# Визуализация сетей с использованием пакета igraph

*Автор: Владимир Шитиков*
https://stok1946.blogspot.com/

## 1. Введение

Большинство естественных объектов и явлений можно представить в виде сетей. **Сеть** (англ. *network*) представляет собой конечный ориентированный граф **G** = (*V*, *E*), где *V* – множество узлов (англ. *nodes*), а $E \subset V \times V$ – множество рёбер (*edges*), определяющих связи между узлами и часто имеющих ненулевой вес (или "пропускную способность" – *capacity*). Ранее (https://stok1946.blogspot.com/2020/01/blog-post.html) мы рассматривали моделирование корреляционных связей в экологических сообществах с помощью сетей, построенных с использованием пакета `qgraph`. В настоящем сообщении мы продолжим этот разговор, рассматривая работу с пакетом `igraph` , который, в принципе, предназначен для сходных целей, но и имеет существенные концептуальные отличия. Если `qgraph` ориентирован, в основном, на пристальный анализ корреляционных связей, то `igraph` имеет более универсальный характер и включает развитый аппарат настройки свойств графической визуализаций. Набор инструментов сетевого анализа пакета `igraph` имеет акцент на эффективность, переносимость и простоту использования, поэтому интерфейс с его библиотеками охотно используется как другими пакетами среды R, так и модулями на языках Python, C/C++, Mathematica.

В настоящее время особое внимание уделяется интерактивной и анимированной визуализации двух- или трехмерных сетей, а также экспорту графиков R в форматы html/javascript. К сожалению, `igraph` может создавать красивые сетевые визуализации, но они исключительно статичны, поэтому для создания интерактивных сетевых визуализаций используют специальные пакеты R. Один из таких пакетов визуализации - `visNetwork`, который использует библиотеку javascript и основан на html widgets. Он совместим с R Markdown documents и RStudio viewer, имеет множество настроек для персонализации сетей, элегантную графическую часть и хорошую производительность, что очень важно при использовании в Shiny.

На просторах интернета легко можно найти многочисленные руководства и примеры использования пакета `igraph` и `visNetwork`, но мы бы порекомендовали подробный обзор *Katherine Ognyanova* «Network visualization with R», любезно переведенный Анной Каплун (https://habr.com/ru/company/infopulse/blog/268917). При подготовке этого поста большую пользу принесли также сообщения:

https://www.statworx.com/en/content-hub/blog/interactive-network-visualization-with-r – *Niklas Junker* «Interactive Network Visualization with R»

https://www.biostars.org/p/285296/ – *Kevin Blighe* «Network plot from expression data in R using `igraph`»

## 2. Формирование исходных корреляционных матриц

Источником для построения сетей в пакете `igraph` являются чаще всего либо две таблицы со списками узлов и ребер, либо произвольная матрица сходства (*adjacency matrix*). К качестве последних используем корреляционные матрицы, сформированные по результатам многолетнего изучения донных сообществ малых и средних рек бассейна Средней и Нижней Волги, а также данные мониторинга абиотических факторов в этом регионе. Эти данные уже использовались в одном из предыдущих сообщений нашего блога «Анализ статистической связи обилия видов и абиотических факторов с использованием индексов нестабильности»: - https://stok1946.blogspot.com/2021/03/blog-post.html.

Комплект исходных данных по теме сообщения включает три таблицы:

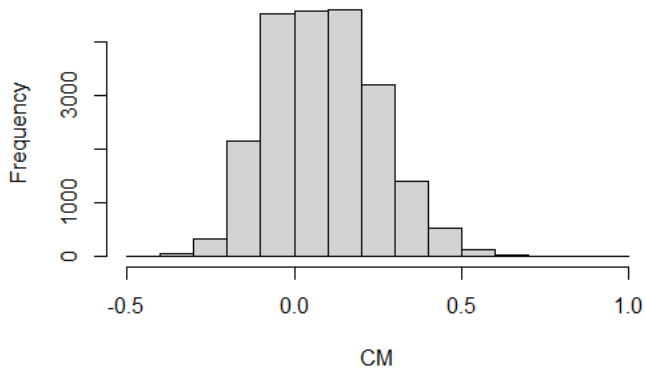
- `TAXA` – усредненные по количеству сделанных проб значения численности 147 видов донных организмов, обнаруженных в 132 реках (участках) изучаемого региона;
- `VAR` – список из 132 рек, включающий их наименование, тип, географические координаты, а также 8 отобранных гео-, метео- и гидрохимических показателей;
- `Species147` – список из наименований 147 видов, включая принадлежность к подсемействам (`Com` – отмеченная встречаемость в реках из 132 всего обследованных).

Таблицы размещены в файле "InStab_dat.RData", который необходимо загрузить с общедоступного ресурса http://www.ievbras.ru/ecostat/Kiril/R/Blog/InStab_dat.RData и поместить в рабочий каталог среды R:

```
#  Загрузка исходных данных из файла
load(file="InStab_dat.RData")
ls()
[1] "Species147" "TAXA"       "VAR"
```

Поставим задачу построить две корреляционных матрицы. Первая – обычная квадратная матрица, симметричная относительно главной диагонали, включает парные коэффициенты корреляции Пирсона между прологарифмированными значениями численностей видов. Вторая прямоугольная матрица включает коэффициенты корреляции между численностью видов, представленными наименованиями строк, и избранными гидрохимическими показателями, размещаемым по столбцам.

```
library(vegan)
#  Логарифмируни значения средних численностей
TAXA_Log <- decostand(TAXA, method="log")
CM <- cor(TAXA_Log)
diag(CM) <- 0
hist(CM,col = "lightgrey")
```




Очевидно, что большинство видов очень слабо коррелируют между собой. Поэтому, оставим только те виды, у которых максимальный коэффициент корреляции хотя бы с одним из остальных видов превышает 0.55.

```
#  Находим максимальные коэффициенты корреляции для каждого вида
Mcor = apply(abs(CM),2,max, na.rm = TRUE)
SpList <- names(Mcor[Mcor > 0.55])
length(SpList)
[1] 46
TAXA_Log46 <- TAXA_Log[,colnames(TAXA_Log) %in% SpList]
colnames(TAXA_Log46)
```

Следуя теоретическим и практическим правилам статистики, удалим из корреляционной матрицы 46×46 все коэффициенты, незначимо отличающиеся от 0, для чего воспользуемся функцией `rcorr` из пакета `Hmisc` .

```
library("Hmisc")
res2 <- rcorr(as.matrix(TAXA_Log46))
res2$r[res2$P > 0.05]<- 0
res2$r[1:5,1:5]   # Матрица коэффициентов кщккуляции
          BiEug.a.  BiHen.h.  BiPis.i.   BiRiv.r.   CeSph.sp
BiEug.a. 1.0000000 0.6303755 0.4306715  0.3728392  0.0000000
BiHen.h. 0.6303755 1.0000000 0.5610212  0.4716187  0.0000000
BiPis.i. 0.4306715 0.5610212 1.0000000  0.4566037  0.0000000
BiRiv.r. 0.3728392 0.4716187 0.4566037  1.0000000 -0.2050623
CeSph.sp 0.0000000 0.0000000 0.0000000 -0.2050623  1.0000000
res2$P[1:5,1:5]   # Матрица p-значений
             BiEug.a.     BiHen.h.     BiPis.i.     BiRiv.r.   CeSph.sp
BiEug.a.           NA 4.440892e-16 2.542427e-07 1.069260e-05 0.07202533
BiHen.h. 4.440892e-16           NA 2.622125e-12 1.142844e-08 0.31761102
BiPis.i. 2.542427e-07 2.622125e-12           NA 3.738172e-08 0.10896958
BiRiv.r. 1.069260e-05 1.142844e-08 3.738172e-08           NA 0.01833900
CeSph.sp 7.202533e-02 3.176110e-01 1.089696e-01 1.833900e-02         NA
#
CM <- as.matrix(as.dist(res2$r))
```

Сформируем теперь прямоугольную матрицу инцидентности 147×3 из коэффициентов корреляции численностей 147 видов с тремя показателями абиотической среды – среднегодовой климатической температурой воздуха (`MTemp`), минерализацией воды (`Miner`) и насыщением воды свободным кислородом (`O2`):

```
str(VAR)
'data.frame':   132 obs. of  13 variables:
 $ Name    : chr  "Актушка" "Аманак" "Анлы" "Б. Вязовка" ...
 $ Type    : Factor w/ 5 levels "верхнее","малая",..: 2 2 2 2 2 1 3 4 5 2 ...
 $ X       : num  49 52 52.3 50.2 50.8 ...
 $ Y       : num  53.4 53.7 53.9 52.5 52.2 ...
 $ NamRiver: Factor w/ 132 levels "Акту","Аман",..: 1 2 3 4 5 6 7 8 9 13 ...
 $ Ground  : int  1 3 5 5 3 3 3 3 3 6 ...
 $ MTemp   : num  47 43 36 47 50 50 61 53 62 78 ...
 $ PrecDQ  : num  82 88 91 86 74 77 96 80 104 53 ...
 $ Alt     : num  98 88 193 110 66 64 23 40 17 10 ...
 $ TRI     : num  94.9 137.5 98.5 41.3 39.2 ...
 $ Miner   : num  860 440 860 2131 530 ...
 $ NH4     : num  0.386 0.18 0.386 0.72 0.69 ...
 $ O2      : num  92.4 87.5 95.5 91.7 140 ...
SelVar <- c("Miner", "O2", "MTemp")
IM <- matrix(0, nrow = 147, ncol = 3)
colnames(IM) <- SelVar
rownames(IM) <- colnames(TAXA)
for (i in 1:3) {
  for (j in 1:147) {
   IM[j,i] <- cor(TAXA_Log[,j],VAR[,SelVar[i]])
  }
}
IM[1:5,]
               Miner          O2      MTemp
AtAth.ib -0.07304784 -0.04424757 -0.2523355
BiEug.a. -0.07658730 -0.01271695 -0.2241143
BiEug.sp -0.14147681 -0.15775507 -0.3309221
BiHen.h. -0.08115863  0.02030030 -0.1900061
BiPis.a  -0.09056371 -0.03564991 -0.2058384
hist(IM)
```

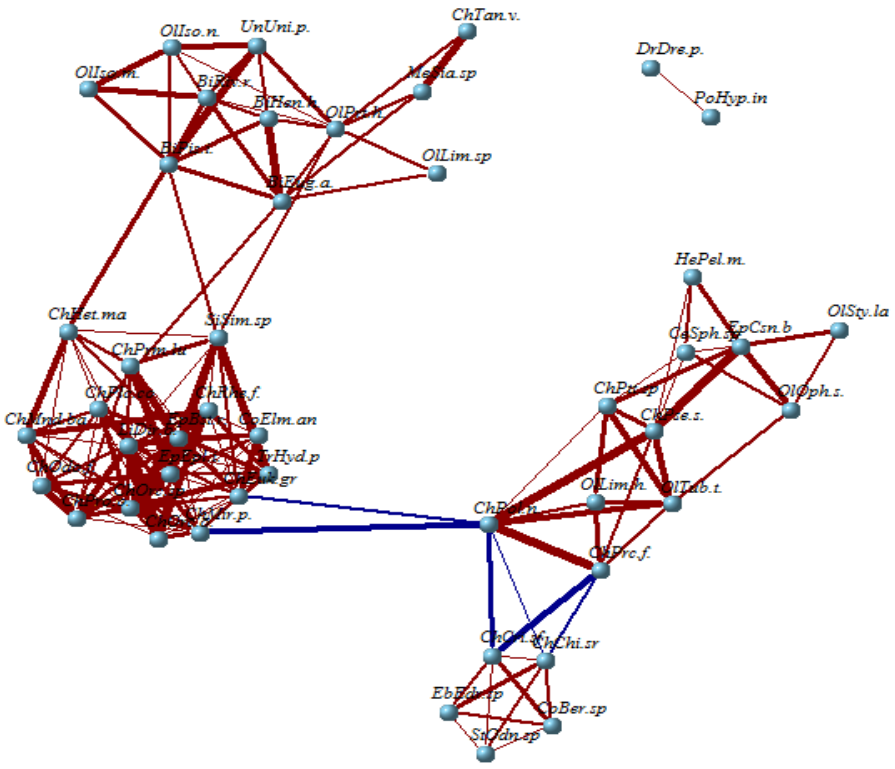
### 3. Построение сети для матрицы взаимосвязи между видами

Загрузим библиотеку `igraph`, сформируем граф на основе корреляционной матрицы из 46×46 видов и выполним ряд профилактических операций, таких как удаление "висячих" узлов, петлей и дублирующихся ребер (хотя в нашем случае это лишняя операция).

```
library(igraph)
# Оохдание графа на основе корреляционной матрицы
# Граф ненаправленный со взвешенными реьрами
g <- graph.adjacency(CM,  mode="undirected",
  weighted=TRUE,  diag=FALSE)
# Упрощение графа (удаление дублирующих ребер и петель
g <- simplify(g, remove.multiple=TRUE, remove.loops=TRUE)
# Удаление узлов, которые ни с кем не связаны
g <- delete.vertices(g, degree(g)==0)
```

Для каждой вершины графа (*vertex*) можно указать следующие настройки: `color` - цвет заливки, `frame.color` - цвет контура, `shape` – форму, `size` - размер, `label` - наборы символов для обозначения вершин, `label.font` – вид шрифта, `label.cex` – размер шрифта, `label.dist` - расстояние между меткой и вершиной, `label.degree` - расположение метки по отношению к вершине. Параметры построений для каждого ребра (*edge*) также содержат большинство из перечисленных спецификаций, но включают дополнительно `width` – ширину ребер, `lty` – тип линий, `arrow.size` – размер стрелок и др. Задать перечисленные атрибуты можно двумя способами: а) заранее установить их как свойства компонентов `V`(*сеть*) и `E`(*сеть*) объекта `igraph`, или б) объявить их в момент визуализации графа функцией `plot`. Оба эти способа задания параметров показаны ниже.

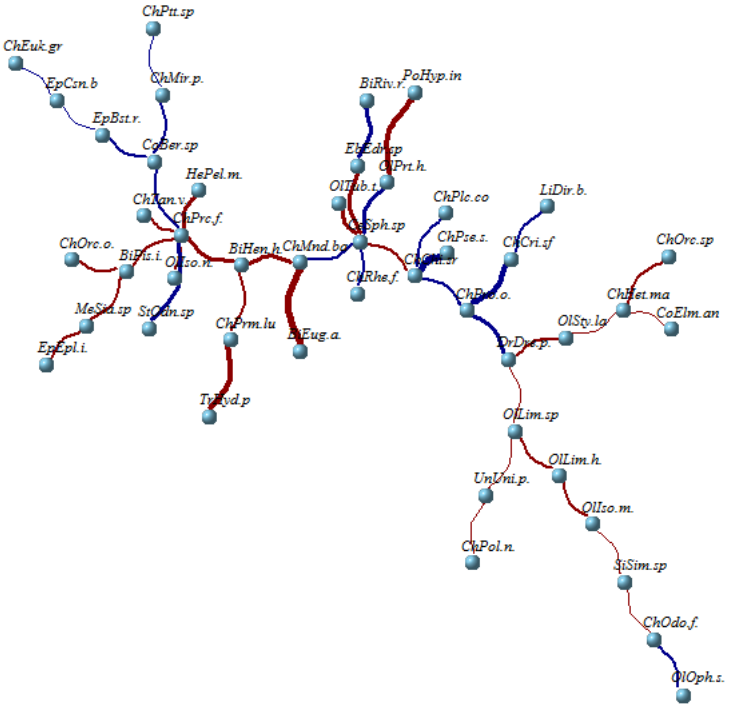
```
# Связи положительных корреляциий окрашены красным
E(g)[which(E(g)$weight>0)]$color <- "darkred"
# Связи отрицательных корреляций окрашены синим
E(g)[which(E(g)$weight<0)]$color <- "darkblue"
# Конвертация весов ребер в абсолютные значения
E(g)$weight <- abs(E(g)$weight)
# Устанавливаем форму и цвет узлов
V(g)$shape <- "sphere"
V(g)$color <- "skyblue"
# Устанавливаем цвет бордюра узлов
V(g)$vertex.frame.color <- "white"
# Устанавливаем ширину ребер по величине корреляции
edgeweights <- E(g)$weight * 9.0
# Удаляем ребра с коэффициентом корреляции меньше среднего
cut.off <- mean(E(g)$weight)
[1] 0.3363319
net.sp <- delete.edges(g, E(g)[weight<cut.off])
# Отображаем сеть графа
par(mar=c(0,0,0,0))
plot(net.sp,
  layout=layout.fruchterman.reingold,
  vertex.size= 5,
  vertex.label.dist=-1,
  vertex.label.degree = pi*3/4,
  vertex.label.color="black",
  vertex.label.font=3,
  vertex.label.cex=0.7,
  edge.width=edgeweights,
  edge.arrow.mode=0)
```




Важное значенре в приведенном скрипте имеет параметр `layout`, отвечающий за стиль размещения узлов сети. В пакет включено много разных алгоритмов, возвращающих координаты каждой вершины графа обеспечивающие минимально возможное количество пересечений ребер. В частности, для графов даже с умеренным числом связей наиболее часто используются силовые алгоритмы размещения Фрюхтермана-Рейнгольда и Камада-Каваи, минимизирующие энергию в "системе пружин".

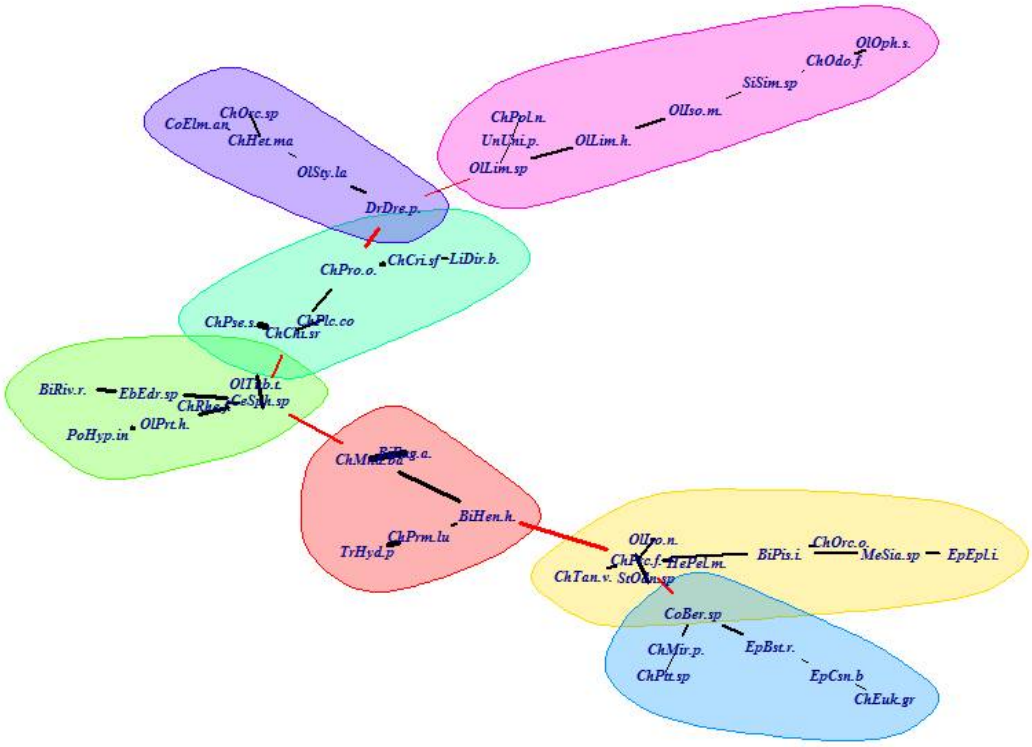
Для достаточно запутанных графов пакет igraph может выполнить построение модели Erdos-Rényi сети и создать соответствующее ей дерево минимального охвата (MST - *Minimal Spanning Tree*). При этом идея моделирования структуры сообщества основана на выделении границ между фрагментами сети с использованием "показателя интенсивности межреберья", оценивающим количество кратчайших путей, проходящих через каждое ребро (подробности см. в описании функции `edge_betweenness`). Таким образом, если постепенно удалять ребра с наименьшим значением показателем, то придем в результате к иерархической структуре, называемую дендрограммой графа. Корень дерева располагается в одной из вершин, ветви образуются узлами с одной входящей и несколькими исходящими связями, а листья - это отдельные вершины, не имеющие исходящих ребер.

```
# Преобразование графа в минимальное связующее дерево
# на основе алгоритма Prim
mst <- mst(g, algorithm="prim")
# Отображаем дерево графа
par(mar=c(0,0,0,0))
plot(mst,
  layout= layout.kamada.kawai,
  edge.curved=TRUE,
  vertex.size= 5,
  vertex.label.dist=-1,
  vertex.label.degree = pi*3/4,
  vertex.label.color="black",
  vertex.label.font=3,
  vertex.label.cex=0.7,
  edge.width=edgeweights,
  edge.arrow.mode=0)
```




Ряд других функций пакета выполняют анализ структуры сообщества на основе выделения границ между отдельными фрагментами графа без построения дерева сети MST. Идея также заключается в том, что ребра, выделяемые в качестве границ, имеют более высокую связность, поскольку все кратчайшие пути от одного кластера к другому должны проходить через них. Функции `edge.betweenness.community` и `make_clusters` реализуют алгоритм последовательного удаления лишних ребер и помечают разной окраской отдельные группы узлов согласно рассчитанным границам.

```
mst.communities <- edge.betweenness.community(mst, weights=NULL, directed=FALSE)
mst.clustering <- make_clusters(mst, membership=mst.communities$membership)
plot( mst.clustering, mst, layout=layout.fruchterman.reingold,
  vertex.shape="none",   vertex.label.font=4,  vertex.label.cex=.7,
  asp=FALSE,  edge.width=edgeweights, edge.arrow.mode=0)
```



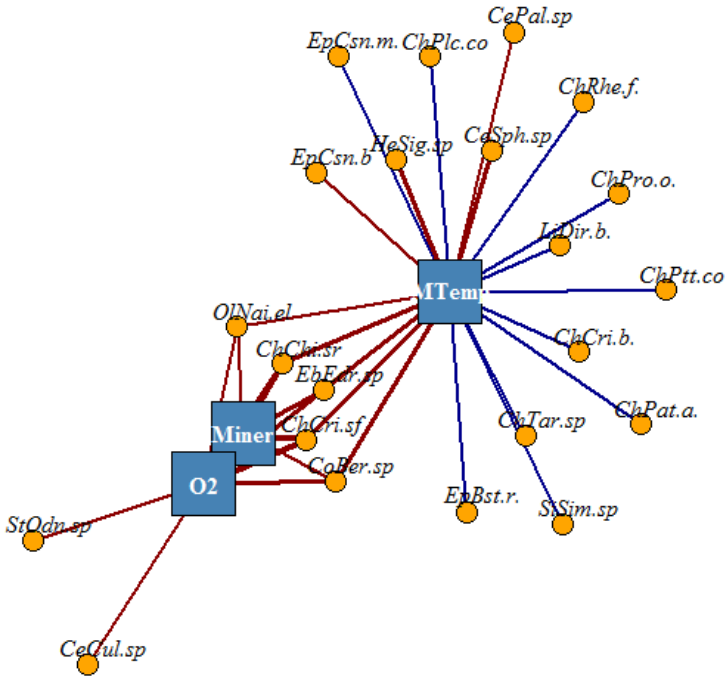
## 4. Построение двудольных сетей с использованием `igraph`

Для парных, или двудольных, графов задаются два разных типа узлов, которые отличаются параметром `type`, равным FALSE для одной группы вершин и TRUE для другой. В этом случае ребра сети связывают узлы разных типов и соответствуют, как правило, матрице инцидентности, которую можно преобразовать в графический объект, используя функцию `graph.incidence`.

```
ig <- graph.incidence(IM, weighted = T)
table(V(ig)$type)
FALSE  TRUE
  147     3
# Связи положительных корреляциий окрашены красным
E(ig)[which(E(ig)$weight>0)]$color <- "darkred"
# Связи отрицательных корреляций окрашены синим
E(ig)[which(E(ig)$weight<0)]$color <- "darkblue"
# Конвертация весов ребер в абсолютные значения
E(ig)$weight <- abs(E(ig)$weight)
```

На этот раз не будем заранее очищать исходную матрицу `IM` от слабых корреляций, поскольку с этим прекрасно справляются функции пакета `igraph` . Удалим все ребра с веeом (т.е. коэффициентом корреляции) менее 0.35, в результате чего число анализируемых видов сократится с 147 до 22.

```
ig.sp <- delete.edges(ig, E(ig)[weight<0.35])
ig.sp <- delete.vertices(ig.sp, degree(ig.sp)==0)
table(V(ig.sp)$type)
FALSE  TRUE
   22     3
V(ig.sp)$color <- c("orange", "steel blue")[V(ig.sp)$type+1]
V(ig.sp)$shape <- c("circle","square")[V(ig.sp)$type+1]
V(ig.sp)$size <- c(7, 20)[V(ig.sp)$type+1]
V(ig.sp)$label.color <- c("black", "white")[V(ig.sp)$type+1]
V(ig.sp)$label.font <- c(3, 2)[V(ig.sp)$type+1]
V(ig.sp)$label.dist <- c(1, 0)[V(ig.sp)$type+1]
edgeweights <- E(ig.sp)$weight * 6.0
par(mar=c(0,0,0,0))
plot(ig.sp, edge.width=edgeweights)
```



## 5. Визуализация сетей с использованием пакета visNetwork

Полученный в предыдущем разделе граф позволяет нам провести отбор подмножества видов, тесно связанных с вершиной MTemp и наиболее чувствительных к изменению широтного градиента. Выполним группировку этих видов на однородные кластеры. Для начала составим список этих видов, вырезав из общей сети соответствующий подграф:

```
ig.spT <- subgraph.edges(ig.sp,
    E(ig.sp)[from(V(ig.sp)["MTemp"])],
    delete.vertices = TRUE)
V(ig.spT)
+ 21/21 vertices, named, from b35a293:
 [1] CePal.sp CeSph.sp ChChi.sr ChCri.b. ChCri.sf ChPat.a. ChPlc.co ChPro.o.
 [9] ChPtt.co ChRhe.f. ChTar.sp CoBer.sp EbEdr.sp EpBst.r. EpCsn.b  EpCsn.m.
[17] HeSig.sp LiDir.b. OlNai.el SiSim.sp MTemp
vdf1 <- get.data.frame(ig.spT, what="vertices")
NamSp20 <- vdf1$name[-21]
```

Рассмотрим, насколько этот список видов представлен корреляционной матрице CM и вершинами графа g , построенном на ее основе в разделе 3:

```
nodes <- get.data.frame(g, what="vertices")
nodes$name[!is.na(match(nodes$name, NamSp20))]
 [1]  "CeSph.sp" "ChChi.sr" "ChCri.sf" "ChPlc.co" "ChPro.o." "ChRhe.f."
 [7] "CoBer.sp" "EbEdr.sp" "EpBst.r." "EpCsn.b"  "LiDir.b." "SiSim.sp"
```

К сожалению, граф g включает только 12 вершин из 20 нужных нам. Поэтому нам нужно начинать сначала и сформировать новую корреляционную матрицу. Но мы, по крайней мере, попрактиковались в сравнении подмножеств двух векторов.

```
TAXA_Log20 <- TAXA_Log[,colnames(TAXA_Log) %in% NamSp20]
tg <- graph.adjacency(cor(TAXA_Log20),  mode="upper", weighted=TRUE)
tg <- simplify(tg)
tg.sp <- delete.edges(tg, E(tg)[weight < 0.4])
```

Задача выделения сообществ в графе известна давно и имеет множество методов и практических приложений ( https://habr.com/ru/company/raiffeisenbank/blog/504820/ ). Самым популярным алгоритмом выделения сообществ является Louvain Community Detection Algorithm. Суть заключается в том, что при перемещении отдельной вершины из одного сообщества в другое нам не надо пересчитывать полностью всю матрицу оценок, а достаточно лишь пересчитать вклад перемещаемой вершины

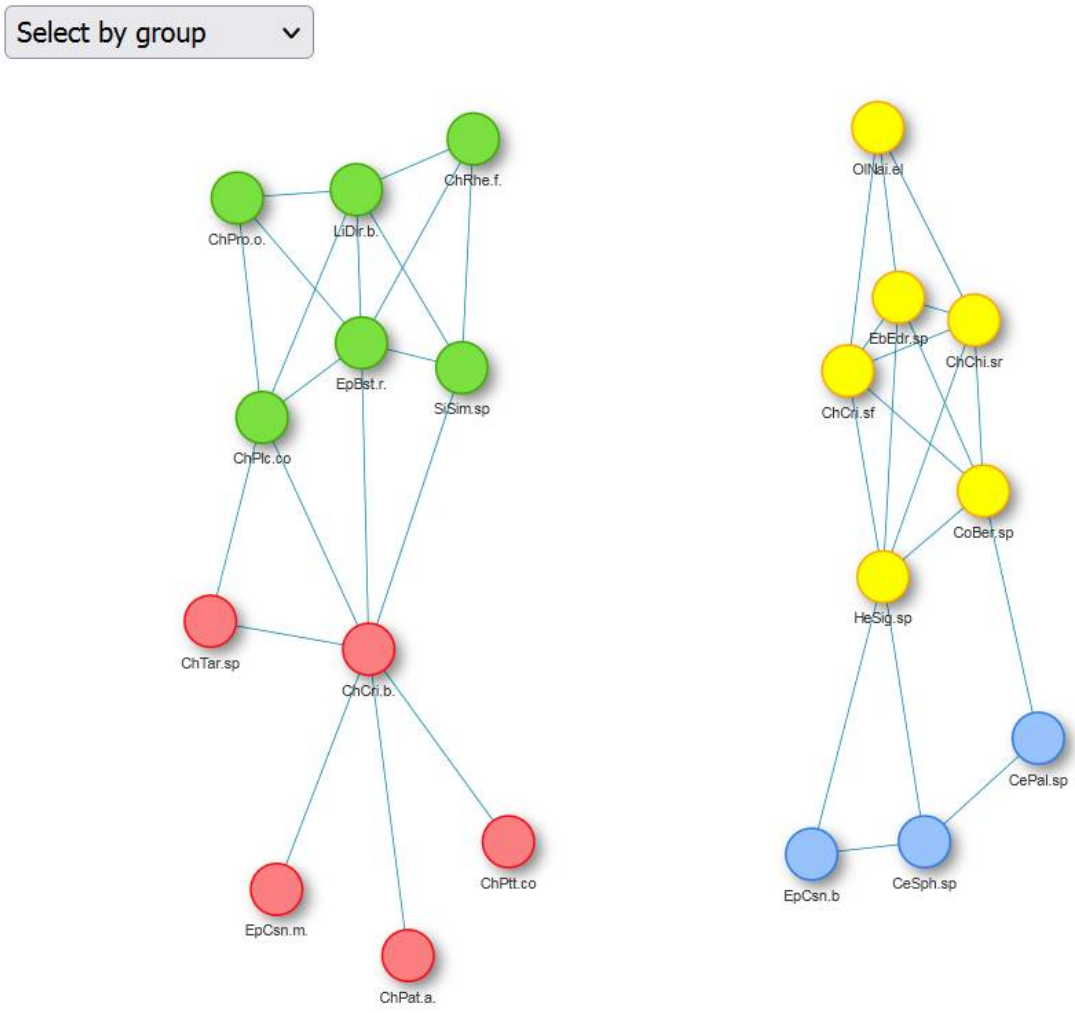
```
#Louvain Comunity Detection
cluster <- cluster_louvain(tg.sp)
cluster_df <- data.frame(as.list(membership(cluster)))
cluster_df <- as.data.frame(t(cluster_df))
cluster_df$label <- rownames(cluster_df)
```

Для того, чтобы воспользоваться визуализацией нашего графа в пакете visNetwork, необходимо сформировать два фрейма со списком вершин nodes и ребер edges.

```
 #
nodes <- get.data.frame(tg.sp, what="vertices")
nodes <- data.frame(id = nodes$name, label = nodes$name)
edges <- get.data.frame(tg.sp, what="edges")[1:2]
# Создание столбца с номером группы
library(tidyverse)
nodes <- left_join(nodes, cluster_df, by = "label")
colnames(nodes)[3] <- "group"
```

Визуализация сети функцией `visNetwork`, выполняется на вкладке вашего Интернет-браузера. В зависимости от задаваемых параметров можно получить изображение разной степени крутизны:

```
library(visNetwork)
# Сеть попроще
visNetwork(nodes, edges)
# Сеть покруче
visNetwork(nodes, edges, width = "100%") %>%
  visIgraphLayout() %>%
  visNodes(shape = "dot", color = list(background = "#0085AF",
      border = "#013848", highlight = "#FF8000"),
    shadow = list(enabled = TRUE, size = 10)
  ) %>%
  visEdges(
    shadow = FALSE, color = list(color = "#0085AF", highlight = "#C62F4B")
  ) %>%
  visOptions(highlightNearest = list(enabled = T, degree = 1, hover = T),
             selectedBy = "group") %>%
  visLayout(randomSeed = 11)
```

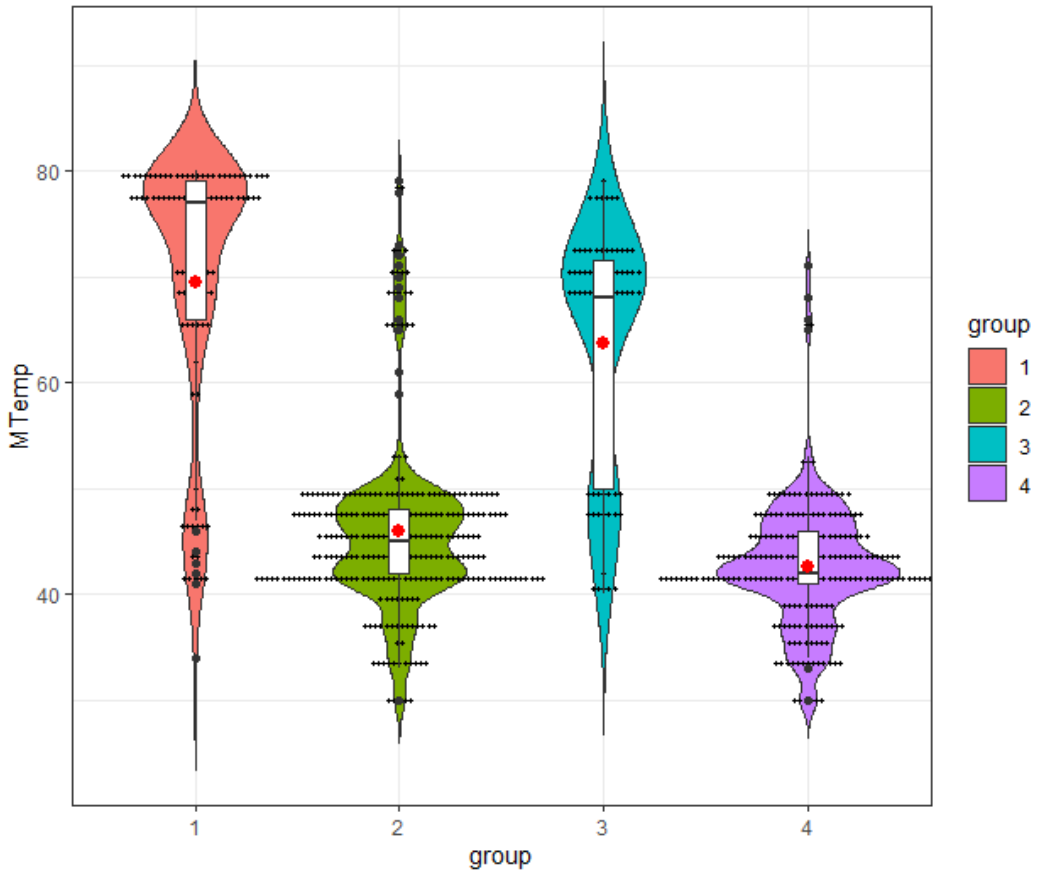



Использование функции `visOptions` делает вашу сеть интерактивной: выбор значения `group` из выпадающего списка выполняет подсветку нужного вам отдельного кластера видов. Крутизну картинки можно еще повысить, изучив документацию к пакету и сопутствующие материалы

Интересно рассмотреть, имеют ли наши упражнения с графами сети реальное экологическое содержание. Для этого оценим, связана ли статистически значимо полученная нами группировка вершин графа с величиной среднегодовой температурой местности (°C*10), где эти группы видов были обнаружены.

```
TAXA20 <- TAXA[,colnames(TAXA) %in% nodes$label]
TAXA20[TAXA20>0] <- 1
TAXA20 <- sweep(TAXA20, MARGIN=1, VAR$MTemp, `*`)
library(reshape2)
TempByGroup <- subset(melt(TAXA20),value > 0)
colnames(TempByGroup) <- c("label", "MTemp")
TempByGroup <- left_join(TempByGroup, nodes[,-1], by = "label")
TempByGroup$group <- as.factor(TempByGroup$group)
summary(aov(MTemp ~ group, data = TempByGroup))
             Df Sum Sq Mean Sq F value Pr(>F)
group         3  66539   22180   276.4 <2e-16 ***
Residuals   643  51606      80
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

ggplot(TempByGroup, aes(x = group, y = MTemp )) +
  geom_violin(aes(fill = group),trim = FALSE) +
  geom_dotplot(binaxis='y', stackdir='center', dotsize=0.3) +
  geom_boxplot(width=0.1, fill="white")+
  stat_summary(fun.y=mean, geom="point", shape=20,
               size = 4, color="red", fill="red") +
  theme_bw()
```




Анализ результатов статистического анализа и параметров распределения `MTemp` по группам позволяет выделить сообщества видов, обитаемых преимущественно в водотоках засушливой полупустыни (гр. 1) и равнинных степных реках (гр. 3) . Виды, характерные для лесостепного региона, образовали группы 2 и 4, которые отличаются между собой влиянием каких-то иных различных факторов.

Чтобы представить интерактивную визуализацию полученных выше результатов клиентам Вашей сети, или глобальному Интернету, скрипт на основе `visNetwork` можно интегрировать в приложение Shiny (см. например, наше сообщение https://stok1946.blogspot.com/2021/01/shiny.html). Для этого исходные данные подготавливаются "оффлайн", файлы узлов и ребер сохраняются на сервере или в рабочем каталоге, после чего наше изображение формируется "онлайн" внутри приложения Shiny. Вот минимальный пример кода, которую можно использовать для Shiny:

```
save(nodes,  file = "nodes.RData")
save(edges,  file = "edges.RData")
```

Блок `global.R`:

```
library(shiny)
library(visNetwork)
```

Блок `server.R`:

```
shinyServer(function(input, output) {
  output$network <- renderVisNetwork({
    load("nodes.RData")
    load("edges.RData")
    visNetwork(nodes, edges) %>%
      visIgraphLayout()
  })
})
```

Блок `ui.R`:

```
shinyUI(
  fluidPage(
    visNetworkOutput("network")
  )
)
```

Другой путь публикации формируемых сетевых объектов – экспорт графики R в html/javascript с помощью таких библиотек, как `rcharts` и `htmlwidgets`. которые пригодятся при создании веб-графиков непосредственно из R. Создание интерактивных визуализаций сетей с помощью javascript-библиотеки D3 возможно также на базе пакета `networkD3`.