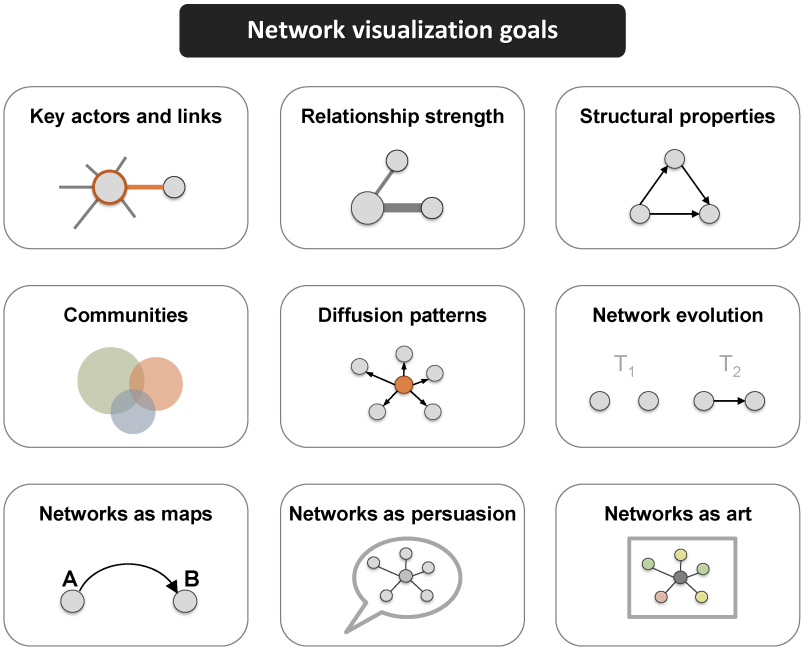
Inleiding tot visualiseren van netwerkdata in R

Robin Khalfa

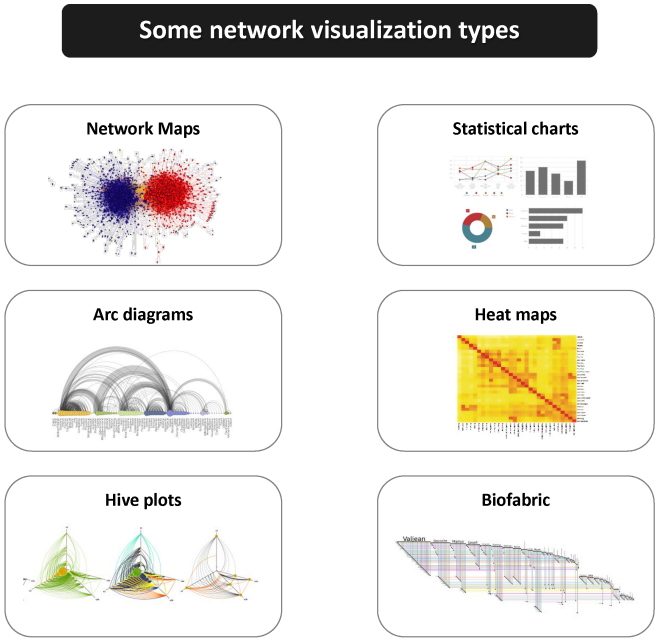
2025-05-23

1. Inleiding

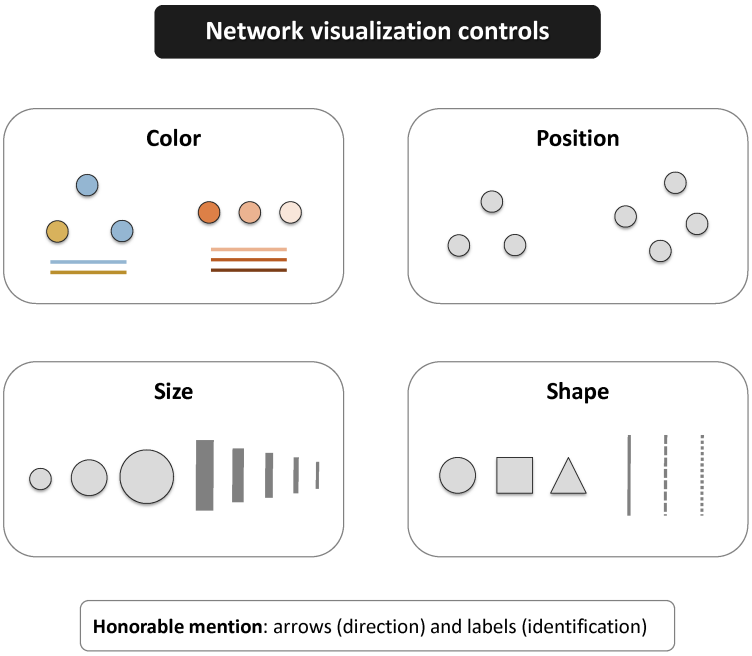
Visualisatie vormt een fundamenteel onderdeel van de sociale netwerkanalyse. Het effectief ontwerpen van een netwerkvisualisatie vereist een duidelijke doelstelling: welke structurele eigenschappen willen we onder de aandacht brengen? Welke onderzoeksvragen of patronen willen we blootleggen? Het antwoord op deze vragen bepaalt in grote mate de vorm en inhoud van de gekozen visualisatie.



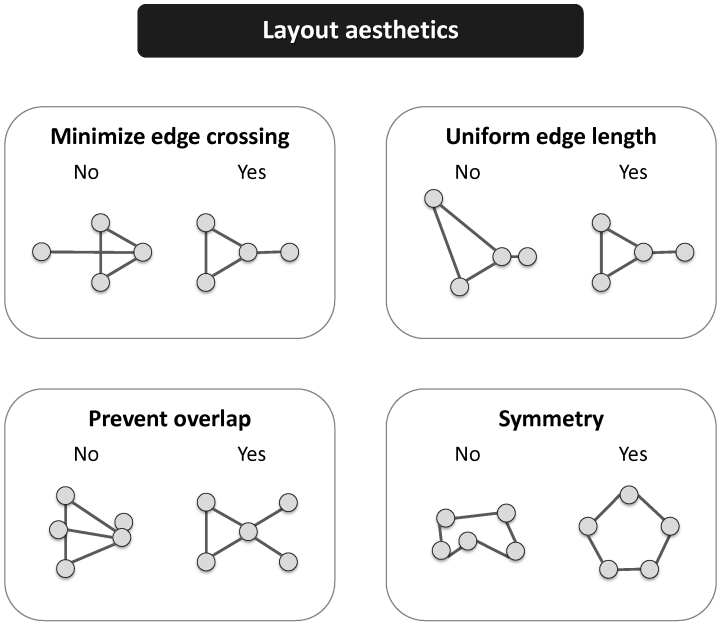
Hoewel netwerkdiagrammen (zogenaamde network maps) vaak als de standaard worden beschouwd voor het visualiseren van grafen, zijn ze zeker niet de enige mogelijkheid. Afhankelijk van de onderzoeksvraag kunnen alternatieve representatievormen, zoals matrixplots, eenvoudige statistische grafieken of frequentietabellen van netwerkkarakteristieken, beter geschikt zijn. Niet alle netwerkstructuren lenen zich immers tot een betekenisvolle ruimtelijke voorstelling.



Indien men toch kiest voor een netwerkkaart, spelen visuele parameters zoals kleur, grootte, vorm en positie van de knopen een centrale rol. Deze elementen helpen om structurele kenmerken zoals centraliteit, clustervorming of groepslidmaatschap visueel te onderscheiden.



Bovendien zijn moderne grafische layout-algoritmen geoptimaliseerd voor snelheid en esthetiek: ze minimaliseren het aantal overlappende elementen en kruisingen tussen randen, en streven naar een evenwichtige spreiding van de knopen, met gelijkaardige afstanden tussen verbindingen.



R biedt met packages zoals `igraph`, `sna` en `ggraph` een breed scala aan mogelijkheden om netwerkvisualisaties te genereren, aan te passen en te analyseren. In dit onderdeel demonstreren we hoe deze tools ingezet kunnen worden om netwerken op een inzichtelijke en visueel coherente wijze te presenteren.

In wat volgt demonstreren we enkele leuke manieren waarop visualisaties tot stand kunnen worden gebracht in R en hebben we aandacht voor verschillende packages en de meest belangrijke parameters die daarbij centraal staan.

Deze notebook is als volgt opgebouwd:

1. **Kleuren in R plots**
2. **igraph**
3. **ggplot**
4. **ggraph**

We laden alvast de belangrijkste packages in:

```
# Instellen van de node attributes
#install.packages("igraph")
#install.packages("network")
#install.packages("sna")
#install.packages("ggraph")
#install.packages("visNetwork")
#install.packages("threejs")
#install.packages("networkD3")
#install.packages("ndtv")
```

2. Kleuren in R plots

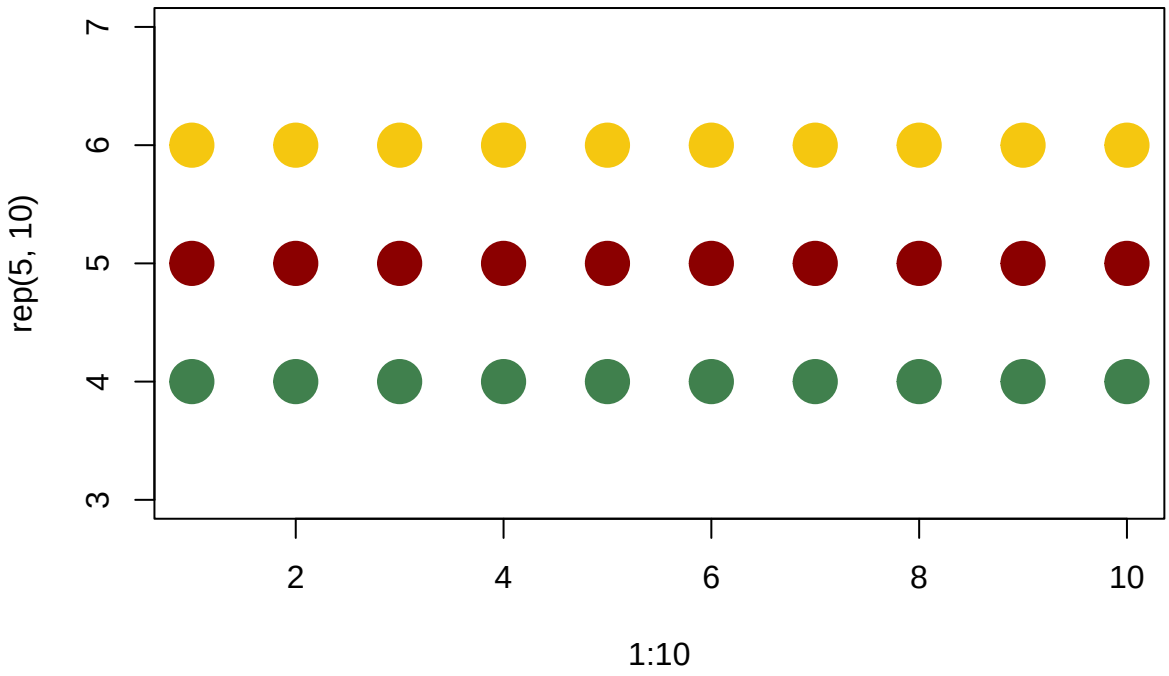
Kleuren zijn niet enkel esthetisch aantrekkelijk, ze spelen ook een cruciale rol in het visueel onderscheiden van objecttypes of attributenniveaus. In de meeste R-functies kunnen kleuren gespecificeerd worden via hun

naam, hexadecimale waarde of RGB-waarden.

In onderstaande eenvoudige base R-plot wordt gebruikgemaakt van:

- `x` en `y`: de coördinaten van het punt
- `pch`: de vorm van het symbool
- `cex`: de grootte van het punt
- `col`: de kleur van het punt

```
plot(x=1:10, y=rep(5,10), pch=19, cex=3, col="dark red")
points(x=1:10, y=rep(6, 10), pch=19, cex=3, col="557799")
points(x=1:10, y=rep(4, 10), pch=19, cex=3, col=rgb(.25, .5, .3))
```

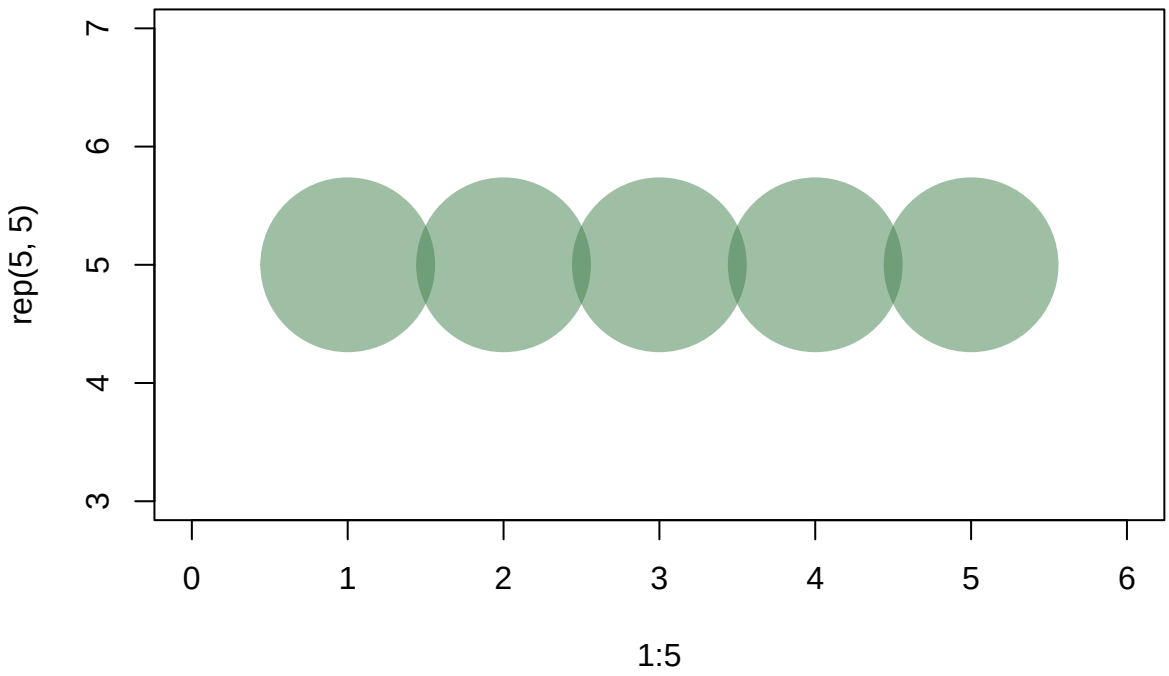


Merk op dat RGB-waarden in R standaard een schaal van 0 tot 1 hanteren. Wil je werken met het meer traditionele 0–255 bereik, dan gebruik je:

```
#rgb(10, 100, 100, maxColorValue=255)
```

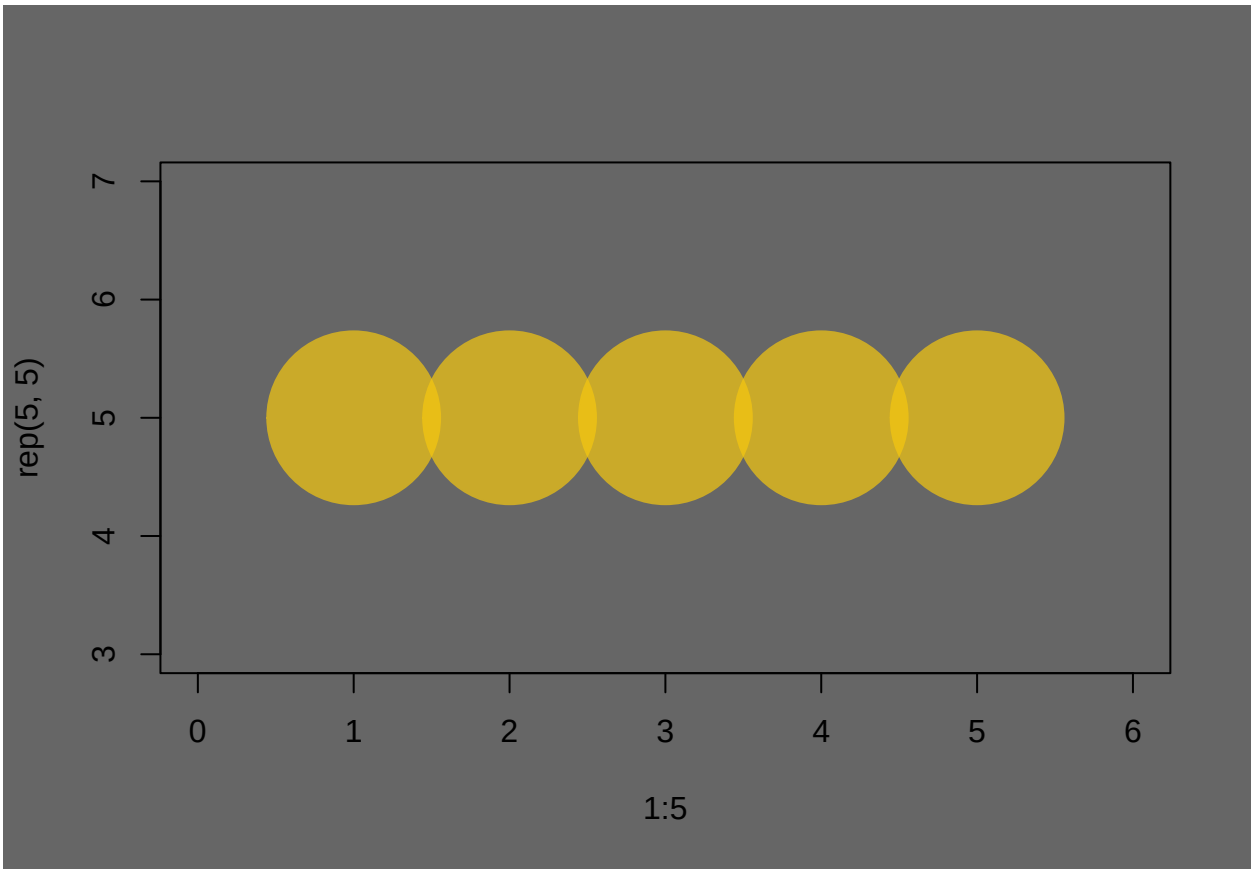
De parameter alpha (tussen 0 en 1) bepaalt verder de transparantie van een element:

```
plot(x=1:5, y=rep(5,5), pch=19, cex=12, col=rgb(.25, .5, .3, alpha=.5), xlim=c(0,6))
```



Heb je een hexkleur en wil je transparantie toevoegen, gebruik dan `adjustcolor` uit het `grDevices`-package. Je kan ook de achtergrondkleur van de plot aanpassen met `par(bg=...)`:

```
par(bg="gray40")
col.tr <- grDevices::adjustcolor("557799", alpha=0.7)
plot(x=1:5, y=rep(5,5), pch=19, cex=12, col=col.tr, xlim=c(0,6))
```



Andere nuttige instellingen via `par()` zijn:

- `mar`: marges van de plot (bv. `par(mar=c(5, 4, 4, 2))`)
- `new=TRUE`: voeg toe aan bestaande plot i.p.v. te overschrijven

Via deze functie krijg je zicht op alle kleuren in base R:

```
colors()                              # lijst alle kleurnamen op
```

```
##   [1] "white"                "aliceblue"            "antiquewhite"
##   [4] "antiquewhite1"        "antiquewhite2"        "antiquewhite3"
##   [7] "antiquewhite4"        "aquamarine"           "aquamarine1"
##  [10] "aquamarine2"          "aquamarine3"          "aquamarine4"
##  [13] "azure"                "azure1"               "azure2"
##  [16] "azure3"               "azure4"               "beige"
##  [19] "bisque"               "bisque1"              "bisque2"
##  [22] "bisque3"              "bisque4"              "black"
##  [25] "blanchedalmond"       "blue"                 "blue1"
##  [28] "blue2"                "blue3"                "blue4"
##  [31] "blueviolet"           "brown"                "brown1"
##  [34] "brown2"               "brown3"               "brown4"
##  [37] "burlywood"            "burlywood1"           "burlywood2"
##  [40] "burlywood3"           "burlywood4"           "cadetblue"
##  [43] "cadetblue1"           "cadetblue2"           "cadetblue3"
##  [46] "cadetblue4"           "chartreuse"           "chartreuse1"
```

```
##  [49] "chartreuse2"          "chartreuse3"          "chartreuse4"
##  [52] "chocolate"            "chocolate1"           "chocolate2"
##  [55] "chocolate3"           "chocolate4"           "coral"
##  [58] "coral1"               "coral2"               "coral3"
##  [61] "coral4"               "cornflowerblue"       "cornsilk"
##  [64] "cornsilk1"            "cornsilk2"            "cornsilk3"
##  [67] "cornsilk4"            "cyan"                 "cyan1"
##  [70] "cyan2"                "cyan3"                "cyan4"
##  [73] "darkblue"             "darkcyan"             "darkgoldenrod"
##  [76] "darkgoldenrod1"       "darkgoldenrod2"       "darkgoldenrod3"
##  [79] "darkgoldenrod4"       "darkgray"             "darkgreen"
##  [82] "darkgrey"             "darkkhaki"            "darkmagenta"
##  [85] "darkolivegreen"       "darkolivegreen1"      "darkolivegreen2"
##  [88] "darkolivegreen3"      "darkolivegreen4"      "darkorange"
##  [91] "darkorange1"          "darkorange2"          "darkorange3"
##  [94] "darkorange4"          "darkorchid"           "darkorchid1"
##  [97] "darkorchid2"          "darkorchid3"          "darkorchid4"
## [100] "darkred"              "darksalmon"           "darkseagreen"
## [103] "darkseagreen1"        "darkseagreen2"        "darkseagreen3"
## [106] "darkseagreen4"        "darkslateblue"        "darkslategray"
## [109] "darkslategray1"       "darkslategray2"       "darkslategray3"
## [112] "darkslategray4"       "darkslategrey"        "darkturquoise"
## [115] "darkviolet"           "deeppink"             "deeppink1"
## [118] "deeppink2"            "deeppink3"            "deeppink4"
## [121] "deepskyblue"          "deepskyblue1"         "deepskyblue2"
## [124] "deepskyblue3"         "deepskyblue4"         "dimgray"
## [127] "dimgrey"              "dodgerblue"           "dodgerblue1"
## [130] "dodgerblue2"          "dodgerblue3"          "dodgerblue4"
## [133] "firebrick"            "firebrick1"           "firebrick2"
## [136] "firebrick3"           "firebrick4"           "floralwhite"
## [139] "forestgreen"          "gainsboro"            "ghostwhite"
## [142] "gold"                 "gold1"                "gold2"
## [145] "gold3"                "gold4"                "goldenrod"
## [148] "goldenrod1"           "goldenrod2"           "goldenrod3"
## [151] "goldenrod4"           "gray"                 "gray0"
## [154] "gray1"                "gray2"                "gray3"
## [157] "gray4"                "gray5"                "gray6"
## [160] "gray7"                "gray8"                "gray9"
## [163] "gray10"               "gray11"               "gray12"
## [166] "gray13"               "gray14"               "gray15"
## [169] "gray16"               "gray17"               "gray18"
## [172] "gray19"               "gray20"               "gray21"
## [175] "gray22"               "gray23"               "gray24"
## [178] "gray25"               "gray26"               "gray27"
## [181] "gray28"               "gray29"               "gray30"
## [184] "gray31"               "gray32"               "gray33"
## [187] "gray34"               "gray35"               "gray36"
## [190] "gray37"               "gray38"               "gray39"
## [193] "gray40"               "gray41"               "gray42"
## [196] "gray43"               "gray44"               "gray45"
## [199] "gray46"               "gray47"               "gray48"
## [202] "gray49"               "gray50"               "gray51"
## [205] "gray52"               "gray53"               "gray54"
## [208] "gray55"               "gray56"               "gray57"
```

```
## [211] "gray58"          "gray59"          "gray60"
## [214] "gray61"          "gray62"          "gray63"
## [217] "gray64"          "gray65"          "gray66"
## [220] "gray67"          "gray68"          "gray69"
## [223] "gray70"          "gray71"          "gray72"
## [226] "gray73"          "gray74"          "gray75"
## [229] "gray76"          "gray77"          "gray78"
## [232] "gray79"          "gray80"          "gray81"
## [235] "gray82"          "gray83"          "gray84"
## [238] "gray85"          "gray86"          "gray87"
## [241] "gray88"          "gray89"          "gray90"
## [244] "gray91"          "gray92"          "gray93"
## [247] "gray94"          "gray95"          "gray96"
## [250] "gray97"          "gray98"          "gray99"
## [253] "gray100"         "green"           "green1"
## [256] "green2"          "green3"          "green4"
## [259] "greenyellow"     "grey"            "grey0"
## [262] "grey1"           "grey2"           "grey3"
## [265] "grey4"           "grey5"           "grey6"
## [268] "grey7"           "grey8"           "grey9"
## [271] "grey10"          "grey11"          "grey12"
## [274] "grey13"          "grey14"          "grey15"
## [277] "grey16"          "grey17"          "grey18"
## [280] "grey19"          "grey20"          "grey21"
## [283] "grey22"          "grey23"          "grey24"
## [286] "grey25"          "grey26"          "grey27"
## [289] "grey28"          "grey29"          "grey30"
## [292] "grey31"          "grey32"          "grey33"
## [295] "grey34"          "grey35"          "grey36"
## [298] "grey37"          "grey38"          "grey39"
## [301] "grey40"          "grey41"          "grey42"
## [304] "grey43"          "grey44"          "grey45"
## [307] "grey46"          "grey47"          "grey48"
## [310] "grey49"          "grey50"          "grey51"
## [313] "grey52"          "grey53"          "grey54"
## [316] "grey55"          "grey56"          "grey57"
## [319] "grey58"          "grey59"          "grey60"
## [322] "grey61"          "grey62"          "grey63"
## [325] "grey64"          "grey65"          "grey66"
## [328] "grey67"          "grey68"          "grey69"
## [331] "grey70"          "grey71"          "grey72"
## [334] "grey73"          "grey74"          "grey75"
## [337] "grey76"          "grey77"          "grey78"
## [340] "grey79"          "grey80"          "grey81"
## [343] "grey82"          "grey83"          "grey84"
## [346] "grey85"          "grey86"          "grey87"
## [349] "grey88"          "grey89"          "grey90"
## [352] "grey91"          "grey92"          "grey93"
## [355] "grey94"          "grey95"          "grey96"
## [358] "grey97"          "grey98"          "grey99"
## [361] "grey100"         "honeydew"        "honeydew1"
## [364] "honeydew2"       "honeydew3"       "honeydew4"
## [367] "hotpink"         "hotpink1"        "hotpink2"
## [370] "hotpink3"        "hotpink4"        "indianred"
```

```
## [373] "indianred1"           "indianred2"           "indianred3"
## [376] "indianred4"           "ivory"                "ivory1"
## [379] "ivory2"               "ivory3"               "ivory4"
## [382] "khaki"                "khaki1"               "khaki2"
## [385] "khaki3"               "khaki4"               "lavender"
## [388] "lavenderblush"        "lavenderblush1"       "lavenderblush2"
## [391] "lavenderblush3"       "lavenderblush4"       "lawngreen"
## [394] "lemonchiffon"         "lemonchiffon1"        "lemonchiffon2"
## [397] "lemonchiffon3"        "lemonchiffon4"        "lightblue"
## [400] "lightblue1"           "lightblue2"           "lightblue3"
## [403] "lightblue4"           "lightcoral"           "lightcyan"
## [406] "lightcyan1"           "lightcyan2"           "lightcyan3"
## [409] "lightcyan4"           "lightgoldenrod"       "lightgoldenrod1"
## [412] "lightgoldenrod2"      "lightgoldenrod3"      "lightgoldenrod4"
## [415] "lightgoldenrodyellow" "lightgray"            "lightgreen"
## [418] "lightgrey"            "lightpink"            "lightpink1"
## [421] "lightpink2"           "lightpink3"           "lightpink4"
## [424] "lightsalmon"          "lightsalmon1"         "lightsalmon2"
## [427] "lightsalmon3"         "lightsalmon4"         "lightseagreen"
## [430] "lightskyblue"         "lightskyblue1"        "lightskyblue2"
## [433] "lightskyblue3"        "lightskyblue4"        "lightslateblue"
## [436] "lightslategray"       "lightslategrey"       "lightsteelblue"
## [439] "lightsteelblue1"      "lightsteelblue2"      "lightsteelblue3"
## [442] "lightsteelblue4"      "lightyellow"          "lightyellow1"
## [445] "lightyellow2"         "lightyellow3"         "lightyellow4"
## [448] "limegreen"            "linen"                "magenta"
## [451] "magenta1"             "magenta2"             "magenta3"
## [454] "magenta4"             "maroon"               "maroon1"
## [457] "maroon2"              "maroon3"              "maroon4"
## [460] "mediumaquamarine"     "mediumblue"           "mediumorchid"
## [463] "mediumorchid1"        "mediumorchid2"        "mediumorchid3"
## [466] "mediumorchid4"        "mediumpurple"         "mediumpurple1"
## [469] "mediumpurple2"        "mediumpurple3"        "mediumpurple4"
## [472] "mediumseagreen"       "mediumslateblue"      "mediumspringgreen"
## [475] "mediumturquoise"      "mediumvioletred"      "midnightblue"
## [478] "mintcream"            "mistyrose"            "mistyrose1"
## [481] "mistyrose2"           "mistyrose3"           "mistyrose4"
## [484] "moccasin"             "navajowhite"          "navajowhite1"
## [487] "navajowhite2"         "navajowhite3"         "navajowhite4"
## [490] "navy"                 "navyblue"             "oldlace"
## [493] "olivedrab"            "olivedrab1"           "olivedrab2"
## [496] "olivedrab3"           "olivedrab4"           "orange"
## [499] "orange1"              "orange2"              "orange3"
## [502] "orange4"              "orangered"            "orangered1"
## [505] "orangered2"           "orangered3"           "orangered4"
## [508] "orchid"               "orchid1"              "orchid2"
## [511] "orchid3"              "orchid4"              "palegoldenrod"
## [514] "palegreen"            "palegreen1"           "palegreen2"
## [517] "palegreen3"           "palegreen4"           "paleturquoise"
## [520] "paleturquoise1"       "paleturquoise2"       "paleturquoise3"
## [523] "paleturquoise4"       "palevioletred"        "palevioletred1"
## [526] "palevioletred2"       "palevioletred3"       "palevioletred4"
## [529] "papayawhip"           "peachpuff"            "peachpuff1"
## [532] "peachpuff2"           "peachpuff3"           "peachpuff4"
```

```
## [535] "peru"                "pink"                "pink1"
## [538] "pink2"               "pink3"               "pink4"
## [541] "plum"                "plum1"               "plum2"
## [544] "plum3"               "plum4"               "powderblue"
## [547] "purple"              "purple1"             "purple2"
## [550] "purple3"             "purple4"             "red"
## [553] "red1"                "red2"                "red3"
## [556] "red4"                "rosybrown"           "rosybrown1"
## [559] "rosybrown2"          "rosybrown3"          "rosybrown4"
## [562] "royalblue"           "royalblue1"          "royalblue2"
## [565] "royalblue3"          "royalblue4"          "saddlebrown"
## [568] "salmon"              "salmon1"             "salmon2"
## [571] "salmon3"             "salmon4"             "sandybrown"
## [574] "seagreen"            "seagreen1"           "seagreen2"
## [577] "seagreen3"           "seagreen4"           "seashell"
## [580] "seashell1"           "seashell2"           "seashell3"
## [583] "seashell4"           "sienna"              "sienna1"
## [586] "sienna2"             "sienna3"             "sienna4"
## [589] "skyblue"             "skyblue1"            "skyblue2"
## [592] "skyblue3"            "skyblue4"            "slateblue"
## [595] "slateblue1"          "slateblue2"          "slateblue3"
## [598] "slateblue4"          "slategray"           "slategray1"
## [601] "slategray2"          "slategray3"          "slategray4"
## [604] "slategrey"           "snow"                "snow1"
## [607] "snow2"               "snow3"               "snow4"
## [610] "springgreen"         "springgreen1"        "springgreen2"
## [613] "springgreen3"        "springgreen4"        "steelblue"
## [616] "steelblue1"          "steelblue2"          "steelblue3"
## [619] "steelblue4"          "tan"                 "tan1"
## [622] "tan2"                "tan3"                "tan4"
## [625] "thistle"             "thistle1"            "thistle2"
## [628] "thistle3"            "thistle4"            "tomato"
## [631] "tomato1"             "tomato2"             "tomato3"
## [634] "tomato4"             "turquoise"           "turquoise1"
## [637] "turquoise2"          "turquoise3"          "turquoise4"
## [640] "violet"              "violetred"           "violetred1"
## [643] "violetred2"          "violetred3"          "violetred4"
## [646] "wheat"               "wheat1"              "wheat2"
## [649] "wheat3"              "wheat4"              "whitesmoke"
## [652] "yellow"              "yellow1"             "yellow2"
## [655] "yellow3"             "yellow4"             "yellowgreen"
```

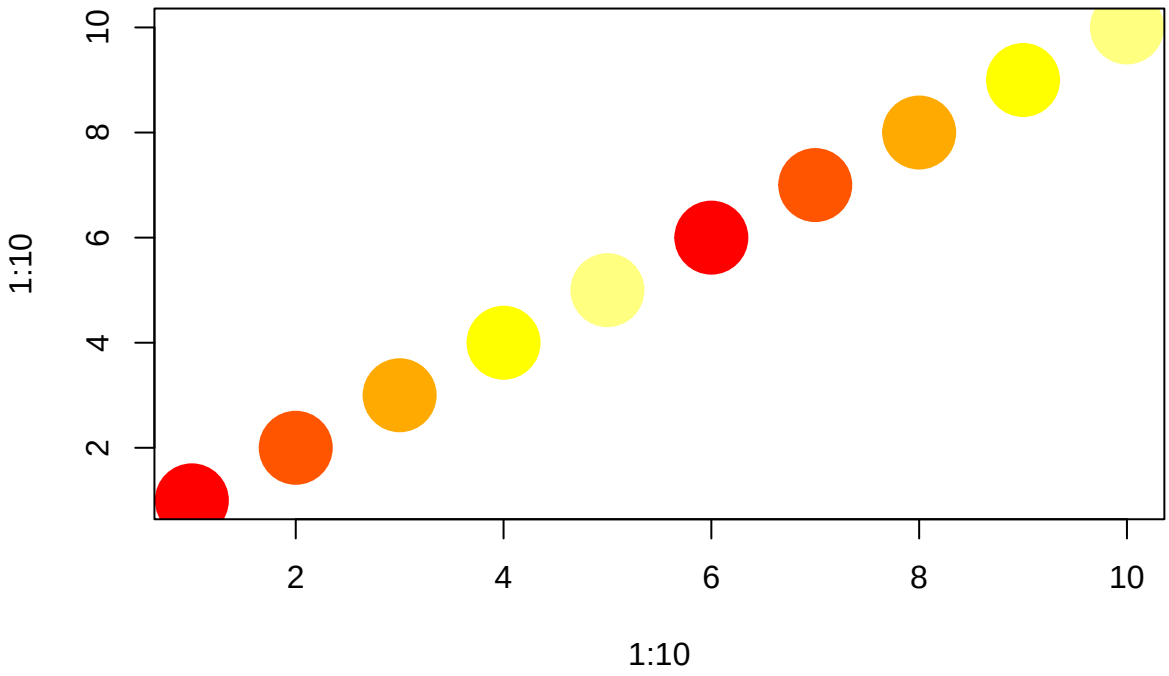
```
grep("blue", colors(), value=TRUE) # alle kleurnamen met "blue"
```

```
##  [1] "aliceblue"       "blue"            "blue1"           "blue2"
##  [5] "blue3"           "blue4"           "blueviolet"      "cadetblue"
##  [9] "cadetblue1"      "cadetblue2"      "cadetblue3"      "cadetblue4"
## [13] "cornflowerblue"  "darkblue"        "darkslateblue"   "deepskyblue"
## [17] "deepskyblue1"    "deepskyblue2"    "deepskyblue3"    "deepskyblue4"
## [21] "dodgerblue"      "dodgerblue1"     "dodgerblue2"     "dodgerblue3"
## [25] "dodgerblue4"     "lightblue"       "lightblue1"      "lightblue2"
## [29] "lightblue3"      "lightblue4"      "lightskyblue"    "lightskyblue1"
## [33] "lightskyblue2"   "lightskyblue3"   "lightskyblue4"   "lightslateblue"
## [37] "lightsteelblue"  "lightsteelblue1" "lightsteelblue2" "lightsteelblue3"
```

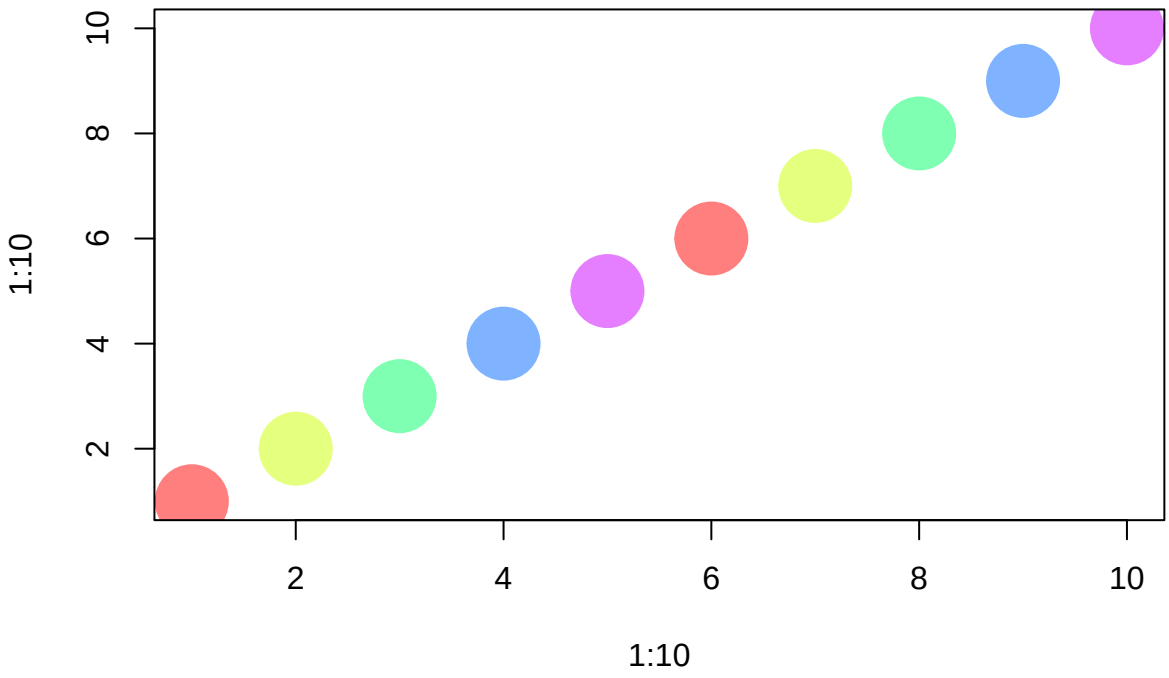
```
## [41] "lightsteelblue4" "mediumblue"      "mediumslateblue" "midnightblue"
## [45] "navyblue"        "powderblue"      "royalblue"       "royalblue1"
## [49] "royalblue2"      "royalblue3"      "royalblue4"      "skyblue"
## [53] "skyblue1"        "skyblue2"        "skyblue3"        "skyblue4"
## [57] "slateblue"       "slateblue1"      "slateblue2"      "slateblue3"
## [61] "slateblue4"      "steelblue"       "steelblue1"      "steelblue2"
## [65] "steelblue3"      "steelblue4"
```

Voor contrasterende of analoge kleurenreeksen bestaan ingebouwde paletten zoals `heat.colors()` en `rainbow()`:

```
pal1 <- heat.colors(5, alpha=1)
pal2 <- rainbow(5, alpha=.5)

plot(x=1:10, y=1:10, pch=19, cex=5, col=pal1)
```

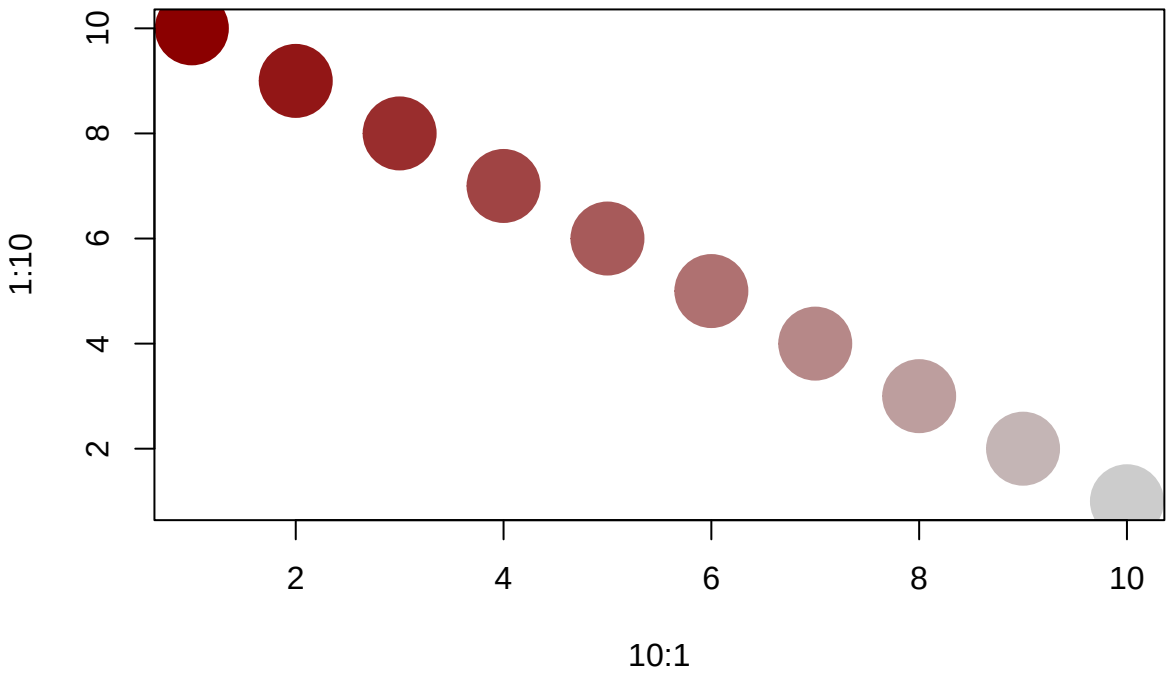



```
plot(x=1:10, y=1:10, pch=19, cex=5, col=pal2)
```



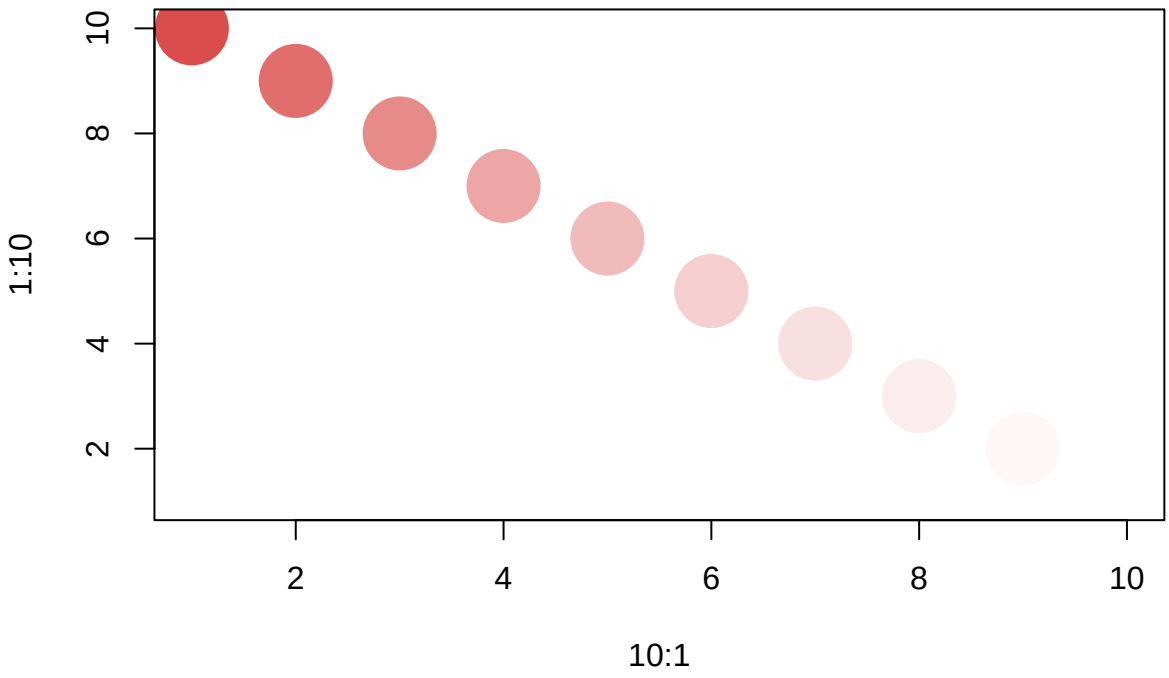
Met `colorRampPalette()` kan je aangepaste kleurgradaties genereren. De functie retourneert een nieuwe functie waarmee je zelf het aantal gewenste kleuren bepaalt:

```
palf <- colorRampPalette(c("gray80", "dark red"))
plot(x=10:1, y=1:10, pch=19, cex=5, col=palf(10))
```



Wil je transparantie toevoegen, gebruik dan `alpha=TRUE`:

```
palf <- colorRampPalette(c(rgb(1,1,1, .2), rgb(.8,0,0, .7)), alpha=TRUE)
plot(x=10:1, y=1:10, pch=19, cex=5, col=palf(10))
```



3. Visualisaties met behulp van het `igraph`-package

We starten met het demonstreren van enkele functies binnen het `igraph`-package die ons toelaten simpele visualisaties te maken van sociale netwerken. We maken hiervoor opnieuw gebruik van het classroom netwerk. We laden dit netwerk eerst opnieuw in en voeren enkele basisoperaties uit:

```
# Inlezen van de data
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.3.3
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:igraph':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
class_mat <- read.csv(file = "Data/class555_matrix.csv", header = TRUE)
class_mat <- as.matrix(class_mat)
rownames(class_mat) <- 1:nrow(class_mat)
colnames(class_mat) <- 1:ncol(class_mat)

class_matrix_net <- igraph::graph_from_adjacency_matrix(adjmatrix = class_mat,
                                                        mode = "directed")

class_att <- read.csv(file = "Data/class555_attributedata.csv", header = TRUE)

class_matrix_net <- igraph::set_vertex_attr(graph = class_matrix_net,
                                    name = "gender",
                                    value = class_att$gender)


class_matrix_net <- igraph::set_vertex_attr(graph = class_matrix_net,
                                    name = "grade",
                                    value = class_att$grade)

class_matrix_net <- igraph::set_vertex_attr(graph = class_matrix_net,
                                    name = "race",
                                    value = class_att$race)
class_matrix_net
```

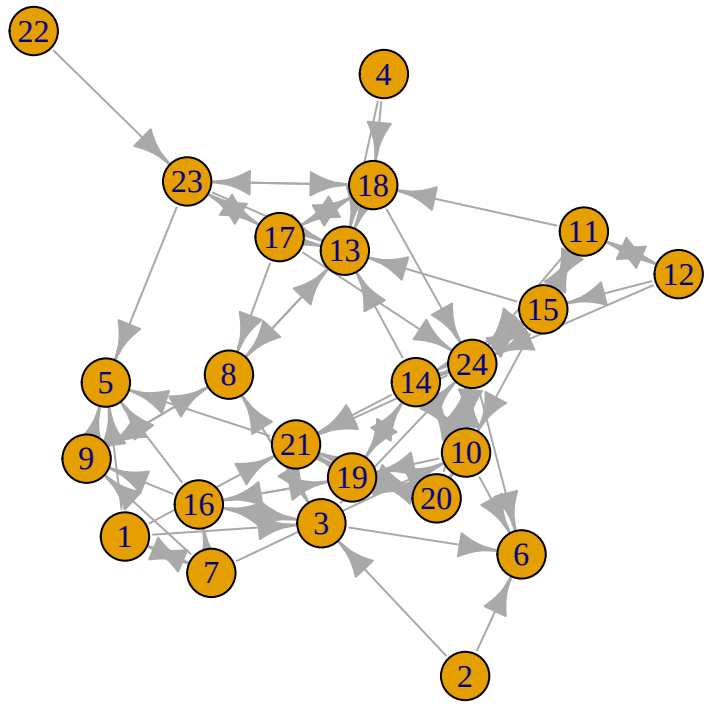
```
## IGRAPH a924ffd DN-- 24 77 --
## + attr: name (v/c), gender (v/c), grade (v/n), race (v/c)
## + edges from a924ffd (vertex names):
##  [1] 1 ->3  1 ->5  1 ->7  1 ->21 2 ->3  2 ->6  3 ->6  3 ->8  3 ->16 3 ->24
## [11] 4 ->13 4 ->18 7 ->1  7 ->9  7 ->10 7 ->16 8 ->3  8 ->9  8 ->13 9 ->5
## [21] 9 ->8  10->6  10->14 10->19 10->20 10->24 11->12 11->15 11->18 11->24
## [31] 12->11 12->15 12->24 13->8  14->10 14->13 14->19 14->21 14->24 15->10
## [41] 15->11 15->13 15->14 15->24 16->3  16->5  16->9  16->19 17->8  17->13
```

```
## [51] 17->18 17->23 17->24 18->13 18->17 18->23 18->24 19->14 19->16 19->20
## [61] 19->21 20->19 20->21 20->24 21->5  21->19 21->20 22->23 23->5  23->13
## [71] 23->17 23->18 24->6  24->10 24->14 24->15 24->21
```

Netwerkplots bieden een intuïtieve manier om de eigenschappen van een netwerk te verkennen. Zo kan een onderzoeker bijvoorbeeld geïnteresseerd zijn in de manier waarop demografische kenmerken (zoals gender of etniciteit) zich vertalen naar vriendschapsrelaties. Of men kan willen nagaan of er bijzonder centrale knooppunten in het netwerk zijn. Uiteraard betreft dit geen formele toetsing, maar het bekijken van een visuele representatie van het netwerk vormt vaak een nuttig vertrekpunt voor verdere analyse. In wat volgt, verkennen we eerst de rol van gender in het netwerk (met andere woorden: in welke mate komt gender overeen met vriendschapsgroepen?).

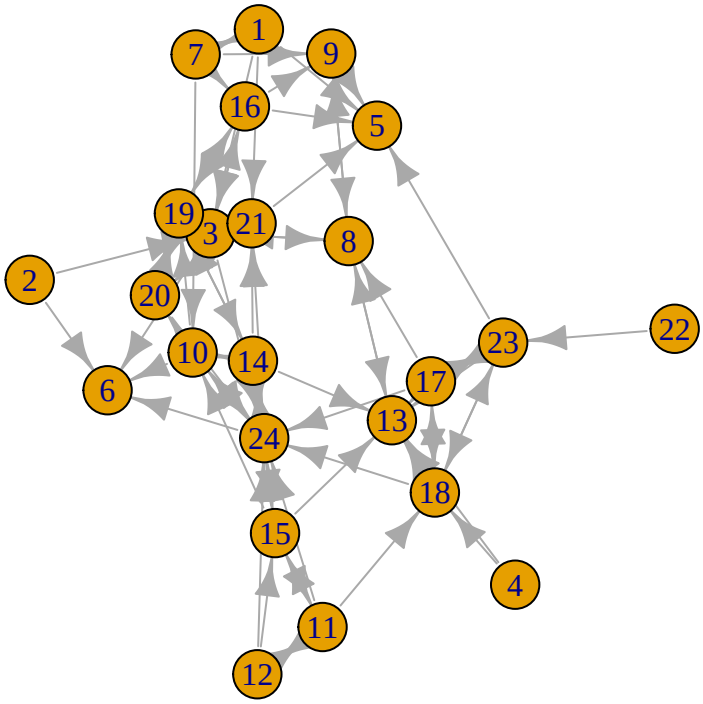
We starten met een simpel netwerkplot die we vervolgens uitbreiden met enkele interessante functies:

```
par(mfrow=c(1,1), mar = c(2, 2, 1, 1))
plot(class_matrix_net)
```



Belangrijk om op te merken is dat telkens we de code hierboven laten lopen in R, de positie van het netwerk wijzigt. Dit kunnen we simpel oplossen:

```
par(mfrow=c(1,1), mar = c(2, 2, 1, 1))
set.seed(1234)  # Kies een willekeurig getal, bv. 1234
layout_stabiel <- igraph::layout_with_fr(class_matrix_net)

# Vervolgens gebruik je dit om te plotten
plot(class_matrix_net, layout = layout_stabiel)
```

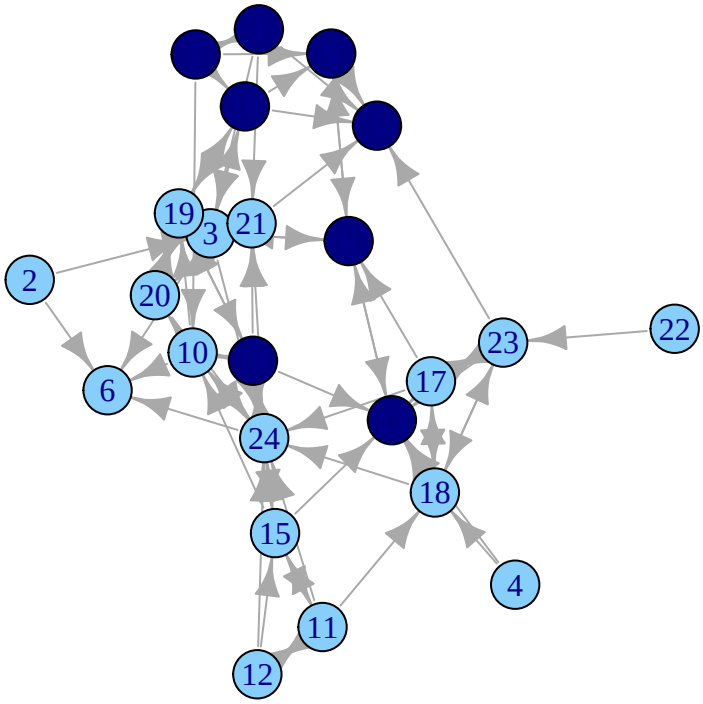


Dit plot ziet er goed uit, maar kan visueel veel beter. Het zegt ons ook niets over het geslacht, bijvoorbeeld. Laten we de nodes een meer betekenisvolle kleur geven. We kleuren de knooppunten per geslacht en maken jongens marineblauw en meisjes licht hemelsblauw. We maken eerst een vector die de gewenste kleur van elk knooppunt aangeeft.

```
par(mfrow=c(1,1), mar = c(2, 2, 1, 1))
cols <- ifelse(class_att$gender == "Female", "lightskyblue", "navy") # ifelse functie om kleur toe te k
table(cols, class_att$gender) # check de distributie
```

```
##
## cols           Female Male
##   lightskyblue     16    0
##   navy              0    8
```

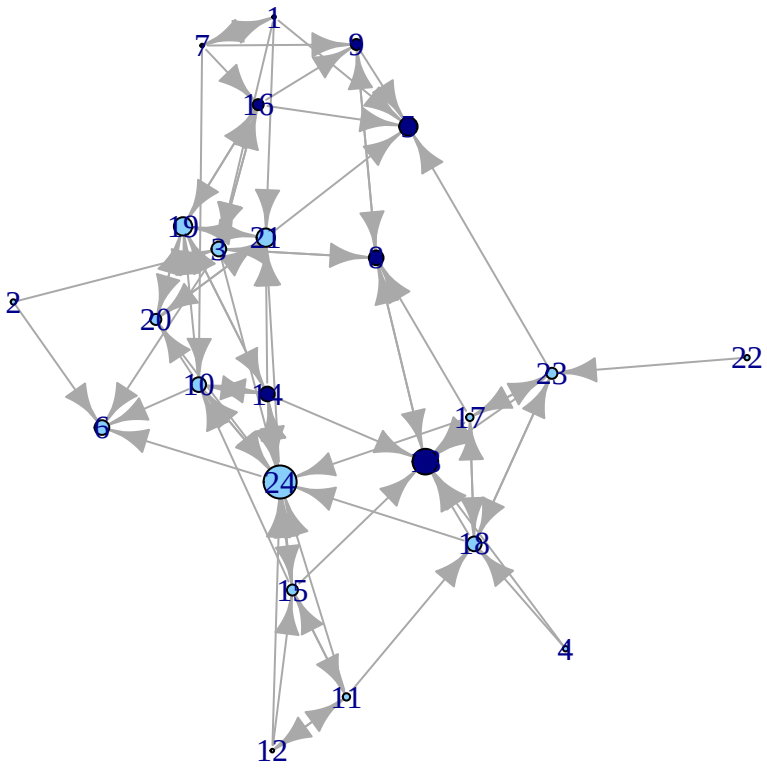
```
igraph::V(class_matrix_net)$color <- cols # kleur van elke node instellen (gebaseerd op de kleuren gede
plot(class_matrix_net, layout = layout_stabiel)
```



Het is ook mogelijk om de kleuren in de functie zelf in te stellen met behulp van een `vertex.color-argument`: `plot(class_net, vertex.color = cols)`

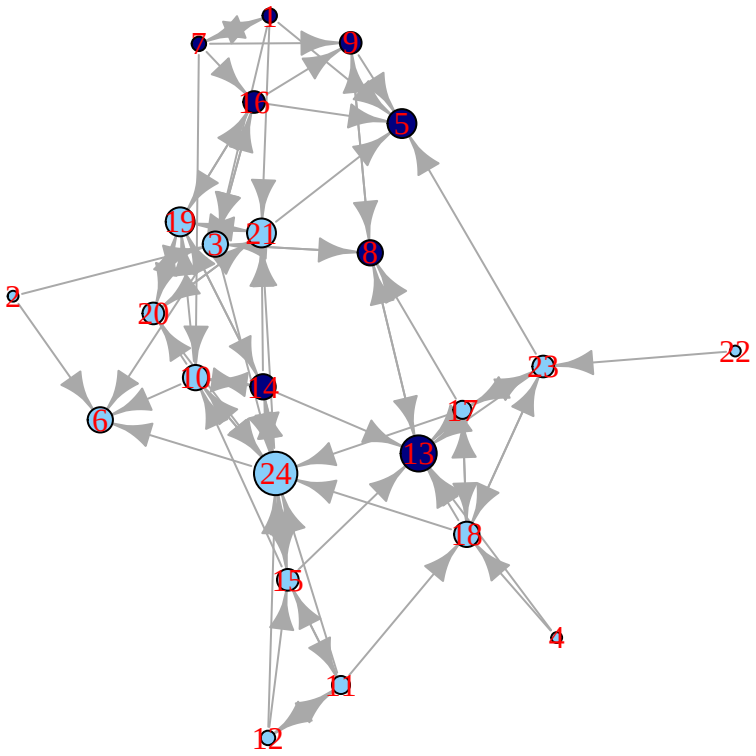
Op basis van deze plot kunnen we zien dat het netwerk zich wel degelijk langs lijnen verdeelt, met een kleine groep jongens en vervolgens een grotere groep meisjes (die onderling verdeeld zijn). We zien ook dat twee jongens geen deel uitmaken van de 'jongensgroep' en onevenredig verbonden zijn met meisjes. Een onderzoeker kan ook geïnteresseerd zijn in welke nodes de meeste nominaties ontvangen. We kunnen bijvoorbeeld willen weten of bepaalde meisjes/jongens echt belangrijk zijn in het netwerk. Om dit gemakkelijker te kunnen zien, kunnen we de indegree bepalen:

```
par(mfrow=c(1,1), mar = c(2, 2, 1, 1))
indeg <- igraph::degree(class_matrix_net, mode = "in") #indegree
plot(class_matrix_net, vertex.size = indeg, margin = -.10, layout = layout_stabiel)
```



Hier maken we alle nodes een beetje groter (door een 3 toe te voegen aan indegree), maar de knooppunten worden nog steeds gesorteerd op indegree. We veranderen ook de kleur van de labels om ze wat gemakkelijker leesbaar te maken:

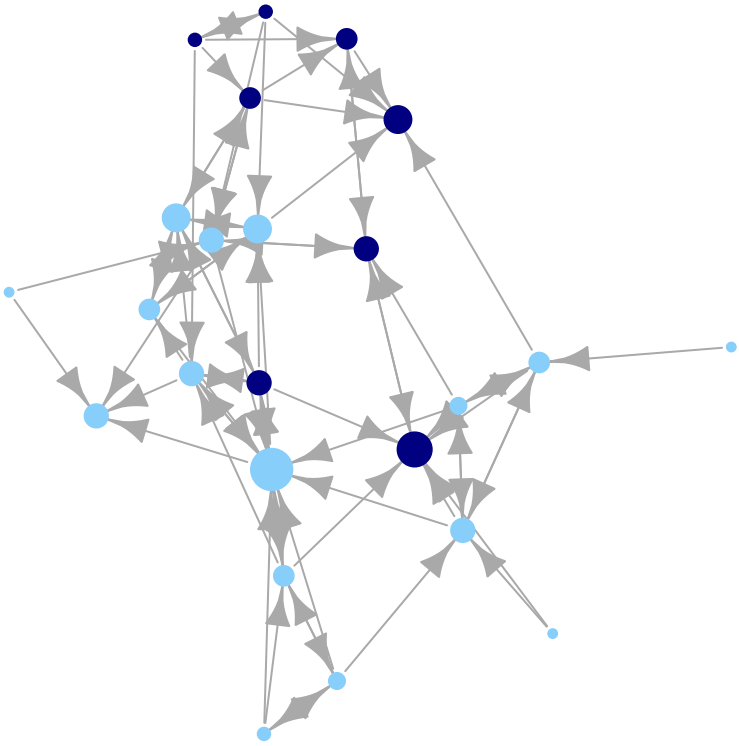
```
par(mfrow=c(1,1), mar = c(2, 2, 1, 1))
plot(class_matrix_net, vertex.size = indeg + 3, vertex.label.color = "red",
     margin = -.10, layout = layout_stabiel)
```



We zien dat één jongen (ID 13) en één meisje (ID 24) een opvallend hoog aantal nominaties ontvangen – mogelijk zijn zij centrale of populaire actoren binnen verschillende subgroepen. Daarnaast merken we op dat de jongens in de "jongensgroep" over het algemeen een lage indegree hebben, aangezien zij enkel met elkaar bevriend zijn en er relatief weinig jongens aanwezig zijn in het netwerk.

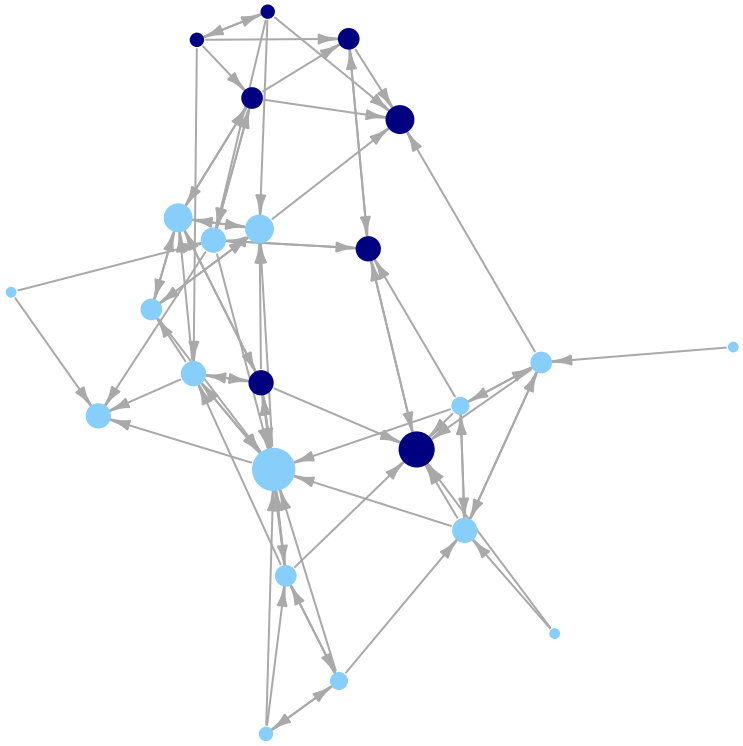
We gaan de plot nu wat opkuisen om deze visueel aantrekkelijker te maken. Dit is van belang bij het aanmaken van figuren voor publicaties, websites of presentaties. We starten door het uiterlijk van de knopen aan te passen: we verwijderen de labels met het argument vertex.label, en halen de zwarte randlijnen rond de knopen weg met het argument `vertex.frame.color`.

```
par(mfrow=c(1,1), mar = c(2, 2, 1, 1))
plot(class_matrix_net, vertex.size = indeg + 3, vertex.label = NA,
     vertex.frame.color = NA, margin = -.10, layout = layout_stabiel)
```



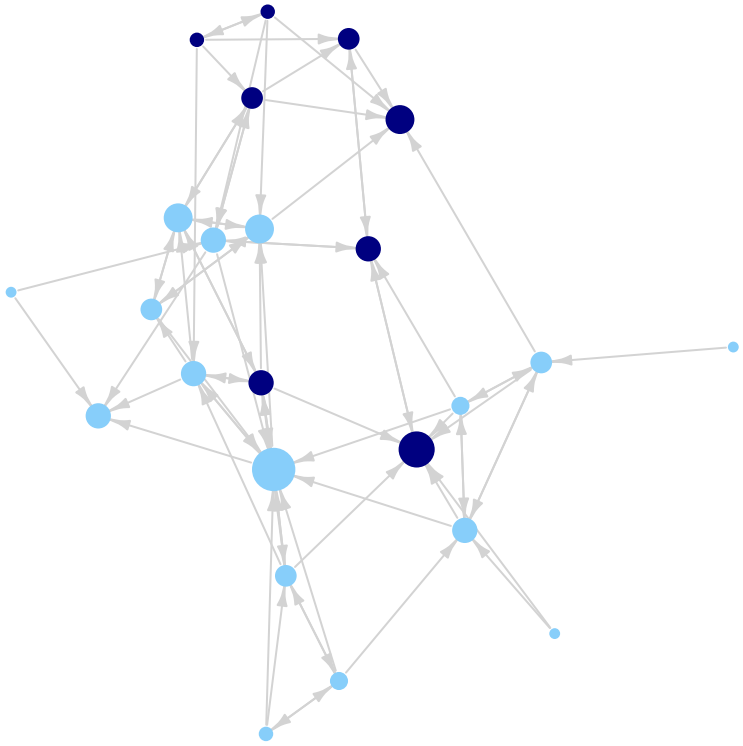
Laten we die pijlen een beetje kleiner maken met de argumenten `edge.arrow.size` en `edge.arrow.width`:

```
par(mfrow=c(1,1), mar = c(2, 2, 1, 1))
plot(class_matrix_net, vertex.size = indeg + 3, vertex.label = NA,
     vertex.frame.color = NA, edge.arrow.size = .5, edge.arrow.width = .75,
     margin = -.10, layout = layout_stabiel)
```



Vervolgens kunnen we ook het uiterlijk van de verbindingslijnen (edges) aanpassen. We veranderen de kleur van de lijnen naar lichtgrijs met behulp van het argument `edge.color`. In het algemeen vergemakkelijkt een lichtere kleur van de verbindingen de interpretatie van netwerkplots, vooral wanneer het netwerk groot en/of dicht is. Op deze manier helpen lichte verbindingslijnen om een visueel onaantrekkelijke en moeilijk leesbare "haarbal"-structuur te vermijden.

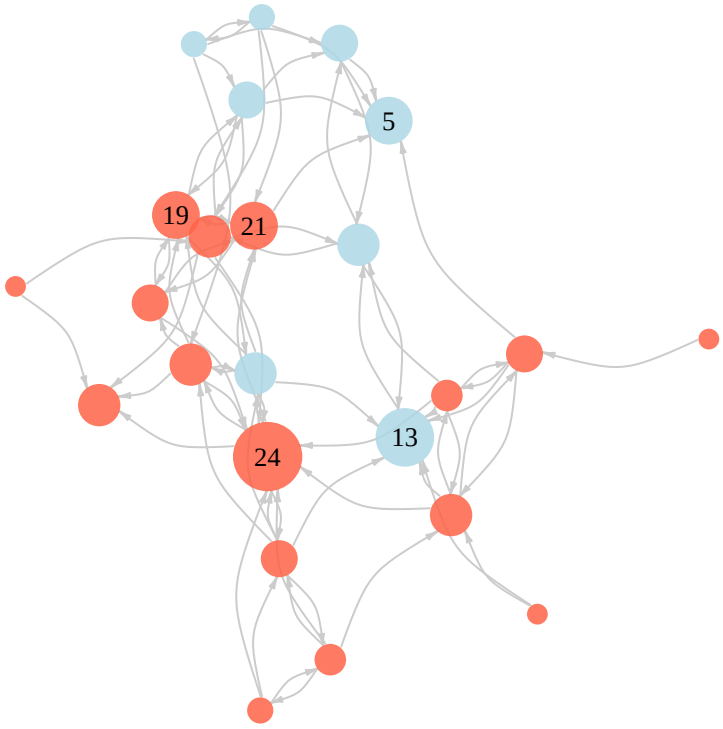
```
par(mfrow=c(1,1), mar = c(2, 2, 1, 1))
plot(class_matrix_net, vertex.size = indeg + 3, vertex.label = NA,
     vertex.frame.color = NA, edge.arrow.size = .5, edge.arrow.width = .75,
     margin = -.10, layout = layout_stabiel, edge.color = 'light gray')
```



Laten we onze igraph plot nog wat verder upgraden, kunnen jullie aangeven welke veranderingen hier werden doorgevoerd zonder naar het plot te kijken?

```
#install.packages('scales')
library(scales)

# Stel layout en nodekleur op basis van geslacht in
#layout_stabiel <- layout_with_fr(class_matrix_net)
vertex_colors <- ifelse(igraph::V(class_matrix_net)$gender == "Male", "lightblue", "tomato")

# Bereken schaal voor knoopgrootte
indeg <- igraph::degree(class_matrix_net, mode = "in")
vertex_sizes <- rescale(indeg, to = c(6, 20))  # schaal van knoopgrootte

# Labels enkel voor belangrijke nodes
igraph::V(class_matrix_net)$label <- ifelse(indeg > 4, igraph::V(class_matrix_net)$name, NA)

# Plot met verbeterde esthetiek
par(mfrow = c(1, 1), mar = c(2, 2, 1, 1))
plot(class_matrix_net,
     layout = layout_stabiel,
     vertex.size = vertex_sizes,
     vertex.color = adjustcolor(vertex_colors, alpha.f = 0.85),
     vertex.frame.color = NA,
     vertex.label = igraph::V(class_matrix_net)$label,
     vertex.label.cex = 0.8,
```

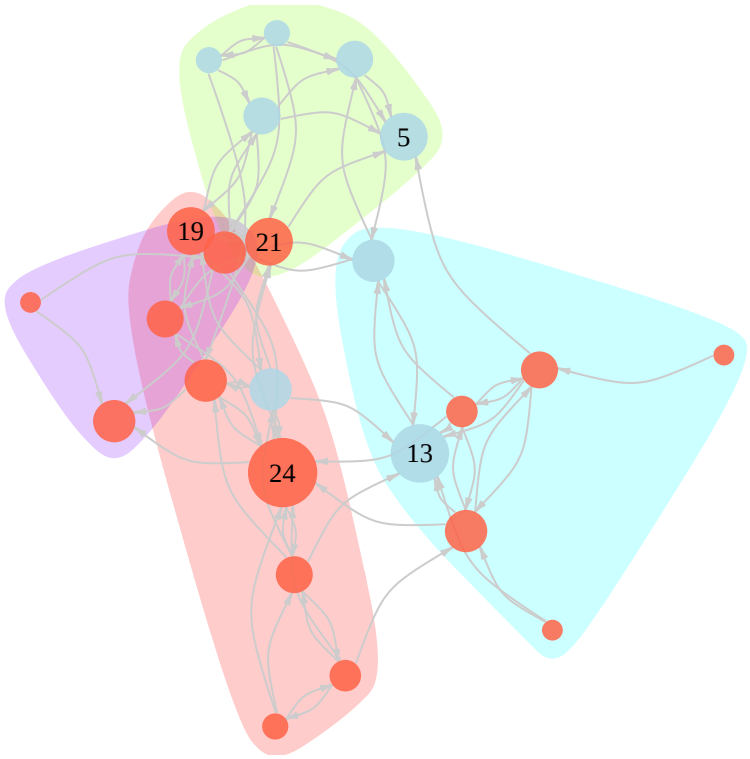
```
    vertex.label.color = "black",
    edge.color = "gray80",
    edge.width = 1,
    edge.arrow.size = 0.3,
    edge.arrow.width = 0.7,
    edge.curved = 0.4,
    margin = -0.1)
```



We kunnen ook de communities toevoegen aan het plot:

```
#install.packages('scales')
library(scales)

# Stel layout en nodekleur op basis van geslacht in
#layout_stabiel <- layout_with_fr(class_matrix_net)
vertex_colors <- ifelse(igraph::V(class_matrix_net)$gender == "Male", "lightblue", "tomato")

# Bereken schaal voor knoopgrootte
indeg <- igraph::degree(class_matrix_net, mode = "in")
vertex_sizes <- rescale(indeg, to = c(6, 20))  # schaal van knoopgrootte

# Labels enkel voor belangrijke nodes
igraph::V(class_matrix_net)$label <- ifelse(indeg > 4, igraph::V(class_matrix_net)$name, NA)

communities <- igraph::cluster_fast_greedy(igraph::as_undirected(class_matrix_net))
igraph::V(class_matrix_net)$community <- igraph::membership(communities)
```

```r
# Lijst van knopen per community
community_list <- communities$membership
group_nodes <- split(igraph::V(class_matrix_net), community_list)

# Kleuren voor de omtrekcirkels van communities
group_colors <- rainbow(length(group_nodes), alpha = 0.2)


# Plot met verbeterde esthetiek én community info
par(mar = c(2, 2, 1, 1))
plot(class_matrix_net,
     layout = layout_stabiel,
     vertex.size = vertex_sizes,
     vertex.color = adjustcolor(vertex_colors, alpha.f = 0.85),
     vertex.frame.color = NA,
     vertex.label = igraph::V(class_matrix_net)$label,
     vertex.label.cex = 0.8,
     vertex.label.color = "black",
     edge.color = "gray80",
     edge.width = 1,
     edge.arrow.size = 0.3,
     edge.arrow.width = 0.7,
     edge.curved = 0.4,
     margin = -0.1,
     mark.groups = group_nodes,              # <-- Cirkels rond groepen
     mark.col = group_colors,                # <-- Cirkelkleur
     mark.border = NA)                       # Geen rand rond de cirkels
```



Voor een overzicht van alle parameters voor netwerkvisualisatie in `igraph (cheat sheet)`:

NODES

Argument	Beschrijving
`vertex.color`	Kleur van de knoop
`vertex.frame.color`	Randkleur van de knoop
`vertex.shape`	Vorm van de knoop: `"none"`, `"circle"`, `"square"`, `"csquare"`, `"rectangle"`, `"crectangle"`, `"vrectangle"`, `"pie"`, `"raster"`, `"sphere"`
`vertex.size`	Grootte van de knoop (standaard: 15)
`vertex.size2`	Tweede dimensie van de knoop (bijv. voor een rechthoek)
`vertex.label`	Labels voor knopen
`vertex.label.family`	Lettertypefamilie voor labels (bijv. `"Times"`, `"Helvetica"`)
`vertex.label.font`	Letterstijl: 1 = normaal, 2 = vet, 3 = cursief, 4 = vet cursief, 5 = symbool
`vertex.label.cex`	Schaalfactor voor labelgrootte
`vertex.label.dist`	Afstand tussen label en knoop
`vertex.label.degree`	Positie van het label t.o.v. de node

EDGES

Argument	Beschrijving
`edge.color`	Kleur van de randen

Argument	Beschrijving
`edge.width`	Dikte van de randen (standaard = 1)
`edge.arrow.size`	Grootte van pijlpunt (standaard = 1)
`edge.arrow.width`	Breedte van pijlpunt (standaard = 1)
`edge.lty`	Lijntype: 0 = geen lijn, 1 = solide, 2 = gestreept, 3 = puntjes, 4 = punt-streep, 5 = lange streep, 6 = dubbele streep
`edge.label`	Labels voor randen
`edge.label.family`	Lettertypefamilie voor randlabels
`edge.label.font`	Stijl randlabel: 1 = normaal, 2 = vet, enz.
`edge.label.cex`	Grootte van randlabels
`edge.curved`	Kromming van randen (0 tot 1, `FALSE` = 0, `TRUE` = 0.5)
`arrow.mode`	Bepaalt richting van pijlen: 0 = geen, 1 = terug, 2 = vooruit, 3 = beide

OVERIG

Argument	Beschrijving
`margin`	Marges rond het plot (vector van lengte 4)
`frame`	Indien `TRUE`, wordt het plot omkaderd
`main`	Plot-titel
`sub`	Ondertitel
`asp`	Aspect ratio van de plot (y/x)
`palette`	Kleurenschema voor knopenkleuren
`rescale`	Herbereken coördinaten naar [-1, 1] (standaard = `TRUE`)

4. Visualisaties met behulp van het `ggplot`-package

Visualisaties met behulp van het `ggplot2`-package

Het `ggplot2`-pakket is een krachtig en flexibel hulpmiddel voor het creëren van visueel aantrekkelijke en informatief rijke grafieken in R. Gebaseerd op het principe van de *Grammar of Graphics* (Wilkinson, 2005), biedt `ggplot2` een consistente en modulaire aanpak voor het opbouwen van visualisaties. In plaats van te vertrekken vanuit een specifieke grafiekvorm, stelt `ggplot2` de gebruiker in staat om grafieken op te bouwen door systematisch lagen toe te voegen: data, esthetische mappings (*aesthetics*), geometrische objecten (*geoms*), schalen, thema's en meer.

Een belangrijk voordeel van `ggplot2` is de declaratieve aard ervan: je beschrijft **wat** je wil visualiseren, niet **hoe** het precies getekend moet worden. Hierdoor ontstaat een hoge mate van controle, herbruikbaarheid en reproduceerbaarheid, wat bijzonder waardevol is in wetenschappelijke en academische contexten.

Binnen deze sectie tonen we hoe men met `ggplot2` gegevens op een overzichtelijke en esthetisch verzorgde manier kan visualiseren. We behandelen basiselementen zoals punten- en lijngrafieken, alsook meer geavanceerde toepassingen zoals facettering, kleurgebruik en aangepaste thema's.

Laat ons eerste nodige packages installeren en laden en `igraph` loskoppelen:

```
library(igraph)
detach(package:igraph)
library(network)
```

```
## Warning: package 'network' was built under R version 4.3.3
```

```
##
## 'network' 1.19.0 (2024-12-08), part of the Statnet Project
## * 'news(package="network")' for changes since last version
## * 'citation("network")' for citation information
## * 'https://statnet.org' for help, support, and other information
```

```
library(intergraph)
```

```
## Warning: package 'intergraph' was built under R version 4.3.3
```

```
library(ggplot2)
#install.packages('GGally')
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

We lezen eerst opnieuw nog even de data in:

```
# Inlezen van de data
#library(igraph)
#library(dplyr)
class_mat <- read.csv(file = "Data/class555_matrix.csv", header = TRUE)
class_mat <- as.matrix(class_mat)
rownames(class_mat) <- 1:nrow(class_mat)
colnames(class_mat) <- 1:ncol(class_mat)

class_matrix_net <- igraph::graph_from_adjacency_matrix(adjmatrix = class_mat,
                                                        mode = "directed")

class_att <- read.csv(file = "Data/class555_attributedata.csv", header = TRUE)

class_matrix_net <- igraph::set_vertex_attr(graph = class_matrix_net,
                                    name = "gender",
                                    value = class_att$gender)


class_matrix_net <- igraph::set_vertex_attr(graph = class_matrix_net,
                                    name = "grade",
                                    value = class_att$grade)

class_matrix_net <- igraph::set_vertex_attr(graph = class_matrix_net,
                                    name = "race",
                                    value = class_att$race)
class_matrix_net
```

```
## IGRAPH aa2525c DN-- 24 77 --
## + attr: name (v/c), gender (v/c), grade (v/n), race (v/c)
```

```
## + edges from aa2525c (vertex names):
##  [1] 1 ->3  1 ->5  1 ->7  1 ->21 2 ->3  2 ->6  3 ->6  3 ->8  3 ->16 3 ->24
## [11] 4 ->13 4 ->18 7 ->1  7 ->9  7 ->10 7 ->16 8 ->3  8 ->9  8 ->13 9 ->5
## [21] 9 ->8  10->6  10->14 10->19 10->20 10->24 11->12 11->15 11->18 11->24
## [31] 12->11 12->15 12->24 13->8  14->10 14->13 14->19 14->21 14->24 15->10
## [41] 15->11 15->13 15->14 15->24 16->3  16->5  16->9  16->19 17->8  17->13
## [51] 17->18 17->23 17->24 18->13 18->17 18->23 18->24 19->14 19->16 19->20
## [61] 19->21 20->19 20->21 20->24 21->5  21->19 21->20 22->23 23->5  23->13
## [71] 23->17 23->18 24->6  24->10 24->14 24->15 24->21
```

We zetten het igraph object om in een network object:

```
class_net_sna <- asNetwork(class_matrix_net)
class_net_sna
```

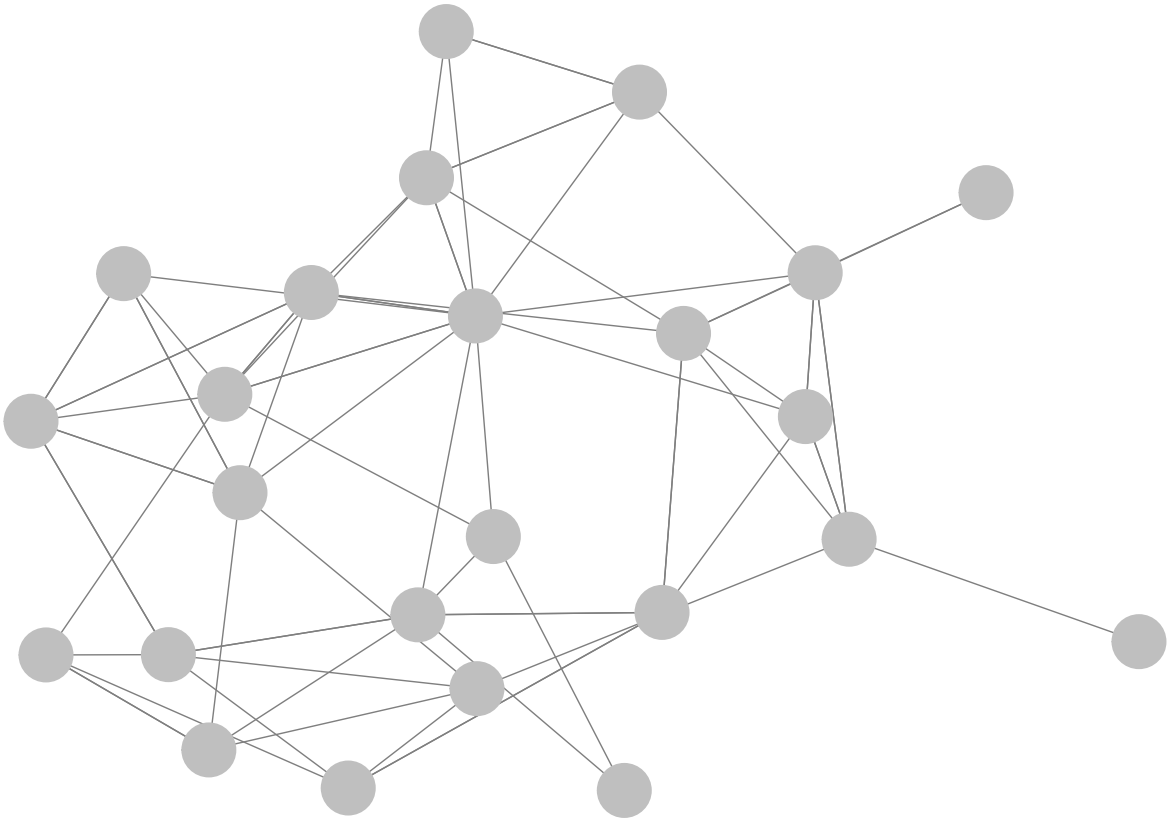
```
##  Network attributes:
##   vertices = 24
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 77
##     missing edges= 0
##     non-missing edges= 77
##
##  Vertex attribute names:
##     gender grade race vertex.names
##
## No edge attributes
```

We voegen indegree opnieuw toe als een attribuut (normaal is dit reeds opgeslagen in de environment):

```
set.vertex.attribute(class_net_sna, attrname = "indeg", value = indeg)
```

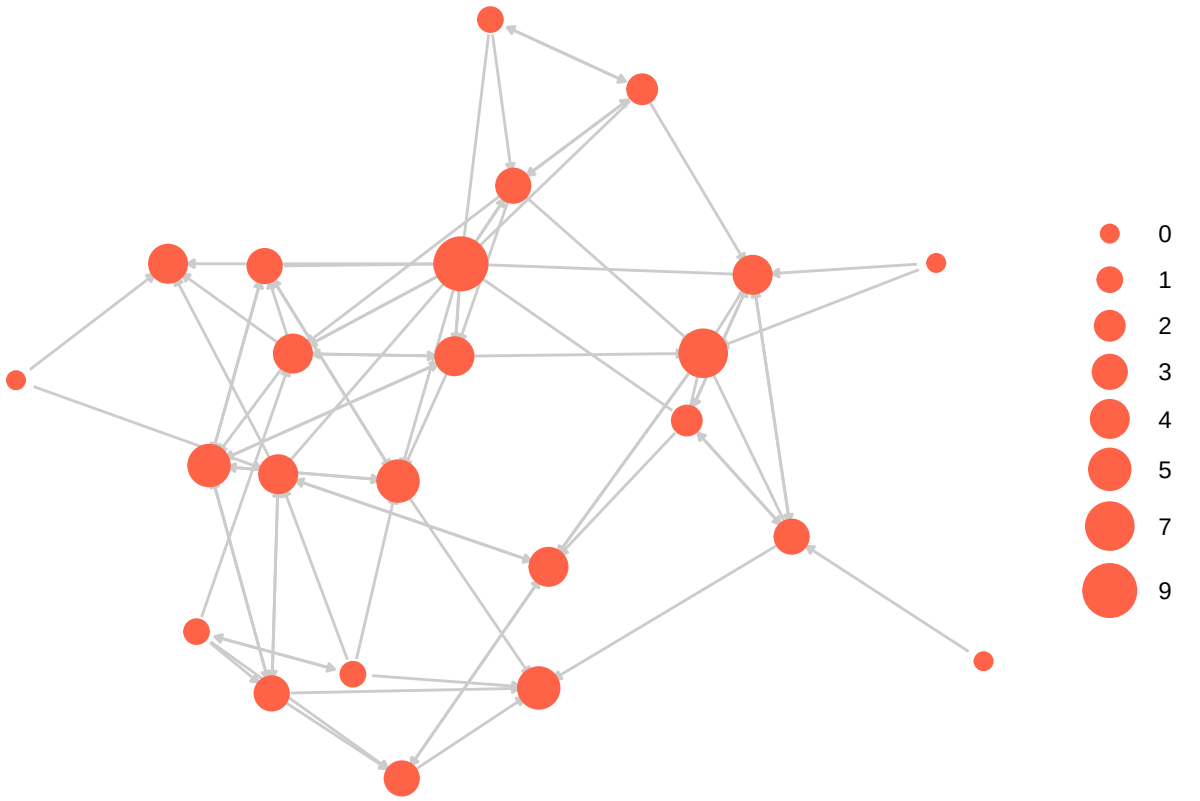
Hier demonstreren we hoe netwerken gevisualiseerd kunnen worden met behulp van de functie `ggnet2()` uit het `GGally`-pakket (Schloerke et al., 2021). De standaardplot ziet er als volgt uit:

```
ggnet2(class_net_sna)
```



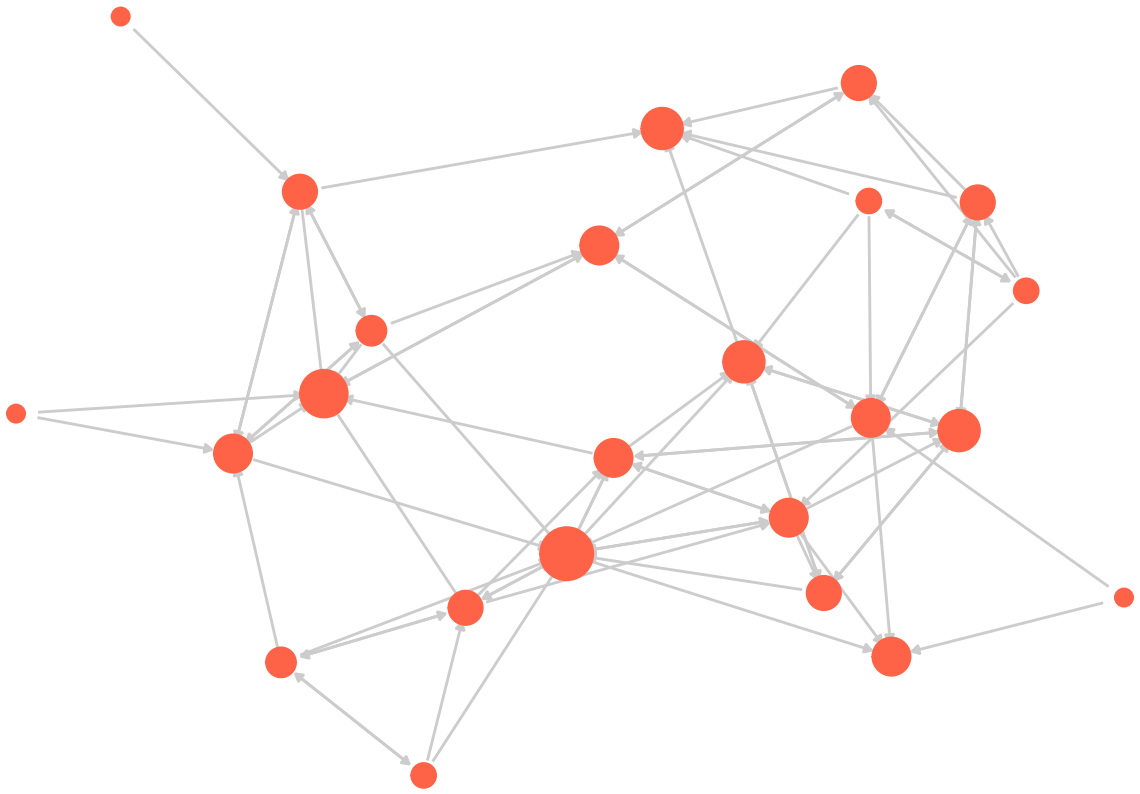
Laat ons nu enkele parameters configureren, zoals de grootte van de nodes, kleur van de nodes, grootte van de edges etc. . .

```
ggnet2(class_net_sna, node.size = indeg, node.color = c('tomato'),
       edge.size = .5, arrow.size = 3, arrow.gap = 0.02,
       edge.color = "grey80")
```



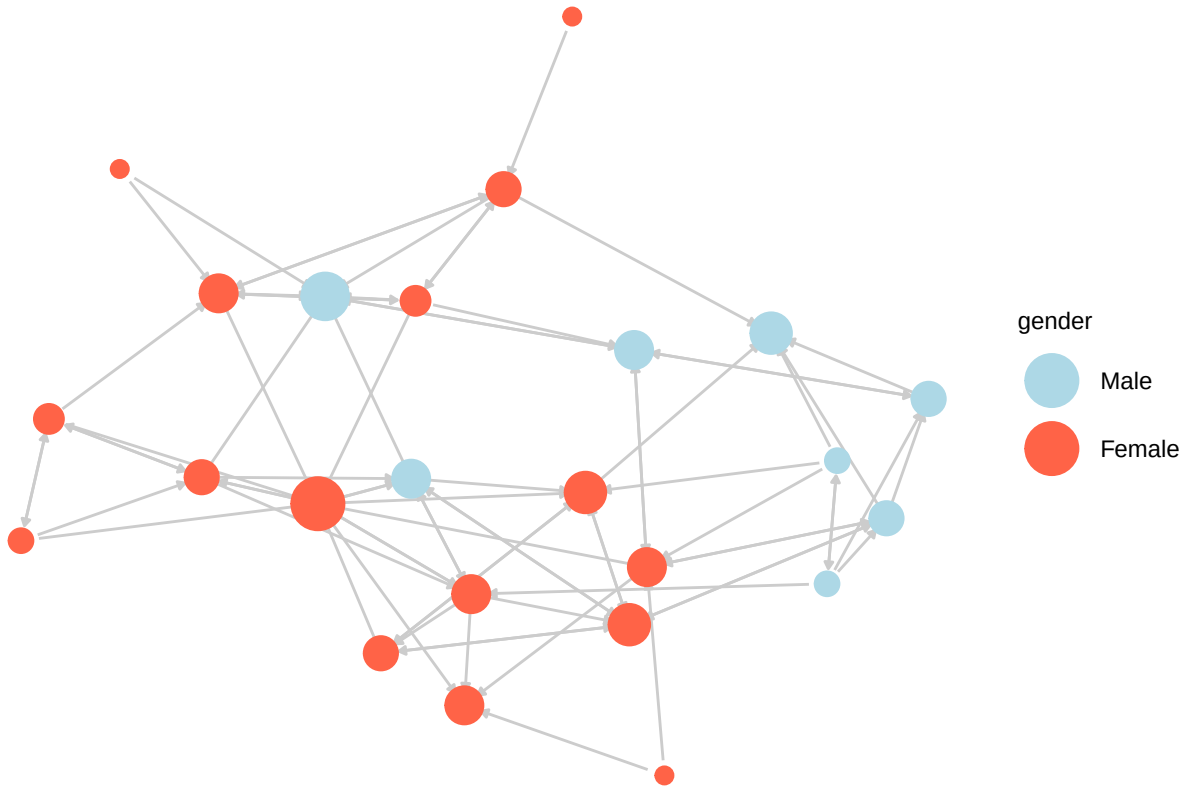
Hier maken we dezelfde plot, maar verwijderen we de legenda voor de knoopgrootte. Dit wordt bereikt (volgens de logica van ggplot) door een +-optie toe te voegen, gevolgd door de functie guides() die de legenda regelt, waarbij deze hier wordt uitgeschakeld.

```
ggnet2(class_net_sna, node.size = indeg, node.color = c('tomato'),
       edge.size = .5, arrow.size = 3, arrow.gap = 0.02,
       edge.color = "grey80", arrow.type = ) +
  guides(size = 'none')
```



Laat ons nu de nodes kleuren volgens gender:

```
ggnet2(class_net_sna, node.size = indeg,  node.color = "gender", # hier kleuren we gender in
       palette = c("Male" = "lightblue", "Female" = "tomato"),
       edge.size = .5, arrow.size = 3, arrow.gap = 0.02,
       edge.color = "grey80") +
  guides(size = "none")
```



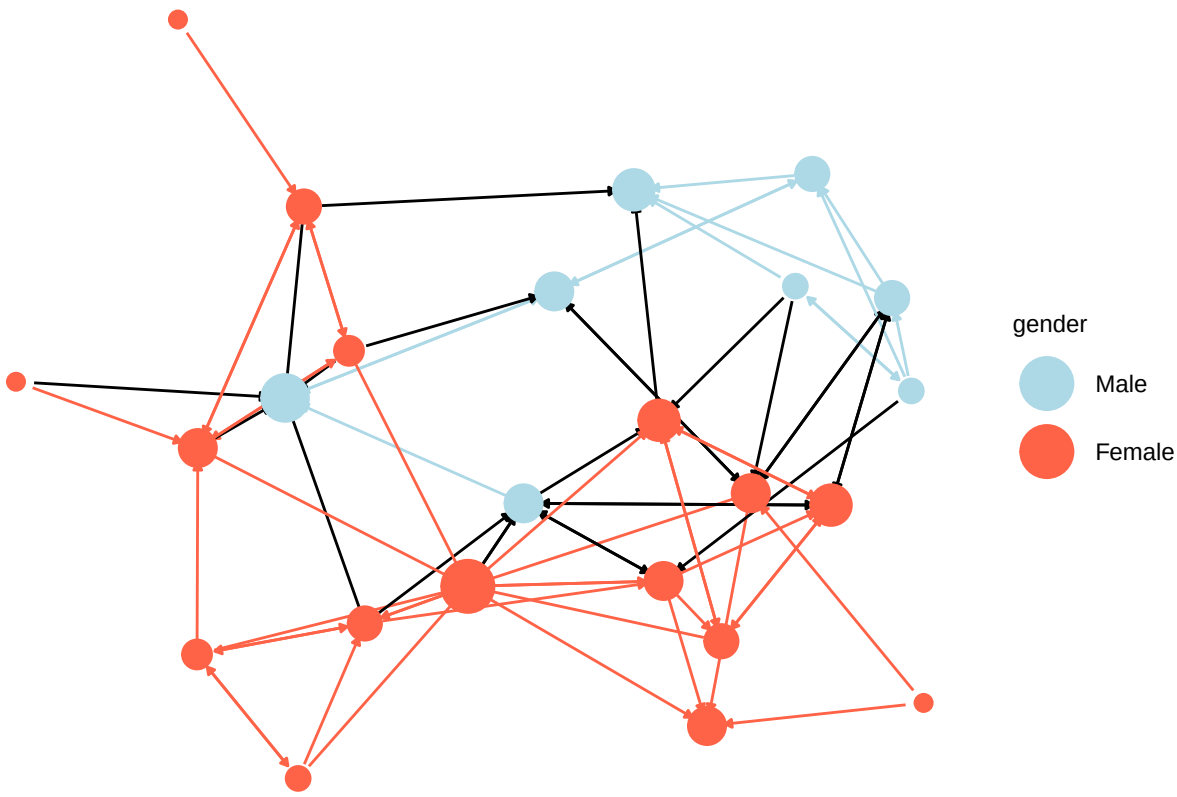
De functie `ggnet2()` maakt het eenvoudig om de randen (edges) op verschillende manieren in te kleuren.

In de volgende plot willen we de verbindingen tussen en binnen genders visueel onderscheiden:

- Alle randen van **meisjes naar meisjes** worden tomaatrood (`tomato`).
- Alle randen van **jongens naar jongens** worden lichtblauw (`lightblue`).
- Alle randen tussen **meisjes en jongens** worden zwart.

Dit wordt bereikt door het argument `edge.color` als volgt in te stellen:

```
ggnet2(class_net_sna,  node.size = indeg,  node.color = "gender",
       palette = c("Male" = "lightblue", "Female" = "tomato"),
       edge.size = .5, arrow.size = 3, arrow.gap = 0.02,
       edge.color = c("color", "black")) +
  guides(size = "none")
```



Hier geven we een zeer korte demonstratie met het `ggnetwork`-pakket (Briatte, 2023).

`ggnetwork` vervult gelijkaardige functies als `ggnet2()`, maar vereist dat de gebruiker meer rechtstreeks gebruikmaakt van de `ggplot`-functionaliteit.

Dit is vooral nuttig voor onderzoekers die een grotere mate van controle over het netwerkplot wensen.

```
#install.packages('ggnetwork')
library(ggnetwork)
```

```
## Warning: package 'ggnetwork' was built under R version 4.3.3
```

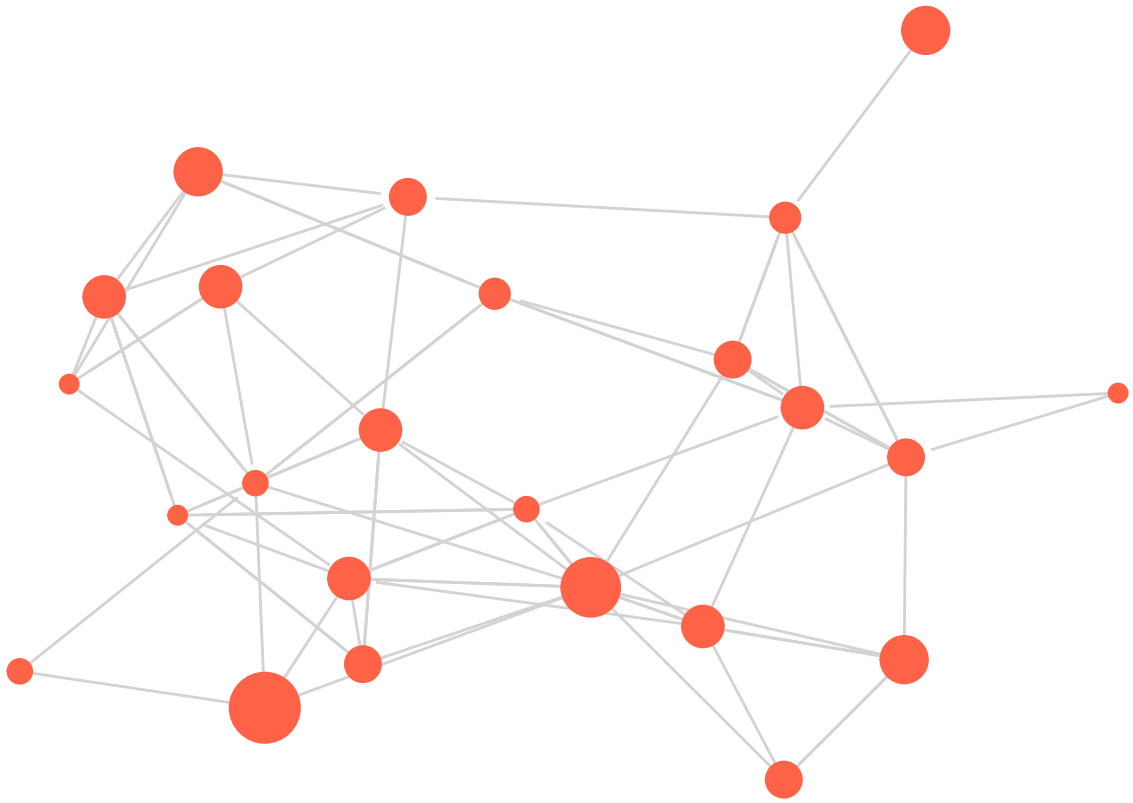
Laten we onze netwerkplot opnieuw opbouwen met behulp van de functie `ggplot()`.

`ggplot()` werkt door elke gewenste eigenschap (of laag) stapsgewijs aan de plot toe te voegen. Naast het netwerk zelf voegen we:

- een `aes()`-functie (aesthetic) toe om de locatie van de knopen te bepalen,
- een `geom_edges()`-functie om de eigenschappen van de randen te definiëren,
- een `geom_nodes()`-functie om de eigenschappen van de knopen vast te leggen,
- en de `theme_blank()`-functie om de achtergrond leeg te maken.

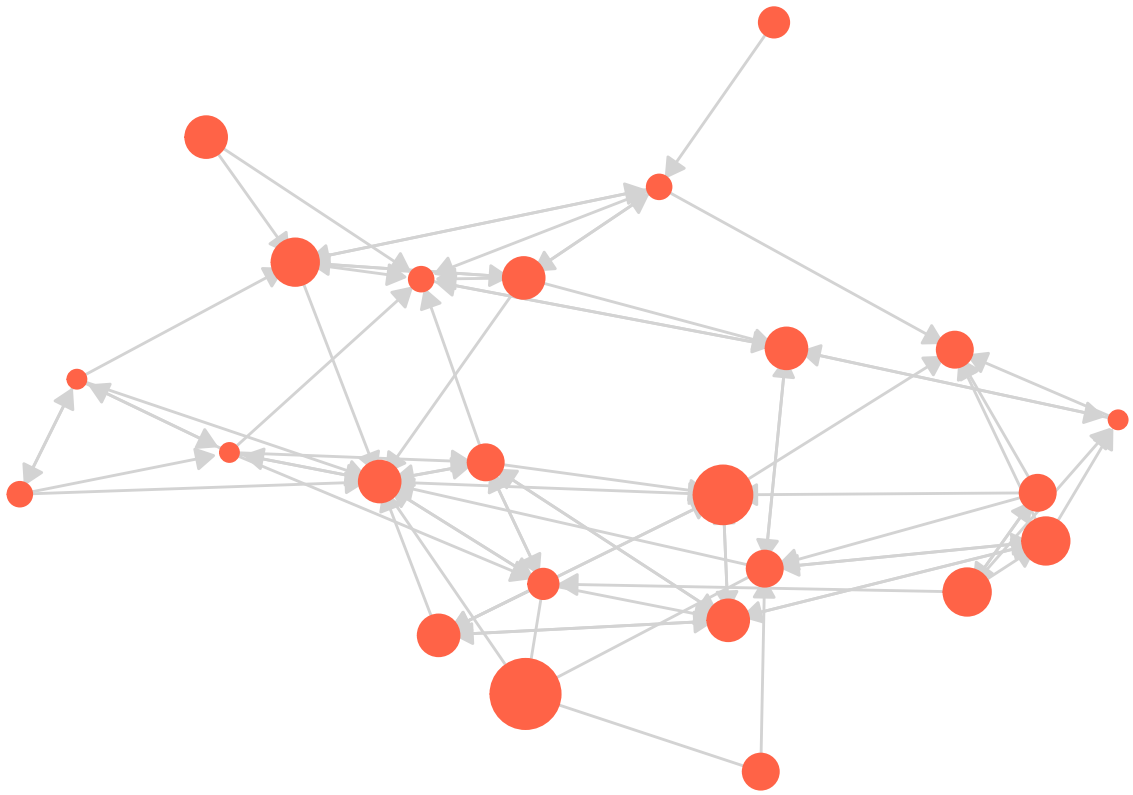
Op deze manier krijgen we volledige controle over de opbouw en het uiterlijk van het netwerkplot.

```
ggplot(class_net_sna, aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_edges(color = "lightgray") +
  geom_nodes(color = c('tomato'), size = indeg + 3) +
  theme_blank()
```



Laten we nu even de pijlen toevoegen van de relaties en verbindingen:

```
ggplot(class_net_sna, arrow.gap = .015,
       aes(x = x, y = y, xend = xend, yend = yend)) +
 geom_edges(color = "lightgray",
            arrow = arrow(length = unit(7.5, "pt"), type = "closed")) +
 geom_nodes(color = c('tomato'), size = indeg + 3) +
 theme_blank()
```



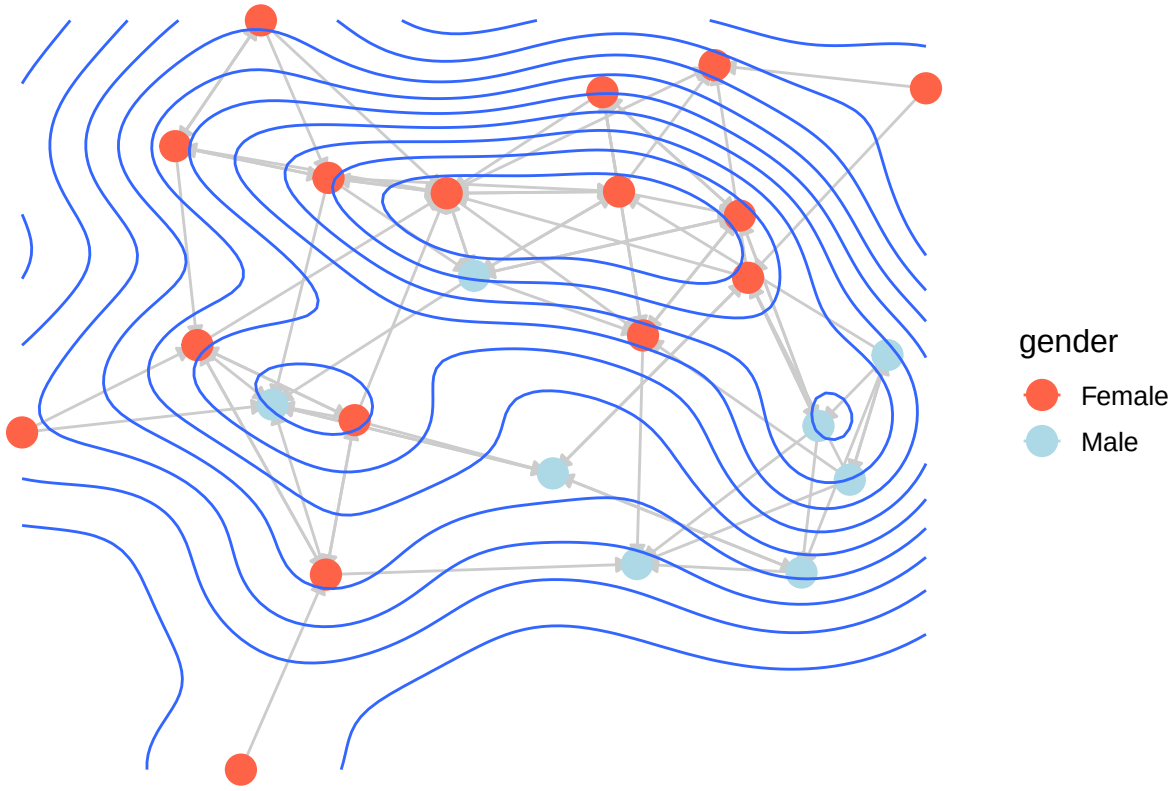
Als een ander voorbeeld gebruiken we `ggplot()` om contourplots van ons netwerk te maken. Contourplots bieden een topografische representatie van het netwerk, wat vooral nuttig is bij **zeer grote en dichte netwerken**.

We gebruiken dezelfde basisaanpak met de `ggplot()`-functie als hierboven om het netwerk te plotten, maar voegen daarbovenop een **topografische laag** toe die de dichtheid van verschillende regio's binnen het netwerk weergeeft.

Dit wordt gerealiseerd door de functie `geom_density_2d()` toe te voegen aan de `ggplot`-aanroep.

```
ggplot(class_net_sna, arrow.gap = .01, aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_edges(color = "grey80", arrow = arrow(length = unit(5, "pt"), type = "closed")) +
  geom_nodes(aes(color = gender), size = 5) +
  scale_color_manual(values = c("Male" = "lightblue", "Female" = "tomato")) +
  theme_blank()+
  geom_density_2d()
```

```
## Warning: The following aesthetics were dropped during statistical transformation: xend
## and yend.
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a ‘group‘ aesthetic or to convert a numerical
##   variable into a factor?
```



Vervolgens bieden interactieve netwerkplots een meer **'hands-on' ervaring**, waarbij de gebruiker rechtstreeks in de visualisatie bepaalde knopen kan accentueren, het netwerk kan roteren, de layout kan aanpassen, enzovoort.

Dit is bijzonder nuttig bij de **exploratieve fase** van een netwerkonderzoek, wanneer men de structuur en eigenschappen van het netwerk beter wil begrijpen.

Daarnaast vormt het een **natuurlijke manier om netwerken te presenteren op een website** of in een digitaal rapport.

In wat volgt maken we gebruik van het `networkD3`-pakket.

```
library(networkD3)
```

Het `networkD3`-pakket indexeert de nodes vanaf 0 (in plaats van vanaf 1, zoals typisch in R).

Daarom moeten we een edgelist en attributenbestand aanmaken waarbij de ID's beginnen vanaf 0.

In dit voorbeeld halen we de kolommen `sender` en `receiver` uit de oorspronkelijke edgelist en trekken we 1 af van de ID's:

```
class_edges <- read.csv(file = "Data/class555_edgelist.csv", header = TRUE)
class_edges_zeroindex <- class_edges[, c("sender", "receiver")] - 1
class_edges_zeroindex
```

```
##    sender receiver
## 1       0        2
## 2       0        4
```

```
## 3       0        6
## 4       0       20
## 5       1        2
## 6       1        5
## 7       2        5
## 8       2        7
## 9       2       15
## 10      2       23
## 11      3       12
## 12      3       17
## 13      6        0
## 14      6        8
## 15      6        9
## 16      6       15
## 17      7        2
## 18      7        8
## 19      7       12
## 20      8        4
## 21      8        7
## 22      9        5
## 23      9       13
## 24      9       18
## 25      9       19
## 26      9       23
## 27     10       11
## 28     10       14
## 29     10       17
## 30     10       23
## 31     11       10
## 32     11       14
## 33     11       23
## 34     12        7
## 35     13        9
## 36     13       12
## 37     13       18
## 38     13       20
## 39     13       23
## 40     14        9
## 41     14       10
## 42     14       12
## 43     14       13
## 44     14       23
## 45     15        2
## 46     15        4
## 47     15        8
## 48     15       18
## 49     16        7
## 50     16       12
## 51     16       17
## 52     16       22
## 53     16       23
## 54     17       12
## 55     17       16
## 56     17       22
```

```
## 57     17       23
## 58     18       13
## 59     18       15
## 60     18       19
## 61     18       20
## 62     19       18
## 63     19       20
## 64     19       23
## 65     20        4
## 66     20       18
## 67     20       19
## 68     21       22
## 69     22        4
## 70     22       12
## 71     22       16
## 72     22       17
## 73     23        5
## 74     23        9
## 75     23       13
## 76     23       14
## 77     23       20
```

We maken ook een nieuw attributenbestand aan waarbij we 1 aftrekken van de ID's.

Daarnaast voegen we de variabelen `gender` en `indeg` toe, om respectievelijk de kleur en de grootte van de knopen te bepalen.

```
class_attributes_zeroindex <- data.frame(
  id = class_att$id - 1,
  indeg = indeg,
  gender = class_att$gender
)
```

We zijn nu klaar om een eenvoudige interactieve plot te maken met de functie `forceNetwork()`.

Er zijn veel mogelijke argumenten, maar we focussen hier op de belangrijkste:

- `Links`: de edgelist van interesse
- `Nodes`: het attributenbestand
- `Source`: de variabele in de edgelist die de zender van de verbinding aanduidt
- `Target`: de variabele in de edgelist die de ontvanger van de verbinding aanduidt
- `Group`: de 'groep' van elke node, gebaseerd op een attribuut of netwerkgroep
- `Nodesize`: de variabele uit het attributenbestand die de grootte van de nodes bepaalt
- `NodeID`: de variabele uit het attributenbestand die de naam of ID van de node weergeeft

In dit voorbeeld gebruiken we de edgelist en het attributenbestand die we hierboven hebben geconstrueerd.

We gebruiken gender als groepsvariabele en bepalen de grootte van de nodes op basis van de indegree (indeg).

```r
forceNetwork(Links = class_edges_zeroindex,
             Nodes = class_attributes_zeroindex,
             Source = "sender", Target = "receiver",
             Group = "gender", Nodesize = "indeg", NodeID = "id",
             opacity = 0.9, bounded = FALSE, opacityNoHover = .2)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

5. Visualisaties met behulp van het ggraph-package

In dit deel introduceren we het `ggraph`-package, eveneens een handige tool voor het visualiseren van netwerken binnen het `tidyverse`-ecosysteem. `ggraph` biedt uitgebreide mogelijkheden om complexe netwerken op een flexibele en esthetisch verantwoorde wijze te presenteren. Door de nauwe integratie met `ggplot2` kunnen gebruikers laag voor laag grafische elementen toevoegen en zo gedetailleerde en aanpasbare netwerkvisualisaties realiseren. Deze aanpak stelt onderzoekers in staat om zowel de structuur als de eigenschappen van netwerken inzichtelijk te maken en te communiceren.

Laat ons starten met een eenvoudige plot:

```r
library(ggraph)
```

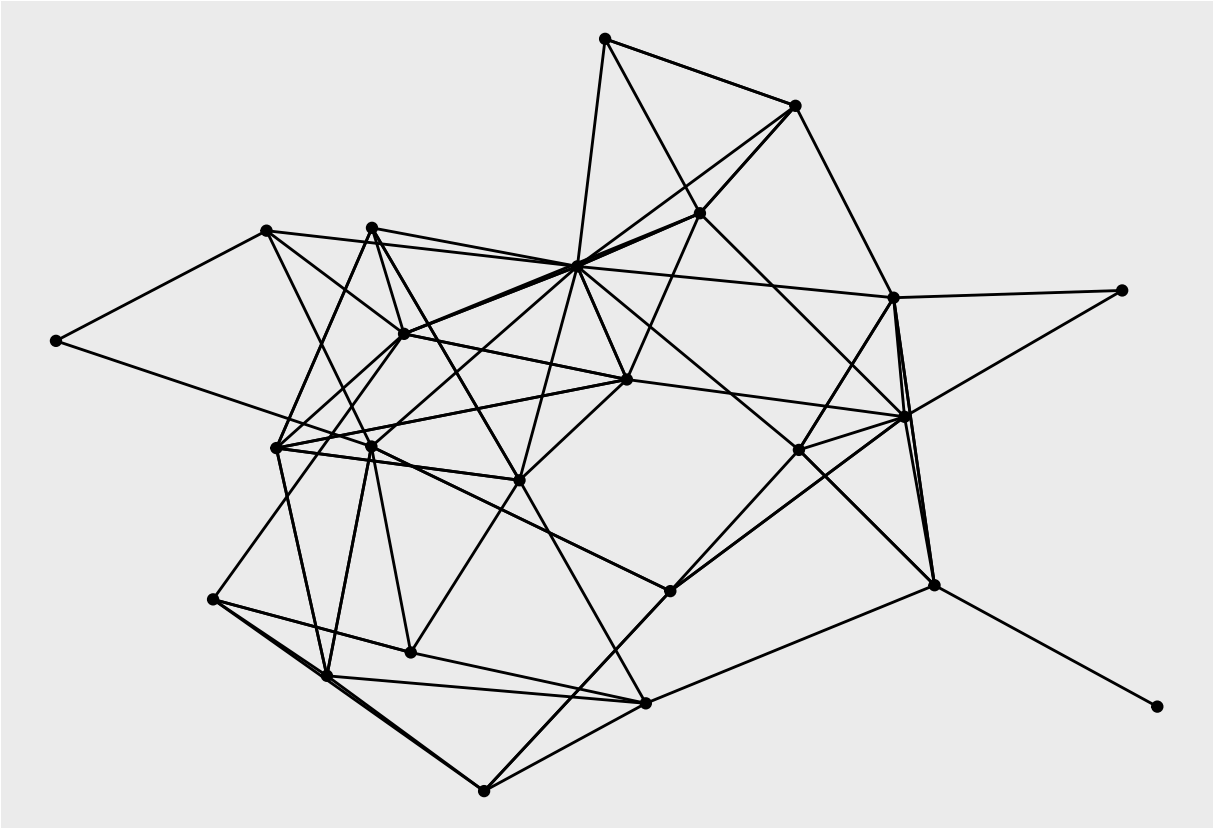
```
## Warning: package 'ggraph' was built under R version 4.3.3
```

```r
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.3.3

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:network':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges, delete.vertices,
##     get.edge.attribute, get.edges, get.vertex.attribute, is.bipartite,
##     is.directed, list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
ggraph(class_matrix_net) +
  geom_edge_link() +   # add edges to the plot
  geom_node_point()    # add nodes to the plot
```

```
## Using "stress" as default layout
```

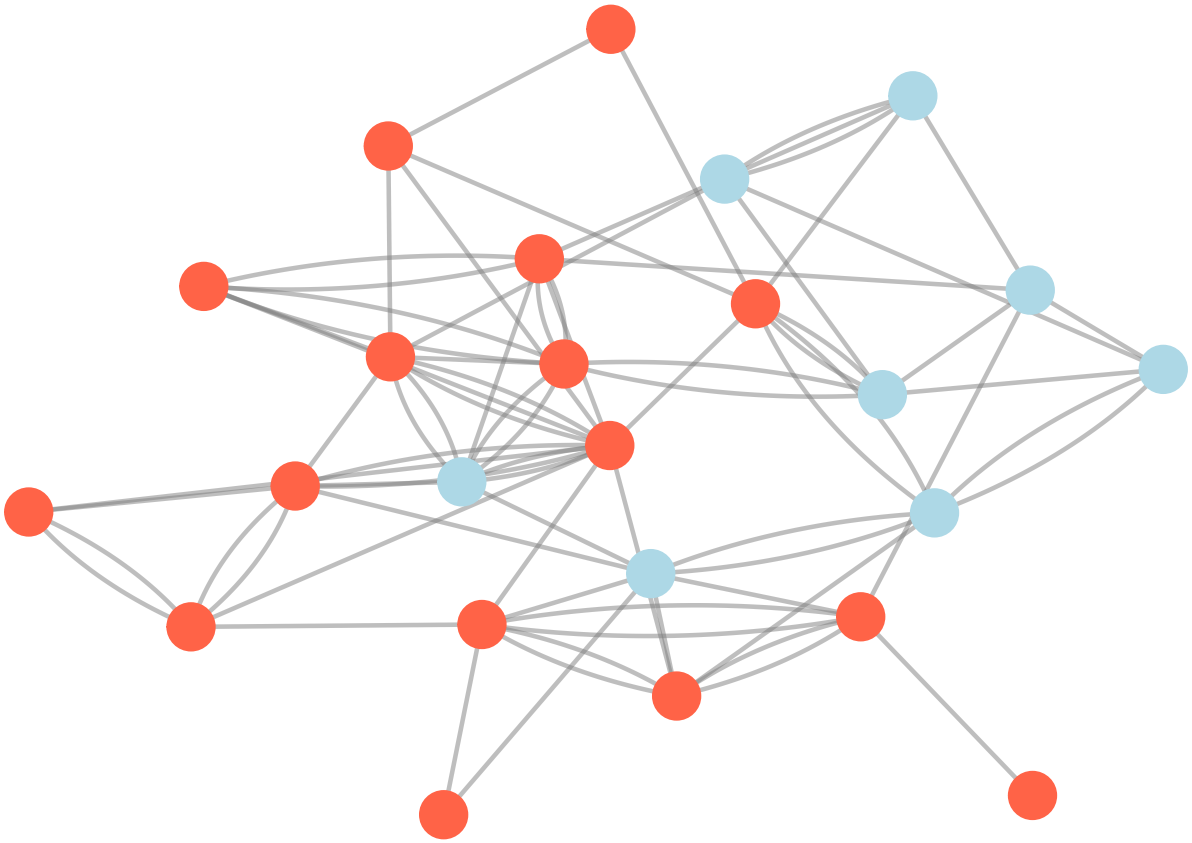


In `ggraph` kunnen we gebruikmaken van verschillende functies om de randen van het netwerk te visualiseren:

- `geom_edge_link()` voor rechte verbindingen,
- `geom_edge_arc()` voor gebogen lijnen,
- en `geom_edge_fan()` om overlappende multiplex-edges overzichtelijk uit te spreiden.

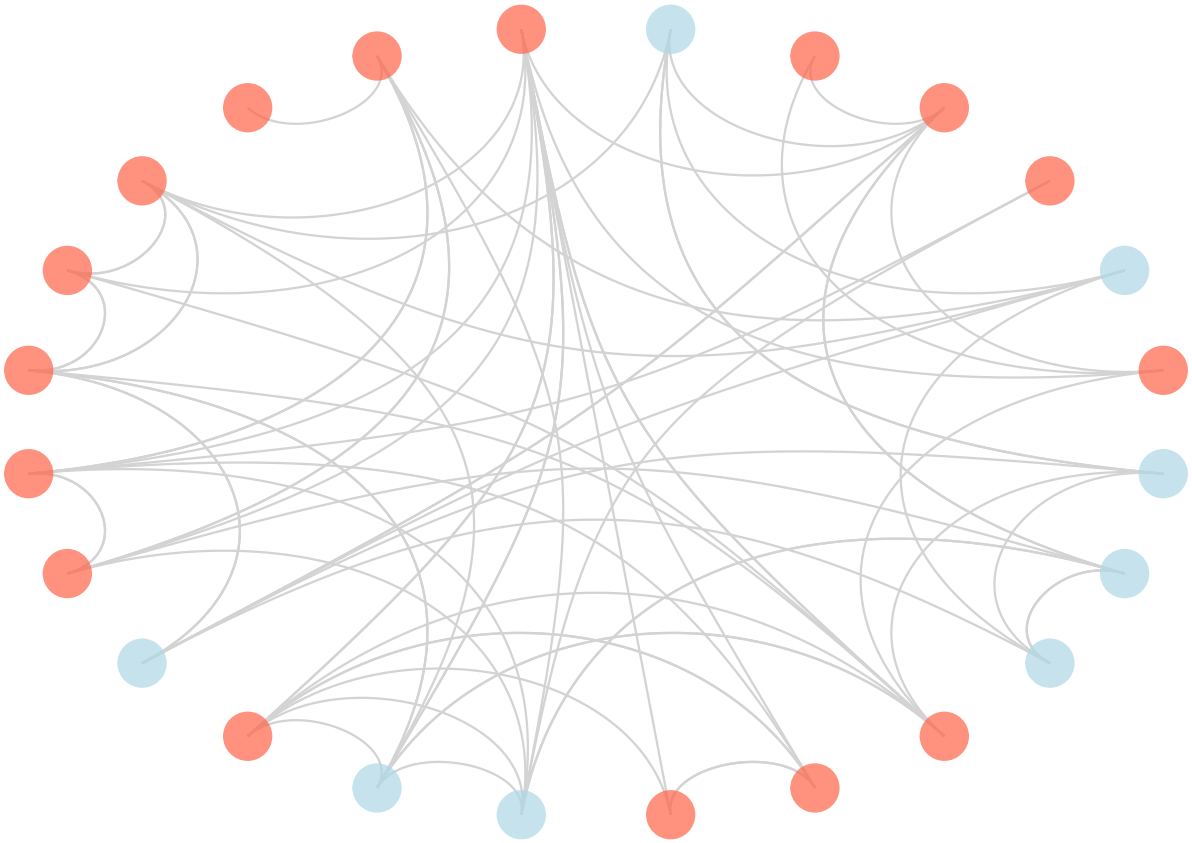
Net als bij andere pa ckageskunnen visuele eigenschappen van het netwerk worden ingesteld via belangrijke functiewaarden. Zo kunnen nodes worden aangepast qua kleur, vulkleur, vorm, grootte en randdikte. Edges kunnen worden ingesteld op kleur, dikte en lijntype. Daarnaast regelt de parameter `alpha` de transparantie van deze elementen.

```
colrs <- c("Male" = "lightblue", "Female" = "tomato")
gender_vals <- V(class_matrix_net)$gender
V(class_matrix_net)$color <- colrs[as.character(gender_vals)]

ggraph(class_matrix_net, layout="gem") +
  geom_edge_fan(color="gray50", width=0.8, alpha=0.5) +
  geom_node_point(color=V(class_matrix_net)$color, size=8) +
  theme_void()
```



Net zoals in `ggplot2` kunnen we verschillende thema's aan een plot toevoegen. Voor een strakkere en overzichtelijke uitstraling kun je gebruikmaken van een minimalistisch of leeg thema, zoals `theme_minimal()` of `theme_void()`.

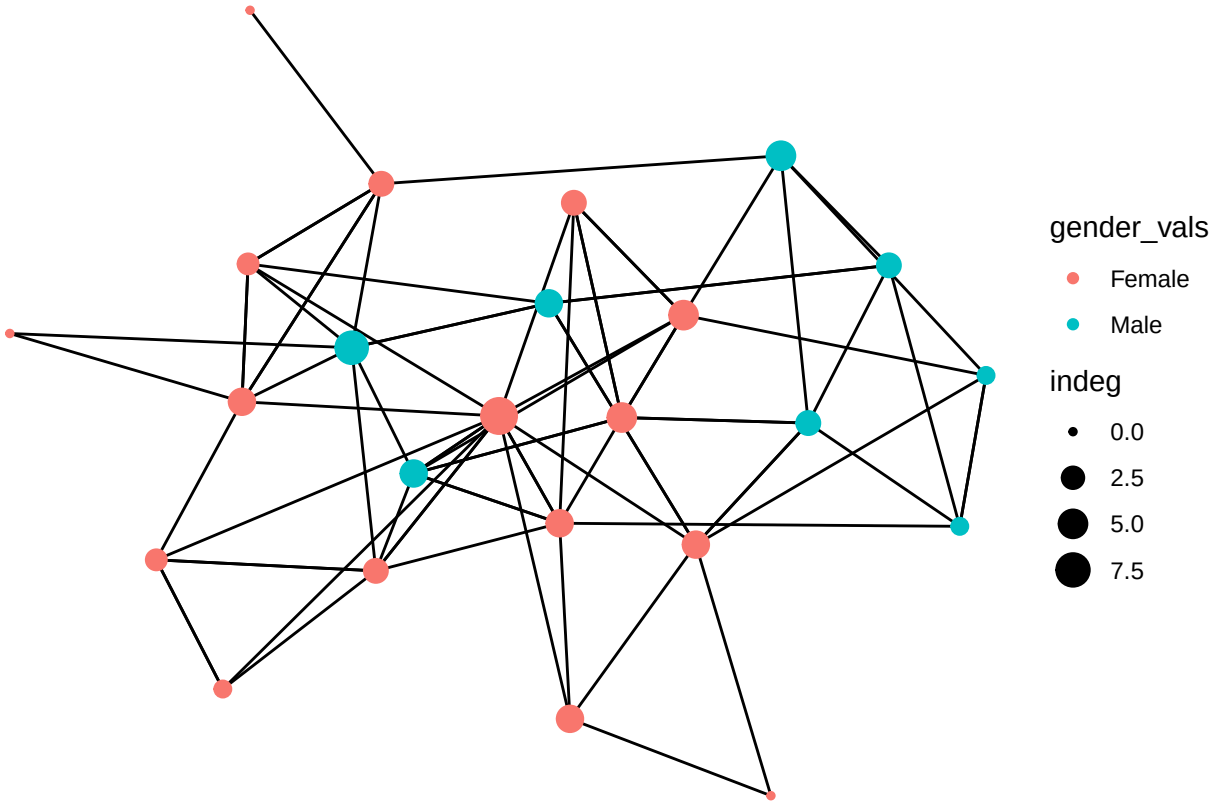
```
ggraph(class_matrix_net, layout = 'linear', circular = TRUE) +
    geom_edge_arc(color = "lightgray", width=0.4) +
    geom_node_point(color=V(class_matrix_net)$color, size=8, alpha = 0.7) +
    theme_void()
```



Het `ggraph`-package gebruikt eveneens de traditionele `ggplot2`-manier van het mappen van aesthetics: dat wil zeggen het specificeren welke elementen uit de data corresponderen met verschillende visuele eigenschappen van de grafiek.

Dit gebeurt met de functie `aes()`, waarmee visuele parameters worden gekoppeld aan attribuutnamen uit de data.

```
ggraph(class_matrix_net, layout = "gem") +
  geom_edge_link(aes(color = class_matrix_net$gender)) +
  geom_node_point(aes(size = indeg, color = gender_vals)) +
  theme_void()
```



Einde van deze notebook