Exploring propr

David Lovell

December 5, 2016

What's in the Cane Toad data?

caneToad.counts contains 20 observations (toads) of the counts of 57580 variables (mRNAs). Each observation is associated with a location:

```
table(caneToad.groups)
```

```
## caneToad.groups
## QLD  WA
##  10  10
```

Zero replacement

Thom kept the RNAs that had at least 10 counts in at least 10 toads

```
keep <- apply(caneToad.counts, 2, function(x) sum(x >= 10) >= 10)
# This line would keep only the RNAs that have at least 10 counts in _all_ toads
# keep <- apply(caneToad.counts, 2, function(x) min(x) >= 10)
counts <- caneToad.counts[,keep]
```

This means we keep:

```
addmargins(table(keep))
```

```
## keep
## FALSE  TRUE   Sum
## 31806 25774 57580
```

Still, of the 515480 counts in this subset of the cane toad data, there are still 1427 zeros. The routines in propr automatically replace zero counts with 1. However, no such imputation happens for the scripts that I have developed. So, here I make this replacement explicitly

```
counts[counts ==0] <- 1
```

Transform and rearrange

Take the centred log ratio of counts, put that in a dataframe with the observation ID and location:

```
counts.clr <- clr(counts, check=TRUE)

counts.wide <- data.frame(
  ID=factor(1:nrow(counts)),
  Location=factor(caneToad.groups),
  counts.clr)
counts.long <- gather(counts.wide, RNA, count, -c(ID, Location))
```

Reorder the RNA factor

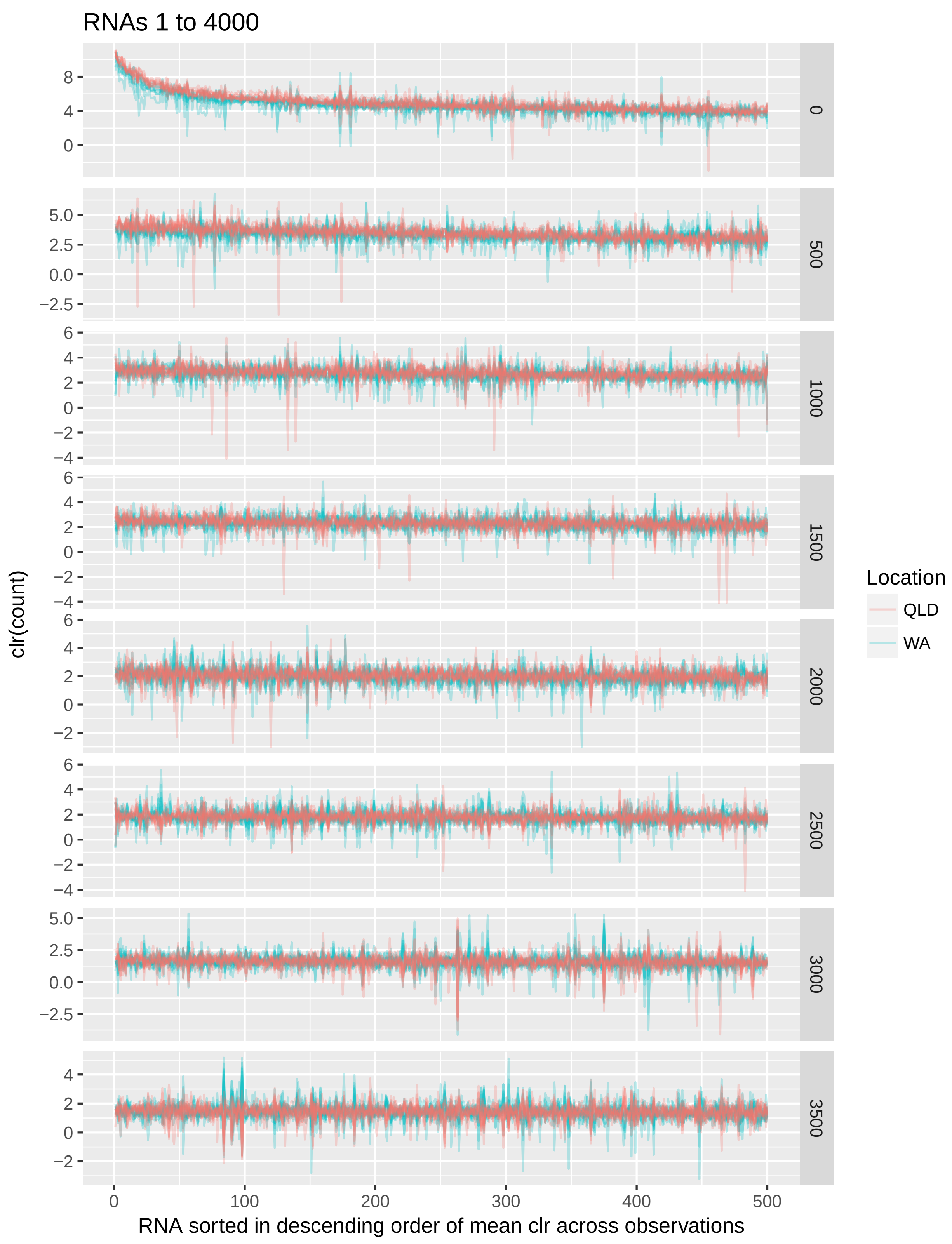
Set the levels of the `RNA` factor in descending order of mean count; this is for plotting RNA counts in a more comprehensible way

```
counts.long$RNA <- reorder(counts.long$RNA, -counts.long$count, mean)
#counts.long$RNA <- reorder(counts.long$RNA,
# ifelse(counts.long$Location=="QLD", -counts.long$count, 0), mean)
```

Look at the data

If we plot the sorted RNA counts in chunks we can see everything in a few pages

```
plot4k <- function(n=0){
  ggplot(
    subset(counts.long, n*4000 < as.numeric(RNA) & as.numeric(RNA) <= (n+1)*4000),
    aes(x=1 + (as.numeric(RNA) -1) %% 500, y=count, group=ID)) +
    geom_line(aes(colour=Location), alpha=0.25) +
    facet_grid(((as.numeric(RNA)-1) %/% 500 * 500) ~., scales="free_y") +
    labs(title=sprintf("RNAs %d to %d", n*4000 + 1, (n+1)*4000),
         x="RNA sorted in descending order of mean clr across observations",
         y="clr(count)")
}
```


RNAs 1 to 4000
clr(count)
RNA sorted in descending order of mean clr across observations
0
500
1000
1500
2000
2500
3000
3500
Location
QLD
WA

Check that Quinn and Lovell's implementations of propr agree

We sourced Lovell's implementation from `./propr-functions.R`. Now we are going to check to ensure that results from Quinn's and Lovell's implementations agree.

Let's just work with a few components

```
counts.sub <- counts[,1:4]
```

Lovell's implementation uses `phiDF()` to return a list of β, p, r^2, $\mathrm{var}(\log(x_i, x_j))$, ϕ and now ϕ_s. These results are returned as the lower triangles of the matrices to save space:

```
counts.phiDF <- phiDF(counts.sub)
```

Quinn's implementation uses `phis()`, `phit()` and `perb()` to return ϕ_s, ϕ and ρ_p respectively:

```
counts.phis <- phis(counts.sub)
counts.phit <- phit(counts.sub)
counts.perb <- perb(counts.sub)
```

The next three comparisons check for equality between ϕ_s, ϕ and ρ_p as calculated by Lovell's and Quinn's implementations. Note that we have to compare the lower triangle of the matrices:

```
all.equal(
  counts.phis@matrix[lower.tri(counts.phis@matrix)],
  counts.phiDF$phisym)
```

```
## [1] TRUE
```

```
all.equal(
  counts.phit@matrix[lower.tri(counts.phit@matrix)],
  counts.phiDF$phi)
```

```
## [1] TRUE
```

```
all.equal(
  counts.perb@matrix[lower.tri(counts.perb@matrix)],
  (1 - counts.phiDF$phisym)/(1 + counts.phiDF$phisym))
```

```
## [1] TRUE
```

While it's good that these results agree, let's check further by calculating ϕ_s, and ϕ directly.

```
counts.sub.clr <- clr(counts.sub)
CovXY <- var(counts.sub.clr)
VarX.Plus.VarY  <- outer(diag(CovXY), diag(CovXY), "+")
D <- ncol(counts.sub.clr)
VarX <- matrix(rep(diag(CovXY), D), ncol = D, byrow = TRUE)


all.equal(
  (VarX.Plus.VarY - 2 * CovXY)/(VarX.Plus.VarY + 2 * CovXY),
  phis(counts.sub)@matrix,
  check.attributes=FALSE
)
```

```
## [1] TRUE
```

```
all.equal(
  ((VarX.Plus.VarY - 2 * CovXY)/VarX)[lower.tri(VarX)],
```

```r
  phit(counts.sub)@matrix[lower.tri(VarX)]
)
```

```
## [1] TRUE
```

Continue with the cane toad analysis

Calcluate ρ_p and ϕ_s

Here we calculate both $\rho_p(\text{clr}(x_i), \text{clr}(x_j))$ and $\phi_s(\text{clr}(x_i), \text{clr}(x_j))$ for the cane toad counts. These

```
rho.p <- perb(caneToad.counts, select = keep)
```

```
## Alert: Replacing 0s in "count matrix" with 1.
```

```
phi.s <- phis(caneToad.counts, select = keep)
```

```
## Alert: Replacing 0s in "count matrix" with 1.
```

Since these should be related monotonically by

$$\phi_s = \frac{1 - \rho_p}{1 + \rho_p}$$

we should end up with the same pairs when we apply the following thresholds:

```
best.rho.p <- rho.p[">", 0.995]
best.phi.s <- phi.s["<", (1 - 0.995)/(1 + 0.995)]
top.rho.p <- simplify(best.rho.p)
top.phi.s <- simplify(best.phi.s)
```

. . . and indeed we do:

```
all.equal(
  top.rho.p@pairs, top.phi.s@pairs
)
```

```
## [1] TRUE
```

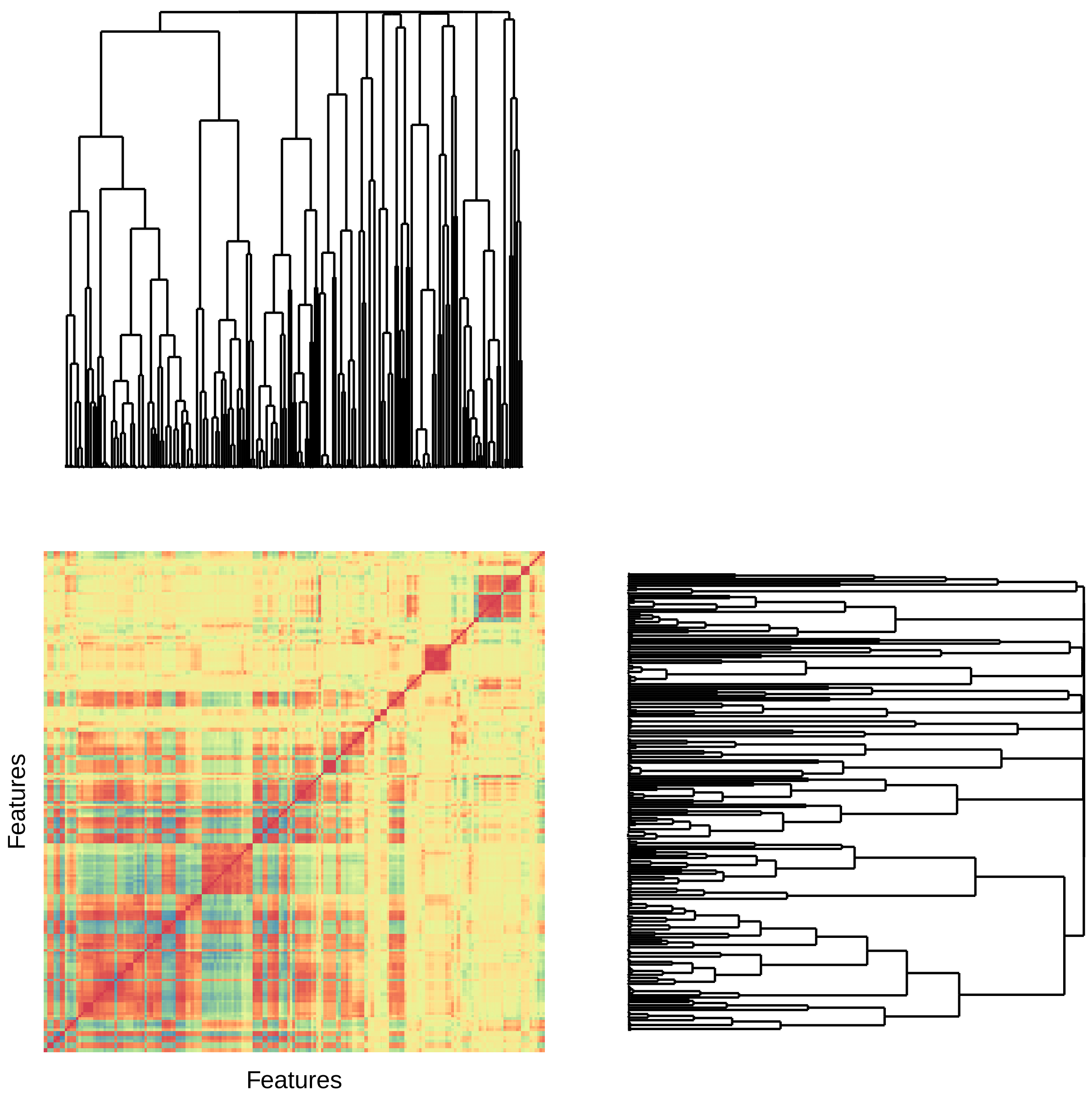
Visualise ρ_p and ϕ_s

Clustered heatmaps

Here are the clustered heatmaps for these two different distances. Note that the dendrogram of the ρ_p matrix shows less distinct clusters than the dendrogram of the ϕ_s matrix. I think this is because the values of ρ_p are constrained within the hypercube $[0, 1]^D$ while the values of ϕ_s can range over $[0, \infty)^D$.

```
dendrogram(top.rho.p)
```

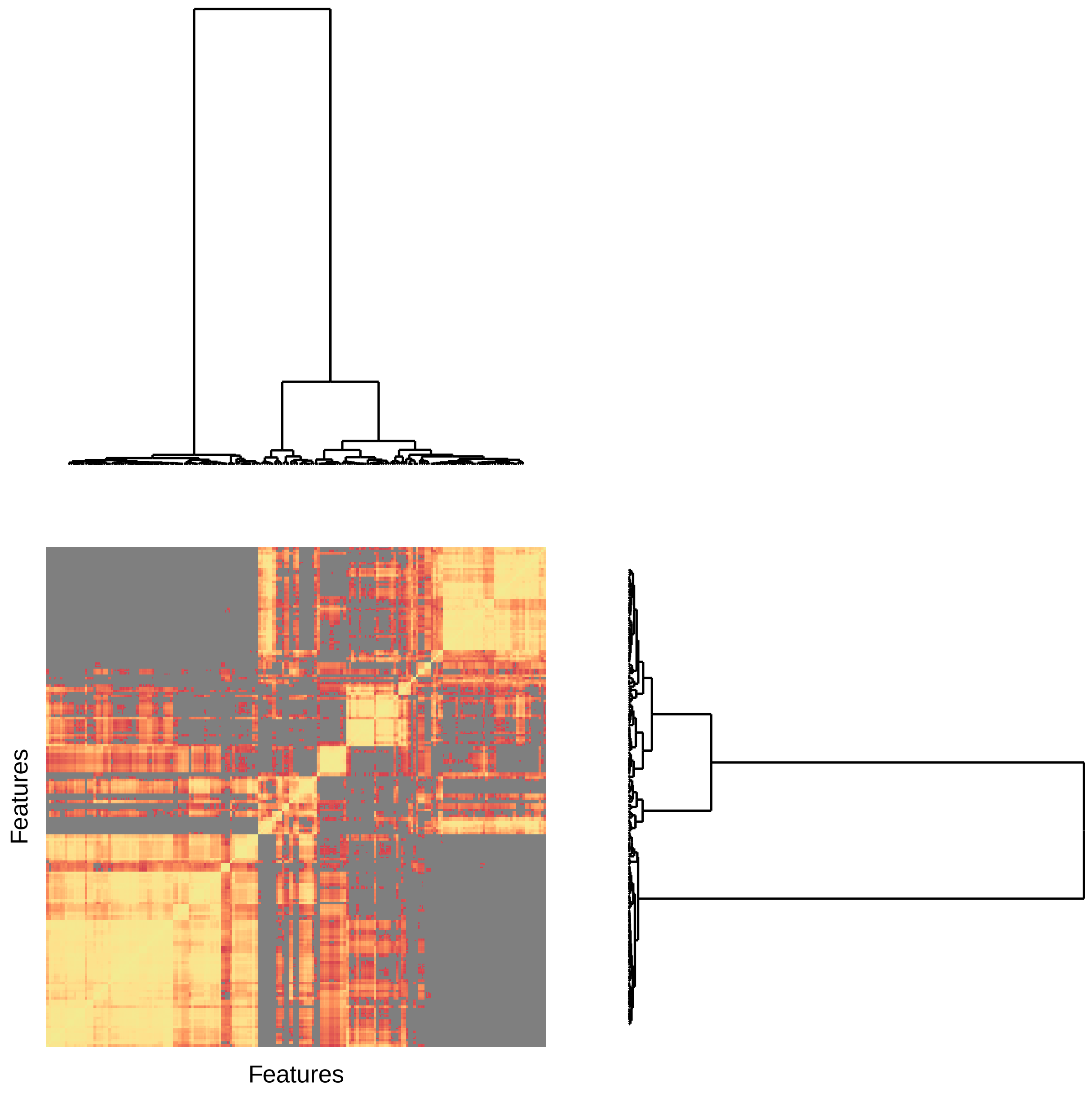
```
## Alert: Generating plot using indexed feature pairs.
```



```
## 'dendrogram' with 2 branches and 403 members total, at height 0.9999977
```

```r
dendrogram(top.phi.s)
```

```
## Alert: Generating plot using indexed feature pairs.
```



```
## 'dendrogram' with 2 branches and 403 members total, at height 62.5556
```

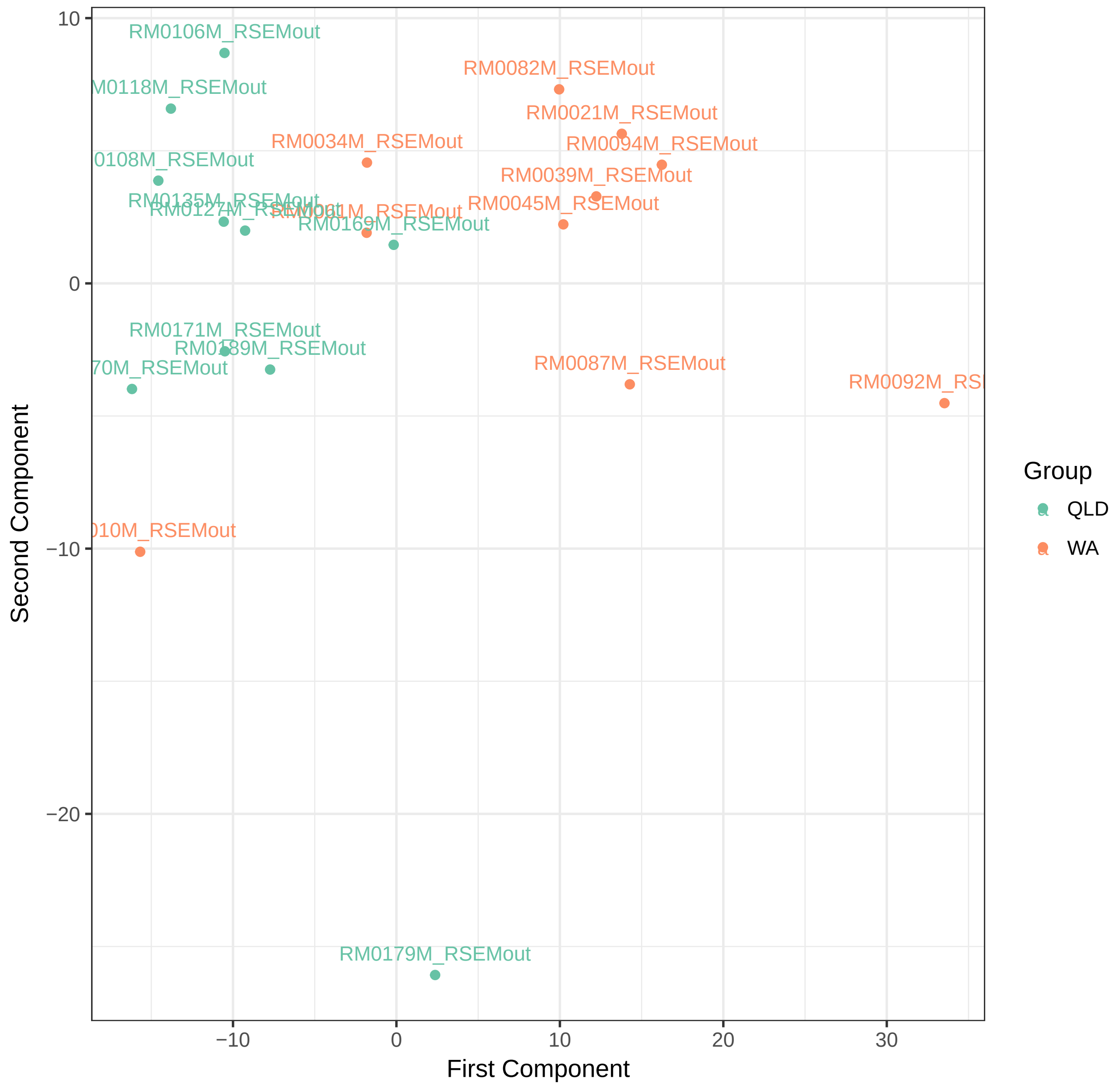
PCA projections

Here are the projections of the ρ_p and ϕ_s matrices onto their first two principal components. I'm not sure what these tell us about the data though. First comes the projection performed by `propr::mds()`

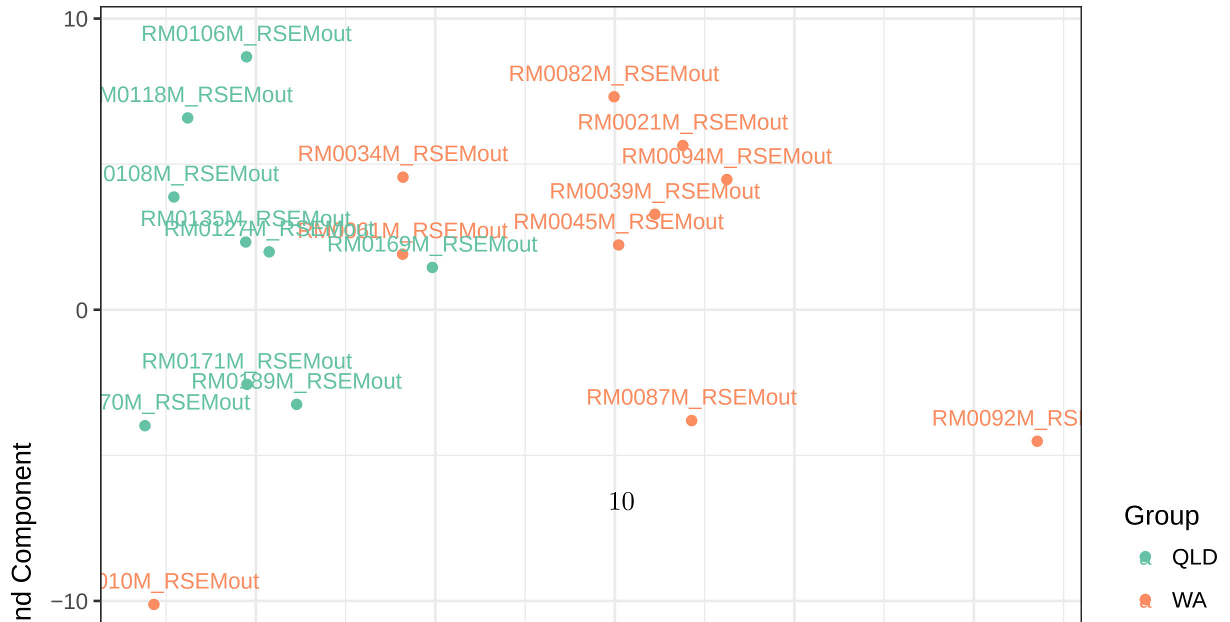
```
mds(top.rho.p, group=caneToad.groups)
```

```
## Alert: This function ignores index.
```

*lr−transformed MDS Plot

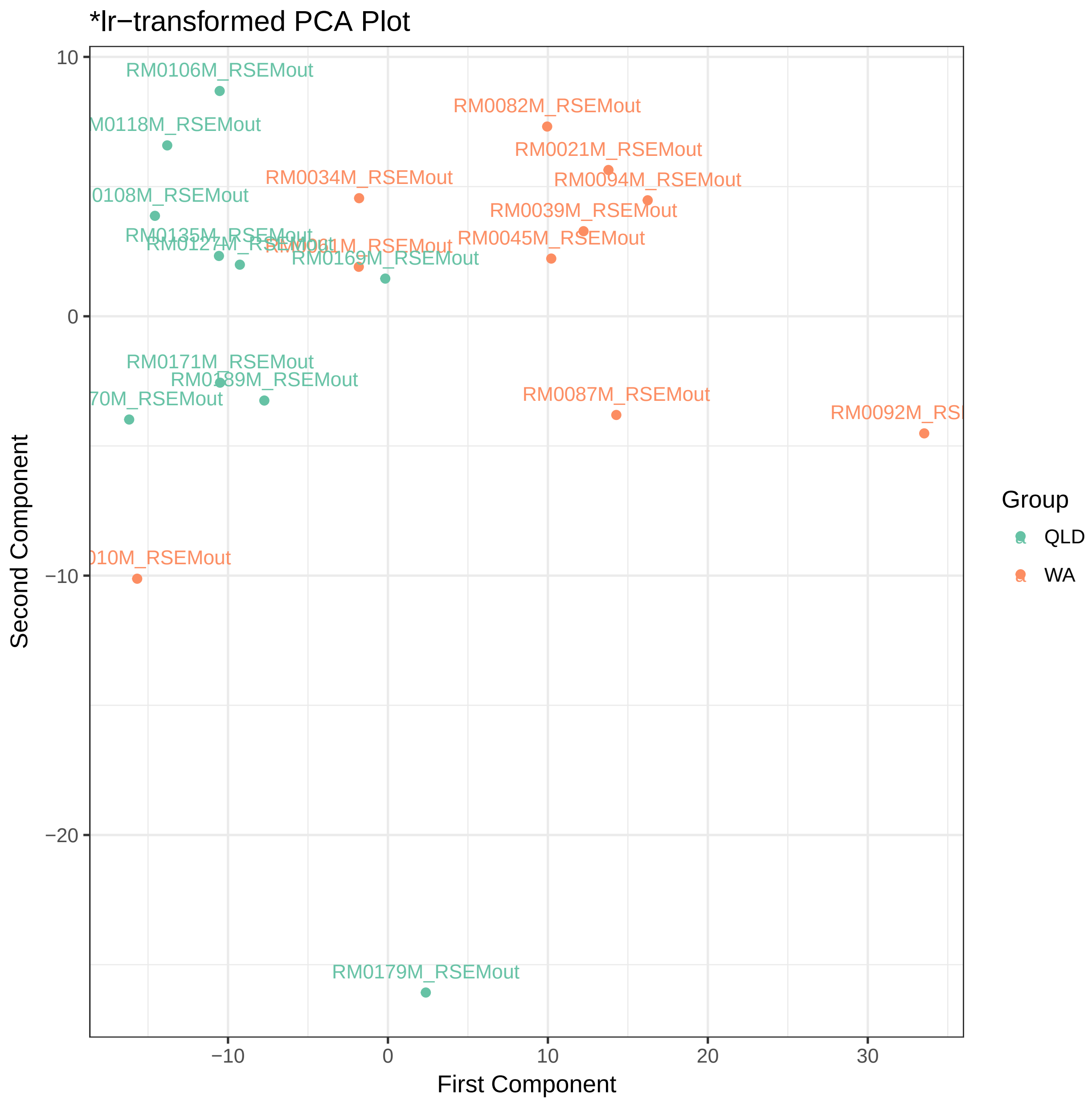


*lr−transformed MDS Plot



We have to butcher `propr::mds()` a bit to create a function to do the PCA projection of the ϕ_s matrix:

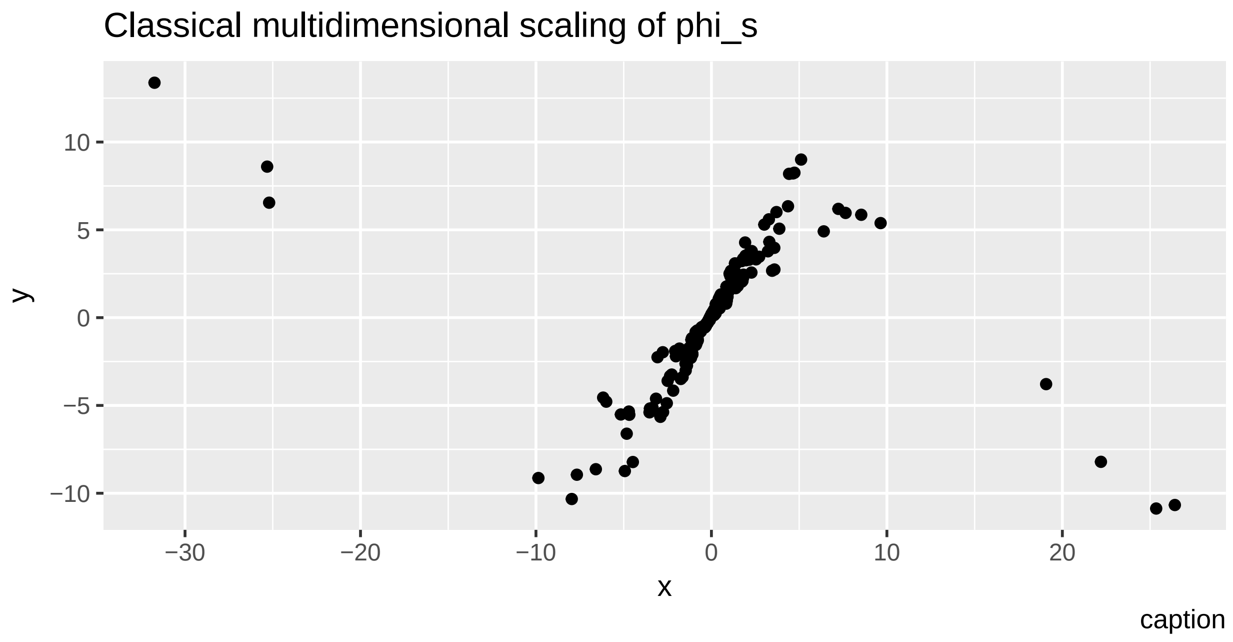
```r
pca <- function(object, group){
  if (missing(group)) {
    group <- rep("None", nrow(object@logratio))
  }
  df <- data.frame(
    ID = rownames(object@logratio),
    Group = as.character(group),
    prcomp(object@logratio)$x[, c(1, 2)])
  g <- ggplot2::ggplot(ggplot2::aes_string(ID = "ID"), data = df) +
    ggplot2::geom_point(ggplot2::aes_string(x = "PC1", y = "PC2",
                                            colour = "Group")) +
    ggplot2::theme_bw() +
    ggplot2::xlab("First Component") +
    ggplot2::ylab("Second Component") +
    ggplot2::scale_colour_brewer(palette = "Set2", name = "Group") +
    ggplot2::ggtitle("*lr-transformed PCA Plot") +
    ggplot2::geom_text(
      ggplot2::aes_string(x = "PC1", y = "PC2", label = "ID", colour = "Group"),
      data = df, size = 3, vjust = -1)
  return(g)
}

pca(top.phi.s, group=caneToad.groups)
```


*lr−transformed PCA Plot
RM0106M_RSEMout
M0118M_RSEMout
0108M_RSEMout
RM0135M_RSEMout
RM0127M_RSEMout
RM0061M_RSEMout
RM0169M_RSEMout
RM0034M_RSEMout
RM0082M_RSEMout
RM0021M_RSEMout
RM0094M_RSEMout
RM0039M_RSEMout
RM0045M_RSEMout
RM0171M_RSEMout
RM0189M_RSEMout
70M_RSEMout
RM0087M_RSEMout
RM0092M_RS
010M_RSEMout
RM0179M_RSEMout
10
0
−10
−20
−10
0
10
20
30
First Component
Second Component
Group
QLD
WA

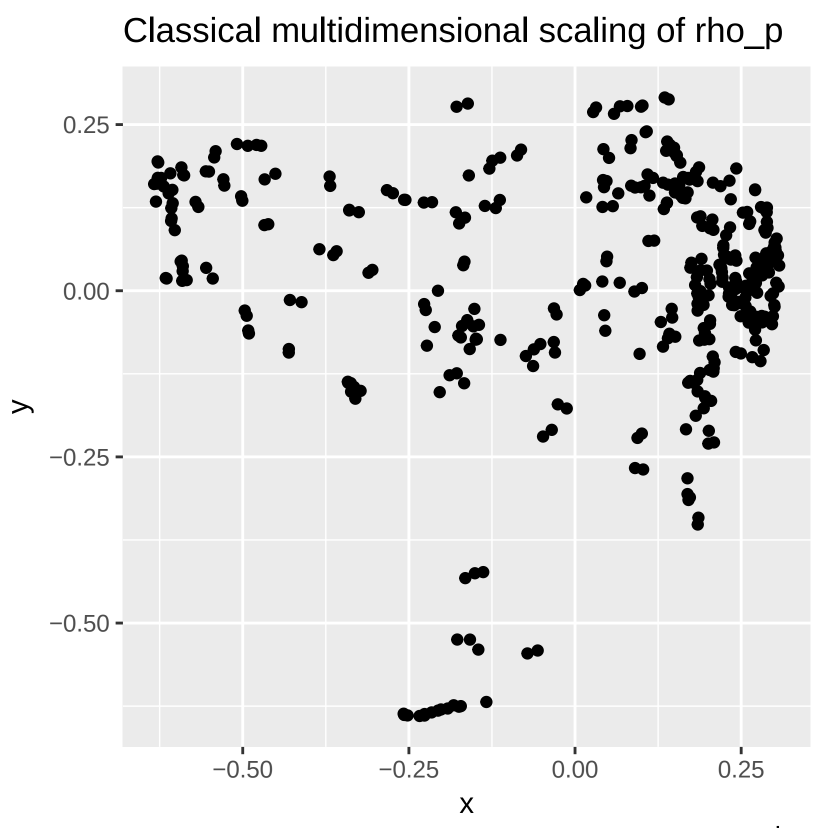
Multidimensional scaling

Here we use `cmdscale()` to perform classical multidimensional scaling (MDS) (also known as principal coordinates analysis (Gower, 1966)) of the ϕ_s matrix:

```r
fit <- cmdscale(top.phi.s@matrix, eig=TRUE, k=2)
fit.df <- data.frame(x=fit$points[,1], y=fit$points[,2], mRNA=colnames(top.phi.s@counts))
ggplot(fit.df, aes(x=x, y=y)) +
  geom_point() +
  coord_equal() +
  labs(title="Classical multidimensional scaling of phi_s", caption="caption")
```



```r
fit <- cmdscale(1-abs(top.rho.p@matrix), eig=TRUE, k=2)
fit.df <- data.frame(x=fit$points[,1], y=fit$points[,2], mRNA=colnames(top.rho.p@counts))
ggplot(fit.df, aes(x=x, y=y)) +
  geom_point() +
  coord_equal() +
  labs(title="Classical multidimensional scaling of rho_p", caption="caption")
```



caption

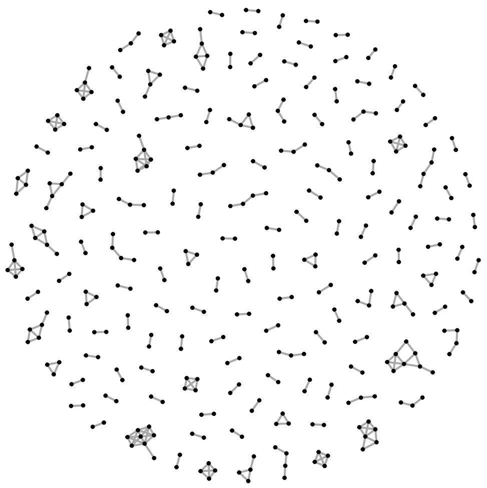
Graph layout

I'm going to use the output of `phiDF()` to illustrate how I would do a graph layout of the strongly proportional mRNAs.

```
counts.phiDF <- phiDF(top.phi.s@counts)
```

This next graph connects all the mRNA pairs that are strongly proportional with an edge.

```
require(igraph)
g <- graph.data.frame(
  subset(counts.phiDF, phisym <  (1 - 0.995)/(1 + 0.995)),
  directed=FALSE)
plot(
  g,
  layout=layout.fruchterman.reingold(g, weight=1/E(g)$phisym),
  vertex.size=1,
  vertex.color="black",
  vertex.label=NA
  )
```



We can see that there are duos, trios, quartets and a few larger sets of mRNAs that behave proportionally across the 20 different individuals.