

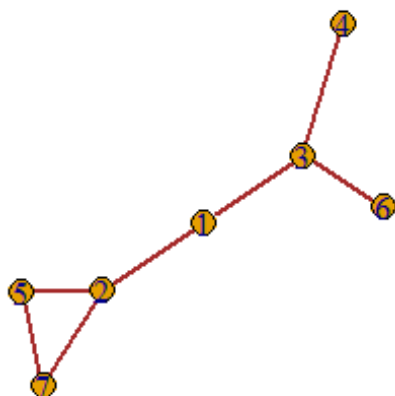
Vytváranie sietí v balíku igraph

```
library(igraph)
```

graph_from...

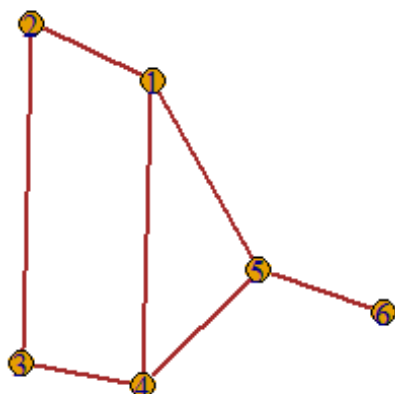
Funkciu `graph_from_literal` sme už videli, dá sa spraviť napríklad:

```
g <- graph_from_literal(1-2, 1-3, 2-5, 2-7, 3-4, 3-6, 5-7)
plot(g, edge.width = 2, edge.color = "brown")
```



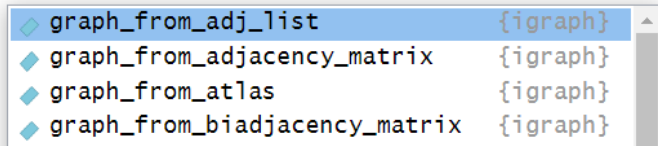
Vrcholy nemusíme spájať iba po dvoch, môžeme zapísať aj dlhšiu “cestu” spájajúcu viac vrcholov:

```
g <- graph_from_literal(1-2-3-4-1-5-6, 5-4)
plot(g, edge.width = 2, edge.color = "brown")
```

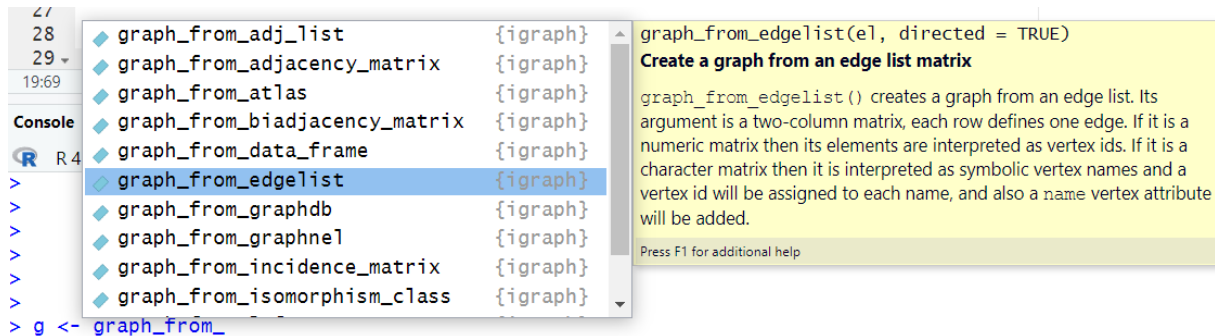


Keď začítame písať názov funkcie `graph_from_literal`, môžeme si všimnúť, že sa nám začne ponúkať viac funkcií, ktoré sa začínajú slovami `graph_from`. Funkciu vytvárajúcu graf z matice susednosti `graph_from_adjacency_matrix` sme už používali, môžeme sa pozrieť aj na ďalšie a zistiť, z čoho ešte môžeme graf vytvoriť.

```
> g <- graph_from_
```



Pri prechádzaní zoznamom ponúkaných možností sa nám zobrazuje aj informácia o funkciách.



Môžeme si samozrejme zobrazíť dostupnú informáciu pomocou otáznika pred názvom funkcie, napríklad ?graph_from_literal. Takisto po zapísaní funkcie sa nám zobrazí, aké parametre potrebuje.

```
>
>
> g <- graph_from_edgelist()
```

graph_from_edgelist(e1, directed = TRUE)

Pomocou tabulátora pri písaní ich môžeme získať v nasledovnej forme:

```
> g <- graph_from_edgelist()
e1 = graph_from_edgelist()
directed = graph_from_edgelist()
```

e1
The edge list, a two column matrix, character or numeric.
Press F1 for additional help

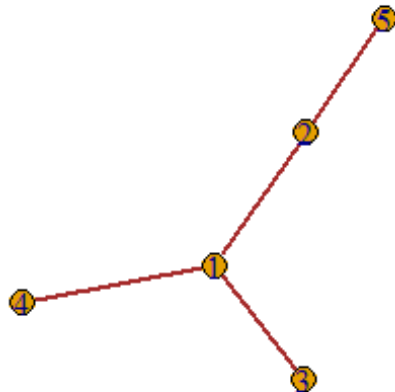
V prípade tejto funkcie **graph_from_edgelist** teda potrebujeme zoznam hrán v tvare **matice s dvoma stĺpcami, pričom každý riadok zodpovedá jednej hrane**. Prakticky je užitočné vytvoriť si ju napríklad v Exceli, niekedy môže byť výstupom z nejakého výpočtu, v tomto ukážkovom príklade si ju vytvoríme v R-ku.

```
hrany <- rbind(c(1, 2),
               c(1, 3),
               c(1, 4),
               c(2, 5))
```

```
hrany
##      [,1] [,2]
## [1,]    1    2
## [2,]    1    3
## [3,]    1    4
## [4,]    2    5
```

Teraz môžeme spraviť:

```
g <- graph_from_edgelist(hrany, directed = FALSE)
plot(g, edge.width = 2, edge.color = "brown")
```



Analogicky funguje `graph_from_data_frame`. Ako vidno aj z názvu, potrebuje ako vstup *data frame*. To môže byť užitočné, ak sú po načítaní dáta práve v tomto formáte. Okrem toho nám umožňuje priradiť hranám atribúty, napríklad váhu.

```
hrany <- data.frame(from = c(1, 1, 1, 2),
                    to = c(2, 3, 4, 5),
                    weight = c(3, 3, 5, 1))
```

```
hrany
```

```
##   from to weight
## 1    1  2      3
## 2    1  3      3
## 3    1  4      5
## 4    2  5      1
```

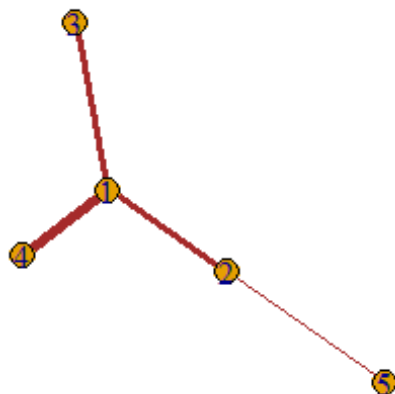
```
g <- graph_from_data_frame(hrany, directed = FALSE)
```

Graf, ktorý vznikol, je vážený:

```
is_weighted(g)
```

```
## [1] TRUE
```

```
plot(g, edge.width = E(g)$weight, edge.color = "brown")
```



Pri vytváraní ukážkových príkladov sa niekedy môže hodiť `graph_from_atlas`

<code>graph_from_adj_list</code>	<code>{igraph}</code>
<code>graph_from_adjacency_matrix</code>	<code>{igraph}</code>
<code>graph_from_atlas</code>	<code>{igraph}</code>
<code>graph_from_biadjacency_matrix</code>	<code>{igraph}</code>
<code>graph_from_data_frame</code>	<code>{igraph}</code>
<code>graph_from_edgelist</code>	<code>{igraph}</code>
<code>graph_from_graphdb</code>	<code>{igraph}</code>
<code>graph_from_graphnel</code>	<code>{igraph}</code>
<code>graph_from_incidence_matrix</code>	<code>{igraph}</code>
<code>graph_from_isomorphism_class</code>	<code>{igraph}</code>

`graph_from_atlas(n)`
Create a graph from the Graph Atlas

`graph_from_atlas()` creates graphs from the book 'An Atlas of Graphs' by Roland C. Read and Robin J. Wilson. The atlas contains all undirected graphs with up to seven vertices, numbered from 0 up to 1252. The graphs are listed:

1. in increasing order of number of nodes;
2. for a fixed number of nodes, in increasing order of the number of edges;
3. for fixed numbers of nodes and edges, in increasing order of the degree sequence, for example 111223 < 112222;
4. for fixed degree sequence, in increasing number of automorphisms.

Press F1 for additional help

Je to sada grafov s malým počtom vrcholov a ak chceme niečo ukázať na malom grafe, ktorý nemusí mať špeciálne vlastnosti, môžeme niektorý vyskúšať. Možno po pár pokusoch budeme mať vhodný graf pre náš príklad a bude jednoduché ho vytvoriť.

add...

Do už vytvoreného grafu môžeme pridávať vrcholy a hrany.

```
> g <- add_
```

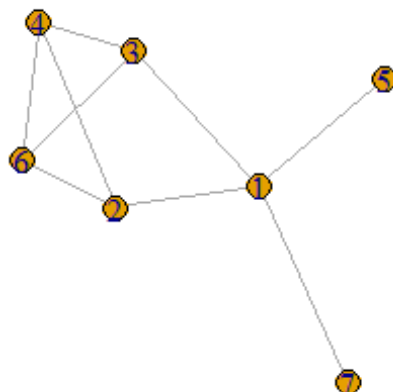
<code>add_edges</code>	<code>{igraph}</code>
<code>add_layout_</code>	<code>{igraph}</code>
<code>add_shape</code>	<code>{igraph}</code>
<code>add_vertices</code>	<code>{igraph}</code>
<code>add.edges</code>	<code>{igraph}</code>
<code>add.scope</code>	<code>{stats}</code>

`add_vertices(graph, nv, ..., attr = list())`
Add vertices to a graph

If attributes are supplied, and they are not present in the graph, their values for the original vertices of the graph are set to NA.

Press F1 for additional help

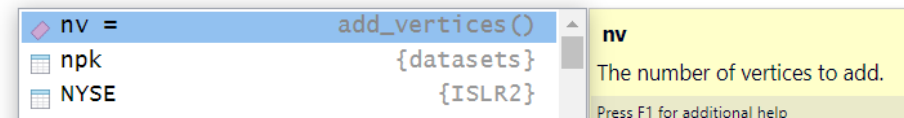
```
g <- graph_from_atlas(500)  
plot(g)
```



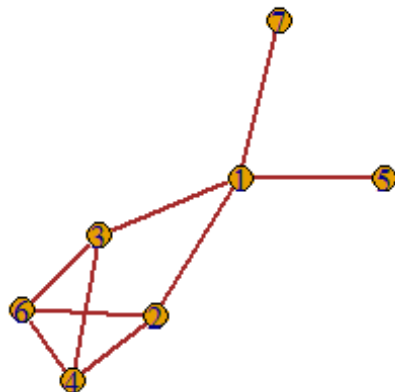
Pridáme do horeuvedeného grafu vrcholy od 8 do 12. Pri zapisovaní funkcie hodiť `add_vertices` vidíme, že budeme potrebovať parameter `nv`. Bud' si to pozrieme v helpe alebo

si necháme poradiť RStudiosom – v každom prípade sa dozvieme, že treba zadať počet pridávaných vrcholov.

```
> g <- add_vertices(g, n)
```



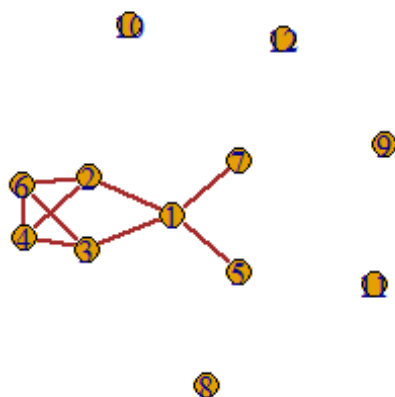
```
g <- graph_from_atlas(500)
plot(g, edge.width = 2, edge.color = "brown")
```



```
g <- add_vertices(g, nv = 5)
```

Pre neskoršie porovnávanie definujeme pred kreslením *layout*, teda rozloženie vrcholov:

```
set.seed(123)
lay <- layout_nicely(g)
plot(g, layout = lay, edge.width = 2, edge.color = "brown")
```



Pridávať môžeme aj hrany – funkcia má logický názov – **add_edges**. Dozvieme sa, že máme zadať postupnosť vrcholov.

```
> g <- add_edges(g, edge)
```

```
edges = add_edges()
edge {igraph}
edge.attributes {igraph}
edge.attributes<- {igraph}
```

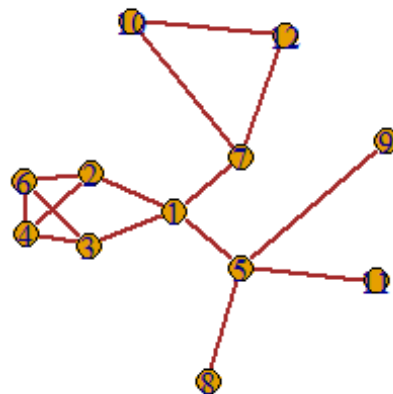
edges

The edges to add, a vertex sequence with even number of vertices.

Press F1 for additional help

Postupne teda píšeme dvojice vrcholov, ktoré sa majú spojiť a vytvoríme takto jeden vektor, ktorý bude parametrom funkcie

```
g <- add_edges(g, edges = c(8, 5, 11, 5, 9, 5, 12, 7, 10, 12, 10, 7))
plot(g, layout = lay, edge.width = 2, edge.color = "brown")
```



Atribúty

S atribútmi sme sa stretli, keď sme zo siete karate klubu nastavovali váhu všetkých hrán na jednotkovú (buď sme tento atribút zmazali alebo ho nastavili na jednotkovú hodnotu). Teraz si ako cvičenie nastavíme atribút farby pre hrany našej siete na hnedú a pridáme niekoľko hrán inej farby. Pri písaní funkcie `add_edges` pre pridávanie hrán sme si všimli parameter `attr`, tak ho teraz využijeme. Toto sme už teda videli:

```
>
> set_edge_attr(graph, name, index = E(graph), value)
> set_edge_attr(g, |
```

Spravíme:

```
g <- set_edge_attr(g, name = "color", value = "brown")
```

Pri pridávaní hrán funkciou `add_edges` majú byť dodávané atribúty typu *list*:

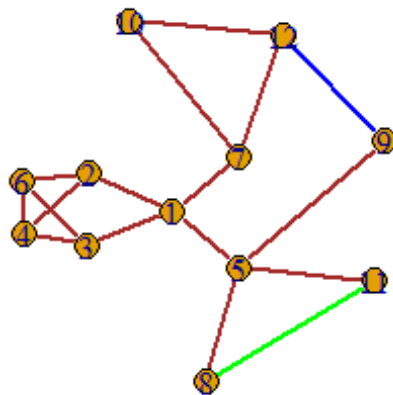
```
), attr|
```

```
attr = add_edges()
attenu {datasets}
attitude {datasets}
```

attr

A named list, its elements will be added as edge attributes, for the newly added edges. See also details below.

```
g <- add_edges(g,
  edges = c(9, 12, 8, 11),
  attr = list(color = c("blue", "green")))
plot(g, layout = lay, edge.width = 2)
```



Vytváranie špeciálnych grafov: make...

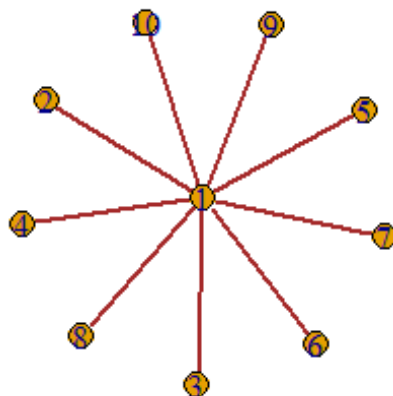
Začneme písať **make** a objavia sa možnosti, medzi ktorými je aj vytváranie špeciálnych typov grafov. Napríklad:

```
> make_
```

◆ make_full_graph	{igraph}	make_star(n, mode = c("in", "out", "mutual", "undirected"), center = 1) Create a star graph, a tree with n vertices and n - 1 leaves star() creates a star graph, in this every single vertex is connected to the center vertex and nobody else. Press F1 for additional help
◆ make_graph	{igraph}	
◆ make_chordal_ring	{igraph}	
◆ make_kautz_graph	{igraph}	
◆ make_lattice	{igraph}	
◆ make_line_graph	{igraph}	
◆ make_neighborhood_graph	{igraph}	
◆ make_ring	{igraph}	
◆ make_star	{igraph}	
◆ make_tree	{igraph}	

Teda jeden vrchol je v strede a ostatné sú s ním spojené, pričom iné hrany v sieti nie sú:

```
g <- make_star(10, mode = "undirected")
plot(g, edge.width = 2, edge.color = "brown")
```



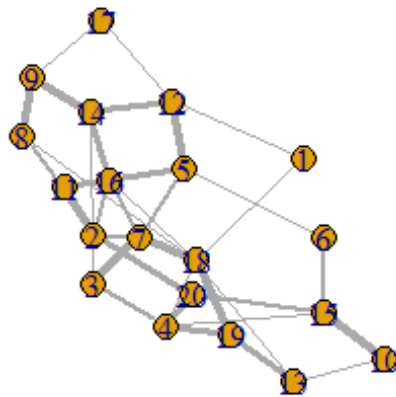
Môže sa hodiť aj napríklad `make_full_graph`, `make_ring`. Treba chvíľu skúšať, čo všetko je v balíku k dispozícii.

Naopak: máme graf, chceme zapísať jeho hrany a pod.

Ak vytvárame graf, môžeme v určitom okamihu chcieť vyexportovať zoznam jeho hrán alebo vrcholov spolu s ich vlastnosťami.

Vytvoríme si nejaký graf a priradíme hranám váhy.

```
set.seed(123)
g <- sample_gnp(n = 20, p = 0.2)
g <- set_edge_attr(g, name = "weight", value = runif(length(E(g))))
plot(g, edge.width = 5*E(g)$weight)
```



Zostrojíme *data frame* s informáciou o hranách – keďže chceme zo siete spraviť *data frame*, funkcia sa volá `as_data_frame`:

```
df <- as_data_frame(g, what = "edges")
head(df)
```

```
##   from to    weight
## 1    3  4 0.5999890
## 2    5  6 0.3328235
## 3    2  7 0.4886130
## 4    3  7 0.9544738
## 5    5  7 0.4829024
## 6    8  9 0.8903502
```

Takýto výstup môžeme napríklad zapísať do súboru.