HumanIBD

Jin Kweon

4/17/2019

#Library

```
suppressMessages(library(OmicsPLS)) #Main package
suppressMessages(library(ggplot2))
suppressMessages(library(PMA))
suppressMessages(library(reshape2))
suppressMessages(library(moments)) #calculate skewness and kurtosis
suppressMessages(library(magrittr)) #pipe like operator
suppressMessages(library(gridExtra))
suppressMessages(library(stringr))
suppressMessages(library(plotrix)) #draw.circle
suppressMessages(library(corrplot)) #coorrelation plot
suppressMessages(library(parallel))
```

#Variable pick

```
crossval_o2m_adjR2_fixed <- function(X, Y, a, ax, ay, nr_folds, nr_cores = 1,
                               stripped = TRUE, p_thresh = 3000,
                               q_thresh = p_thresh, tol = 1e-10, max_iterations = 100)
{
  tic = proc.time()
  if(any(abs(colMeans(X)) > 1e-5)){message("Data is not centered, proceeding...")}
  kcv = nr_folds
  stopifnot(ncol(X) > max(a)+max(ax) , ncol(Y) > max(a)+max(ay) , nrow(X) >= kcv)
  stopifnot(nr_cores == abs(round(nr_cores)))
  if(nr_folds==1){stop("Cross-validation with 1 fold does not make sense, use 2 folds or more")}
  cl_crossval_o2m <- NULL
  on.exit({if(!is.null(cl_crossval_o2m)) stopCluster(cl_crossval_o2m)})

  parms = data.frame(a = a)
  parms = apply(parms,1,as.list)

  if(Sys.info()[["sysname"]] == "Windows" && nr_cores > 1){
    cl_crossval_o2m <- makePSOCKcluster(nr_cores)
    clusterEvalQ(cl_crossval_o2m, library(O2PLS))
    clusterExport(cl_crossval_o2m, varlist = ls(), envir = environment())
    outp=parLapply(cl_crossval_o2m,parms,function(e){
      parms = data.frame(nx = ax)
      parms = merge(parms,data.frame(ny = ay))
      parms = apply(parms,1,as.list)
      R2grid = matrix(colMeans(suppressMessages(adjR2(Y, X, e$a, ax, ay,
                                         stripped = stripped, p_thresh = p_thresh,
                                         q_thresh = q_thresh, tol = tol, max_iterations = 
                      nrow = length(ay), byrow=TRUE)
      nxny = which(R2grid == max(R2grid), arr.ind = TRUE)[1,]
      a_mse = suppressMessages(loocv_combi(X,Y,e$a,ax[nxny[2]],ay[nxny[1]],app_err=F,func=o2m,kcv=kcv,
                          stripped = stripped, p_thresh = p_thresh,
                          q_thresh = q_thresh, tol = tol, max_iterations = max_iterations)[[1]])
      c(a_mse, e$a, ax[nxny[2]],ay[nxny[1]])
    })
  } else {
    outp=mclapply(mc.cores=nr_cores,parms,function(e){
      parms = data.frame(nx = ax)
      parms = merge(parms,data.frame(ny = ay))
      parms = apply(parms,1,as.list)
      R2grid = matrix(colMeans(suppressMessages(adjR2(Y, X, e$a, ax, ay,
                                    stripped = stripped, p_thresh = p_thresh,
                                    q_thresh = q_thresh, tol = tol, max_iterations = max_iterations)))
                      nrow = length(ay), byrow=TRUE)
      nxny = which(R2grid == max(R2grid), arr.ind = TRUE)[1,]
      a_mse = suppressMessages(loocv_combi(X,Y,e$a,ax[nxny[2]],ay[nxny[1]],app_err=F,func=o2m,kcv=kcv,
                          stripped = stripped, p_thresh = p_thresh,
                          q_thresh = q_thresh, tol = tol, max_iterations = max_iterations)[[1]])
      c(a_mse, e$a, ax[nxny[2]],ay[nxny[1]])
    })
  }
  outp2 = matrix(unlist(outp), nrow = length(a), byrow = T)
  outp2 <- as.data.frame(outp2)
  outp2 <- cbind(outp2, rep(round((proc.time() - tic)[3],2), nrow(outp2)))
  names(outp2) <- c("MSE", "n", "nx", "ny", "time")
  message("minimum is at n = ", outp2[,2][which.min(outp2[,1])], sep = ' ')
  message("Elapsed time: ", round((proc.time() - tic)[3],2), " sec")
  return(outp2)
}



    set.seed(28051968);
    data1 <- readRDS("norm.data1")
    data2 <- readRDS("norm.data2")

    table_cv <- crossval_o2m_adjR2_fixed(data1[,-1], data2[,-1], 1:10, c(0,1,3,5,7,10, 12), c(0,1,3,5,7
```

```
## Data is not centered, proceeding...

## minimum is at n = 1

## Elapsed time: 43.68 sec
```

```
    table_cv <- table_cv[order(table_cv$MSE), ]

    for(i in 1:5){ #output the best optimal 5 models based on MSE
      name <- paste0("fit", i)
      assign(name, o2m(data1[,-1], data2[,-1], table_cv$n[i], table_cv$nx[i], table_cv$ny[i]))
    }
```

```
## X has class data.frame, trying to convert with as.matrix.

## Y has class data.frame, trying to convert with as.matrix.
## Data is not centered, proceeding...

## X has class data.frame, trying to convert with as.matrix.

## Y has class data.frame, trying to convert with as.matrix.

## Data is not centered, proceeding...

## X has class data.frame, trying to convert with as.matrix.

## Y has class data.frame, trying to convert with as.matrix.

## Data is not centered, proceeding...

## X has class data.frame, trying to convert with as.matrix.

## Y has class data.frame, trying to convert with as.matrix.

## Data is not centered, proceeding...

## X has class data.frame, trying to convert with as.matrix.

## Y has class data.frame, trying to convert with as.matrix.

## Data is not centered, proceeding...
```

```
summary_variance <- data.frame(matrix(ncol = 15, nrow = 5))
x <- c("n", "nx", "ny", "X - noise", "Y - noise", "X_joint", "Y_joint", "X_orth", "Y_orth",
       "X_by_Y", "Y_by_X",
       "X_joint_by_Y_joint", "Y_joint_by_X_joint", "MSE", "time")
colnames(summary_variance) <- x

for(i in 1:5){ #output the best optimal 5 models based on MSE
  name <- paste0("fit", i)
  assign(name, o2m(data1[,-1], data2[,-1], table_cv$n[i], table_cv$nx[i], table_cv$ny[i]))
  getting <- get(name)
  vec <- c(table_cv$n[i],
           table_cv$nx[i],
           table_cv$ny[i],
           round(getting$R2X * 100, 3),
           round(getting$R2Y * 100, 3),
           round(getting$R2Xcorr * 100, 3),
           round(getting$R2Ycorr * 100, 3),
           round(getting$R2X_YO * 100, 3),
           round(getting$R2Y_XO * 100, 3),
           round(getting$R2Xhat * 100, 3),
           round(getting$R2Yhat * 100, 3),
           round(getting$R2Xhat/getting$R2Xcorr * 100, 3),
           round(getting$R2Yhat/getting$R2Ycorr * 100, 3),
           table_cv$MSE[i],
           table_cv$time[i]
           )
  summary_variance[i,] <- vec
}
```

```
## X has class data.frame, trying to convert with as.matrix.

## Y has class data.frame, trying to convert with as.matrix.

## Data is not centered, proceeding...

## X has class data.frame, trying to convert with as.matrix.
## Y has class data.frame, trying to convert with as.matrix.

## Data is not centered, proceeding...

## X has class data.frame, trying to convert with as.matrix.

## Y has class data.frame, trying to convert with as.matrix.

## Data is not centered, proceeding...

## X has class data.frame, trying to convert with as.matrix.

## Y has class data.frame, trying to convert with as.matrix.

## Data is not centered, proceeding...

## X has class data.frame, trying to convert with as.matrix.

## Y has class data.frame, trying to convert with as.matrix.

## Data is not centered, proceeding...
```

```
summary_variance
```

```
##   n nx ny X - noise Y - noise X_joint Y_joint X_orth Y_orth X_by_Y Y_by_X
## 1 1  3  3    68.092    58.131  43.410  31.059 24.682 27.072 41.273 29.530
## 2 2  7  5    77.953    71.688  48.383  49.216 29.569 22.473 42.869 37.939
## 3 3  3  3    82.824    74.515  59.933  52.963 22.890 21.552 52.034 42.648
## 4 4  5  5    88.444    81.020  61.721  58.547 26.723 22.473 52.619 46.556
## 5 8  5  5    91.921    90.620  69.666  72.149 22.256 18.471 58.945 59.848
##   X_joint_by_Y_joint Y_joint_by_X_joint      MSE  time
## 1             95.077             95.077 585596.1 43.68
## 2             88.602             77.087 596997.1 43.68
## 3             86.820             80.524 629975.4 43.68
## 4             85.252             79.518 647996.8 43.68
## 5             84.611             82.951 656877.2 43.68
```

Comment:

This is one of the methods to pick the number of components for joint and orthogonals. Here, I go for adjusted R^2 for getting it. (One thing you might already spot it, is that it is cross validation working - meaning that the answer can be slightly changed everytime you run this option)

Basically, you either pick the numbers using the summary_variance table. You can use MSE (the lower the better), X by Y (X that can can explained by Y) and Y by X, or X joint by Y joint (joint X that can explained by Y) and Y joint by X joint.

There are way more beautiful inside logics behind it, but since this is really not important for now, I will skip it, but let me know if you are interested.

#Main - table1

```
fit <- o2m(data1[,-1], data2[,-1], 4, 3, 2) #user can change it!
```

```
## X has class data.frame, trying to convert with as.matrix.
## Y has class data.frame, trying to convert with as.matrix.
```

```
## Data is not centered, proceeding...
```

```
summary_score_noise <- data.frame(matrix(ncol = 6, nrow = 2))
x <- c("X_joint_score", "X_ortho_score",
       "Y_joint_score", "Y_ortho_score",
       "X_noise", "Y_noise"
       )
colnames(summary_score_noise) <- x
rownames(summary_score_noise) <- c("Absolute", "Relative")

summary_score_noise[1,] <- c(sum(summary(fit)[12]$flags$varXjoint),
                             sum(summary(fit)[12]$flags$varXorth),
                             sum(summary(fit)[12]$flags$varYjoint),
                             sum(summary(fit)[12]$flags$varYorth),
                             (summary(fit)[12]$flags$ssqX - sum(summary(fit)[12]$flags$varXjoint) -
                             (summary(fit)[12]$flags$ssqY - sum(summary(fit)[12]$flags$varYjoint) -
                             )

summary_score_noise[2,] <- c(round(fit$R2Xcorr * 100, 3),
                             (round(fit$R2X * 100, 3) - round(fit$R2Xcorr * 100, 3)),
                              round(fit$R2Ycorr * 100, 3),
                              (round(fit$R2Y * 100, 3) - round(fit$R2Ycorr * 100, 3)),
                             (100 - round(fit$R2X * 100, 3)),
                             (100 - round(fit$R2Y * 100, 3))
                             )



summary_score_noise
```

```
##          X_joint_score X_ortho_score Y_joint_score Y_ortho_score      X_noise
## Absolute  7.735133e+13  2.936881e+13  7.156425e+14  1.742906e+14 1.472679e+13
## Relative  6.369100e+01  2.259400e+01  6.044300e+01  1.399700e+01 1.371500e+01
##               Y_noise
## Absolute 2.940669e+14
## Relative 2.556000e+01
```

Comment:

Say that you go for the first one in the table: 'fit.'

This is basically telling you how much variations joint, orthogonal, and noise is taking.

It tells you both absolute and relative values. The relative Joint + orthogonal + noise should be 100.

#Main - table2

```
model_summary <- data.frame(matrix(ncol = 3, nrow = summary(fit)[12]$flags$n + 1))
x <- c("Component", "Omics1", "Omics2")
colnames(model_summary) <- x
```

```
    y <- c(1:summary(fit)[12]$flags$n, "sum")
    model_summary[,1] <- y

    first <- c()
    second <- c()
    for (i in 1:summary(fit)[12]$flags$n){
      first[i] <- summary(fit)[12]$flags$varXjoint[i]/summary(fit)[12]$flags$ssqX
      second[i] <- summary(fit)[12]$flags$varYjoint[i]/summary(fit)[12]$flags$ssqY
    }

    first <- c(first, sum(first))
    second <- c(second, sum(second))

    model_summary[,2] <- round(first, 4) * 100
    model_summary[,3] <- round(second, 4) * 100

    model_summary
```

```
##   Component Omics1 Omics2
## 1         1  43.65  31.95
## 2         2   6.82  17.16
## 3         3   9.15   5.07
## 4         4   4.07   6.25
## 5       sum  63.69  60.44
```

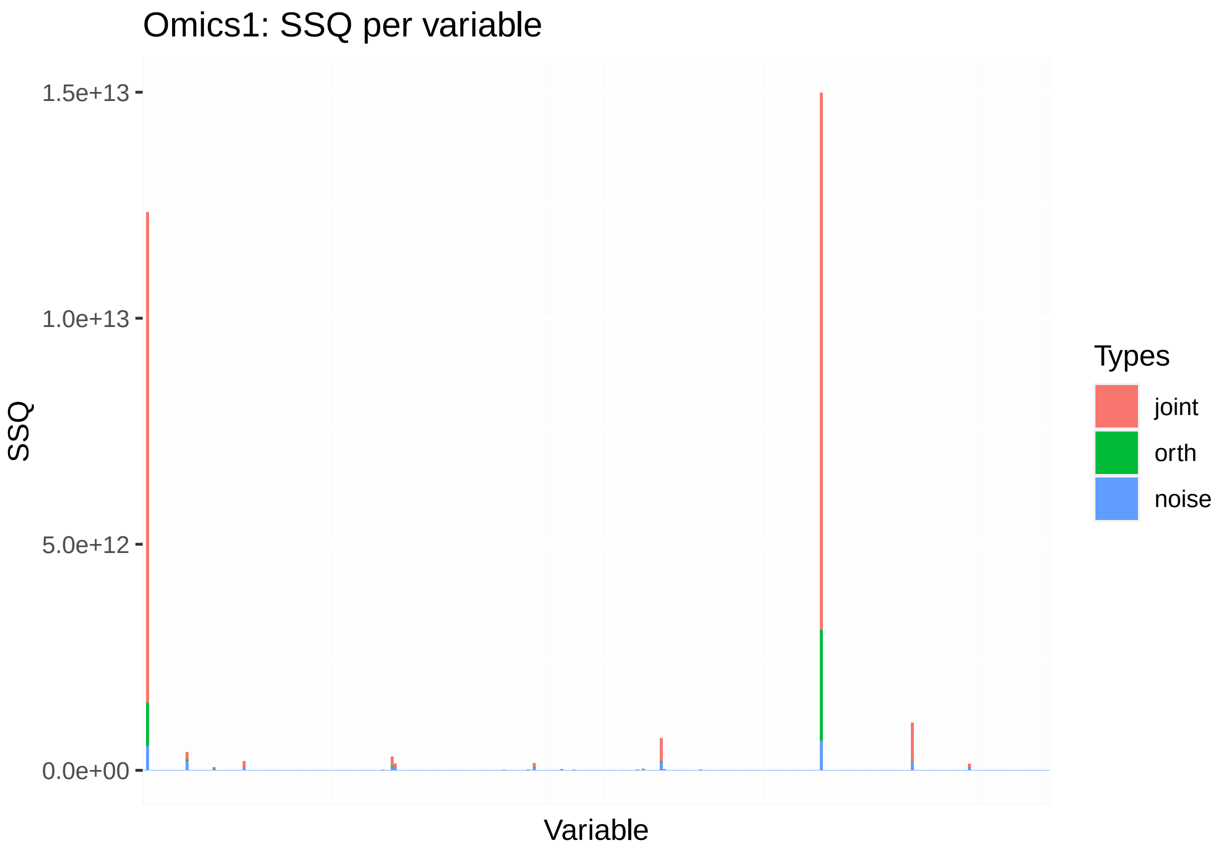
Comment:

This table tells you how much variations each joint component takes for each omics. The last row has the values of variations for entire joint components.

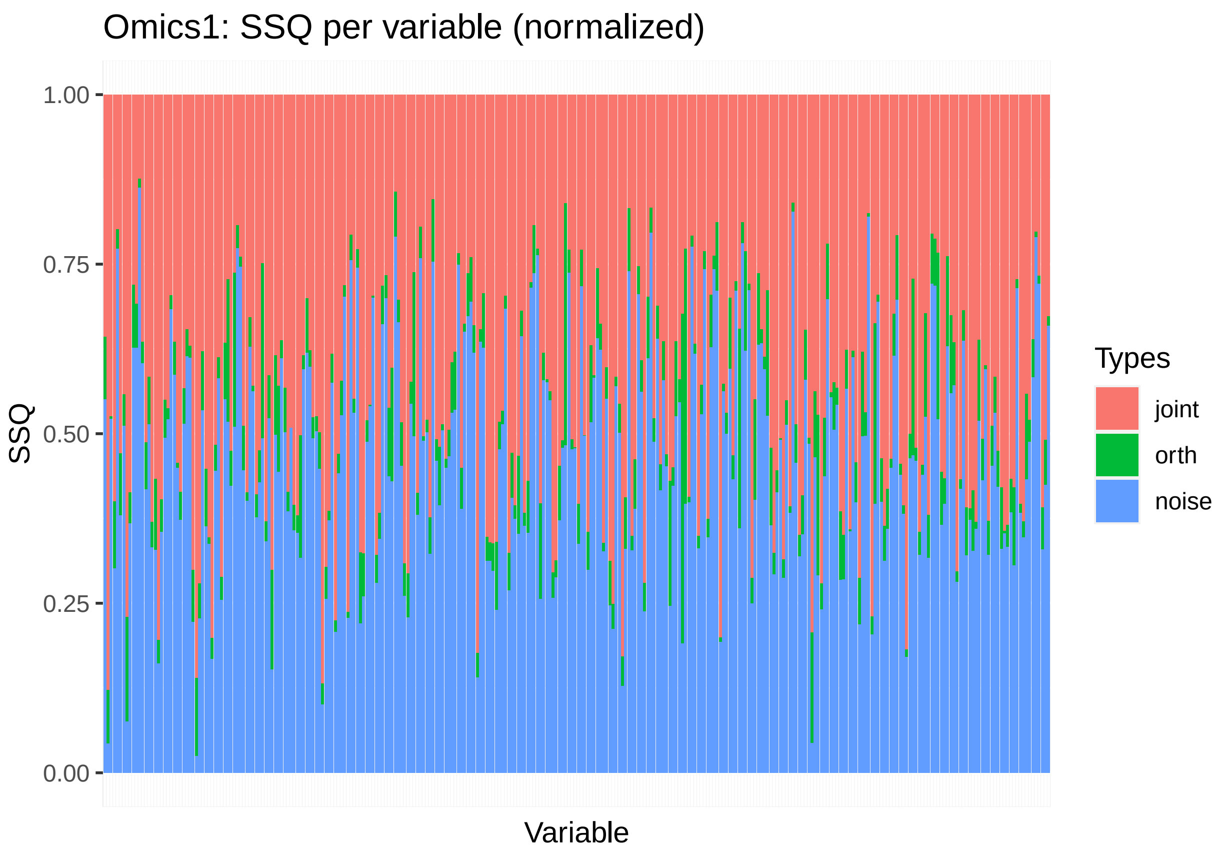
#Main - plot1

```
#first omics
if(fit$flags$nx == 0){
      xsq <- rbind(apply((fit$Tt %*% t(fit$W.))^2, 2, sum),
                   apply((fit$E)^2, 2, sum));
      xsq <- t(xsq);
      xsq_new <- xsq;
      if(dim(xsq)[1] > 300){
        tt <- sample(nrow(xsq), 300);
        xsq <- xsq[tt,,drop=FALSE]
      }
      colnames(xsq) <- c("joint", "noise");
      sum_xsq <- apply(xsq_new, 1, sum);
      xsq <- melt(xsq);
      xsq$Var2 <- factor(xsq$Var2, labels = c("joint", "noise"));
```

```r
    }else{
      xsq <- rbind(apply((fit$Tt %*% t(fit$W.))^2, 2, sum),
                   apply((fit$T_Yosc %*% t(fit$P_Yosc.))^2, 2, sum),
                   apply((fit$E)^2, 2, sum));

      xsq <- t(xsq);
      xsq_new <- xsq;
      if(dim(xsq)[1] > 300){
        tt <- sample(nrow(xsq), 300);
        xsq <- xsq[tt,,drop=FALSE]
      }
      colnames(xsq) <- c("joint", "orth", "noise");
      sum_xsq <- apply(xsq_new, 1, sum);
      xsq <- melt(xsq);
      xsq$Var2 <- factor(xsq$Var2, labels = c("joint", "orth", "noise"));

    }


ggplot(xsq, aes(x = Var1, y = value, fill = Var2)) + geom_bar(stat = "identity") +
      labs(title = "Omics1: SSQ per variable", y = "SSQ", x = "Variable") +
       theme(axis.text.x=element_blank(),
            axis.ticks.x=element_blank()) +
      guides(fill=guide_legend(title="Types"))
```



```r
ggplot(xsq, aes(x = Var1, y = value, fill = Var2)) + geom_bar(stat = "identity", position = "fill") +
  labs(title = "Omics1: SSQ per variable (normalized)", y = "SSQ", x = "Variable") +
   theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  guides(fill=guide_legend(title="Types"))
```



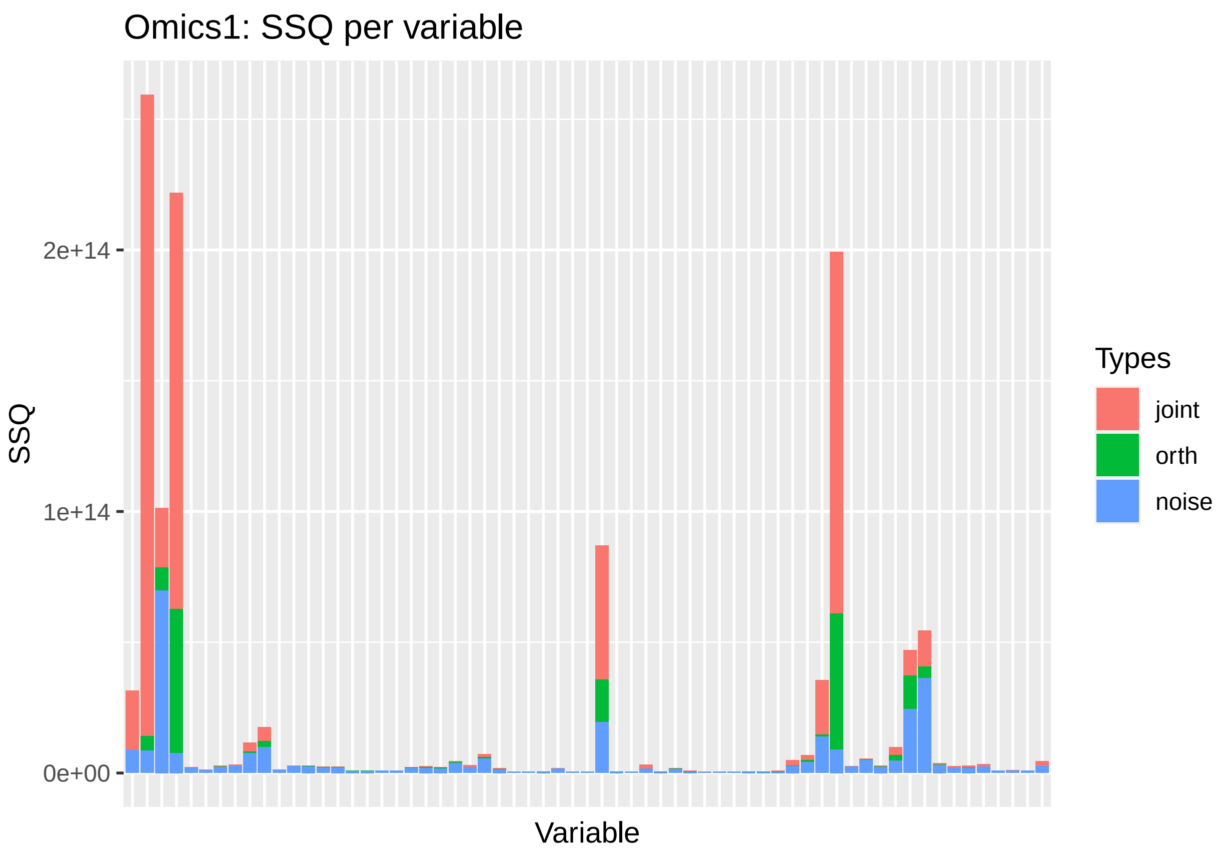
```
melt(sort(sum_xsq, decreasing = T)[1:10]) #total SSQ
```

```
##                  value
## X194.04872 2.776599e+13
## X279.23323 1.499243e+13
## X465.30512 1.237006e+13
## X281.249   1.234581e+13
## X299.25954 1.143179e+13
## X471.24157 7.700326e+12
## X471.24258 3.468964e+12
## X783.58014 3.399999e+12
## X391.28574 2.942349e+12
## X255.23318 2.870867e+12
```

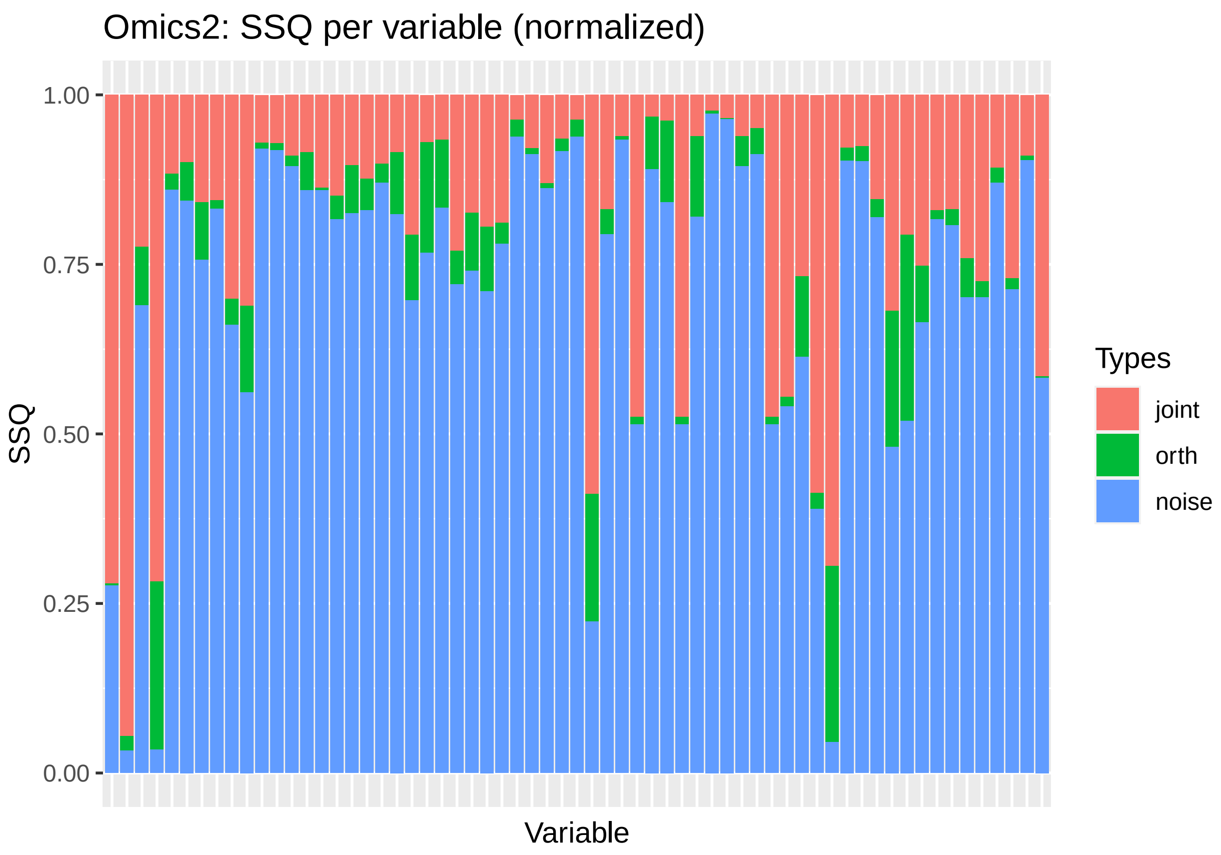
```
melt(sort(xsq_new[,1], decreasing = T)[1:10]) # Joint only
```

```
##                  value
## X194.04872 1.392060e+13
## X279.23323 1.189543e+13
## X465.30512 1.136460e+13
## X281.249   1.084331e+13
## X299.25954 7.872479e+12
## X255.23318 2.471135e+12
## X391.28574 2.392577e+12
## X783.58014 2.211791e+12
## X471.24157 1.889151e+12
## X391.28592 1.094413e+12
```

```r
#Second omics


if(fit$flags$ny == 0){
      ysq <- rbind(apply((fit$U %*% t(fit$C.))^2, 2, sum),
             apply((fit$Ff)^2, 2, sum))
      ysq <- t(ysq)
      ysq_new <- ysq;
      if(dim(ysq)[1] > 500){
        tt <- sample(nrow(ysq), 500);
        ysq <- ysq[tt,,drop=FALSE]
      }
      colnames(ysq) <- c("joint", "noise");
      sum_ysq <- apply(ysq_new, 1, sum)
      ysq <- melt(ysq)
      ysq$Var2 <- factor(ysq$Var2, labels = c("joint", "noise"))


    }else{
      ysq <- rbind(apply((fit$U %*% t(fit$C.))^2, 2, sum),
             apply((fit$U_Xosc %*% t(fit$P_Xosc.))^2, 2, sum),
             apply((fit$Ff)^2, 2, sum))

      ysq <- t(ysq)
      ysq_new <- ysq;
      if(dim(ysq)[1] > 500){
        tt <- sample(nrow(ysq), 500);
        ysq <- ysq[tt,,drop=FALSE]
      }
      colnames(ysq) <- c("joint", "orth", "noise")
      sum_ysq <- apply(ysq_new, 1, sum)
      ysq <- melt(ysq)
      ysq$Var2 <- factor(ysq$Var2, labels = c("joint", "orth", "noise"))

    }



ggplot(ysq, aes(x = Var1, y = value, fill = Var2)) + geom_bar(stat = "identity") +
  labs(title = "Omics1: SSQ per variable", y = "SSQ", x = "Variable") +
   theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  guides(fill=guide_legend(title="Types"))
```



```r
ggplot(ysq, aes(x = Var1, y = value, fill = Var2)) + geom_bar(stat = "identity", position = "fill") +
  labs(title = "Omics2: SSQ per variable (normalized)", y = "SSQ", x = "Variable") +
   theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  guides(fill=guide_legend(title="Types"))
```



```
melt(sort(sum_ysq, decreasing = T)[1:10]) #total SSQ
```

```
##                                      value
## X__Escherichia_Shigella         2.593143e+14
## X__Faecalibacterium_1           2.219366e+14
## X__Bacteroides                  1.993109e+14
## X__Haemophilus                  1.013290e+14
## X__Bacteroides_4                8.693808e+13
## X___Eubacterium_rectale_group   5.449689e+13
## X___Ruminococcus_gnavus_group   4.697864e+13
## X__Bacteroides_1                3.557028e+13
## X__Fusobacterium                3.151879e+13
## X___Eubacterium_rectale_group_1 1.768035e+13
```

```
melt(sort(ysq_new[,1], decreasing = T)[1:10]) # Joint only
```

```
##                                      value
## X__Escherichia_Shigella         2.452767e+14
## X__Faecalibacterium_1           1.592035e+14
## X__Bacteroides                  1.384380e+14
## X__Bacteroides_4                5.113613e+13
## X__Fusobacterium                2.270383e+13
## X__Haemophilus                  2.268791e+13
## X__Bacteroides_1                2.088487e+13
## X___Eubacterium_rectale_group   1.373424e+13
## X___Ruminococcus_gnavus_group   9.696970e+12
## X___Eubacterium_rectale_group_1 5.500321e+12
```

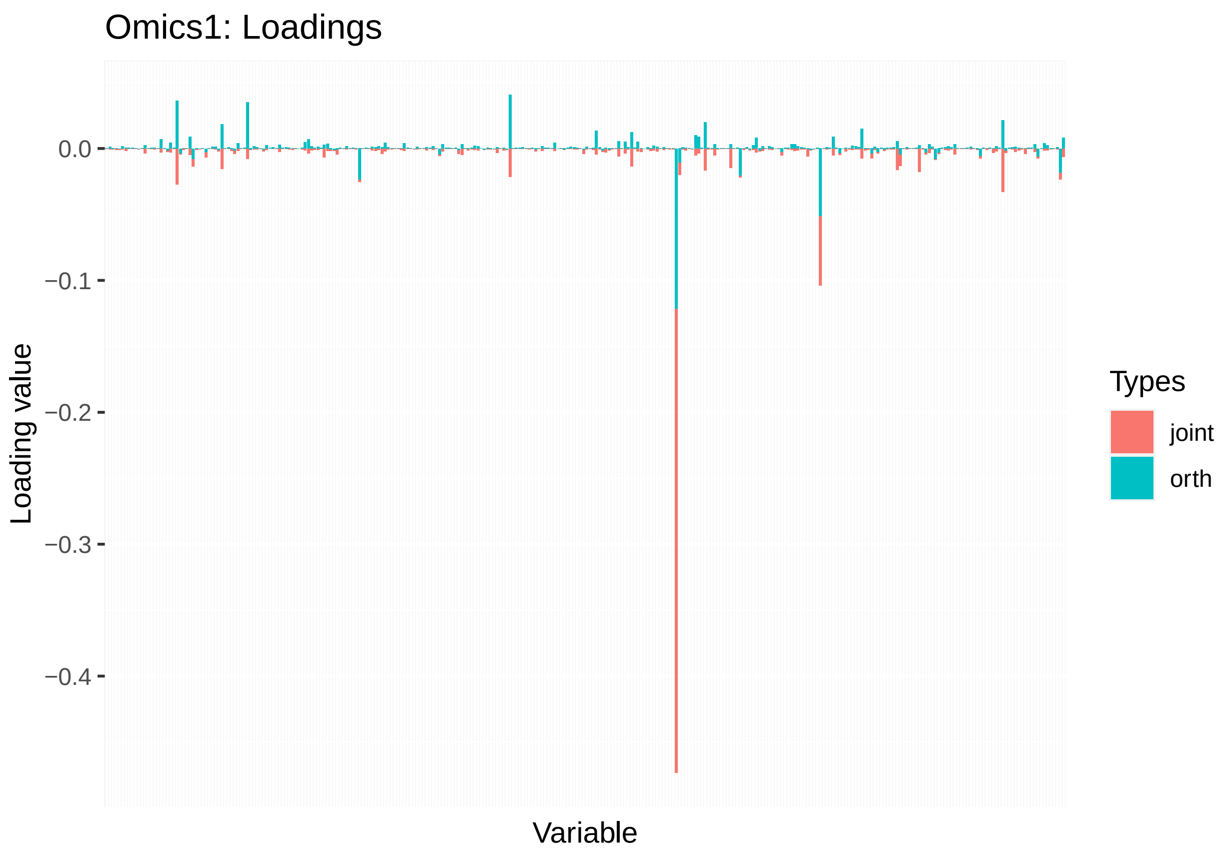
Comment:

It shows Sum of squares (SSQ) for each variable (the default is randomly chosen 500 variables if there are more than 500) can gives some insights how each variable is important for different types of variations (joint, ortho, and noise).

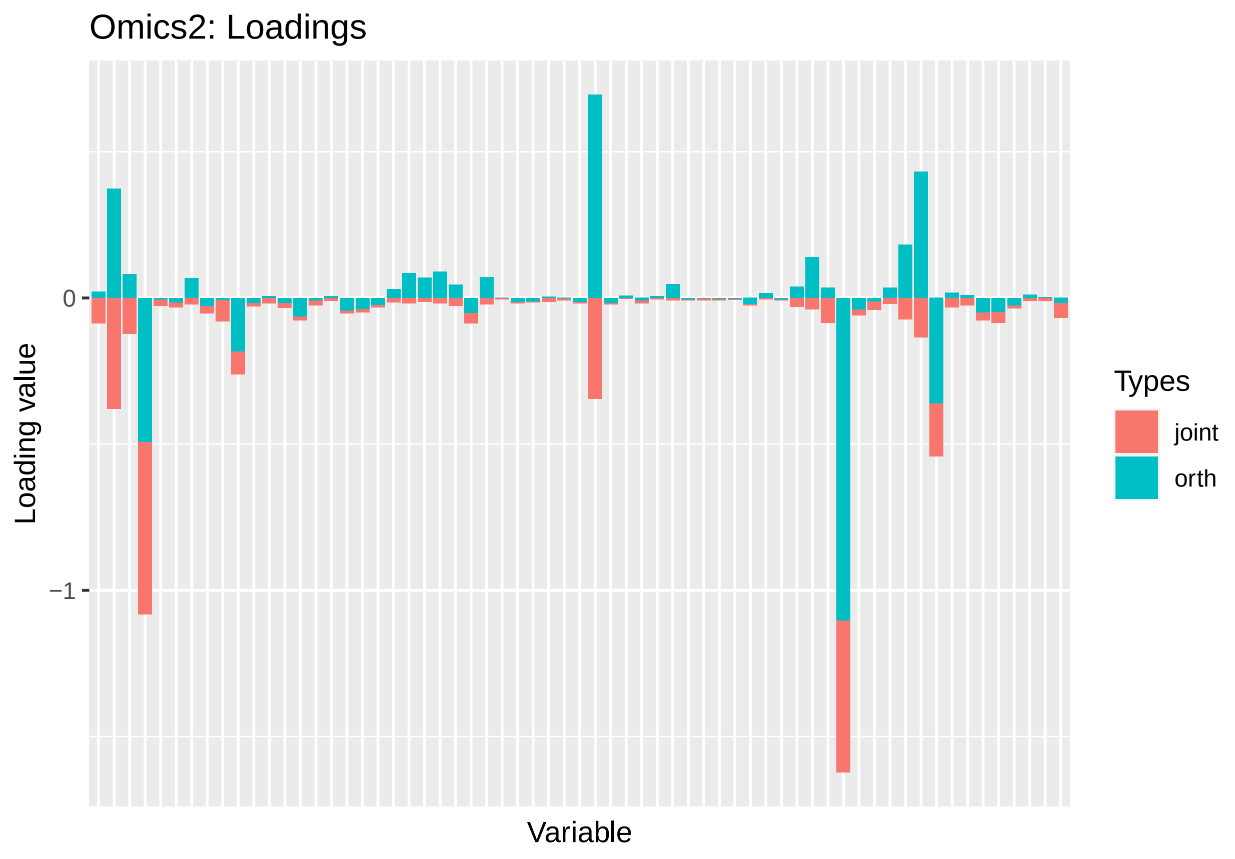
When it is not normalized, one outlier can lead us to bias understanndings. So, I also made a normalized plot for SSQ. In here, everything is normalized to 1. This is normalized plot of SSQ. It shows how much joint, orthogonal, and noise each variable takes when each variable is normalized to 1. (In here, you cannot directly compare one variable to the other)

#Main - plot2

```r
options(scipen = 999)

    load1x <- rbind(fit$W.[,1], fit$P_Yosc.[,1])
    rownames(load1x) <- c("joint", "orth")
    load1x <- t(load1x)

    abssum_load1x <- apply(abs(load1x), 1, sum)
load1xall <- melt(sort(abssum_load1x, decreasing = T)[1:10]) #sum of absolute joint and orth
colnames(load1xall) <- "Absolute value"
load1xjoint <- melt(sort(abs(load1x[,1]), decreasing = T)[1:10]) #sum of joint
colnames(load1xjoint) <- "Absolute value"
    if(dim(load1x)[1] > 300){
        tt <- sample(nrow(load1x), 300)
        load1x <- load1x[tt,,drop=FALSE]
    }

    load1x <- melt(load1x)
    load1x$Var2 <- factor(load1x$Var2, labels = c("joint", "orth"))

ggplot(load1x, aes(x = Var1, y = value, fill = Var2)) + geom_bar(stat = "identity") +
      labs(title = "Omics1: Loadings", y = "Loading value", x = "Variable") +
       theme(axis.text.x=element_blank(),
            axis.ticks.x=element_blank()) +
      guides(fill=guide_legend(title="Types"))
```



```
load1y <- rbind(fit$C.[,1], fit$P_Xosc.[,1])
rownames(load1y) <- c("joint", "orth")
load1y <- t(load1y)

abssum_load1y <- apply(abs(load1y), 1, sum)
load1yall <-melt(sort(abssum_load1y, decreasing = T)[1:10])
colnames(load1yall) <- "Absolute value"
load1yjoint <-melt(sort(abs(load1y[,1]), decreasing = T)[1:10])
colnames(load1yjoint) <- "Absolute value"

    if(dim(load1y)[1] > 300){
        tt <- sample(nrow(load1y), 300)
        load1y <- load1y[tt,,drop=FALSE]
    }

load1y <- melt(load1y)
load1y$Var2 <- factor(load1y$Var2, labels = c("joint", "orth"))

ggplot(load1y, aes(x = Var1, y = value, fill = Var2)) + geom_bar(stat = "identity") +
  labs(title = "Omics2: Loadings", y = "Loading value", x = "Variable") +
   theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  guides(fill=guide_legend(title="Types"))
```



```
load1xall
```

```
##              Absolute value
## X194.04872      3.2477605
## X299.25954      0.8859363
## X281.249        0.7430862
## X279.23323      0.7241148
## X150.05482      0.6876794
## X783.58014      0.6775770
## X391.28574      0.5105432
## X465.30512      0.4734592
## X255.23318      0.4185182
## X471.24157      0.2685489
```

```
load1xjoint
```

```
##              Absolute value
## X279.23323      0.4675062
## X281.249        0.4421101
## X299.25954      0.3727246
## X465.30512      0.3522099
## X391.28574      0.2062574
## X783.58014      0.2030530
## X255.23318      0.2023889
## X194.04872      0.1871103
## X471.24157      0.1682036
## X391.28592      0.1333435
```

```
load1yall
```

```
##                                    Absolute value
## X__Bacteroides                          1.6232009
## X__Faecalibacterium_1                   1.0826416
## X__Bacteroides_4                        1.0389841
## X__Escherichia_Shigella                 0.7536518
## X___Ruminococcus_gnavus_group           0.5666996
## X___Eubacterium_rectale_group           0.5421912
## X___Eubacterium_rectale_group_1         0.2619869
## X__Lachnoclostridium                    0.2555881
## X__Haemophilus                          0.2038476
## X__Lachnospiraceae_NK4A136_group_1      0.1773050
```

```
load1yjoint
```

```
##                                 Absolute value
## X__Faecalibacterium_1               0.59138532
## X__Bacteroides                      0.52093878
## X__Escherichia_Shigella             0.37917874
## X__Bacteroides_4                    0.34418013
## X___Eubacterium_rectale_group       0.18114598
## X___Ruminococcus_gnavus_group       0.13437316
## X__Haemophilus                      0.12259870
## X__Fusobacterium                    0.08588521
## X__Bacteroides_1                    0.08570693
## X___Eubacterium_rectale_group_1     0.07969726
```

Comment:

Now, the next logical step we should do is look at joint loading plots for each dataset. (for now, be aware that I am not comparing the two different datasets. I am just looking at the loading plots from one dataset - doing it dataset by dataset. Doing this makes it more sense.)

Based on the heights and directions of variables, the user can tell the significance of each variable. It is helpful to see how the signs/directions are different amongst variables, since this tells you that the variables with different signs are associated with the score in opposite directions.

Remember the loadings stand for the importance to each variable to the corresponding subspace/score.

Also, I gave the tables from top n variables either from 'both sum of absolute values of joint and orthogonal' or 'sum of absolute joint only' variables.

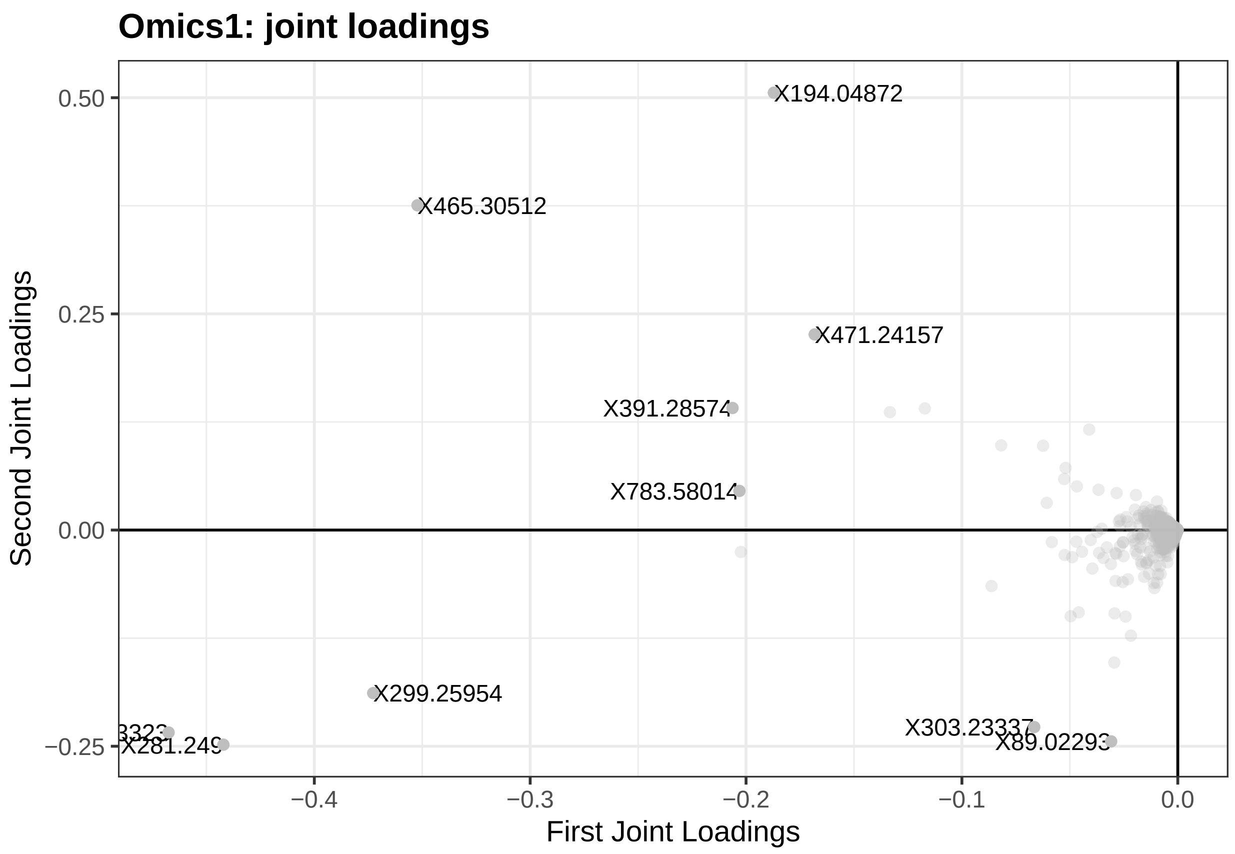
Actually, these diagrams can help the users A LOT: if we know the groups and clusters, we might be able to get ideas which cluster/group is having high/low loadings. Furthermore, outlier can be spotted as well.

Here, as an example, I only merged joint 1 and orthogonal 1 loadings. But, on the website, I am adding some features for users can choose the combinations of joint and orthogonal (i.e. joint 2 with orthogonal 3)

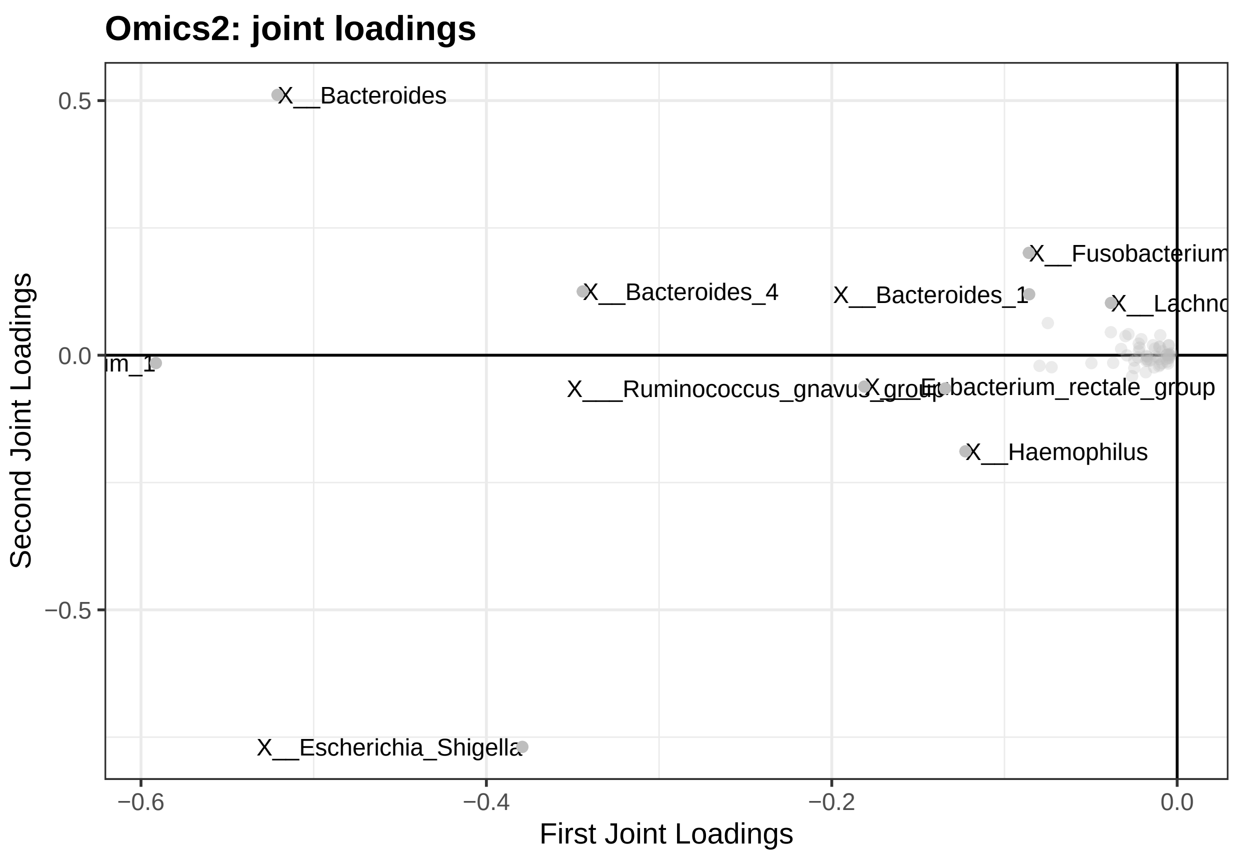
#Main - plot3

```
xjoints <- sqrt(loadings(fit, "Xjoint", 1:2) %>% raise_to_power(2) %>% rowSums)

tablex <- melt(sort(xjoints,decreasing=T)[1:10])
colnames(tablex) <- "Euclidean_dist"
tablex <- cbind(tablex, loadings(fit, "Xjoint", 1:2)[order(xjoints,decreasing=T)[1:10],])
colnames(tablex)[c(2,3)] <- c("Loading_x-axis", "Loading_y-axis")

xjoints[-(order(xjoints,decreasing=T)[1:10])] = 0
```

```r
xjoints <- sign(xjoints)


plot(fit, loading_name="Xj", i=1, j=2, label = "c", use_ggplot2 = TRUE,
     alpha = xjoints,
     aes(label = c(stringr::str_sub(names(xjoints), start = 0))),
     hjust = rep(c(0, 1), length.out = length(xjoints)), size = 3) +
  theme_bw() +
  geom_point(alpha = 0.3+0.7*xjoints, col = 'grey') +
  labs(title = "Omics1: joint loadings",
       x = "First Joint Loadings", y = "Second Joint Loadings") +
  theme(plot.title = element_text(face='bold'))
```



```r
yjoints <- sqrt(loadings(fit, "Yjoint", 1:2) %>% raise_to_power(2) %>% rowSums)

tabley <- melt(sort(yjoints,decreasing=T)[1:10])
colnames(tabley) <- "Euclidean_dist"
tabley <- cbind(tabley, loadings(fit, "Yjoint", 1:2)[order(yjoints,decreasing=T)[1:10],])
colnames(tabley)[c(2,3)] <- c("Loading_x-axis", "Loading_y-axis")

yjoints[-(order(yjoints,decreasing=T)[1:10])] = 0
yjoints <- sign(yjoints)


plot(fit, loading_name="Yj", i=1, j=2, label = "c", use_ggplot2 = TRUE,
     alpha = yjoints,
     aes(label = c(stringr::str_sub(names(yjoints), start = 0))),
    hjust = rep(c(0, 1), length.out = length(yjoints)), size = 3) +
  theme_bw() +
  geom_point(alpha = 0.3+0.7*yjoints, col = 'grey') +
  labs(title = "Omics2: joint loadings",
       x = "First Joint Loadings", y = "Second Joint Loadings") +
  theme(plot.title = element_text(face='bold'))
```



```r
tablex
```

```
##             Euclidean_dist Loading_x-axis Loading_y-axis
## X194.04872       0.5392800    -0.18711026     0.50577927
## X279.23323       0.5228381    -0.46750616    -0.23408911
## X465.30512       0.5148846    -0.35220991     0.37557205
## X281.249         0.5070127    -0.44211008    -0.24819457
## X299.25954       0.4176971    -0.37272456    -0.18853976
## X471.24157       0.2819396    -0.16820358     0.22626863
## X391.28574       0.2499398    -0.20625737     0.14116594
## X89.02293        0.2463159    -0.03086236    -0.24437484
## X303.23337       0.2373136    -0.06649183    -0.22780816
## X783.58014       0.2080391    -0.20305301     0.04527417
```

```r
tabley
```

```
##                                Euclidean_dist Loading_x-axis Loading_y-axis
## X__Escherichia_Shigella             0.8575088    -0.37917874    -0.76911954
## X__Bacteroides                      0.7298569    -0.52093878     0.51118871
## X__Faecalibacterium_1               0.5915853    -0.59138532    -0.01538208
## X__Bacteroides_4                    0.3662704    -0.34418013     0.12527578
## X__Haemophilus                        0.2250248    -0.12259870    -0.18869484
## X__Fusobacterium                       0.2185931    -0.08588521     0.20101407
## X___Eubacterium_rectale_group          0.1913147    -0.18114598    -0.06154240
## X___Ruminococcus_gnavus_group          0.1494044    -0.13437316    -0.06531101
## X__Bacteroides_1                       0.1472266    -0.08570693     0.11970793
## X__Lachnospiraceae_NK4A136_group_1     0.1096163    -0.03833736     0.10269358
```

Comment:

Now, the next logical step should be the little bit more complicated on: the two dimensions. So, I am comparing joint loading v.s. joint loading from the same dataset. (you cannot compare joint loadings from the two differennt datasets as the features are different)

As you might already realize, this will not work if the user end up choosing only one joint component. In most of the cases, they will choose more than one.

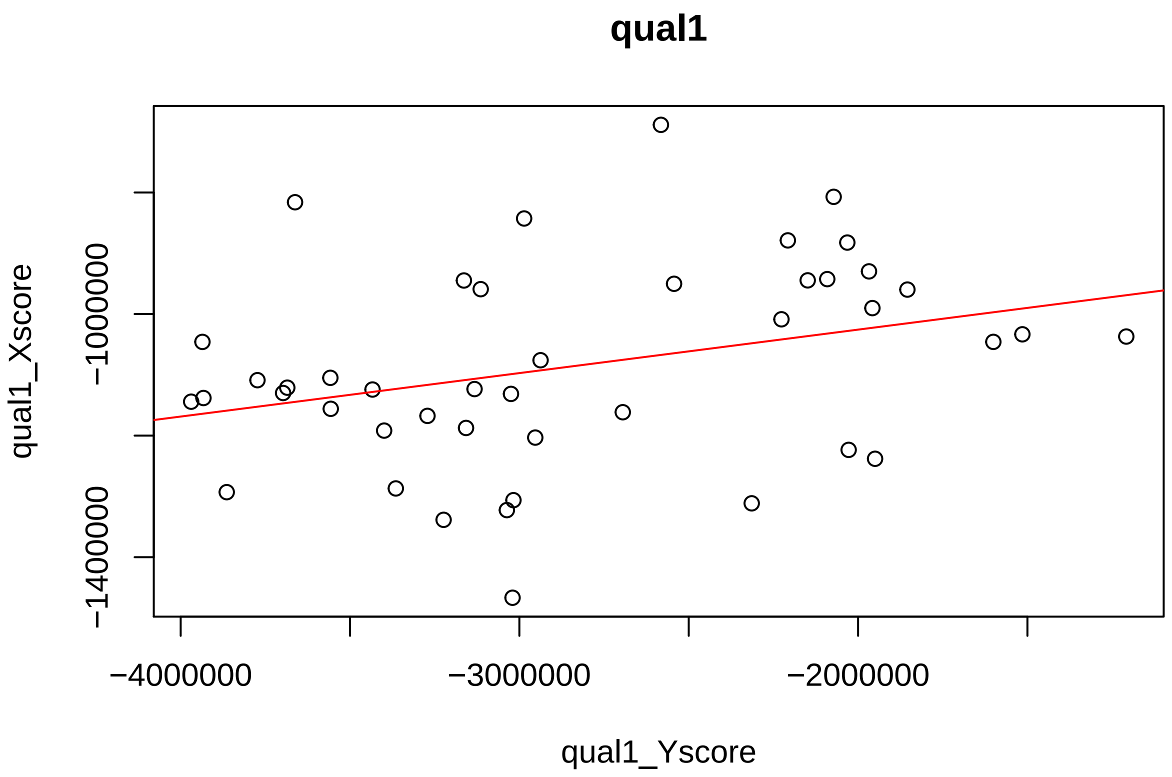
The plots indicate relative importance of each variable to corresponding component in omics dataset.

I also displayed it with the top 10 highest euclidean distance from the origin, so, people can get better ideas.

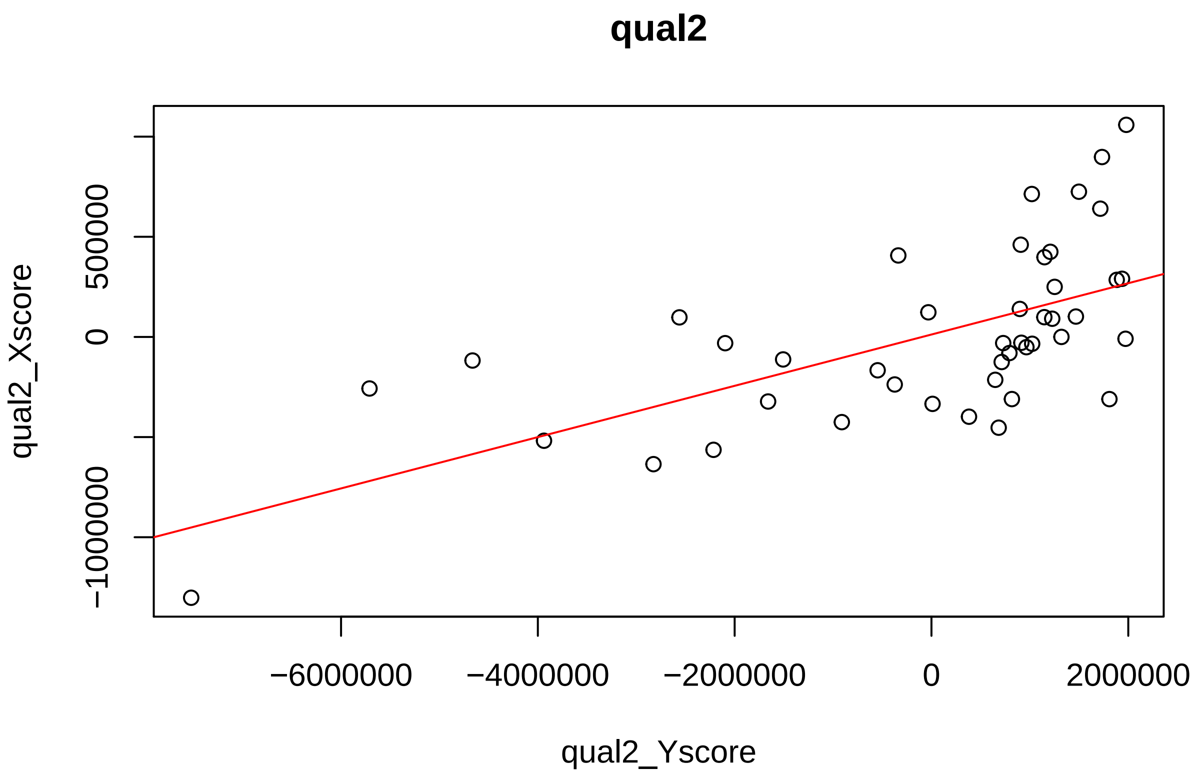
#Main - plot4

```
#quality of predictions of X with Y
for(i in 1:dim(fit$Tt)[2]){
  name <- paste0("qual", i)
  assign(name, lm(fit$Tt[,i] ~ fit$U[,i]))
  getting <- get(name)
  print(summary(getting)[4]) #coefficients
  print(summary(getting)[9]) #adj R^2

  plot(fit$U[,i], fit$Tt[,i], main = name, xlab = paste0(name, "_Yscore"), ylab = paste0(name, "_Xscore
}
```

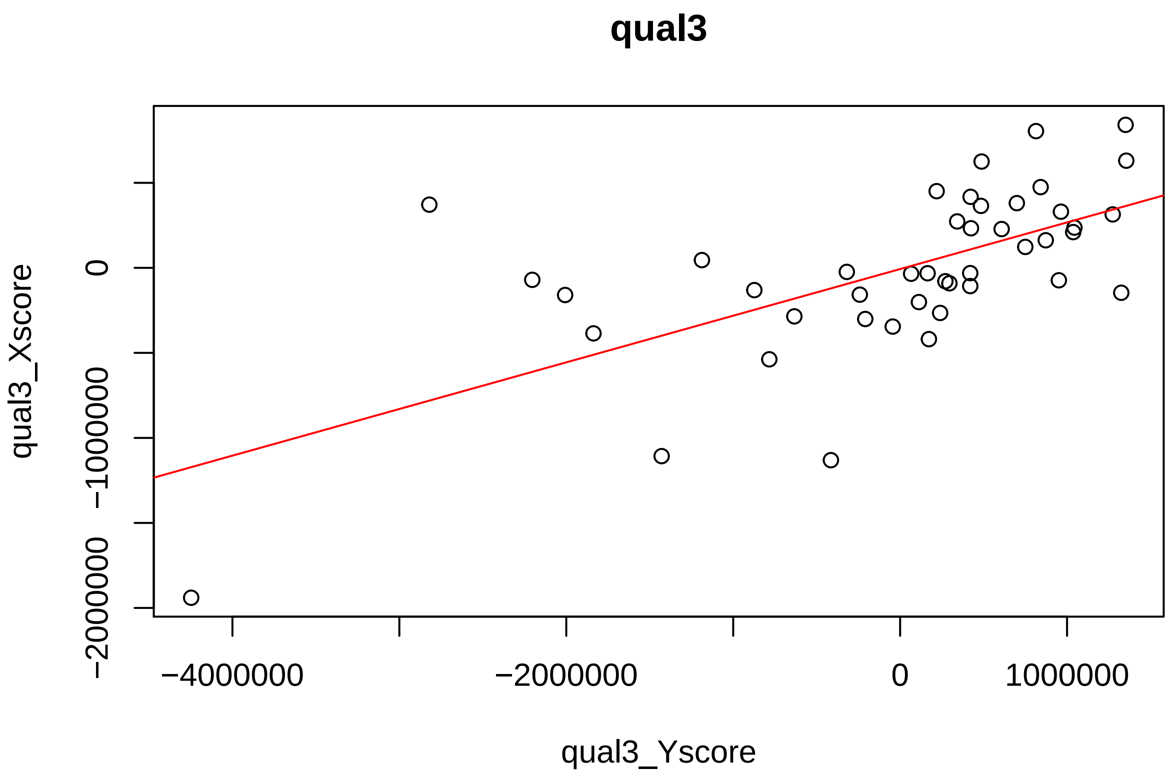
```
## $coefficients
##                     Estimate     Std. Error   t value             Pr(>|t|)
## (Intercept) -882620.42165266 94436.70523360 -9.346159 0.000000000008117929
## fit$U[, i]        0.07150593     0.03220549  2.220303 0.031849077424513039
##
## $adj.r.squared
## [1] 0.08373674
```



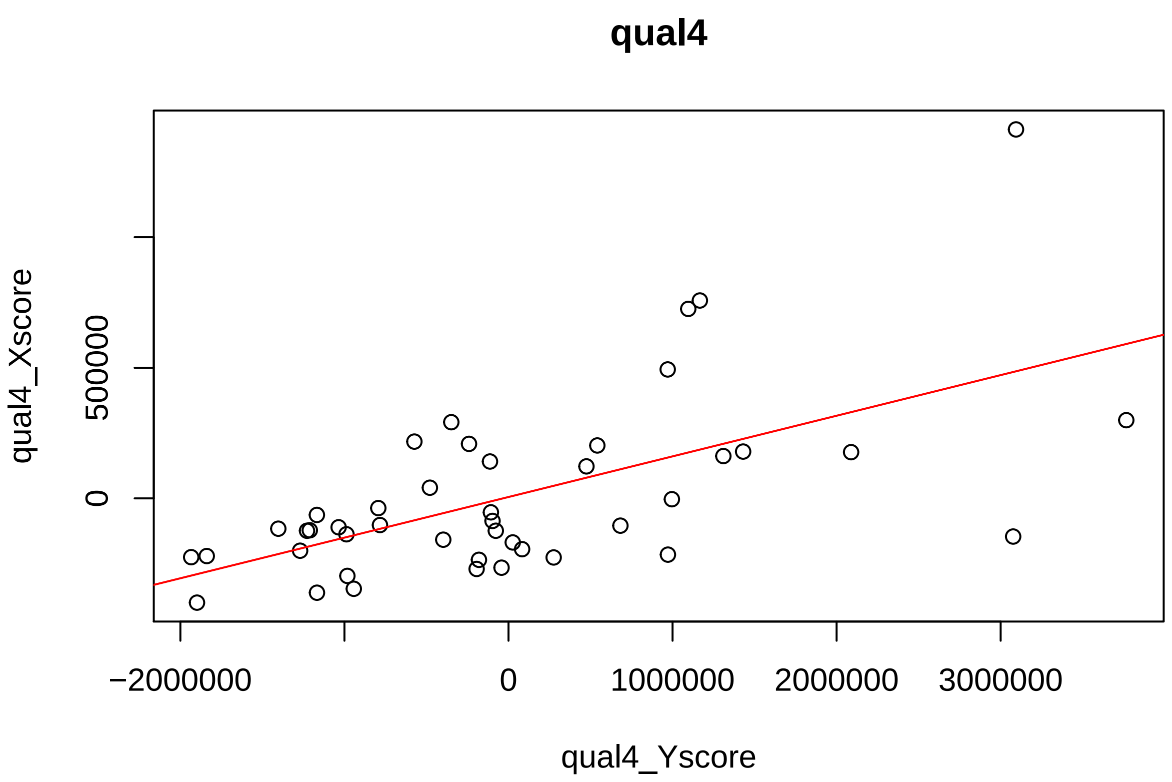
```
## $coefficients
##                  Estimate   Std. Error  t value        Pr(>|t|)
## (Intercept) 12067.9909966 51788.0587201 0.2330265 0.816872759777
## fit$U[, i]      0.1280743     0.0240982 5.3146824 0.000003811671
##
## $adj.r.squared
## [1] 0.3878642
```



```
## $coefficients
##                   Estimate     Std. Error   t value       Pr(>|t|)
## (Intercept) -7031.0660512 59674.33499381 -0.117824 0.906768834674
## fit$U[, i]      0.2743636     0.05106519  5.372811 0.000003149739
##
## $adj.r.squared
## [1] 0.3932304
```



```
## $coefficients
##                 Estimate     Std. Error   t value      Pr(>|t|)
## (Intercept) 5271.2013690 41303.14095750 0.1276223 0.89905699394
## fit$U[, i]     0.1555522     0.03184589 4.8845290 0.00001547451
##
## $adj.r.squared
## [1] 0.3470863
```

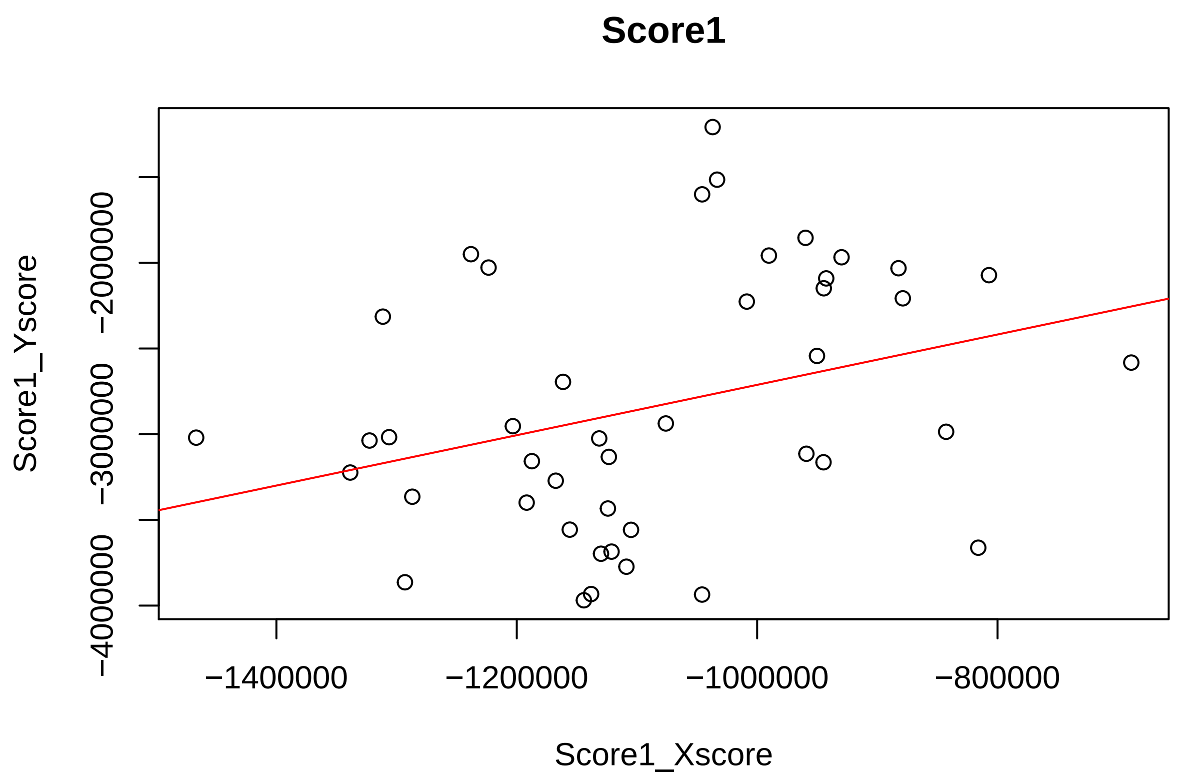


```r
jointpred <- 1 #user can change it!
tablen <- 10 #user can change it!

        getting <- lm(fit$Tt[,jointpred] ~ fit$U[,jointpred])

        tableans <- melt(sort(cooks.distance(getting), decreasing = F)[1:tablen])
        colnames(tableans) <- "Cook's distance"

        tableans <- cbind(tableans, fit$U[,jointpred][order(cooks.distance(getting), decreasing=F)[1:tal
        colnames(tableans)[2] <- "X-axis(Omics2) score"

        tableans <- cbind(tableans, fit$Tt[,jointpred][order(cooks.distance(getting), decreasing=F)[1:ta
        colnames(tableans)[3] <- "Y-axis(Omics1) score"

        tableans[,1] <- round(tableans, 5)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, , 1, value = structure(list(`Cook's
## distance` = c(0.00001, : provided 3 variables to replace 1 variables
```

```r
        tableans2 <- tableans

#quality of predictions of Y with X
for(i in 1:dim(fit1$U)[2]){
  name <- paste0("Score", i)
  assign(name, lm(fit$U[,i] ~ fit$Tt[,i]))
  getting <- get(name)
  print(summary(getting)[4]) #coefficients
  print(summary(getting)[9]) #adj R^2

  plot(fit$Tt[,i], fit$U[,i], main = name, xlab = paste0(name, "_Xscore"), ylab = paste0(name, "_Yscore
}
```

```
## $coefficients
##                     Estimate     Std. Error   t value   Pr(>|t|)
## (Intercept) -1243106.318261 726281.5470438 -1.711604 0.09434606
## fit$Tt[, i]        1.469042      0.6616403  2.220303 0.03184908
##
## $adj.r.squared
## [1] 0.08373674
```



```
        getting <- lm(fit$U[,jointpred] ~ fit$Tt[,jointpred])

        tableans <- melt(sort(cooks.distance(getting), decreasing = F)[1:tablen])
        colnames(tableans) <- "Cook's distance"

        tableans <- cbind(tableans, fit$Tt[,jointpred][order(cooks.distance(getting), decreasing=F)[1:ta
        colnames(tableans)[2] <- "X-axis(Omics1) score"

        tableans <- cbind(tableans, fit$U[,jointpred][order(cooks.distance(getting), decreasing=F)[1:tab
        colnames(tableans)[3] <- "Y-axis(Omics2) score"

        tableans[,1] <- round(tableans, 5)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, , 1, value = structure(list(`Cook's
## distance` = c(0.00002, : provided 3 variables to replace 1 variables
```

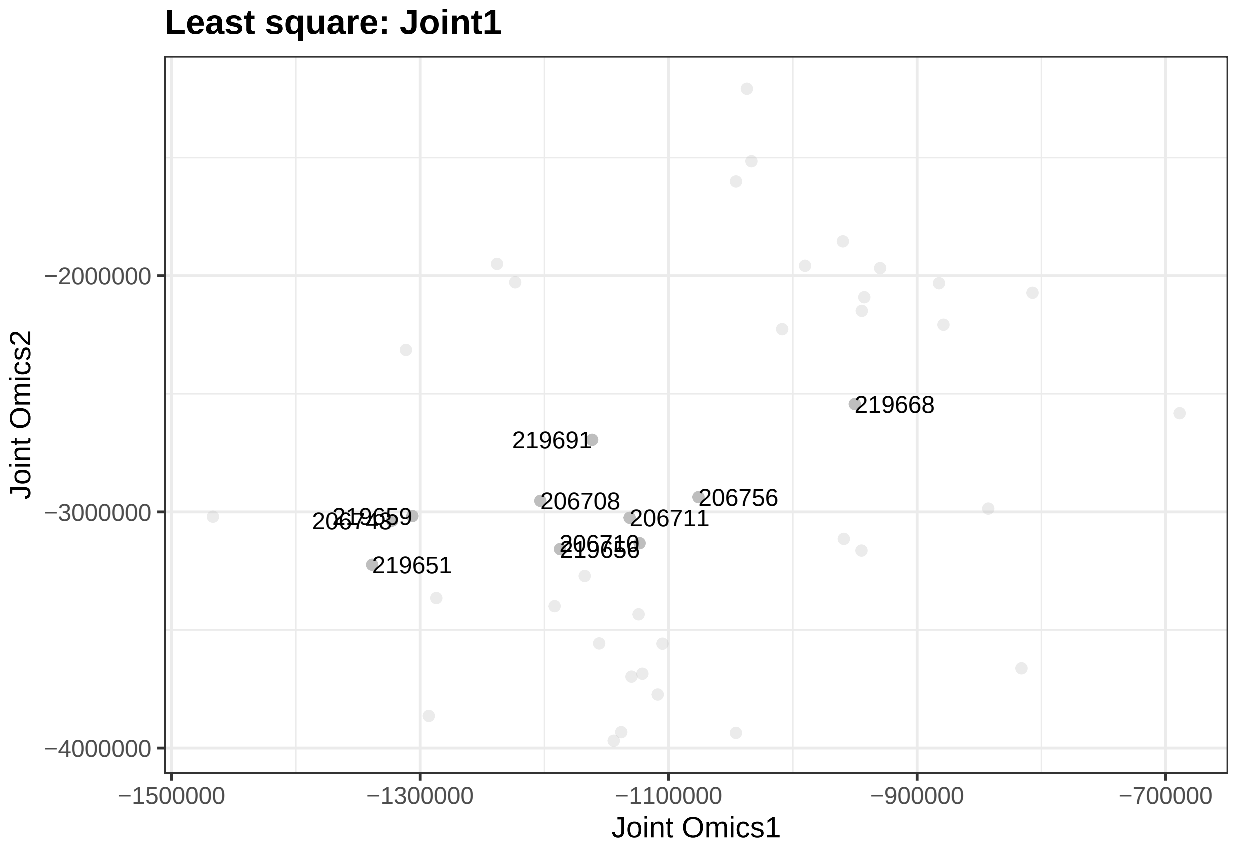
```
tableans
```

```
##        Cook's distance X-axis(Omics1) score Y-axis(Omics2) score
## 219651         0.00002           -1338565.2             -3223719
## 206708         0.00012           -1203255.8             -2953130
## 206756         0.00030          -1075981.0              -2937444
## 206711         0.00036          -1131399.9              -3024836
## 219668         0.00037           -950215.5              -2543281
## 219656         0.00095          -1187420.6              -3157452
## 206710         0.00140          -1123379.2              -3132404
## 219659         0.00150          -1306239.7              -3017521
## 206743         0.00178          -1322524.0              -3036918
## 219691         0.00186          -1161521.1              -2694666
```

```
tableans2
```

```
##        Cook's distance X-axis(Omics2) score Y-axis(Omics1) score
## 206702         0.00001             -3433620              -1124175
## 206756         0.00014             -2937444              -1075981
## 206710         0.00016             -3132404              -1123379
## 215075         0.00035             -3556920              -1155892
## 224324         0.00035             -3697552              -1129911
## 206711         0.00054             -3024836              -1131400
## 206720         0.00074             -3685263              -1121137
## 224327         0.00089             -3969002              -1144177
## 206739         0.00092             -2226221              -1008608
## 224323         0.00101             -3558065              -1104929
```

```
#plots with the texts based on low cook's distance

        getting <- lm(fit$U[,jointpred] ~ fit$Tt[,jointpred])

        datas <- data.frame(Omics1 = fit$Tt[,jointpred], Omics2 = fit$U[,jointpred])
        datas_copy <- datas
        datas_copy[-(order(cooks.distance(getting), decreasing=F)[1:tablen]), ] = 0
        datas_copy <- sign(datas_copy)
        datas_copy <- abs(datas_copy)

        g <- ggplot(datas, aes(x = Omics1, y = Omics2)) + geom_point(alpha = 0.3+0.7*datas_copy[,1], co
        g <- g + geom_text(aes(label = c(stringr::str_sub(rownames(datas_copy), start = 0))),
                           hjust = rep(c(0, 1), length.out = nrow(datas_copy)), size = 3,
                           alpha = datas_copy[,1])
        g <- g + theme_bw() + labs(title = paste0("Least square: Joint", jointpred),
                  x = "Joint Omics1", y = "Joint Omics2") +
          theme(plot.title = element_text(face='bold'))

        g
```



Comment:

Now, we still stay in the 2-dimension; however, now, we look into the score instead of loadings. Score helps us in predictions. Score is a new component (or new axis).

One good thing about O2PLS function is it provides the least square linear model of T and U. (i.e. U = TB_T + H, where B_T is a square matrix of representing regression coefficents for two models and H is a noise) (T = UB_U + H_hat)

To see how perfect the prediction between each score, I plotted linear plot and spitted out all the p-values and adjusted r-square, and coefficients.

On my website, I would plan to include both of “predictions of X with Y” and “predictions of Y with X”, although they are sort of repetitive (as this is linear, linear of Y on X is flipped from linear of X on Y), since this can be quite confusing when there are many samples.

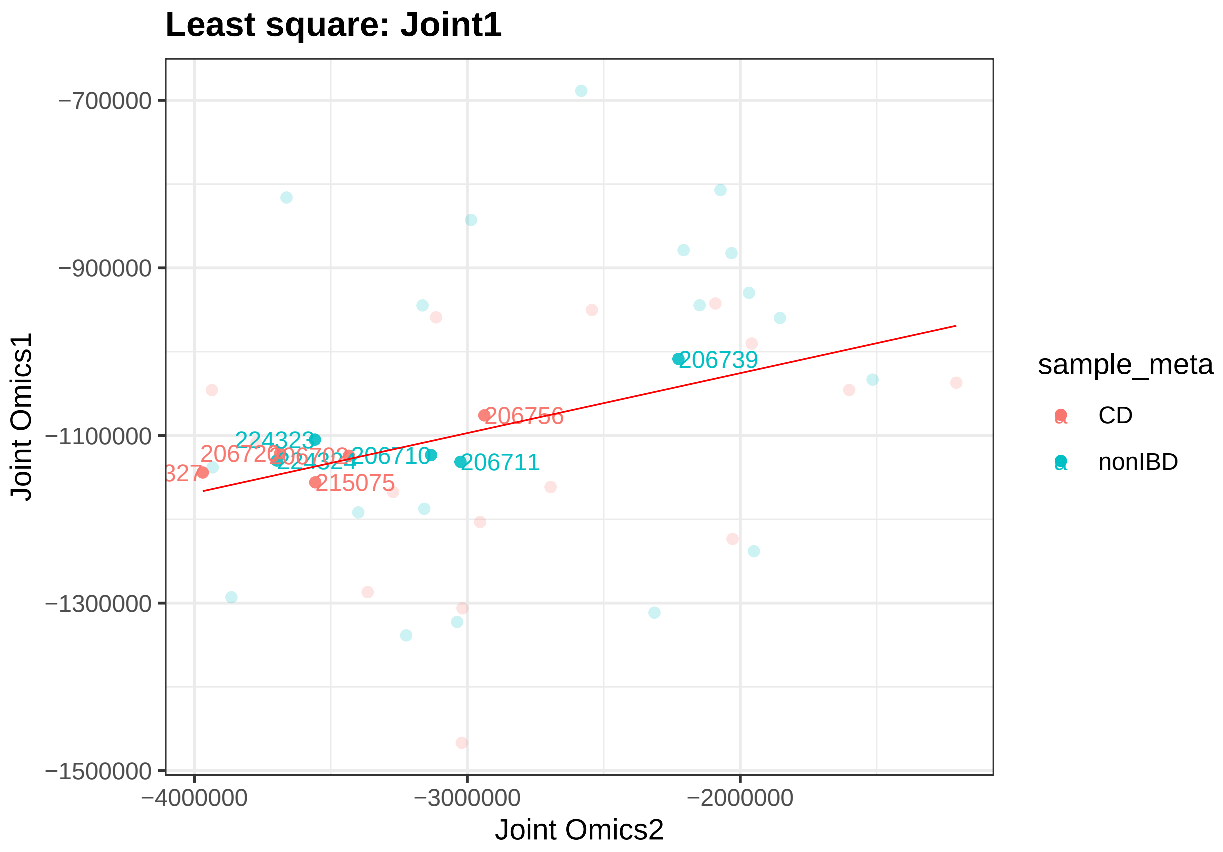
Also, my plan is I am also going to utilize cook’s distance (combination of outliers and leverages: https://en.wikipedia.org/wiki/Cook%27s_distance#Definition) for the users. I will talk about it in more details in coming Friday’s lab meeting. [since you guys said cook’s distance is not nnecessary, I am going to make it optionnal for the users]

*Understand this plot as a 2D PCA score plots. In PCA, we do not use score v.s. score plot as they are orthogonal, but, here, in joint components, we can do this! That’s the beauty of this O2PLS!

#Main - plot5

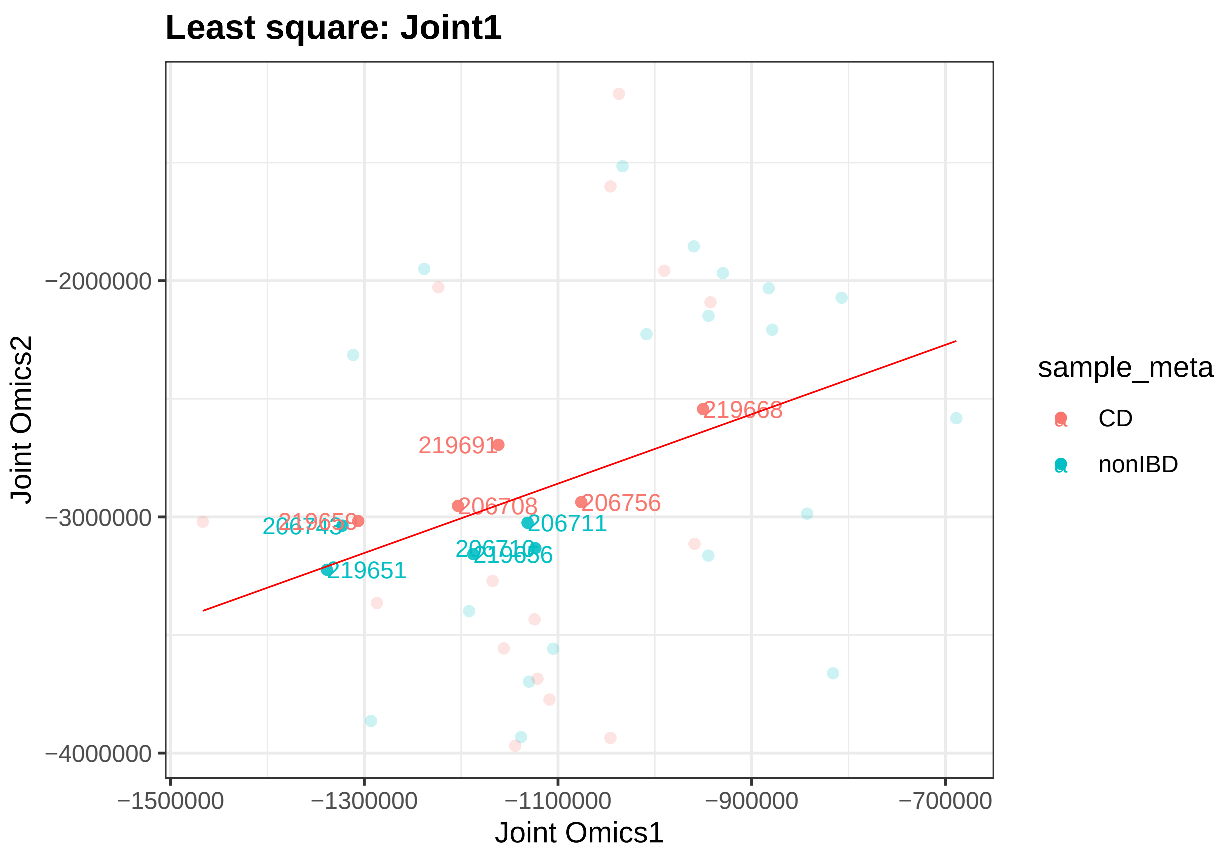
```
#Omics2 based on Omics1
getting <- lm(fit$U[,jointpred] ~ fit$Tt[,jointpred])

        datas <- data.frame(Omics1 = fit$Tt[,jointpred], Omics2 = fit$U[,jointpred])
        datas_copy <- datas
        datas_copy[-(order(cooks.distance(getting), decreasing=F)[1:tablen]), ] = 0
        datas_copy <- sign(datas_copy)
        datas_copy <- abs(datas_copy)

datas <- cbind(data1[,1], datas)
            colnames(datas)[1] <- "sample_meta"

            g <- ggplot(datas, aes(x = Omics1, y = Omics2, col = sample_meta)) +
              geom_point(alpha = 0.2+0.7*datas_copy[,1])
            g <- g + geom_text(aes(label = c(stringr::str_sub(rownames(datas_copy), start = 0))),
                               hjust = rep(c(0, 1), length.out = nrow(datas_copy)), size = 3,
                               alpha = datas_copy[,1])
            g <- g + theme_bw() + labs(title = paste0("Least square: Joint", jointpred),
                     x = "Joint Omics1", y = "Joint Omics2", fill = "sample-meta") +
              theme(plot.title = element_text(face='bold'))
            g <- g + stat_smooth(method="lm", se=FALSE, col = "red", size = 0.3)

g2 <- g


#Omics1 based on Omics2

 getting <- lm(fit$Tt[,jointpred] ~ fit$U[,jointpred])

        datas <- data.frame(Omics2 = fit$U[,jointpred], Omics1 = fit$Tt[,jointpred])
        datas_copy <- datas
        datas_copy[-(order(cooks.distance(getting), decreasing=F)[1:tablen]), ] = 0
        datas_copy <- sign(datas_copy)
        datas_copy <- abs(datas_copy)

datas <- cbind(data1[,1], datas)
            colnames(datas)[1] <- "sample_meta"

            g <- ggplot(datas, aes(x = Omics2, y = Omics1, col = sample_meta)) +
              geom_point(alpha = 0.2+0.7*datas_copy[,1])
            g <- g + geom_text(aes(label = c(stringr::str_sub(rownames(datas_copy), start = 0))),
                               hjust = rep(c(0, 1), length.out = nrow(datas_copy)), size = 3,
                               alpha = datas_copy[,1])
            g <- g + theme_bw() + labs(title = paste0("Least square: Joint", jointpred),
                     x = "Joint Omics2", y = "Joint Omics1") +
              theme(plot.title = element_text(face='bold'))
            g <- g + stat_smooth(method="lm", se=FALSE, col = "red", size = 0.3)



g
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
g2
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Comment:

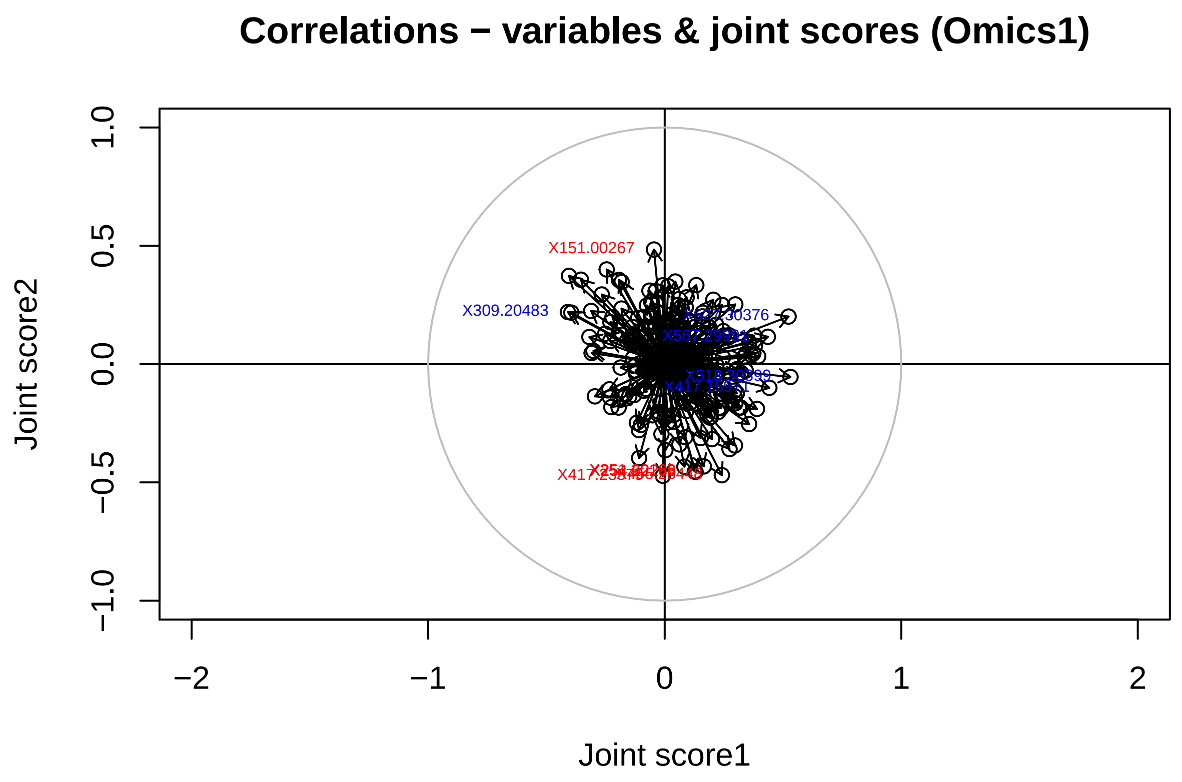
So, we looked at how well we could predict omics1 based on omics2 in joint components, or vice versa. So, the next step we should take is draw the plots with sample meta-data we have.

In here, we can see how the sample meta-data is spreaded out on the joint plots!

#Main - plot6

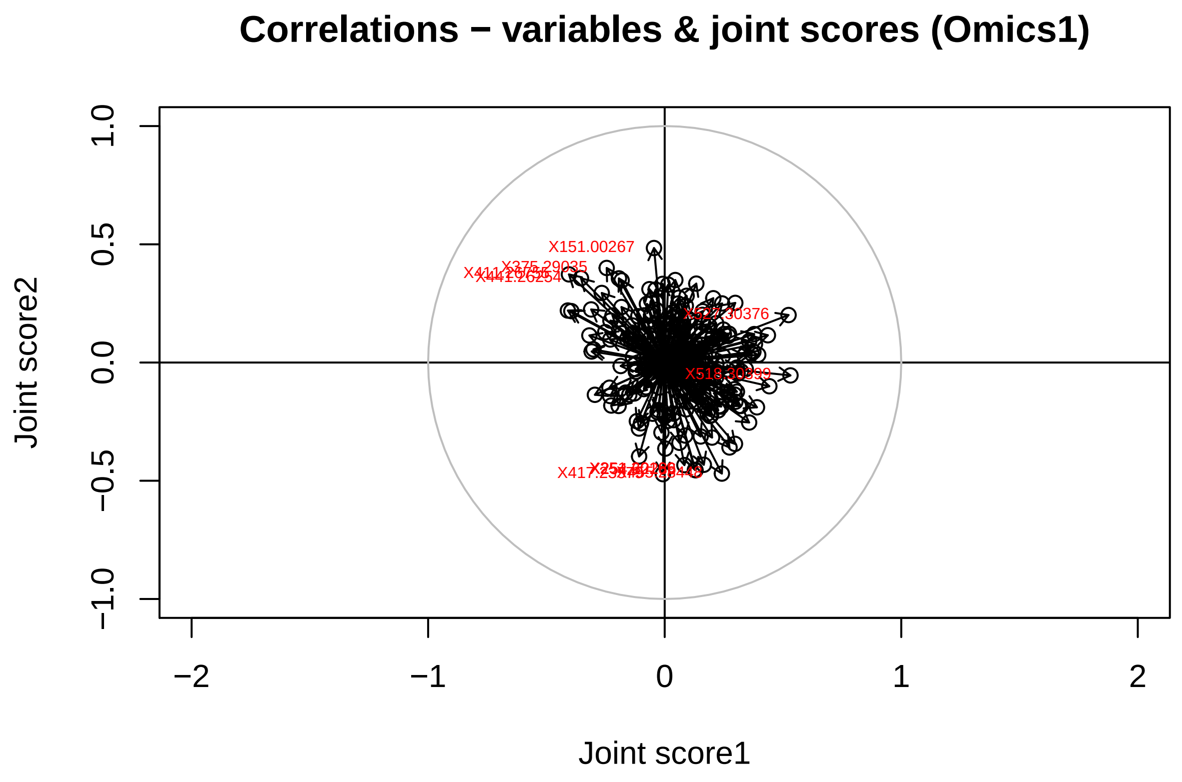
```
#Omics1
cors <- as.data.frame(cbind(pc1 = t(cor(fit$Tt, data1[,-1]))[,1],
                                pc2 = t(cor(fit$Tt, data1[,-1]))[,2],
                    length = sqrt((t(cor(fit$Tt, data1[,-1]))[,1])^2 + (t(cor(fit$Tt, data1[,-1]))[,2])
                      )
                    )

if(dim(cors)[1] > 200){ #for visual purpose
    tt <- sample(nrow(cors), 200);
    cors <- cors[tt,,drop=FALSE]
}
```

```
#1. getting the five absolute values of high pc1 and five absolute values of high pc2
plot(cors$pc1, cors$pc2,
     main = "Correlations - variables & joint scores (Omics1)",
     xlab = "Joint score1", ylab = "Joint score2",
     xlim = c(-1.0, 1.0), ylim = c(-1.0, 1.0), asp = 1)
abline(h = 0, v = 0, lty = 1)
arrows(0, 0, cors$pc1, cors$pc2,
       length = 0.07, angle = 30, code = 3)
draw.circle(0, 0, 1, border = "gray", lty = 1, lwd = 1)
text(cors$pc1[order(abs(cors$pc2), decreasing = T)[1:5]],
     cors$pc2[order(abs(cors$pc2), decreasing = T)[1:5]],
     labels = rownames(cors)[order(abs(cors$pc2), decreasing = T)[1:5]],
     pos = 2,
     col = "red", cex = 0.5)
text(cors$pc1[order(abs(cors$pc1), decreasing = T)[1:5]],
     cors$pc2[order(abs(cors$pc1), decreasing = T)[1:5]],
     labels = rownames(cors)[order(abs(cors$pc1), decreasing = T)[1:5]],
     pos = 2,
     col = "blue", cex = 0.5)
```



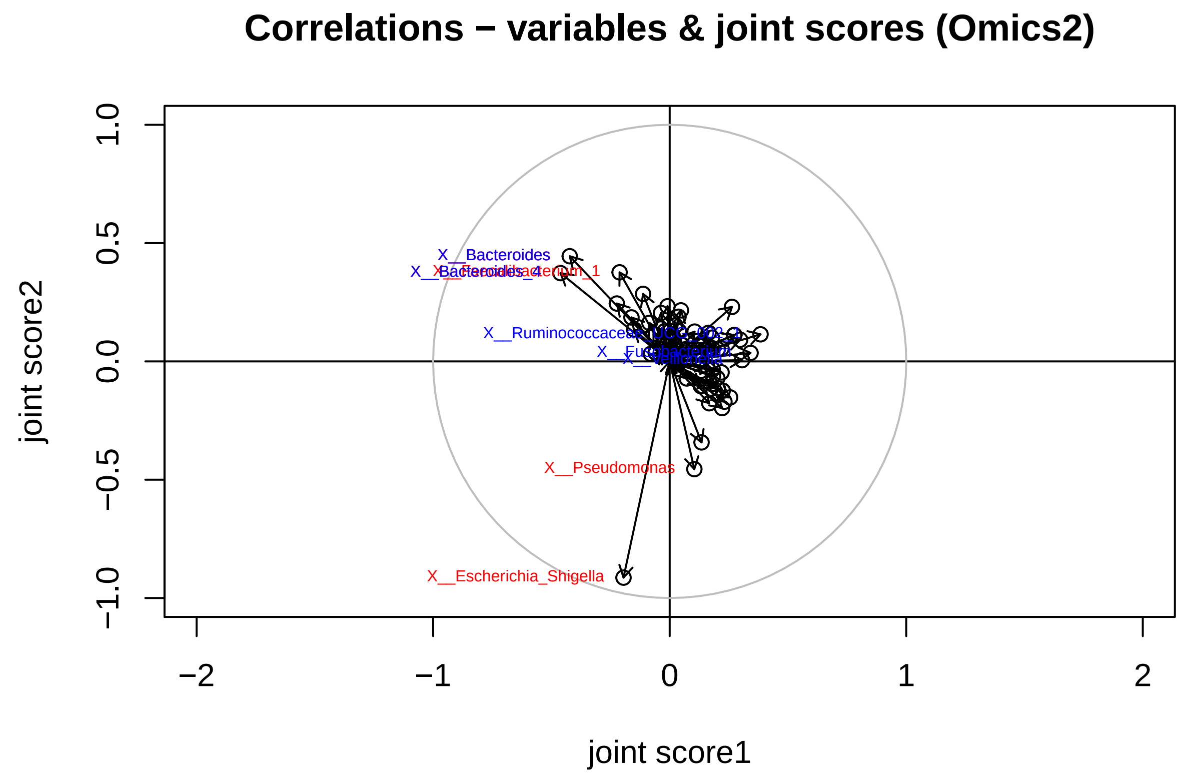
```
#2. getting the 10 high length
plot(cors$pc1, cors$pc2,
     main = "Correlations - variables & joint scores (Omics1)",
     xlab = "Joint score1", ylab = "Joint score2",
```

```r
     xlim = c(-1.0, 1.0), ylim = c(-1.0, 1.0), asp = 1)
abline(h = 0, v = 0, lty = 1)
arrows(0, 0, cors$pc1, cors$pc2,
       length = 0.07, angle = 30, code = 3)
draw.circle(0, 0, 1, border = "gray", lty = 1, lwd = 1)
text(cors$pc1[order(cors$length, decreasing = T)[1:10]],
     cors$pc2[order(cors$length, decreasing = T)[1:10]],
     labels = rownames(cors)[order(cors$length, decreasing = T)[1:10]],
     pos = 2,
     col = "red", cex = 0.5)
```

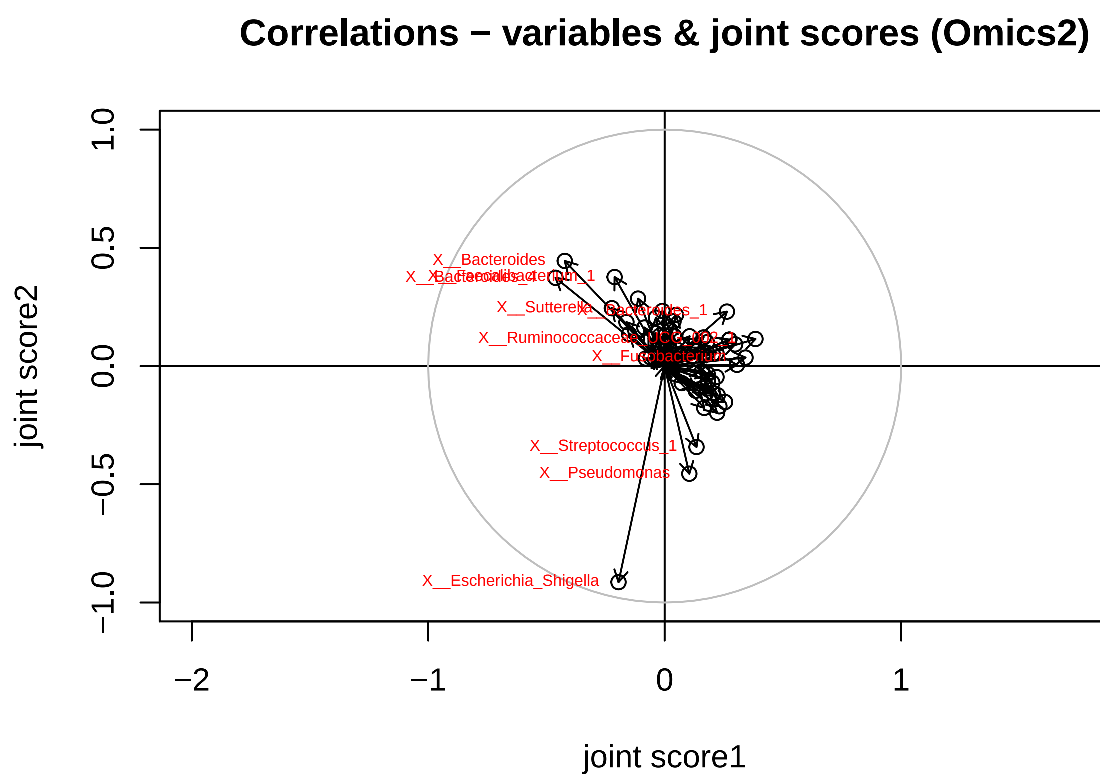


```r
#Omics2

#The max of the length can be sqrt(2), so i normalized by dividing it.
cors <- as.data.frame(cbind(pc1 = t(cor(fit$U, data2[,-1]))[,1], pc2 = t(cor(fit$U, data2[,-1]))[,2],
                    length = sqrt((t(cor(fit$U, data2[,-1]))[,1])^2 + (t(cor(fit$U, data2[,-1]))[,2])^2
                    )

if(dim(cors)[1] > 200){ #for visual purpose
    tt <- sample(nrow(cors), 200);
    cors <- cors[tt,,drop=FALSE]
}

#1. getting the five absolute values of high pc1 and five absolute values of high pc2
plot(cors$pc1, cors$pc2,
     main = "Correlations - variables & joint scores (Omics2)",
     xlab = "joint score1", ylab = "joint score2",
     xlim = c(-1.0, 1.0), ylim = c(-1.0, 1.0), asp = 1)
abline(h = 0, v = 0, lty = 1)
```

```
arrows(0, 0, cors$pc1, cors$pc2,
       length = 0.07, angle = 30, code = 3)
draw.circle(0, 0, 1, border = "gray", lty = 1, lwd = 1)
text(cors$pc1[order(abs(cors$pc2), decreasing = T)[1:5]],
     cors$pc2[order(abs(cors$pc2), decreasing = T)[1:5]],
     labels = rownames(cors)[order(abs(cors$pc2), decreasing = T)[1:5]],
     pos = 2,
     col = "red", cex = 0.5)
text(cors$pc1[order(abs(cors$pc1), decreasing = T)[1:5]],
     cors$pc2[order(abs(cors$pc1), decreasing = T)[1:5]],
     labels = rownames(cors)[order(abs(cors$pc1), decreasing = T)[1:5]],
     pos = 2,
     col = "blue", cex = 0.5)
```



```
#2. getting the 10 high euclidean dist
plot(cors$pc1, cors$pc2,
     main = "Correlations - variables & joint scores (Omics2)",
     xlab = "joint score1", ylab = "joint score2",
     xlim = c(-1.0, 1.0), ylim = c(-1.0, 1.0), asp = 1)
abline(h = 0, v = 0, lty = 1)
arrows(0, 0, cors$pc1, cors$pc2,
       length = 0.07, angle = 30, code = 3)
draw.circle(0, 0, 1, border = "gray", lty = 1, lwd = 1)
text(cors$pc1[order(cors$length, decreasing = T)[1:10]],
     cors$pc2[order(cors$length, decreasing = T)[1:10]],
     labels = rownames(cors)[order(cors$length, decreasing = T)[1:10]],
     pos = 2,
     col = "red", cex = 0.5)
```



Comment:

Now, I am going to present component x v.s. component y (i.e. comp 1 v.s. comp 2) on the joint components from the same omics!

This can be interpreted in the same way as PCA correlation circle.

The radius of the circles is 1, meaning that as the maximum length of arrows are all 1, (I normalized it to 1) and the closer to the circle, the higher correlation they are related with. Also, the direction of the arrows can help us understand the relationships between two components. (pick 300 randomly if there are more than 300 variables)

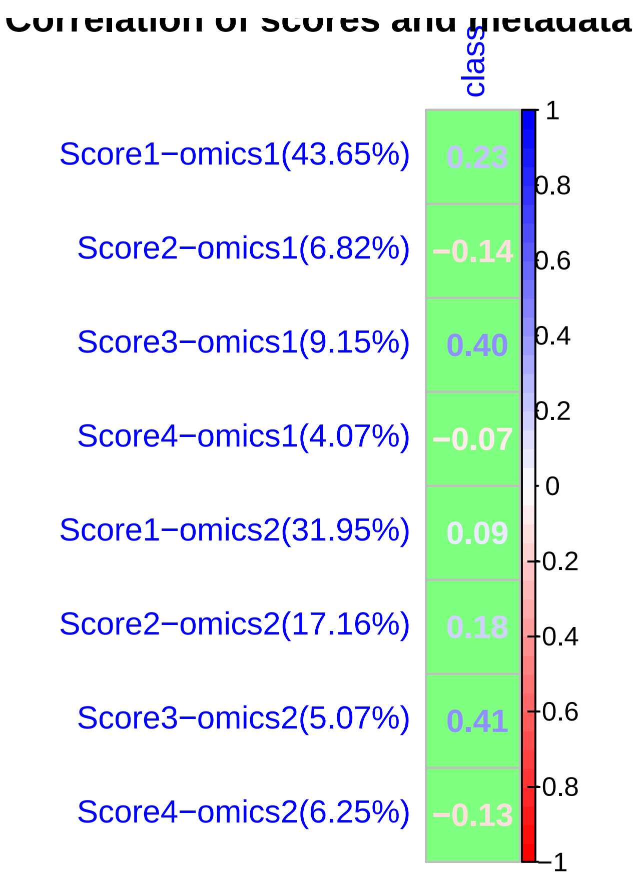
So, the goal of this is: I calculated the correlations between scores and the original datasets (from the same dataset), that way, what we could get from here is that we would know how the joint components are related with the origianl dataset (speicifically, the variables). Let's say we know that the component1 from omics2 is closely related to the brain tumor, and then, we could perform this, and we might get some idea that what kinds of variables are related to this component.

Again, this will not be displayed if you have only one joint component.

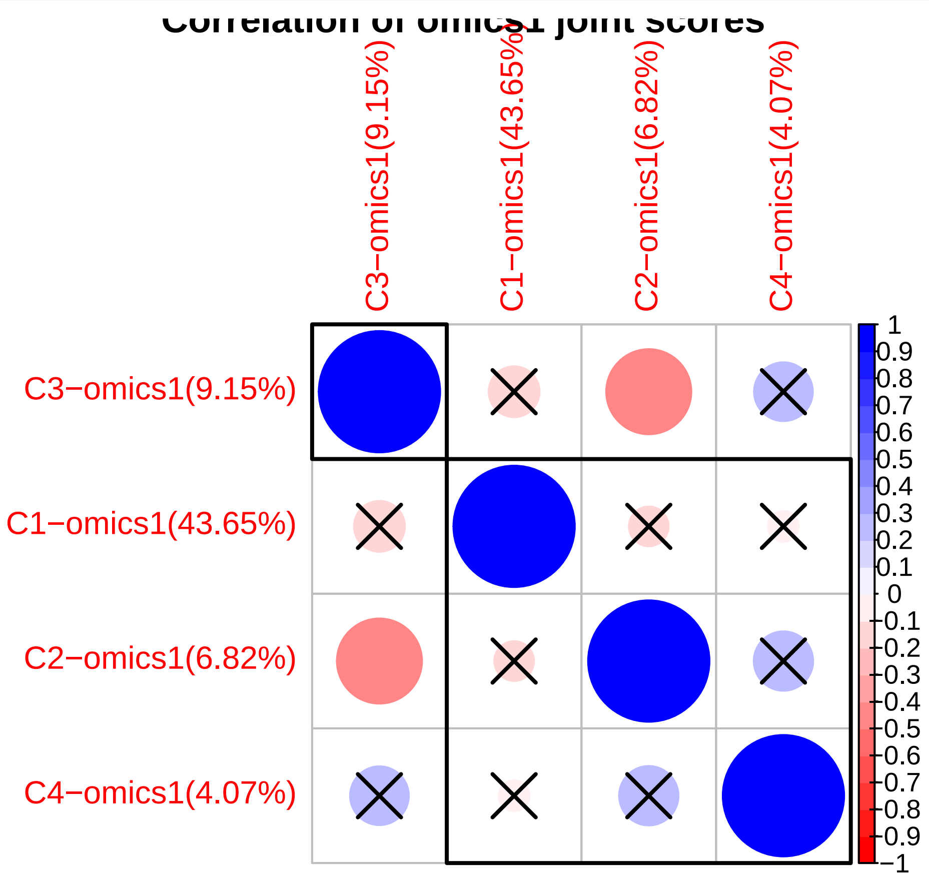
#Main - plot7

```
    col3 <- colorRampPalette(c("red", "white", "blue"))

xx <- factor(as.numeric(data1[,1]), levels = as.numeric((1:length(levels(data1[,1])))))
```

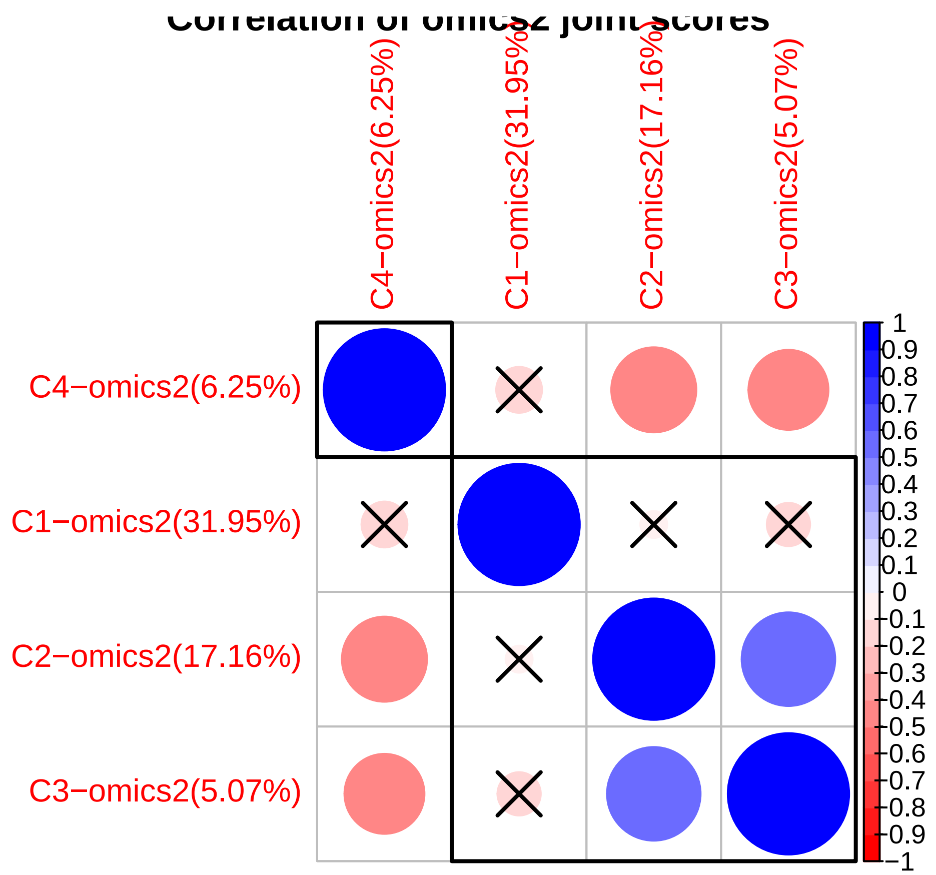
```
        xx <- as.numeric(as.character(xx))
        xx <- as.data.frame(xx)
        colnames(xx) <- colnames(data1)[1]

        M1 <- cor(fit$Tt, xx)
        x <- c()
        for(i in 1:nrow(M1)){
          x[i] <- paste0("Score", i, "-omics1(", model_summary$Omics1[i], "%)")
        }
        rownames(M1) <- x

        M2 <- cor(fit$U, xx)
        y <- c()
        for(i in 1:nrow(M2)){
          y[i] <- paste0("Score", i, "-omics2(", model_summary$Omics2[i], "%)")
        }
        rownames(M2) <- y

corrplot(rbind(M1, M2), method = "number", bg = "#7FFF7F",
           title = "Correlation of scores and metadata", col = col3(40),
           tl.col = "blue", cl.pos = "r", cl.ratio = 0.5)
```



```
jointx <- fit$Tt
    x <- c()
    for(i in 1:ncol(jointx)){
      x[i] <- paste0("C", i, "-omics1(", model_summary$Omics1[i], "%)")
    }
    colnames(jointx) <- x

    col3 <- colorRampPalette(c("red", "white", "blue"))
res1 <- cor.mtest(jointx, conf.level = .95)
corrplot(cor(jointx), p.mat = res1$p,
             order = "hclust", insig = "pch", addrect = 2, col = col3(20),
             title = "Correlation of omics1 joint scores"
             )
```

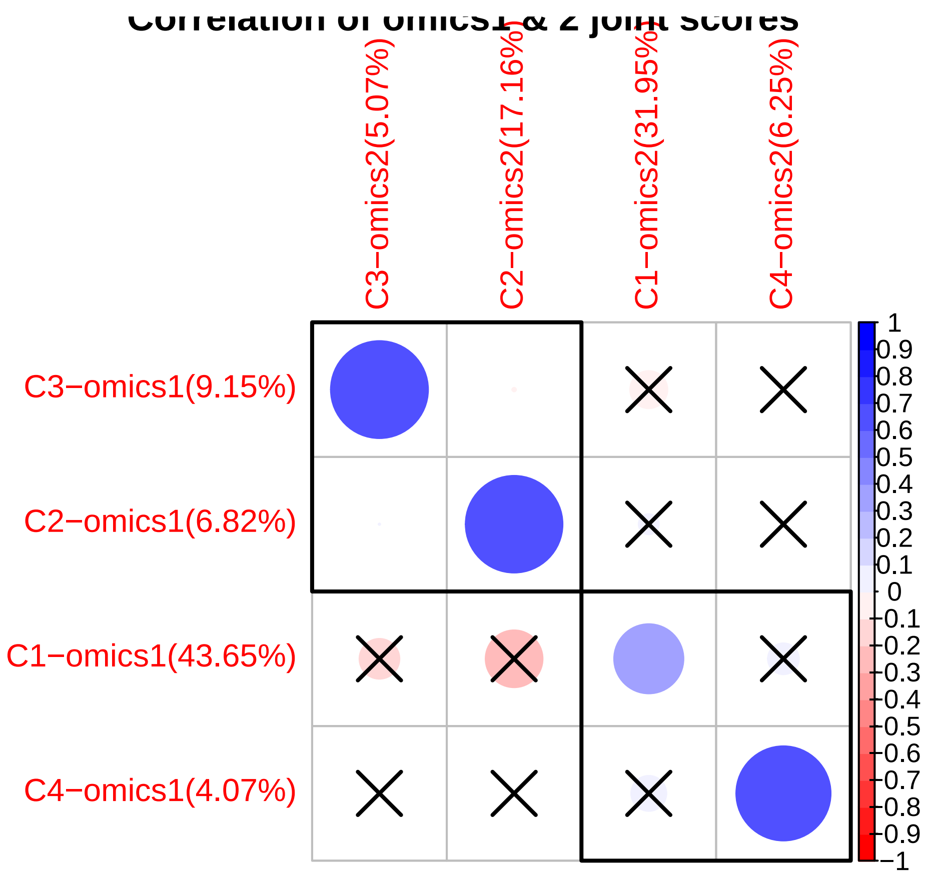


```
jointy <- fit$U
    y <- c()
    for(i in 1:ncol(jointy)){
      y[i] <- paste0("C", i, "-omics2(", model_summary$Omics2[i], "%)")
    }
    colnames(jointy) <- y
res1 <- cor.mtest(jointy, conf.level = .95)
corrplot(cor(jointy), p.mat = res1$p,
             order = "hclust", insig = "pch", addrect = 2, col = col3(20),
         title = "Correlation of omics2 joint scores"
             )
```



```
res1 <- cor.mtest(cbind(jointx, jointy), conf.level = .95)
corrplot(cor(jointx, jointy), p.mat = res1$p,
             order = "hclust", insig = "pch", addrect = 2, col = col3(20),
         title = "Correlation of omics1 & 2 joint scores"
             )
```

```
## Warning in corrplot(cor(jointx, jointy), p.mat = res1$p, order = "hclust", :
## p.mat and corr may be not paired, their rownames and colnames are not totally
## same!
```



Now, it's time for us do the univariate analysis: look at the correlation.

It show the correlations between all the joint scores and sample meta-data.

Also, i made few correlations plots to prove each joint scores are not cross-related (Add cross on no significant coefficient & hierarchial clustering and divide by boxes & correlation circles).

On the website, things will look way better, since I will make the user to choose: 12 shapes and 7 colors (thus, 84 combinations for them)

Last but not least, the website would have more implementations, so I recommend you to go visit the pages to have more fun on it!!! ^^