Reproducibility of scientific articles

Aleksander Podsiad

March 28, 2020

Contents

Introduction

In this report, I intend to try to reproduce three scientific articles on visualisation in R. I am going to check if the graphs are created correctly and all packages or data sources are still available.

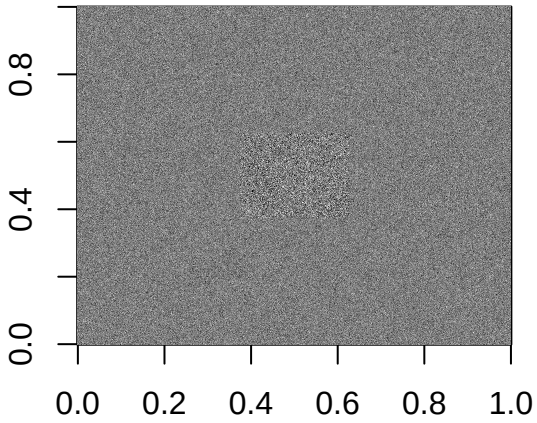
A Multiscale Test of Spatial Stationarity for Textured Images

First article is about `LS2Wstat` package (Taylor, Eckley and Nunes, 2014).
Link to article: `https://doi.org/10.32614/RJ-2014-002`

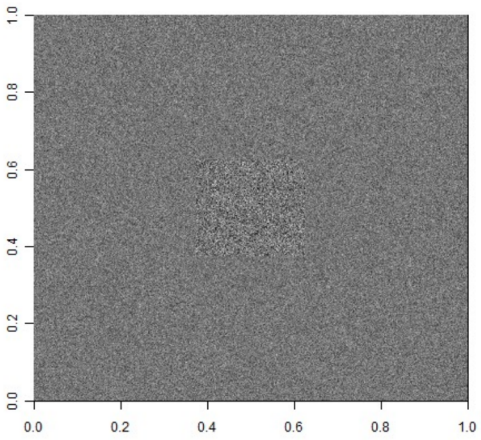
Simulating textured images with LS2Wstat

Function call

```
library("LS2Wstat")
library(dplyr)
set.seed(1)
X <- simTexture(n = 512, K = 1, imtype = "NS5", sd = 1.6, prop = 0.25)[[1]]
image(plotmtx(X), col = grey(255:0/256))
```



Original image



It is actually quite accurate. Reproduced correctly.

Testing the spatial stationarity of images

Function cddews

```
TSvalue <- avespecvar(cddews(X, smooth = FALSE))
```

```
## Took  4.707 seconds
```

```
TSvalue
```

```
## [1] 0.2044345
```

```
# 0.2044345
```

This one is also reproduced correctly.

Function TOS2D

Function call

```
Xbstest <- TOS2D(X, nsamples = 100)
```

Results

```
pval <- getpval(Xbstest$samples)
```

```
## Observed bootstrap is  0.204
## p-value is  0.00990099
```

```
summary(Xbstest)
```

```
##
## 2D bootstrap test of stationarity
##       object of class TOS2D
## ---------------------------------
##
## summary
## =======
## data: X
## Observed test statistic: 0.204
## bootstrap p-value: 0.01
```

```
print(Xbstest)
```

```
##
## 2D bootstrap test of stationarity
##       object of class TOS2D
## ---------------------------------
##
## summary
## =======
## data: X
## Observed test statistic: 0.204
## bootstrap p-value: 0.01
##
## Number of bootstrap realizations: 100
## spectral statistic used: avespecvar
```

Other textured images

Error

This line produces error:

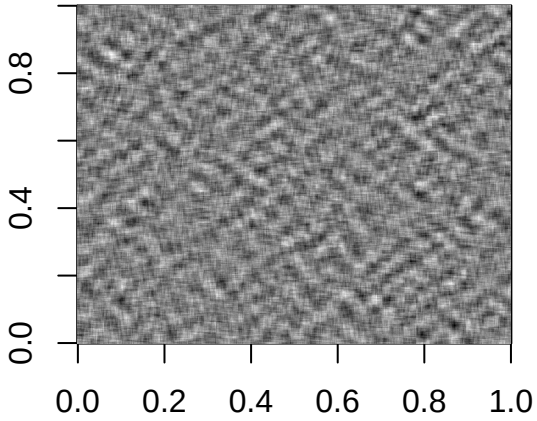
```
Haarimage <- simTexture(512, K = 1, imtype = "S5")[[1]]
```

```
## Error in simTexture(512, K = 1, imtype = "S5")[[1]]: indeks jest poza granicami
```

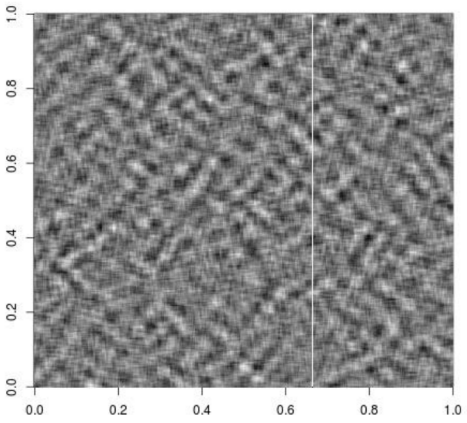
Firstly it seems not to be reproducible, but after some research I found out that the method `"S5"` is no longer available for use in the function call. I should use method `"S4"` instead. However this is not a good sign for the article reproducibility.

Function call

```
Haarimage <- simTexture(512, K = 1, imtype = "S4")[[1]]
image(plotmtx(Haarimage), col = grey(255:0/256))
```



Original image



It is also quite accurate. Reproduced (despite problems).

Function TOS2D

```
Haarimtest <- TOS2D(Haarimage, smooth = FALSE, nsamples = 100)
```

Results

```
summary(Haarimtest)
```

```
##
## 2D bootstrap test of stationarity
##       object of class TOS2D
## ---------------------------------
##
## summary
## =======
## data: Haarimage
## Observed test statistic: 0.631
## bootstrap p-value: 0.673
```

That is not what we wanted (missing two lines), but it can be easily produced by calling `print` function. Probably the article author made mistake here.

```
print(Haarimtest)
```

```
##
## 2D bootstrap test of stationarity
##       object of class TOS2D
## ---------------------------------
##
## summary
## =======
## data: Haarimage
## Observed test statistic: 0.631
## bootstrap p-value: 0.673
##
## Number of bootstrap realizations: 100
## spectral statistic used: avespecvar
```

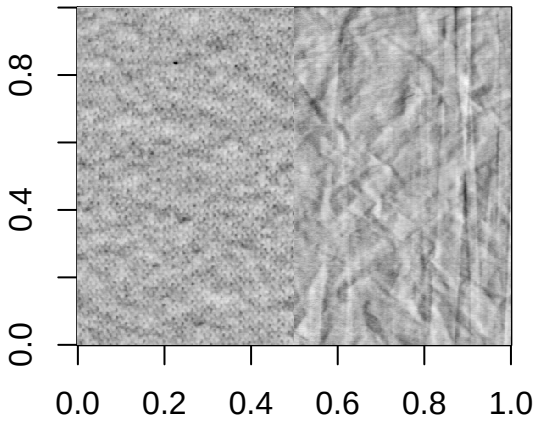
Another image

Create image

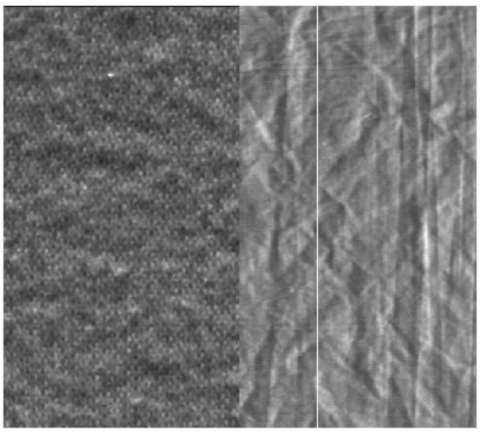
```
data(textures)
montage1 <- cbind(A[1:512, 1:256], B[, 1:256])
montage1zm <- medpolish(montage1)$residuals
```

```
## 1: 15350.79
## Final: 15276.89
```

```
image(plotmtx(montage1zm), col = grey(255:0/256))
```



Original image



These images also look very similar, however the original ones are always darker.

Quadtree decomposition

Function call

```
QTdecX <- imageQT(X, test = TOS2D, nsamples = 100)
```

Results

```
print(QTdecX)
```

```
##
## 2D quadtree decomposition
##  object of class imageQT
## ---------------------------------
##
## summary
## =======
## data: X
## Indices of non-stationary sub-images:
##  0 1 2 3 03 12 21 30
## Indices of stationary sub-images: 00 01 02 10 11 13 20 22 23 31 32 33 030 031 032 033 120 121 122 123
```

```
##
## minimum testing region: 64
```

```
QTdecX$resl
```

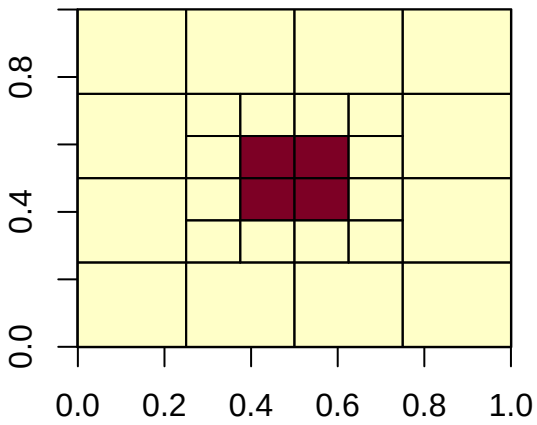
```
## [[1]]
## [1] FALSE
##
## [[2]]
## [1] FALSE FALSE FALSE FALSE
##
## [[3]]
##  [1]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE
## [12]  TRUE FALSE  TRUE  TRUE  TRUE
##
## [[4]]
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [15] TRUE TRUE
```

Plotting a quadtree decomposition

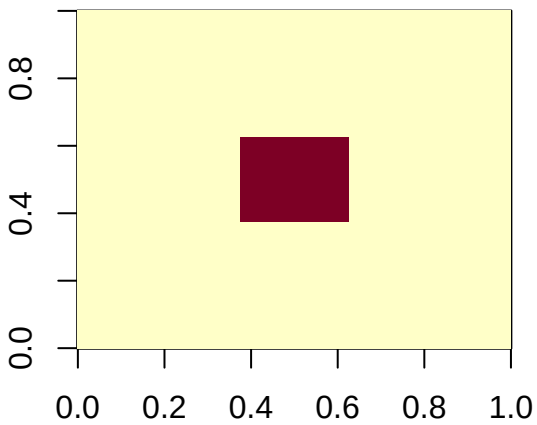
```
texclass <- c(rep(1, times = 15), rep(c(2, 1, 1), times = 4), 1)
texclass
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 2 1 1 2 1 1 1
```

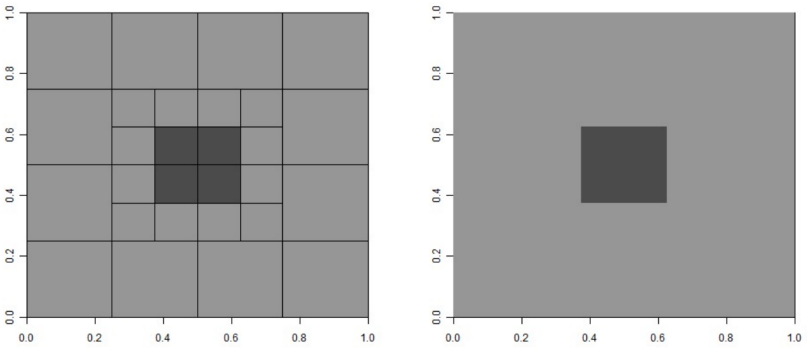
```
plot(QTdecX, texclass, class = TRUE, QT = TRUE)
```




```
plot(QTdecX, texclass, class = TRUE, QT = FALSE)
```



Original images



These images are not identical to original ones, but the whole concept is preserved. Reproduced.

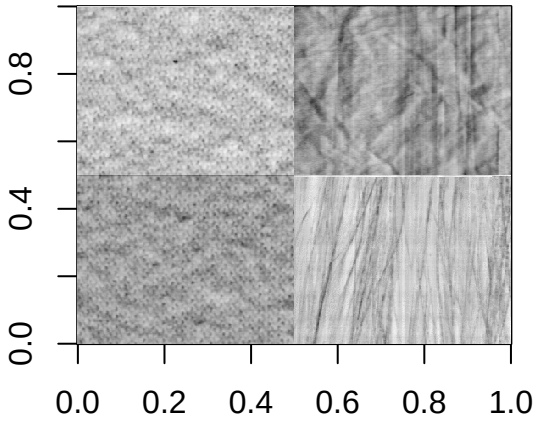
Image montage

Function call

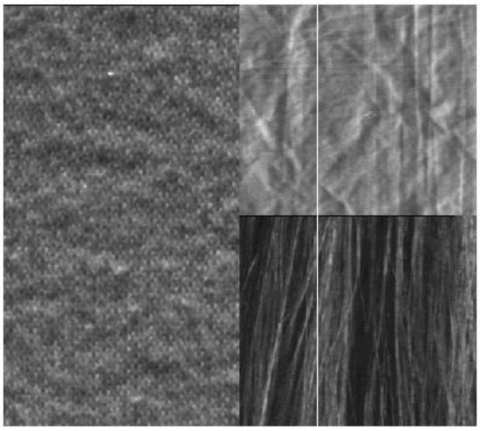
```
data(textures)
Y <- cbind(A[1:512, 1:256], rbind(B[1:256, 1:256], C[1:256, 1:256]))
Yzm <- medpolish(Y)$residuals
```

```
## 1: 21038.45
## Final: 21010.62
```

```
image(plotmtx(Yzm), col = grey(255:0/256))
```



Original image



Function imageQT

```
QTdecYzm <- imageQT(Yzm, test = TOS2D, nsamples = 100)
```

```
print(QTdecYzm)
```

```
##
## 2D quadtree decomposition
##  object of class imageQT
## ---------------------------------
##
## summary
## =======
## data: Yzm
## Indices of non-stationary sub-images:
##  none
## Indices of stationary sub-images: 0 1 2 3
##
## minimum testing region: 64
```

Summary

This article turned out to be almost fully reproducible, despite minor complications.

Network Visualization with ggplot2

Second article (Tyner, Briatte and Hofmann, 2017) is about network visualisation using three methods:

- function `ggnet2` from the `GGally` package;
- package `geomnet`;
- package `ggnetwork`

Link to article: `https://doi.org/10.32614/RJ-2017-023`

Mad Men Relationship Network Example

Loading packages:

```
library(network)
library(sna)
library(GGally)
library(geomnet)
library(ggnetwork)
library(igraph)
```

geomnet

```
data(madmen, package = "geomnet")
MMnet <- fortify(as.edgedf(madmen$edges), madmen$vertices)
set.seed(10052016)
ggplot(data = MMnet, aes(from_id = from_id, to_id = to_id)) +
  geom_net(aes(colour = Gender), layout.alg = "kamadakawai",
           size = 2, labelon = TRUE, vjust = -0.6, ecolour = "grey60",
           directed =FALSE, fontsize = 3, ealpha = 0.5) +
  scale_colour_manual(values = c("#FF69B4", "#0099ff")) +
  xlim(c(-0.05, 1.05)) +
  theme_net() +
  theme(legend.position = "bottom")
```

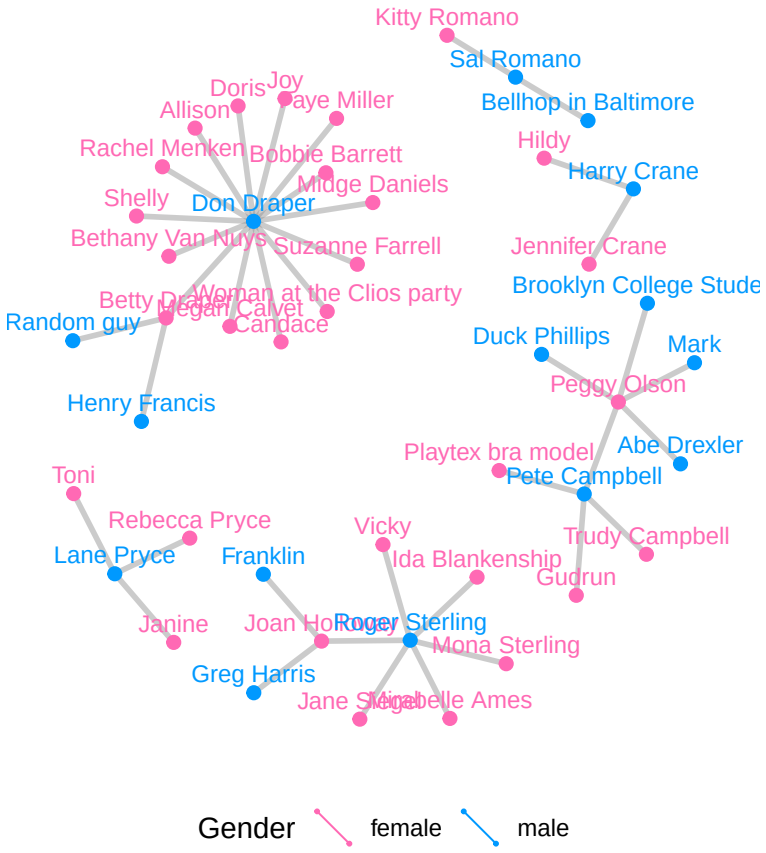


Figure 1: Mad Men relationship network included in the `gcookbook` package by Winston Chang visualized using `geomnet`.

ggnet2

```
mm.net <- network::network(madmen$edges[, 1:2], directed = FALSE)
rownames(madmen$vertices) <- madmen$vertices$label
mm.net %v% "gender" <- as.character(
  madmen$vertices[ network.vertex.names(mm.net), "Gender"]
)
mm.col <- c("female" = "#ff69b4", "male" = "#0099ff")
set.seed(10052016)
ggnet2(mm.net, color = mm.col[ mm.net %v% "gender" ],
       label = TRUE, label.color = mm.col[ mm.net %v% "gender" ],
       size = 2, vjust = -0.6, mode = "kamadakawai", label.size = 3)
```

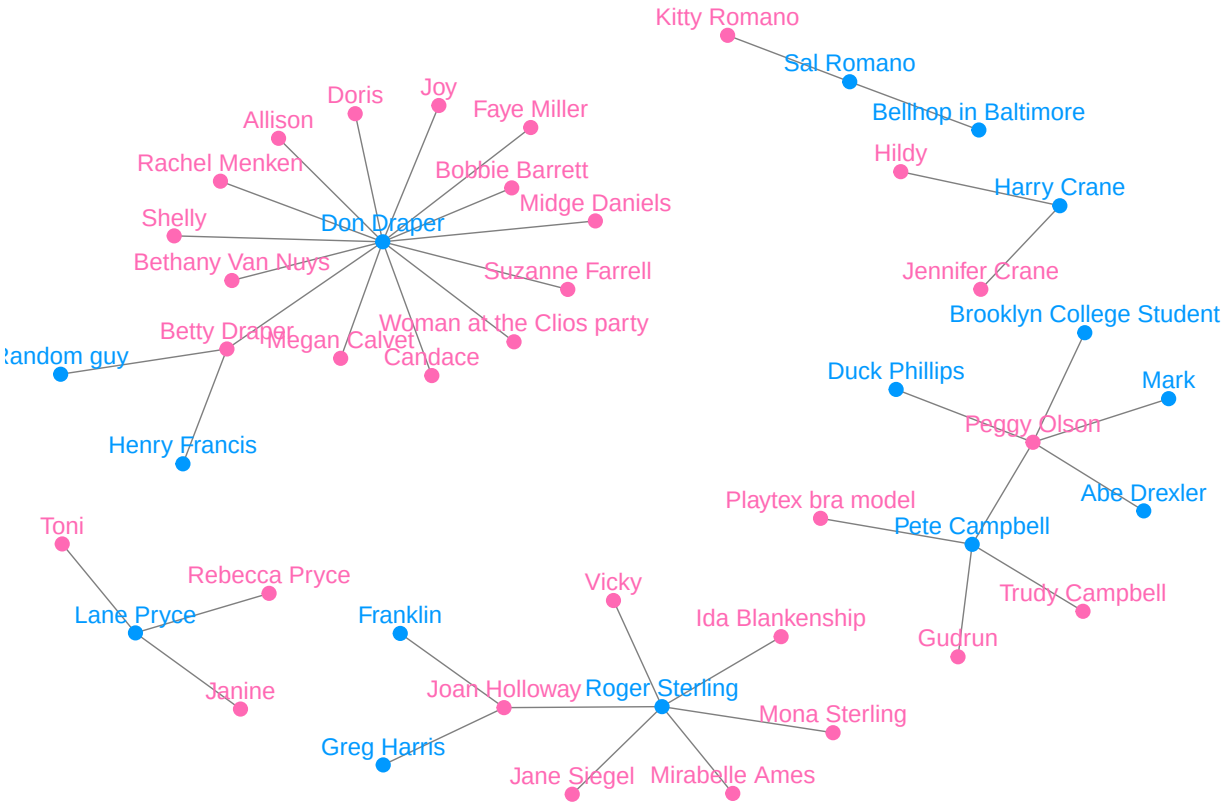


Figure 2: Mad Men example using the `ggnet2` function in the `GGally` package.

ggnetwork

```
set.seed(10052016)
ggplot(data = ggnetwork(mm.net, layout = "kamadakawai"),
       aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "grey50") +
  geom_nodes(aes(colour = gender), size = 2) +
  geom_nodetext(aes(colour = gender, label = vertex.names),
                size = 3, vjust = -0.6) +
  scale_colour_manual(values = mm.col) +
  xlim(c(-0.05, 1.05)) +
  theme_blank() +
```

```
  theme(legend.position = "bottom")
```

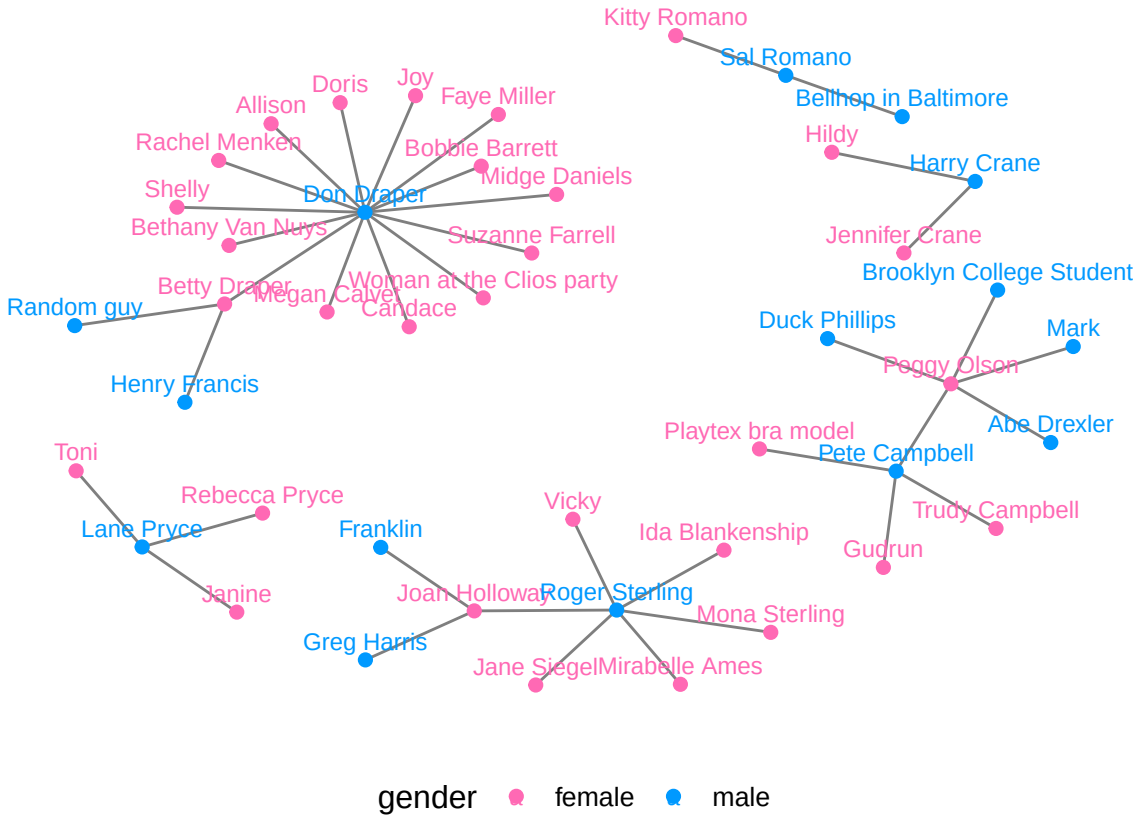


Figure 3: Mad Men example using the `ggnetwork` package.

Blood Donation Example

Blood donation "network": which blood types can give and receive?

ggnet2

```
data(blood, package = "geomnet")
set.seed(12252016)
ggnet2(network::network(blood$edges[, 1:2], directed=TRUE),
       mode = "circle", size = 15, label = TRUE,
       arrow.size = 10, arrow.gap = 0.05, vjust = 0.5,
       node.color = "darkred", label.color = "grey80")
```

geomnet

```
set.seed(12252016)
ggplot(data = blood$edges, aes(from_id = from, to_id = to)) +
  geom_net(colour = "darkred", layout.alg = "circle", labelon = TRUE, size = 15,
           directed = TRUE, vjust = 0.5, labelcolour = "grey80",
           arrowsize = 1.5, linewidth = 0.5, arrowgap = 0.05,
           selfloops = TRUE, ecolour = "grey40") +
  theme_net()
```

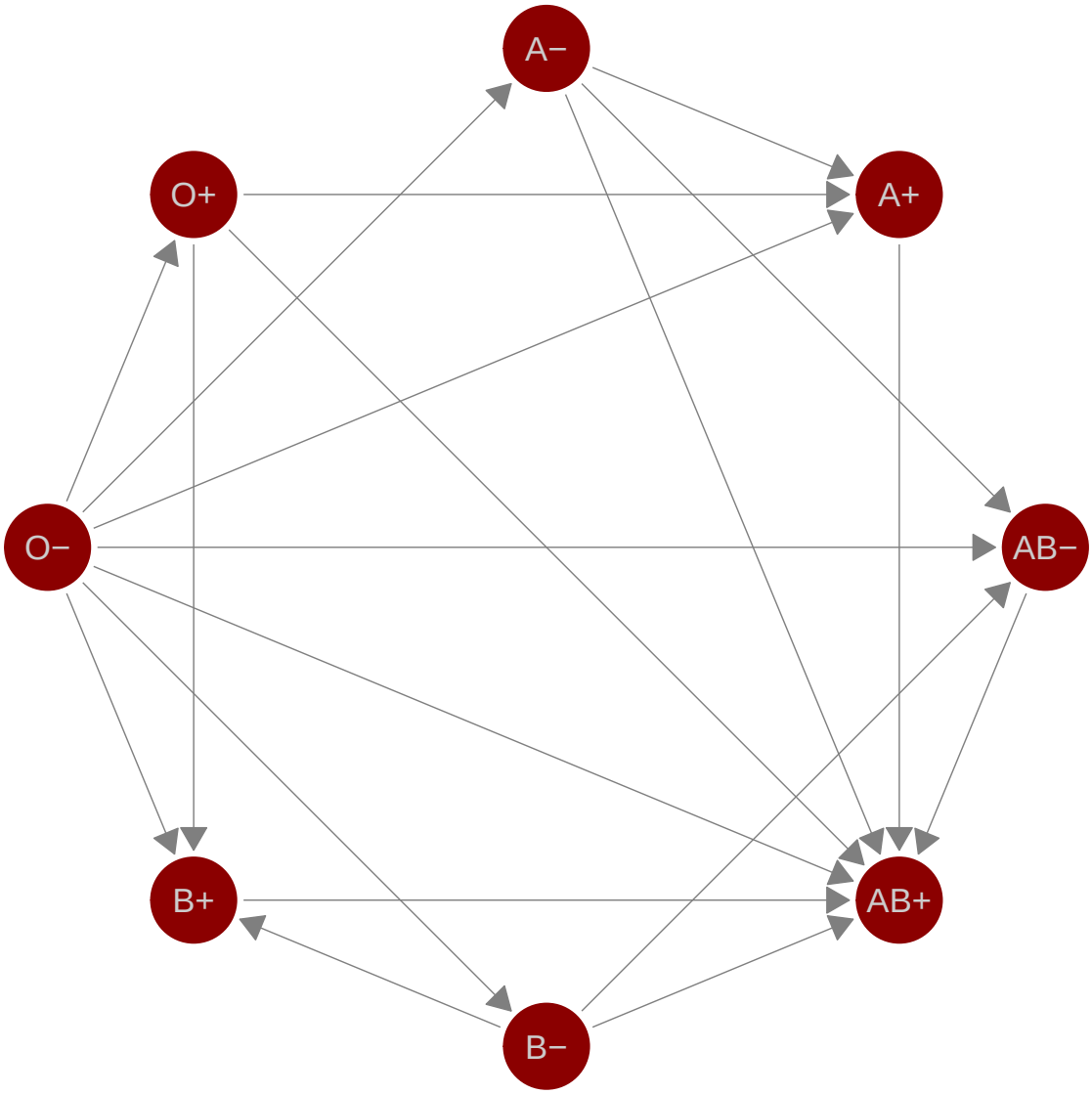


Figure 4: `ggnet` implementation

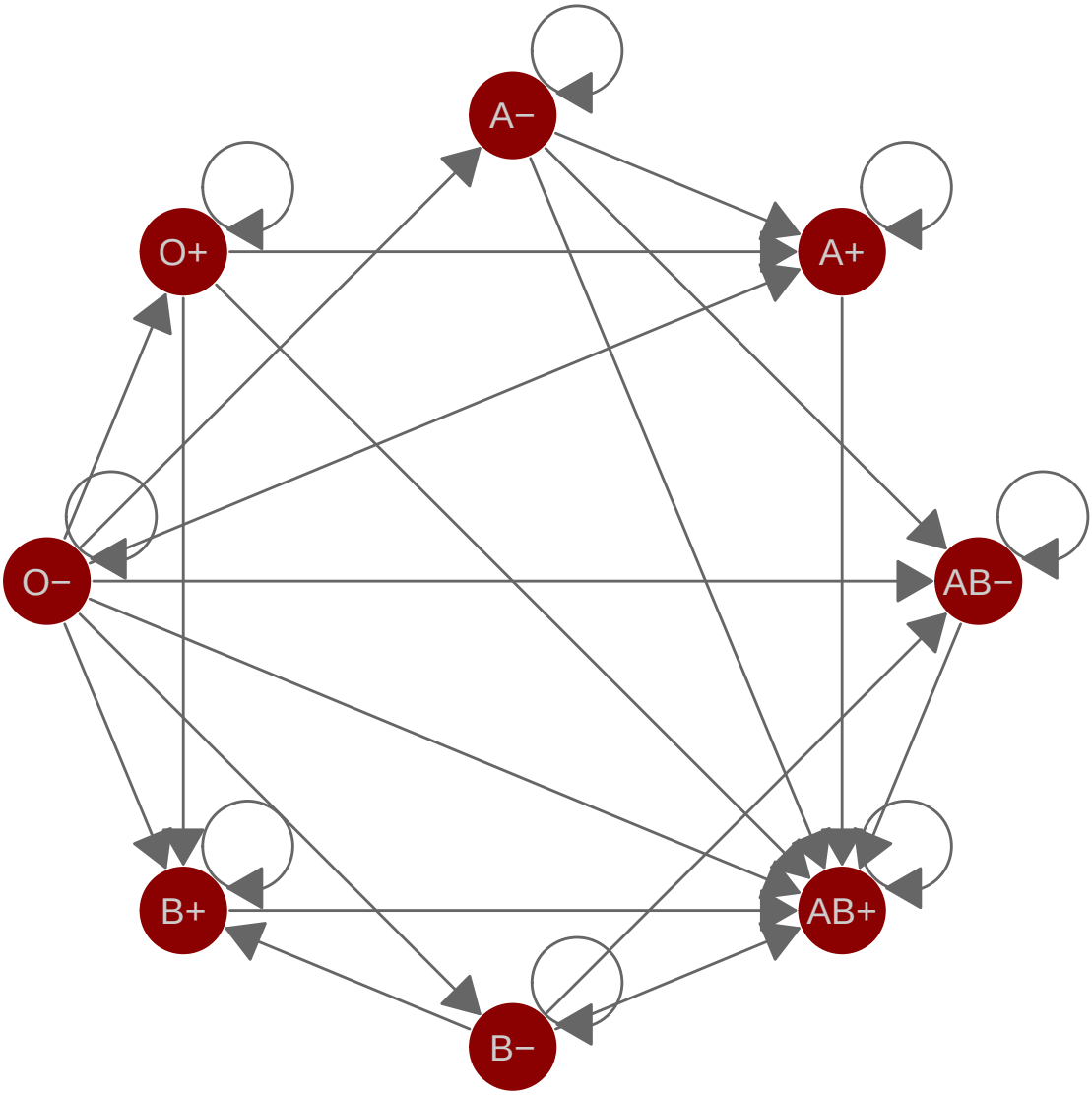


Figure 5: `geom_net` implementation

ggnetwork

```
set.seed(12252016)
ggplot(ggnetwork(network::network(blood$edges[, 1:2]),
                 layout = "circle", arrow.gap = 0.05),
       aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "grey50",
             arrow = arrow(length = unit(10, "pt"), type = "closed")) +
  geom_nodes(size = 15, color = "darkred") +
  geom_nodetext(aes(label = vertex.names), color = "grey80") +
  theme_blank()
```

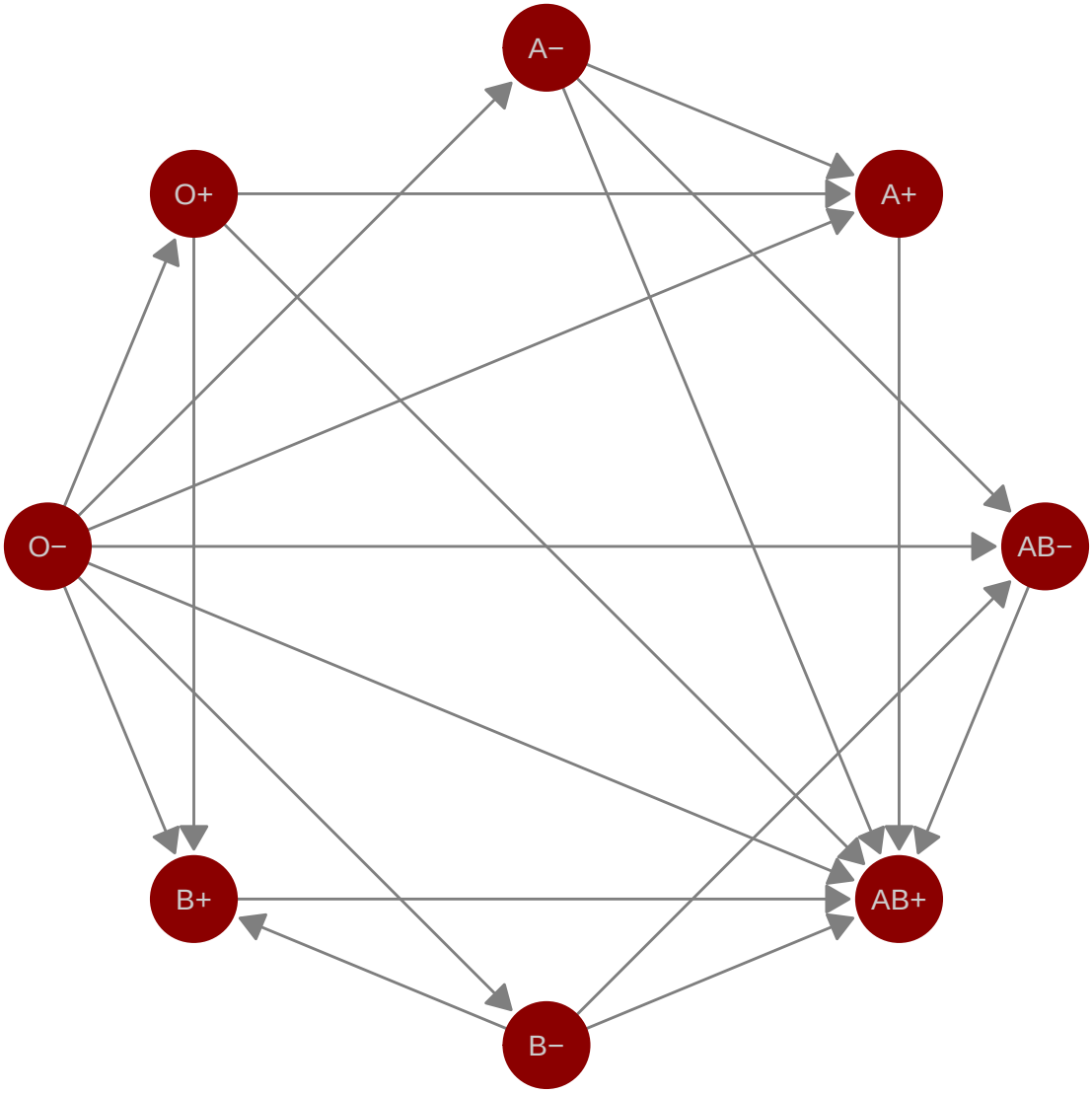


Figure 6: `ggnetwork` implementation

Email Network Example

A faux company's email network provided by the 2014 VAST Challenge.

ggnet2

```
data(email, package = 'geomnet')
em.cet <- as.character(
  email$nodes$CurrentEmploymentType)
names(em.cet) = email$nodes$label
edges <- subset(email$edges, nrecipients < 54)

em.net <- edges[, c("From", "to") ]
em.net <- network::network(em.net, directed = TRUE)
em.net %v% "curr_empl_type" <-
  em.cet[ network.vertex.names(em.net) ]
set.seed(10312016)
ggnet2(em.net, color = "curr_empl_type",
       size = 4, palette = "Set1",
       arrow.size = 5, arrow.gap = 0.02,
       edge.alpha = 0.25, mode = "fruchtermanreingold",
       edge.color = c("color", "grey50"),
       color.legend = "Employment Type") +
  theme(legend.position = "bottom")
```

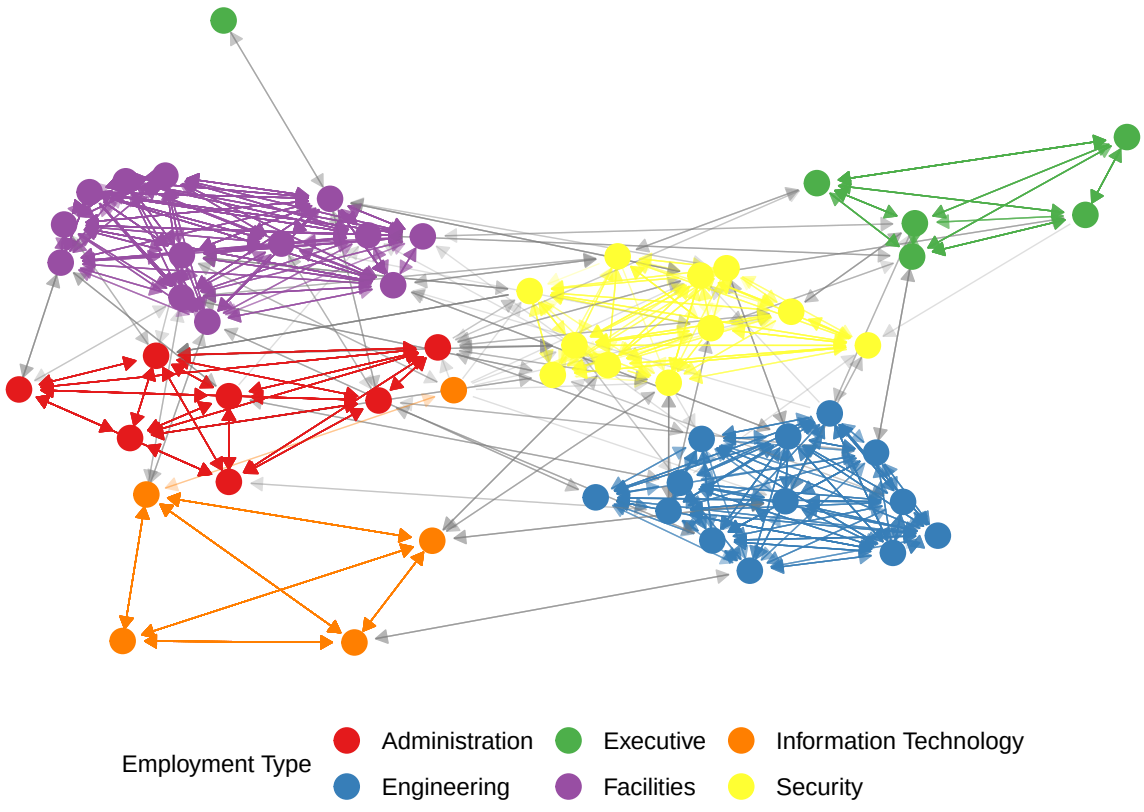


Figure 7: The company's email network visualized with `ggnet2`.

geomnet

```
email$edges <- email$edges[, c(1,5,2:4,6:9)]
emailnet <- fortify(
```

```
  as.edgedf(subset(email$edges, nrecipients < 54)),
  email$nodes)
set.seed(10312016)
ggplot(data = emailnet,
       aes(from_id = from_id, to_id = to_id)) +
  geom_net(layout.alg = "fruchtermanreingold",
    aes(colour = CurrentEmploymentType,
        group = CurrentEmploymentType,
        linewidth = 3 * (...samegroup.. / 8 + .125)),
    ealpha = 0.25,
    size = 4, curvature = 0.05,
    directed = TRUE, arrowsize = 0.5) +
  scale_colour_brewer("Employment Type", palette = "Set1") +
  theme_net() +
  theme(legend.position = "bottom")
```

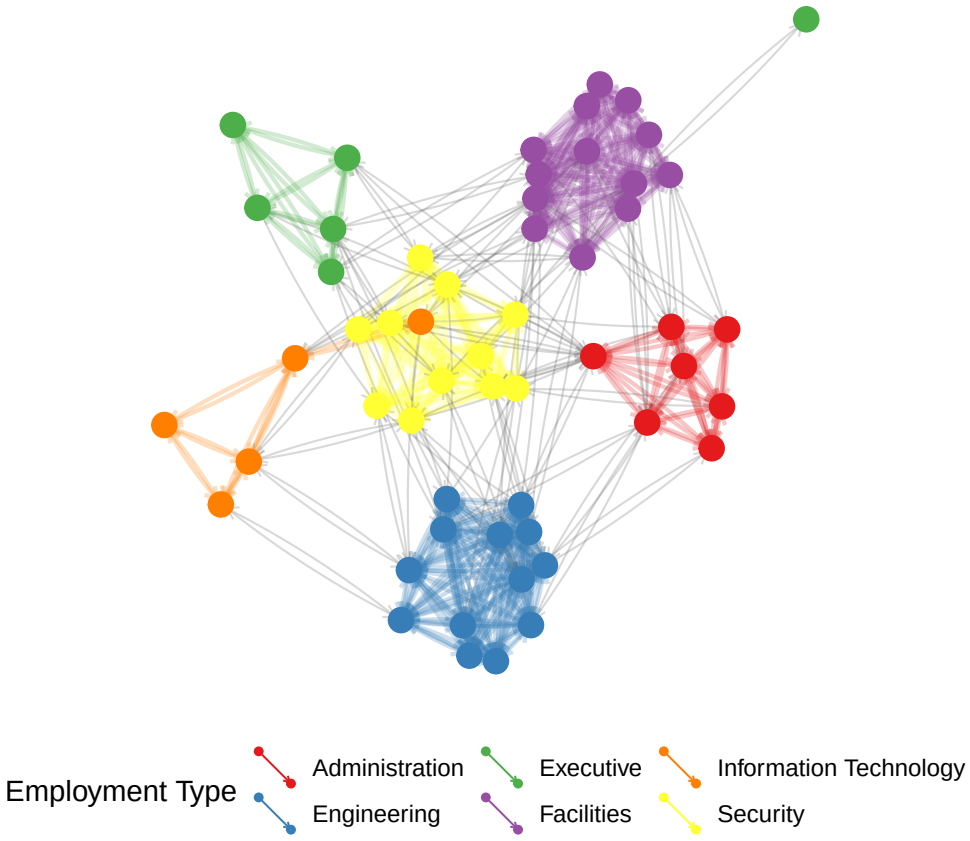


Figure 8: The company's email network visualized with `geomnet`.

ggnetwork

```
set.seed(10312016)
ggplot(ggnetwork(em.net, arrow.gap = 0.02, layout = "fruchtermanreingold"),
       aes(x, y, xend = xend, yend = yend)) +
  geom_edges(
    aes(color = curr_empl_type),
    alpha = 0.25,
```

```
    arrow = arrow(length = unit(5, "pt"),
                  type = "closed"),
    curvature = 0.05) +
  geom_nodes(aes(color = curr_empl_type),
             size = 4) +
  scale_color_brewer("Employment Type",
                     palette = "Set1") +
  theme_blank() +
  theme(legend.position = "bottom")
```

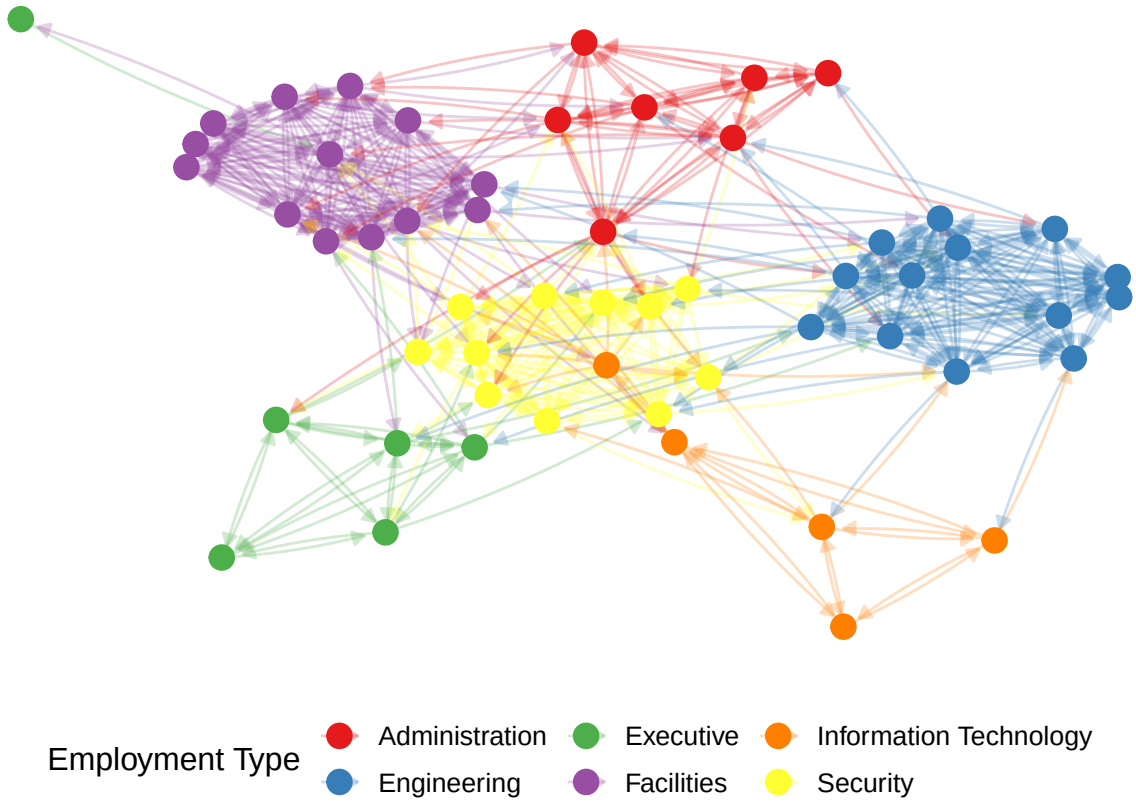


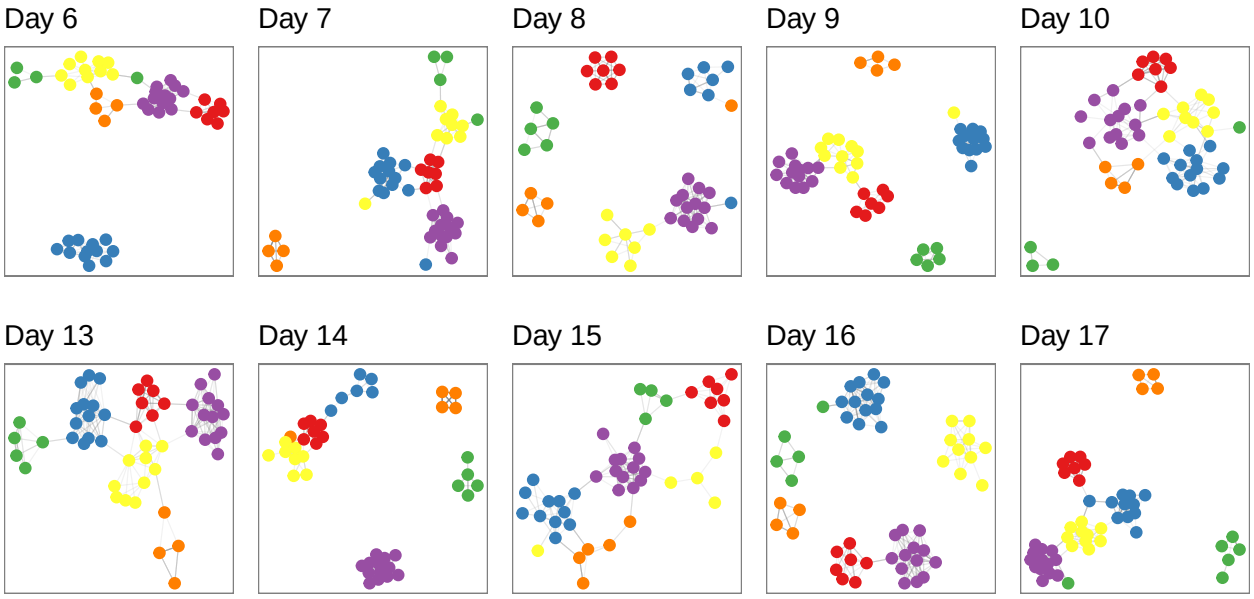
Figure 9: The company's email network visualized with `ggnetwork`.

Small Multiples Email Example

ggnet2

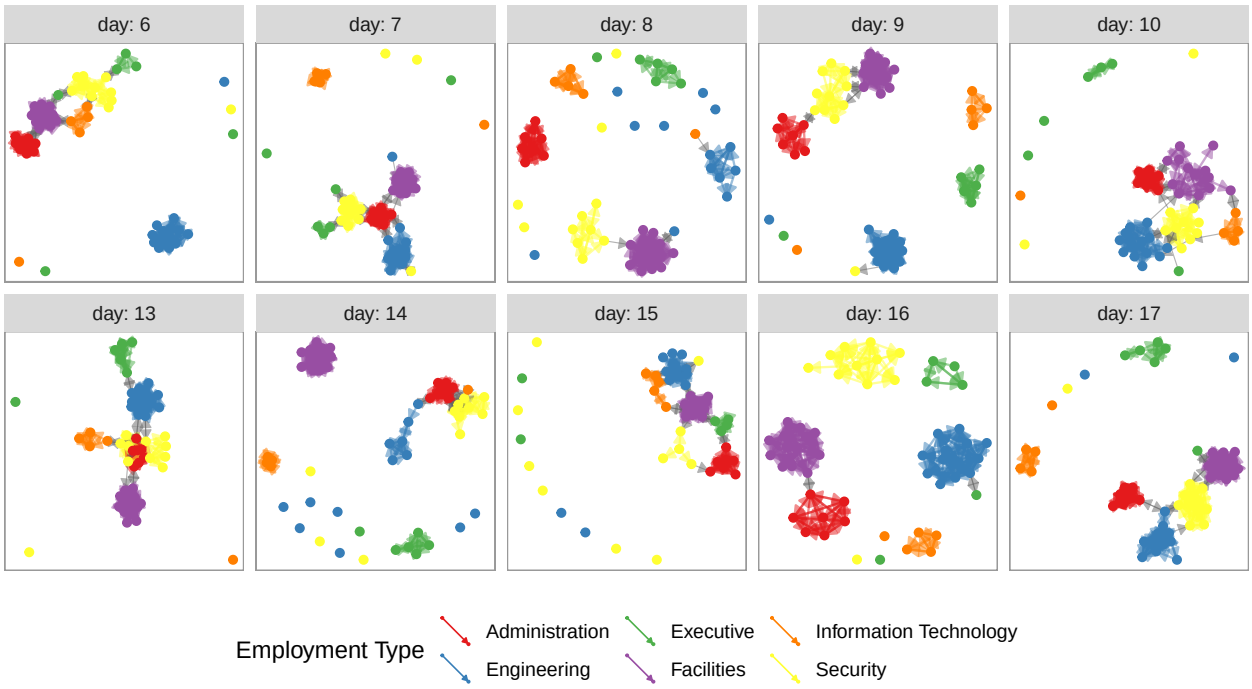
```
em.day <- subset(email$edges, nrecipients < 54)[, c("From", "to", "day") ]
em.day <- lapply(unique(em.day$day),
                 function(x) subset(em.day, day == x)[, 1:2 ])
em.day <- lapply(em.day, network, directed = TRUE)

for (i in 1:length(em.day)) {
  em.day[[ i ]] %v% "curr_empl_type" <-
    em.cet[ network.vertex.names(em.day[[ i ]]) ]
  em.day[[ i ]] %n% "day" <- unique(email$edges$day)[ i ]
}
```

```
g <- list(length(em.day))
set.seed(7042016)
for (i in 1:length(em.day)) {
  g[[ i ]] <- ggnet2(em.day[[ i ]], size = 2, color = "curr_empl_type",
                     palette = "Set1", arrow.size = 0, arrow.gap = 0.01,
                     edge.alpha = 0.1, legend.position = "none",
                     mode = "kamadakawai") +
    ggtitle(paste("Day", em.day[[ i ]] %n% "day")) +
    theme(panel.border = element_rect(color = "grey50", fill = NA),
          aspect.ratio = 1)
}
grid.arrange <- getFromNamespace("grid.arrange", asNamespace("gridExtra"))
grid.arrange(grobs = g, nrow = 2)
```



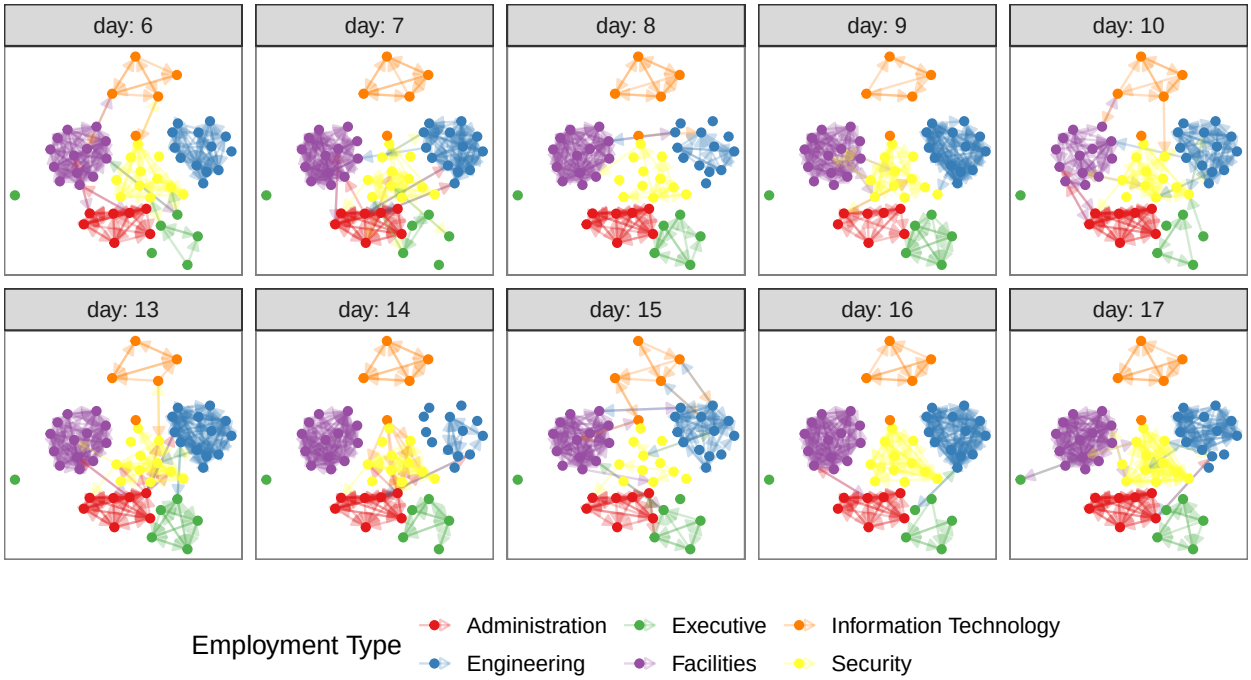
geomnet

```
emailnet <- fortify(as.edgedf(subset(email$edges, nrecipients < 54)), email$nodes, group = "day")

set.seed(7042016)
ggplot(data = emailnet, aes(from_id = from, to_id = to_id)) +
  geom_net(layout.alg = "kamadakawai",
    aes(colour = CurrentEmploymentType,
        group = CurrentEmploymentType,
        linewidth = 2 * (...samegroup.. / 8 + .125)),
        arrowsize = .5,
        directed = TRUE, fiteach = TRUE, ealpha = 0.5, size = 1.5, na.rm = FALSE) +
  scale_colour_brewer("Employment Type", palette = "Set1") +
  theme_net() +
  facet_wrap(~day, nrow = 2, labeller = "label_both") +
  theme(legend.position = "bottom",
        panel.border = element_rect(fill = NA, colour = "grey60"),
        plot.margin = unit(c(0, 0, 0, 0), "mm"))
```



ggnetwork

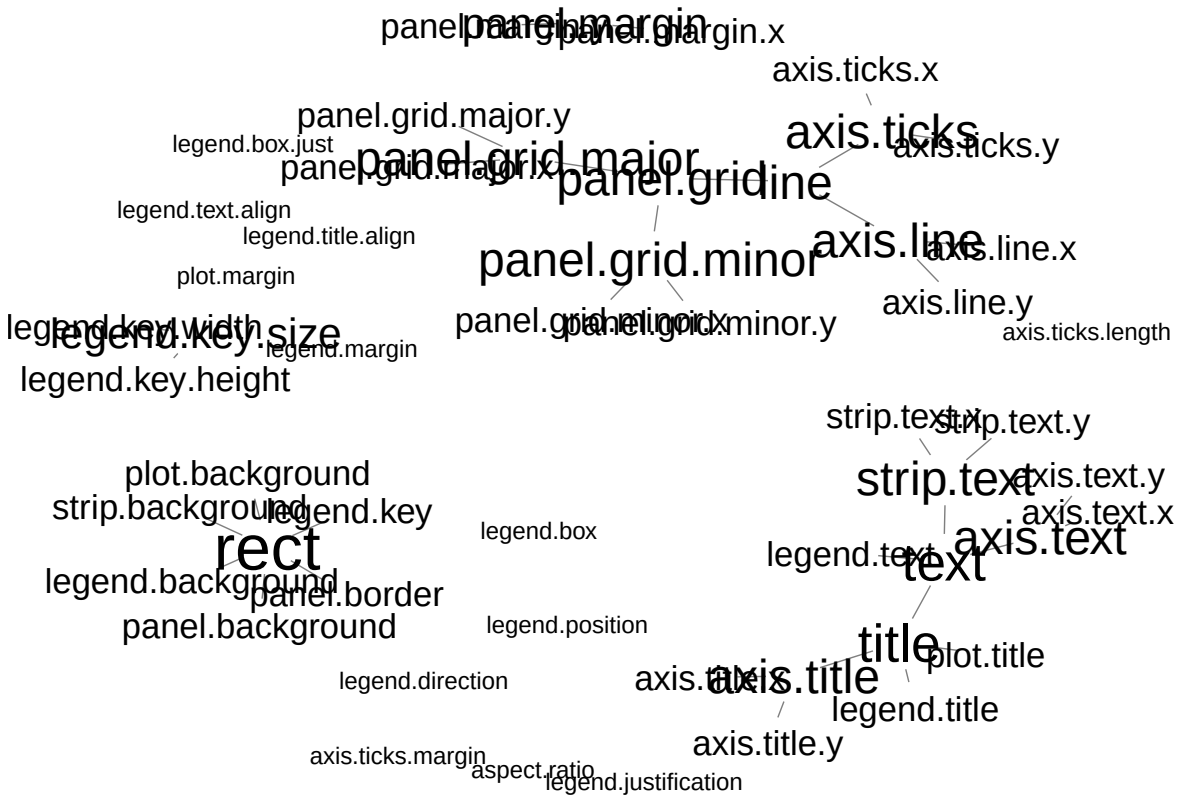
```
edges <- subset(email$edges, nrecipients < 54)
edges <- edges[, c("From", "to", "day") ]
em.net <- network::network(edges[, 1:2])
network::set.edge.attribute(em.net, "day", edges[, 3])
em.net %v% "curr_empl_type" <- em.cet[ network.vertex.names(em.net) ]

set.seed(7042016)
ggplot(ggnetwork(em.net, arrow.gap = 0.02, by = "day",
                 layout = "kamadakawai"),
       aes(x, y, xend = xend, yend = yend)) +
  geom_edges(
    aes(color = curr_empl_type),
    alpha = 0.25,
    arrow = arrow(length = unit(5, "pt"), type = "closed")) +
  geom_nodes(aes(color = curr_empl_type), size = 1.5) +
  scale_color_brewer("Employment Type", palette = "Set1") +
  facet_wrap(~day, nrow = 2, labeller = "label_both") +
  theme_facet(legend.position = "bottom")
```



Theme Element Inheritance Network

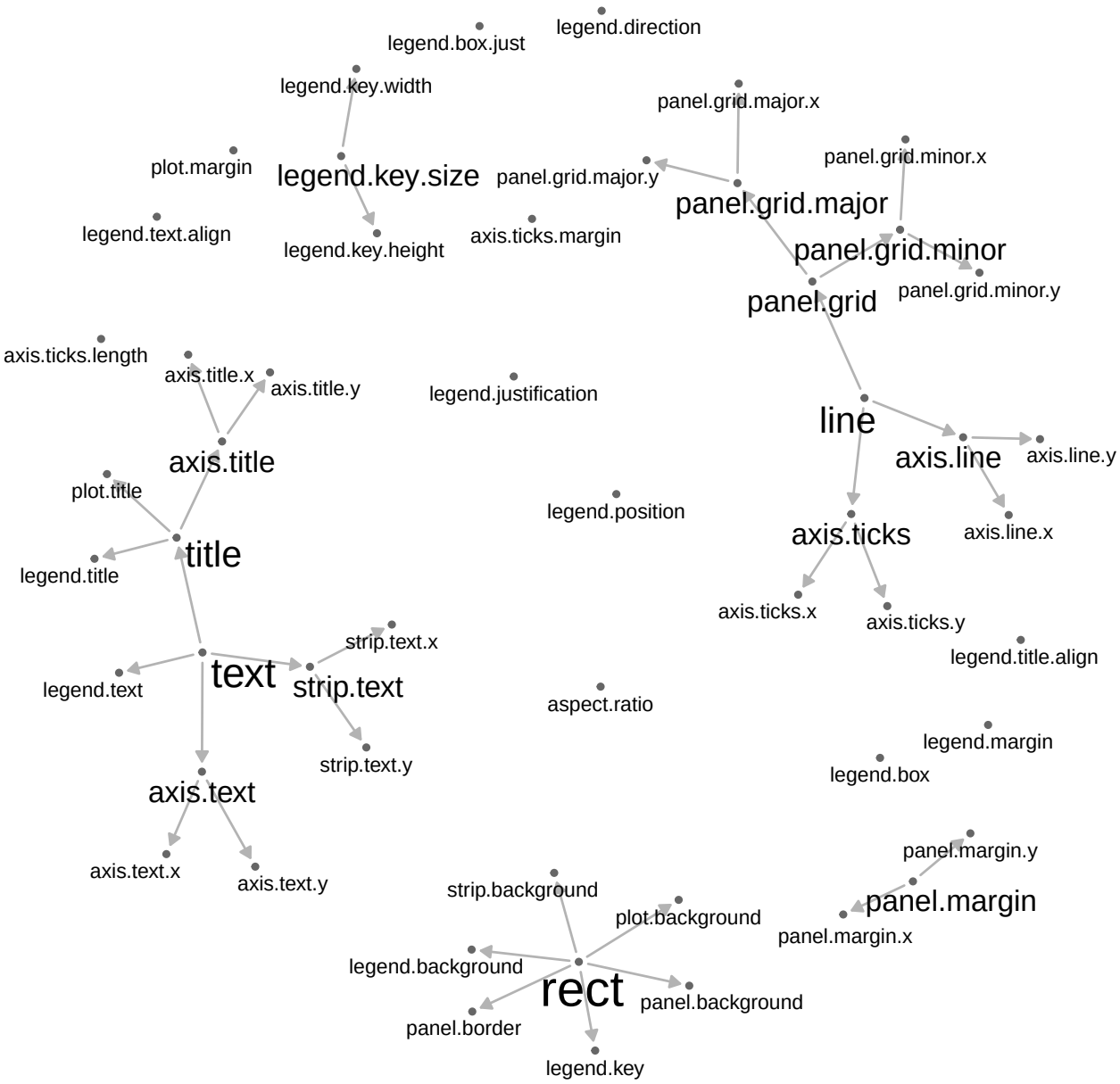
ggnet2

```
data(theme_elements, package = "geomnet")

te.net <- network::network(theme_elements$edges)

te.net %v% "size" <-
  sqrt(10 * (sna::degree(te.net) + 1))
set.seed(3272016)
ggnet2(te.net, label = TRUE, color = "white", label.size = "size",
       mode = "fruchtermanreingold", layout.exp = 0.15)
```



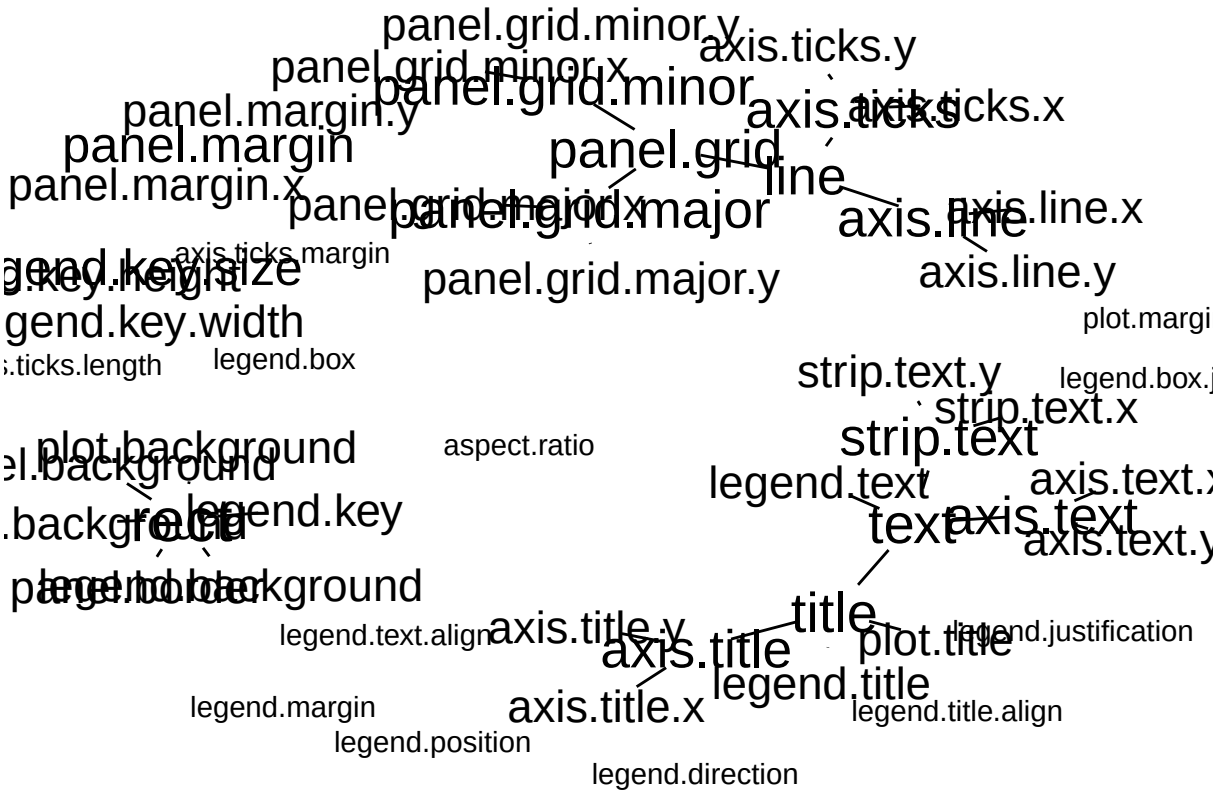
geomnet

```
TEnet <- fortify(as.edgedf(theme_elements$edges[,c("parent", "child")]), theme_elements$vertices)
TEnet <- TEnet %>%
  group_by(from_id) %>%
  mutate(degree = sqrt(10 * n() + 1))

set.seed(3272016)
ggplot(data = TEnet,
       aes(from_id = from_id, to_id = to_id)) +
  geom_net(layout.alg = "fruchtermanreingold",
    aes(fontsize = degree), directed = TRUE,
    labelon = TRUE, size = 1, labelcolour = 'black',
    ecolour = "grey70", arrowsize = 0.5,
    linewidth = 0.5, repel = TRUE) +
  theme_net() +
  xlim(c(-0.05, 1.05))
```



ggnetwork

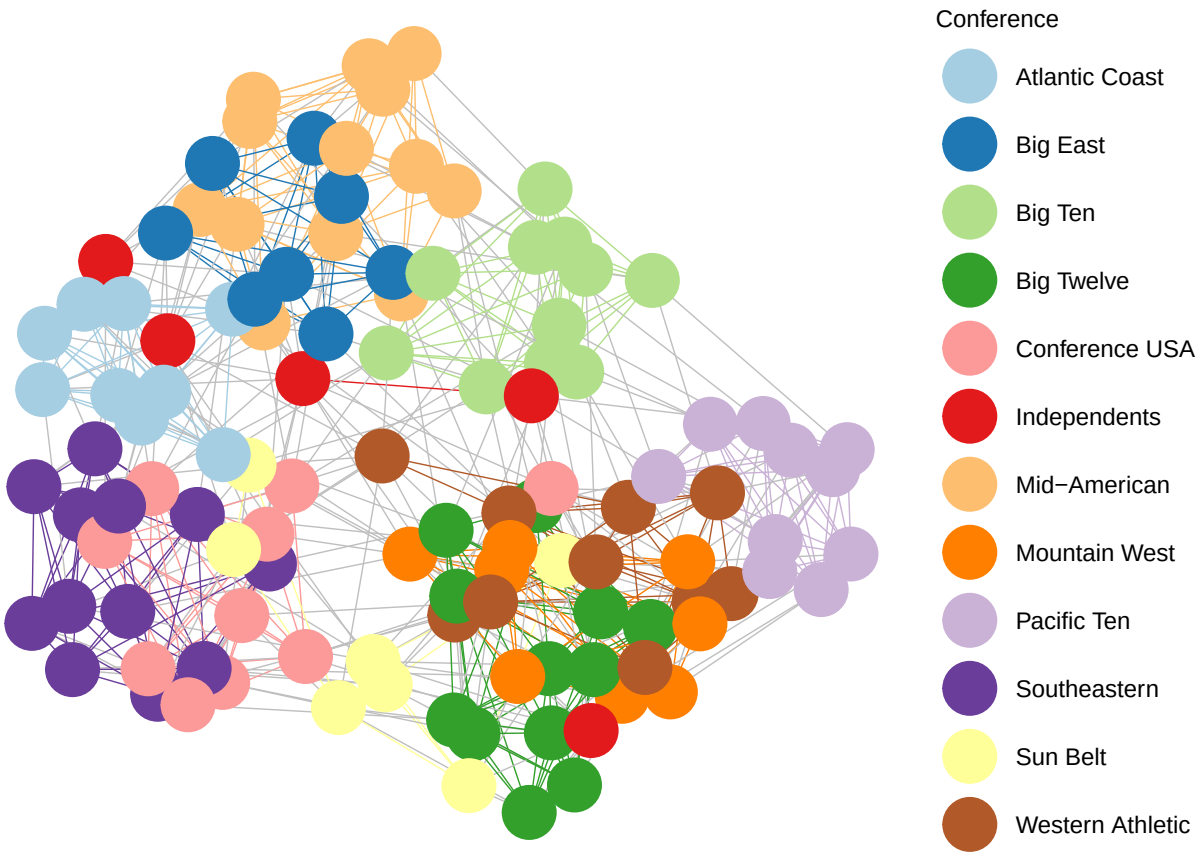
```
set.seed(3272016)
ggplot(ggnetwork(te.net, layout = "fruchtermanreingold"),
       aes(x, y, xend = xend, yend = yend)) +
  geom_edges() +
  geom_nodes(size = 12, color = "white") +
  geom_nodetext(
    aes(size = size, label = vertex.names)) +
  scale_size_continuous(range = c(4, 8)) +
  guides(size = FALSE) +
  theme_blank()
```



College Football

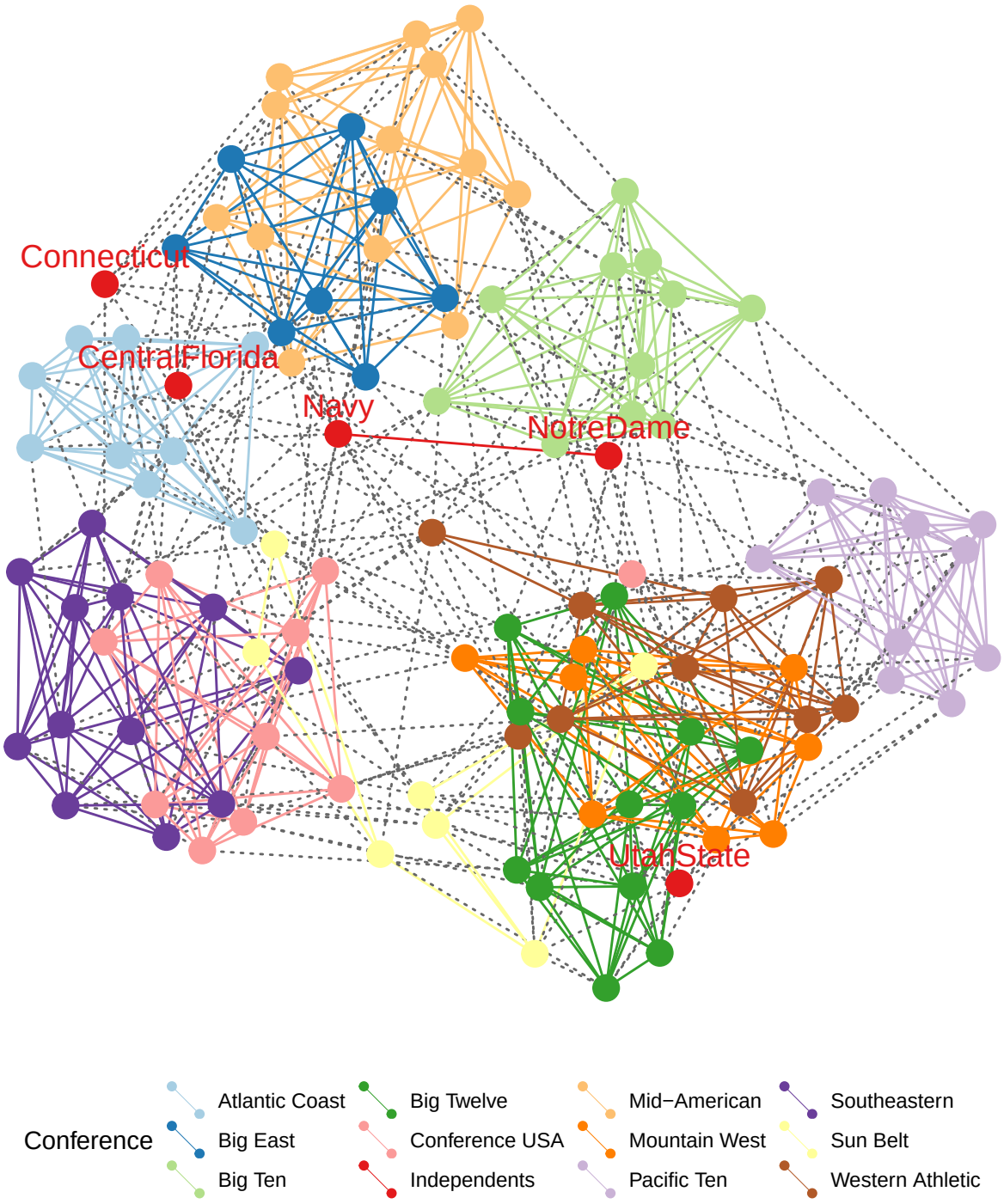
ggnet2

```
data(football, package = 'geomnet')
rownames(football$vertices) <-
  football$vertices$label
fb.net <- network::network(football$edges[, 1:2],
                  directed = TRUE)
fb.net %v% "conf" <-
  football$vertices[
    network.vertex.names(fb.net), "value"
    ]
network::set.edge.attribute(
  fb.net, "same.conf",
  football$edges$same.conf)
set.seed(5232011)
ggnet2(fb.net, mode = "fruchtermanreingold",
       color = "conf",  palette = "Paired",
       color.legend = "Conference",
       edge.color = c("color", "grey75"))
```



geomnet

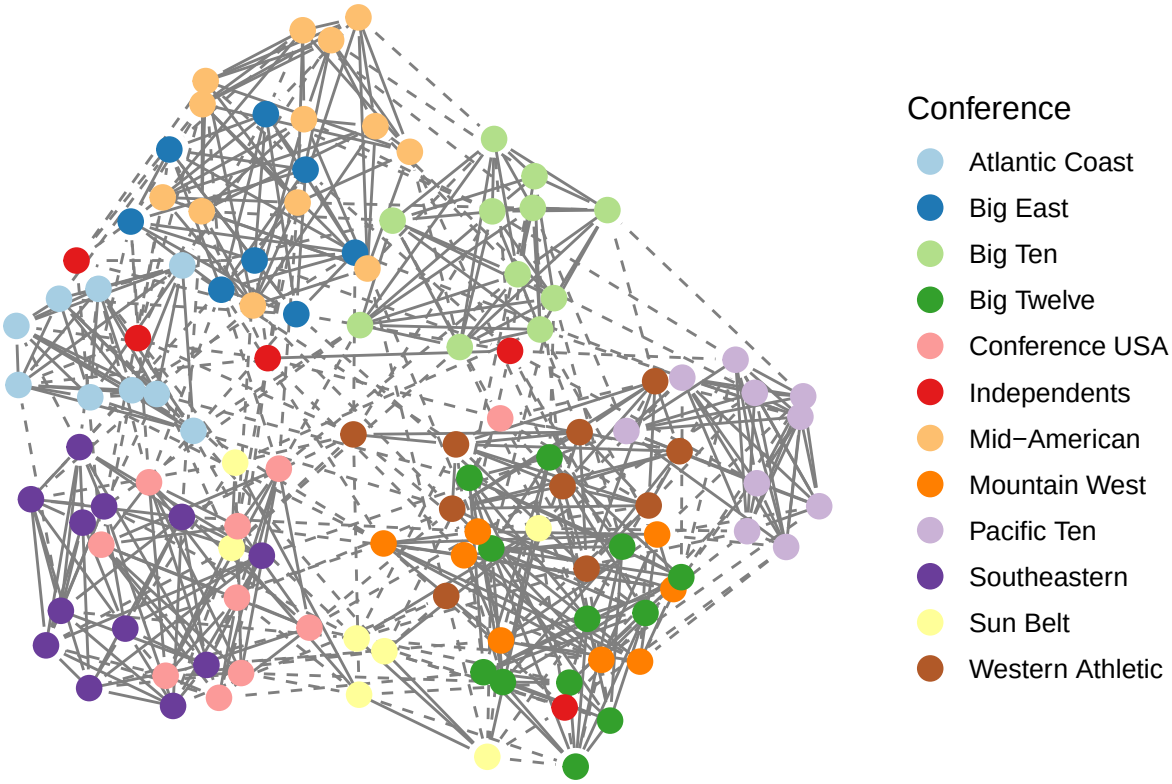
```
ftnet <- fortify(as.edgedf(football$edges), football$vertices)
ftnet$schools <- ifelse(
  ftnet$value == "Independents", ftnet$from_id, "")
set.seed(5232011)
ggplot(data = ftnet,
       aes(from_id = from_id, to_id = to_id)) +
  geom_net(layout.alg = 'fruchtermanreingold',
    aes(colour = value, group = value,
        linetype = factor(same.conf != 1),
        label = schools),
    linewidth = 0.5,
    size = 5, vjust = -0.75, alpha = 0.3) +
  theme_net() +
  theme(legend.position = "bottom") +
  scale_colour_brewer("Conference", palette = "Paired")  +
  guides(linetype = FALSE)
```



ggnetwork

```
set.seed(5232011)
ggplot(
  ggnetwork(
    fb.net,
    layout = "fruchtermanreingold"),
  aes(x, y, xend = xend, yend = yend)) +
  geom_edges(
```

```
    aes(linetype = as.factor(same.conf)),
    color = "grey50") +
  geom_nodes(aes(color = conf), size = 4) +
  scale_color_brewer("Conference",
                     palette = "Paired") +
  scale_linetype_manual(values = c(2,1)) +
  guides(linetype = FALSE) +
  theme_blank()
```



Southern Women (Bipartite Network) Example

Load Data

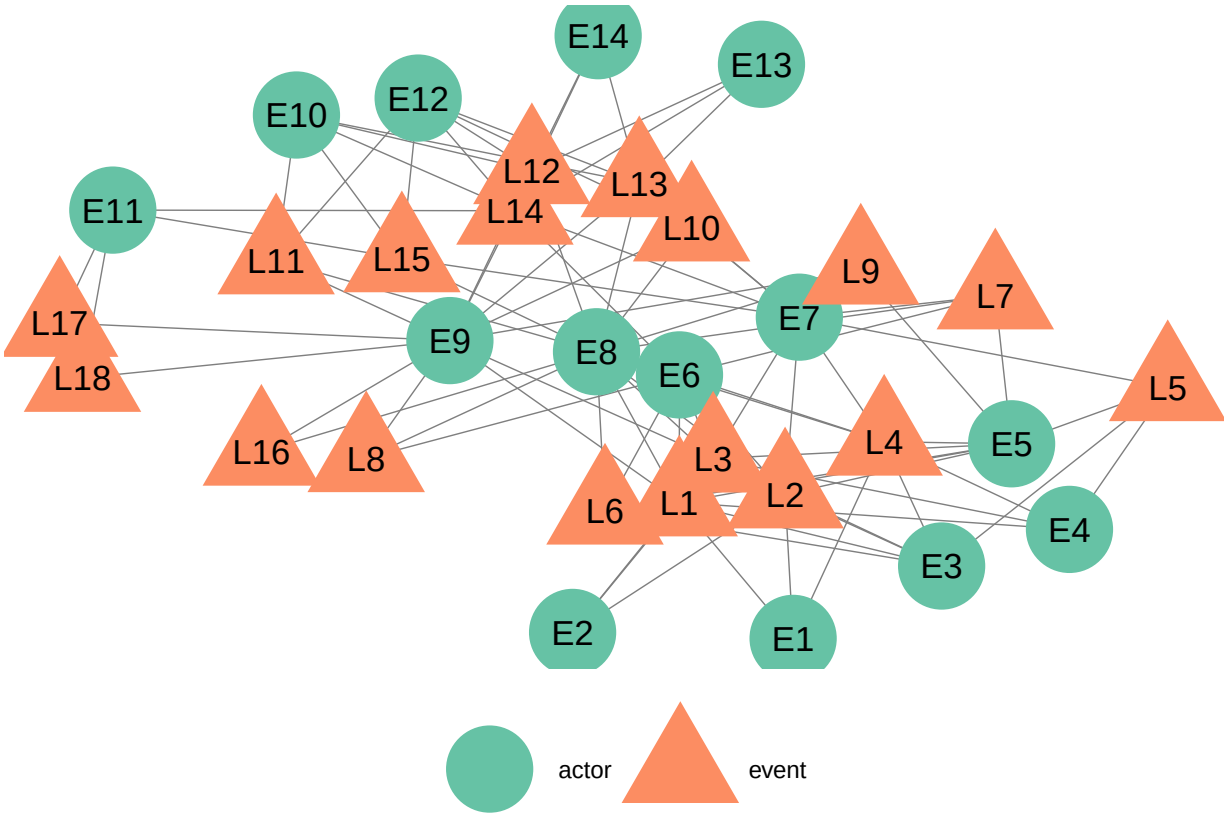
```
library(tnet)
data(tnet)
elist <- data.frame(Davis.Southern.women.2mode)
names(elist) <- c("Lady", "Event")
detach(package:tnet)
detach(package:igraph)
head(elist)
```

```
##   Lady Event
## 1    1     1
## 2    1     2
## 3    1     3
## 4    1     4
## 5    1     5
## 6    1     6
```

```
elist$Lady <- paste("L", elist$Lady, sep="")
elist$Event <- paste("E", elist$Event, sep="")
davis <- elist
names(davis) <- c("from", "to")
davis <- rbind(davis, data.frame(from=davis$to, to=davis$from))
davis$type <- factor(c(rep("Lady", nrow(elist)), rep("Event", nrow(elist))))
```

ggnet2

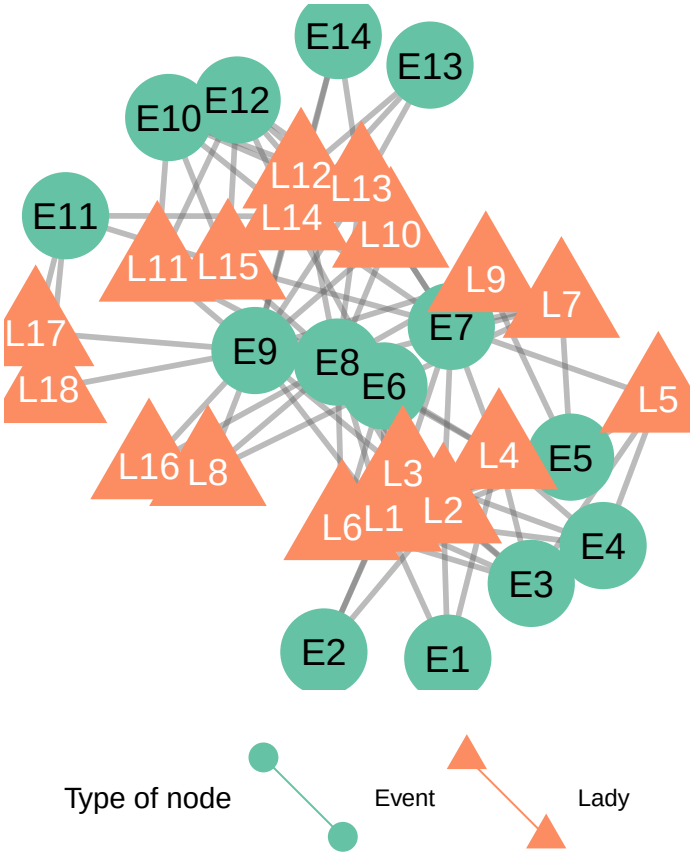
```
bip = xtabs(~Event+Lady, data=elist)
bip = network::network(bip,
              matrix.type = "bipartite",
              ignore.eval = FALSE,
              names.eval = "weights")
set.seed(8262013)
ggnet2(bip, color = "mode", palette = "Set2",
    shape = "mode", mode = "kamadakawai",
    size = 15, label = TRUE) +
theme(legend.position="bottom")
```



geomnet

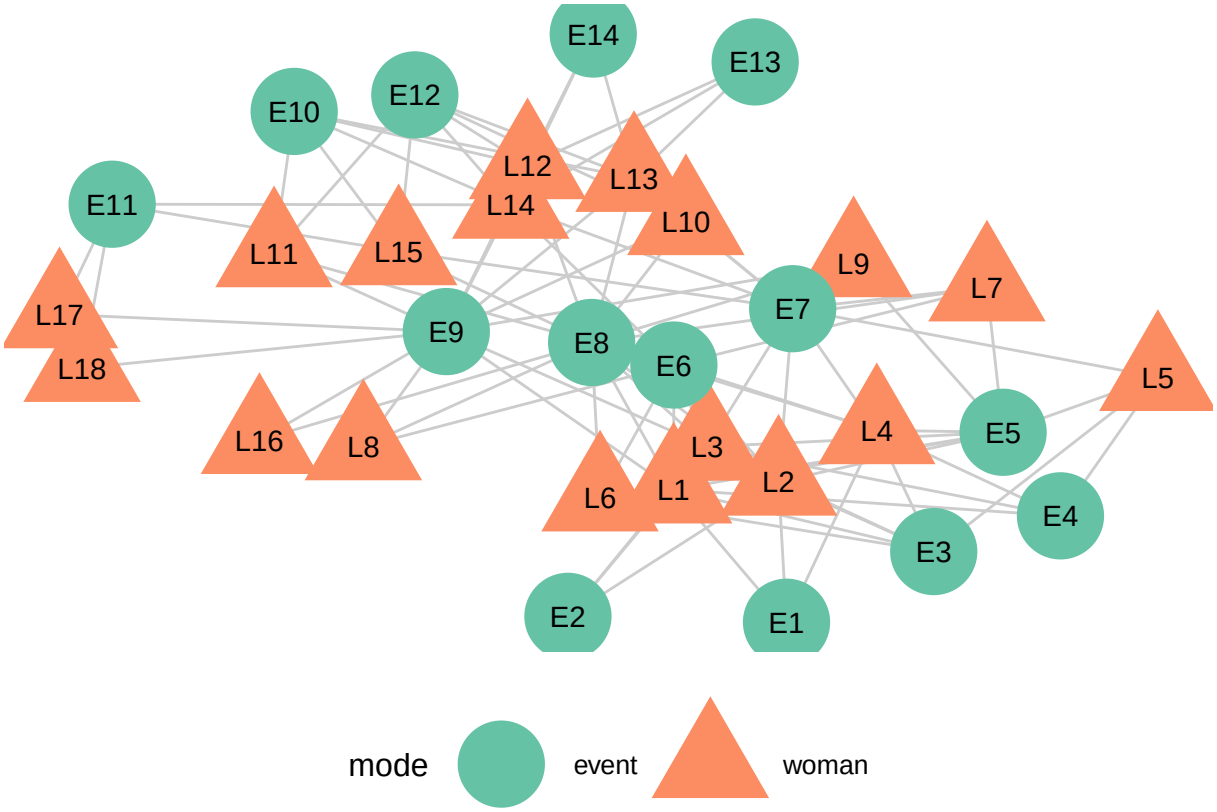
```
davis$lcolour <-
  c("white", "black")[as.numeric(davis$type)]
set.seed(8262013)
ggplot(data = davis) +
```

```
  geom_net(layout.alg = "kamadakawai",
    aes(from_id = from, to_id = to,
        colour = type, shape = type),
    size = 15, labelon = TRUE, ealpha = 0.25,
    vjust = 0.5, hjust = 0.5,
    labelcolour = davis$lcolour) +
  theme_net() +
  scale_colour_brewer("Type of node", palette = "Set2") +
  scale_shape("Type of node") +
  theme(legend.position = "bottom")
```



ggnetwork

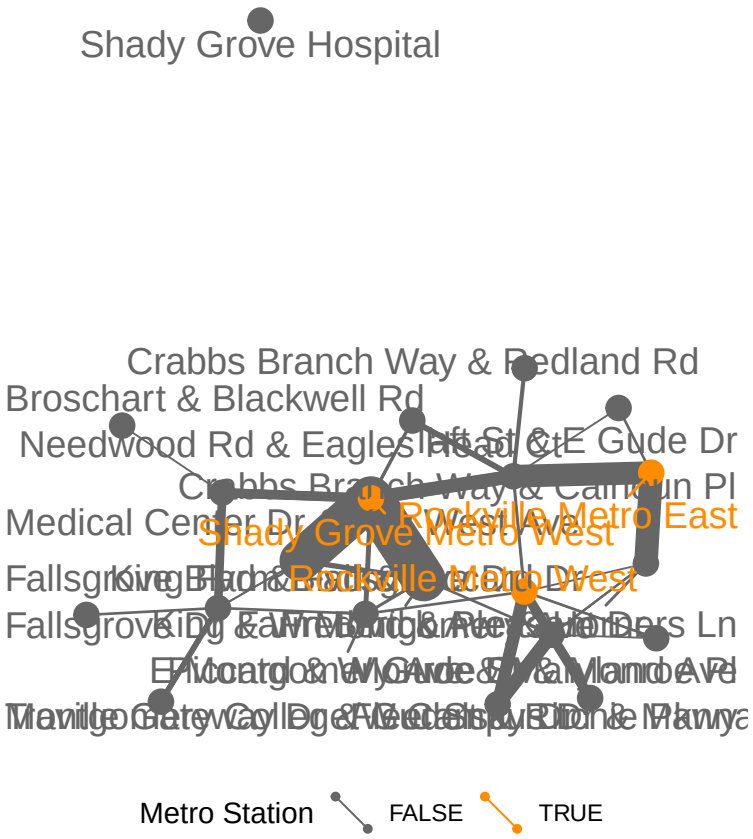
```
network::set.vertex.attribute(bip, "mode",
  c(rep("event", 14), rep("woman", 18)))
set.seed(8262013)
ggplot(data = ggnetwork(bip, layout = "kamadakawai"),
       aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_edges(colour = "grey80") +
  geom_nodes(aes(colour = mode, shape = mode), size = 15) +
  geom_nodetext(aes(label = vertex.names)) +
  scale_colour_brewer(palette = "Set2") +
  theme_blank() +
  theme(legend.position = "bottom")
```



Captial Bikeshare

geomnet

```
data(bikes, package = 'geomnet')
tripnet <- fortify(as.edgedf(bikes$trips), bikes$stations[,c(2,1,3:5)])
tripnet$Metro = FALSE
idx <- grep("Metro", tripnet$from_id)
tripnet$Metro[idx] <- TRUE
set.seed(1232016)
ggplot(aes(from_id = from_id, to_id = to_id), data = tripnet) +
  geom_net(aes(linewidth = n / 15, colour = Metro),
           labelon = TRUE, repel = TRUE) +
  theme_net() +
  xlim(c(-0.1, 1.1)) +
  scale_colour_manual("Metro Station", values = c("grey40", "darkorange")) +
  theme(legend.position = "bottom")
```



Data Preparation

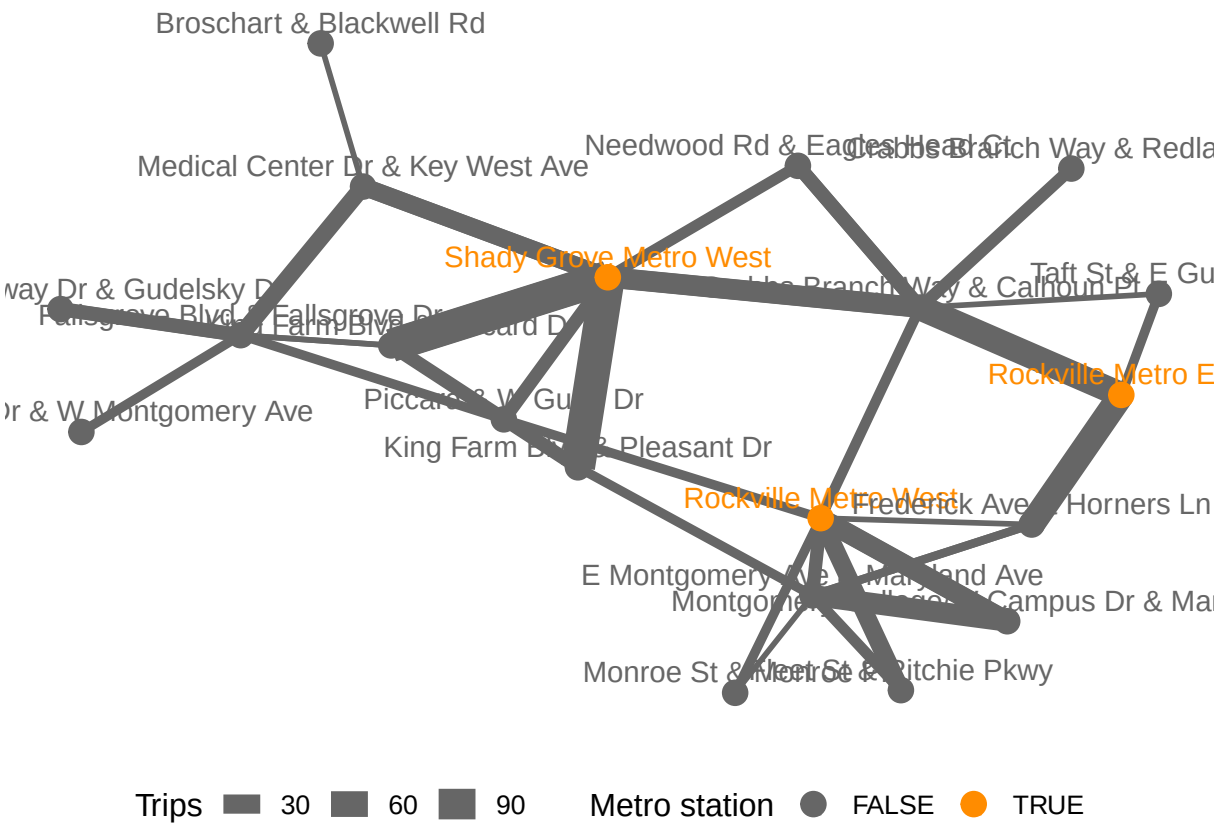
```
bikes.net <- network::network(bikes$trips[, 1:2 ], directed = FALSE)
network::set.edge.attribute(bikes.net, "n", bikes$trips[, 3 ] / 15)
bikes.net %v% "station" <-  grepl("Metro", network.vertex.names(bikes.net))
bikes.net %v% "station" <-  1 + as.integer(bikes.net %v% "station")
rownames(bikes$stations) <- bikes$stations$name
bikes.net %v% "lon" <-
  bikes$stations[ network.vertex.names(bikes.net), "long" ]
bikes.net %v% "lat" <-
  bikes$stations[ network.vertex.names(bikes.net), "lat" ]
bikes.col <- c("grey40", "darkorange")
```

ggnet2

```
set.seed(1232016)
ggnet2(bikes.net, mode = "fruchtermanreingold", size = 4, label = TRUE,
       vjust = -0.5, edge.size = "n", layout.exp = 1.1,
       color = bikes.col[ bikes.net %v% "station" ],
       label.color = bikes.col[ bikes.net %v% "station" ])
```

ggnetwork

```
set.seed(1232016)
ggplot(data = ggnetwork(bikes.net, layout.alg = "fruchtermanreingold"),
         aes(x, y, xend = xend, yend = yend)) +
  geom_edges(aes(size = n), color = "grey40") +
  geom_nodes(aes(color = factor(station)), size = 4) +
  geom_nodetext(aes(label = vertex.names, color = factor(station)),
                vjust = -0.5) +
  scale_size_continuous("Trips", breaks = c(2, 4, 6), labels = c(30, 60, 90)) +
  scale_colour_manual("Metro station", labels = c("FALSE", "TRUE"),
                      values = c("grey40", "darkorange")) +
  theme_blank() +
  theme(legend.position = "bottom", legend.box = "horizontal")
```

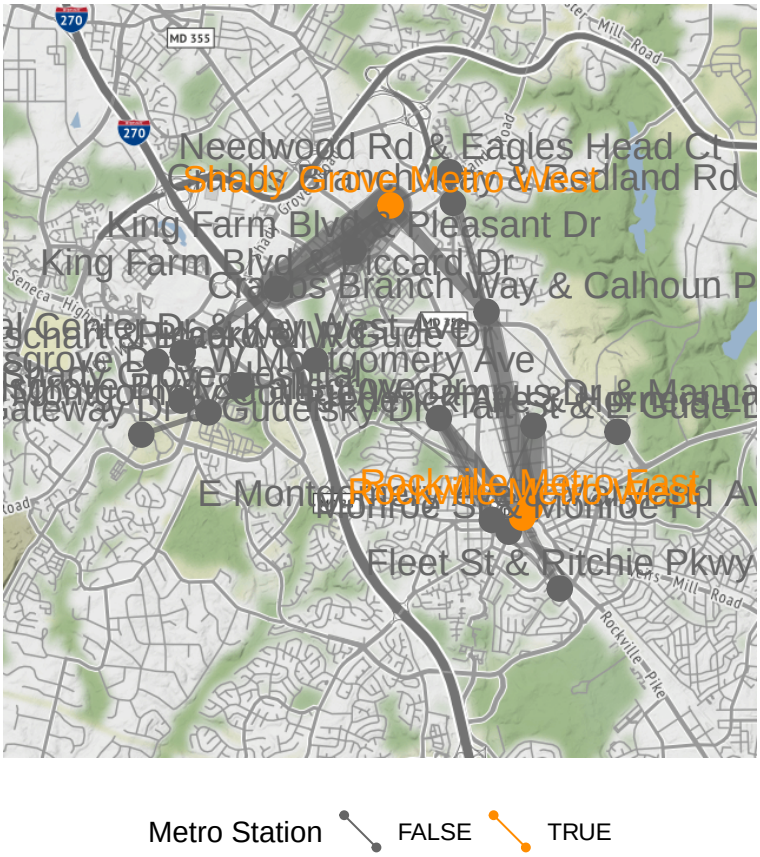


Geographically Accurate Layout

```
metro_map <- ggmap::get_map(location = c(left = -77.22257, bottom = 39.05721,
                                         right = -77.11271, top = 39.14247))
```

geomnet

```
  ggmap::ggmap(metro_map) +
  geom_net(data = tripnet, layout.alg = NULL, labelon = TRUE,
           vjust = -0.5, ealpha = 0.5,
           aes(from_id = from_id,
               to_id = to_id,
               x = long, y = lat,
               linewidth = n / 15,
               colour = Metro)) +
  scale_colour_manual("Metro Station", values = c("grey40", "darkorange")) +
  theme_net() %+replace% theme(aspect.ratio=NULL, legend.position = "bottom") +
  coord_map()
```



Summary

Although the code was identical to the one from supplementary materials, some of the results were not reproduced correctly. The proportions of some graphs were incorrect which made them impossible to read. Consequently this article is not fully reproducible.

Empirical Mode Decomposition and Hilbert Spectrum

Third article (Donghoh and Hee-Seok, 2009) is about `EMD` package used for decomposing a signal into so-called intrinsic mode function.

Link to article: `https://doi.org/10.32614/RJ-2009-002`

Intrinsic mode function

Data preparation

```
ndata <- 3000
tt <- seq(0, 9, length=ndata)
xt <- sin(pi * tt)
```

Function call

```
library(EMD)
extrema(xt)
```

```
## $minindex
##      [,1] [,2]
## [1,]  501  501
## [2,] 1167 1167
## [3,] 1834 1834
## [4,] 2500 2500
##
## $maxindex
##      [,1] [,2]
## [1,]  168  168
## [2,]  834  834
## [3,] 1500 1501
## [4,] 2167 2167
## [5,] 2833 2833
##
## $nextreme
## [1] 9
##
## $cross
##       [,1] [,2]
##  [1,]    1    1
##  [2,]  334  335
##  [3,]  667  668
##  [4,] 1000 1001
##  [5,] 1333 1334
##  [6,] 1667 1668
##  [7,] 2000 2001
##  [8,] 2333 2334
##  [9,] 2666 2667
##
## $ncross
## [1] 9
```

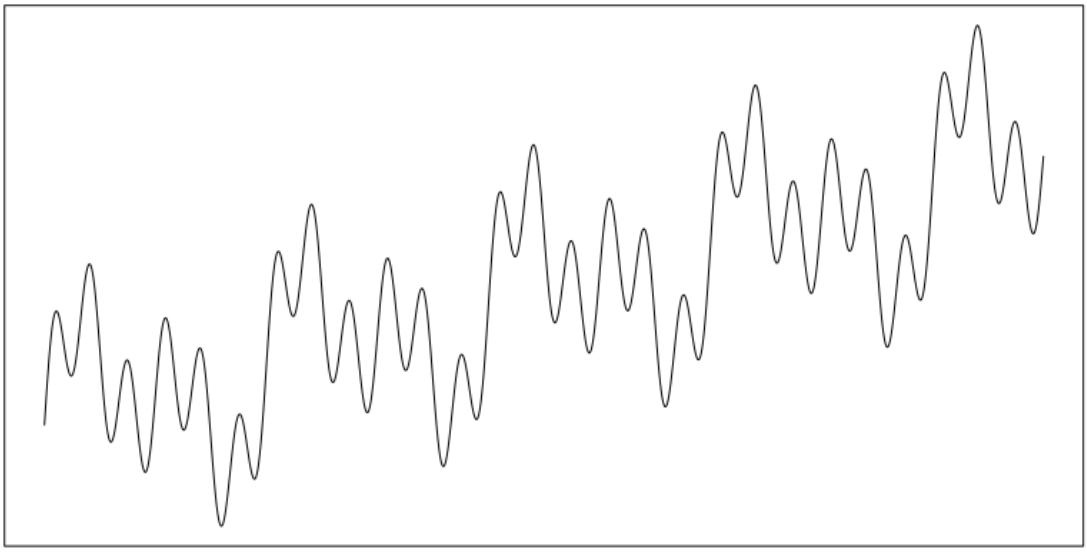
Sifting process

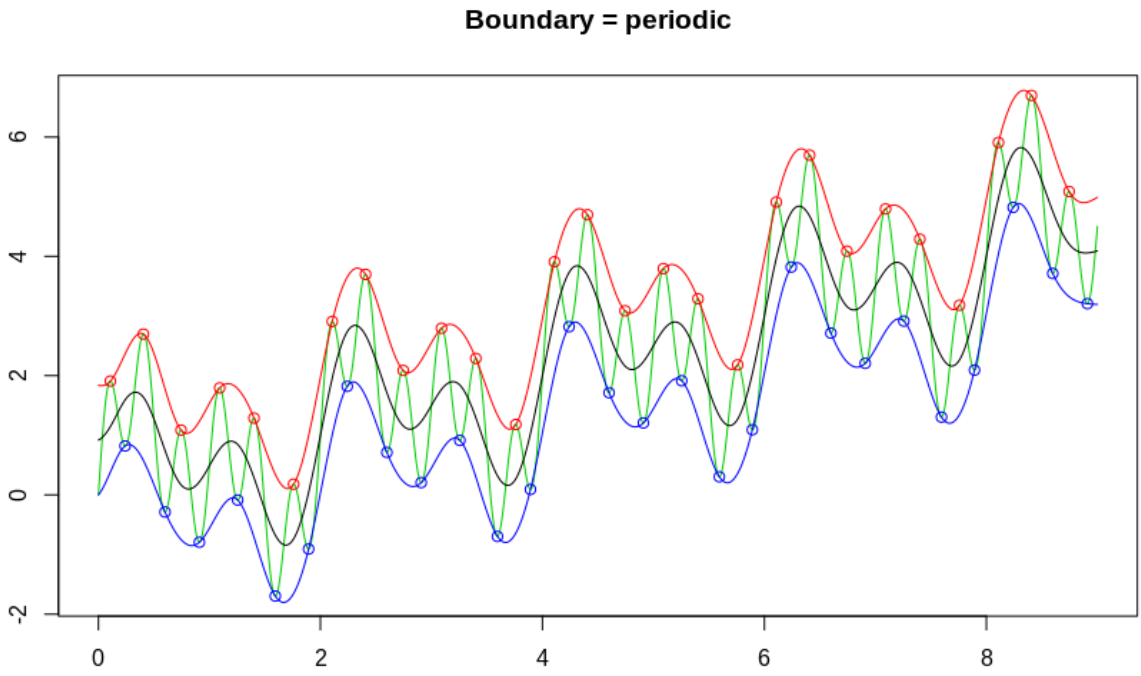
Data preparation

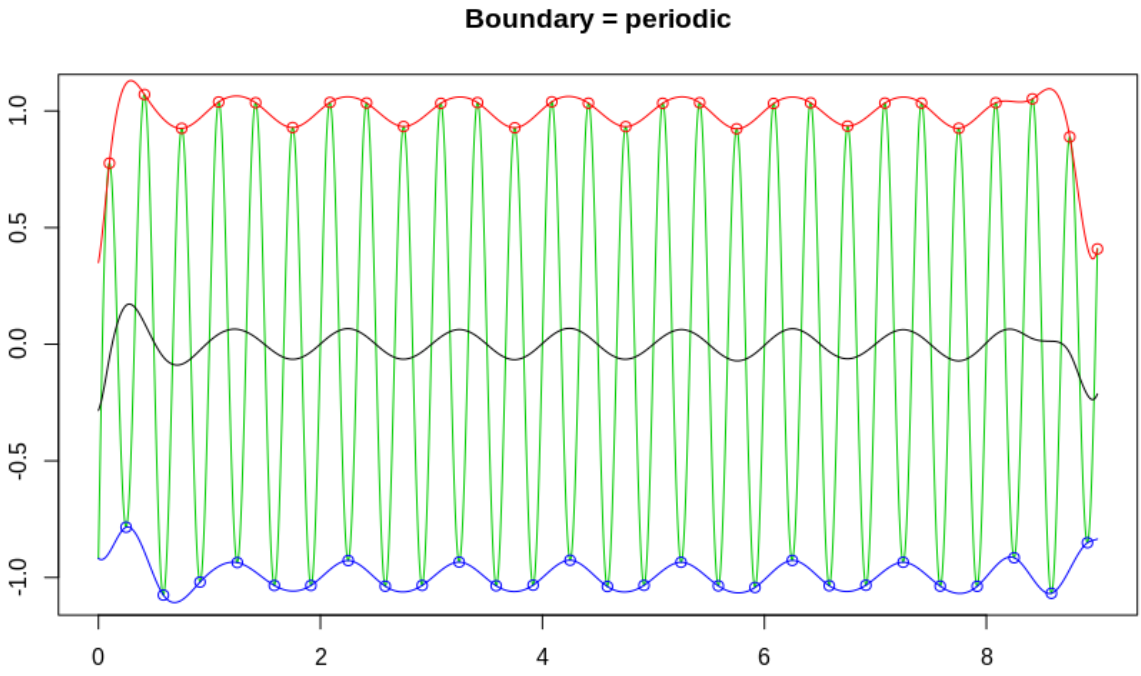
```
ndata <- 3000
par(mfrow=c(1,1), mar=c(1,1,1,1))
tt2 <- seq(0, 9, length=ndata)
xt2 <- sin(pi * tt2) + sin(2* pi * tt2) + sin(6 * pi * tt2)  + 0.5 * tt2
```

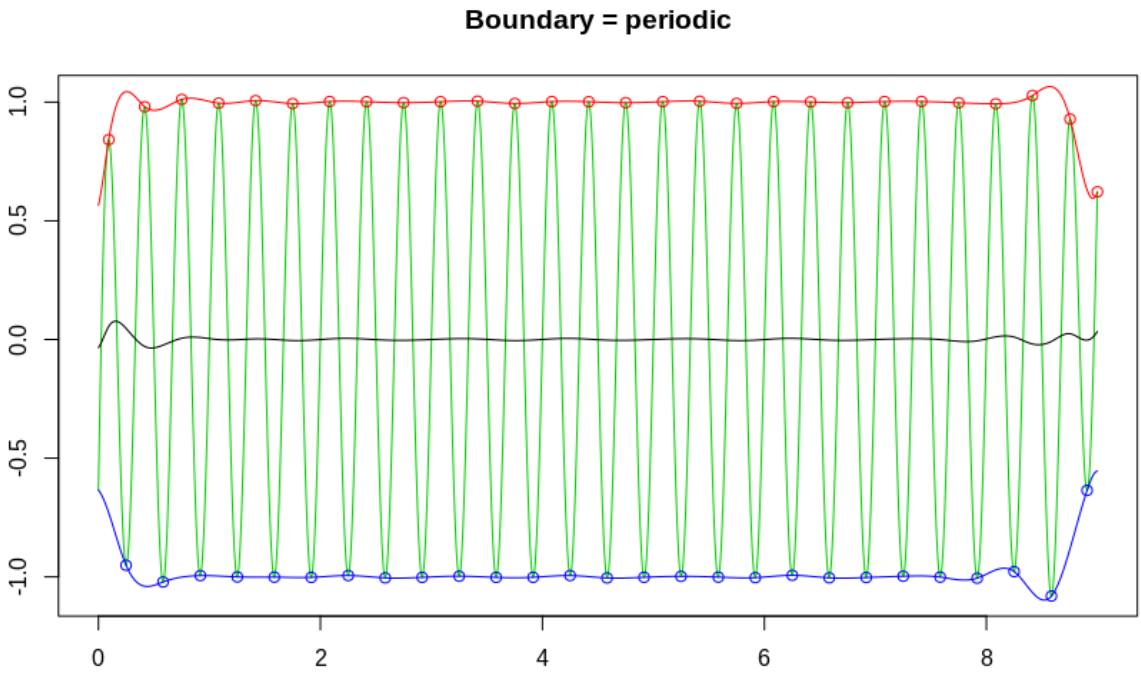
Results

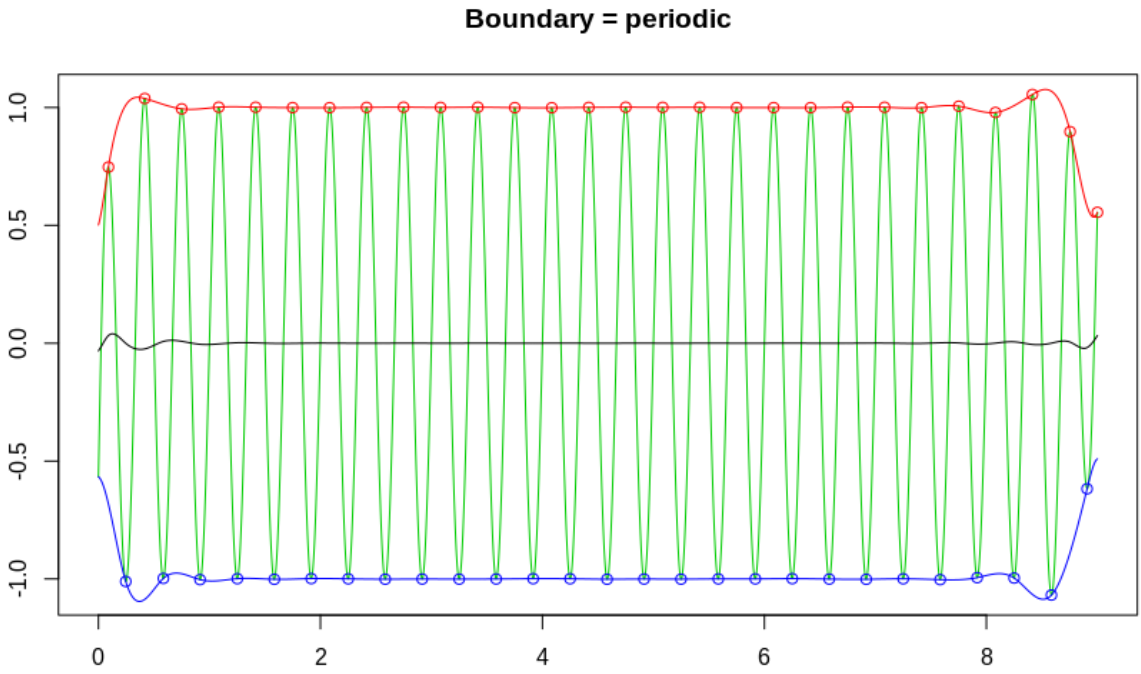
```
plot(tt2, xt2, xlab="", ylab="", type="l", axes=FALSE); box()
tryimf <- extractimf(xt2, tt2, check=TRUE)
```

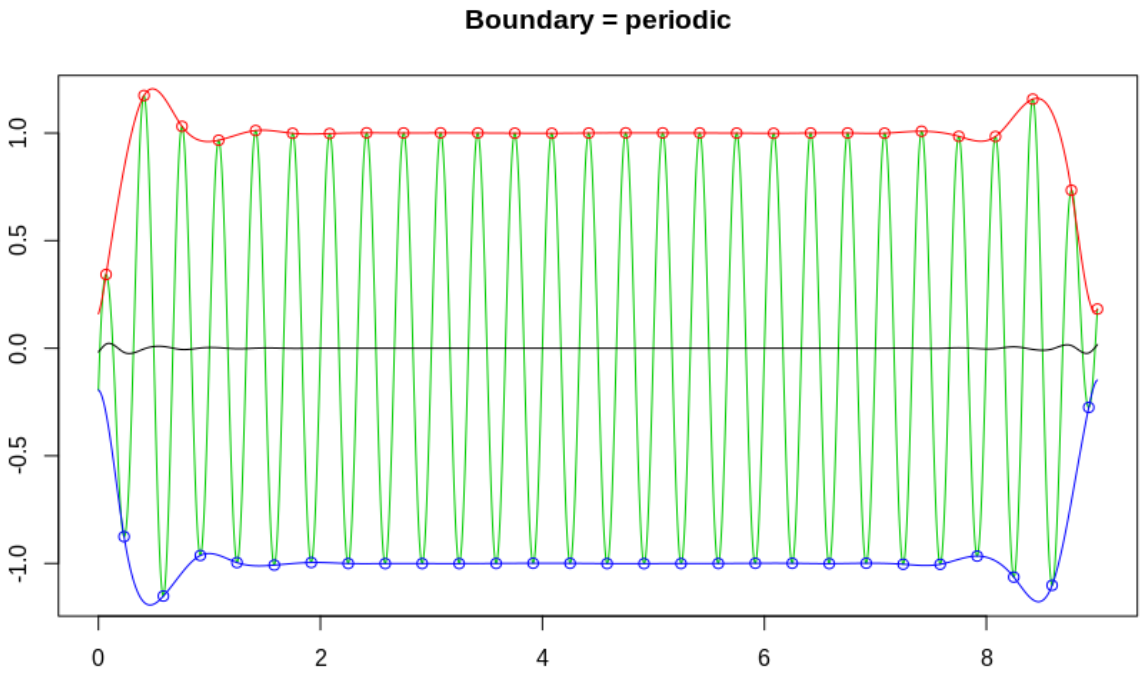



Boundary = periodic
6
4
2
0
-2
0
2
4
6
8


Boundary = periodic
1.0
0.5
0.0
-0.5
-1.0
0
2
4
6
8


Boundary = periodic
1.0
0.5
0.0
-0.5
-1.0
0
2
4
6
8


Boundary = periodic
1.0
0.5
0.0
-0.5
-1.0
0
2
4
6
8


Boundary = periodic
1.0
0.5
0.0
-0.5
-1.0
0
2
4
6
8

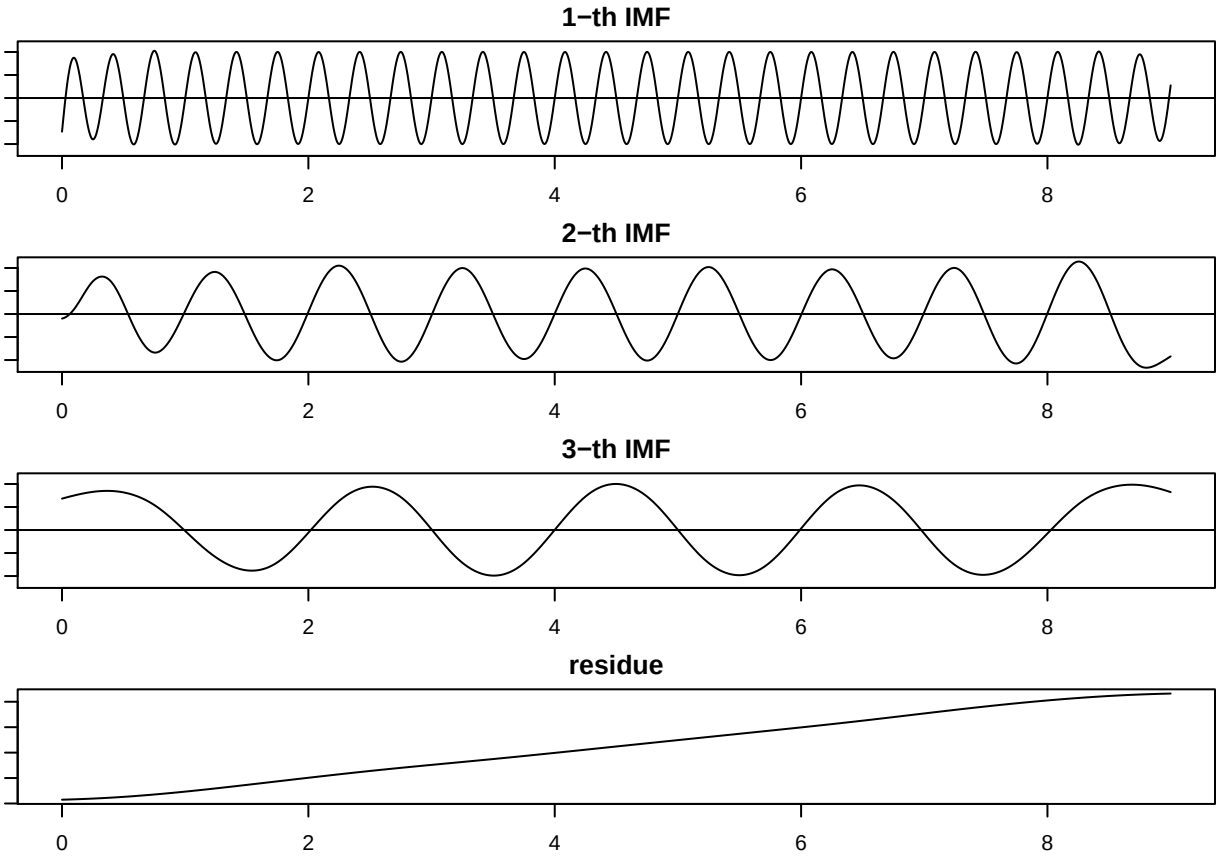
Empirical mode decomposition

Function call

```
par(mfrow=c(3,1), mar=c(2,1,2,1))
try <- emd(xt2, tt2, boundary="wave")
```

Plotting IMF

```
par(mfrow=c(3,1), mar=c(2,1,2,1))
par(mfrow=c(try$nimf+1, 1), mar=c(2,1,2,1))
rangeimf <- range(try$imf)
for(i in 1:try$nimf){
  plot(tt2, try$imf[,i], type="l", xlab="", ylab="",
       ylim=rangeimf, main= paste(i, "-th IMF", sep=""));
  abline(h=0)
}
plot(tt2, try$residue, xlab="", ylab="", main="residue", type="l")
```



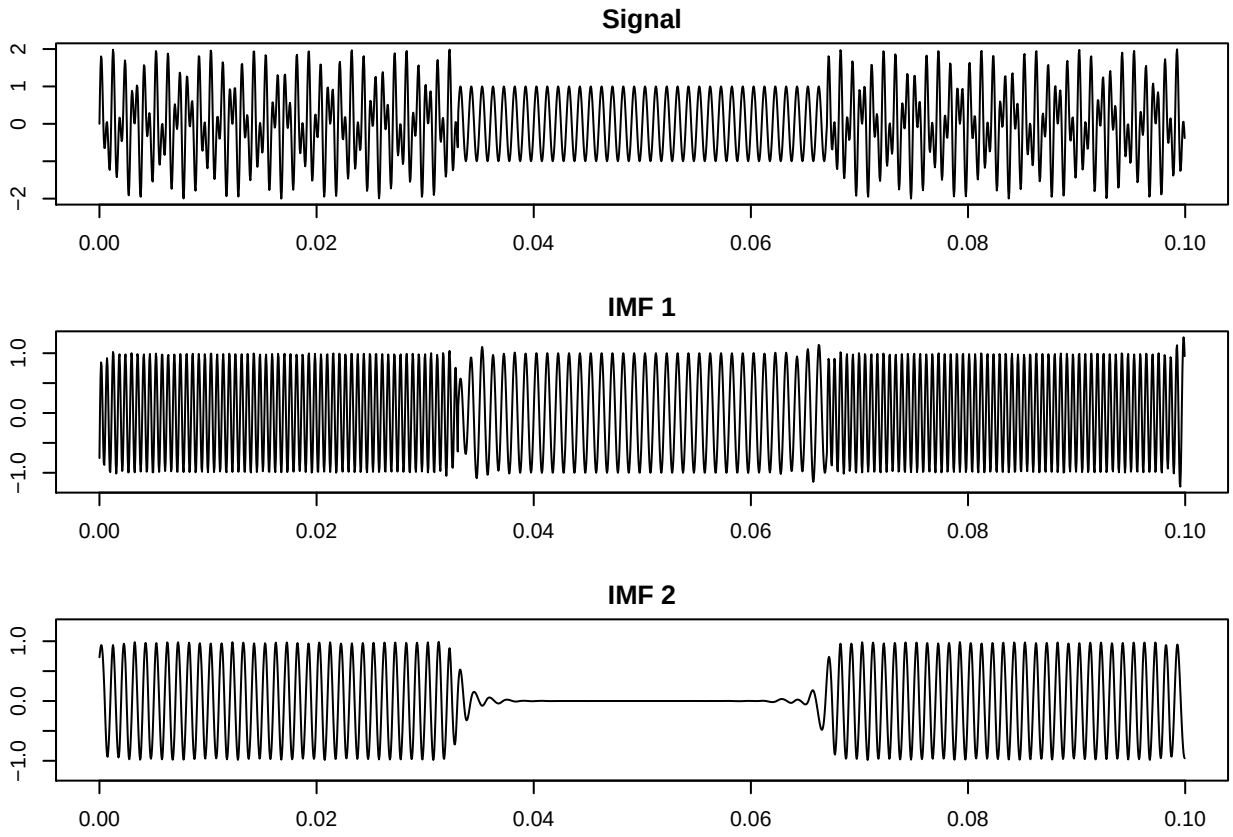
Intermittence

Mode mixing

```
tt <- seq(0, 0.1, length = 2001)[1:2000]
f1 <- 1776; f2 <- 1000
xt <- sin(2*pi*f1*tt) * (tt <= 0.033 | tt >= 0.067) + sin(2*pi*f2*tt)
```

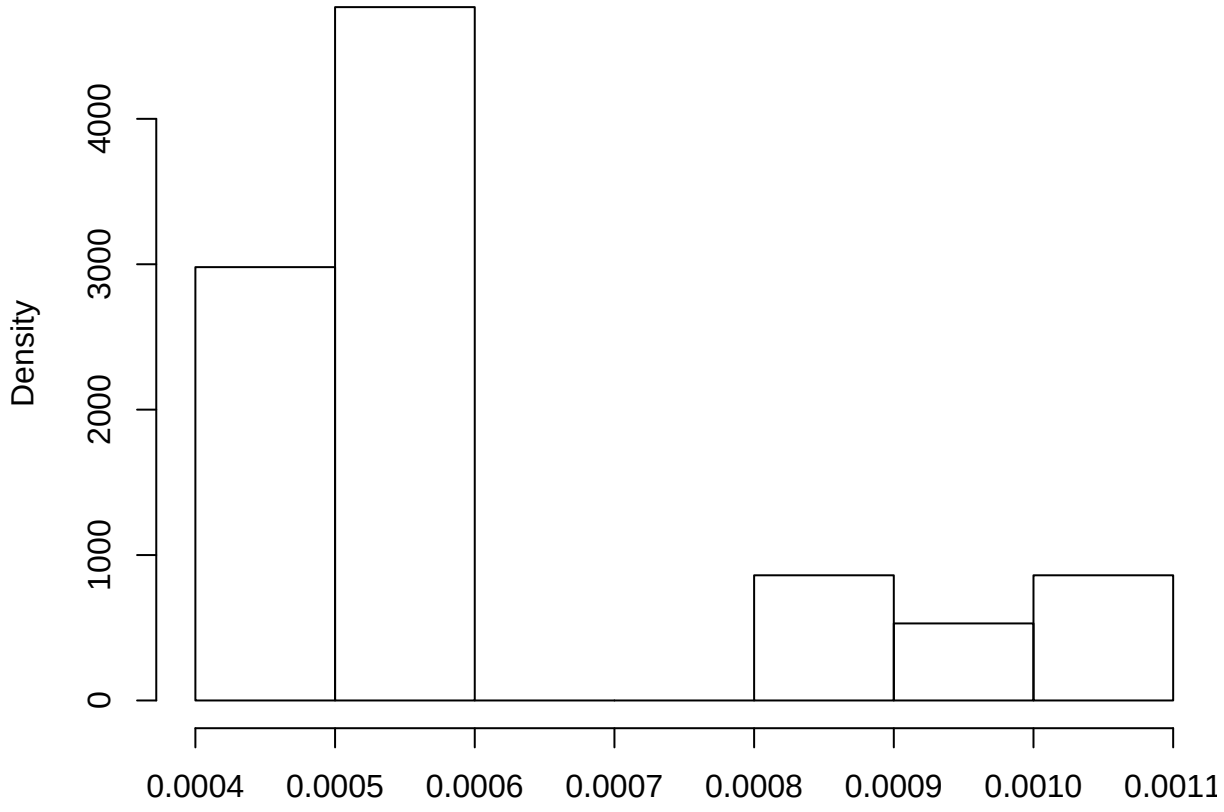
EMD

```
interm1 <- emd(xt, tt, boundary="wave", max.imf=2, plot.imf=FALSE)
par(mfrow=c(3, 1), mar=c(3,2,2,1))
plot(tt, xt, main="Signal", type="l")
rangeimf <- range(interm1$imf)
plot(tt, interm1$imf[,1], type="l", xlab="", ylab="", ylim=rangeimf, main="IMF 1")
plot(tt, interm1$imf[,2], type="l", xlab="", ylab="", ylim=rangeimf, main="IMF 2")
```



Histogram of empirical period

```
par(mfrow=c(1,1), mar=c(2,4,1,1))
tmpinterm <- extrema(interm1$imf[,1])
zerocross <- as.numeric(round(apply(tmpinterm$cross, 1, mean)))
hist(diff(tt[zerocross[seq(1, length(zerocross), by=2)]]), freq=FALSE, xlab="", main="")
```

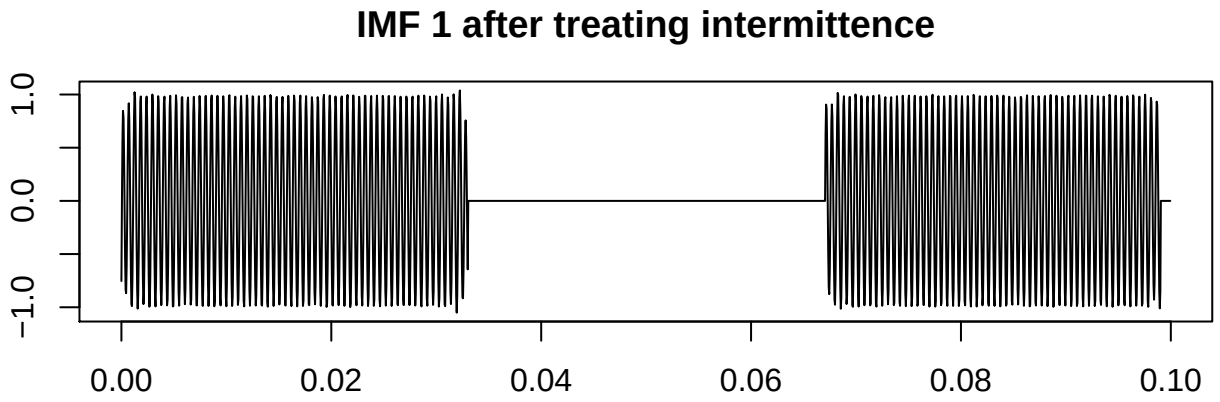


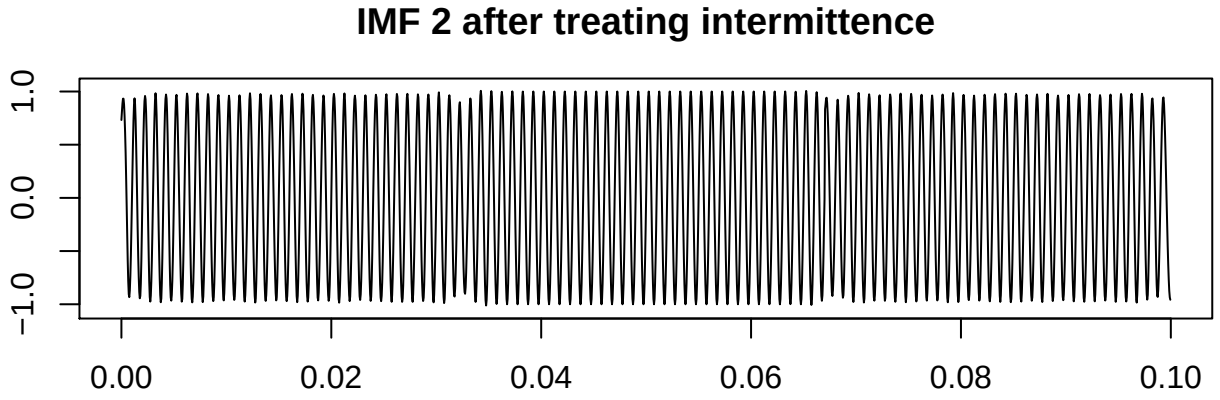
Treating intermittence

```
interm2 <- emd(xt, tt, boundary="wave", max.imf=2, plot.imf=FALSE, interm=0.0007)
```

Plot of each IMF

```
par(mfrow=c(2,1), mar=c(2,2,3,1), oma=c(0,0,0,0))
rangeimf <- range(interm2$imf)
plot(tt,interm2$imf[,1], type="l", main="IMF 1 after treating intermittence",
     xlab="", ylab="", ylim=rangeimf)
plot(tt,interm2$imf[,2], type="l", main="IMF 2 after treating intermittence",
     xlab="", ylab="", ylim=rangeimf)
```

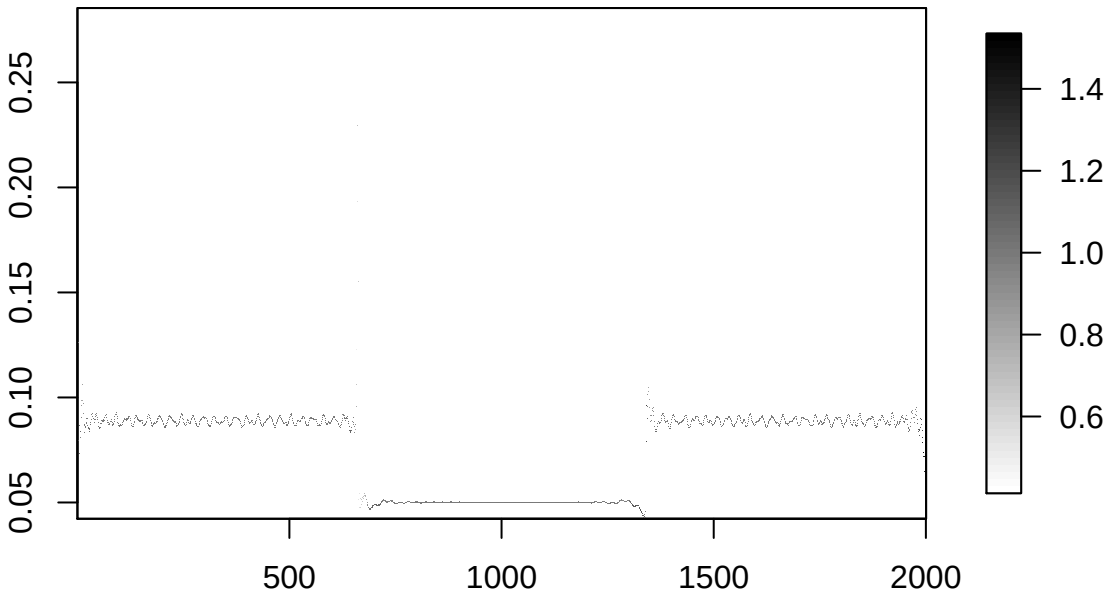




Hilbert spectrum

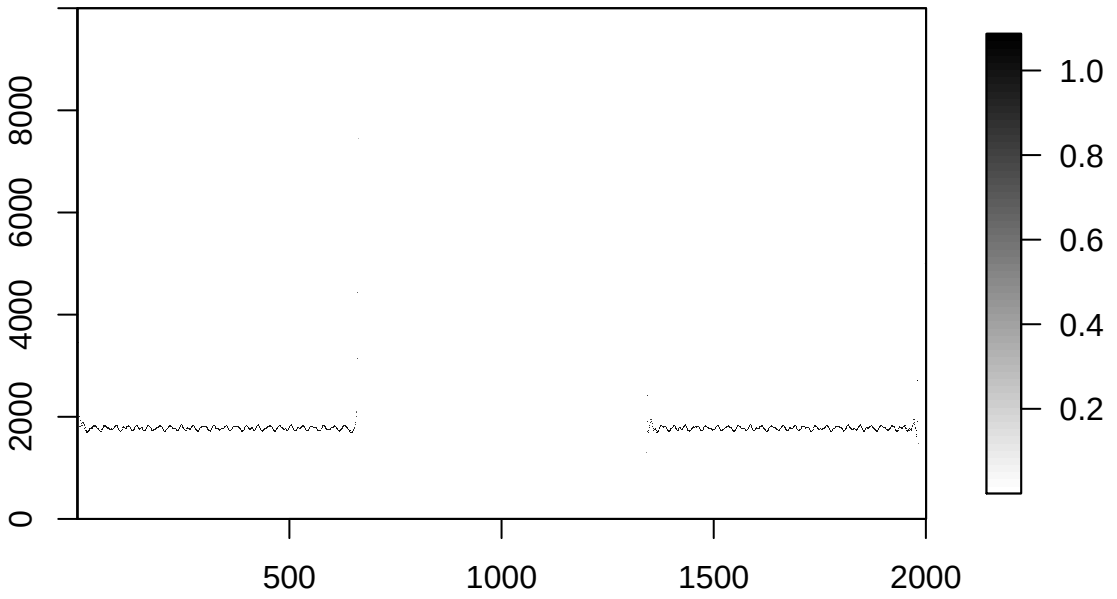
First Spectrogram

```
test1 <- hilbertspec(interm1$imf)
spectrogram(test1$amplitude[,1], test1$instantfreq[,1])
```



Second spectrogram

```
test2 <- hilbertspec(interm2$imf, tt=tt)
spectrogram(test2$amplitude[,1], test2$instantfreq[,1])
```



Extension to two dimensional image

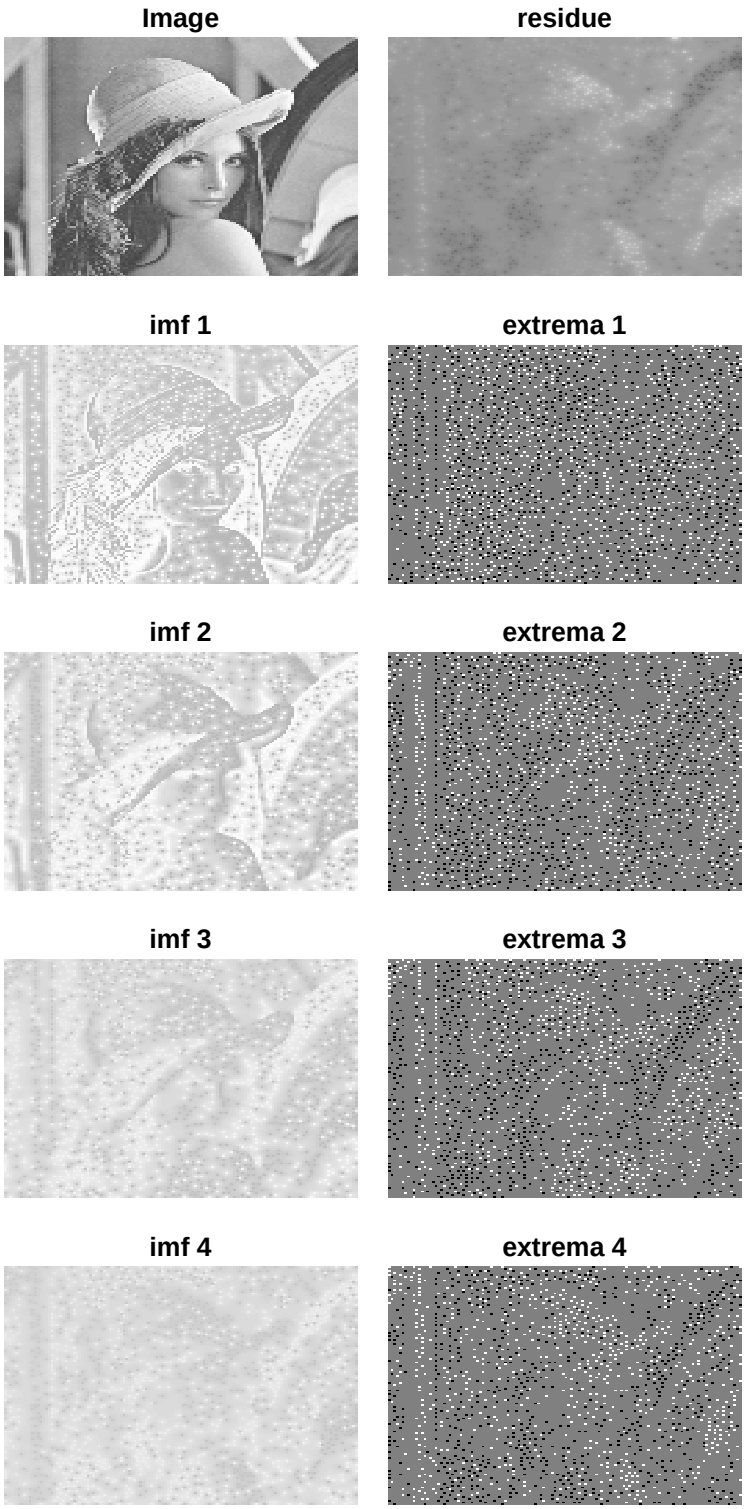
Data loading and decomposition

```
data(lena)
z <- lena[seq(1, 512, by=4), seq(1, 512, by=4)]
```

```
lenadecom <- emd2d(z, max.imf = 4)
```

Result plot

```
imageEMD(z=z, emdz=lenadecom, extrema=TRUE, col=gray(0:100/100))
```



Summary

This article is fully and easily reproducible, in spite of being the oldest one (2009).

Conclusion

Out of these three articles only two are reporoducible. The second article was not possible to be reproduced correctly, because the graphs had distorted proportions which made them often very hard or even impossible to interpret. This happend mainly because of the package `geomnet` which was removed from cran repository last month (2020-02-19). I was able to download it from github repository, but it was not stable nor official version. Nontheless, I was positively surprised that the third article was the easiest to reproduce despite being released more than ten years ago.