Florida Bay food web

Patrik Cavina

2025-12-17

Analisi della food web Florida Bay nella stagione delle piogge

Introduzione

Viene definita **food web** l'insieme delle catene alimentari all'interno di un ecosistema, che si collegano tra di loro formando una rete alimentare. I collegamenti all'interno di una food web indicano i trasferimenti di energia tra specie, ovvero quale specie viene mangiata da un'altra.

La seguente analisi studia la food web della Florida bay nel periodo delle piogge. L'analisi inizia con il caricamento del dataset in formato paj e il preprocessamento dei dati calcolando il livello trofico per ogni specie. Successivamente vengono poste otto domande di ricerca, alla quale sarà data risposta in ogni capitolo dell'analisi.

```
library(jsonlite)
library(readr)
library(igraph)
library(tidygraph)
library(tidyverse)
library(dplyr)
library(ggraph)
library(reshape2)
library(graphlayouts)

library(lpSolve)
library(lpSolveAPI)

library(patchwork)

get_tl_colors = function() {
  colors = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A", "#E6AB02")
  return(colors)
}
```

Dataset

> Origine del dataset: * food web (http://konect.cc/networks/foodweb-baywet/) il cui corrispondente file .paj viene ricavato da .paj (https://web.archive.org/web/20221115182159/http://vlado.fmf.uni-lj.si/pub/networks/data/bio/foodweb/baywet.paj)

Il dataset è una food web contenente le interazioni tra le specie presenti nella Florida Bay, un'ampia laguna all'estremo sud della florida, durante la stagione delle pioggie. Ogni nodo della rete rappresenta una taxa (specie, genere o famiglia), mentre ogni arco direzionato indica che una taxa ne usa un'altra come cibo (es. un arco tra i e j, indica che j usa i come cibo).

```
load_florida_graph = function(path) {
  stopifnot(file.exists(path))

  x = readLines(path, warn = FALSE)

  # --- 1) Estrai e leggi Network ---
  i0 = grep("^\\*Network\\b", x, ignore.case = TRUE)[1]
  if (is.na(i0)) stop("Nessuna sezione *Network trovata.")

  next_sec = grep("^\\*(Partition|Vector|Cluster)\\b", x[-seq_len(i0)], ignore.case = TRUE)
[1]
  i1 = if (is.na(next_sec)) length(x) else i0 + next_sec - 1

  tmp = tempfile(fileext = ".net")
  on.exit(unlink(tmp))
  writeLines(x[i0:i1], tmp, useBytes = TRUE)
  g = read_graph(tmp, format = "pajek")

  # --- 2) Helper per leggere Partition/Vector ---
  read_vals = function(pattern) {
    h = grep(pattern, x, ignore.case = TRUE)[1]
    v_line = h + grep("^\\*Vertices\\b", x[-(1:h)], ignore.case = TRUE)[1]
    n = as.integer(sub("^\\*Vertices\\s+", "", x[v_line], ignore.case = TRUE))
    as.numeric(x[(v_line + 1):(v_line + n)])
  }

  # --- 3) Assegna attributi ---
  V(g)$eco_type = as.integer(read_vals("^\\*Partition\\s+ECO types\\b"))
  V(g)$bio_mass = read_vals("^\\*Vector\\s+bio-masses\\b")

  # --- 4) Rinomina primo attributo archi in "carbon_flow" ---
  if (length(igraph::edge_attr_names(g)) > 0) {
    igraph::E(g)$carbon_flow <- igraph::edge_attr(g)[[1]]
    g <- igraph::delete_edge_attr(g, igraph::edge_attr_names(g)[1])
  }

  return(g)
}

# Caricamento della food web
fw_g_io = load_florida_graph("data/florida.paj")
```

Il dataset è formato dal seguente numero di nodi e archi

```
cat("# Nodi: ", vcount(fw_g_io), "\n")
```

```
## # Nodi:  128
```

```
cat("# Archi: ", ecount(fw_g_io))
```

```
## # Archi:  2106
```

Ogni nodo è composto dai seguenti attributi:

- name: nome della taxa
- eco_type: ruolo nella food web
  - 1 = taxa
  - 2 = detriti (“Particulate Organic Carbon” e “Dissolved Organic Carbon”)
  - 3 = input (energia proveniente dall’esterno della food web)
  - 4 = output (energia dispersa all’esterno della food web)
  - 5 = respiration
- bio_mass: quantità di biomassa, misurata in $\frac{\text{g C}}{m^2}$

All’interno della rete i nodi indicati come detriti sono: * Water POC: materiale organico di dimensione millimetrica, presente nell’acqua * Benthic POC: materiale organico di dimensione millimetrica, presente sui fondali (detriti) * DOC: materia organica di dimensioni microscopiche, funge da nutrimento per batteri e altri microrganismi

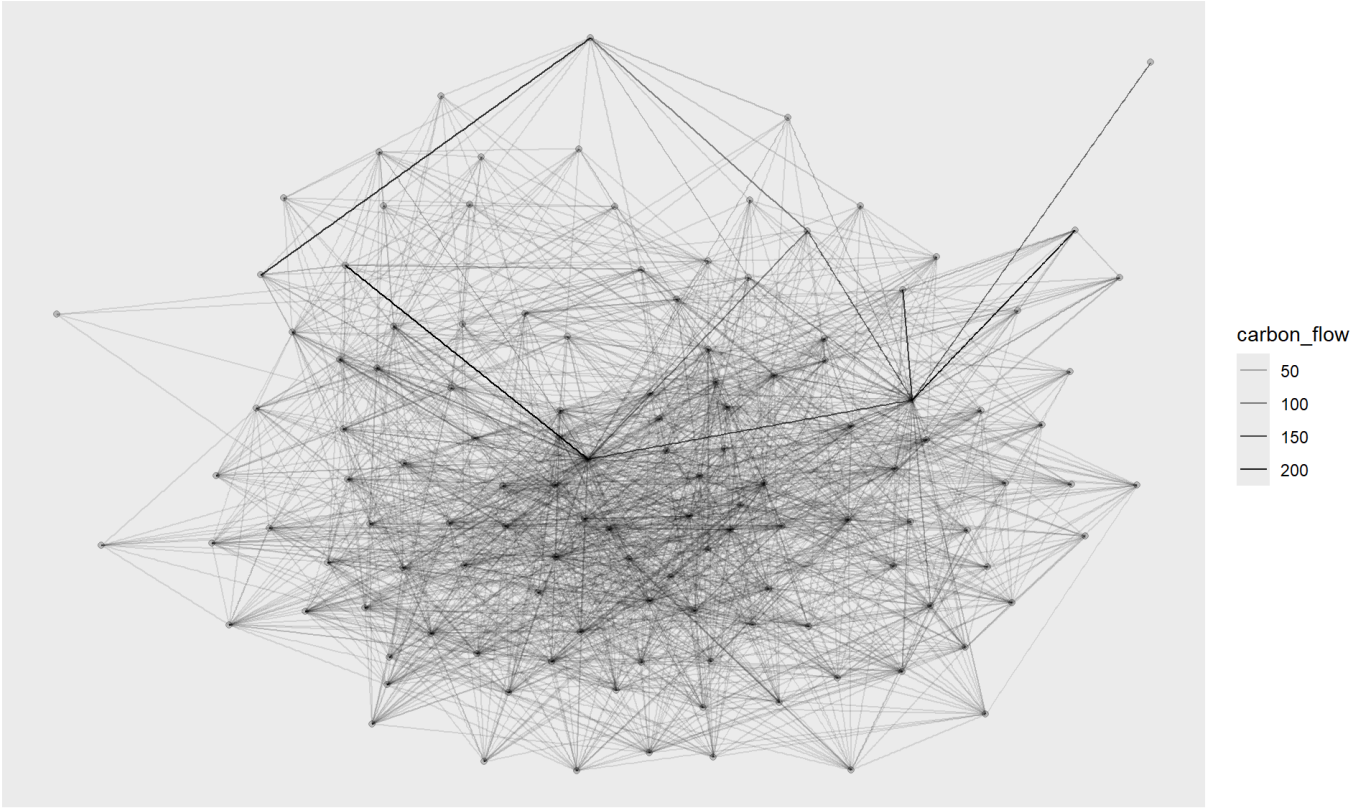
Ogni arco è etichettato dai seguenti attributi:

- carbon_flow: quantità di carbonio scambiata tra le due taxa, misurata in $\frac{\text{g C}}{m^2 \cdot year}$

Oltre al grafo `fw_g_io` vengono preparati anche:

- fw_g: grafo senza i vertici “Input”, “Output” e “Respiration”
- A: matrice di adiacenza associata a “fw_g”, dove avremo:
  - righe: indicano produttori
  - colonne: indicano consumatori
- A_norm: matrice di adicenza normalizzata per colonne, in questo modo otteniamo per ogni consumatore la percentuale che coprono i produttori nella sua dieta.

```
# Rimuoviamo i vertici Input, Output, Respiration
fw_g = delete_vertices(fw_g_io, c('Input', 'Output', 'Respiration'))

# Righe: preda
# Colonne: predatore
A = as_adjacency_matrix(fw_g, names = TRUE, attr = "carbon_flow", sparse = FALSE)

# Normalizziamo per colonne per trovare le percentuali di ogni specie nella dieta di un preda
tore
A_norm = apply(A, 2, function(x)(x / sum(x)))
A_norm[is.nan(A_norm)] = 0

ggraph(fw_g, layout = "lgl") +
  geom_node_point(alpha = 0.2) +
  geom_edge_link2(aes(alpha = carbon_flow))
```



> I termini taxa e specie verrano usati in maniera intercambiabile.

Trophic level

Il **livello trofico** (TL) è un numero intero che indica la posizione di un organismo all'interno della food web. Una food web è composta da diverse catene alimentari, successione di organismi che mangiano altri organismi, e il livello trofico indica quanti passi separano la specie dall'inizio della catena. I valori possibili possono variare da 1 a 4 o 5, è raro trovare specie con livelli superiori. Avremo quindi:

- livello 1: Produttori primari (come le piante)
- livello 2: Erbivori
- livello 3: Carnivori/Onnivori
- livello $$4: Predatori apicali

Dato che non è sempre possibile assegnare una posizione intera all'interno della food web, possiamo usare la **fractional trophic level (TL)** per assegnare il livello trofico a ogni specie.

Il TL viene definito come:

$$TL_i = 1 + \sum_j (TL_j \cdot DC_{ij})$$

con:

- TL_i livello trofico della specie i
- TL_j livello trofico della specie j predata da i
- DC_{ij} frazione della specie i nella dieta della specie j

In forma matriciale:

$$TL = 1 + DC \cdot TL$$

Possiamo ridurci a un sistema del tipo $Ax = b$ per calcolare il livello trofico di ogni specie, infatti:

$$TL - DC \cdot TL = 1$$
$$(I - DC) \cdot TL = 1$$

Dove TL è il vettore delle incognite x e $(I - DC)$ la matrice nota A.

```
# TL calcola il livello trofico per ogni specie nella matrice DC.
#
# INPUT
# DC: matrice delle frazioni di dieta, sulle righe sono presenti i predatori e sulle colonne
le prede.
#
# OUTPUT
# tl: vettore dei livelli trofici per ogni specie
TL = function (DC) {
  n  = nrow(DC)
  I  = diag(n)
  b  = rep(1, n)

  tl = solve(I - DC, b)
  names(tl) = rownames(DC)
  return(tl)
}
```

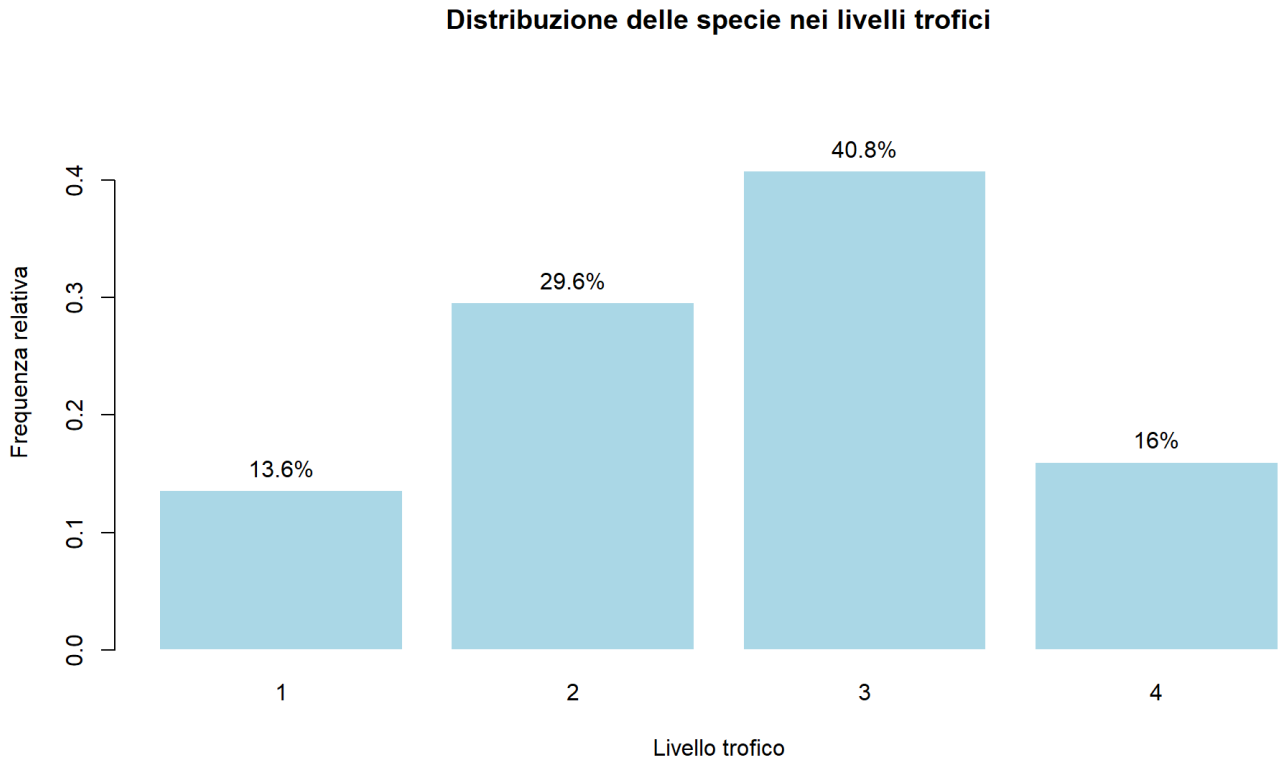
Calcoliamo il livello trofico di ogni specie

```
# Rimozione self-loops (cannibalismo)
diag(A_norm) = 0

# Rimuoviamo gli archi entranti per i detriti POC e DOC, in questo modo questi
# vertici risultano come risorse basali (livello trofico 1)
A_norm2 = A_norm
A_norm2[, "Water POC"] = 0
A_norm2[, "Benthic POC"] = 0
A_norm2[, "DOC"] = 0

# Livello trofico per ogni specie
tls = TL(t(A_norm2))

# Livello trofico intero per ogni specie
tls_int = as.integer(tls)
names(tls_int) = names(tls)

# Assegniamo il livello trofico come attributo dei vertici
V(fw_g)$trophic_lvl = tls[V(fw_g)$name]

# Assegniamo il livello trofico intero come attributo dei vertici
V(fw_g)$trophic_lvl_int = tls_int[V(fw_g)$name]

# Assegniamo il livello trofico anche per la versione del grafo con input e output
fw_g_io = fw_g_io %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  left_join(
    igraph::as_data_frame(fw_g, what = "vertices")
      %>% select(name, trophic_lvl, trophic_lvl_int),
    by = 'name'
  ) %>%
  mutate(
    trophic_lvl = case_when(
      name == "Input" ~ 0, # Consideriamo l'input come livello trofico 0
      name == "Output" ~ 5, # Consideriamo l'output come livello trofico 5
      TRUE ~ trophic_lvl
    ),
    trophic_lvl_int = case_when(
      name == "Input" ~ 0.0,
      name == "Output" ~ 5.0,
      TRUE ~ trophic_lvl_int
    )
  ) %>%
  as.igraph()

# Riassunto dei livelli trofici
summary(tls)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.201   3.278   2.931   3.619   4.602
```

Visualizziamo la distribuzione dei livelli trofici all'interno della rete.

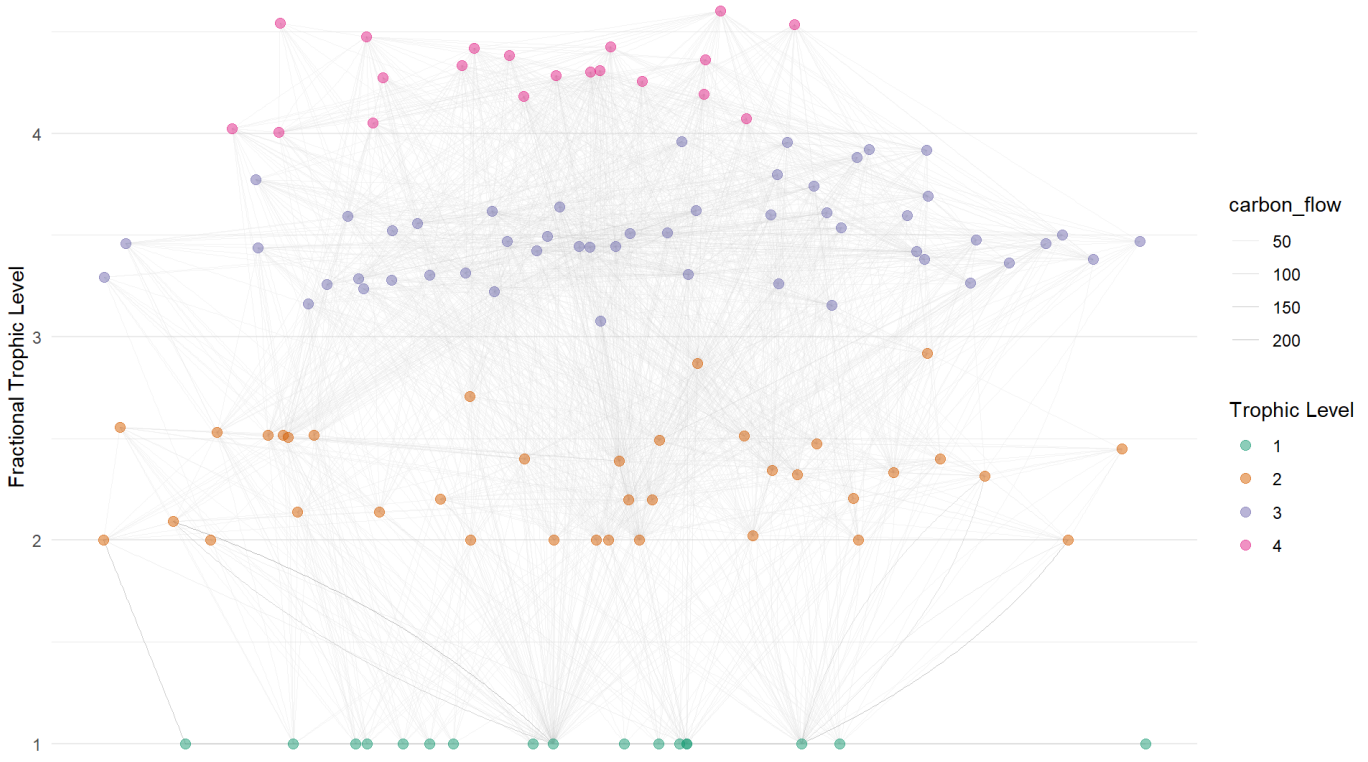
```
# Calcolo la frazioni di specie per ogni ogni livello trofico
frac = table(tls_int) / length(tls_int)

# Distribuzione dei livelli trofici
bp = barplot(frac,
             col = "lightblue",
             main = "Distribuzione delle specie nei livelli trofici",
             xlab = "Livello trofico",
             ylab = "Frequenza relativa",
             border = "white",
             ylim = c(0, max(frac) * 1.2)
)
text(bp, frac, labels = paste0(round(frac * 100, 1), "%"), pos = 3)
```



Notiamo che la maggior parte delle specie in questa food web sono di livello trofico 2 o 3, con una maggioranza significativa nel livello 3. Questo significa che la maggior parte delle specie sono onnivore o carnivore.

Grazie al livello trofico possiamo visualizzare la food web in maniera più ordinata.

```
coords = layout_igraph_constrained_stress(
  fw_g,
  coord = V(fw_g)$trophic_lvl,
  fixdim = "y"
)

ggraph(fw_g, layout = "manual", x = coords$x, y = coords$y) +
  geom_edge_fan(aes(alpha = carbon_flow), edge_colour = "grey", linewidth = 0.25) +
  geom_node_point(aes(color = factor(trophic_lvl_int)), size = 2.3, alpha=0.5) +
  scale_color_manual(values = get_tl_colors(), name = "Trophic Level") +
  scale_y_continuous(name = "Fractional Trophic Level") +
  theme_minimal() +
  theme(
    axis.title.x = element_blank(),
    axis.text.x  = element_blank(),
    axis.ticks.x = element_blank(),
    panel.grid.major.x = element_blank(),
    panel.grid.minor.x = element_blank()
  )
```



Domande di ricerca

1. Come si distribuisce la biomassa tra i livelli trofici? Risposta 1
2. Qual'è l'efficienza di trasferimento energetico (Ecological Efficency) tra i livelli trofici? (10% law) Risposta 2
3. Quali sono le specie con più potere all'interno della food web? Risposta 3
4. Quali sono le specie apicali e basali? Risposta 4
5. Quali sono le specie centrali? Risposta 5
6. Studiando la connettività della rete, questa risulta resiliente? Risposta 6
7. Analizzando le comunità della rete, quali sotto food web emergono? C'è varietà trofica all'interno delle comunità? Risposta 7

8. Qual’è la lunghezza massima dei cammini minimi? Si presenta lo small-world effect all’interno della rete?
   Risposta 8

Biomass within trophic levels

Iniziamo analizzando come si distribuisce la biomassa nei livelli trofici. Consideriamo i detriti come livello trofico 0, in modo da poterne vedere la relazione con gli altri livelli.

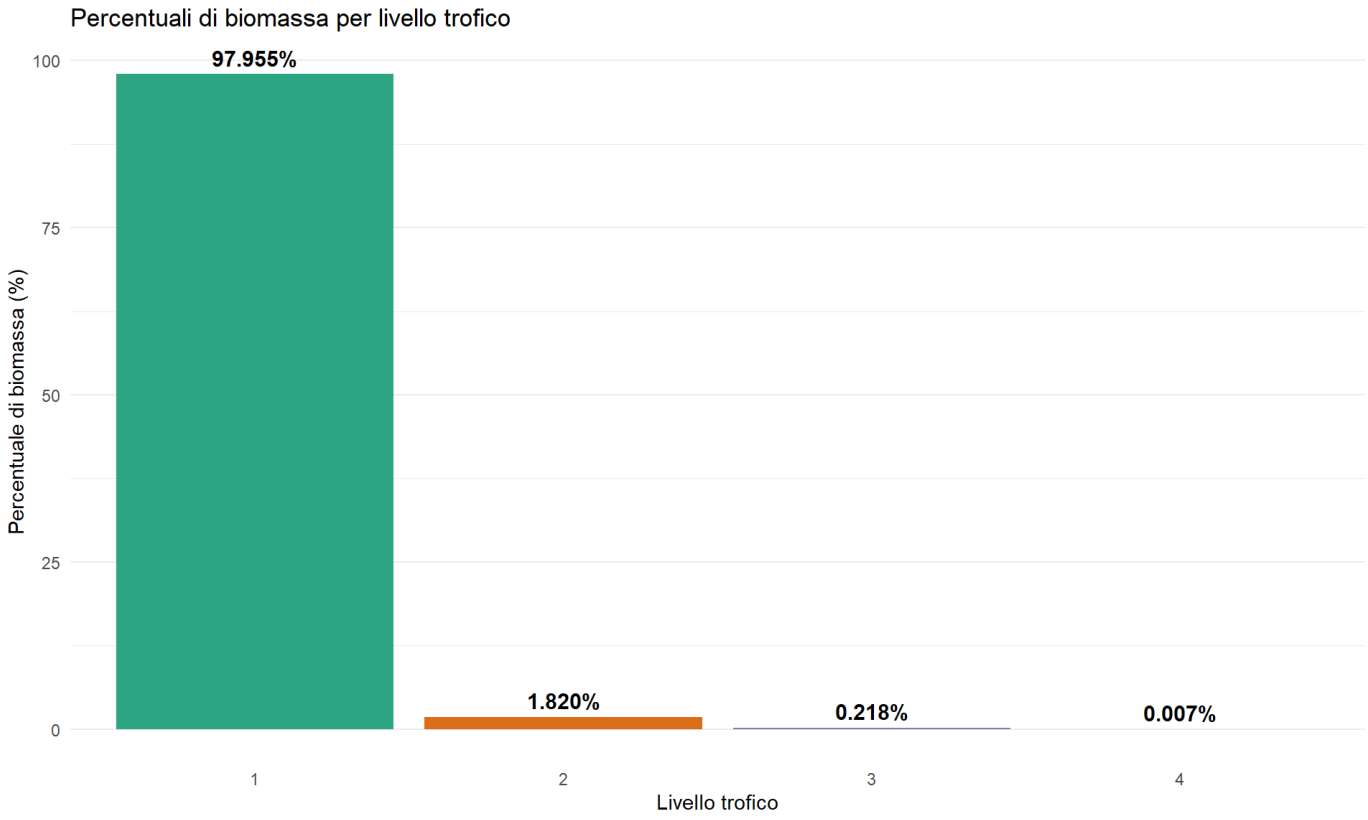
```
biomass_tl = fw_g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  mutate(
    trophic_lvl = ifelse(eco_type == 2, 0.0, trophic_lvl),       # Indichiamo con 0 i detriti
    trophic_lvl_int = ifelse(eco_type == 2, 0, trophic_lvl_int), # Indichiamo con 0 i detriti
  ) %>%
  as_tibble() %>%
  group_by(trophic_lvl_int) %>%
  summarise(tot_biomass = sum(bio_mass), .groups = "drop") %>%
  mutate(perc = tot_biomass / sum(tot_biomass) * 100)
biomass_tl
```

trophic_lvl_int	tot_biomass	perc
0	641.6031650	81.2920715
1	144.6345133	18.3254071
2	2.6868898	0.3404329
3	0.3212039	0.0406970
4	0.0109818	0.0013914

Vediamo che la maggior parte della biomassa in questo ecosistema è contenuta nei detriti.

Analizziamo la biomassa “viva” contenuta solo nei livelli trofici da 1 a 4.

```
df = fw_g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  filter(eco_type == 1) %>%
  as_tibble() %>%
  group_by(trophic_lvl_int) %>%
  summarise(tot_biomass = sum(bio_mass)) %>%
  mutate(perc = tot_biomass / sum(tot_biomass) * 100)


ggplot(df, aes(x = factor(trophic_lvl_int), y = perc, fill = factor(trophic_lvl_int))) +
  geom_col(alpha = 0.9) +
  geom_text(aes(label = sprintf("%.3f%%", perc)),
            vjust = -0.5, fontface = "bold", size = 4) +
  scale_fill_manual(values = get_tl_colors()) +
  labs(
    title = "Percentuali di biomassa per livello trofico",
    x = "Livello trofico",
    y = "Percentuale di biomassa (%)"
  ) +
  theme_minimal() +
  theme(
    legend.position = "none",
    panel.grid.major.x = element_blank()
  )
```

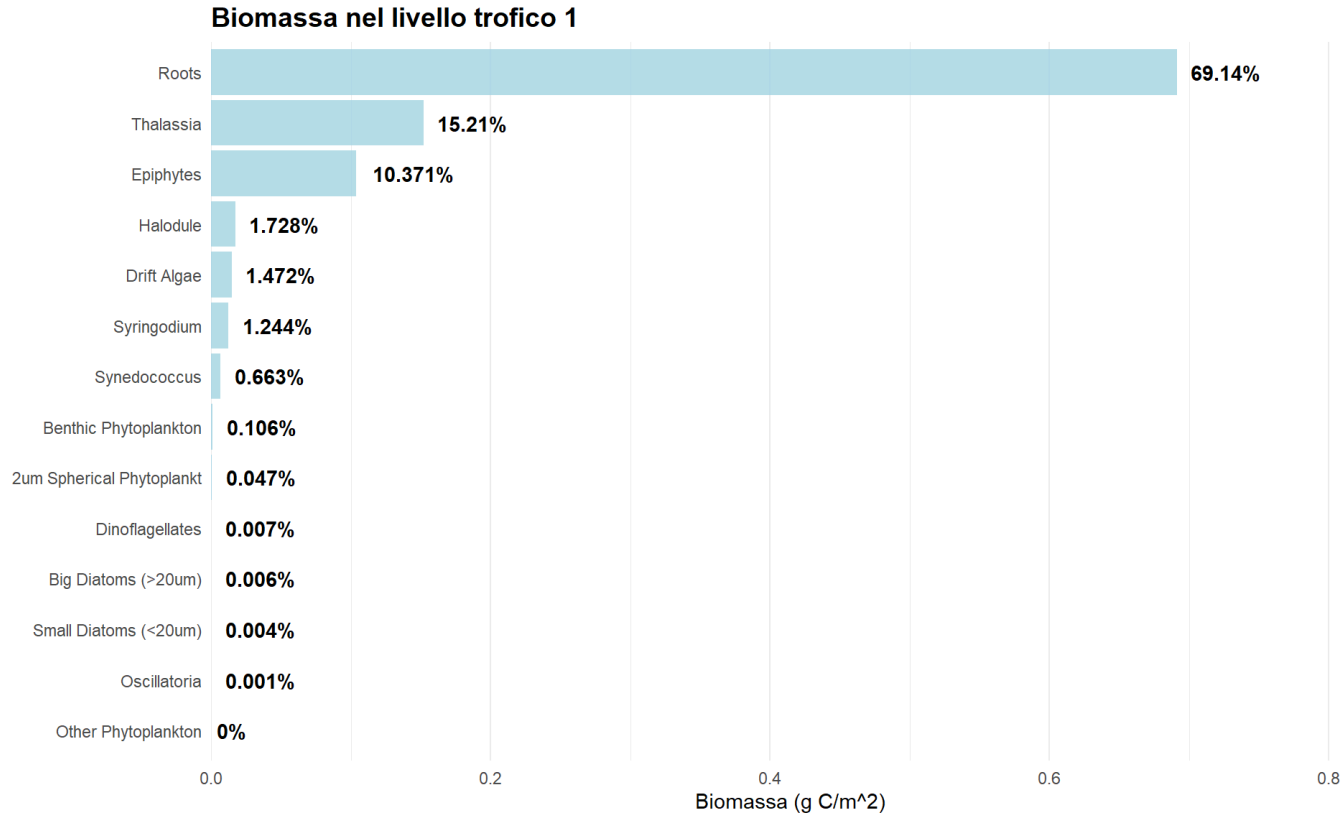


La maggior parte della biomassa “viva” è contenuta nel primo livello trofico per un 97.96%, mentre tutti gli altri livelli insieme ne contengono poco più del 2%.

Inoltre si nota come la percentuale di biomassa cala all’aumentare del livello trofico. Questo rispecchia il fatto che specie di livello basso necessitano di meno energia per sopravvivere ed è quindi più facile per loro proliferare.

Data la dominanza del livello trofico 1, è utile identificare quali produttori primari sono più presenti all'interno di questo livello.

```
# Recuperiamo la biomassa totale nel livello 1
tot_lv1_biomass = filter(biomass_tl, trophic_lvl_int == 1)$tot_biomass

# Preparazione dati
top_biomass = fw_g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  filter(eco_type == 1 & trophic_lvl_int == 1) %>%
  mutate(bio_mass = bio_mass / tot_lv1_biomass) %>%
  select(name, bio_mass) %>%
  as_tibble()

# Usiamo reorder per riordinare le barre nella visualizzazione
ggplot(top_biomass, aes(x = reorder(name, bio_mass), y = bio_mass)) +
  geom_col(fill = "lightblue", alpha = 0.9) +
  geom_text(
    aes(label = paste0(round(bio_mass, 5)*100, "%")),
    hjust = -0.2,
    fontface = "bold"
  ) +
  coord_flip() +
  scale_y_continuous(expand = expansion(mult = c(0, 0.2))) +
  labs(
    title = "Biomassa nel livello trofico 1",
    x = NULL,
    y = "Biomassa (g C/m^2)"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    panel.grid.major.y = element_blank()
  )
```



Ecological efficency

Viene definita **ecological efficiency** l'efficienza con cui l'energia è trasferita da un livello trofico a quello successivo.

Empiricamente solo circa il 10% dell'energia trasferita da un livello trofico viene effettivamente immagazzinata come biomassa (tessuto organico) nel livello trofico successivo. Il restante 90% dell'energia viene dissipato attraverso tre meccanismi principali:

- Respirazione cellulare: gli organismi utilizzano gran parte dell'energia per i processi metabolici, producendo calore che viene disperso nell'ambiente
- Digestione incompleta: non tutto il materiale organico ingerito viene assimilato; una parte viene espulsa come rifiuto non digerito
- Perdite durante il trasferimento: energia spesa nella ricerca del cibo, nella crescita, nella riproduzione e in altre attività vitali

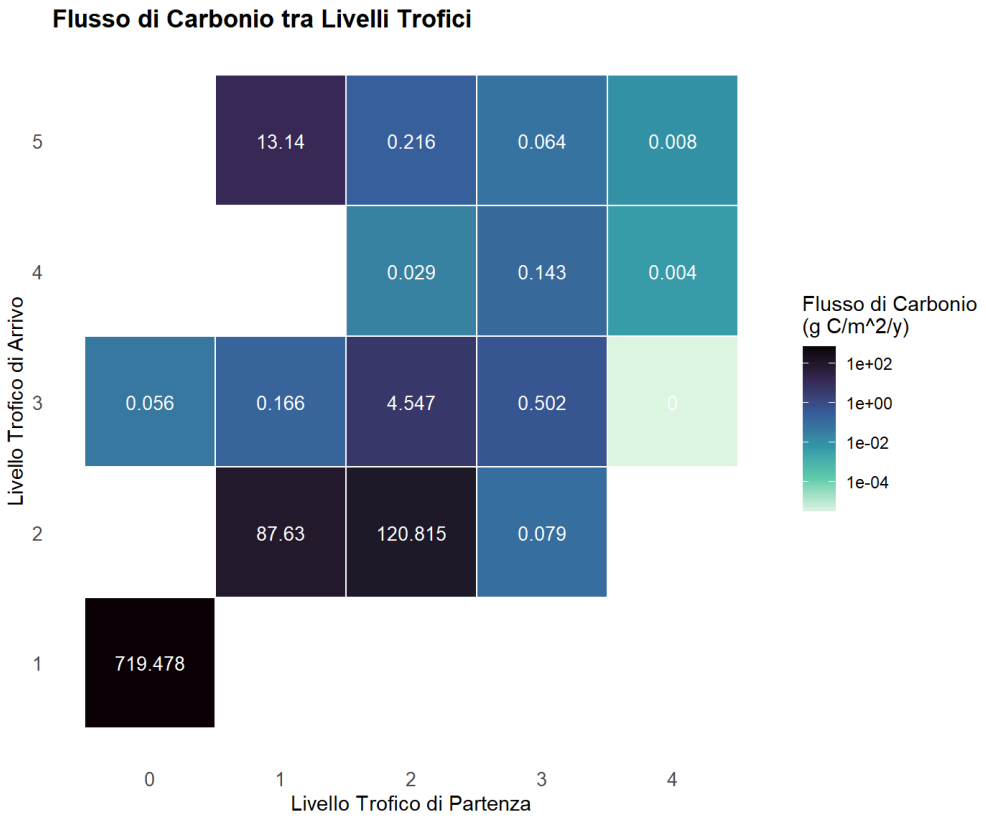
Analizziamo l'efficienza di questa rete trofica raggruppando l'energia sugli archi delle specie, da un livello a quello successivo. Si noti Per calcolare l'efficienza tra ogni livello successivo, sarà sufficienza calcolare:

$$\text{Ecological\_efficency}_i = \frac{Output_{i+1}}{Input_i}$$

Dove:

- $Input_i$: indica il totale dell'energia ricevuto al livello trofico i
- $Output_{i+1}$: indica il totale dell'energia trasmessa al livello trofico $i+1$

```
# Rimuoviamo tutto ciò che non è vivo
g = fw_g_io %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  filter(!name %in% c("Respiration", "Water POC", "Benthic POC", "DOC"))

# Costruiamo il dataframe contenente la quantità di carbonio scambiato da un
# livello trofico ad un altro
energy_flow = g %>%
  as_tbl_graph() %>%
  activate(edges) %>%
  mutate(
    from_tl = .N()$trophic_lvl_int[from],
    to_tl = .N()$trophic_lvl_int[to]
  ) %>%
  as_tibble() %>%
  group_by(from_tl, to_tl) %>%
  summarise(carbon_flow = sum(carbon_flow), .groups = "drop")

# Heatmap del flusso di carbonio tra livelli trofici
ggplot(energy_flow, aes(x = from_tl, y = to_tl, fill = carbon_flow)) +
  geom_tile(color = "white", linewidth = 0.5) +
  geom_text(aes(label = round(carbon_flow, 3)), color = "white", size = 3.5) +
  scale_fill_viridis_c(
    option = "mako",
    trans = "log10",
    direction = -1, # Inverte la scala
    name = "Flusso di Carbonio\n(g C/m^2/y)"
  ) +
  labs(
    title = "Flusso di Carbonio tra Livelli Trofici",
    x = "Livello Trofico di Partenza",
    y = "Livello Trofico di Arrivo"
  ) +
  theme_minimal() +
  theme(
    panel.grid = element_blank(),
    plot.title = element_text(face = "bold"),
    axis.text = element_text(size = 10)
  ) +
  coord_fixed()
```



Notiamo subito che la quantità di energia trasferita cala di circa un ordine di grandezza ad ogni livello trofico (antidiagonale). E' inoltre interessante notare che all'interno del secondo livello trofico c'è un grande scambio intra-livello di energia.

Calcoliamo l'efficienza per ogni livello trofico successivo.

```
# Teniamo solo i flussi di energia tra livelli successivi
energy_flow = filter(energy_flow, to_tl == from_tl + 1)

# Per ogni coppia calcoliamo l'efficienza
eff = vector()
for (i in 2:nrow(energy_flow)) {
  inp_flow = energy_flow[i-1,]$carbon_flow
  out_flow = energy_flow[i,]$carbon_flow
  eff[i-1] = out_flow / inp_flow   # Il ciclo parte da indice 2, quindi [i-1]
}

eco_eff = tibble(
  "Trophic Level" = 1:4,
  "Ecological efficency %" = eff * 100
)
eco_eff
```

Trophic Level	Ecological efficency %
1	12.179627
2	5.188328
3	3.135331
4	5.769956

```
round(mean(eff) * 100, 2)
```

```
## [1] 6.57
```

Risulta così che questa rete ha un'efficienza media del 6.57%.

Powerfull species in energy transfer

Essendo il potere definito come $x_i = \sum_k \frac{a_{k,i}}{x_k}$, in questo contesto considerando la matrice $\text{preda} \times \text{predatore}$, una specie è tanto più potente quanto tanta più assorbe energia da specie non potenti.

```
# create incidence matrix for a graph
make_incidence = function(g) {
  n = vcount(g)
  m = ecount(g)
  # get edges as a matrix
  E = ends(g, E(g), names = FALSE)
  B = matrix(0, nrow = n, ncol = m)
  # build incidence matrix
  for (i in 1:m) {
    B[E[i,1], i] = 1
    B[E[i,2], i] = 1
  }
  return(B)
}

regularify = function (g) {
  n = vcount(g)
  m = ecount(g)
  B = make_incidence(g)
  # objective function
  obj = rep(0, m + 1)
  # constraint matrix
  con = cbind(B, rep(-1, n))
  # direction of constraints
  dir = rep("=", n)
  # right hand side terms
  rhs = -degree(g)
  # solve the LP problem
  sol = lp("max", obj, con, dir, rhs)
  # get solution
  if (sol$status == 0) {
    s = sol$solution
    # weights
    w = s[1:m] + 1
    # weighted degree
    d = s[m+1]
  }
  # return the solution
  if (sol$status == 0) {
    return(list(weights = w, degree = d))
  }
  else {
    return(NULL)
  }
}

# Compute power x = (1/x) A
#INPUT
# A = graph adjacency matrix
# t = precision
# OUTPUT
# A list with:
# vector = power vector
# iter = number of iterations
power = function(A, t) {
```

```
  n = dim(A)[1];
  # x_2k
  x0 = rep(0, n);
  # x_2k+1
  x1 = rep(1, n);
  # x_2k+2
  x2 = rep(1, n);
  diff = 1
  eps = 1/10^t;
  iter = 0;
  while (diff > eps) {
    x0 = x1;
    x1 = x2;
    x2 = (1/x2) %*% A;
    diff = sum(abs(x2 - x0));
    iter = iter + 1;
  }
  # it holds now: alpha x2 = (1/x2) A
  alpha = ((1/x2) %*% A[,1]) / x2[1];
  # hence sqrt(alpha) * x2 = (1/(sqrt(alpha) * x2)) A
  x2 = sqrt(alpha) %*% x2;
  return(list(vector = as.vector(x2), iter = iter))
}
```

Verifichiamo se la matrice è regolarizzabile

```
g = fw_g_io %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  # filter(eco_type == 1 | eco_type == 2) %>%
  filter(eco_type == 1) %>%
  as.igraph()

regularify(g)
```

```
## NULL
```

Non essendo regolarizzabile perturbiamo la matrice sulla diagonale,

```
B = as_adjacency_matrix(g, sparse = FALSE, attr="carbon_flow")
I = diag(1, vcount(g))
BI = B + I

V(g)$power = power(BI, 6)$vector

df = g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  select(name, trophic_lvl, trophic_lvl_int, power) %>%
  arrange(-power) %>%
  as_tibble()

df %>%
  select(name, trophic_lvl_int, power) %>%
  filter(power >= quantile(power, 0.95))
```

name	trophic_lvl_int	power
Water Flagellates	2	61.714839
Other Copepoda	2	19.829540
Oithona nana	2	15.220996
Meiofauna	2	12.656483
Other Zooplankton	2	10.633314
Water Cilitaes	2	5.177599
Benthic Ciliates	2	3.456621

Risulta che le specie con maggior potere sono:

- Water Flagellates
- Other Copepoda
- Oithona nana
- Meiofauna
- Other Zooplankton
- Water Cilitaes
- Benthic Ciliates

Tutte specie di livello trofico 2.

Vediamo ora come si distribuisce il potere tra i livelli trofici.

```
df %>%
  group_by(trophic_lvl_int) %>%
  summarise(avg_power = mean(power))
```

trophic_lvl_int	avg_power
1	1.000000
2	4.584604
3	1.025880

trophic_lvl_int	avg_power
4	1.004044

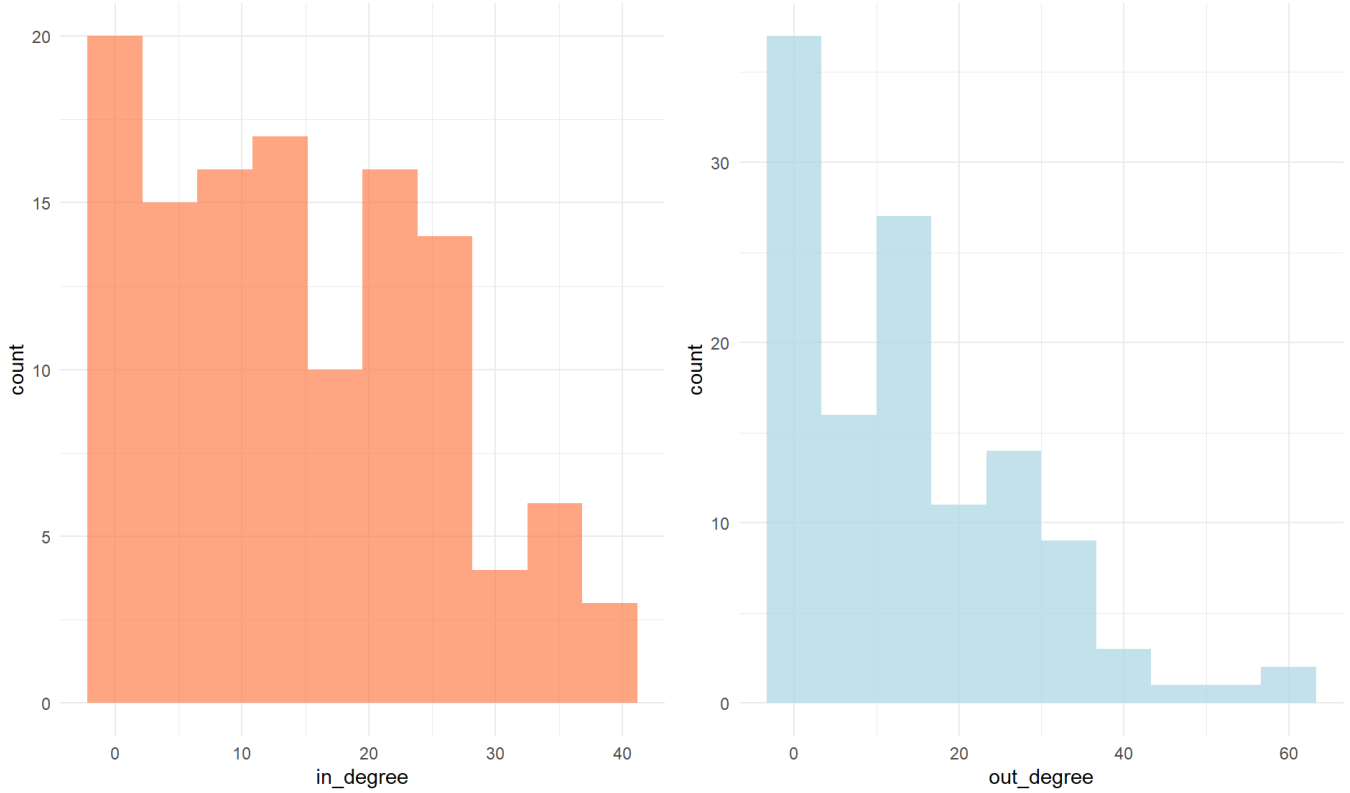
Notiamo che la maggior parte del potere è concentrato nel livello trofico 2, il quale funge da importante intermediario con gli altri livelli. Questo livello “accumula” molta energia grazie anche alle considerazioni della 10%-law. Se avessimo considerato anche l’input e l’output avremmo notato che le specie più potenti sarebbero state quelle basali (livello trofico 1), in quanto ricevono grandi quantità di energia naturale, rispetto a quanta ne ricevono i livelli successivi.

Apex & Basal Species

Ci concentriamo ora a studiare i gradi della food web. Ricordiamo che:

- in-degree: indica il numero di specie che preda
- out-degree: indica il numero di specie da cui è predata

```
deg_df = fw_g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  filter(eco_type == 1) %>%
  filter(!node_is_isolated()) %>%
  mutate(
    in_degree = centrality_degree(mode = "in"),
    out_degree = centrality_degree(mode = "out"),
    total_degree = centrality_degree(mode = "all"),
  ) %>%
  mutate(
    ratio_deg = ifelse(out_degree == 0, Inf, in_degree / out_degree) # Rapporto tra numero di
predatori e numero di prede
  ) %>%
  as_tibble()

# Distribuzione in-degree
in_deg_plot = ggplot(deg_df) +
  geom_histogram(aes(in_degree), bins=10, fill="coral", alpha=0.7) +
  theme_minimal()

# Distribuzione out-degree
out_deg_plot = ggplot(deg_df) +
  geom_histogram(aes(out_degree), bins=10, fill="lightblue", alpha=0.7) +
  theme_minimal()

in_deg_plot + out_deg_plot
```



La distribuzione dei gradi entranti risulta leggermente in una distribuzione a coda lunga, molto più marcata nei gradi uscenti. Possiamo dire che tendenzialmente:

- i predatori hanno probabilità simili di avere qualisasi numero di prede;
- molte prede sono consumate da pochi predatori, mentre poche prede sono consumate da molti.

Già solo con i gradi entranti e uscenti siamo in grado di estarre quali specie sono in cima alla catena alimentare e quali invece sono alla base. Possiamo considerare il rapporto tra i gradi entranti e uscenti come prima metrica per identificare:

- **specie basali**: produttori primari che vengono solo predati
- **specie apicali** predano altri e non sono quasi mai predati

```
# specie apicali
deg_df %>%
  arrange(-ratio_deg) %>%
  select(
    name,
    in_degree, out_degree, # ratio_deg,
    trophic_lvl
  ) %>%
  head(10)
```

name	in_degree	out_degree	trophic_lvl
Sharks	28	0	4.534386
Grouper	30	0	3.957014
Mackerel	10	0	4.542244
Loon	33	0	4.310491
Greeb	36	0	4.286215

name	in_degree	out_degree	trophic_lvl
Predatory Ducks	37	0	4.301262
Raptors	37	0	4.601776
Kingfisher	19	0	4.274697
Crocodiles	39	0	4.363022
Hawksbill Turtle	15	0	3.236525

Come ci si aspettava, le specie apicali sono di livello trofico 4 o 3. Nelle prime posizioni abbiamo tutte specie che non vengono predate da nessun altro.

Per quanto riguarda le specie basali, analogamente a sopra, avremo le specie di livello trofico 1 e 2. Filtriamo quindi le specie di livello trofico 1 che non predano alcuna specie. E estraiamo le specie basali di livello 2 o superiore.

```
# specie basali
# nota: vengono rimosse le specie che sono solo produttori, quindi livello trofico uguale a 1 (come ad esempio le piante o phytoplankton)
deg_df %>%
  arrange(ratio_deg) %>%
  filter(trophic_lvl_int != 1) %>%
  select(
    name,
    out_degree, in_degree, # ratio_deg,
    trophic_lvl
  ) %>%
  head(10)
```

name	out_degree	in_degree	trophic_lvl
Free Bacteria	9	0	2.000000
Herbivorous Shrimp	60	2	2.000000
Thor Floridanus	26	1	2.000000
Epiphytic Gastropods	23	1	2.000000
Benthic Flagellates	10	1	2.000000
Detritivorous Crabs	30	3	2.331722
Detritivorous Amphipods	36	4	2.491564
Detritivorous Polychaetes	34	4	2.323848
Benthic Crustaceans	33	4	2.513566
Detritivorous Gastropods	37	5	2.024430

Notiamo che tra le prime 10 specie basali abbiamo:

- detritivori (Detritivorous)
- specie che vivono a stretto contatto con il fondale (Benthic)

Central Species

Ci concentriamo ora a studiare quali specie risultano maggiormente centrali all'interno dell'ecosistema. Per fare questo **invertiamo la direzione degli archi**, in quanto siamo interessati a quali specie sono vitali nella dieta di altri specie. Consideriamo il grafo non pesato per questa analisi.

> In questa analisi non verranno considerati i detriti, considerando solo le specie "vive". Rimuovendo POC e DOC, un solo nodo "Roots" si disconnette dal grafo e questo verrà rimosso con `filter(!node_is_isolated())`.

In questa analisi useremo le seguenti misure di centralità:

- **betweeness**: indica quali specie sono le più ricorrenti nei cammini minimi;
- **harmonic**: (a differenza della closeness $\frac{1}{d(u,v)} = 0$ se non esiste un cammino tra u e v). Indica quali specie sono le più vicine alle altre
- **page rank**: misura l'importanza di una specie basandosi sull'importanza delle specie che la consumano/utilizzano

Trophic level analysis

Iniziamo osservando come si comportano le misure di centralità tra i livelli trofici.

```
g = fw_g %>%
  reverse_edges() %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  filter(!name %in% c("Benthic POC", "Water POC", "DOC")) %>%
  filter(!node_is_isolated()) %>%
  mutate(
    harm = centrality_harmonic(mode = "in", normalized = TRUE),
    betw = centrality_betweenness(directed = TRUE, normalized = TRUE),
    prank = centrality_pagerank()
  ) %>%
  as.igraph()

g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  as_tibble() %>%
  select(name, trophic_lvl, trophic_lvl_int, betw, harm, prank) %>%
  group_by(trophic_lvl_int) %>%
  summarise(
    'AVG Betweeness' = mean(betw),
    'AVG Harmonic' = mean(harm),
    'AVG Page Rank' = mean(prank)
  ) %>%
  rename('Trophic LV.' = trophic_lvl_int)
```

Trophic LV.	AVG Betweeness	AVG Harmonic	AVG Page Rank
1	0.0000000	0.3988462	0.0202433
2	0.0035519	0.3443393	0.0108107

Trophic LV.	AVG Betweeness	AVG Harmonic	AVG Page Rank
3	0.0043417	0.1865251	0.0052677
4	0.0003094	0.0187500	0.0034093

La **betweeness**, come potevamo aspettarci, è particolarmente bassa nei livelli estremi (1 e 4) in quanto le specie in questi livelli saranno specialmente nodi di inizio o fine nella rete trofica.

Per quanto riguarda la **harmonic** centrality si nota che all'aumentare della livello trofico, questa cala.

```
cor(V(g)$harm, V(g)$trophic_lvl)
```

```
## [1] -0.6797157
```

Infatti, calcolando la correlazione tra harmonic centrality e livello trofico, risulta una correlazione negativa. Questo risultato è spiegato dal fatto che ogni specie ha, tendenzialmente, un cammino con i livelli più bassi. Salendo di livello meno specie hanno cammini con i livelli sottostanti pesando 0 nel calcolo della metrica.

Infine il **page rank**, ha un comportamento simile alla harmonic centrality, questo è dovuto al fatto che ogni specie deve avere una fonte primaria di energia che sono tendenzialmente i produttori (livello trofico 1).

```
cor(V(g)$prank, V(g)$trophic_lvl)
```

```
## [1] -0.5990347
```

Anche in questo caso risulta una leggera correlazione negativa tra page rank e livello trofico.

Single species analysis

Analizziamo la betweeness sulle singole specie.

Risulta che il 5% delle specie più presenti all'interno di cammini minimi sono le seguenti:

```
g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  select(name, trophic_lvl_int, betw) %>%
  filter(betw >= quantile(betw, 0.95)) %>%
  arrange(-betw) %>%
  as_tibble()
```

name	trophic_lvl_int	betw
Predatory Gastropods	3	0.0285971
Echinoderma	2	0.0225531
Predatory Polychaetes	3	0.0189589
Other Cnidaridae	3	0.0183825
Other Demersal Fishes	2	0.0183040
Predatory Shrimp	3	0.0162558
Eels	3	0.0161395

Analizziamo la harmonic centrality sulle singole specie.

```
g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  select(name, trophic_lvl, harm) %>%
  filter(harm >= quantile(harm, 0.95)) %>%
  arrange(-harm) %>%
  as_tibble()
```

name	trophic_lvl	harm
Herbivorous Shrimp	2.000000	0.5548611
Predatory Shrimp	3.307168	0.5506944
Pink Shrimp	2.390210	0.5087500
Synedococcus	1.000000	0.4986111
Bivalves	2.199143	0.4958333
Omnivorous Crabs	2.342303	0.4873611
Detritivorous Amphipods	2.491564	0.4708333

I risultati rispecchiano l'analisi aggrata, eccezzione fatta per i "Predatory Shrimp" di livello trofico 3. Questo risultato potrebbe essere dovuto al fatto che questa specie viene sfruttata da molti predatori e a sua volta si nutre di molte specie di livello inferiore.

Infine analizziamo il page rank per le singole specie.

```
g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  select(name, trophic_lvl_int, prank) %>%
  filter(prank >= quantile(prank, 0.95)) %>%
  arrange(-prank) %>%
  as_tibble()
```

name	trophic_lvl_int	prank
Benthic Phytoplankton	1	0.0705527
Epiphytes	1	0.0367717
Synedococcus	1	0.0334800
Benthic Flagellates	2	0.0309407
Benthic Ciliates	2	0.0217128
Herbivorous Shrimp	2	0.0171770
2um Spherical Phytoplankt	1	0.0170146

I risultati sono coerenti con l'analisi aggragata precedente ponendo come specie di maggior importanza specie basali.

Connectivity

Ci concentriamo ora a studiare la food web globalmente.

```
# connectance calcola la "connectance" di una rete trofica (food web)
# La connectance misura la proporzione di collegamenti trofici realizzati
# rispetto a tutti i possibili collegamenti tra le specie
#
# INPUT
# g: grafo della food web
# directed: se TRUE calcola la directed connectance (default),
#           se FALSE calcola l'interactive connectance
#
# OUTPUT
# connectance: valore di connectance (compreso tra 0 e 1)
#
# FORMULE
# Directed connectance: C = L / S^2
# Interactive connectance: C = 2L / [S(S-1)]
# dove L = numero di collegamenti, S = numero di specie
connectance = function(g, directed = TRUE) {
  # defined from https://www.pnas.org/doi/10.1073/pnas.192407699
  # from page 5 of https://indico.ictp.it/event/a08145/session/28/contribution/17/material/0/
1.pdf

  n_species = vcount(g)
  n_links = ecount(g)

  if (directed) {
    # Directed connectance
    connectance = n_links / (n_species^2)
  } else {
    # Interactive connectance (collegamenti non direzionali)
    connectance = (2 * n_links) / (n_species * (n_species - 1))
  }

  return(connectance)
}
```

Una prima misura per stabilità della rete contro la perdita di specie è la **connectance**, definita come:

$$C = \frac{L}{S^2}$$

dove:

- L: numero di link
- S: numero di specie nella rete

Questa misura compresa tra 0 e 1, rappresenta la frazione di tutti i possibili legami di alimentazione che sono effetivamente presenti rispetto a tutti quelli realizzabili.

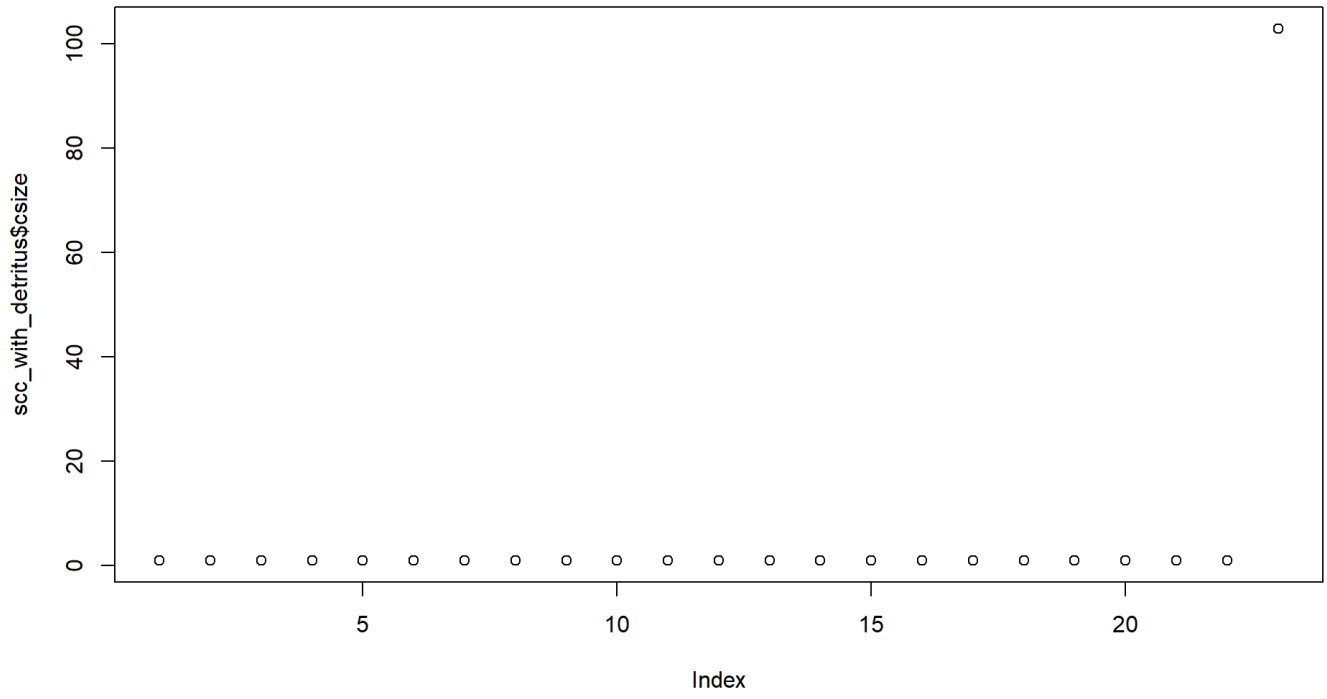
```
# g = fw_g %>%
#   as_tbl_graph() %>%
#   activate(nodes) %>%
#   filter(eco_type == 1) %>%
#   as.igraph()

round(connectance(fw_g), 2)
```

```
## [1] 0.12
```

La nostra rete risulta con una connectance di 0.12, sintomo di una rete che potrebbe risultare poco robusta.

Analizziamo ora le componenti fortemente connesse all'interno della rete. Iniziamo mantenendo anche i detriti.

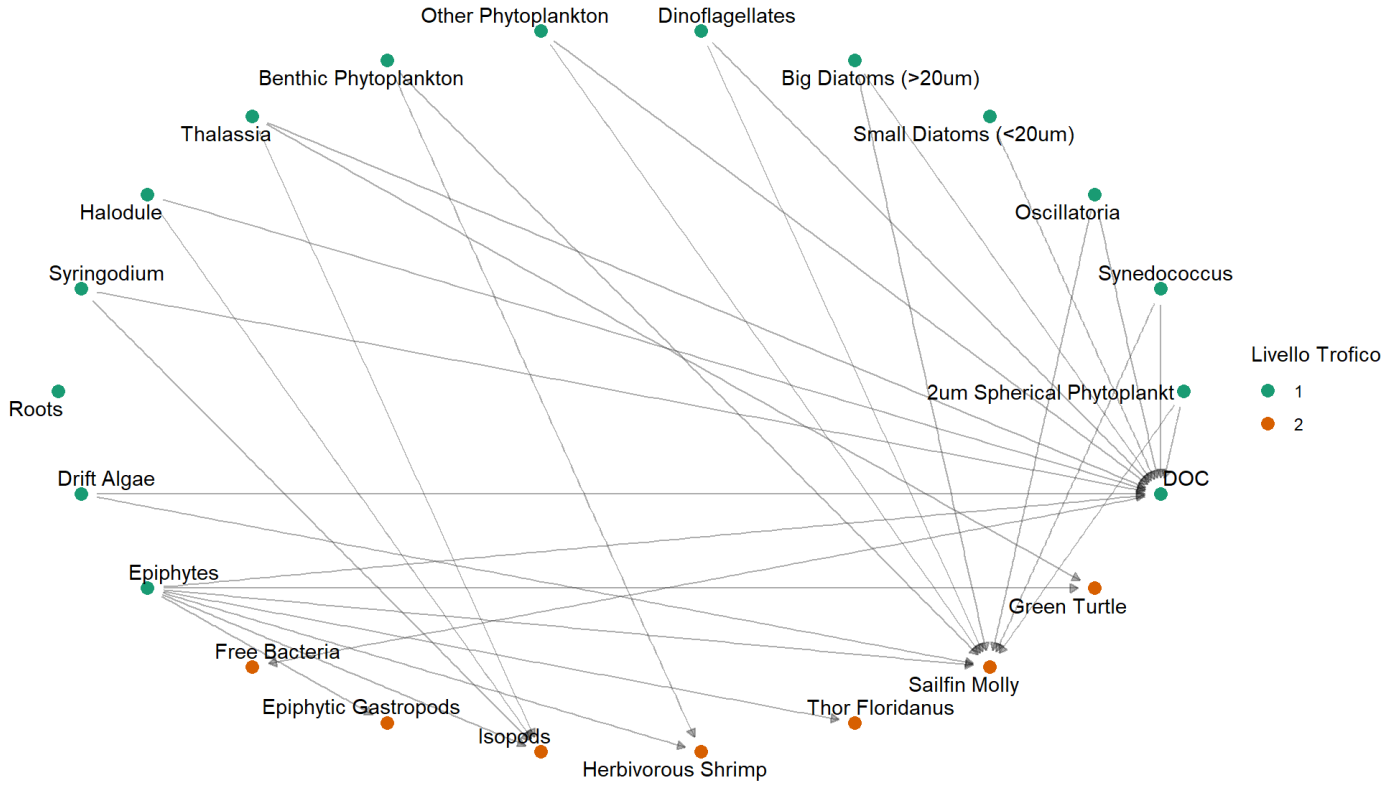
```
scc_with_detritus = components(fw_g, mode = "strong")

plot(scc_with_detritus$csize)
```



Mantenendo i detriti troviamo una SCC che comprende la maggior parte dei nodi. Visualizziamo quali nodi rimangono fuori dalla SCC.

```
largest_scc_id = which.max(scc_with_detritus$csize)
nodes_outside_scc = which(scc_with_detritus$membership != largest_scc_id)
nodes_outside_names = names(nodes_outside_scc)

fw_g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  filter(name %in% nodes_outside_names) %>%
  ggraph(layout = 'circle') +
  geom_edge_link(alpha = 0.3,
                 arrow = arrow(length = unit(1.7, 'mm'), type = "closed"),
                 start_cap = circle(3, 'mm'),
                 end_cap = circle(3, 'mm')) +
  geom_node_point(aes(color = factor(trophic_lvl_int)), size = 3) +
  geom_node_text(aes(label = name),
                 repel = TRUE) +
  scale_color_manual(values = get_tl_colors(),
                     name = "Livello Trofico") +
  labs(title = "Nodi fuori dalla SCC principale") +
  theme_void() +
  theme(
    plot.title = element_text(face = "bold"),
    legend.position = "right"
  )
```

Nodi fuori dalla SCC principale



Emerge che i produttori basali (livello trofico 1) sono fuori dalla SCC in quanto sono sorgenti unidirezionali che non ricevono flussi di riotrno, la loro fonte di energia è principalmente il Sole. E' però interessante notare che ci siano alcune specie di livello 2 che non fanno parte delle SCC.
Cosa caratterizza queste specie?

```
fw_g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  filter(name %in% nodes_outside_names & trophic_lvl_int > 1) %>%
  select(name, trophic_lvl, trophic_lvl_int) %>%
  as_tibble()
```

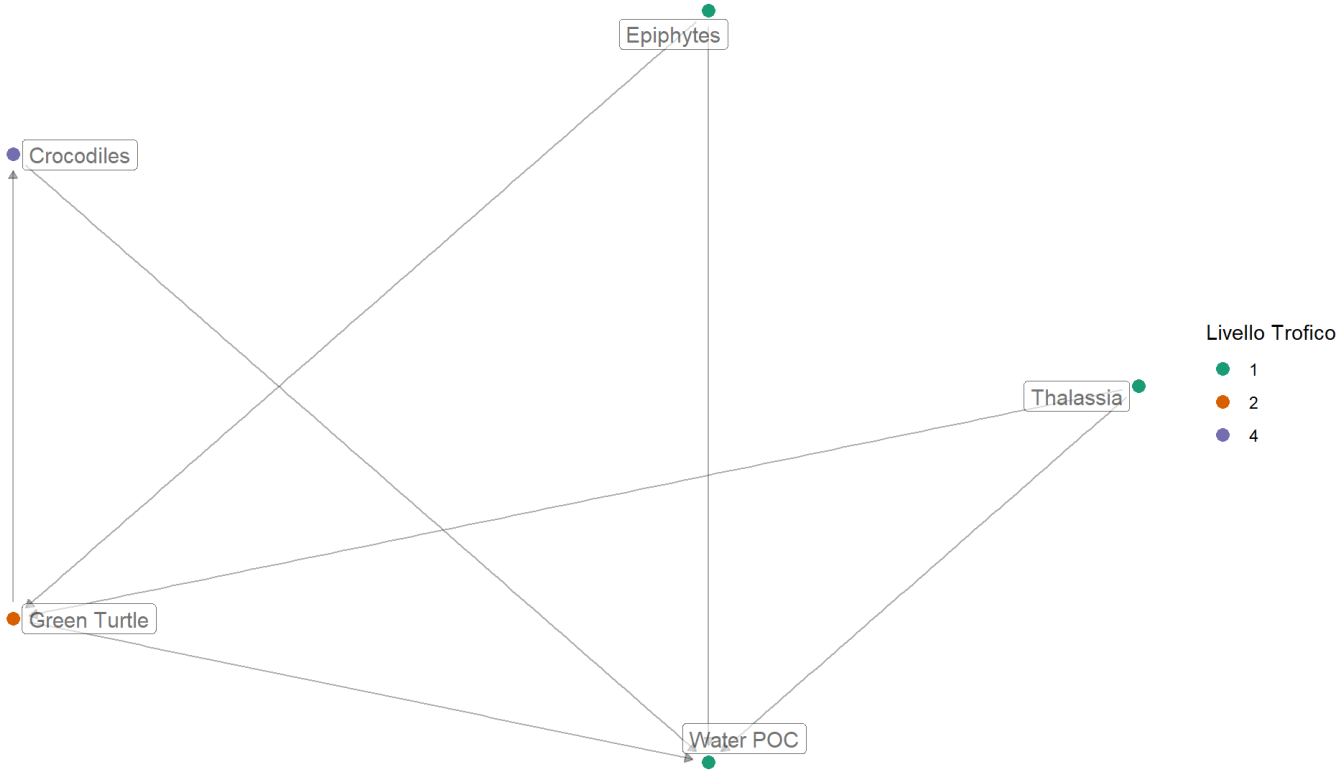
name	trophic_lvl	trophic_lvl_int
Free Bacteria	2	2
Epiphytic Gastropods	2	2
Isopods	2	2
Herbivorous Shrimp	2	2
Thor Floridanus	2	2
Sailfin Molly	2	2
Green Turtle	2	2

Queste specie sono puramente erbivore (livello trofico = 2) e ricavano energia solo dal livello precedente, non si reintroducono all'interno della SCC. La specializzazione degli erbivori, potrebbe indicare vulnerabilità all'estinzione. Tuttavia la grande disponibilità di biomassa al primo livello (come visto precedentemente 98%) garantisce un'abbondanza di risorse. La vulnerabilità di queste specie sarà quindi dovuta a:

- distruzione degli habitat
- cambiamenti qualitativi dell'ambiente
- disturbi antropici: pesca, inquinamento, cambiamento climatico

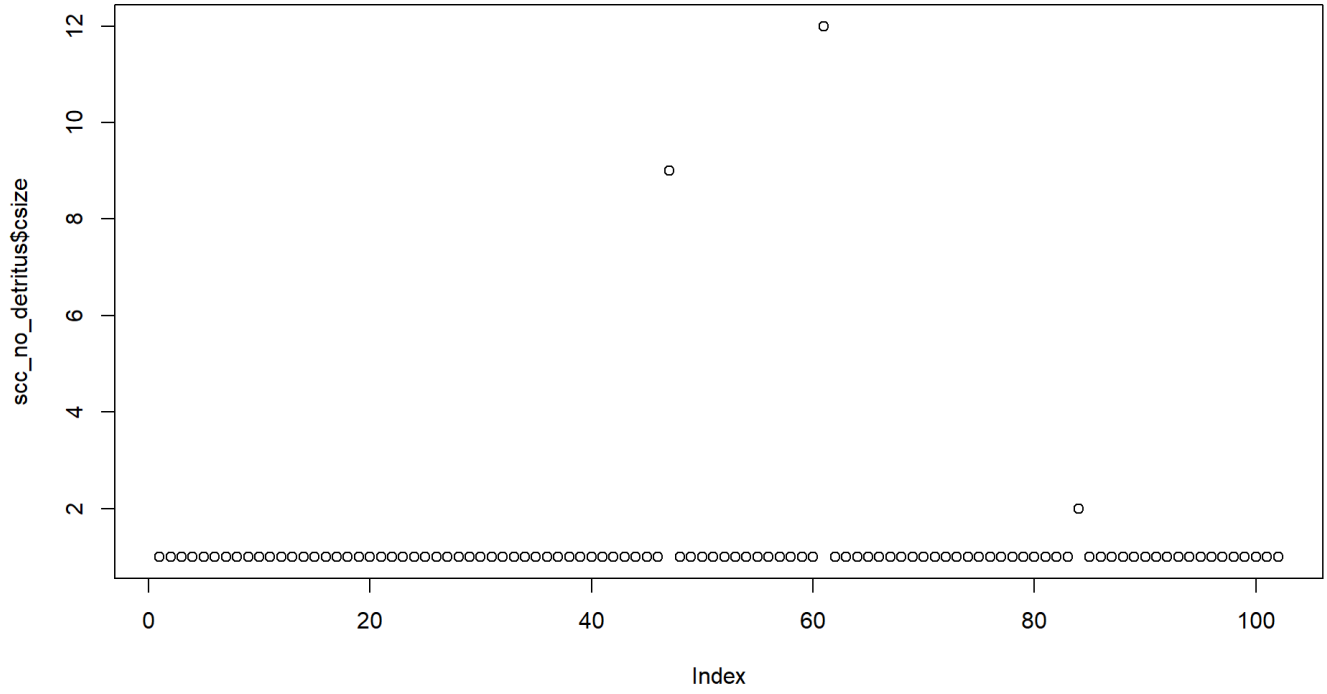
Visualizziamo nel dettaglio prede e predatori della Green Turtle.

```
green_turtle_id = which(V(fw_g)$name == "Green Turtle")
neighbors_all = neighbors(fw_g, green_turtle_id, mode = "all")
nodes_to_keep = c(green_turtle_id, neighbors_all)

induced_subgraph(fw_g, nodes_to_keep) %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  ggraph(layout = "circle") +
  geom_edge_fan(alpha = 0.3,
                arrow = arrow(length = unit(1.7, 'mm'), type = "closed"),
                start_cap = circle(3, 'mm'),
                end_cap = circle(3, 'mm')) +
  geom_node_point(aes(color = factor(trophic_lvl_int)), size = 3) +
  geom_node_label(
    aes(label = name),
    size = 4,
    repel = TRUE,
    alpha = 0.55
  ) +
  scale_color_manual(values = get_tl_colors(),
                     name = "Livello Trofico") +
  theme_void()
```



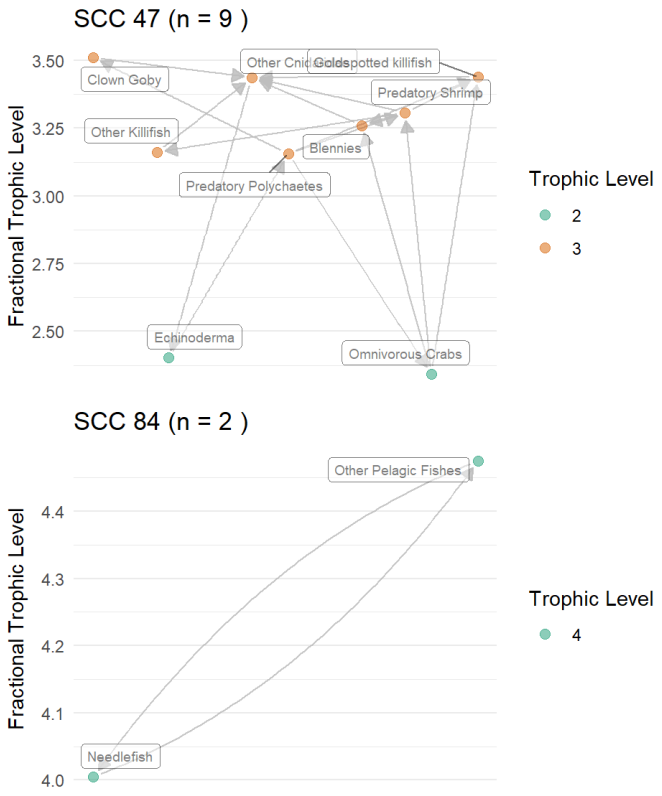
Recuperiamo le SCC rimuovendo i detriti.

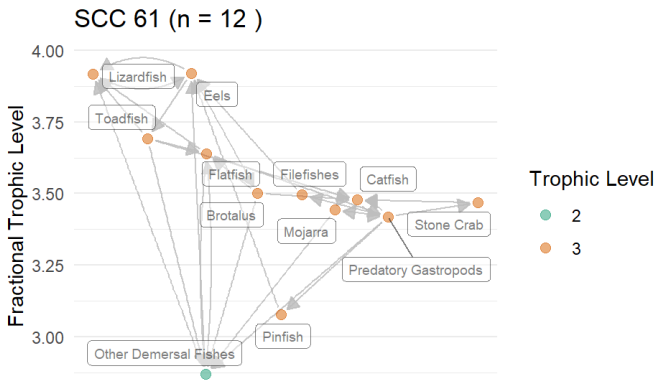
```
g = fw_g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  filter(eco_type == 1) %>%
  as.igraph()

scc_no_detritus = components(g, mode = "strong")

plot(scc_no_detritus$csize)
```



Si nota che rimuovendo i detriti la rete diventa sconnessa con piccoli cicli trofici isolati. Questo mostra quanto sia importante il ruolo dei detriti all'interno di un ecosistema, e ancora di più il ruolo di specie detrivore che reinseriscono in circolo la biomassa non consumata.

```
# Trovo le SCC con dimensione > 1
sccs_id = which(scc_no_detritus$csize != 1)

# Creiamo una lista di plot, uno per ogni SCC
plot_list = lapply(sccs_id, function(scc_idx) {
  # Recupera nodi di questa SCC
  species_in_scc = which(scc_no_detritus$membership == scc_idx)

  # Creo il sottografo per questa SCC
  fw_g_scc = induced_subgraph(fw_g, species_in_scc)

  # Calcolo il layout + plot
  coords = layout_igraph_constrained_stress(
    fw_g_scc,
    coord = V(fw_g_scc)$trophic_lvl,
    fixdim = "y"
  )

  plot = ggraph(fw_g_scc, layout = "manual", x = coords$x, y = coords$y) +
    geom_edge_fan(
      alpha = 0.8,
      edge_colour = "grey",
      arrow = arrow(length = unit(2.5, 'mm'), type = "closed"),
      start_cap = circle(1.5, 'mm'),
      end_cap = circle(1.5, 'mm')
    ) +
    geom_node_point(aes(color = factor(trophic_lvl_int)), size = 2.3, alpha=0.5) +
    geom_node_label(
      aes(label = name),
      size = 2.5,
      repel = TRUE,
      max.overlaps = 20,
      alpha = 0.5
    ) +
    scale_color_manual(values = get_tl_colors(), name = "Trophic Level") +
    scale_y_continuous(name = "Fractional Trophic Level") +
    ggtitle(paste("SCC", scc_idx, "(n =", length(species_in_scc), ")")) +
    theme_minimal() +
    theme(
      axis.title.x = element_blank(),
      axis.text.x  = element_blank(),
      axis.ticks.x = element_blank(),
      panel.grid.major.x = element_blank(),
      panel.grid.minor.x = element_blank()
    )
  return(plot)
})

# Combino tutti i plot affiancati
wrap_plots(plot_list, nrow = 2)
```





Communities

L'analisi delle comunità all'interno della food web rilevano **compartimenti**, ovvero sotto gruppi di specie che interagiscono fortemente tra di loro. Questi compartimenti possono essere visti come sotto food web. Teoricamente ogni compartimento risutla stabile rispetto agli altri, ovvero un'alterazione di un compartimento (es. estinzione di una specie) non dovrebbe influenzare significativamente gli altri.

Per eseguire questa analisi si procederà con la seguente pipeline:

1. Esecuzione di diversi metodi di calcolo delle comunità (Louvain, Edge Betweenness, Fast Greedy, Label Propagation, etc…).
2. Scelta di due algoritmi che producono maggiore modularità.
3. Analisi dei livelli trofici all'interno delle comunità.

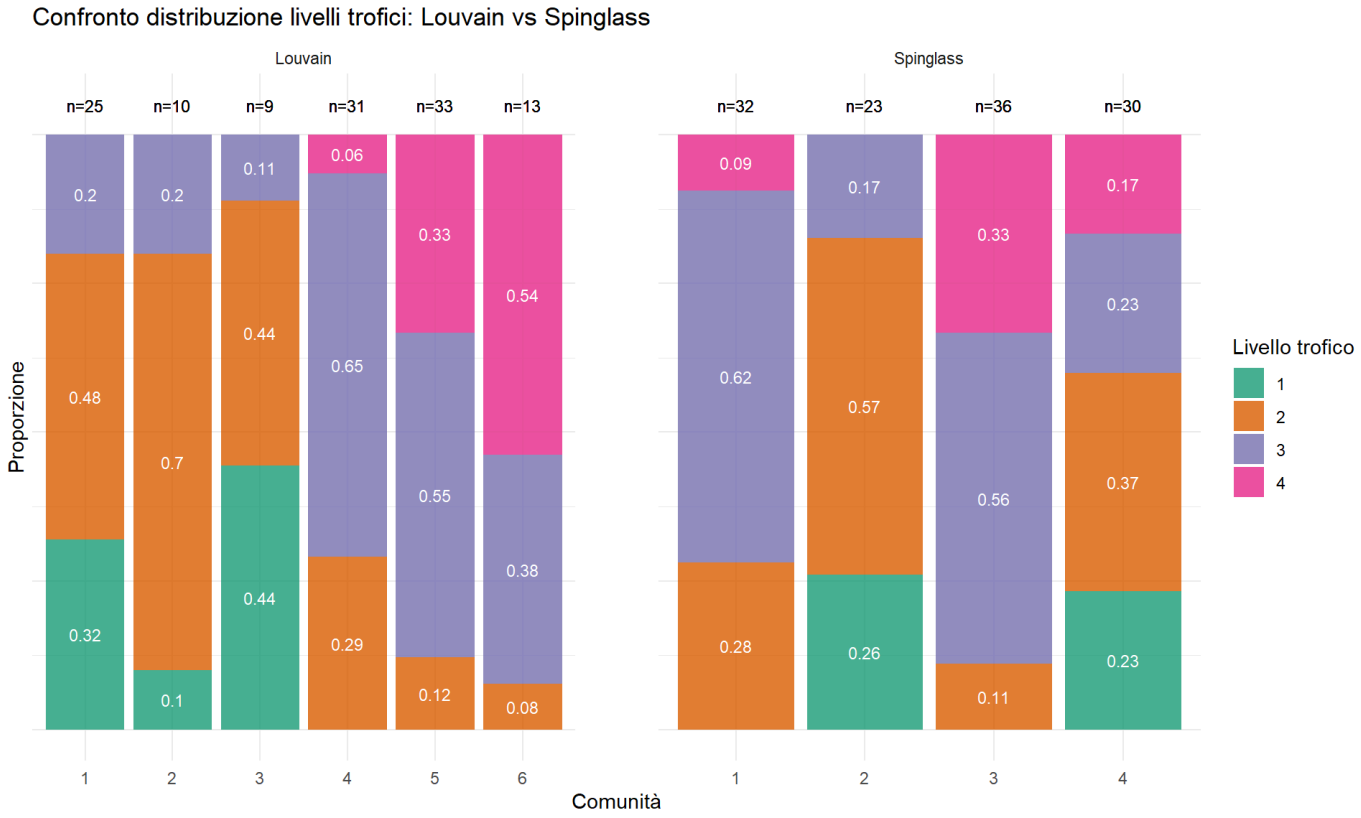
```
# Preparazione grafo
g = fw_g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  filter(eco_type == 1) %>%
  filter(!node_is_isolated()) %>%
  as.igraph() %>%
  as_undirected(mode = "collapse")

methods = list(
  "Louvain" = cluster_louvain,
  "Edge Betweenness" = cluster_edge_betweenness,
  "Fast Greedy" = cluster_fast_greedy,
  "Label Propagation" = cluster_label_prop,
  "Leading Eigenvector" = cluster_leading_eigen,
  "Walktrap" = cluster_walktrap,
  "Spinglass" = function(graph) cluster_spinglass(graph, spins = 10),
  "Infomap" = cluster_infomap
  # "Optimal" = cluster_optimal
)

# Compute modularity for each method
results = data.frame(
  Method = character(),
  Modularity = numeric(),
  Length = numeric(),
  stringsAsFactors = FALSE
)

for (method in names(methods)) {
  tryCatch({
    # Set seed for reproducibility
    set.seed(67)

    # Detect communities
    communities = methods[[method]](g)

    # Compute modularity
    modularity_value = modularity(communities)

    # Store the result
    results = rbind(results, data.frame(
      Method = method,
      Modularity = modularity_value,
      Length = length(communities)
    ))

    # Store result in graph
    g = set_vertex_attr(g, method, value = membership(communities))

  }, error = function(e) {
    # Handle any errors (e.g., if a method is not applicable)
    cat("Error with method:", method, "\n")
  })
}
```

```
# Sort results by modularity in decreasing order
results[order(-results$Modularity), ]
```

	Method	Modularity	Length
7	Spinglass	0.1947463	4
1	Louvain	0.1924205	6
3	Fast Greedy	0.1561395	4
5	Leading Eigenvector	0.1447770	3
6	Walktrap	0.1045574	2
2	Edge Betweenness	0.0456751	58
4	Label Propagation	0.0000000	1
8	Infomap	0.0000000	1

Dall’analisi delle comunità, risulta che i metodi “Spinglass” e “Louvain” producono maggiore modularità. Nonostante il numero di comunità sia differente la modularità è molto simile. Verifichiamo, ora come sono distribuiti i livelli trofici tra le comunità.

```
df = g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  select(trophic_lvl_int, Spinglass, Louvain) %>%
  as_tibble() %>%
  pivot_longer(
    cols = c(Spinglass, Louvain),
    names_to = "algorithm",
    values_to = "community"
  )

df %>%
  group_by(algorithm, community, trophic_lvl_int) %>%
  count() %>%
  group_by(algorithm, community) %>%
  mutate(
    perc = n / sum(n),
    size = sum(n)
  ) %>%
  ggplot(aes(fill = factor(trophic_lvl_int), x = factor(community), y = perc)) +
    geom_bar(position = position_stack(reverse = TRUE), stat = "identity", alpha = 0.8) +
    geom_text(aes(x = factor(community), y = 1.05, label = paste0("n=", size)), size = 3) +
    geom_text(aes(label = round(perc, 2)),
              position = position_stack(vjust = 0.5, reverse = TRUE),
              color = "white", size = 3) +
    facet_wrap(~algorithm, scale = "free_x") +
    labs(
      title = "Confronto distribuzione livelli trofici: Louvain vs Spinglass",
      x = "Comunità",
      y = "Proporzione",
      fill = "Livello trofico"
    ) +
    scale_fill_manual(values = get_tl_colors()) +
    theme_minimal() +
    theme(
      panel.spacing = unit(3, "lines"),
      axis.text.y = element_blank(),
      axis.ticks.y = element_blank()
    )
```



Nonostante i due metodi restituiscano un numero di comunità differenti, emrge che in entrambi i casi esistono delle comunità che al proprio interno non possiedono specie di livello trofico 1. Questa caratteristica potrebbe indicare dei compartimenti di specie carnivore/onnivore che fanno maggior affidamento sul secondo livello.

Small world

Verifichiamo ora se in questa rete è presente lo **small world effect**. Consideriamo il modello teorico che stima la distanza media come log2 del numero di nodi.

```
log2(vcount(fw_g))
```

```
## [1] 6.965784
```

```
mean_distance(fw_g)
```

```
## [1] 2.429463
```

La rete con all'interno i detriti presenta un forte small world effect, infatti il modello sovrastima la distanza media.

Proviamo a rimuovere i detriti per verificare se i risultati cambiano.

```
g = fw_g %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  filter(eco_type == 1)
log2(vcount(g))
```

```
## [1] 6.930737
```

```
mean_distance(g)
```

```
## [1] 1.96481
```

Rimuovendo i detriti la distanza media diminuisci, in quanto questi connettono diverse specie da vari livelli trofici aggiungendo un nodo intermedio in più. E' importante considerare che la lunghezza delle catene è una misura qualitativa della food web, catene più corte indicano maggior robustezza della rete trofica.

Infine calcoliamo il diametro di questa rete

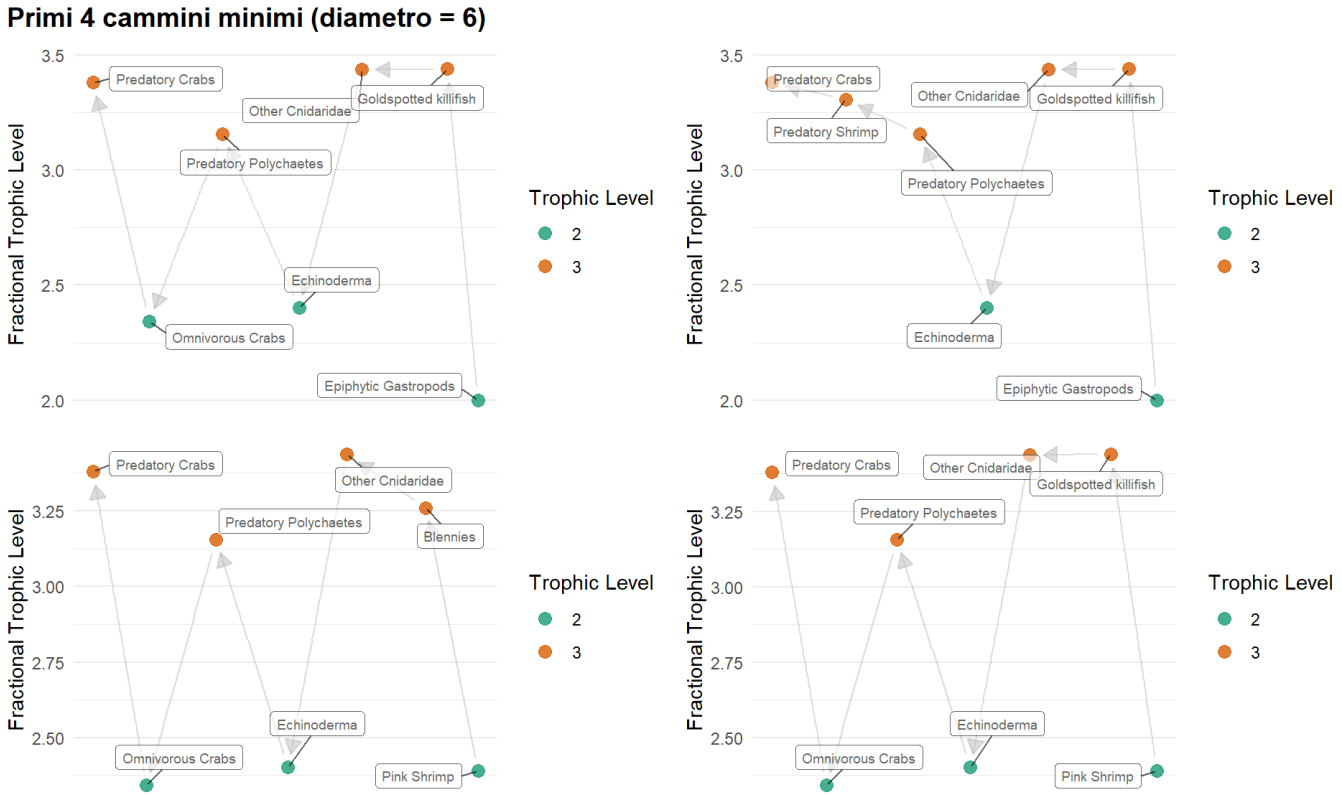
```
diameter(g)
```

```
## [1] 6
```

Visualizziamo i cammini minimi di lunghezza massima.

```
# Calcola la lunghezza del diametro
diam = diameter(g, directed = TRUE)
# Calcola la matrice delle distanze
dist_matrix = distances(g, mode = "out")
# Trova tutte le coppie di nodi con distanza = diametro
pairs = which(dist_matrix == diam, arr.ind = TRUE)

# Estrai tutti i cammini minimi
diameter_edges = apply(pairs, 1, function(pair) {
  all_shortest_paths(g, from = pair[1], to = pair[2], mode = "out")$epath
})
# Rendiamo i cammini una lista di liste
diameter_edges = unlist(diameter_edges, recursive = FALSE)


nrows = 2
ncols = 2
nplots = nrows * ncols
ndiams = length(diameter_edges)
diameters = diameter_edges[1:min(nplots, ndiams)]

plot_list = lapply(diameters, function(edges){

  # Crea subgrafo per questo cammino
  fw_g_path = subgraph_from_edges(g, eids = edges)

  coords_path = layout_igraph_constrained_stress(
    fw_g_path,
    coord = V(fw_g_path)$trophic_lvl,
    fixdim = "y"
  )

  p = ggraph(fw_g_path, layout = "manual", x = coords_path$x, y = coords_path$y) +
    geom_edge_fan(
      edge_colour = "grey",
      alpha = 0.5,
      arrow = arrow(length = unit(3, 'mm'), type = "closed"),
      end_cap = circle(2.5, 'mm'),
      start_cap = circle(2.5, 'mm')
    ) +
    geom_node_point(aes(color = factor(trophic_lvl_int)), size = 3, alpha = 0.8) +
    geom_node_label(
      aes(label = name),
      size = 2.5,
      repel = TRUE,
      alpha = 0.6,
      box.padding = 0.5
    ) +
    scale_color_manual(values = get_tl_colors(), name = "Trophic Level") +
    scale_y_continuous(name = "Fractional Trophic Level") +
    theme_minimal() +
    theme(
      axis.title.x = element_blank(),
      axis.text.x = element_blank(),
      axis.ticks.x = element_blank(),
```

```
      panel.grid.major.x = element_blank(),
      panel.grid.minor.x = element_blank()
    )

  return(p)
})


wrap_plots(plot_list, ncol = ncols, nrow = nrows) +
  plot_annotation(
    title = paste0("Primi ", length(plot_list), " cammini minimi (diametro = ", diam, ")"),
    theme = theme(plot.title = element_text(size = 14, face = "bold"))
  )
```



Conclusioni

In questa analisi abbiamo analizzato diversi aspetti locali, comunitari e globali della rete. e' stato mostrato come sotto alcuni aspetti la food web potrebbe essere considerata fragile come nell'analisi della connecttività, mentre sotto gli aspetti di compartimentazione la rete potrebbe sembrare più resiliente ai cambiamenti.