Untitled

Project 3

Import packages

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.6
v forcats   1.0.1     v stringr   1.6.0
v ggplot2   4.0.1     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.2.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
library(igraph)
```

```
Attaching package: 'igraph'

The following objects are masked from 'package:lubridate':

    %--%, union

The following objects are masked from 'package:dplyr':

    as_data_frame, groups, union
```

```
The following objects are masked from 'package:purrr':

    compose, simplify

The following object is masked from 'package:tidyr':

    crossing

The following object is masked from 'package:tibble':

    as_data_frame

The following objects are masked from 'package:stats':

    decompose, spectrum

The following object is masked from 'package:base':

    union
```

```
library(pheatmap)
library(brainGraph)
library(reticulate)
library(ggpubr)
```

Import data and experimental design

Load sample data

```
counts_raw <- read.csv("data/expression_matrix.csv",
                       row.names = 1, header=FALSE)
col_meta <- read.csv("data/columns_metadata.csv", header = TRUE)
row_meta <- read.csv("data/rows_metadata.csv", header = TRUE)
```

Parse data

Create unique identifier

```r
col_meta <- col_meta %>% mutate(
  uid = paste0(donor_id, "_", structure_id)
)
```

Format row and column names in expression matrix

```r
row.names(counts_raw) <- make.unique(row_meta$gene_symbol)
colnames(counts_raw) <- col_meta$uid
```

Add Life Stage and System info to column metadata

```r
col_meta <- col_meta %>%
  mutate(
    LifeStage = case_when(
      # 1. PRENATAL (Conception to Birth)
      grepl("pcw", age) ~ "Prenatal",

      # 2. INFANCY (Birth to 4 yrs)
      age %in% c("4 mos", "10 mos", "1 yrs", "2 yrs", "3 yrs", "4 yrs") ~ "Infancy",

      # 3. ADOLESCENCE (8 yrs to 21 yrs)
      age %in% c("8 yrs", "11 yrs", "13 yrs", "15 yrs", "18 yrs", "19 yrs", "21 yrs") ~ "Adol

      # 4. ADULTHOOD (23+ yrs)
      age %in% c("23 yrs", "30 yrs", "36 yrs", "37 yrs", "40 yrs") ~ "Adult",
      )
  )
```

```r
col_meta <- col_meta %>%
  mutate(
    Fine_System = case_when(
      # --- SENSORY SYSTEMS ---
      structure_acronym %in% c("V1C", "Ocx", "ITC") ~ "Visual",
      structure_acronym %in% c("M1C", "S1C", "M1C-S1C") ~ "Somatomotor",
      structure_acronym %in% c("A1C", "STC", "TCx") ~ "Auditory_Temporal",

      # --- ASSOCIATION SYSTEMS ---
      structure_acronym %in% c("DFC", "VFC", "MFC") ~ "Frontal Executive",
      structure_acronym %in% c("IPC", "PCx") ~ "Parietal_Attn",
```

```
      # --- DEEP BRAIN SYSTEMS ---
      structure_acronym %in% c("OFC", "AMY", "HIP") ~ "Limbic System",
      structure_acronym %in% c("STR", "MD", "DTH") ~ "Core_Subcortex",
      structure_acronym %in% c("CB", "CBC") ~ "Cerebellum",

      # --- FETAL ONLY ---
      structure_acronym %in% c("MGE", "LGE", "CGE", "URL") ~ "Fetal_Transient",
      )
  )
```

```
systems <- c("Visual", "Somatomotor", "Auditory_Temporal",
             "Frontal Executive", "Parietal_Attn", "Limbic System",
             "Core_Subcortex")
stage <- c("Prenatal", "Infancy", "Adolescence", "Adult")

col_meta$LifeStage <- factor(col_meta$LifeStage, levels = stage)
col_meta$Fine_System <- factor(col_meta$Fine_System, levels = systems)
```

```
col_meta %>% group_by(donor_id, Fine_System, LifeStage) %>%
  summarise(Count = n()) %>%
  filter(Count >=3) %>% group_by(Fine_System, LifeStage) %>%
  summarize(n())
```

```
`summarise()` has grouped output by 'donor_id', 'Fine_System'. You can override
using the `.groups` argument.
`summarise()` has grouped output by 'Fine_System'. You can override using the
`.groups` argument.
```

```
# A tibble: 9 x 3
# Groups:   Fine_System [3]
  Fine_System       LifeStage   `n()`
  <fct>             <fct>       <int>
1 Frontal Executive Prenatal       14
2 Frontal Executive Infancy         4
3 Frontal Executive Adolescence     7
4 Frontal Executive Adult           3
5 Limbic System     Prenatal       13
6 Limbic System     Infancy         3
7 Limbic System     Adolescence     5
8 Limbic System     Adult           5
9 <NA>              Prenatal        2
```

```
validDonors <- col_meta %>% group_by(donor_id, LifeStage, Fine_System) %>% summarise(Count =
```

```
`summarise()` has grouped output by 'donor_id', 'LifeStage'. You can override
using the `.groups` argument.
```

```
head(validDonors)
```

```
# A tibble: 6 x 4
# Groups:   donor_id, LifeStage [4]
  donor_id LifeStage   Fine_System       Count
     <int> <fct>       <fct>             <int>
1    12287 Prenatal    Frontal Executive     3
2    12288 Prenatal    Frontal Executive     3
3    12288 Prenatal    Limbic System         3
4    12289 Adolescence Frontal Executive     3
5    12289 Adolescence Limbic System         3
6    12290 Adult       Frontal Executive     3
```

```
validDonors_meta <- data.frame()

for (system in systems) {
  donors_sys <- validDonors %>% filter(Fine_System == system)
  sys_data <- col_meta %>% filter(
    Fine_System == system,
    donor_id %in% donors_sys$donor_id)

  validDonors_meta <- rbind(validDonors_meta, sys_data)
  }
```

```
grouping_info <- validDonors_meta %>%
  mutate(
    SampleID = paste0(donor_id, LifeStage, Fine_System, sep="_")
  )

global_counts <- counts_raw[, validDonors_meta$uid]
```

```
keep_genes <- rowSums(global_counts >= 1) >= 8 # >=1 because data is already Log2

clean_counts <- global_counts[keep_genes, ]
```

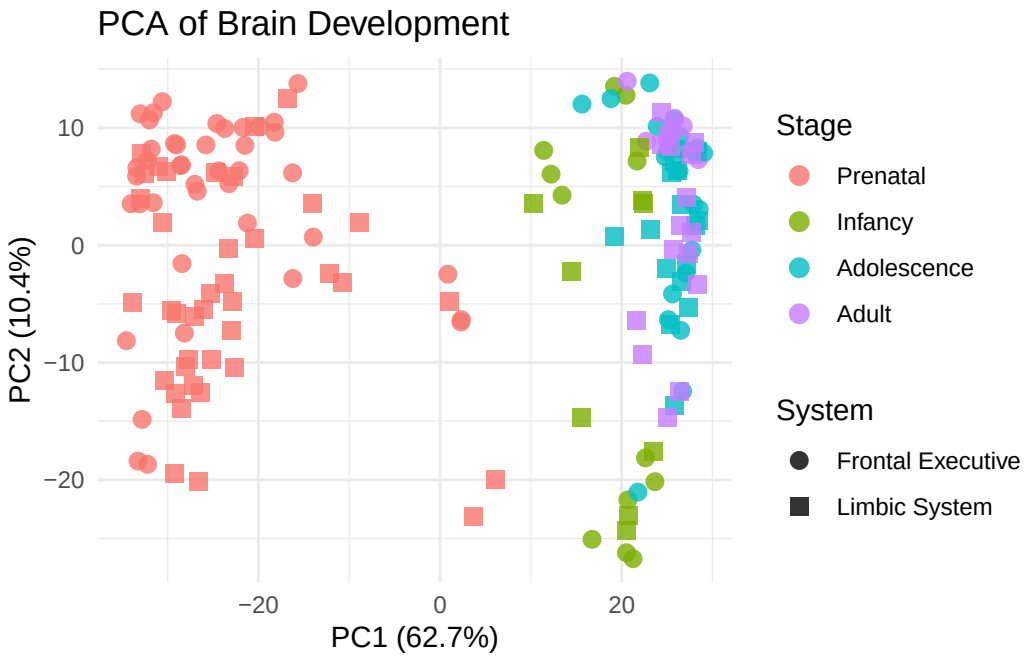
Quality check data

Data clustering

```
vsd_mat <- log2(clean_counts + 1)
gene_vars <- apply(vsd_mat, 1, var)
top_genes <- order(gene_vars, decreasing=TRUE)[1:1000]
vsd_mat <- vsd_mat[top_genes, ]
```

PCA

```
pca_res <- prcomp(t(vsd_mat), scale. = TRUE)
var_explained <- round(100 * summary(pca_res)$importance[2, 1:2], 1)

pca_df <- data.frame(PC1 = pca_res$x[,1],
                     PC2 = pca_res$x[,2],
                     Donor = as.character(validDonors_meta$donor_id),
                     Stage = validDonors_meta$LifeStage,
                     System = validDonors_meta$Fine_System)

p <- ggplot(pca_df, aes(x = PC1, y = PC2, color = Stage, shape = System)) +
  geom_point(size = 3, alpha = 0.8) +
  scale_shape_manual(values = c(16, 15, 17, 18, 3, 4, 8, 6)) +
  guides(color = guide_legend(order = 1),
         shape = guide_legend(order = 2)) +
  labs(title = "PCA of Brain Development",
       x = paste0("PC1 (", var_explained[1], "%)"),
       y = paste0("PC2 (", var_explained[2], "%)")) +
  theme_minimal()

p
```



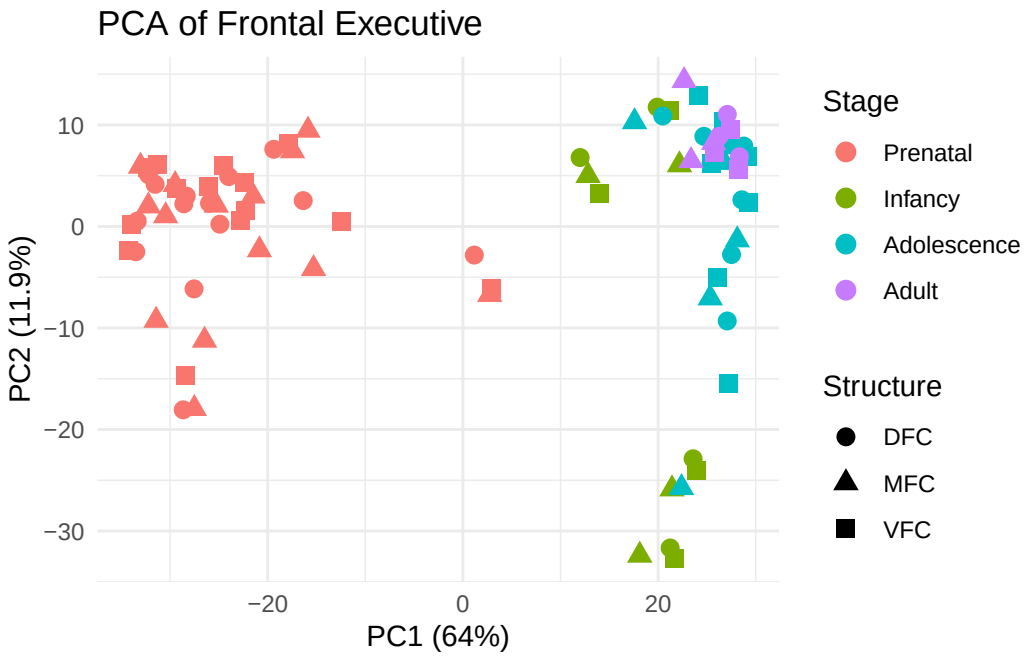

```
# save image
ggsave(filename = "results/global_PCA.png",
       plot = p, width = 6, height = 4)
```

PCA per systems

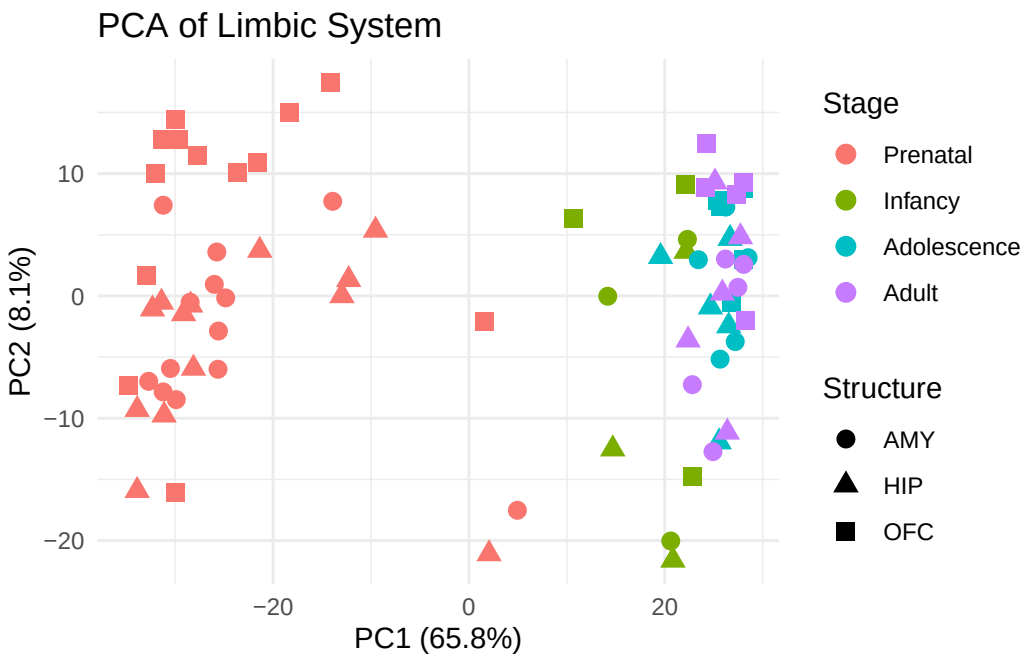
```
system_pca <- list()
for (sys in unique(validDonors_meta$Fine_System)) {
  system_samples <- validDonors_meta %>%
    filter(Fine_System == sys)

  sys_counts <- clean_counts[, system_samples$uid]

  vsd_sys <- log2(sys_counts + 1)
  gene_vars_sys <- apply(vsd_sys, 1, var)
  top_genes_sys <- order(gene_vars_sys, decreasing=TRUE)[1:1000]
  vsd_sys <- vsd_sys[top_genes_sys,]

  pca_res <- prcomp(t(vsd_sys), scale. = TRUE)

  var_explained <- round(100 * summary(pca_res)$importance[2, 1:2], 1)

  pca_df <- data.frame(PC1 = pca_res$x[,1],
```

```
                        PC2 = pca_res$x[,2],
                        Structure = as.character(system_samples$structure_acronym),
                        Stage = system_samples$LifeStage)

  p<- ggplot(pca_df, aes(x = PC1, y = PC2, color = Stage, shape = Structure)) +
    geom_point(size = 3) +
    theme_minimal() +
    guides(color = guide_legend(order = 1),
           shape = guide_legend(order = 2))  +
    labs(title = paste("PCA of", sys),
         x = paste0("PC1 (", var_explained[1], "%)"),
         y = paste0("PC2 (", var_explained[2], "%)")
         )

  #save image
  ggsave(filename = paste0("results/", sys, "_PCA.png"),
         plot = p, width = 6, height = 4)
  system_pca[[sys]] <- p
}

system_pca
```

```
$`Frontal Executive`
```



$`Limbic System`

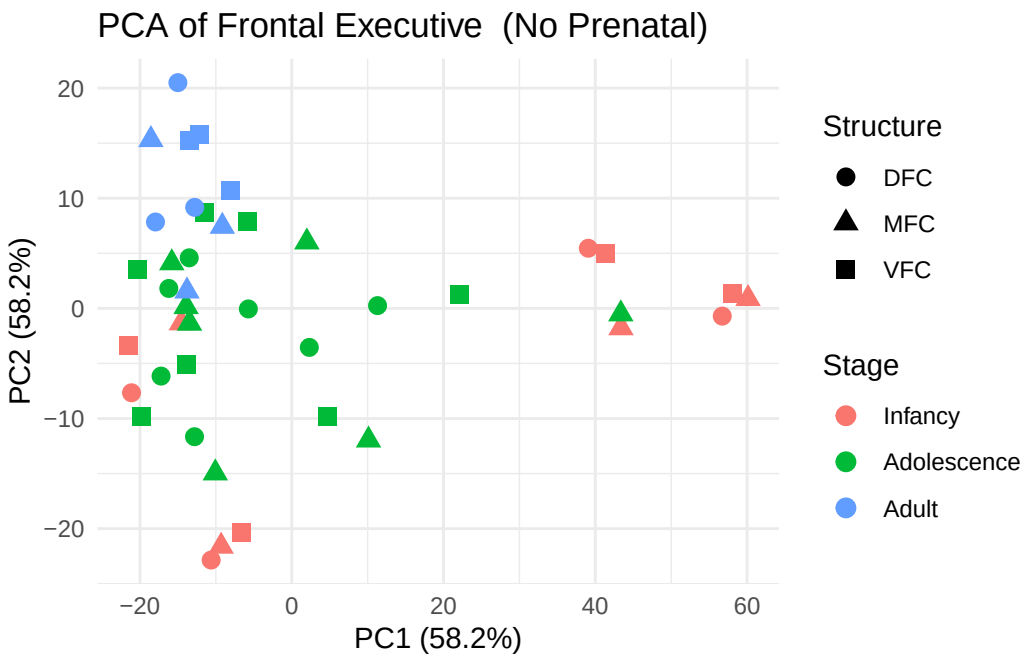


PCA plots without prenatal

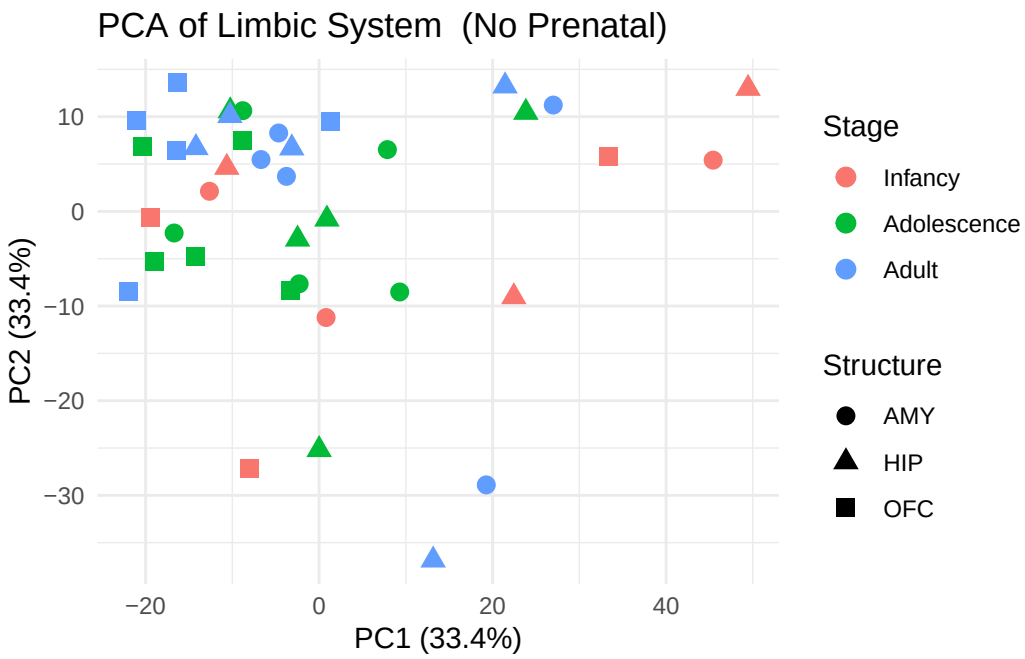
```
system_pca_no_prenatal <- list()
for (sys in unique(validDonors_meta$Fine_System)) {
  system_samples <- validDonors_meta %>%
    filter(Fine_System == sys, LifeStage != "Prenatal")
  sys_counts <- clean_counts[, system_samples$uid]

  vsd_sys <- log2(sys_counts + 1)
  gene_vars_sys <- apply(vsd_sys, 1, var)
  top_genes_sys <- order(gene_vars_sys, decreasing=TRUE)[1:1000]
  vsd_sys <- vsd_sys[top_genes_sys,]

  pca_res <- prcomp(t(vsd_sys), scale. = TRUE)

  var_explained <- round(100 * summary(pca_res)$importance[2, 1:2], 1)

  pca_df <- data.frame(PC1 = pca_res$x[,1],
                       PC2 = pca_res$x[,2],
```

```
                       Structure =
                      as.character(system_samples$structure_acronym),
                       Stage = system_samples$LifeStage)
  p<- ggplot(pca_df, aes(x = PC1, y = PC2, color = Stage, shape = Structure)) +
    geom_point(size = 3) +
    theme_minimal() +
    labs(title = paste("PCA of", sys, " (No Prenatal)"),
         x = paste0("PC1 (", var_explained[1], "%)") ,
         y = paste0("PC2 (", var_explained[1], "%)")
         )

  #save image
  ggsave(filename = paste0("results/", sys, "_PCA_no_prenatal.png"),
         plot = p, width = 6, height = 4)

  system_pca_no_prenatal[[sys]] <- p
}

system_pca_no_prenatal
```

```
$`Frontal Executive`
```



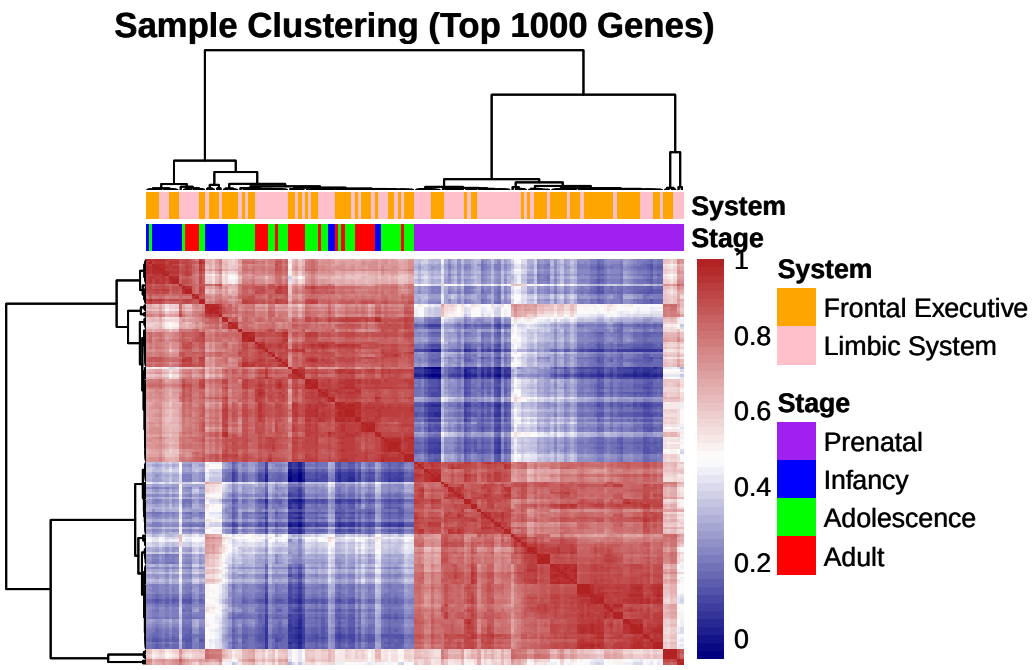
$`Limbic System`



Sample-to-sample correlation

```
# Calculate Sample-to-Sample Correlation
sample_dist <- cor(vsd_mat)

# Create Annotation Bar for the side
# Again, use validDonors_meta to ensure alignment
anno_col <- data.frame(
  Stage = validDonors_meta$LifeStage,
  System = validDonors_meta$Fine_System
  #donorID = as.character(validDonors_meta_grouped$donor_id)
)
rownames(anno_col) <- colnames(sample_dist)

# Plot
map <- pheatmap(sample_dist,
         main = "Sample Clustering (Top 1000 Genes)",
         annotation_col = anno_col,
         annotation_colors = list(
```

```
            Stage = c("Prenatal" = "purple",
                      "Infancy" = "blue",
                      "Adolescence" = "green",
                      "Adult" = "red"),
            System = c("Frontal Executive" = "orange",
                       "Limbic System" = "pink"
                       )),
          show_rownames = FALSE,
          show_colnames = FALSE,
          clustering_distance_rows = "correlation",
          clustering_distance_cols = "correlation",
          color = colorRampPalette(c("navy", "white", "firebrick"))(50))

map
```




```
# save image
ggsave(filename = "results/sample_correlation_heatmap.png",
       plot = map, width = 6, height = 4)
```

Build individual networks and measure topology and controllability

```
vsd_systems <- list()
topology_results <- data.frame()

for (sys in unique(validDonors_meta$Fine_System)) {

  print(paste("Processing System:", sys))

  system_samples <- validDonors_meta %>%
    filter(Fine_System == sys)

  system_counts <- clean_counts[, system_samples$uid]
  vsd_systems[[sys]] <- log2(system_counts + 1)
  gene_vars <- apply(vsd_systems[[sys]], 1, var)
  top_genes <- order(gene_vars, decreasing=TRUE)[1:300]
  vsd_systems[[sys]] <- vsd_systems[[sys]][top_genes, ]

  for (stage in unique(validDonors_meta$LifeStage)) {
    print(paste("  Processing Stage:", stage))
    stage_samples <- system_samples %>%
      filter(LifeStage == stage)

    for (d in stage_samples$donor_id) {
          contains_d <- grepl(d, stage_samples$uid)
          sub_vsd <- vsd_systems[[sys]][,
                                        stage_samples$uid[contains_d],
                                        drop = FALSE]
          stage_vars <- apply(sub_vsd, 1, var)
          valid_genes <- stage_vars > 0

          sub_vsd <- sub_vsd[valid_genes, ]

          adj <- cor(t(sub_vsd), method = "pearson")

          adj <- abs(adj)^2
          diag(adj) <- 0
          adj[is.na(adj)] <- 0

          #edge_weights <- adj[upper.tri(adj)]
          #cutoff <- quantile(edge_weights, 0.75)
          #adj[adj < cutoff] <- 0
```

```
          # Network topology metrics

           g <- graph_from_adjacency_matrix(adj,
                                            mode = "undirected",
                                            weighted = TRUE)

           g_dist <- g
           E(g_dist)$weight <- 1 / E(g)$weight
           glob_eff <- efficiency(g_dist, type = "global")

           clust_coeff <- transitivity(g, type = "global")

           comm <- cluster_louvain(g)
           mod_score <- modularity(comm)

           topology_results <- rbind(topology_results, data.frame(
             Donor = d,
             System = sys,
             Stage = stage,
             #Clustering = clust_coeff,
             Efficiency = glob_eff,
             Modularity = mod_score
          ))
    }
  }
}
```

```
[1] "Processing System: Frontal Executive"
[1] "  Processing Stage: Prenatal"
[1] "  Processing Stage: Infancy"
[1] "  Processing Stage: Adolescence"
[1] "  Processing Stage: Adult"
[1] "Processing System: Limbic System"
[1] "  Processing Stage: Prenatal"
[1] "  Processing Stage: Infancy"
[1] "  Processing Stage: Adolescence"
[1] "  Processing Stage: Adult"
```

```
py_require("nctpy")
nct <- import("nctpy.metrics")
utils <- import("nctpy.utils")
```

```r
control_results <- data.frame()

for (sys in unique(validDonors_meta$Fine_System)) {

  print(paste("Processing System:", sys))

  system_samples <- validDonors_meta %>% filter(Fine_System == sys)
  system_counts <- clean_counts[, system_samples$uid]

  vsd_temp <- log2(system_counts + 1)
  gene_vars <- apply(vsd_temp, 1, var)
  top_genes <- order(gene_vars, decreasing=TRUE)[1:300]
  vsd_sys <- vsd_temp[top_genes, ]

  for (stag in unique(validDonors_meta$LifeStage)) {

    stage_samples <- system_samples %>% filter(LifeStage == stag)

    for (d in stage_samples$donor_id) {

      contains_d <- grepl(d, stage_samples$uid)

      sub_vsd <- vsd_sys[, stage_samples$uid[contains_d], drop = FALSE]

      # Filter constant genes for this specific donor
      valid_genes <- apply(sub_vsd, 1, var) > 0
      sub_vsd <- sub_vsd[valid_genes, ]

      adj <- cor(t(sub_vsd), method = "pearson")

      adj <- abs(adj)^2
      diag(adj) <- 0
      adj[is.na(adj)] <- 0

      if (max(adj) > 0) {

        # Normalize for Control Theory
        A_norm <- utils$matrix_normalization(adj, system = "discrete")

        # Average Controllability
        ac_vector <- nct$ave_control(A_norm, system = "discrete")
        mean_avg_ctrl <- mean(as.numeric(ac_vector))
```

```
        # Modal Controllability
        mc_vector <- nct$modal_control(A_norm)
        mean_mod_ctrl <- mean(as.numeric(mc_vector))

        control_results <- rbind(control_results, data.frame(
          Donor = d,
          System = sys,
          Stage = stag,
          Avg_Controllability = mean_avg_ctrl,
          Mod_Controllability = mean_mod_ctrl
        ))
      }
    }
  }
}
```

```
[1] "Processing System: Frontal Executive"
[1] "Processing System: Limbic System"
```

```
    ev <- eigen(adj)$values
    max_lambda <- max(Re(ev))

    if(max_lambda > 0) {
        A_norm <- adj / (max_lambda * 1.00001)
    } else {
        A_norm <- adj
    }
```

Combine topology and controllability results

```
results <- topology_results %>%
  left_join(control_results,
            by = c("Donor", "System", "Stage"))
```

```
Warning in left_join(., control_results, by = c("Donor", "System", "Stage")): Detected an unexpected many-to-many relationship between `x` and `y`.
i Row 1 of `x` matches multiple rows in `y`.
i Row 1 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

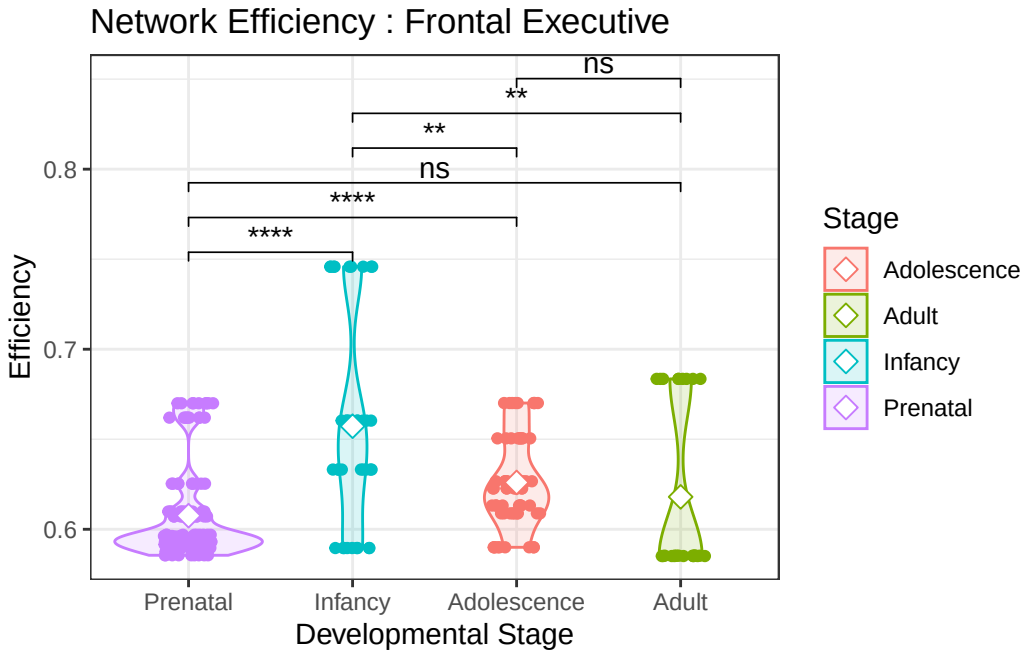
```r
head(results)
```

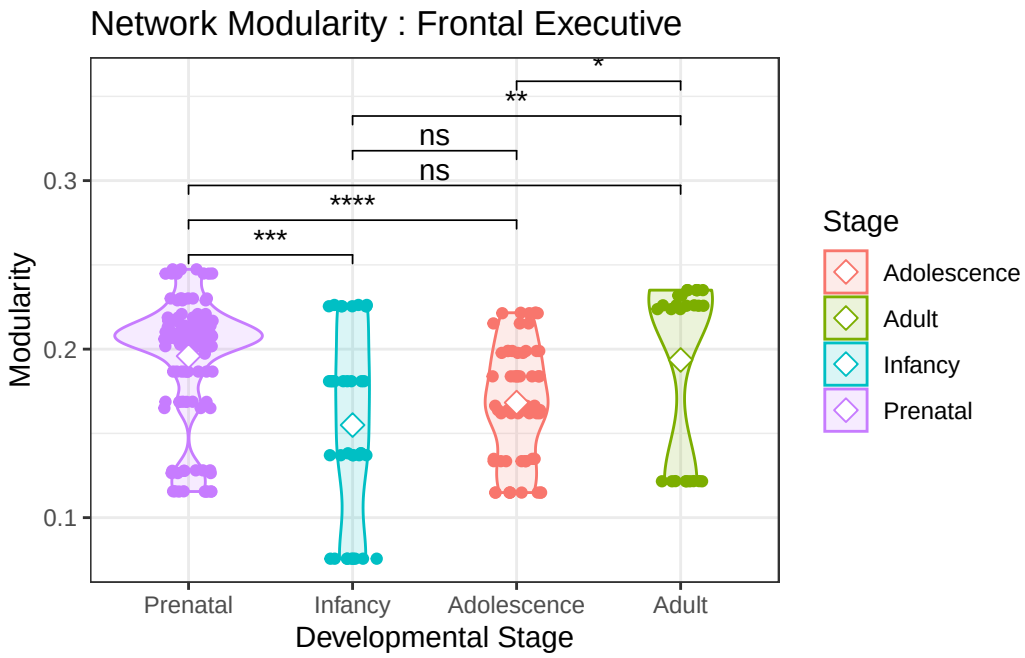
```
  Donor            System    Stage Efficiency Modularity Avg_Controllability
1 13058 Frontal Executive Prenatal  0.5930697  0.2099931              1.26371
2 13058 Frontal Executive Prenatal  0.5930697  0.2099931              1.26371
3 13058 Frontal Executive Prenatal  0.5930697  0.2099931              1.26371
4 13058 Frontal Executive Prenatal  0.5930697  0.2026925              1.26371
5 13058 Frontal Executive Prenatal  0.5930697  0.2026925              1.26371
6 13058 Frontal Executive Prenatal  0.5930697  0.2026925              1.26371
  Mod_Controllability
1           0.9952612
2           0.9952612
3           0.9952612
4           0.9952612
5           0.9952612
6           0.9952612
```

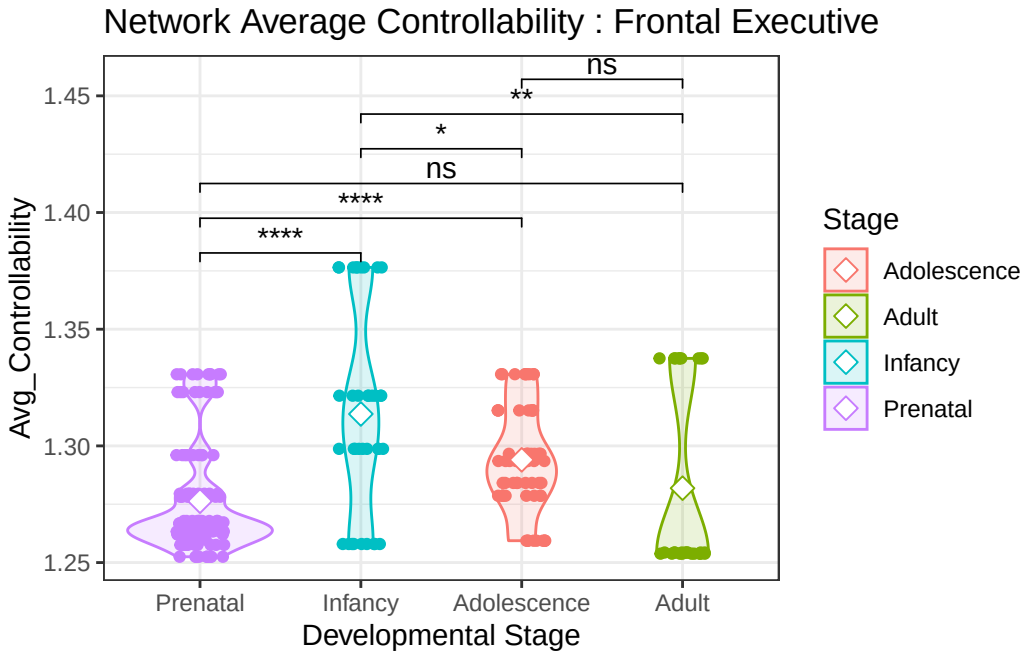
Statistics and Plots

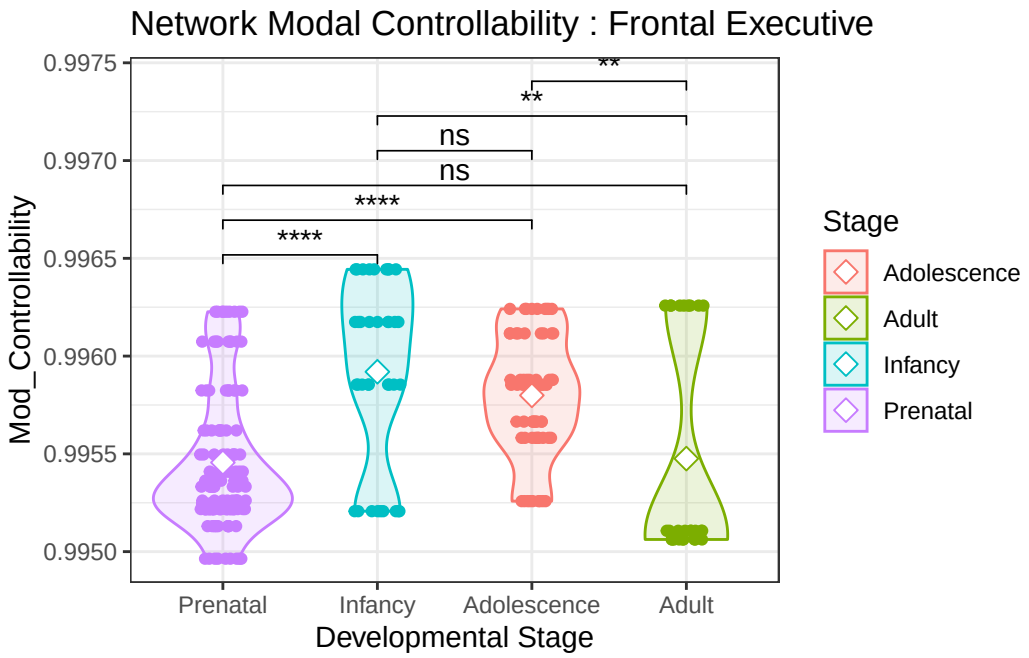
```r
stage <- c("Prenatal", "Infancy", "Adolescence", "Adult")

for (sys in unique(results$System)) {
  system_data <- results %>%
    filter(System == sys)

  stages <- unique(system_data$Stage)
  comparison_list <- combn(stages, 2, simplify = FALSE)

  for (met in colnames(system_data[4:ncol(system_data)])) {
    nameMetrics <- case_when(
      met == "Avg_Controllability" ~ "Average Controllability",
      met == "Mod_Controllability" ~ "Modal Controllability",
      TRUE ~ met
    )

    p <- ggplot(system_data, aes(x = Stage, y = .data[[met]],
                                 fill = Stage, color = Stage)) +
      geom_violin(alpha = 0.15) +
      geom_point(position=position_jitter(h=0.0,w=0.15)) +
```

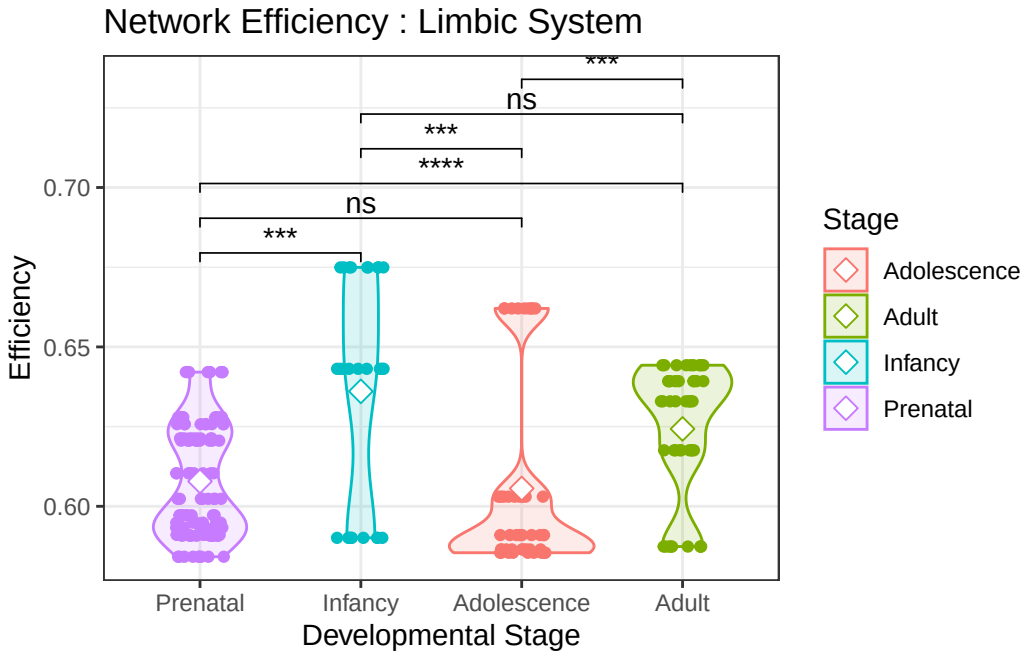
```
      stat_compare_means(comparisons = comparison_list,
                         label = "p.signif",
                         method = "t.test",
                         p.adjust.method = "bonferroni",
                         show.legend = FALSE) +

      # add mean
      stat_summary(fun = mean, geom = "point", shape = 23, size = 3,
                   fill = "white") +

      scale_x_discrete(limits = stages) +

      theme_bw() +
      labs(title = paste("Network", nameMetrics, ":", sys),
           x = "Developmental Stage",
           y = paste0(met))
    print(p)

    #save image
    ggsave(filename = paste0("results/", sys, "_", met, "_violin_plot.png"),
           plot = p, width = 6, height = 4)
  }
}
```

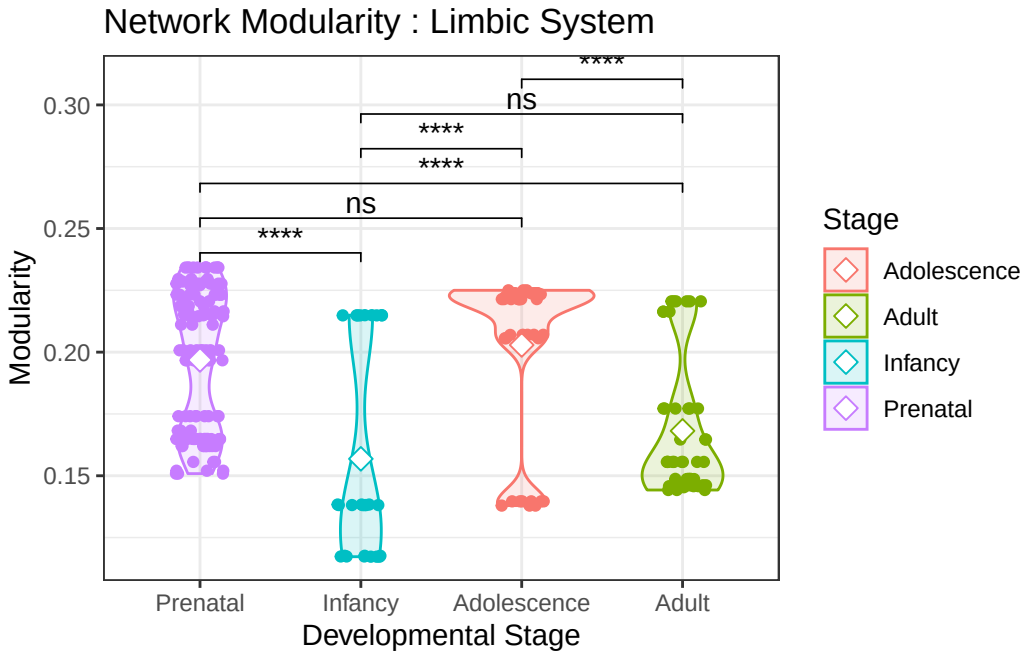


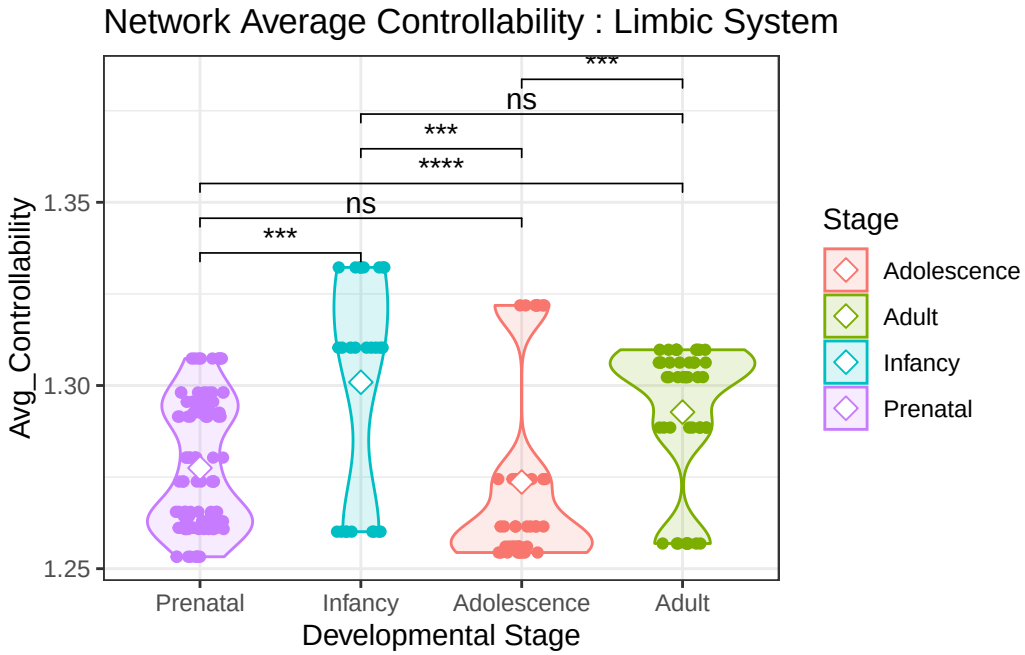

Network Modularity : Frontal Executive
Modularity
0.3
0.2
0.1
*
**
ns
ns
****
***
Prenatal
Infancy
Adolescence
Adult
Developmental Stage
Stage
Adolescence
Adult
Infancy
Prenatal

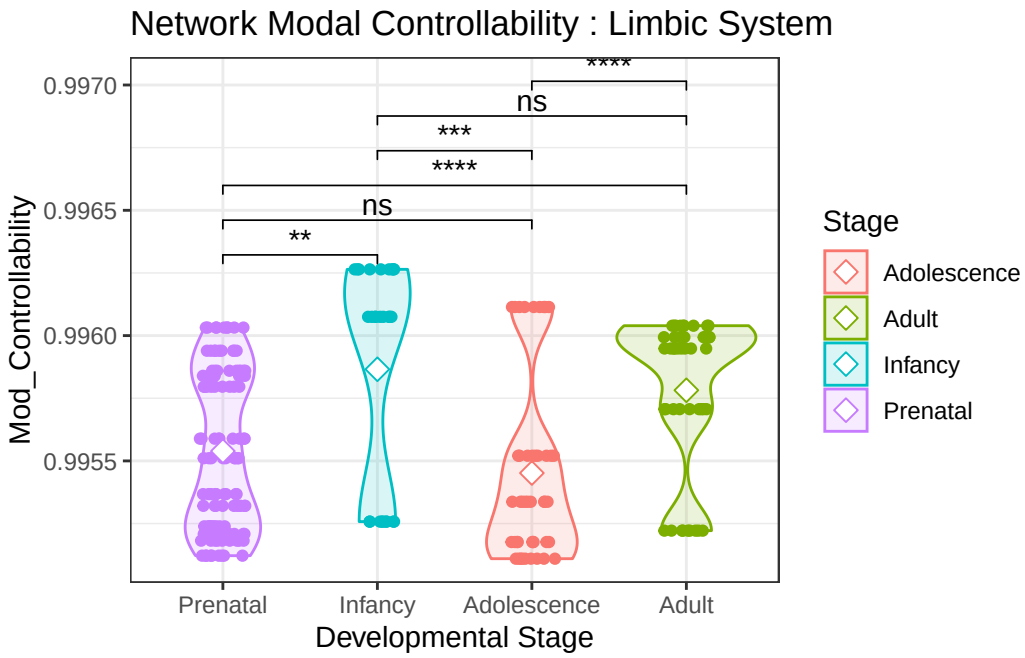

Network Average Controllability : Frontal Executive
Avg_Controllability
1.45
1.40
1.35
1.30
1.25
ns
**
*
ns
****
****
Prenatal
Infancy
Adolescence
Adult
Developmental Stage
Stage
Adolescence
Adult
Infancy
Prenatal


Network Modal Controllability : Frontal Executive
Mod_Controllability
0.9975
0.9970
0.9965
0.9960
0.9955
0.9950
**
**
ns
ns
****
****
Prenatal
Infancy
Adolescence
Adult
Developmental Stage
Stage
Adolescence
Adult
Infancy
Prenatal


Network Efficiency : Limbic System
Efficiency
0.70
0.65
0.60
***
ns
***
****
ns
***
Prenatal
Infancy
Adolescence
Adult
Developmental Stage
Stage
Adolescence
Adult
Infancy
Prenatal


Network Modularity : Limbic System
Modularity
0.30
0.25
0.20
0.15
****
ns
****
****
ns
****
Prenatal
Infancy
Adolescence
Adult
Developmental Stage
Stage
Adolescence
Adult
Infancy
Prenatal

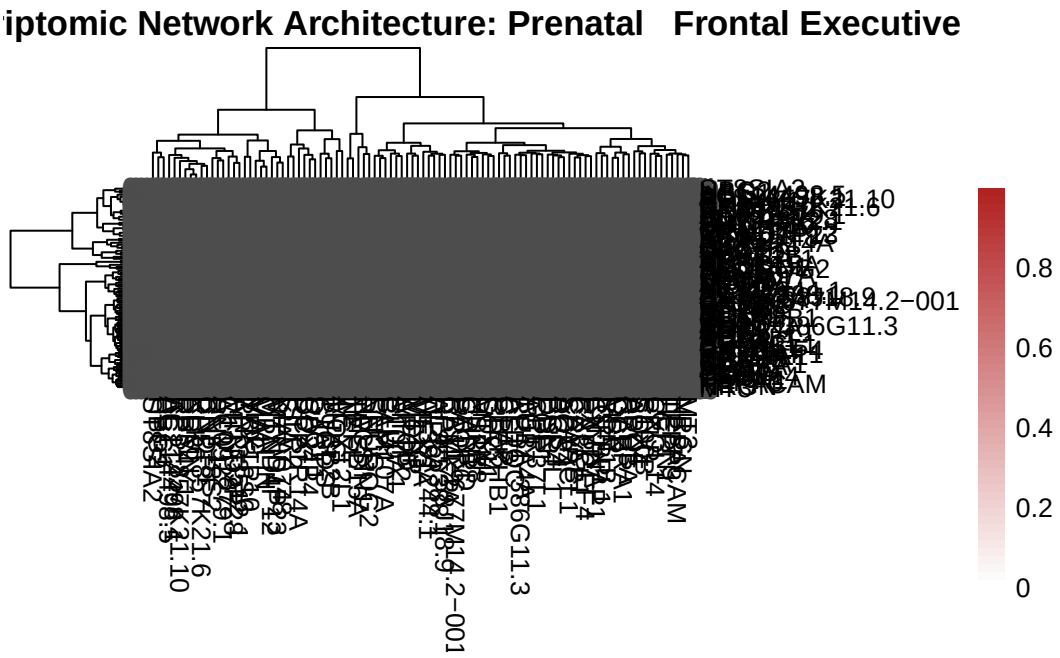

Network Average Controllability : Limbic System
Avg_Controllability
1.35
1.30
1.25
***
ns
***
****
***
Prenatal
Infancy
Adolescence
Adult
Developmental Stage
Stage
Adolescence
Adult
Infancy
Prenatal

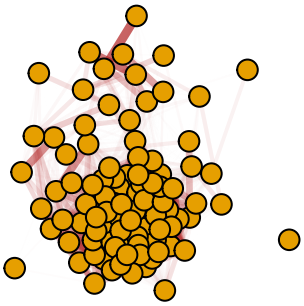


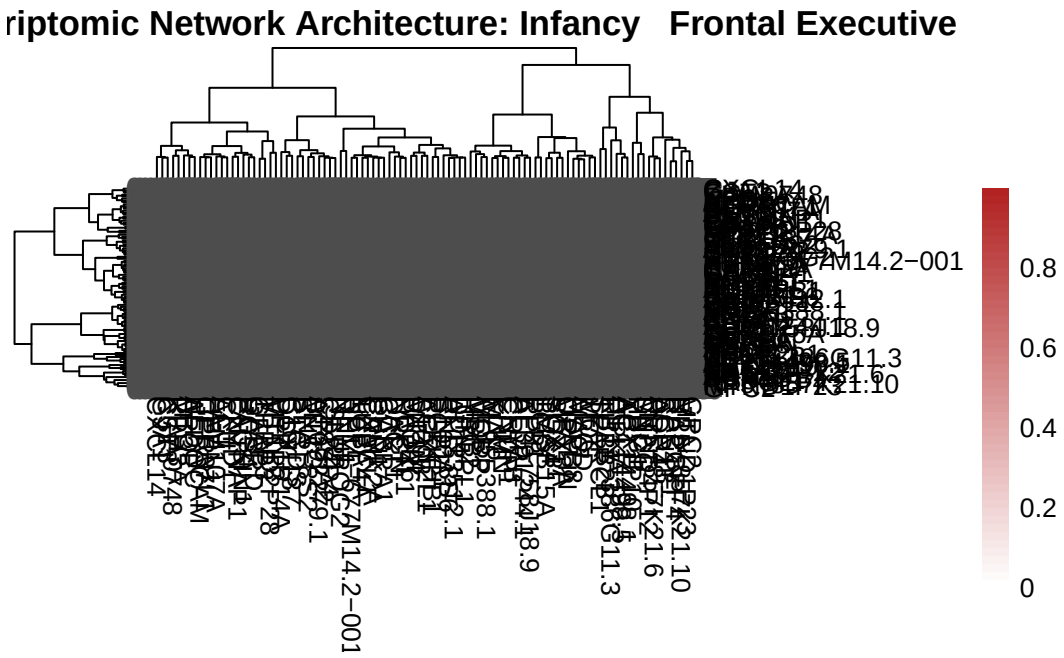
Visualize networks

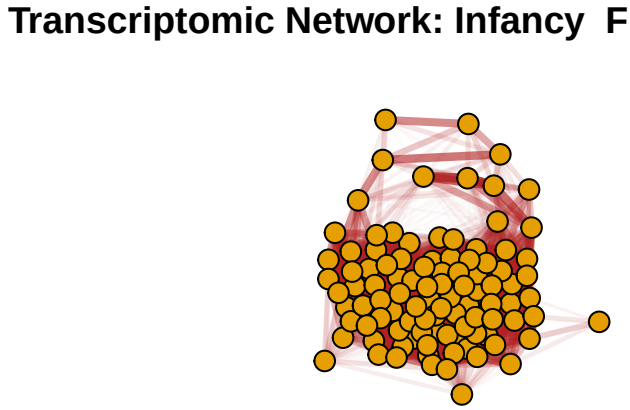
```
for (sys in unique(validDonors_meta$Fine_System)) {
  system_samples <- validDonors_meta %>%
    filter(Fine_System == sys)

  system_counts <- clean_counts[, system_samples$uid]
  vsd_systems[[sys]] <- log2(system_counts + 1)
  gene_vars <- apply(vsd_systems[[sys]], 1, var)
  top_genes <- order(gene_vars, decreasing=TRUE)[1:100]
  vsd_systems[[sys]] <- vsd_systems[[sys]][top_genes, ]


  for (stag in unique(validDonors_meta$LifeStage)) {
    system_stage_samples <- system_samples %>%
      filter(LifeStage == stag)

    sub_vsd <- vsd_systems[[sys]][,system_stage_samples$uid]
    stage_vars <- apply(sub_vsd, 1, var)
    valid_genes <- stage_vars > 0

    sub_vsd <- sub_vsd[valid_genes, ]
```

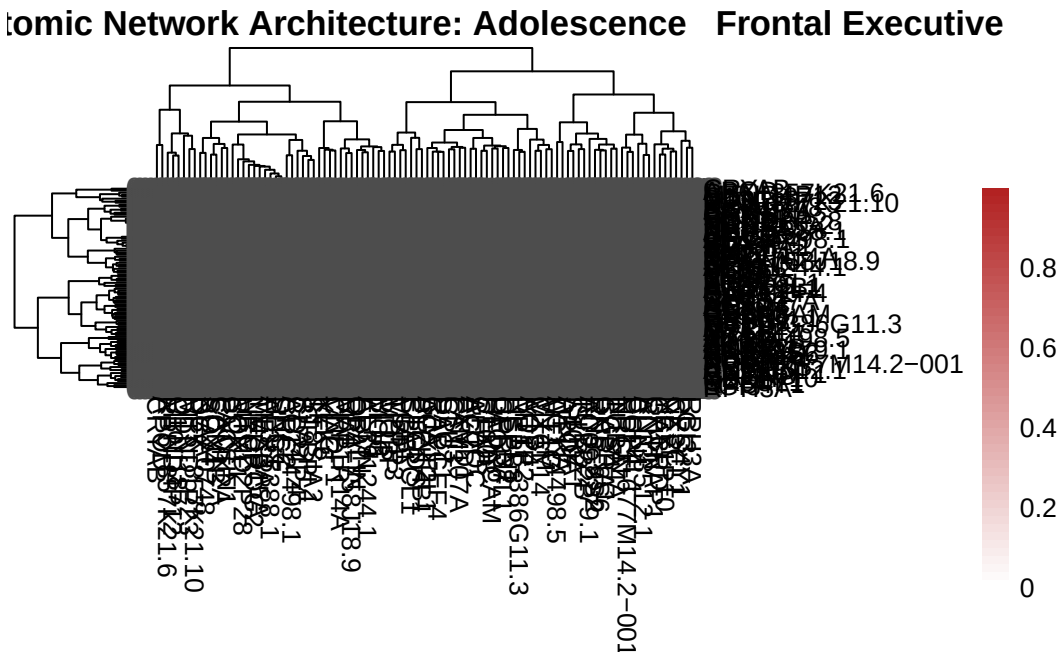
```
    adj <- cor(t(sub_vsd), method = "pearson")


    adj <- abs(adj)^2
    diag(adj) <- 0
    adj[is.na(adj)] <- 0

    map <- pheatmap(adj,
         main = paste("Transcriptomic Network Architecture:",
                      stag, " ", sys),
         cluster_rows = TRUE,    # Cluster similar systems together
         cluster_cols = TRUE,
         display_numbers = TRUE, # Show the correlation values
         fontsize_number = 12,
         color = colorRampPalette(c("white", "firebrick"))(50))

    map

    g <- graph_from_adjacency_matrix(adj,
                                     mode = "undirected",
                                     weighted = TRUE)
    E(g)$width <- E(g)$weight * 10
    edge_colors <- sapply(E(g)$weight,
                          function(w) alpha("firebrick", w^3))

    p <- plot(g,
         vertex.label = NA,
         vertex.size = 15,

         edge.width = E(g)$weight * 5,
         main = paste("Transcriptomic Network:",
                      stag, "", sys),
         edge.color = edge_colors
         )

    p


  }

}
```

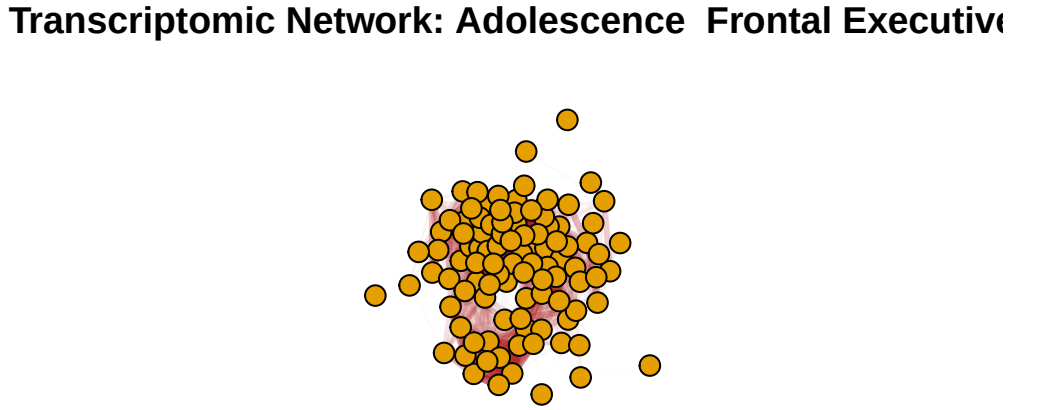

iptomic Network Architecture: Prenatal Frontal Executive
0.8
0.6
0.4
0.2
0


Transcriptomic Network: Prenatal Frontal Executive

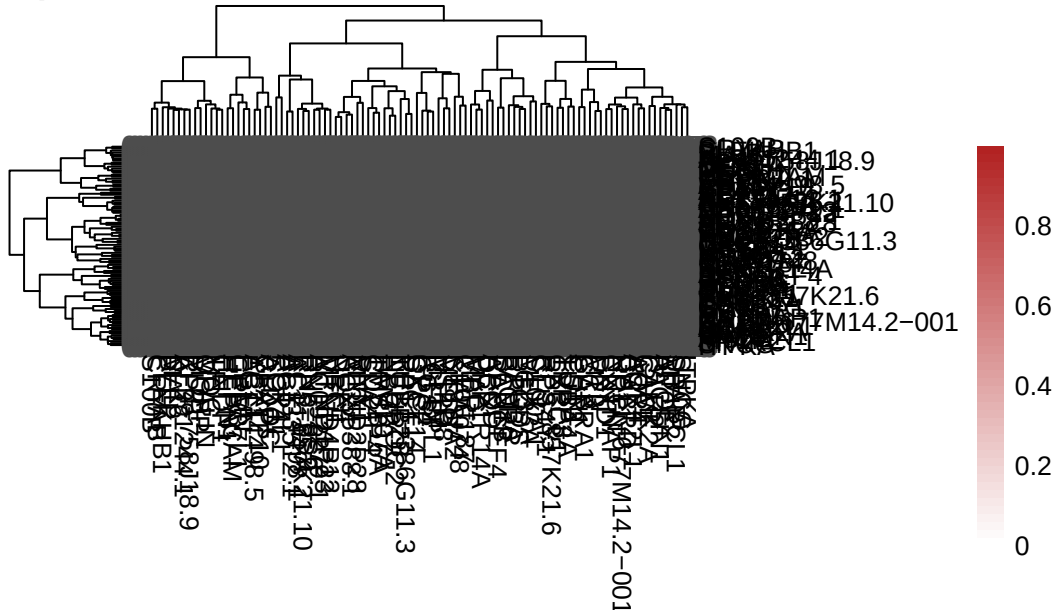

riptomic Network Architecture: Infancy Frontal Executive
0.8
0.6
0.4
0.2
0


Transcriptomic Network: Infancy Frontal Executive

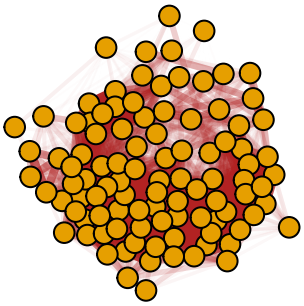

tomic Network Architecture: Adolescence Frontal Executive
0.8
0.6
0.4
0.2
0

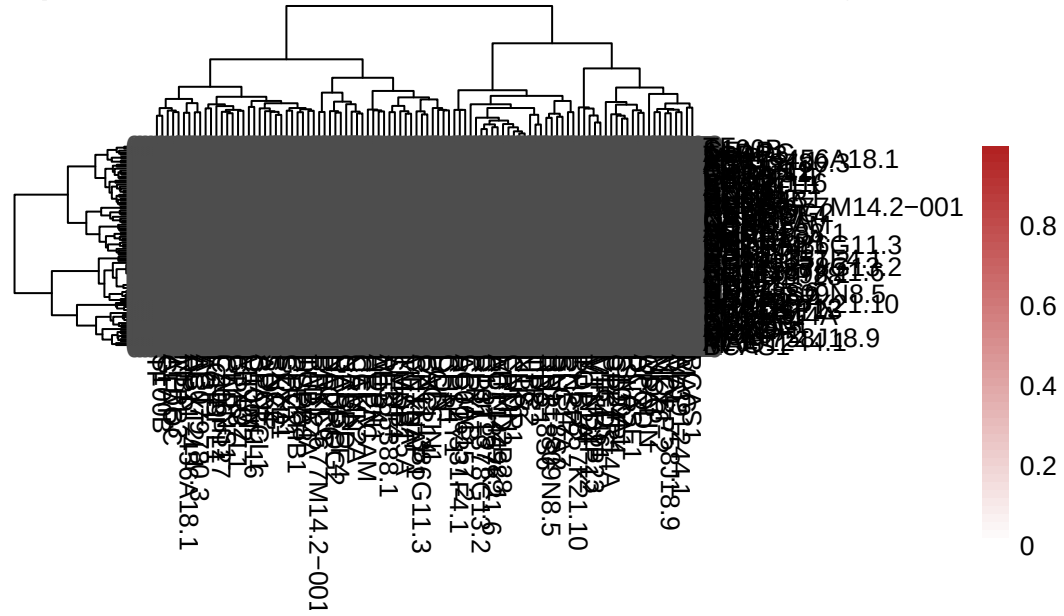

Transcriptomic Network: Adolescence Frontal Executive

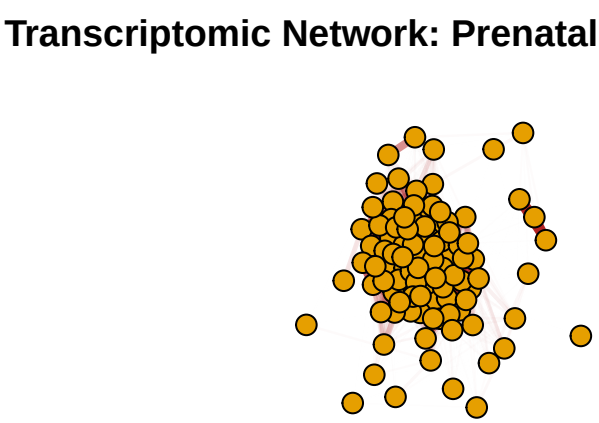
criptomic Network Architecture: Adult Frontal Executive



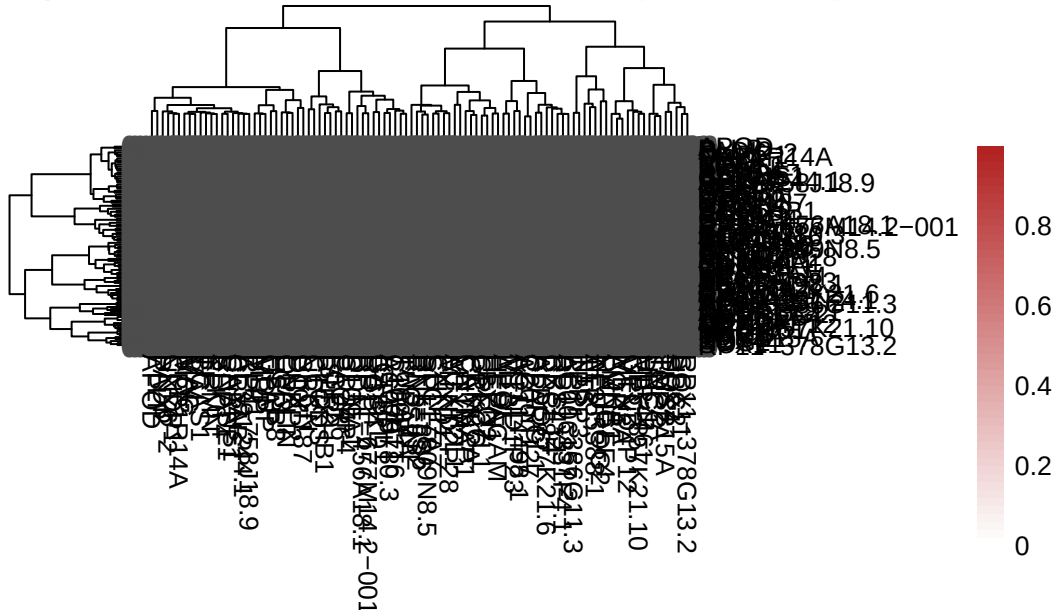
Transcriptomic Network: Adult Frontal Executive



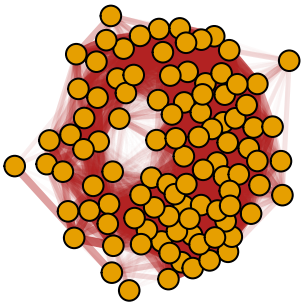

criptomic Network Architecture: Prenatal Limbic System
0.8
0.6
0.4
0.2
0

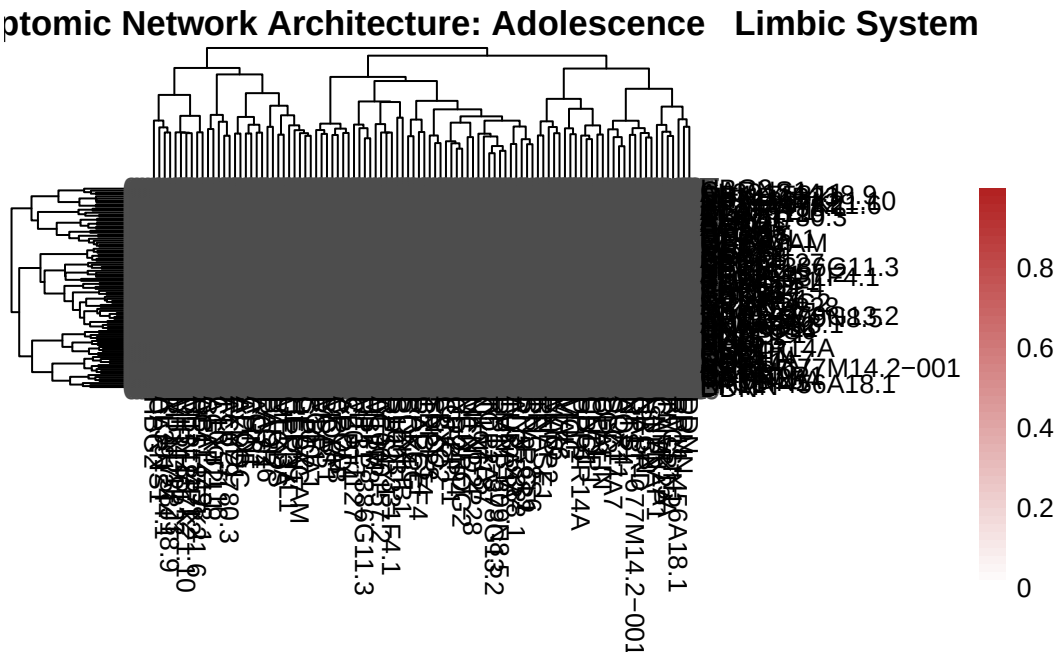

Transcriptomic Network: Prenatal Limbic System

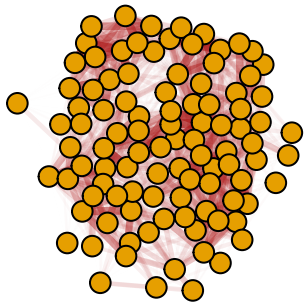
criptomic Network Architecture: Infancy Limbic System



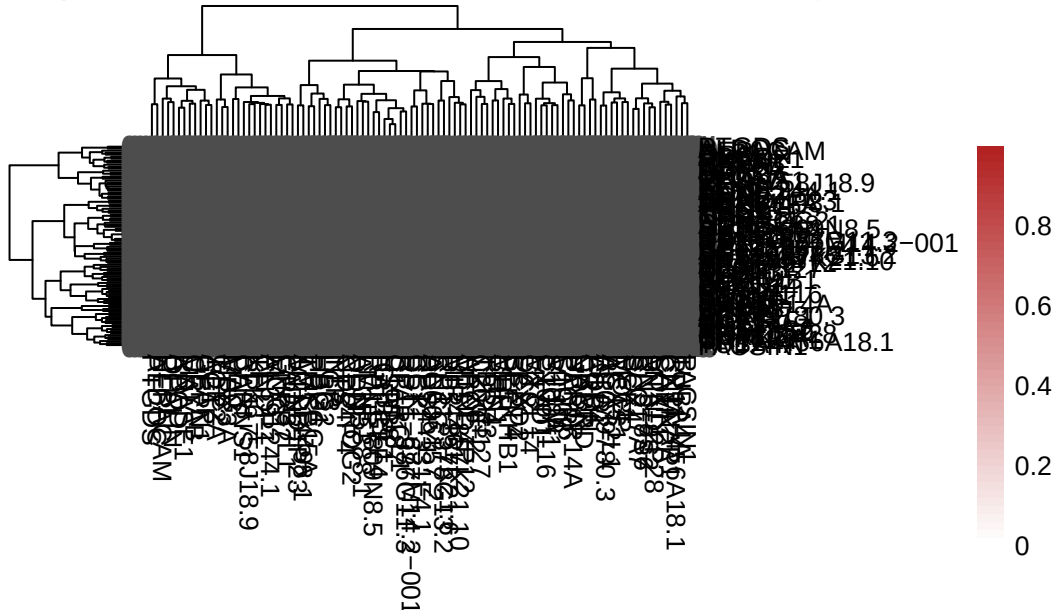
Transcriptomic Network: Infancy Limbic System




ptomic Network Architecture: Adolescence Limbic System
0.8
0.6
0.4
0.2
0


Transcriptomic Network: Adolescence Limbic System

scriptomic Network Architecture: Adult Limbic System



Transcriptomic Network: Adult Limbic System