Structured factorization for single-cell gene expression data: analysis of lung adenocarcinoma scRNA-seq data

Antonio Canale, Luisa Galtarossa, Davide Risso, Lorenzo Schiavon, Giovanni Toto

Last modified: June 07, 2023

This vignette contains the code developed and used for the analysis of lung adenocarcinoma scRNA-seq data in section 3 of the paper.

Introduction

We analyze a subset of a large study with scRNA-seq data for three lung adenocarcinoma cell lines. The original data are high quality in terms of exon mapping and unique transcript counts per cell. The data have already undergone post-sampling quality control. Pre-analysis of the data was performed exploiting the pipe-line of pre-analysis for scRNA-seq data, `scPipe`. Cells with high percentages of mitochondrial genes and less than 1000 detected genes are excluded. The resulting data has a gene expression matrix of $n=199$ cells and $p=949$ genes. Along with this gene expression matrix (`y`), quality control information about the cells (`x`) and the genes (`wT`, `wB`) are also available.

Load packages and data

```
library(cosin)
library(GGally)
library(ggplot2)
library(ggrepel)
library(igraph)
library(network)
library(RColorBrewer)
```

We import four matrices containing lung adenocarcinoma scRNA-seq data:

- `y`: matrix of gene expression,
- `wT`: gene-specific technical meta-covariates,
- `wB`: gene-specific biological meta-covariates,
- `x`: cell-specific covariates.

```
y <- readRDS("data/observed_data")
wT <- readRDS("data/metacovariate_mean_data")
wB <- readRDS("data/metacovariate_cov_data")
x <- readRDS("data/covariate_data")
covariate <- readRDS("data/covariate_data")
```

We also define colors for the plots:

```
unipdred <- alpha("#9B0014", 0.9)
mybeige <- alpha("#e1cd9d", 0.9)
mygrey <- alpha("#5e5a55", 0.9)
myavion <- alpha("lightsteelblue2", 0.9)
mygreen <- alpha("palegreen3", 0.9)
mypink <- alpha("thistle3", 0.9)
mypurple <- alpha("mediumpurple3", 0.9)
# gradient palette beige - red
funcColor <- colorRampPalette(c("#b08b36", "white", unipdred))
```

Posterior computation

We apply `COSIN` to lung adenocarcinoma scRNA-seq data. We run the adaptive Gibbs sampler for 20000 iterations discarding the first 5000 iterations, then we thin the Markov chain, saving every 2-th sample. We adapt the number of active factors at iteration t with probability $\Pr(t) = \exp(-1 - 5 \cdot 10^{-4} t)$ after at least 100 iterations.

```
# wTformula <- as.formula("~ length + gc_content")
# wBformula <- as.formula(paste("~", paste(colnames(wB),collapse = " + ")))
# xFormula <- as.formula("~ unaligned + aligned_unmapped+mapped_to_exon +
#                         mapped_to_intron + ambiguous_mapping +
#                         mapped_to_MT + number_of_genes + total_count_per_cell")
#
# Mcmc <- cosin(y = y, wT = wT, wB = wB, x = x, y_max = Inf,
#               wTformula = wTformula, wBformula = wBformula, xFormula = xFormula,
#               stdwT = TRUE, stdwB = TRUE, stdx = TRUE,
#               sd_gammaT = 1, sd_gammaB = 1, sd_beta = 1/3,
#               a_theta = 1, b_theta = 1, a_sigma = 1, b_sigma = 1,
#               alpha = 10, p_constant = NULL,
#               kinit = NULL, kmax = NULL, kval = 4,
#               nrun = 20000, thin = 2,
#               start_adapt = 100, b0 = 1, b1 = 5*10^(-4),
#               seed = 123, output = "all", verbose = TRUE)
#
# lpost_mode <- lposterior(out_MCMC = Mcmc)
# lpost_mean <- posterior_mean(out_MCMC = Mcmc)
# contribution_means <- contributions(out_MCMC = Mcmc,
#                                     reference = length(Mcmc$numFactors))
#
# saveRDS(Mcmc, "results/cosin_mcmc.RDS")
# saveRDS(lpost_mode, "results/lposterior.RDS")
# saveRDS(lpost_mean, "results/posterior_mean.RDS")
# saveRDS(contribution_means, "results/cont_postmean_ref.RDS")

Mcmc <- readRDS("results/cosin_mcmc.RDS")
lpost_mode <- readRDS("results/lposterior.RDS")
lpost_mean <- readRDS("results/posterior_mean.RDS")
contribution_means <- readRDS("results/cont_postmean_ref.RDS")
```

Post-processing

First, we compute credible intervals for mean coefficients and report summaries of β coefficients in the table below. The last column illustrates that only a small proportion of β's for any covariate has 0.9-level posterior credible intervals not including zero.

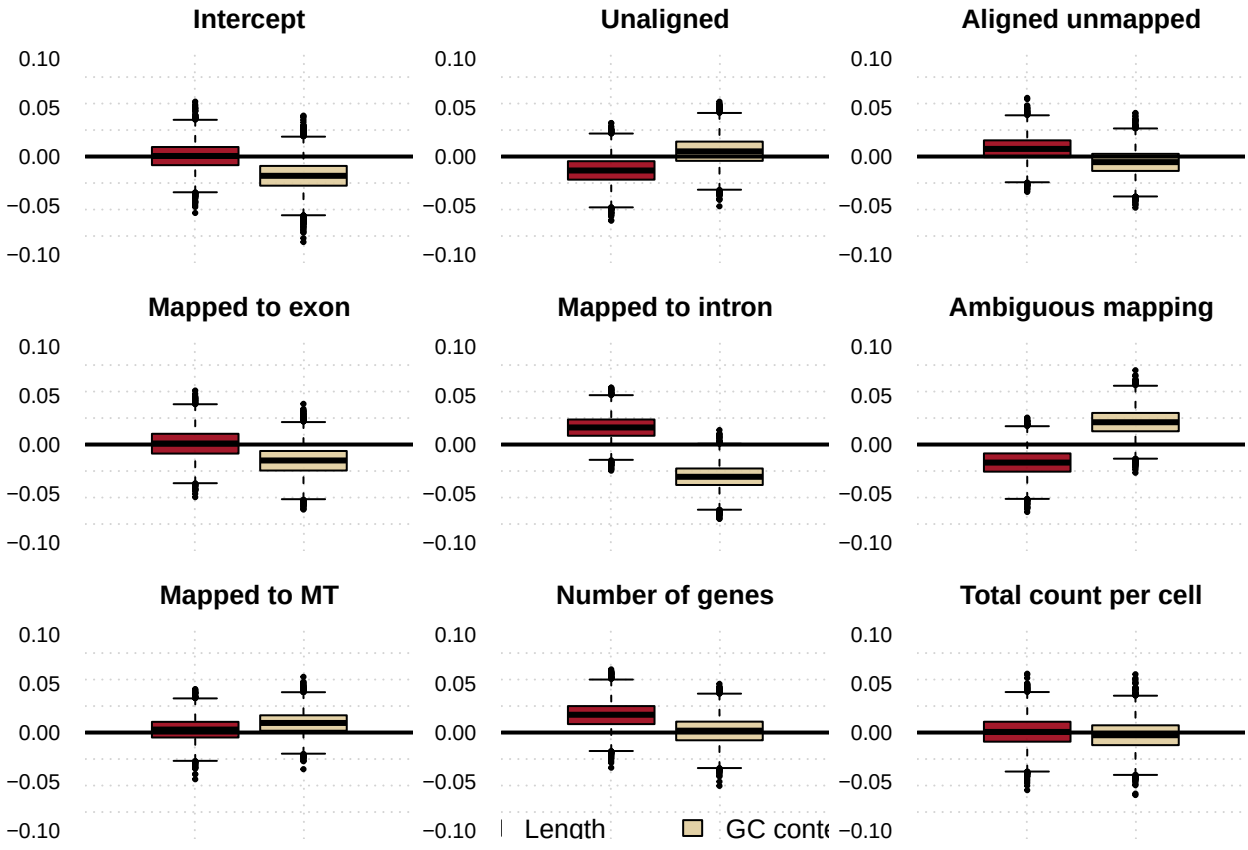
```
# computing credible intervals for mean coefficients
Q.mat <- matrix(NA, prod(dim(Mcmc$beta[[1]])), 2)
for(i in 1:nrow(Q.mat)) {
  Q.mat[i,]  <- quantile(sapply(Mcmc$beta, "[[",i), probs = c(0.05, 0.95))
}
# genes characterized by 0.9-level credible intervals not including zero
sign_coeff <- matrix(0, nrow(Mcmc$beta[[1]]), ncol(Mcmc$beta[[1]]))
sign_coeff[which(apply(Q.mat, 1, function(x) prod(x))>0)] <- 1
# summary results
data.frame("covariate" = colnames(Mcmc$x),
           "min" = apply(lpost_mean$beta, 2, min),
           "Q0.25" = apply(lpost_mean$beta, 2, function(var) quantile(var, 0.25)),
           "Q0.50" = apply(lpost_mean$beta, 2, function(var) quantile(var, 0.50)),
           "Q0.75" = apply(lpost_mean$beta, 2, function(var) quantile(var, 0.75)),
           "max" = apply(lpost_mean$beta, 2, max),
           "0.9-level" = colSums(sign_coeff))
```

```
##              covariate        min       Q0.25         Q0.50      Q0.75
## 1          (Intercept) -0.5043966  0.08896348  0.1642383659 0.27914230
## 2            unaligned -0.3707292 -0.06182066  0.0041213771 0.06741375
## 3    aligned_unmapped -0.6272238 -0.06678130  0.0104288659 0.07131877
## 4      mapped_to_exon -0.2175375 -0.01315483  0.0370949657 0.08842277
## 5    mapped_to_intron -0.2958537 -0.07517779 -0.0088672833 0.05504428
## 6   ambiguous_mapping -0.3794013 -0.05095050  0.0183782998 0.08612660
## 7        mapped_to_MT -0.2310874 -0.04517660 -0.0005902716 0.04437754
## 8     number_of_genes -0.3382839  0.04963785  0.1172476285 0.18856262
## 9 total_count_per_cell -0.1616019  0.01225842  0.0510486590 0.09333834
##         max X0.9.level
## 1 0.5791243         85
## 2 0.3256638          0
## 3 0.4133193         14
## 4 0.3383102          0
## 5 0.3780441         22
## 6 0.3685660          0
## 7 0.2915206         20
## 8 0.5165752         12
## 9 0.2605260          0
```

The next figure displays the posterior distributions of the Γ_T coefficients, illustrating the influence of meta-covariates on the covariate effects.

```
par(mfrow=c(3, 3), mar=c(0.5, 3.5, 2.5, 0.))
covariates_names <- c("Intercept", "Unaligned", "Aligned unmapped",
                      "Mapped to exon", "Mapped to intron", "Ambiguous mapping",
                      "Mapped to MT", "Number of genes", "Total count per cell")
for(i in 1:ncol(Mcmc$x)) {
  plot(1, 1, pch = "", xlim = c(0.1, 2.9), ylim = c(-0.1, 0.1),
```

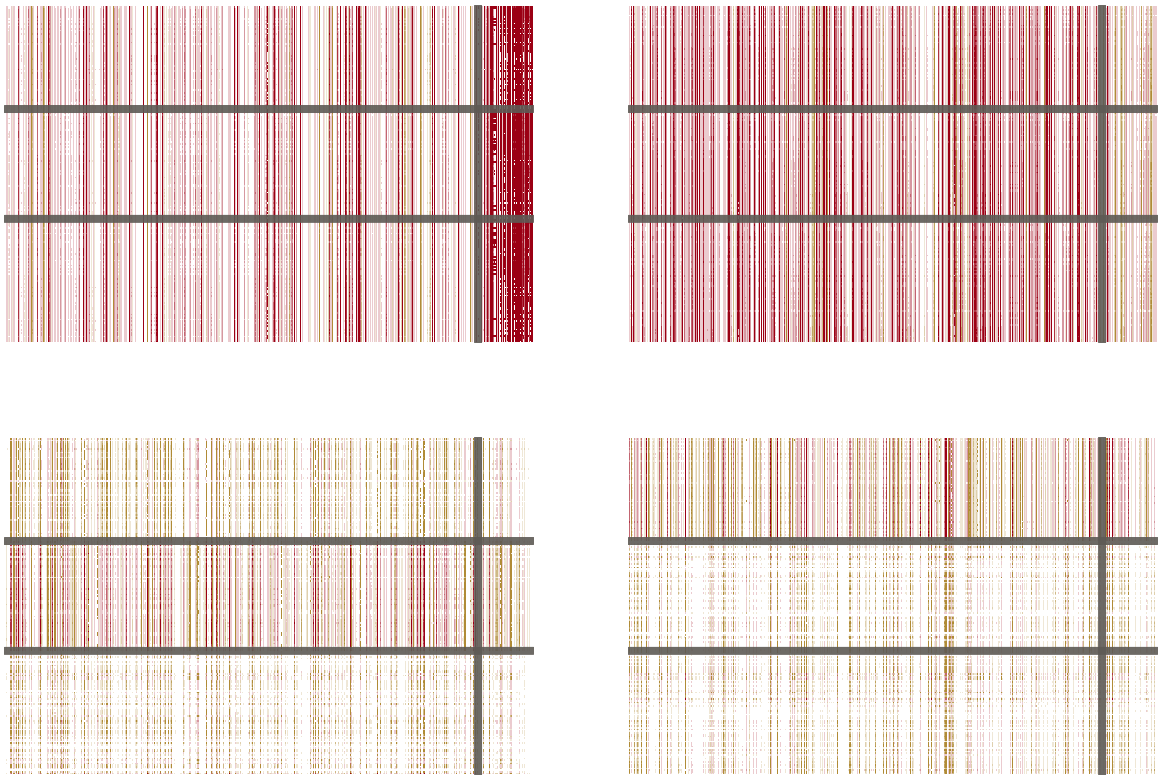
```
      axes = FALSE, xlab = "", ylab = "")
  grid(nx = 3, ny = 8)
  datatoplot <- cbind(sapply(Mcmc$gammaT, "[[",2+3*(i-1)),
                      sapply(Mcmc$gammaT, "[[",3+3*(i-1)))
  boxplot(datatoplot, col =  c(unipdred, mybeige),
          add = TRUE, axes = FALSE, pch = 20)
  axis(2, tick = FALSE, las = 1)
  abline(h = 0, lwd = 2)
  title(main = covariates_names[i])
  if(i == 8) {
    legend("bottom", legend = c("Length", "GC content"),
           fill = c(unipdred, mybeige), cex = 1.25, pt.cex = 2,
           bg = "white", horiz = TRUE, inset = -1/10, bty = "n")
  }
}
```



We focus now on the results obtained thanks to the innovative treatment of the residual term allowed by `COSIN`. The rank-one contributions C_h allow one to decompose the underlying signal in rank-one additive matrices which aid interpretation. We plot below the estimated C_1, C_3, C_4 and C_5: to facilitate interpretation, we re-arranged the row order according to cell lines and the column order according to the COVID-19 disease biological pathway, in particular the genes within the grey square on the right of each contribution matrix belong to the COVID-19 pathway.

```
cells_order <- order(x$cell_line_demuxlet)
# H1975 up to cell 73, H2228 up to 138, HCC827 others
cells_linebreaks <- c(73.5, 138.5)
```

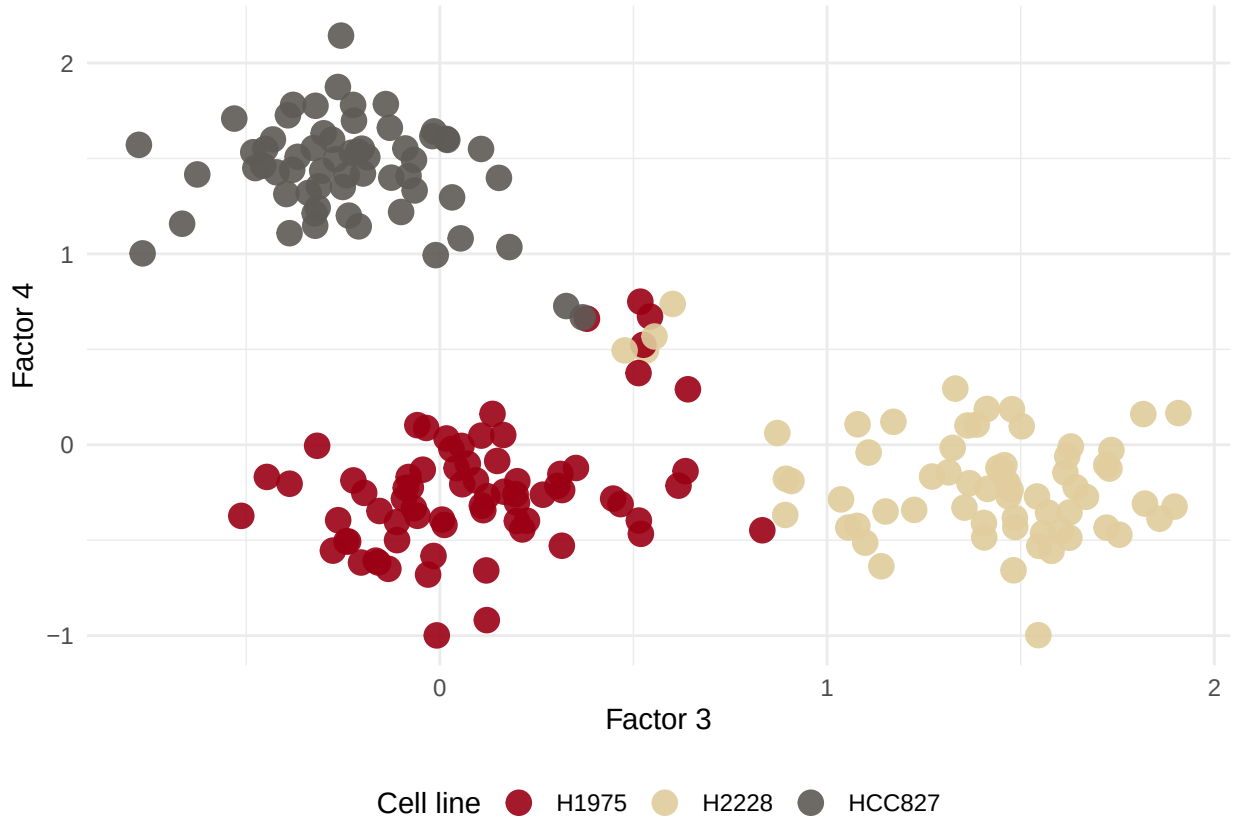
```
# pathway choice
col_idx <- 7  # Coronavirus_disease___COVID_19
pathway_order <- order(wB[, col_idx])
pathway_linebreak <- which(sort(wB[, col_idx+1])==1)[1] + 0.5
# scale regulation
breaks <- c(-0.5, seq(-0.1, -0.005, length = 5), seq(0.02, 0.8, length = 5), 3)
# plot
par(mfrow=c(2,2), mar=c(1.5,1.5,1.5,1.5))
n <- nrow(Mcmc$y); p <- ncol(Mcmc$y)
for(j in c(1,3,4,5)){
  fact_contr_ordered <- contribution_means$C1[[j]][cells_order, pathway_order]
  image(seq(1, p), seq(1, n), t(fact_contr_ordered), breaks = breaks,
        xlab = "", ylab = "", col = funcColor(11), axes = FALSE)
  abline(h = cells_linebreaks, lwd = 4, col = mygrey)
  abline(v = pathway_linebreak, lwd = 4, col = mygrey)
}
```



Then, to investigate the ability of the model in recognizing possibly interpretable latent unobserved covariates, we plot a representative posterior draw of the cell factor scores $\eta_{\cdot 3}$ and $\eta_{\cdot 4}$. Cells are coloured according to the cell lines in order to assess correspondence with the three clusters revealed by our approach.

```
factors_toplot <- data.frame(Factor_8 = lpost_mode$eta_max[, 8],
                             Factor_13 = lpost_mode$eta_max[, 13],
                             Cell_Line = x$cell_line_demuxlet)
ggplot(factors_toplot, aes(x= Factor_13, y= Factor_8, color=Cell_Line))+
  geom_point( size =4) +
```

```
  scale_color_manual("Cell line", values = c(unipdred, mybeige, mygrey )) +
  theme_minimal() + theme(legend.position = "bottom") +
  xlab("Factor 3") + ylab("Factor 4")
```

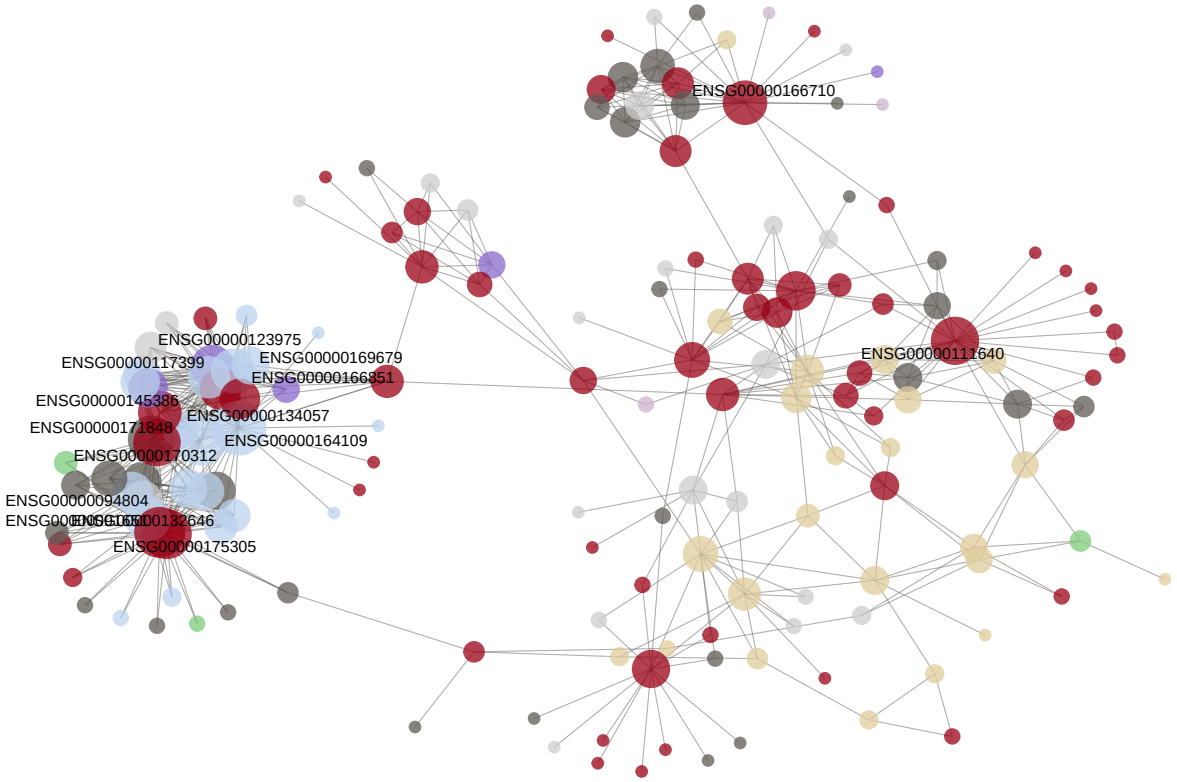


Finally, we construct a graph from the posterior mean of the inverse of the sparse covariance matrix $\Omega = \Lambda\Lambda^\top + \Sigma$, i.e. the partial correlation matrix.

```
threshold <- 0.025
pcorr_postmean_cutted <- lpost_mean$omega_pcorr
colnames(pcorr_postmean_cutted) <- rownames(wB)
pcorr_postmean_cutted[which(abs(lpost_mean$omega_pcorr)<threshold)] <- 0
td <- which(rowSums(abs(pcorr_postmean_cutted)>0)==1)
pcorr_postmean_cutted <- pcorr_postmean_cutted[-td,-td]

network <- graph_from_adjacency_matrix(pcorr_postmean_cutted, weighted = TRUE,
                                       mode = "undirected", diag = FALSE)
net <- network(abs(pcorr_postmean_cutted), ignore.eval = FALSE,
               names.eval = "weights")
wB_numeric <- matrix(NA, p, ncol(wB))
for(i in 1:ncol(wB)){
  wB_numeric[, i] <- as.numeric(wB[,i]) - 1
}
pathway <- rep("Multi-pathway", p)
for(m in 1:ncol(wB)){
  w_path <- which((wB[,m]=="1") & (rowSums(wB_numeric)==1))
  pathway[w_path] <- colnames(wB)[m]
```

```
}
net %v% "pathway" <- pathway[-td]
labels <- rep("", nrow(pcorr_postmean_cutted))
selected_labels <- order(rowSums(abs(pcorr_postmean_cutted)), decreasing = TRUE)[1:15]
labels[selected_labels] <- colnames(pcorr_postmean_cutted)[selected_labels]
# plot
set.seed(5)
ggnet2(net, mode = "fruchtermanreingold", size = "freeman", color="pathway",
       #label = labels, size.cut = 5, size.palette = c(),
       color.palette= c("Multi-pathway" = unipdred,
                        "Metabolic_pathways" = mygrey,
                        "Coronavirus_disease___COVID_19" = mybeige,
                        "Cell_cycle" = myavion,
                        "Pathways_in_cancer" = mypurple,
                        "MAPK_signaling_pathway" = mygreen,
                        "Proteoglycans_in_cancer" = mypink,
                        "Salmonella_infection" = grey(0.8,0.9),
                        "Shigellosis" = grey(0.8,0.9),
                        "Tight_junction" = grey(0.8,0.9),
                        "Amyotrophic_lateral_sclerosis" = grey(0.8,0.9),
                        "Regulation_of_actin_cytoskeleton" = grey(0.8,0.9),
                        "Human_immunodeficiency_virus_1_infection" = grey(0.8,0.9),
                        "Human_papillomavirus_infection" = grey(0.8,0.9),
                        "Parkinson_disease" = grey(0.8,0.9),
                        "PI3K_Akt_signaling_pathway" = grey(0.8,0.9)),
       alpha = 0.75, edge.size = "weights", edge.color = mygrey, edge.alpha = 0.3)+
       geom_text_repel(aes(label = labels), box.padding = 0.1, color="black",
                       size=2 ,point.padding = 0.1,  segment.color = 'black',
                       segment.size = 0.1, max.overlaps = 40) +
       theme(legend.position = "none") +
       guides(size = "none")
```



remark: Figure 3 was generated using this code, however it was modified further for display purposes.