Trabalho 1 | AR

Rede de Contactos Sociais

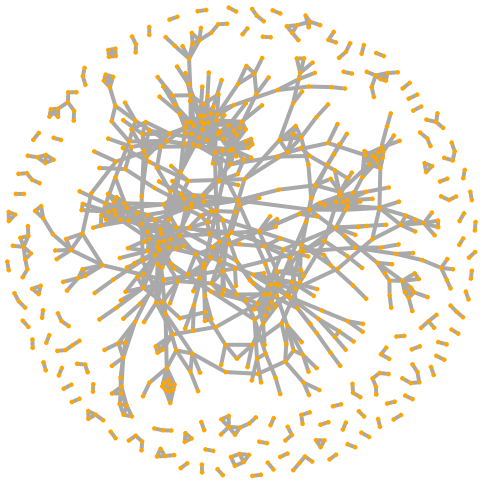
André Silvestre Nº104532 Eliane Gabriel Nº103303
Maria João Lourenço Nº104716 Margarida Pereira Nº105877
Umeima Mahomed Nº 99239

15 de dezembro de 2023

Questão 1

```
# Importar a rede de um ficheiro .txt
rede <- read_graph("trab_links.txt", format = "edgelist", directed=F)

# Representar graficamente a rede
plot(rede, vertex.color= "orange", vertex.label=NA, vertex.size=2,
     vertex.frame.color=NA, edge.width= 2)
```



Nº de Nodos e de Ligações

```
cat("A rede tem", vcount(rede) ,'nodos e', ecount(rede), 'ligações.')
```

```
## A rede tem 787 nodos e 1197 ligações.
```

Densidade

```
edge_density(rede)
```

```
## [1] 0.003870142
```

Graus dos Nodos

```
# Grau de cada nó
grau <- data.frame("Grau (k_i)"=degree(rede,mode='all'))
grau_transposto <- as.data.frame(t(grau))
colnames(grau_transposto) <- 1:vcount(rede) # Cada coluna é um nodo
grau_transposto
```

```
##            1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
## Grau..k_i. 9 4 4 7 1 1 1 3 4  6  4  4  3  1  1  2  1  1  3  1  3  2  3  2  2  1
##            27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
## Grau..k_i.  2  1  5  5  1  5  1  1  1  2  6  2  5  2  2  2  1  1  4  2  2  1  1
##            50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
## Grau..k_i.  1  3  3  2  2  1  1  1  7  6  1  1  1  8  1  2  2  2  6  3  1  5  1
##            73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
## Grau..k_i.  1  3  1  1  3  1  4  3  4  1  1  5  4  1  1  1  4  2  1  1  5  1  4
##            96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113
## Grau..k_i.  3  1  1  3   1   4   2   2   3  11   2   2   3   1   2   1   1   1
##            114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130
## Grau..k_i.   1   5   4   1   2   1   1   2   1   5   3   3  12   4   2   2   1
##            131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
## Grau..k_i.   1   1   1   1   1   1   1  10   1   6   1   2   1   2   2   1   1
##            148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164
## Grau..k_i.   5   1   1   1   1   2   1   2   1   1   1   1   1   3   2   4   8
##            165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181
## Grau..k_i.   5  12   8   7  10  10  11  10   9   4  17   6   8  12  13   4   3
##            182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
## Grau..k_i.   6   3   8   7  13   9   7   4   4   4   2   1   5   3   8  13   8
##            199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
## Grau..k_i.   6   7  10  17  11   9   9   4   6   4   7   5   1   4   1   5   3
##            216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232
## Grau..k_i.   9   6   6   4   1   2   1   5   3   7  10   5   4   3   5  10   5
##            233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249
## Grau..k_i.   6   5   2   6   6   8   8   8   8   8   9   2   8  10   5   9   7
##            250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266
## Grau..k_i.   1   7   4   5   3   6   6   4   7   2  12   5   6   3  11   6   7
```

```
##            267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283
## Grau..k_i.   2   3   6   3   8  12  10   9   3   2  12   7   2   2   8   4   2
##            284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
## Grau..k_i.  10   7   9  11   8   5   6   2   1   2   5   1   8   4   2  16   1
##            301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317
## Grau..k_i.   3   1   1   1   3   1   3   1   1   4   1   2   2   8   8   5   9
##            318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334
## Grau..k_i.   5   2   9   1   2   2   2   7   2   3   3   1   6   7   3   1   1
##            335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351
## Grau..k_i.   1   1   2   4  11   4   5   1   1  10   3   3   4   3   8   3   4
##            352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368
## Grau..k_i.   4   1   1   4  12   2   3   3   2   2   3   1   1   1   2   2   2
##            369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385
## Grau..k_i.  11   2   2  14  12   1   2   3   1   2  16   4   6   6   9   5   1
##            386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402
## Grau..k_i.   1   4   5   1   5   2  11   5   1   1   4   4   5   3   1   1   2
##            403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419
## Grau..k_i.   4   4   2   4   1   2   2   1   4   9   2   2   2   1   2   2  10
##            420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436
## Grau..k_i.   7   1   1   2   5   2   1   1   2   3   3   1   1   1   3   3   3
##            437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453
## Grau..k_i.   3   3   2   5   4   1   1   1   3   7   3   3   1   1   5   1   2
##            454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470
## Grau..k_i.   1   4   4   1   3   3   2   2   6   4   3   5   4   1   7   1   2
##            471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487
## Grau..k_i.   1   1   2   2   2   2   3   2   3   1   1   2   1   1   3   1   1
##            488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504
## Grau..k_i.   5   2   2   6   2   1   2   1   1   1   1   3   3   2   2   4   5
##            505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
## Grau..k_i.   2   4   3   6   1   1   1   2   4   3   2   1   1   1   2   1   1
##            522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538
## Grau..k_i.   3   3   1   1   2   2   3   1   3   2   2   1   2   2   1   1   2
##            539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555
## Grau..k_i.   1   1   3   1   3   5   1   1   1   2   1   1   2   1   2   1   1
##            556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572
## Grau..k_i.   1   1   2   1   1   1   4   1   1   4   1   1   1   1   1   1   2
##            573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589
## Grau..k_i.   3   2   1   1   2   3   1   3   2   1   1   1   2   1   1   1   4
##            590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606
## Grau..k_i.   2   2   1   2   2   1   2   2   1   3   1   2   3   1   1   3   7
##            607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623
## Grau..k_i.   3   2   3   2   2   1   1   1   1   2   1   1   1   1   1   1   2
##            624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640
## Grau..k_i.   1   3   1   5   2   1   1   4   3   1   2   1   2   1   1   1   2
##            641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657
## Grau..k_i.   1   1   1   2   2   2   3   3   3   3   1   2   2   2   2   1   1
##            658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674
## Grau..k_i.   2   1   1   1   1   1   2   2   8   1   3   1   1   1   2   1   1
##            675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691
## Grau..k_i.   1   1   1   2   1   1   3   1   3   1   1   1   1   2   2   3   1
##            692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708
## Grau..k_i.   3   2   1   2   3   4   1   1   1   2   3   1   2   1   1   1   3
##            709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725
## Grau..k_i.   1   1   1   2   1   1   1   1   1   3   2   1   1   1   1   1   2
```

```
##            726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742
## Grau..k_i.   1   5   3   1   1   1   1   1   3   3   1   1   1   1   1   1   2
##            743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759
## Grau..k_i.   1   1   1   1   1   1   2   1   1   2   1   1   3   1   4   1   3
##            760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776
## Grau..k_i.   3   1   1   4   1   1   4   2   1   1   1   1   4   1   2   4   1
##            777 778 779 780 781 782 783 784 785 786 787
## Grau..k_i.   1   1   1   3   1   4   1   3   1   1   1
```

Grau Médio

```
# Grau médio
cat("<k> =",mean(degree(rede,mode='all')))
```

```
## <k> = 3.041931
```

Distribuição dos Graus

```
# Distribuição de frequências do grau
knitr::kable(t(table(factor(degree(rede,mode='all'), levels = 1:max(degree(rede,mode='all'))))))
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
309	156	101	63	41	26	20	21	14	12	8	8	3	1	0	2	2

Extra - Representação dos nodos proporcional ao grau

```
# Representação dos nodos proporcional ao grau
plot(rede, vertex.size=degree(rede,mode="all")*2, vertex.color= "orange",
     vertex.label=NA, vertex.frame.color=NA, edge.width= 2)
```

Rede Conexa ou Não Conexa?

```
# Verificar se a rede é conexa
is_connected <- is.connected(rede)
cat("A rede é conexa?", is_connected, "\n")
```

```
## A rede é conexa? FALSE
```

```
if (!is_connected) {
  # Se a rede não for conexa, calcular o nº de componentes conexas e as dim. máx e mín dessas component
  components <- clusters(rede)
  num_components <- components$no
  min_size <- min(components$csize)
  max_size <- max(components$csize)
  cat("Número de componentes conexas:", num_components, "\n")
  cat("Dimensão mínima das componentes conexas:", min_size, "\n")
  cat("Dimensão máxima das componentes conexas:", max_size, "\n")
}
```

```
## Número de componentes conexas: 104
## Dimensão mínima das componentes conexas: 2
## Dimensão máxima das componentes conexas: 496
```

Associação de Grau

```
# Coeficiente de Assortatividade - Coeficiente de correlação de Pearson para os graus dos nodos adjacen
assortativity_degree(rede)
```

```
## [1] 0.4765607
```

```
# Determina os valores da função knn - Média do grau dos vizinhos
knn(rede)$knnk
```

```
##  [1] 2.763754 3.429487 4.435644 5.067460 5.580488 6.089744 6.764286 7.946429
##  [9] 7.746032 8.275000 7.943182 8.270833 8.230769 7.857143      NaN 7.906250
## [17] 9.264706
```

Média dos Comprimentos dos Caminhos + Curtos

```
# Calcular a média dos caminhos mais curtos pela fórmula
caminhos_mais_curto <- distances(rede, algorithm = 'dijkstra')
soma_inversos <- sum(1 / caminhos_mais_curto[caminhos_mais_curto > 0])
((2 * soma_inversos) / (vcount(rede) * (vcount(rede) - 1)))^(-1)
```

```
## [1] 7.995324
```

```
# Média da Distância - Ignora as distâncias de caminhos inexistentes.
mean_distance(rede)
```

```
## [1] 7.914034
```

Diâmetro da Rede

```
diameter(rede)
```

```
## [1] 21
```

Estudo dos Triângulos da Rede

```
# Determinar o nº de triângulos da rede
sum(count_triangles(rede))
```

```
## [1] 2307
```

```
# Determinar a média do coeficiente de clustering para nodos com grau > 1
mean(transitivity(rede, type = "local")[(degree(rede) > 1)], na.rm = TRUE)
```

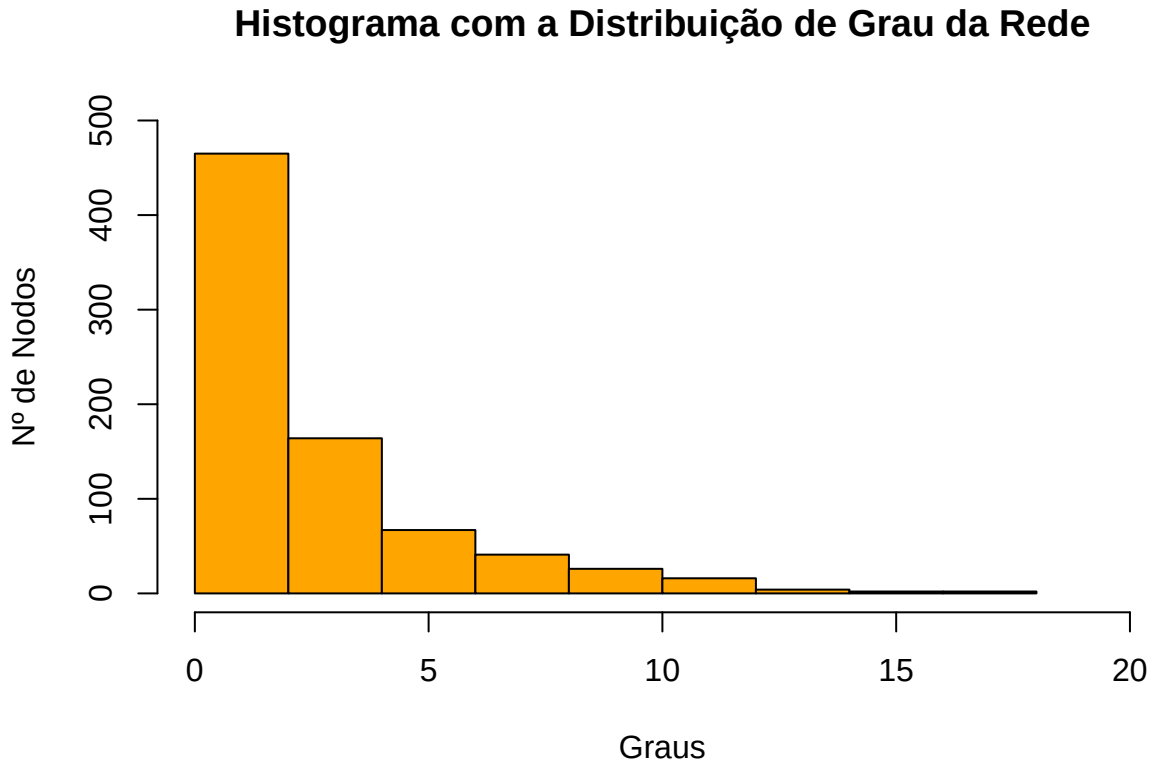
```
## [1] 0.4412442
```

```r
# Determinar o coeficiente de clustering da rede
transitivity(rede,type="global")
```

```
## [1] 0.4199126
```

Parâmetro de Heterogenidade

```r
deg <- degree(rede,mode="total")
hist(deg, col = 'orange',
     main = 'Histograma com a Distribuição de Grau da Rede',
     xlab = 'Graus',
     ylab = 'Nº de Nodos',
     ylim = c(0,500),
     xlim= c(0,20))
```




```r
ht <- mean(deg*deg)/(mean(deg)^2)
ht
```

```
## [1] 1.837585
```

Estudo dos *Hubs* da Rede

```
# closeness(rede)        # Centralidade de proximidade

# centr_clo(rede)        # Normalização

# betweenness(rede)      # Centralidade

# edge.betweenness(rede) # Intermediação de Ligações
```

Decomposição da Rede (Core Decomposition)

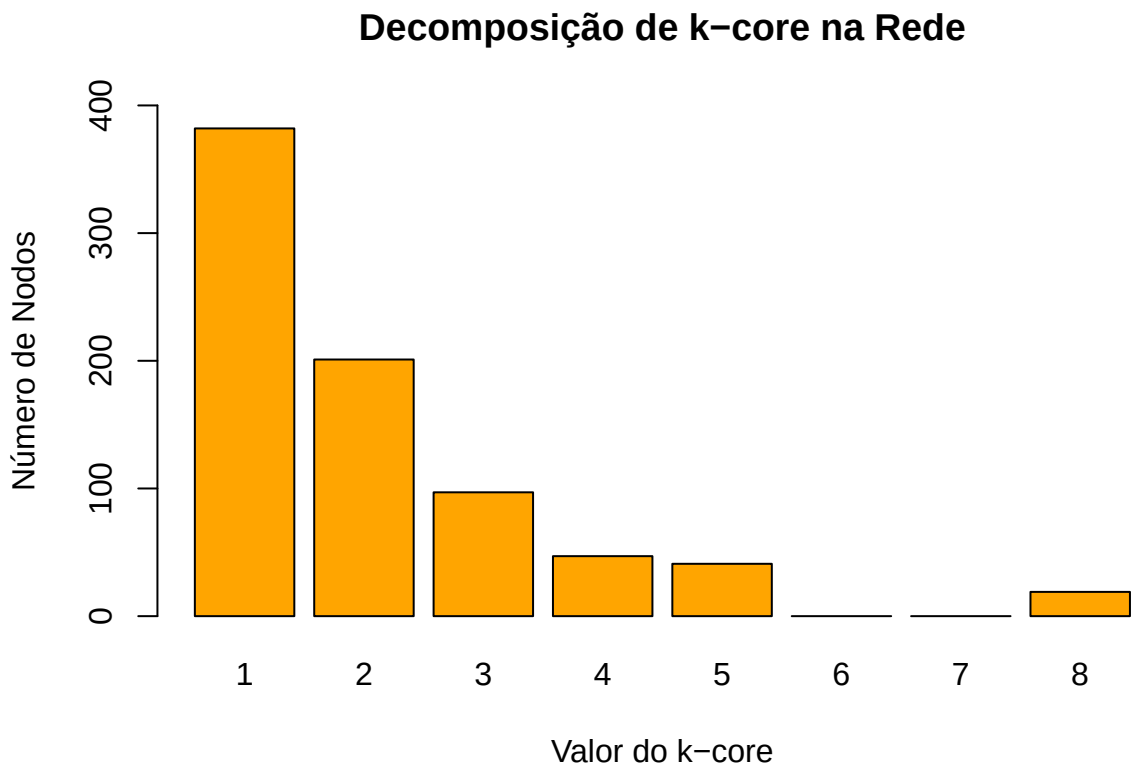
```
# Decomposição de core da rede
coreness <- coreness(rede, mode="all")                  # Escreve num vector o valor do k-core de cada
cat("Conchas (shells) na rede:")
```

```
## Conchas (shells) na rede:
```

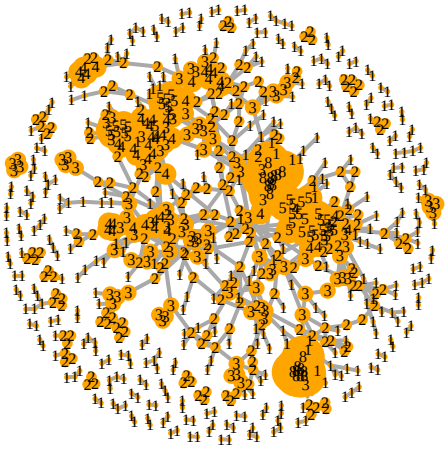
```
knitr::kable(t(table(factor(coreness, levels = 1:8))))
```

1	2	3	4	5	6	7	8
382	201	97	47	41	0	0	19

```
# Representação gráfica do vector kc
barplot(table(factor(coreness, levels = 1:8)),
        col = 'orange',
        main = "Decomposição de k-core na Rede",
        xlab = "Valor do k-core",
        ylab = "Número de Nodos",
        ylim = c(0,400))
```




```
# Representa os nodos com indicação do core e em tamanho proporcional ao valor de kc.
plot(rede,
     vertex.size=coreness*10/4,
     vertex.label=coreness,
     vertex.label.color=c("black"),
     vertex.label.cex = 0.5,
     vertex.color= "orange",
     vertex.frame.color=NA,
     edge.width= 2)
```



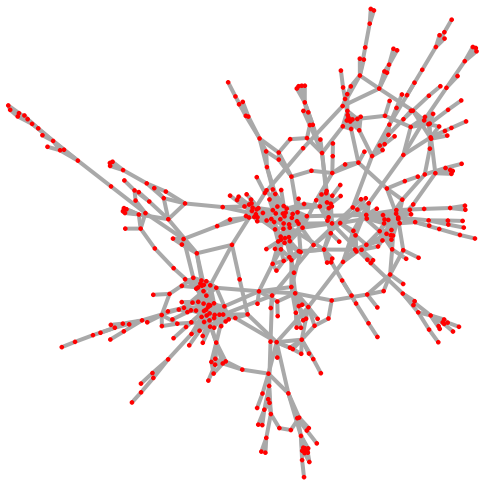
Questão 2

Componente Gigante

```
# components(rede) # A componente gigante é a 1ª do csize

# Encontrar a componente gigante da rede
giant_component <- induced.subgraph(rede, which(components$membership == which.max(components$csize)))

# Representar graficamente a componente gigante da rede
plot(giant_component,
     vertex.color= "red",
     vertex.label=NA,
     vertex.size=2,
     vertex.frame.color=NA,
     edge.width= 2)
```



Nº de Nodos e de Ligações da Subrede Componente Gigante da Rede

```
cat("Número de nós na componente gigante:", vcount(giant_component), "\n")
```

```
## Número de nós na componente gigante: 496
```

```r
cat("Número de ligações na componente gigante:", ecount(giant_component), "\n")
```

```
## Número de ligações na componente gigante: 984
```

Densidade da Componente Gigante

```r
edge_density(giant_component)
```

```
## [1] 0.00801564
```

Graus dos Nodos da Componente Gigante

```r
# Grau de cada nó
grau_cg <- data.frame("Grau (k_i)"=degree(giant_component,mode='all'))
grau_transposto_cg <- as.data.frame(t(grau_cg))
colnames(grau_transposto_cg) <- 1:vcount(giant_component) # Cada coluna é um nodo
grau_transposto_cg
```

```
##            1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
## Grau..k_i. 9 4 4 7 1 3 4 6 4  4  3  1  1  2  3  1  5  5  1  5  1  2  6  2  5  4
##            27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
## Grau..k_i.  2  2  1  2  1  1  7  6  1  8  6  3  1  5  3  3  1  4  3  4  1  1  4
##            50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
## Grau..k_i.  2  5  1  3  1  4  2  2  3 11  2  2  3  1  2  5  4  2  1  5 12  4  2
##            73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
## Grau..k_i.  2  1  1  1  1 10  1  6  1  2  5  1  1  1  1  2  1  3  2  4  8  5 12
##            96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113
## Grau..k_i.  8  7 10 10  11  10   9   4  17   6   8  12  13   4   3   6   3   8
##            114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130
## Grau..k_i.   7  13   9   7   4   4   4   2   1   5   3   8  13   8   6   7  10
##            131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
## Grau..k_i.  17  11   9   9   4   6   4   7   5   1   4   1   5   3   9   6   6
##            148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164
## Grau..k_i.   4   1   2   1   5   3   7  10   5   4   3   5  10   5   6   5   2
##            165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181
## Grau..k_i.   6   6   8   8   8   8   8   9   2   8  10   5   9   7   1   7   4
##            182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
## Grau..k_i.   5   3   6   6   4   7   2  12   5   6   3  11   6   7   2   3   6
##            199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
## Grau..k_i.   3   8  12  10   9   3   2  12   7   2   2   8   4   2  10   7   9
##            216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232
## Grau..k_i.  11   8   5   6   5   1   8   4   2  16   1   3   1   3   1   3   4
##            233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249
## Grau..k_i.   1   2   2   8   8   5   9   5   2   9   2   2   7   2   3   3   1
##            250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266
## Grau..k_i.   6   7   3   1   1   2   4  11   4   5   1   1  10   3   3   4   3
##            267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283
## Grau..k_i.   8   3   4   4   1   1   4  12   2   3   3   2   2   3   1  11   2
```

```
##            284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
## Grau..k_i.   2  14  12   1   2  16   4   6   6   9   5   4   5   1  11   5   1
##            301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317
## Grau..k_i.   1   4   4   5   3   2   4   4   2   4   1   2   2   1   4   9   2
##            318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334
## Grau..k_i.   1  10   7   3   3   3   3   3   2   5   4   1   7   3   3   5   1
##            335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351
## Grau..k_i.   2   1   4   4   1   3   3   2   2   6   4   3   5   4   1   7   1
##            352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368
## Grau..k_i.   2   1   2   2   2   3   1   3   5   6   2   3   3   2   2   4   5
##            369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385
## Grau..k_i.   2   4   3   6   1   2   4   3   2   1   2   3   3   3   1   3   5
##            386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402
## Grau..k_i.   2   1   2   2   1   4   1   1   1   3   2   2   3   1   3   2   4
##            403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419
## Grau..k_i.   2   2   1   2   2   1   3   3   1   1   3   7   3   2   2   1   1
##            420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436
## Grau..k_i.   3   1   5   1   4   3   1   2   1   2   1   1   1   1   2   8   3
##            437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453
## Grau..k_i.   1   2   1   1   2   1   3   1   1   1   3   2   3   4   1   1   1
##            454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470
## Grau..k_i.   2   3   2   1   1   1   3   2   1   5   3   3   3   2   1   1   1
##            471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487
## Grau..k_i.   1   3   1   4   3   3   4   1   4   2   1   1   4   1   2   4   1
##            488 489 490 491 492 493 494 495 496
## Grau..k_i.   1   1   3   1   4   1   3   1   1
```

Grau Médio da Componente Gigante

```
# Grau médio
cat("<k> =",mean(degree(giant_component,mode='all')))
```

```
## <k> = 3.967742
```

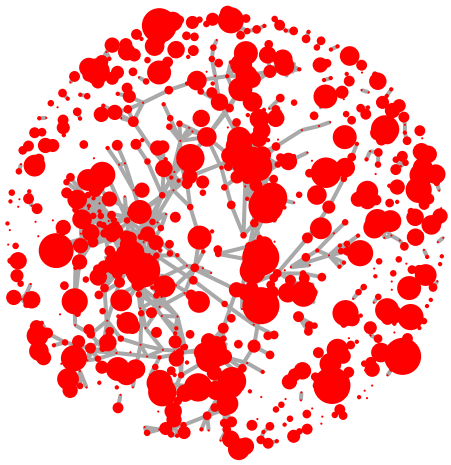
Distribuição dos Graus da Componente Gigante

```
# Distribuição de frequências do grau.
knitr::kable(t(table(factor(degree(giant_component,mode='all'),
                            levels = 1:max(degree(giant_component,mode='all'))))))
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
116	86	79	60	38	26	20	21	14	12	8	8	3	1	0	2	2

Extra - Representação dos nodos proporcional ao grau da Componente Gigante

```r
# Representação dos nodos proporcional ao grau
plot(rede,
     vertex.size=degree(giant_component,mode="all")*1,
     vertex.color= "red",
     vertex.label=NA,
     vertex.frame.color=NA,
     edge.width= 2)
```



Associação de Grau da Componente Gigante

```r
# Coeficiente de Assortatividade - Coeficiente de correlação de Pearson para os graus dos nodos adjacen
assortativity_degree(giant_component)
```

```
## [1] 0.345145
```

```r
#  Determina os valores da função knn - Média do grau dos vizinhos
knn(giant_component)$knnk
```

```
##  [1] 4.913793 4.616279 4.953586 5.195833 5.842105 6.089744 6.764286 7.946429
##  [9] 7.746032 8.275000 7.943182 8.270833 8.230769 7.857143      NaN 7.906250
## [17] 9.264706
```

Média dos Comprimentos dos Caminhos + Curtos da Componente Gigante

```
# Média da Distância - Ignora as distâncias de caminhos inexistentes.
mean_distance(giant_component)
```

```
## [1] 7.933447
```

Diâmetro da rede da Componente Gigante

```
diameter(giant_component)
```

```
## [1] 21
```

Estudo dos Triângulos da rede

```
# Determinar o nº de triângulos da componente gigante
sum(count_triangles(giant_component))
```

```
## [1] 2229
```

```
# Determinar a média do coeficiente de clustering para nodos com grau > 1 da componente gigante
mean(transitivity(giant_component, type = "local")[(degree(giant_component) > 1)], na.rm = TRUE)
```

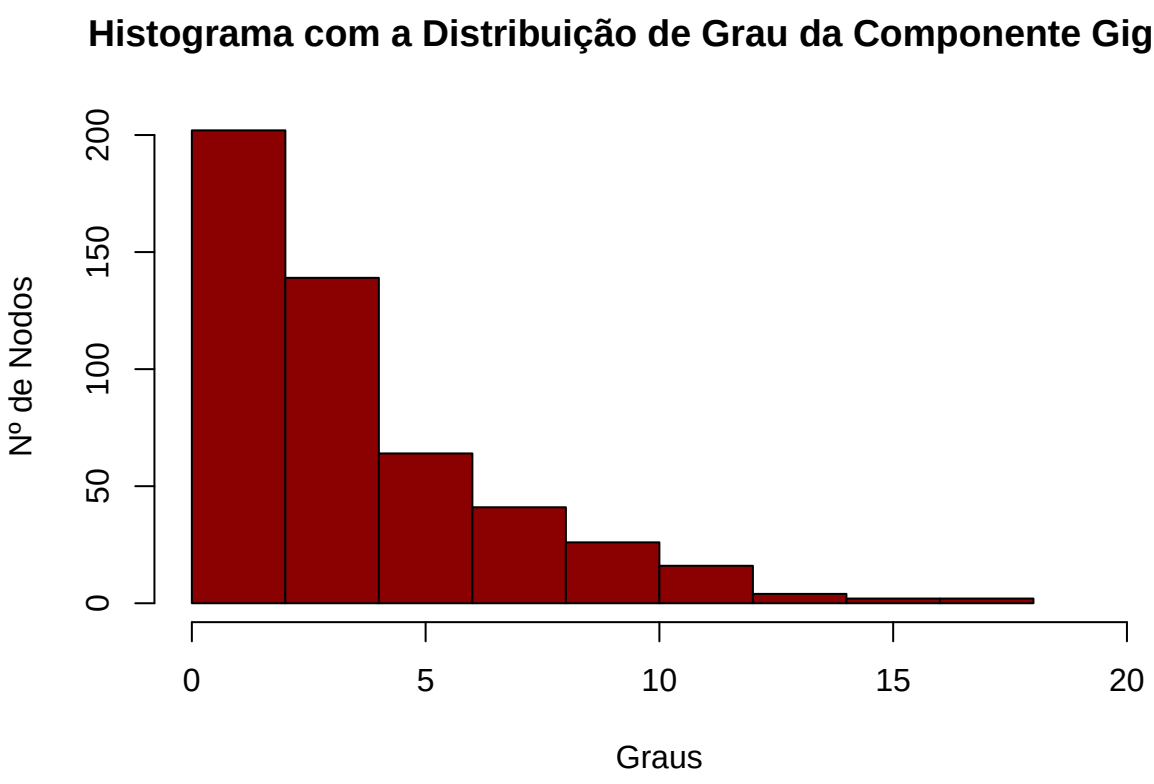
```
## [1] 0.4391616
```

```
# Determinar o coeficiente de clustering da componente gigante
transitivity(giant_component,type="global")
```

```
## [1] 0.419774
```

Parâmetro de Heterogenidade da Componente Gigante

```
deg_giant <- degree(giant_component, mode="total")
hist(deg_giant,
     col = 'darkred',
     main = 'Histograma com a Distribuição de Grau da Componente Gigante',
     xlab = 'Graus',
     ylab = 'Nº de Nodos',
     xlim = c(0,20))
```




```
ht_giant <- mean(deg_giant*deg_giant)/(mean(deg_giant)^2)
ht_giant
```

```
## [1] 1.612086
```

Estudo dos *Hubs* da Componente Gigante

```
# closeness(giant_component)        # Centralidade de proximidade

# centr_clo(giant_component)        # Normalização

# betweenness(giant_component)      # Centralidade

# edge.betweenness(giant_component) # Intermediação de Ligações
```

Decomposição (*Core Decomposition*) da Componente Gigante

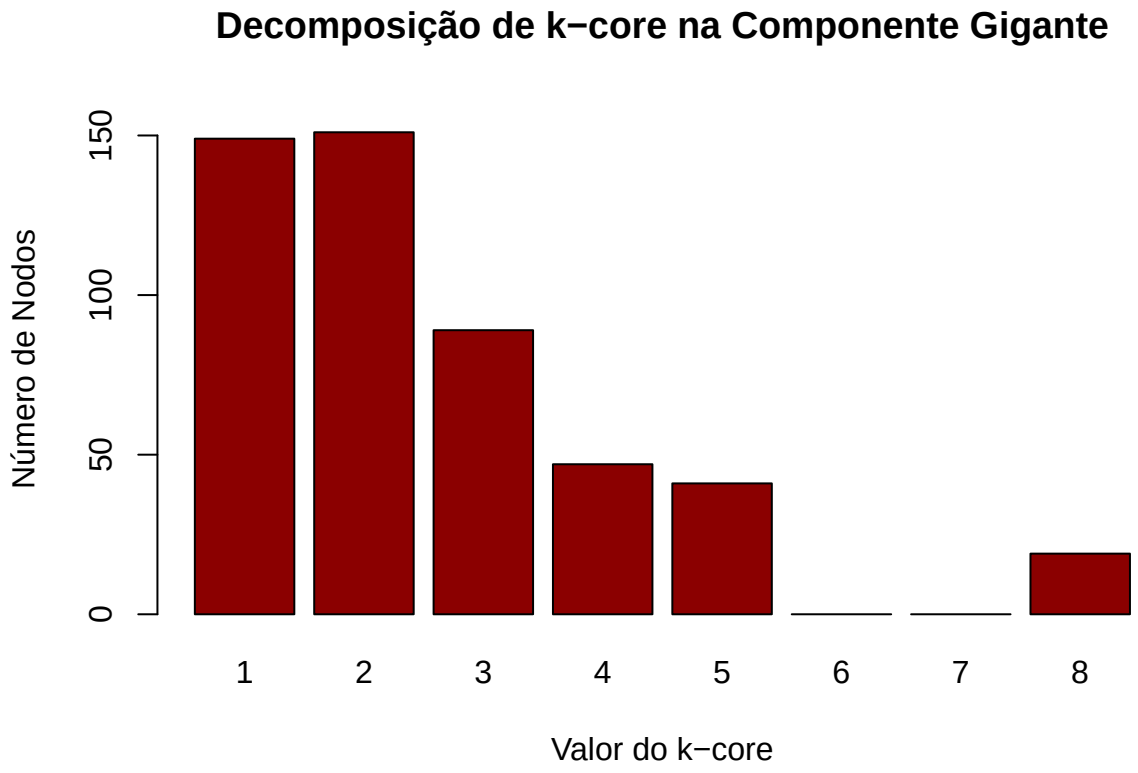
```
# Decomposição de core da componente gigante
coreness_giant <- coreness(giant_component, mode="all")   # Escreve num vector o valor do k-core de cad
cat("Conchas (shells) na componente gigante:")
```

```
## Conchas (shells) na componente gigante:
```

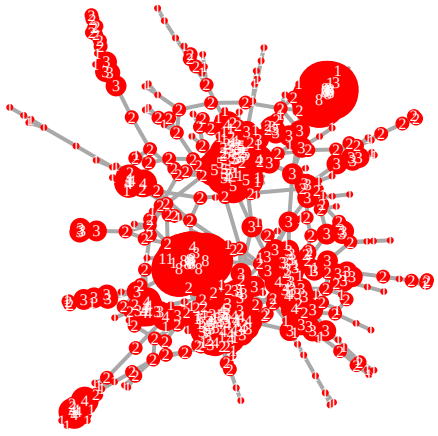
```r
knitr::kable(t(table(factor(coreness_giant, levels = 1:max(coreness_giant)))))
```

1	2	3	4	5	6	7	8
149	151	89	47	41	0	0	19

```r
# Representação gráfica do vector kc da Componente Gigante
barplot(table(factor(coreness_giant, levels = 1:max(coreness_giant))),
        col = 'darkred',
        main = "Decomposição de k-core na Componente Gigante",
        xlab = "Valor do k-core",
        ylab = "Número de Nodos",
        ylim=c(0,160))
```




```r
# Representa os nodos com indicação do core e em tamanho proporcional ao valor de kc.
plot(giant_component,
     vertex.size=coreness_giant*10/3,
     vertex.label=coreness_giant,
     vertex.label.color=c("white"),
     vertex.label.cex = 0.5,
     vertex.color= "red",
     vertex.frame.color=NA, edge.width= 2)
```



Questão 3

```
# Q3: Comparação entre a rede completa e a componente gigante

# Criar um dataframe comparativo dos resultados
df <- data.frame(
  Nodos = c('**Nº de Nodos *(***$N$ ***)***', vcount(rede), vcount(giant_component)),
  Ligacoes = c('**Nº de Ligações *(***$L$***)***', ecount(rede), ecount(giant_component)),
  Densidade = c('**Densidade *(***$d$ ***)***', round(edge_density(rede), 4),
                round(edge_density(giant_component), 4)),
  Grau_Medio = c('**Grau Médio *(***$<k>$***)***',
                 round(mean(degree(rede, mode = "all")), 4),
                 round(mean(degree(giant_component, mode = "all")), 4)),
  Coeficiente_de_Pearson = c('**Coeficiente de Pearson *(***$p$***)***',
                             round(assortativity_degree(rede), 4),
                             round(assortativity_degree(giant_component), 4)),
  Media_Distancia = c('**Distância Média C+c*(***$<l>$***)***',
                      round(mean_distance(rede), 3),
                      round(mean_distance(giant_component), 3)),
  Diametro = c('**Diâmetro *(***$l_{max}$***)***',
               diameter(rede), diameter(giant_component)),
  Triangulos = c('**Nº de Triângulos**', sum(count_triangles(rede)),
                 sum(count_triangles(giant_component))),

  # Coeficiente de Cluster médio dos nodos (<C>)
  Coef_custering_medio = c('**Coeficiente de Cluster médio dos nodos *(***$<C>$***)***',
                           round(mean(
                             transitivity(rede,
                                          type = "local")[(degree(rede) > 1)],
                             na.rm = TRUE), 4),
                           round(mean(
                             transitivity(giant_component,
                                          type = "local")[(degree(giant_component) > 1)],
                             na.rm = TRUE), 4)),

  # Coeficiente de Cluster da rede (C)
  Coef_custering_rede = c('**Coeficiente de Cluster da (sub)rede *(***$C$***)***',
                          round(transitivity(rede, type="global"), 4),
                          round(transitivity(giant_component, type="global"), 4)),
  Heterogeneidade = c('**Heterogeneidade *(***$K$***)***', round(ht, 2), round(ht_giant, 2)),
  N_Conchas = c('**Nº de Conchas**', max(coreness(rede, mode="all")),
                max(coreness(giant_component, mode="all")))

)

# Criar a tabela flextable
ftable <- flextable(as.data.frame(t(df))) %>% colformat_md()

# Personalizar a tabela
set_flextable_defaults(fonts_ignore=TRUE)
ftable <- border_remove(x = ftable) %>%
  hline(i= 1, part = "header", border = fp_border(color = "gray", width = 2)) %>%
  hline_bottom(part = "body", border = fp_border(color = "grey", width = 1)) %>%
```

```
  vline(j=1:2, border = fp_border(color = "white", width = 5)) %>%
  align(j= 2:3, align = "center", part = "all") %>%
  bg(j = 2, bg = "#ED7D31", part = "header") %>%
  bg(j = 3, bg = "darkred", part = "header") %>%
  color(j = 2:3, color = "white", part = "header") %>%
  set_header_labels(V1 = "", V2 = "Rede Completa", V3 = "Componente Gigante") %>%
  bold(bold = TRUE, part = "header") %>%
  autofit()
ftable
```
