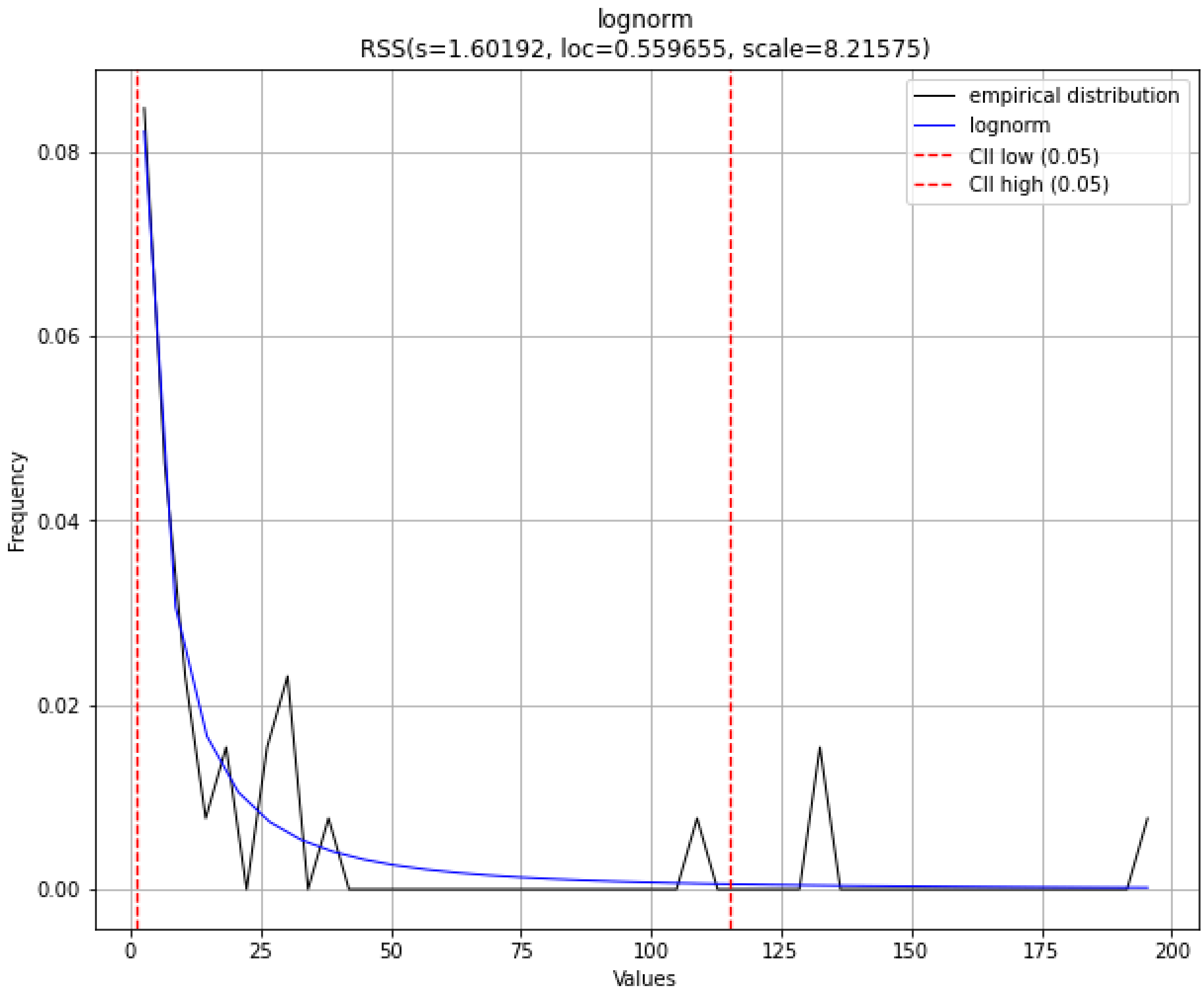

pareto
RSS(b=1.16262, loc=-2.02325, scale=2.22325)
empirical distribution
pareto
CII low (0.05)
CII high (0.05)
Frequency
Values

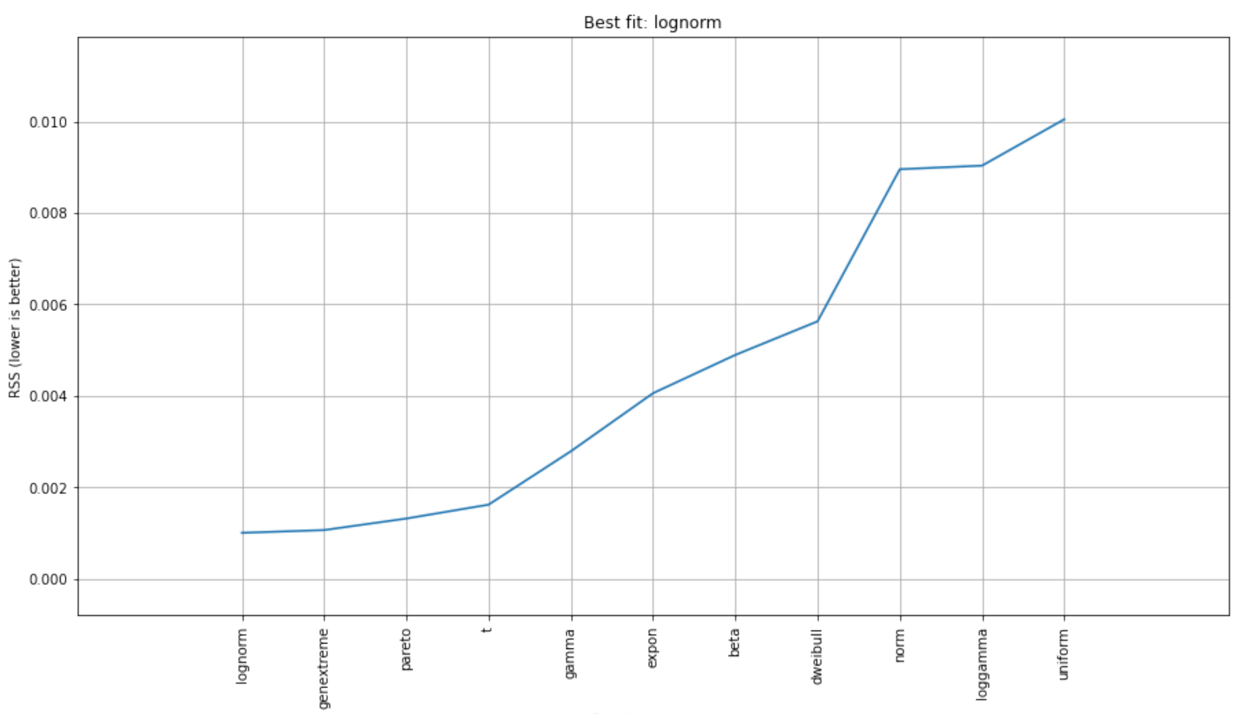

Best fit: pareto
RSS (lower is better)
pareto
lognorm
genextreme
t
gamma
expon
beta
dweibull
norm
loggamma
uniform
Distribution name

eutro_kcal

In [41]:
```
define_analytic(df2.eutro_kcal)
```

```
[distfit] >fit..
[distfit] >transform..
[distfit] >[norm      ] [0.00 sec] [RSS: 0.00895471] [loc=27.182 scale=45.737]
[distfit] >[expon     ] [0.00 sec] [RSS: 0.00406029] [loc=0.708 scale=26.473]
[distfit] >[pareto    ] [0.04 sec] [RSS: 0.0013207] [loc=-1.073 scale=1.781]
[distfit] >[dweibull  ] [0.04 sec] [RSS: 0.0056299] [loc=4.821 scale=38.150]
[distfit] >[t         ] [0.03 sec] [RSS: 0.00162381] [loc=4.444 scale=2.736]
[distfit] >[genextreme] [0.12 sec] [RSS: 0.00106831] [loc=5.104 scale=6.047]
[distfit] >[gamma     ] [0.06 sec] [RSS: 0.00279027] [loc=0.708 scale=88.478]
[distfit] >[lognorm   ] [0.05 sec] [RSS: 0.00100805] [loc=0.560 scale=8.216]
[distfit] >[beta      ] [0.11 sec] [RSS: 0.00489608] [loc=0.708 scale=271.206]
[distfit] >[uniform   ] [0.00 sec] [RSS: 0.0100457] [loc=0.708 scale=196.649]
[distfit] >[loggamma  ] [0.06 sec] [RSS: 0.00903339] [loc=-15689.019 scale=208
2.818]
[distfit] >Compute confidence interval [parametric]
[distfit] >plot..
[distfit] >plot summary..
         distr     score  LLE          loc        scale  \
0      lognorm  0.001008  NaN     0.559655     8.215753
1   genextreme  0.001068  NaN     5.103789     6.046825
2       pareto  0.001321  NaN    -1.072804     1.781223
3            t  0.001624  NaN     4.444247     2.736117
4        gamma   0.00279  NaN     0.708419    88.47821
5        expon   0.00406  NaN     0.708419    26.473128
6         beta  0.004896  NaN     0.708419   271.206342
7     dweibull   0.00563  NaN     4.820513    38.149939
8         norm  0.008955  NaN    27.181547    45.736818
9     loggamma  0.009033  NaN -15689.01931  2082.817592
10     uniform  0.010046  NaN     0.708419   196.648724

                                          arg
0                       (1.6019184188769404,)
1                      (-1.1344711588039438,)
2                        (0.513129990300909,)
3                       (0.5709255233547469,)
4                      (0.40464555885539333,)
5                                          ()
6    (0.16136308655430387, 0.6989472526964302)
7                       (0.5406745797134349,)
8                                          ()
9                       (1893.1325156261078,)
10                                         ()
```

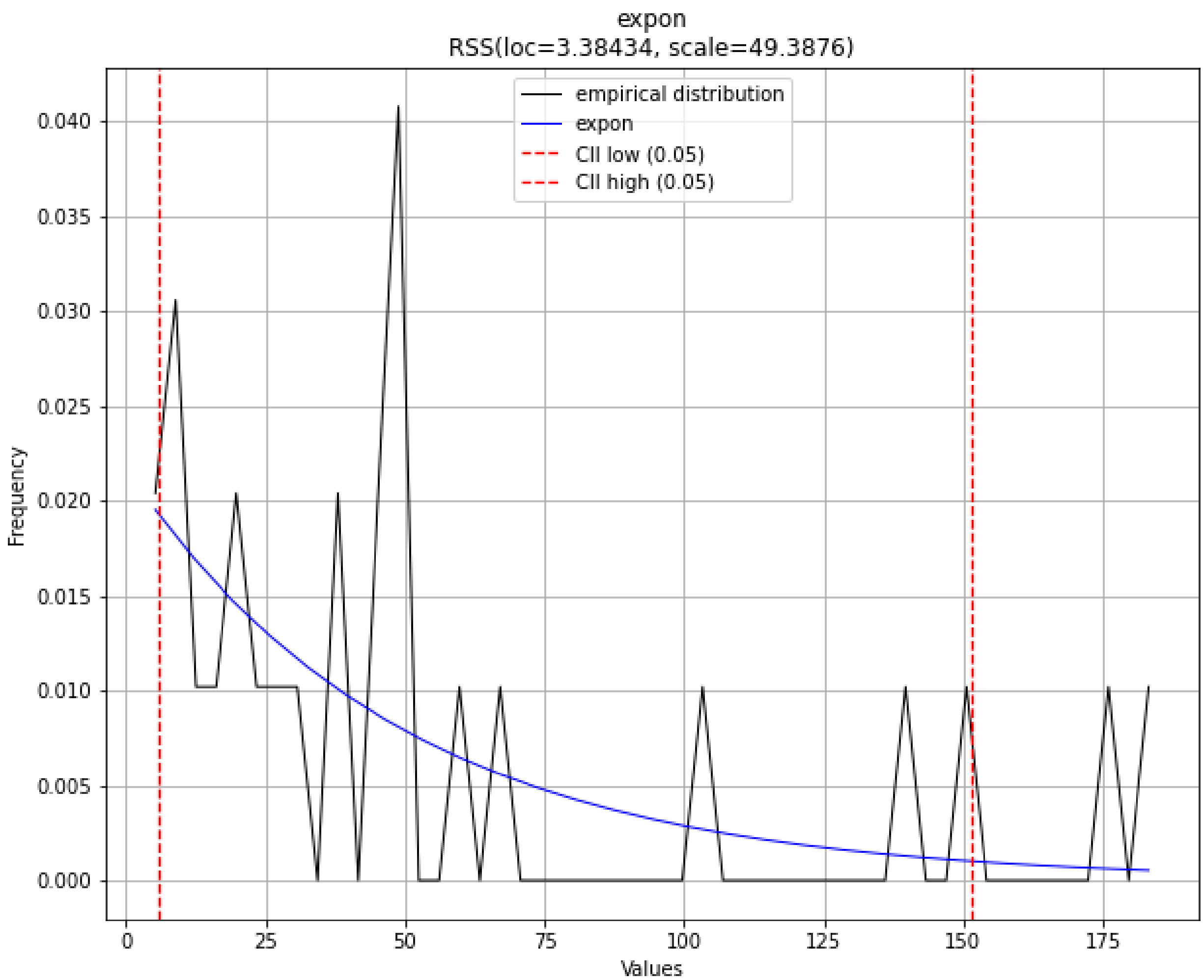


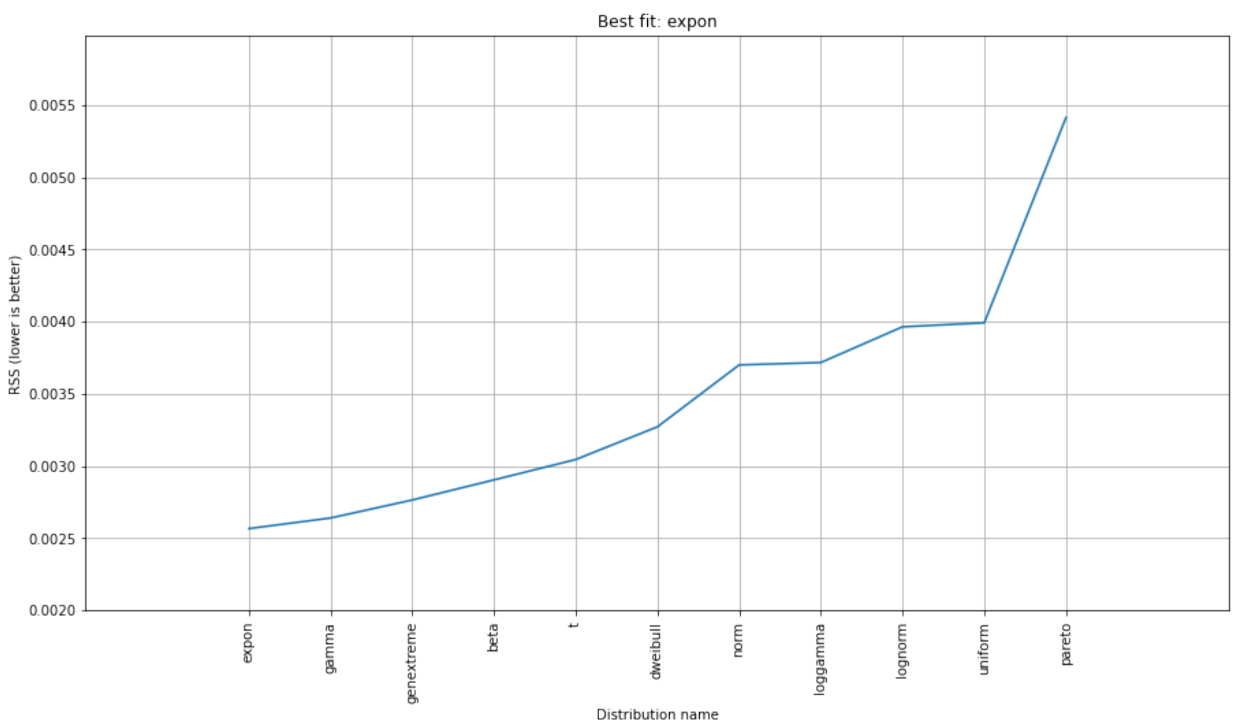


eutro_protein

In [42]:
```
define_analytic(df2.eutro_protein)
```

```
[distfit] >fit..
[distfit] >transform..
[distfit] >[norm      ] [0.00 sec] [RSS: 0.00370074] [loc=52.772 scale=51.061]
[distfit] >[expon     ] [0.00 sec] [RSS: 0.00256606] [loc=3.384 scale=49.388]
[distfit] >[pareto    ] [0.04 sec] [RSS: 0.00541524] [loc=-0.006 scale=3.390]
[distfit] >[dweibull  ] [0.08 sec] [RSS: 0.00327171] [loc=44.552 scale=32.903]
[distfit] >[t         ] [0.05 sec] [RSS: 0.00304485] [loc=33.769 scale=22.573]
[distfit] >[genextreme] [0.16 sec] [RSS: 0.00276446] [loc=24.010 scale=22.476]
[distfit] >[gamma     ] [0.05 sec] [RSS: 0.00263964] [loc=3.384 scale=52.153]
[distfit] >[lognorm   ] [0.15 sec] [RSS: 0.00396489] [loc=3.384 scale=9.235]
[distfit] >[beta      ] [0.12 sec] [RSS: 0.00290372] [loc=3.384 scale=214.044]
[distfit] >[uniform   ] [0.00 sec] [RSS: 0.00399228] [loc=3.384 scale=181.666]
[distfit] >[loggamma  ] [0.07 sec] [RSS: 0.00371723] [loc=-18285.761 scale=240
1.742]
[distfit] >Compute confidence interval [parametric]
[distfit] >plot..
[distfit] >plot summary..
         distr     score  LLE           loc        scale  \
0        expon  0.002566  NaN      3.384338    49.387615
1        gamma   0.00264  NaN      3.384338    52.152513
2   genextreme  0.002764  NaN     24.009789    22.476372
3         beta  0.002904  NaN      3.384338   214.043629
4            t  0.003045  NaN     33.769351    22.572723
5     dweibull  0.003272  NaN      44.55163    32.903087
6         norm  0.003701  NaN     52.771953    51.061142
7     loggamma  0.003717  NaN -18285.760762  2401.741912
8      lognorm  0.003965  NaN      3.384338     9.234843
9      uniform  0.003992  NaN      3.384338    181.66632
10      pareto  0.005415  NaN     -0.005966     3.390304

                                        arg
0                                        ()
1                      (0.8049549512420582,)
2                     (-0.5381239250625844,)
3   (0.45177329932533916, 1.428828686501218)
4                      (1.6052552014531765,)
5                      (0.8428678683020703,)
6                                        ()
7                      (2070.7673937535837,)
8                      (6.8373419973270355,)
9                                        ()
10                     (0.4418323708389803,)
```

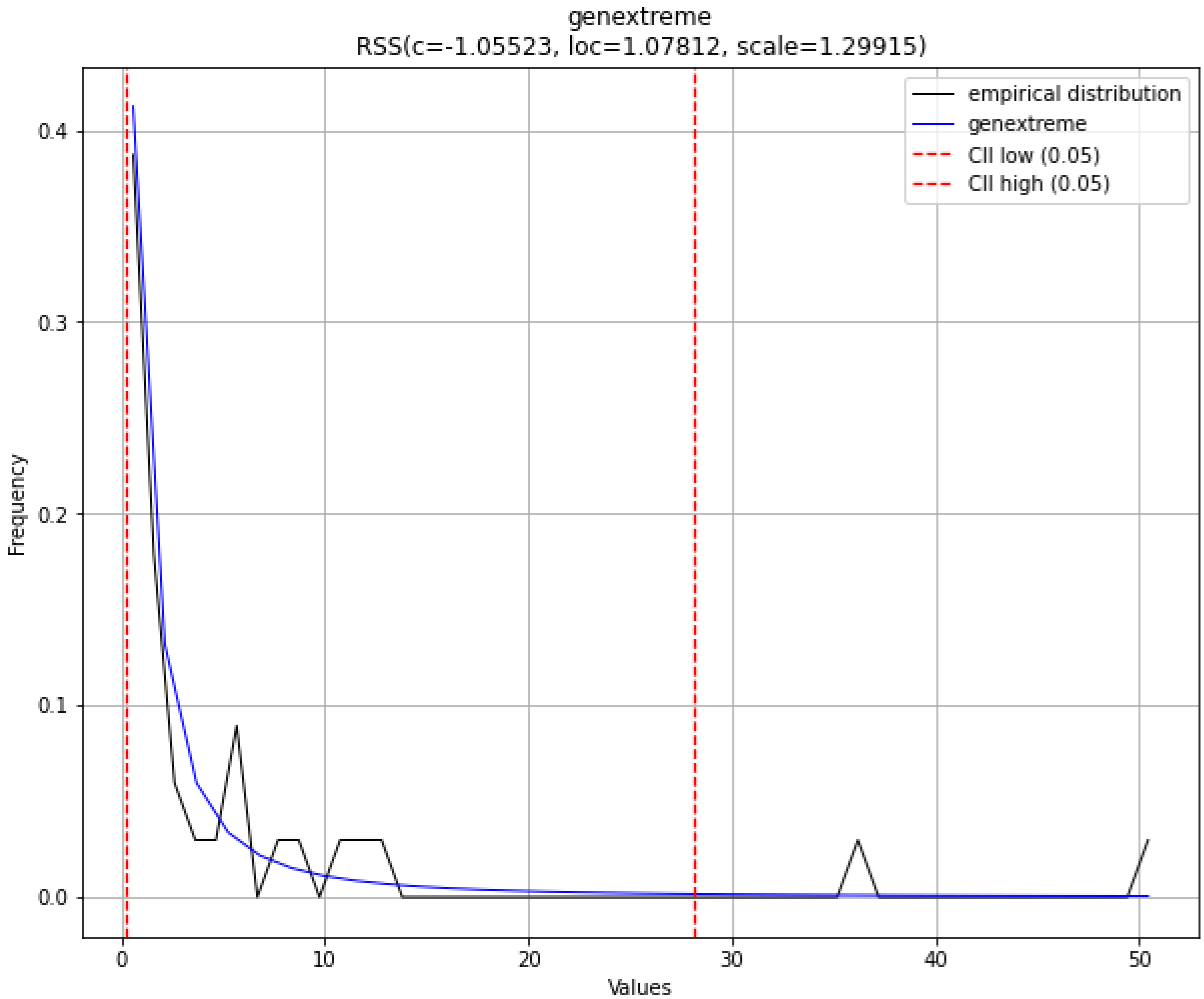


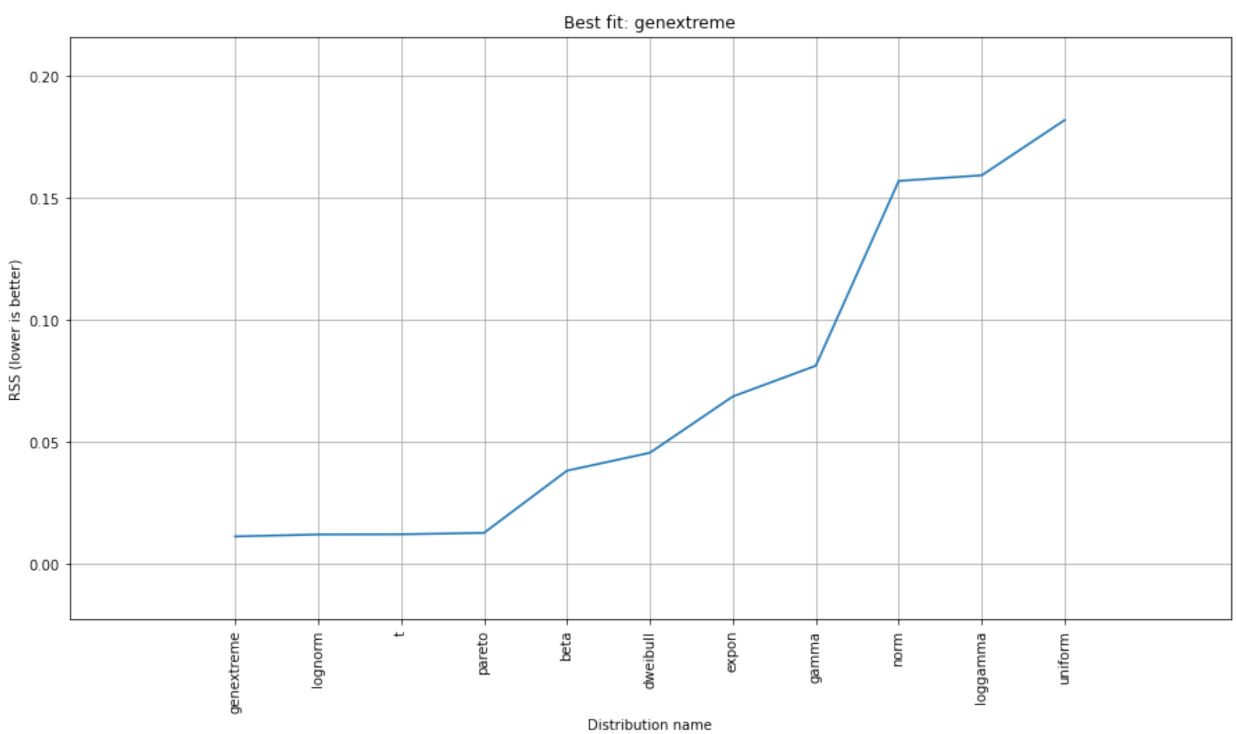


greenhouse_kcal

In [43]:
```
define_analytic(df2.greenhouse_kcal)
```

```
[distfit] >fit..
[distfit] >transform..
[distfit] >[norm      ] [0.00 sec] [RSS: 0.157073] [loc=5.634 scale=10.452]
[distfit] >[expon     ] [0.00 sec] [RSS: 0.0687257] [loc=0.070 scale=5.564]
[distfit] >[pareto    ] [0.11 sec] [RSS: 0.0128174] [loc=-2.243 scale=2.313]
[distfit] >[dweibull  ] [0.04 sec] [RSS: 0.0456502] [loc=0.912 scale=2.035]
[distfit] >[t         ] [0.03 sec] [RSS: 0.0122462] [loc=0.882 scale=0.559]
[distfit] >[genextreme] [0.09 sec] [RSS: 0.011367] [loc=1.078 scale=1.299]
[distfit] >[gamma     ] [0.12 sec] [RSS: 0.0813203] [loc=0.070 scale=20.741]
[distfit] >[lognorm   ] [0.04 sec] [RSS: 0.0121656] [loc=0.028 scale=1.858]
[distfit] >[beta      ] [0.11 sec] [RSS: 0.0383005] [loc=0.070 scale=339.008]
[distfit] >[uniform   ] [0.00 sec] [RSS: 0.182011] [loc=0.070 scale=50.877]
[distfit] >[loggamma  ] [0.11 sec] [RSS: 0.159359] [loc=-4873.427 scale=612.19
7]
[distfit] >Compute confidence interval [parametric]
[distfit] >plot..
[distfit] >plot summary..
         distr     score  LLE          loc       scale  \
0   genextreme  0.011367  NaN     1.078118    1.299153
1      lognorm  0.012166  NaN     0.028067    1.857596
2            t  0.012246  NaN     0.881713    0.559305
3       pareto  0.012817  NaN    -2.243226    2.313145
4         beta    0.0383  NaN     0.069919  339.008481
5     dweibull   0.04565  NaN     0.911681    2.034759
6        expon  0.068726  NaN     0.069919    5.564024
7        gamma   0.08132  NaN     0.069919   20.741011
8         norm  0.157073  NaN     5.633943   10.451526
9     loggamma  0.159359  NaN -4873.426693  612.197207
10     uniform  0.182011  NaN     0.069919    50.87651

                                      arg
0                    (-1.055225612451192,)
1                     (1.498692756645393,)
2                    (0.5718263322076055,)
3                    (1.2327590543209612,)
4   (0.5171302785116825, 32.26869704652188)
5                     (0.520952995636887,)
6                                       ()
7                    (0.6072421416315884,)
8                                       ()
9                    (2892.0978862557868,)
10                                      ()
```

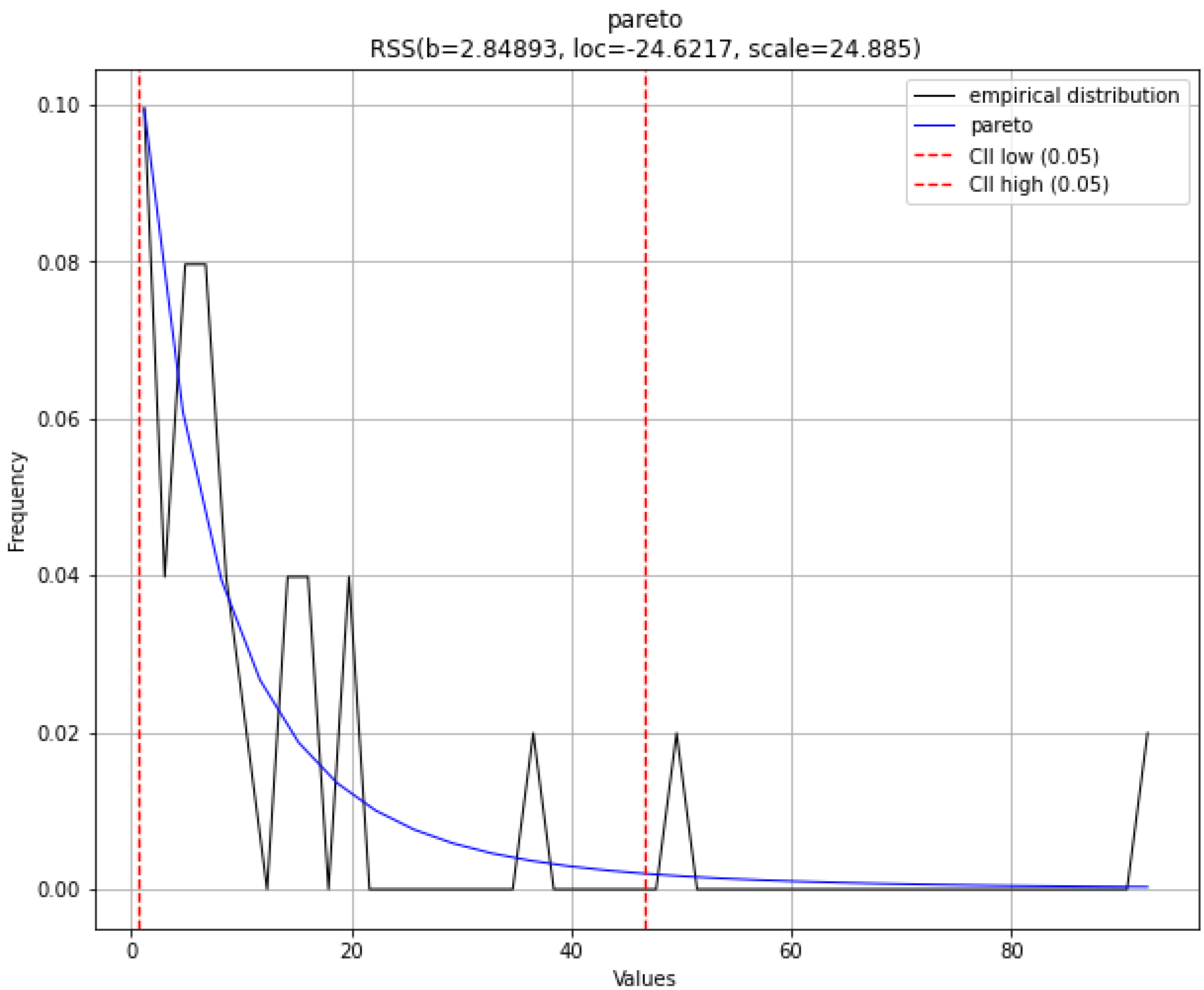


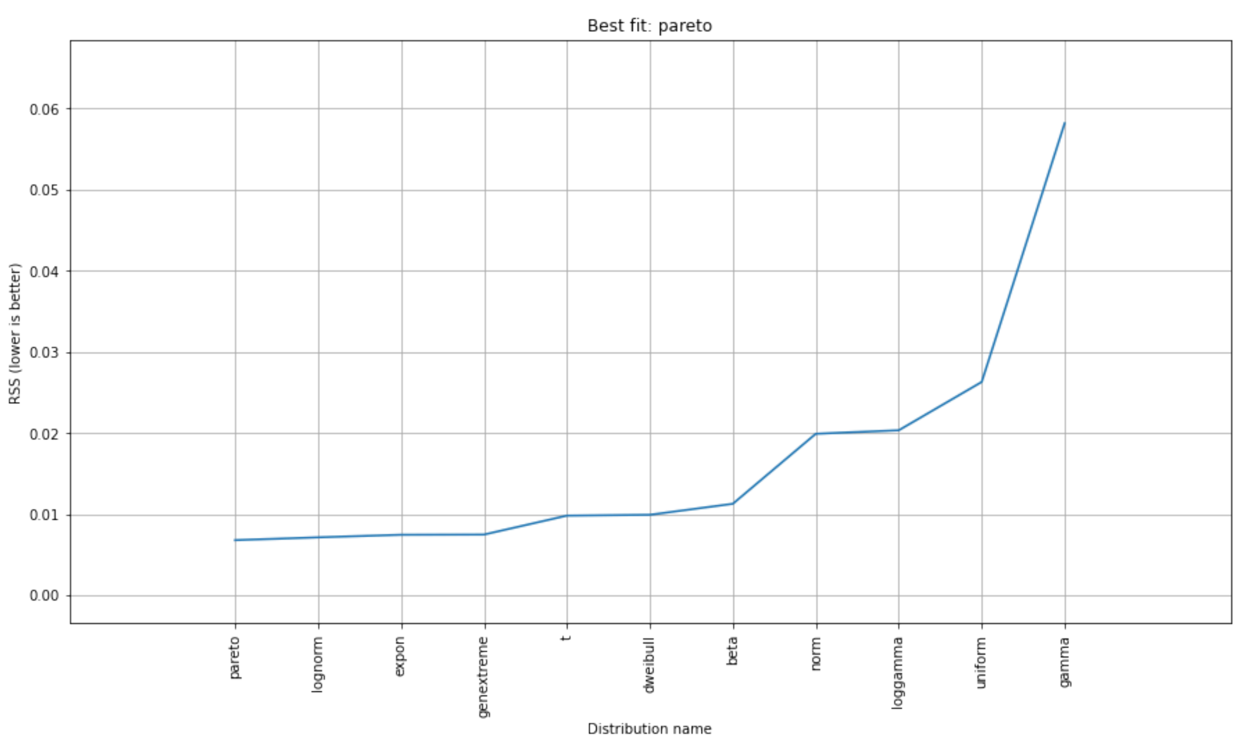


greenhouse_protein

In [44]:
```
define_analytic(df2.greenhouse_protein)
```

```
[distfit] >fit..
[distfit] >transform..
[distfit] >[norm      ] [0.00 sec] [RSS: 0.0199306] [loc=13.525 scale=19.064]
[distfit] >[expon     ] [0.00 sec] [RSS: 0.00748355] [loc=0.263 scale=13.262]
[distfit] >[pareto    ] [0.11 sec] [RSS: 0.00682179] [loc=-24.622 scale=24.88
5]
[distfit] >[dweibull  ] [0.04 sec] [RSS: 0.00994786] [loc=5.699 scale=8.422]
[distfit] >[t         ] [0.03 sec] [RSS: 0.00984277] [loc=6.363 scale=4.853]
[distfit] >[genextreme] [0.10 sec] [RSS: 0.00750938] [loc=4.753 scale=5.176]
[distfit] >[gamma     ] [0.06 sec] [RSS: 0.0581976] [loc=0.263 scale=1.780]
[distfit] >[lognorm   ] [0.03 sec] [RSS: 0.00716703] [loc=-0.271 scale=7.081]
[distfit] >[beta      ] [0.14 sec] [RSS: 0.0113012] [loc=0.263 scale=10116.20
3]
[distfit] >[uniform   ] [0.00 sec] [RSS: 0.026315] [loc=0.263 scale=93.037]
[distfit] >[loggamma  ] [0.09 sec] [RSS: 0.0203604] [loc=-5754.674 scale=787.3
32]
[distfit] >Compute confidence interval [parametric]
[distfit] >plot..
[distfit] >plot summary..
         distr     score  LLE          loc         scale  \
0       pareto  0.006822  NaN    -24.62165     24.884969
1      lognorm  0.007167  NaN    -0.270907      7.080898
2        expon  0.007484  NaN     0.263319     13.261587
3   genextreme  0.007509  NaN     4.752663      5.175541
4            t  0.009843  NaN      6.36255      4.852861
5     dweibull  0.009948  NaN     5.698614       8.42169
6         beta  0.011301  NaN     0.263319  10116.203487
7         norm  0.019931  NaN    13.524906     19.064299
8     loggamma   0.02036  NaN -5754.674168    787.332111
9      uniform  0.026315  NaN     0.263319     93.036681
10       gamma  0.058198  NaN     0.263319      1.780328

                                      arg
0                     (2.8489271336091786,)
1                     (1.2011760765766137,)
2                                        ()
3                    (-0.6498433361793019,)
4                     (1.3070600453775798,)
5                     (0.6968615541390022,)
6   (0.44832657632873363, 742.9516993803135)
7                                        ()
8                     (1520.2291280904146,)
9                                        ()
10                    (0.6471504935522094,)
```

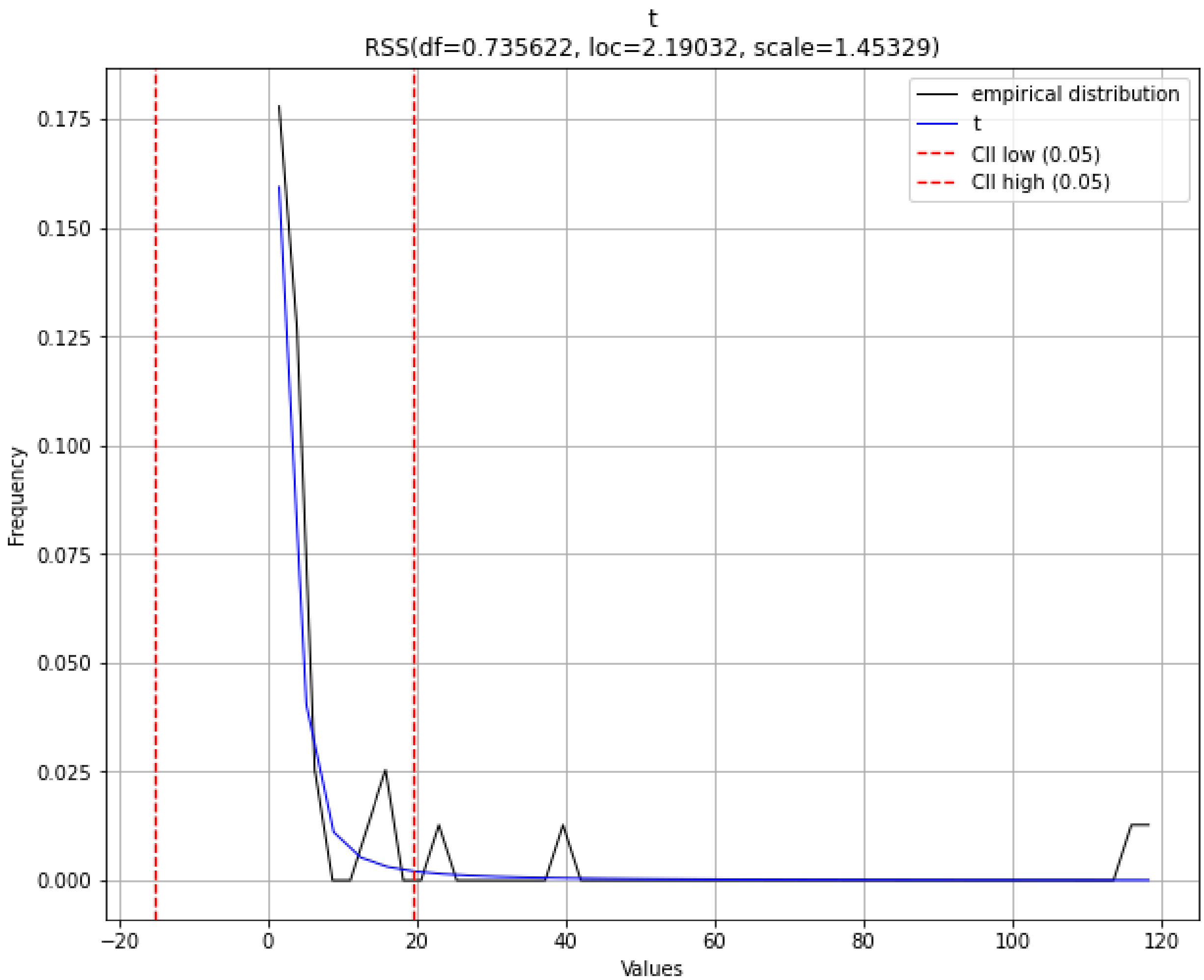

pareto
RSS(b=2.84893, loc=-24.6217, scale=24.885)
empirical distribution
pareto
CII low (0.05)
CII high (0.05)
Frequency
Values
0.10
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80

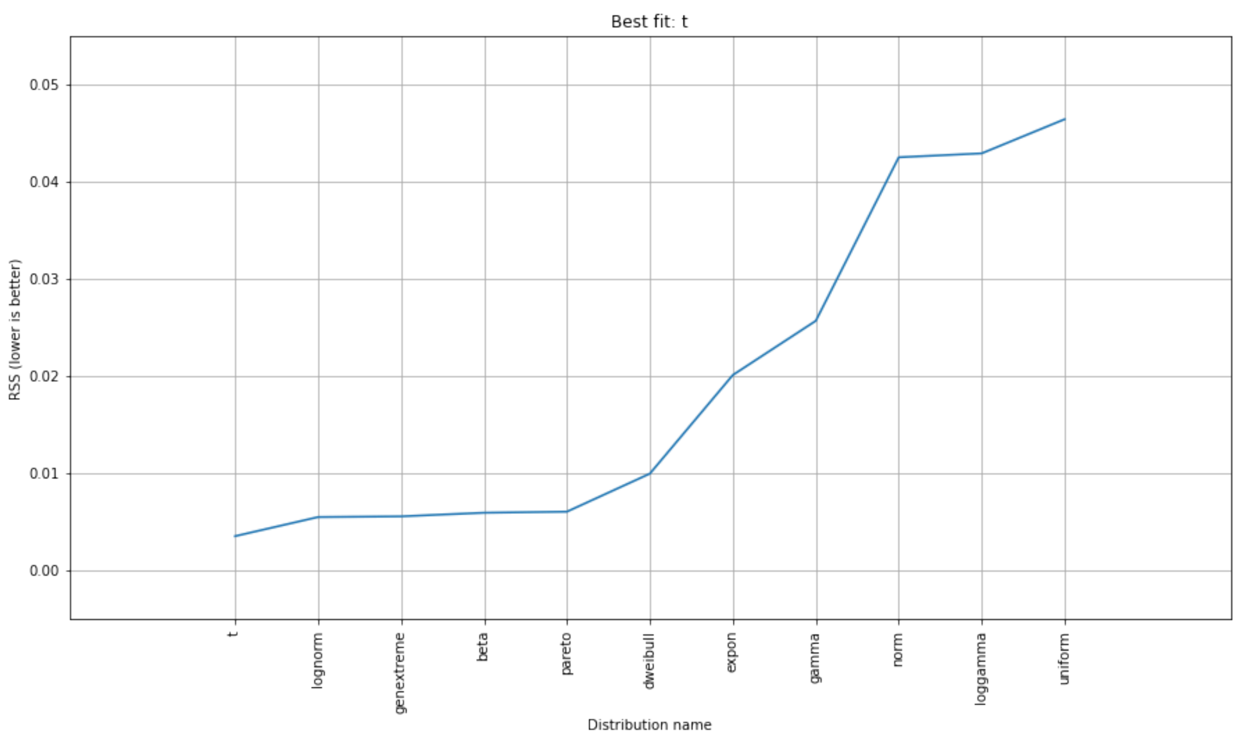

Best fit: pareto
RSS (lower is better)
Distribution name
pareto
lognorm
expon
genextreme
t
dweibull
beta
norm
loggamma
uniform
gamma
0.06
0.05
0.04
0.03
0.02
0.01
0.00

land_kcal

In [45]:
```
define_analytic(df2.land_kcal)
```

```
[distfit] >fit..
[distfit] >transform..
[distfit] >[norm      ] [0.00 sec] [RSS: 0.0424768] [loc=12.423 scale=27.916]
[distfit] >[expon     ] [0.00 sec] [RSS: 0.0200739] [loc=0.274 scale=12.149]
[distfit] >[pareto    ] [0.04 sec] [RSS: 0.00601308] [loc=-1.311 scale=1.585]
[distfit] >[dweibull  ] [0.05 sec] [RSS: 0.00992638] [loc=1.202 scale=4.939]
[distfit] >[t         ] [0.03 sec] [RSS: 0.0034938] [loc=2.190 scale=1.453]
[distfit] >[genextreme] [0.08 sec] [RSS: 0.00553049] [loc=1.947 scale=2.212]
[distfit] >[gamma     ] [0.05 sec] [RSS: 0.025649] [loc=0.274 scale=55.698]
[distfit] >[lognorm   ] [0.04 sec] [RSS: 0.00545835] [loc=0.220 scale=3.047]
[distfit] >[beta      ] [0.11 sec] [RSS: 0.00590217] [loc=0.274 scale=1128.55
2]
[distfit] >[uniform   ] [0.00 sec] [RSS: 0.0463927] [loc=0.274 scale=119.217]
[distfit] >[loggamma  ] [0.07 sec] [RSS: 0.0428798] [loc=-9754.875 scale=1297.
062]
[distfit] >Compute confidence interval [parametric]
[distfit] >plot..
[distfit] >plot summary..
         distr     score  LLE          loc        scale  \
0            t  0.003494  NaN     2.190316     1.453292
1      lognorm  0.005458  NaN     0.219665     3.047392
2   genextreme   0.00553  NaN     1.946698     2.211642
3         beta  0.005902  NaN     0.273756  1128.551837
4       pareto  0.006013  NaN    -1.311073     1.584829
5     dweibull  0.009926  NaN     1.202186      4.93898
6        expon  0.020074  NaN     0.273756     12.14941
7        gamma  0.025649  NaN     0.273756    55.697557
8         norm  0.042477  NaN    12.423165    27.915864
9     loggamma   0.04288  NaN -9754.875041  1297.061863
10     uniform  0.046393  NaN     0.273756   119.217087

                                        arg
0                      (0.7356217024032716,)
1                      (1.6023240567490342,)
2                     (-1.0716451931813178,)
3   (0.4639757623160847, 159.20849181689493)
4                      (0.7605647875832052,)
5                      (0.5204844192351071,)
6                                         ()
7                      (0.5842298750225536,)
8                                         ()
9                      (1864.0303369236226,)
10                                        ()
```

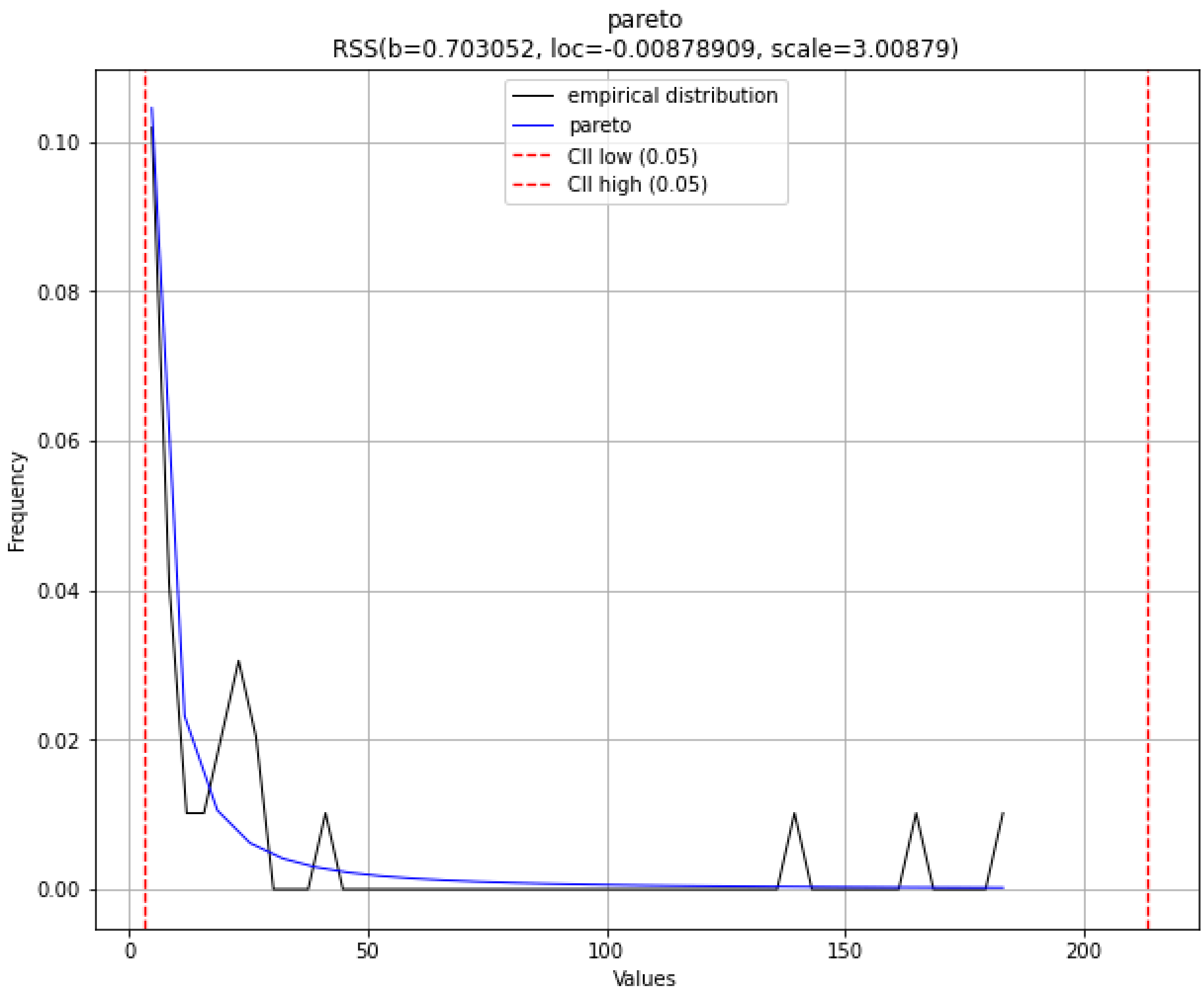


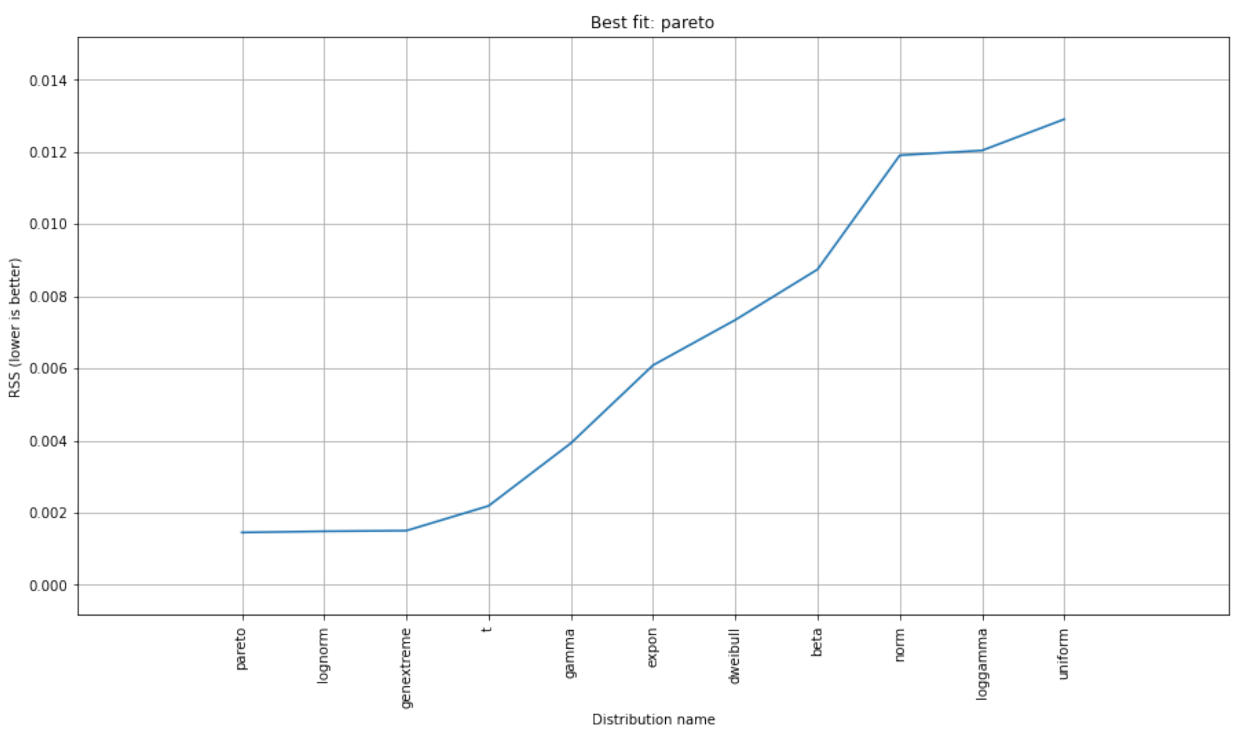


land_protein

In [46]:
```
define_analytic(df2.land_protein)
```

```
[distfit] >fit..
[distfit] >transform..
[distfit] >[norm      ] [0.00 sec] [RSS: 0.0119089] [loc=29.105 scale=48.386]
[distfit] >[expon     ] [0.00 sec] [RSS: 0.00608891] [loc=3.000 scale=26.105]
[distfit] >[pareto    ] [0.04 sec] [RSS: 0.00145098] [loc=-0.009 scale=3.009]
[distfit] >[dweibull  ] [0.08 sec] [RSS: 0.00734386] [loc=5.651 scale=52.710]
[distfit] >[t         ] [0.04 sec] [RSS: 0.0021879] [loc=5.793 scale=3.109]
[distfit] >[genextreme] [0.11 sec] [RSS: 0.00150107] [loc=5.810 scale=4.724]
[distfit] >[gamma     ] [0.08 sec] [RSS: 0.0039246] [loc=3.000 scale=91.296]
[distfit] >[lognorm   ] [0.06 sec] [RSS: 0.00148405] [loc=2.955 scale=5.859]
[distfit] >[beta      ] [0.07 sec] [RSS: 0.00874442] [loc=2.999 scale=181.813]
[distfit] >[uniform   ] [0.00 sec] [RSS: 0.0129079] [loc=3.000 scale=181.813]
[distfit] >[loggamma  ] [0.07 sec] [RSS: 0.0120416] [loc=-15945.016 scale=214
7.887]
[distfit] >Compute confidence interval [parametric]
[distfit] >plot..
[distfit] >plot summary..
         distr     score  LLE           loc        scale  \
0       pareto  0.001451  NaN     -0.008789     3.008789   
1      lognorm  0.001484  NaN      2.955481     5.859158   
2   genextreme  0.001501  NaN      5.809728     4.723881   
3            t  0.002188  NaN      5.793367     3.108967   
4        gamma  0.003925  NaN           3.0    91.296149   
5        expon  0.006089  NaN           3.0    26.105042   
6     dweibull  0.007344  NaN      5.650685     52.71022   
7         beta  0.008744  NaN      2.999217   181.813377   
8         norm  0.011909  NaN     29.105042    48.385625   
9     loggamma  0.012042  NaN -15945.016338  2147.886542   
10     uniform  0.012908  NaN           3.0   181.812594   

                                        arg  
0                       (0.7030515517955596,)  
1                       (1.9541518278210468,)  
2                      (-1.4866813253845765,)  
3                       (0.6061932077410987,)  
4                        (0.360610707308908,)  
5                                          ()  
6                       (0.5601162478968913,)  
7    (0.07202058515196277, 0.35209509631341707)  
8                                          ()  
9                       (1698.4722961214648,)  
10                                         ()  
```

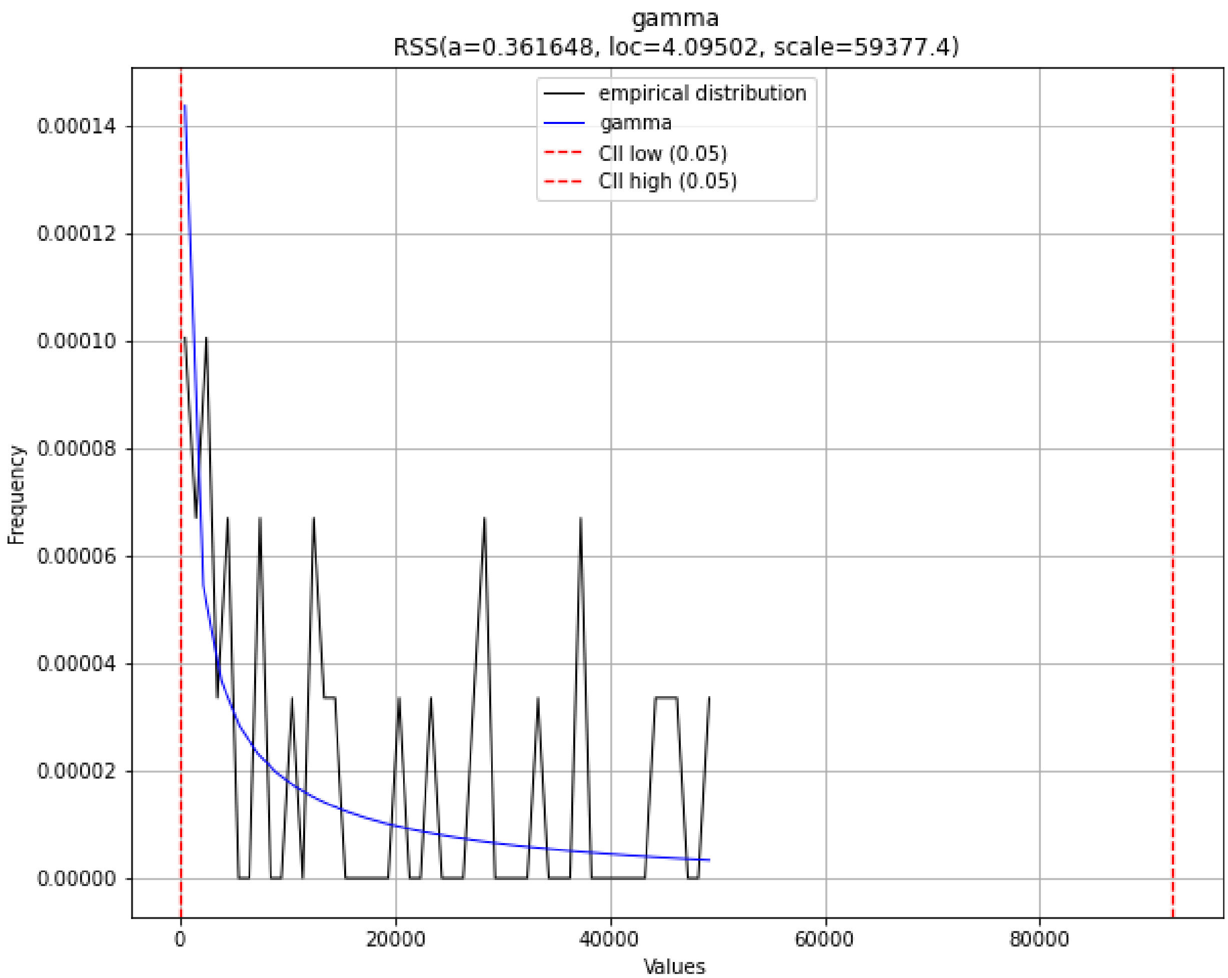

pareto
RSS(b=0.703052, loc=-0.00878909, scale=3.00879)
empirical distribution
pareto
CII low (0.05)
CII high (0.05)
Frequency
Values

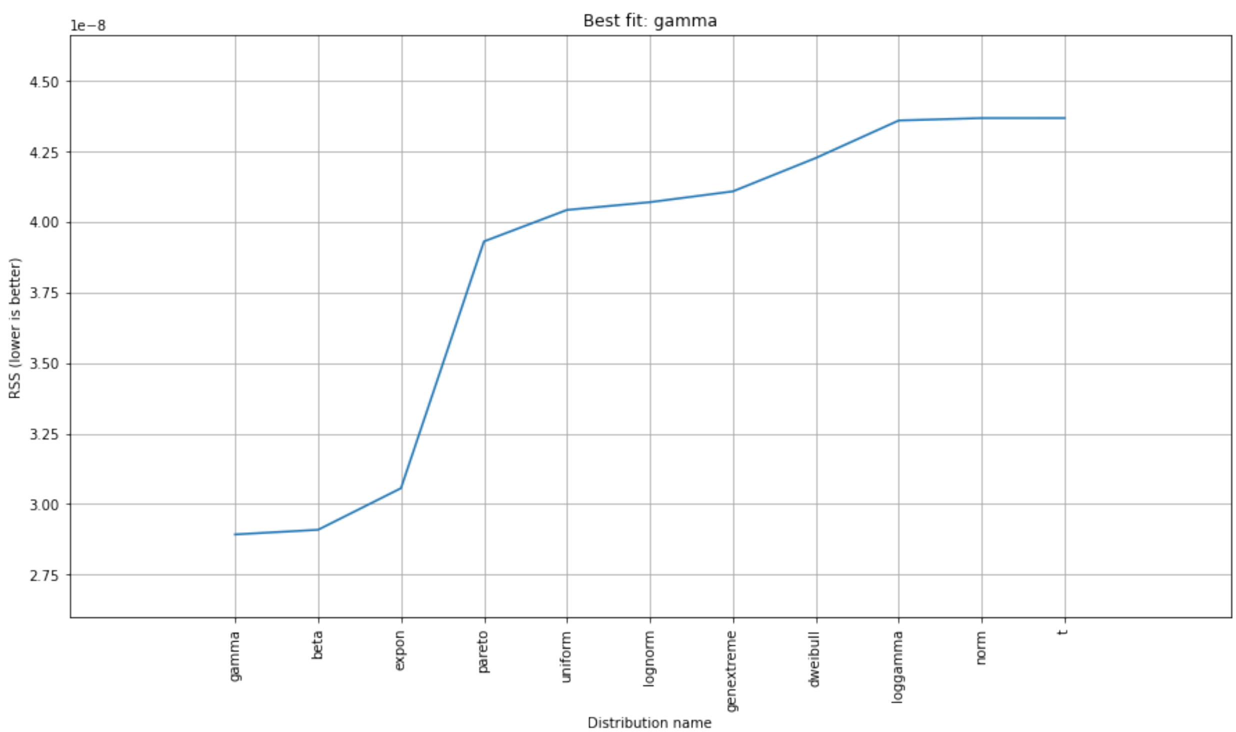

Best fit: pareto
RSS (lower is better)
pareto
lognorm
genextreme
t
gamma
expon
dweibull
beta
norm
loggamma
uniform
Distribution name

water_kcal

In [47]:
```
define_analytic(df2.water_kcal)
```

```
[distfit] >fit..
[distfit] >transform..
[distfit] >[norm      ] [0.00 sec] [RSS: 4.36911e-08] [loc=17380.575 scale=159
59.253]
[distfit] >[expon     ] [0.00 sec] [RSS: 3.05639e-08] [loc=4.095 scale=17376.4
80]
[distfit] >[pareto    ] [0.02 sec] [RSS: 3.93162e-08] [loc=-0.000 scale=4.095]
[distfit] >[dweibull  ] [0.03 sec] [RSS: 4.22757e-08] [loc=14256.949 scale=145
83.629]
[distfit] >[t         ] [0.08 sec] [RSS: 4.36916e-08] [loc=17381.828 scale=159
59.695]
[distfit] >[genextreme] [0.24 sec] [RSS: 4.10913e-08] [loc=7.430 scale=21.052]
[distfit] >[gamma     ] [0.09 sec] [RSS: 2.89244e-08] [loc=4.095 scale=59377.4
03]
[distfit] >[lognorm   ] [0.12 sec] [RSS: 4.07089e-08] [loc=4.095 scale=10.844]
[distfit] >[beta      ] [0.11 sec] [RSS: 2.90887e-08] [loc=4.095 scale=62464.6
85]
[distfit] >[uniform   ] [0.00 sec] [RSS: 4.04326e-08] [loc=4.095 scale=49731.7
87]
[distfit] >[loggamma  ] [0.07 sec] [RSS: 4.36061e-08] [loc=-5044470.683 scale=
678463.825]
[distfit] >Compute confidence interval [parametric]
[distfit] >plot..
[distfit] >plot summary..
         distr score  LLE            loc          scale  \
0        gamma   0.0  NaN       4.095023   59377.402543
1         beta   0.0  NaN       4.095023   62464.684641
2        expon   0.0  NaN       4.095023   17376.480386
3       pareto   0.0  NaN      -0.000482       4.095488
4      uniform   0.0  NaN       4.095023   49731.787327
5      lognorm   0.0  NaN       4.095023      10.843799
6   genextreme   0.0  NaN       7.430153      21.052408
7     dweibull   0.0  NaN   14256.948547   14583.629366
8     loggamma   0.0  NaN -5044470.683269  678463.825178
9         norm   0.0  NaN   17380.575408   15959.252707
10           t   0.0  NaN   17381.827934    15959.69517

                                      arg
0                    (0.36164828995329423,)
1   (0.3499894231004219, 1.4617720611957328)
2                                        ()
3                    (0.13370587569177017,)
4                                        ()
5                       (7.49990082800343,)
6                     (-6.312301725362163,)
7                      (1.380689681400706,)
8                     (1738.8839578786528,)
9                                        ()
10                    (1825896.1047807992,)
```

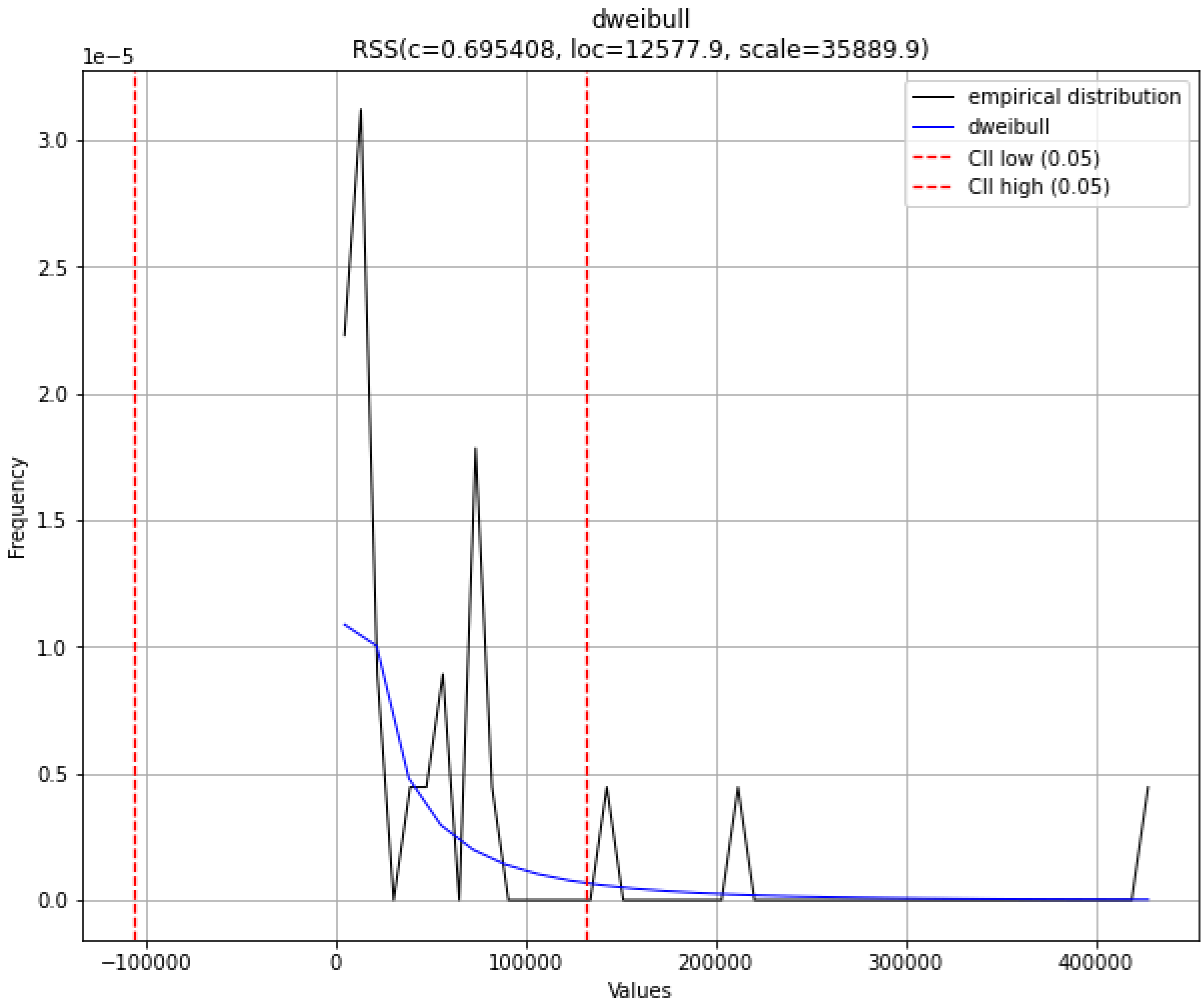


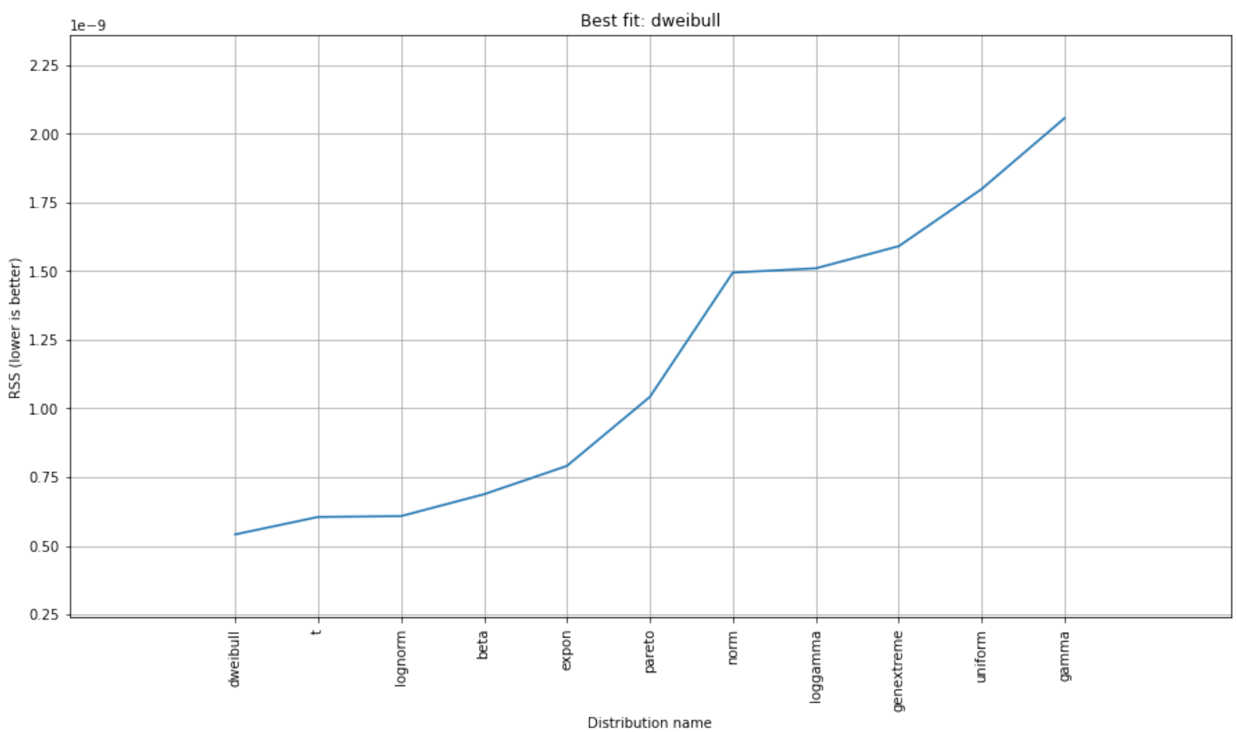


water_protein

In [48]:
```
define_analytic(df2.water_protein)
```

```
[distfit] >fit..
[distfit] >transform..
[distfit] >[norm      ] [0.00 sec] [RSS: 1.49504e-09] [loc=59196.439 scale=881
81.845]
[distfit] >[expon     ] [0.00 sec] [RSS: 7.90702e-10] [loc=421.250 scale=5877
5.189]
[distfit] >[pareto    ] [0.03 sec] [RSS: 1.04226e-09] [loc=-0.592 scale=421.84
2]
[distfit] >[dweibull  ] [0.05 sec] [RSS: 5.41143e-10] [loc=12577.948 scale=358
89.913]
[distfit] >[t         ] [0.08 sec] [RSS: 6.04741e-10] [loc=14212.003 scale=103
20.180]
[distfit] >[genextreme] [0.24 sec] [RSS: 1.59112e-09] [loc=422.655 scale=8.88
6]
[distfit] >[gamma     ] [0.06 sec] [RSS: 2.0579e-09] [loc=-675682963.313 scale
=108886.657]
[distfit] >[lognorm   ] [0.05 sec] [RSS: 6.07847e-10] [loc=-1098.053 scale=291
81.152]
[distfit] >[beta      ] [0.10 sec] [RSS: 6.87468e-10] [loc=421.250 scale=74412
43.966]
[distfit] >[uniform   ] [0.00 sec] [RSS: 1.79966e-09] [loc=421.250 scale=43119
8.750]
[distfit] >[loggamma  ] [0.07 sec] [RSS: 1.51055e-09] [loc=-33598462.347 scale
=4365586.371]
[distfit] >Compute confidence interval [parametric]
[distfit] >plot..
[distfit] >plot summary..
         distr score  LLE              loc           scale  \
0     dweibull   0.0  NaN      12577.94779    35889.913322
1            t   0.0  NaN     14212.003189    10320.179744
2      lognorm   0.0  NaN     -1098.053108    29181.151638
3         beta   0.0  NaN           421.25  7441243.966318
4        expon   0.0  NaN           421.25    58775.188503
5       pareto   0.0  NaN         -0.59166      421.841647
6         norm   0.0  NaN     59196.438503    88181.844626
7     loggamma   0.0  NaN -33598462.346532  4365586.371324
8   genextreme   0.0  NaN       422.654704        8.885533
9      uniform   0.0  NaN           421.25       431198.75
10       gamma   0.0  NaN -675682963.31304   108886.656545

                                     arg
0                    (0.6954082767074626,)
1                     (0.685550658727031,)
2                    (1.2248496234884045,)
3   (0.6754187369806032, 92.88370376059186)
4                                       ()
5                   (0.24197409851261706,)
6                                       ()
7                    (2230.4190908354403,)
8                    (-6.325544260867353,)
9                                       ()
10                   (6205.4227884323245,)
```



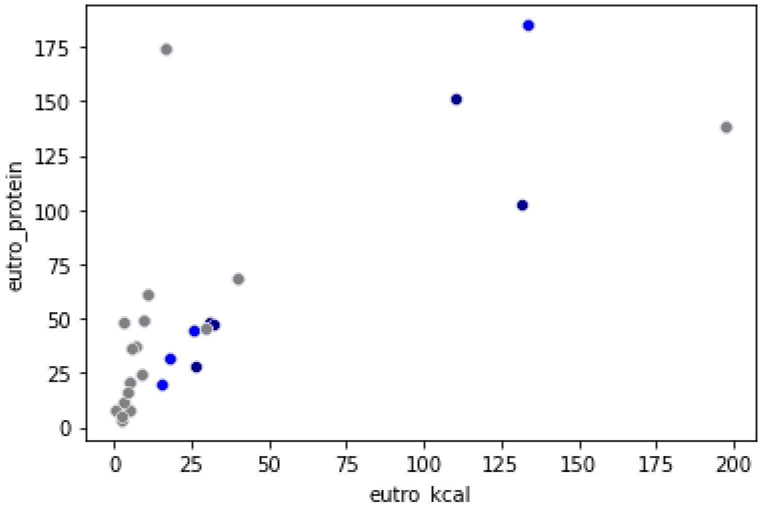


Scatterplots

eutro_kcal versus eutro_protein

In [49]:
```
sns.scatterplot(data=df2, x="eutro_kcal", y="eutro_protein", color = "darkblue'
sns.scatterplot(data=veg, x="eutro_kcal", y="eutro_protein", color = "blue")
sns.scatterplot(data=vegan, x="eutro_kcal", y="eutro_protein", color = "grey")
```

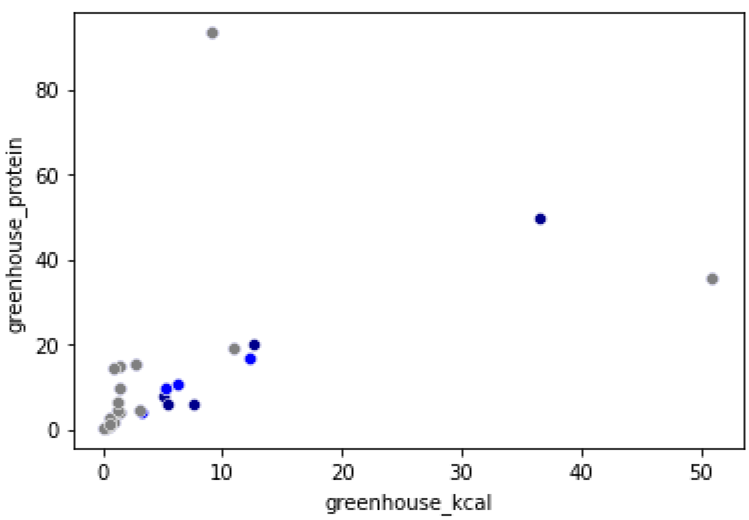
Out[49]: <AxesSubplot:xlabel='eutro_kcal', ylabel='eutro_protein'>



greenhouse_kcal versus greenhouse_protein

In [50]:
```
sns.scatterplot(data=df2, x="greenhouse_kcal", y="greenhouse_protein", color =
sns.scatterplot(data=veg, x="greenhouse_kcal", y="greenhouse_protein", color =
sns.scatterplot(data=vegan, x="greenhouse_kcal", y="greenhouse_protein", color
```

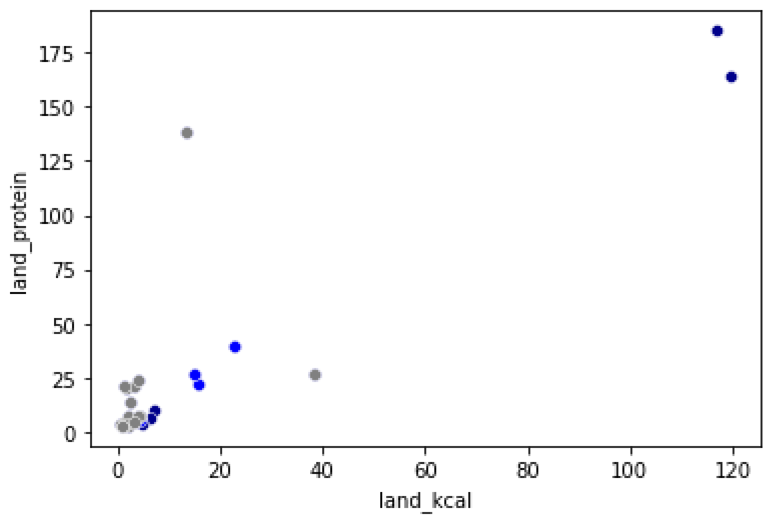
Out[50]: <AxesSubplot:xlabel='greenhouse_kcal', ylabel='greenhouse_protein'>



land_kcal versus land_protein

In [51]:
```
sns.scatterplot(data=df2, x="land_kcal", y="land_protein", color = "darkblue")
sns.scatterplot(data=veg, x="land_kcal", y="land_protein", color = "blue")
sns.scatterplot(data=vegan, x="land_kcal", y="land_protein", color = "grey")
```

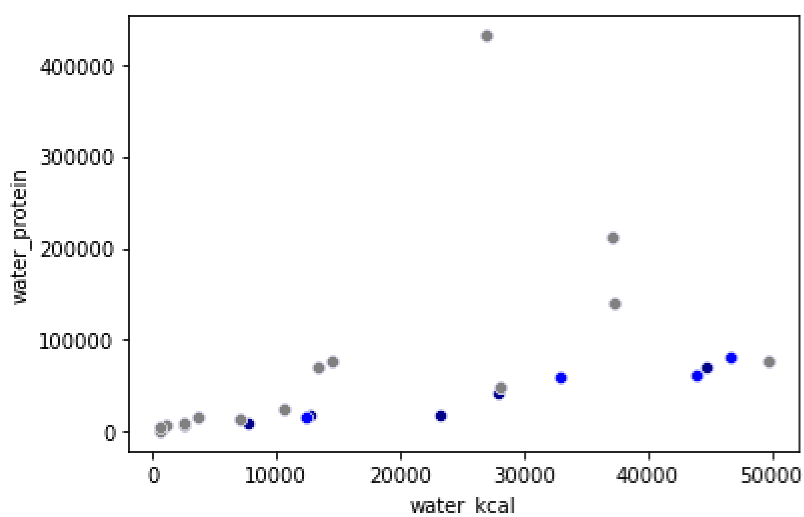
Out[51]: <AxesSubplot:xlabel='land_kcal', ylabel='land_protein'>



water_kcal versus water_protein

In [52]:
```
sns.scatterplot(data=df2, x="water_kcal", y="water_protein", color = "darkblue'
sns.scatterplot(data=veg, x="water_kcal", y="water_protein", color = "blue")
sns.scatterplot(data=vegan, x="water_kcal", y="water_protein", color = "grey")
```

Out[52]: <AxesSubplot:xlabel='water_kcal', ylabel='water_protein'>



Since I am graphing the same environmental impact, but with two different measurements, against one another I had assumed there would be a great sense of linearity. These plots really show the extremity of the outliers, and how far from the rest of the data they are. For the sake of pure statistical analysis, these outliers should aboslutely be removed because it really shows how much these extreme values skew the distribution- however, there is no reason to believe that these outliers are mistakes in data collection. In fact, they are important in the data as a whole.

Hypothesis test

total_em

In [53]:
```
# vegetarian less total emissions than animal-based
ss.mannwhitneyu(x = veg.total_em, y = df2.total_em, alternative = "less")
```

Out[53]:
```
MannwhitneyuResult(statistic=705.5, pvalue=0.1935051341490287)
```

In [54]:
```
# vegan less total emissions than vegetarian
ss.mannwhitneyu(x = vegan.total_em, y = veg.total_em, alternative = "less")
```

Out[54]:
```
MannwhitneyuResult(statistic=558.5, pvalue=0.2718039981752386)
```

In [55]:
```
# vegan less total emissions than animal-based
ss.mannwhitneyu(x = vegan.total_em, y = df2.total_em, alternative = "less")
```

Out[55]:
```
MannwhitneyuResult(statistic=571.5, pvalue=0.07441556070492732)
```

eutro_kcal

In [56]:
```
# variables without Na's
veg_eutro_kcal = [item for item in veg.eutro_kcal if not(math.isnan(item)) == T
df2_eutro_kcal = [item for item in df2.eutro_kcal if not(math.isnan(item)) == T
vegan_eutro_kcal = [item for item in vegan.eutro_kcal if not(math.isnan(item))
```

In [57]:
```
# vegetarian less eutrophying emissions per 1000kcal than animal-based
ss.mannwhitneyu(x = veg_eutro_kcal, y = df2_eutro_kcal, alternative = "less")
```

Out[57]:
```
MannwhitneyuResult(statistic=406.0, pvalue=0.21082861788431284)
```

In [58]:
```
# vegan less eutrophying emissions per 1000kcal than vegetarian
ss.mannwhitneyu(x = vegan_eutro_kcal, y = veg_eutro_kcal, alternative = "less"
```

Out[58]:
```
MannwhitneyuResult(statistic=299.0, pvalue=0.25132626054704515)
```

In [59]:
```
# vegan less eutrophying emissions per 1000kcal than animal-based
ss.mannwhitneyu(x = vegan_eutro_kcal, y = df2_eutro_kcal, alternative = "less"
```

Out[59]:
```
MannwhitneyuResult(statistic=308.0, pvalue=0.07856548215397303)
```

eutro_protein

In [60]:
```
# variables without Na's
veg_eutro_protein = [item for item in veg.eutro_protein if not(math.isnan(item
df2_eutro_protein = [item for item in df2.eutro_protein if not(math.isnan(item
vegan_eutro_protein = [item for item in vegan.eutro_protein if not(math.isnan(i
```

In [61]:
```
# vegetarian less eutrophying emissions per 100g protein than animal-based
ss.mannwhitneyu(x = veg_eutro_protein, y = df2_eutro_protein, alternative = "le
```

Out[61]:
```
MannwhitneyuResult(statistic=272.0, pvalue=0.3110963363416528)
```

In [62]:
```
# vegan less eutrophying emissions per 100g protein than vegetarian
ss.mannwhitneyu(x = vegan_eutro_protein, y = veg_eutro_protein, alternative = '
```
Out[62]:
```
MannwhitneyuResult(statistic=189.0, pvalue=0.4085488642097164)
```

In [63]:
```
# vegan less eutrophying emissions per 100g protein than animal-based
ss.mannwhitneyu(x = vegan_eutro_protein, y = df2_eutro_protein, alternative = '
```
Out[63]:
```
MannwhitneyuResult(statistic=212.0, pvalue=0.23976723529803945)
```

greenhouse_kcal

In [64]:
```
# variables without Na's
veg_greenhouse_kcal = [item for item in veg.greenhouse_kcal if not(math.isnan(
df2_greenhouse_kcal = [item for item in df2.greenhouse_kcal if not(math.isnan(
vegan_greenhouse_kcal = [item for item in vegan.greenhouse_kcal if not(math.is
```

In [65]:
```
# vegetarian less greenhouse emissions per 1000kcal than animal-based
ss.mannwhitneyu(x = veg_greenhouse_kcal, y = df2_greenhouse_kcal, alternative =
```
Out[65]:
```
MannwhitneyuResult(statistic=409.0, pvalue=0.2235931237125317)
```

In [66]:
```
# vegan less greenhouse emissions per 1000kcal than vegetarian
ss.mannwhitneyu(x = vegan_greenhouse_kcal, y = veg_greenhouse_kcal, alternative
```
Out[66]:
```
MannwhitneyuResult(statistic=298.0, pvalue=0.2455101212618333)
```

In [67]:
```
# vegan less greenhouse emissions per 1000kcal than animal-based
ss.mannwhitneyu(x = vegan_greenhouse_kcal, y = df2_greenhouse_kcal, alternative
```
Out[67]:
```
MannwhitneyuResult(statistic=309.0, pvalue=0.08096378575893709)
```

greenhouse_protein

In [68]:
```
# variables without NA's
veg_greenhouse_protein = [item for item in veg.greenhouse_protein if not(math.
df2_greenhouse_protein = [item for item in df2.greenhouse_protein if not(math.
vegan_greenhouse_protein = [item for item in vegan.greenhouse_protein if not(ma
```

In [69]:
```
# vegetarian less greenhouse emissions per 100g protein than animal-based
ss.mannwhitneyu(x = veg_greenhouse_protein, y = df2_greenhouse_protein, altern
```
Out[69]:
```
MannwhitneyuResult(statistic=279.0, pvalue=0.3624338230013679)
```

In [70]:
```
# vegan less greenhouse emissions per 100g protein than vegetarian
ss.mannwhitneyu(x = vegan_greenhouse_protein, y = veg_greenhouse_protein, alte
```
Out[70]:
```
MannwhitneyuResult(statistic=189.0, pvalue=0.4085488642097164)
```

In [71]:
```
# vegan less greenhouse emissions per 100g protein than animal-based
ss.mannwhitneyu(x = vegan_greenhouse_protein, y = df2_greenhouse_protein, alte
```

```
Out[71]: MannwhitneyuResult(statistic=217.0, pvalue=0.27721283362388954)
```

land_kcal

```
In [72]: # variables without Na's
veg_land_kcal = [item for item in veg.land_kcal if not(math.isnan(item)) == Tru
df2_land_kcal = [item for item in df2.land_kcal if not(math.isnan(item)) == Tru
vegan_land_kcal = [item for item in vegan.land_kcal if not(math.isnan(item)) ==
```

```
In [73]: # vegetarian less use of land per 1000kcal than animal-based
ss.mannwhitneyu(x = veg_land_kcal, y = df2_land_kcal, alternative = "less")
```

```
Out[73]: MannwhitneyuResult(statistic=407.0, pvalue=0.21503523567746058)
```

```
In [74]: # vegan less use of land per 1000kcal than vegetarian
ss.mannwhitneyu(x = vegan_land_kcal, y = veg_land_kcal, alternative = "less")
```

```
Out[74]: MannwhitneyuResult(statistic=294.0, pvalue=0.2229861444937108)
```

```
In [75]: # vegan less use of land per 1000kcal than animal-based
ss.mannwhitneyu(x = vegan_land_kcal, y = df2_land_kcal, alternative = "less")
```

```
Out[75]: MannwhitneyuResult(statistic=300.0, pvalue=0.06127734021854017)
```

land_protein

```
In [76]: # variables without Na's
veg_land_protein = [item for item in veg.land_protein if not(math.isnan(item))
df2_land_protein = [item for item in df2.land_protein if not(math.isnan(item))
vegan_land_protein = [item for item in vegan.land_protein if not(math.isnan(ite
```

```
In [77]: # vegetarian less use of land per 100g protein than animal-based
ss.mannwhitneyu(x = veg_land_protein, y = df2_land_protein, alternative = "less
```

```
Out[77]: MannwhitneyuResult(statistic=283.0, pvalue=0.3929983926043856)
```

```
In [78]: # vegan less use of land per 100g protein than vegetarian
ss.mannwhitneyu(x = vegan_land_protein, y = veg_land_protein, alternative = "le
```

```
Out[78]: MannwhitneyuResult(statistic=178.0, pvalue=0.2978541615314826)
```

```
In [79]: # vegan less use of land per 100g protein than animal-based
ss.mannwhitneyu(x = vegan_land_protein, y = df2_land_protein, alternative = "le
```

```
Out[79]: MannwhitneyuResult(statistic=209.0, pvalue=0.21869812209072598)
```

water_kcal

```
In [80]: # variables without Na's
veg_water_kcal = [item for item in veg.water_kcal if not(math.isnan(item)) == T
df2_water_kcal = [item for item in df2.water_kcal if not(math.isnan(item)) == T
vegan_water_kcal = [item for item in vegan.water_kcal if not(math.isnan(item))
```

In [81]:
```
# vegetarian less use of water per 1000kcal than animal-based
ss.mannwhitneyu(x = veg_water_kcal, y = df2_water_kcal, alternative = "less")
```

Out[81]:
```
MannwhitneyuResult(statistic=353.5, pvalue=0.36124798828098603)
```

In [82]:
```
# vegan less use of water per 1000kcal than vegetarian
ss.mannwhitneyu(x = vegan_water_kcal, y = veg_water_kcal, alternative = "less"
```

Out[82]:
```
MannwhitneyuResult(statistic=233.5, pvalue=0.264704078995766)
```

In [83]:
```
# vegan less use of water per 1000kcal than animal-based
ss.mannwhitneyu(x = vegan_water_kcal, y = df2_water_kcal, alternative = "less"
```

Out[83]:
```
MannwhitneyuResult(statistic=260.5, pvalue=0.15056900842093546)
```

water_protein

In [84]:
```
# variables without Na's
veg_water_protein = [item for item in veg.water_protein if not(math.isnan(item
df2_water_protein = [item for item in df2.water_protein if not(math.isnan(item
vegan_water_protein = [item for item in vegan.water_protein if not(math.isnan(
```

In [85]:
```
# vegetarian less use of water per 100g protein than animal-based
ss.mannwhitneyu(x = veg_water_protein, y = df2_water_protein, alternative = "le
```

Out[85]:
```
MannwhitneyuResult(statistic=275.5, pvalue=0.525607681182107)
```

In [86]:
```
# vegan less use of water per 100g protein than vegetarian
ss.mannwhitneyu(x = vegan_water_protein, y = veg_water_protein, alternative = '
```

Out[86]:
```
MannwhitneyuResult(statistic=167.5, pvalue=0.37883402069503974)
```

In [87]:
```
# vegan less use of water per 100g protein than animal-based
ss.mannwhitneyu(x = vegan_water_protein, y = df2_water_protein, alternative = '
```

Out[87]:
```
MannwhitneyuResult(statistic=208.5, pvalue=0.3827475718105846)
```

The idea here was to run hypothesis tests across the board of the varaibles selected. Under my own initial assumption that plant-based diets are better for the environment than animal-based diets, I had wished to complicate this idea by trying to see if they are better across the board or only better across a few environmental impact measures- for example, a vegetarian diet might produce less greenhouse gas emissions during production than an animal-based diet (it does not) but it does not make a difference in the amount of water used to produce the food (which is technically true).

I will admit that I did not originally compare vegan food products to animal-based products (I had originally only compared vegetarian to animal-based and vegan to vegetarian which makes very little sense on why it was left out based on the research question). That being said, the addition of the hypothesis tests that see if vegan food products have less of an environmental impact than animal-based products did not result in any significant results at the 95% level. There were a few instances (total_em, eutro_kcal, greenhouse_kcal, and land_kcal) that would be significant at a 90% level; but since I had intended to use the 95% level from the beginning, it is still true that there is no reason to believe that diet has an environmental impact in any way, based on this data set. In all of the tests, the alternative hypothesis is rejected in favor of the null hypothesis: there is no signifcant difference in environmental impact based on food production.

Regression Analysis

In [88]:
```
results = smf.ols("total_em ~ land_kcal + eutro_kcal + water_kcal + greenhouse_
results.summary()
```

Out[88]:

OLS Regression Results

Dep. Variable:	total_em	**R-squared:**	0.772
Model:	OLS	**Adj. R-squared:**	0.736
Method:	Least Squares	**F-statistic:**	21.22
Date:	Wed, 16 Nov 2022	**Prob (F-statistic):**	9.78e-08
Time:	20:41:58	**Log-Likelihood:**	-94.748
No. Observations:	30	**AIC:**	199.5
Df Residuals:	25	**BIC:**	206.5
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.5255	1.833	1.377	0.181	-1.251	6.302
land_kcal	0.3146	0.060	5.283	0.000	0.192	0.437
eutro_kcal	0.0474	0.052	0.903	0.375	-0.061	0.155
water_kcal	-5.073e-05	8.25e-05	-0.615	0.544	-0.000	0.000
greenhouse_kcal	0.0424	0.277	0.153	0.880	-0.529	0.613

Omnibus:	3.804	**Durbin-Watson:**	1.976
Prob(Omnibus):	0.149	**Jarque-Bera (JB):**	2.387
Skew:	0.406	**Prob(JB):**	0.303
Kurtosis:	4.118	**Cond. No.**	3.80e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.8e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Summary

Statistical/Hypothetical Question

This study aimed to explore the relationship between diet and the environment. It was meant to test the claim that pant-based diets are better for the environment than traditional ones with animal-products. The different diets that were tested was a vegetarian diet that includes some animal products (such as eggs, dairy, cheese, honey) but no meant or seafood, a vegan diet that excludes all animal products, and a traditional “animal-based” diet that excludes none of the food products.

Outcome of EDA

From this data set, it is impossible to reject any of the null hypotheses. Across every environmental impact that was measured, there was no evidence that a plant-based diet (either vegetarian or vegan) was any better for the environment than the animal-based diet. This was concluded based on a non-parametric test (Mann Whitney U test) because no variable was normally distributed.

What was missed during the analysis

I feel like I could have focused more on the relationship between the kcal and protein variable pairs (see Scatterplots section above) to see what kind of information could have been gained from understanding the base of those relationships and trends.

Variables that could have helped with analysis

I think it would have been interesting to look at the emissions that each food product emits due to transportation during production (Transport). In that vein, looking at the emissions at each step of the production process and how they contributed to the final total might have been interesting, as some steps might have contributed more for one of the diet scenarios than the others.

Incorrect assumptions

I really ran with the idea assumption that plant-based diets are better for the environment than traditional animal-based diets, as that really shaped my questions and the analysis performed. If I were to step back and look at the data more closely at the beginning, I might have spent more time doing some simple EDA with all the variables (as I did choose the variables ahead of time based on my research question and initial assumptions).

Challenges

I think the nature of the data set that I chose really became a problem for me. When I first chose this data set at the beginning of this term, I was not very confident in my abilities nor was I very observant when I was initially looking at the data. Because of the size, I do not think my findings (if they were to even be statistically significant within the scope of the study) would be very implicative of the world at large. I also really struggled with determining the equation for the regression analysis- it took me some time to decide what to use for the response variable, as none of the variables suggested a natural linear relationship that would be meaningful. I also did not spend a lot of time looking at the linear relationships between the environmental factors and their two different measurements (see Scatterplots section above), and I think that could have provided interesting information that I did not take the time to fully explore or understand.