Mini-guia do R para tratamento de grafos e mapas

Carregar bibliotecas

```
require(igraph)
require(ggplot2)
require(ggmap)
```

Operações estatísticas básicas do R

```
mean()
std()
var()
summary()
hist()
```

Criação de grafos

Usando matriz de adjacência

```
grafo <- matrix(c(0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0,
0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0), ncol = 5)
plot(graph_from_adjacency_matrix(grafo, mode =
'directed'))
```

Usando lista de adjacência

```
grafo <- list(c(2, 3, 4), c(1, 3, 5), c(1, 2, 5), c(1,
5), c(2, 3, 4))
plot(graph_from_adj_list(grafo, mode = 'all'))
```

Usando lista pareada de nós para formar arestas.

```
grafo <- matrix(c(1, 2, 1, 3, 1, 4, 2, 3, 2, 5, 3, 5, 4,
5), ncol = 2, byrow = TRUE)
plot(graph_from_edgelist(grafo, directed = FALSE))
```

Usando lista dita literal, onde claramente as arestas são definidas.

```
grafo <- graph_from_literal(1--2, 1--3, 1--4, 2--3, 2--5,
3--5, 4--5)
plot(grafo)
```

Conversões forçadas (cast)

Para lista de conversões, procure por comandos começando com "as_" no *help* do igraph para R

```
as_adj(grafo)
as_adj_list(grafo)
as_data_frame(grafo)
as_ids(V(grafo))
as_ids(E(grafo))
```

Extração de informações dos grafos

Obter um vetor contendo todos os vértices/nós de um grafo

```
V(grafo)
V(grafo)$name
V(grafo)$color
V(grafo)$atributo
```

Obter uma "lista" contendo todas as arestas/ligações dos grafos

```
E(grafo)
E(grafo)$weight
E(grafo)$color
E(grafo)$atributo
```

Atribuição de informações aos grafos

Incluir um atributo novo

```
V(grafo)$atributo_novo <- c(...)
E(grafo)$atributo_novo <- c(...)
```

Métricas de grafos/redes complexas

Métricas baseadas em nós

```
vetor_degree <- degree(grafo)
degree_distribution(grafo, cumulative = TRUE)
vetor_strength <- strength(grafo)
```

Este último comando usa os pesos das arestas para calcular o grau ponderado

```
fit_power_law(vetor_degree, xmin = 5)
```

Métricas baseadas em caminho

```
escalar <- diameter(grafo)
vetor <- get_diameter(grafo)
lista <- farthest_vertices(grafo)
escalar <- mean_distance(grafo, directed = FALSE)
matriz <- distances(grafo, weights = NA)
matriz <- distances(grafo, weights = NULL)
distance_table(grafo, directed = FALSE)
vetor <- closeness(grafo, normalized = TRUE)
vetor <- betweenness(grafo, normalized = TRUE)
vetor <- edge_betweenness(grafo, directed = TRUE)
```

Métricas baseadas em agrupamento

```
valor <- transitivity(grafo, type = 'global')
vetor <- transitivity(grafo, type = 'local')
vetor <- transitivity(grafo, type = 'local', weights =
NA)
comunidade <- cluster_walktrap(grafo)
comunidade <- cluster_fastgreedy(grafo)
vetor <- membership(comunidade)
valor <- modularity(comunidade)
matriz <- modularity_matrix(comunidade)
plot(comunidade, grafo)
```

Controle de plotagem de grafos

Plotagem "convencional"

```
plot(grafo, edge.color = vetor)
plot(grafo, edge.width = vetor)
plot(grafo, vertex.size = vetor)
plot(grafo, vertex.label = vetor)
plot(grafo, layout = layout_as_star)
coordenadas <- layout_in_circle(grafo)
plot(grafo, layout = coordenadas)
```

Usando ggplot2

Lembrar que se trata de uma visualização hierárquica. Depende de conjunto de dados descritos através de *data frames* e uma gramática de visualização.

```
ggplot(data = meu_df) +
geom_point(mapping = aes(x = coluna_df1, y = coluna_df2)
```

ou

```
ggplot() +
geom_point(data = meu_df, aes(x = coluna_df1, y =
coluna_df2)
```

ou

```
ggplot() +
geom_point(data = meu_df, aes(x = coluna_df1, y =
coluna_df2, color = coluna_df3, size = coluna_df4)
```

Plotando linhas

```
ggplot() +
geom_segment(data = meu_df, aes(x = coluna_df1, y =
coluna_df2, xend = coluna_df3, yend = coluna_df4)
```

ou

```
ggplot() +
geom_path(data = meu_df, aes(x = coluna_df1, y =
coluna_df2)
```

Usando ggmap

Esta biblioteca consegue carregar dinamicamente mapas do Google Maps (agora pago), OpenStreetMaps (atualmente com algum problema) e Statmen (o único serviço que funciona). Os comandos abaixo são voltados para o statmen

```
require(ggmap)
city_region <- c(left=-49.38039, top=-25.34738, right=-
49.17349, bottom=-25.63794)
get_map(location=city_region, zoom=13) %>% ggmap
mapa <- get_map(location=city_region, zoom=12)
ggmap(mapa)
```

O comando **`ggmap()`** funciona como um **`ggplot()`**, criando uma camada gráfica com o mapa, que pode ser sobreposta por qualquer diretiva de ggplot2.