01-02_all-ref-dist_k-4

Sean Maden

2022-10-24

This vignette shows summary statistics for the restrictive example cell type labels (e.g. K = 4) corresponding to the cell type labels from the TREG paper.

Get cell data and labels

Load data

Append glial cell label

```
sef[[gvarname]] <- ifelse(sef[[kvarname]] %in% glialv, TRUE, FALSE)
```

Cell abundance summaries

Summaries of the cell type labels contained in variables “celltype.treg” and “glial.”

Table of type abundance

```
knitr::kable(table(x = sef[[kvarname]]), align = "c")
```

x	Freq
Excit	24809
Inhib	11067
Oligo	32051
other	9677

Table of type, glial status

```
knitr::kable(table(sef[[gvarname]]), align = "c")
```

Var1	Freq
FALSE	45553
TRUE	32051

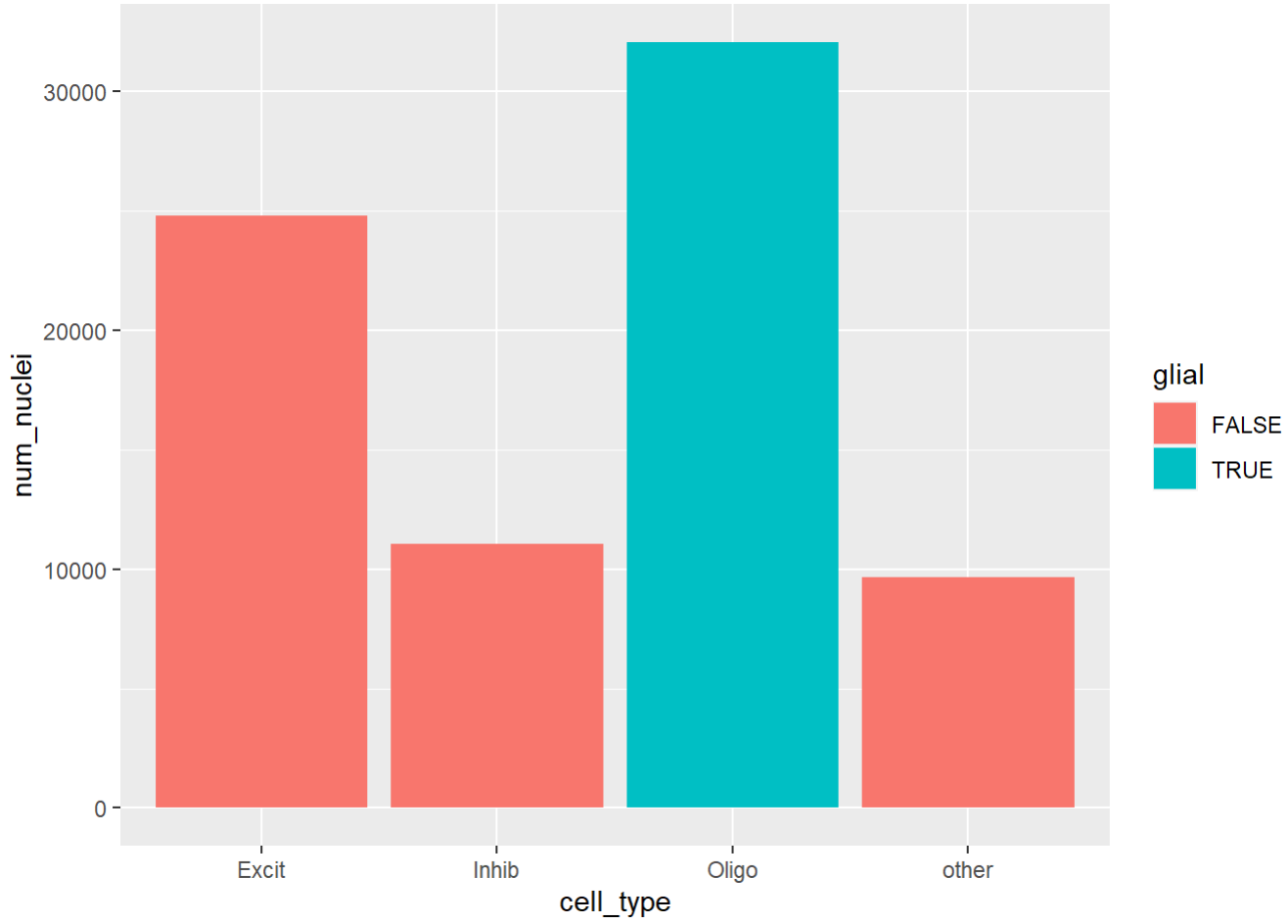
```
knitr::kable(table(sef[[kvarname]], sef[[gvarname]]), align = "c")
```

	FALSE	TRUE
Excit	24809	0
Inhib	11067	0
Oligo	0	32051
other	9677	0

Type abundances barplot

```
dfp <- as.data.frame(table(sef[[kvarname]], sef[[gvarname]]))
colnames(dfp) <- c("cell_type", "glial", "num_nuclei")
dfp[,1] <- as.character(dfp[,1])
dfp[,2] <- as.character(dfp[,2])
```

```
ggplot(dfp, aes(x = cell_type, y = num_nuclei, fill = glial)) + geom_bar(stat = "identity")
```



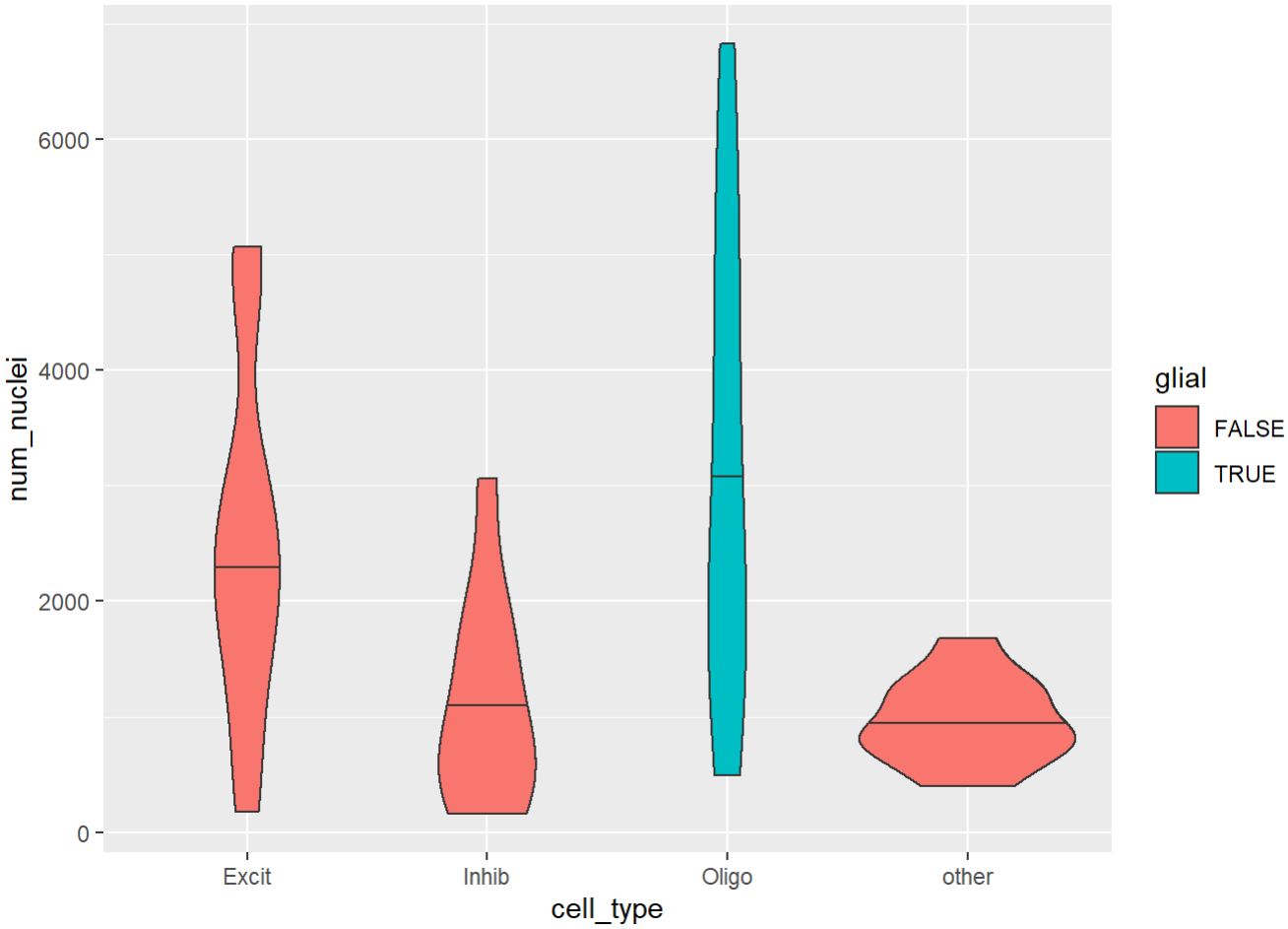
Abundances by donor

Cell nuclei abundances/counts, grouped by donor variable “BrNum”, or the donor brain ID.

Type abundances violin plots

```
cd <- colData(sef)
dv <- unique(sef[[dvarname]])
dfp <- do.call(rbind, lapply(dv, function(di){
  cdi <- cd[cd[,dvarname]==di,]
  dfp <- as.data.frame(table(cdi[,kvarname]))
  dfp$donor <- di
  dfp
}))
colnames(dfp) <- c("cell_type", "num_nuclei", "donor")
dfp$glial <- ifelse(dfp$cell_type %in% c("Oligo"), T, F)
```

```
ggplot(dfp, aes(x = cell_type, y = num_nuclei, fill = glial)) +
  geom_violin(draw_quantiles = 0.5)
```

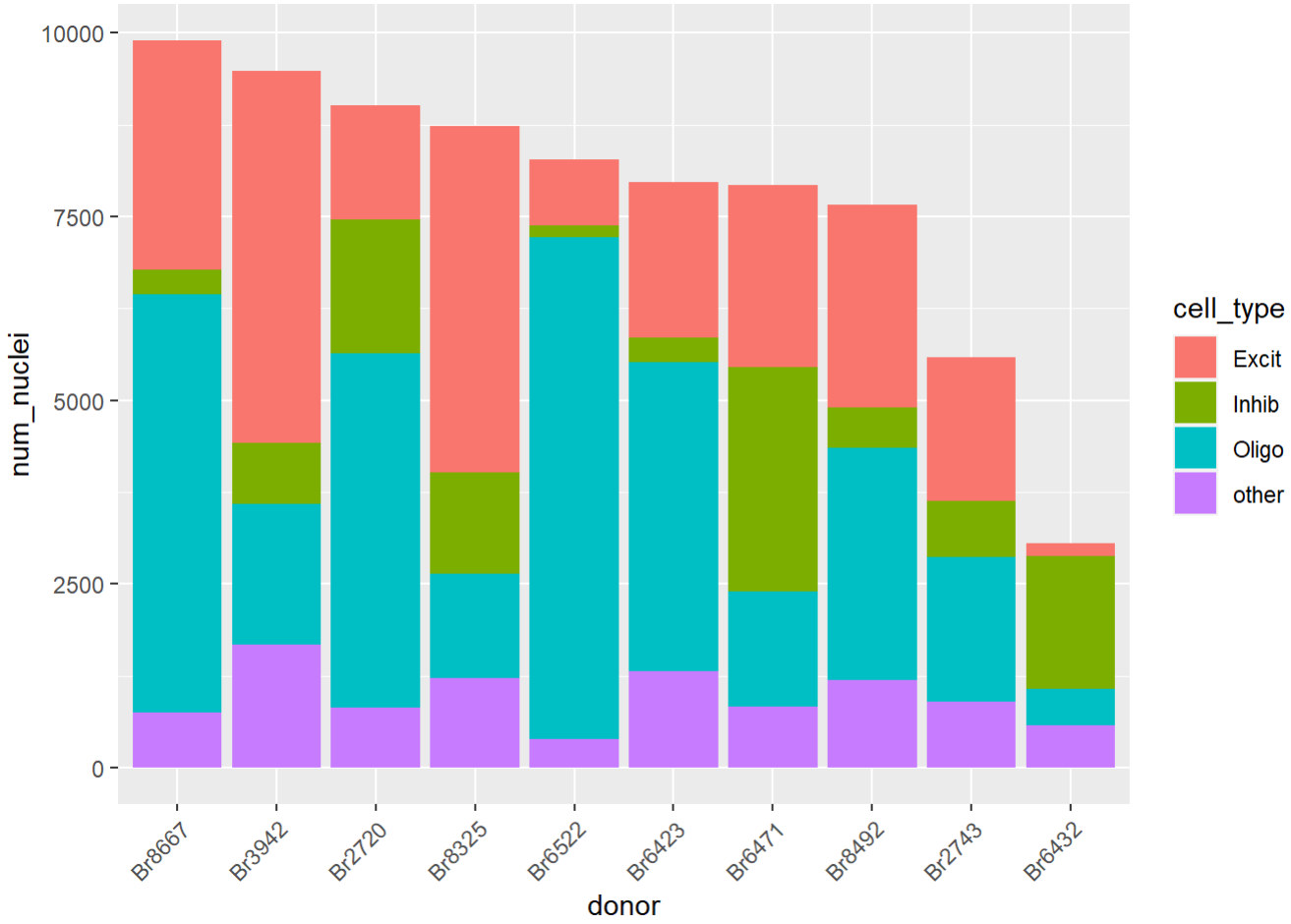


Type abundances stacked barplots

```
# order donor var
sumv <- as.data.frame(table(sef[[dvarname]]))
dfp$donor <- factor(dfp$donor, levels = sumv[,1][rev(order(sumv[,2]))])
```

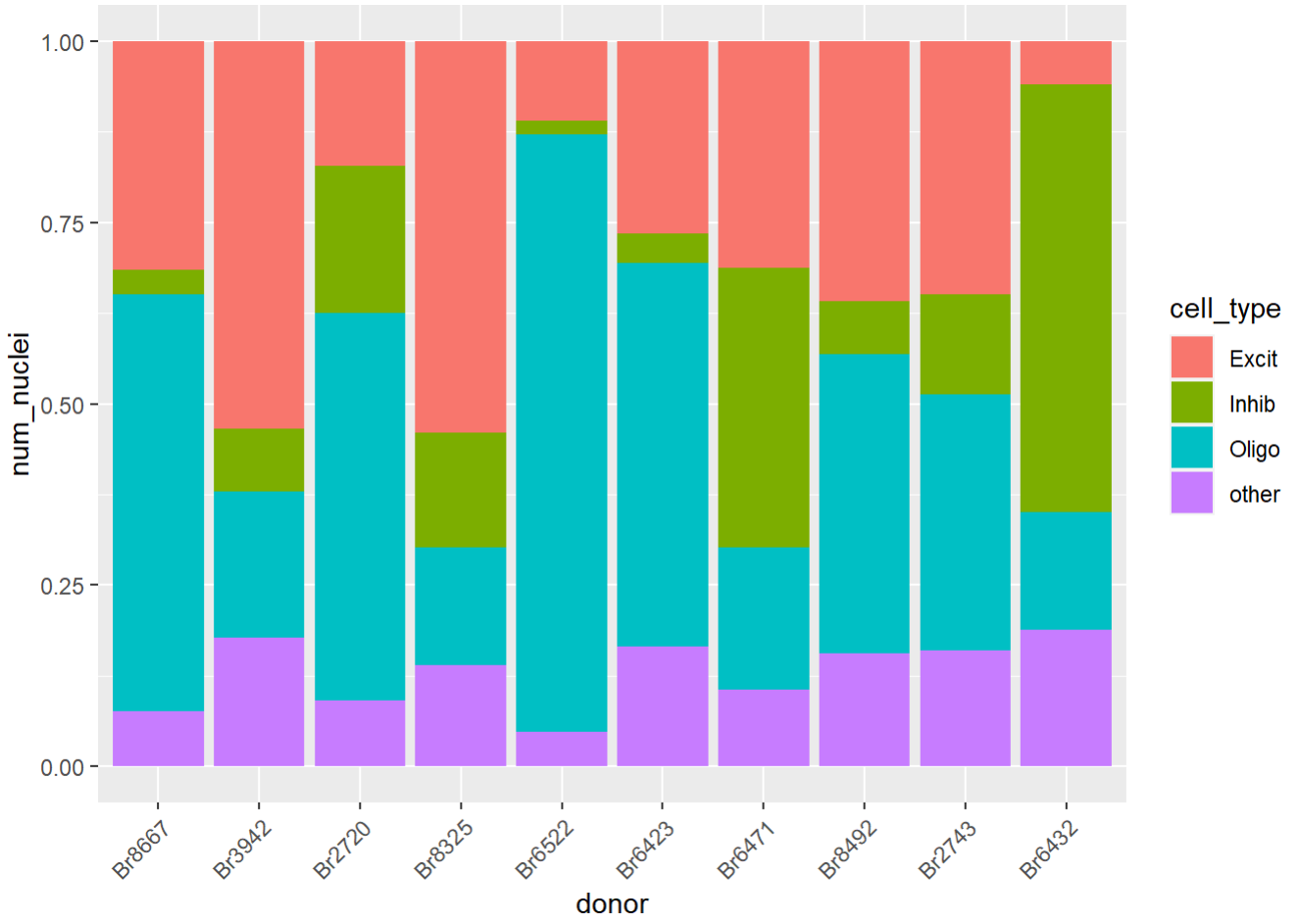
Counts

```
ggplot(dfp, aes(x = donor, y = num_nuclei, fill = cell_type)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Percentages

```
ggplot(dfp, aes(x = donor, y = num_nuclei, fill = cell_type)) +
  geom_bar(position = "fill", stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

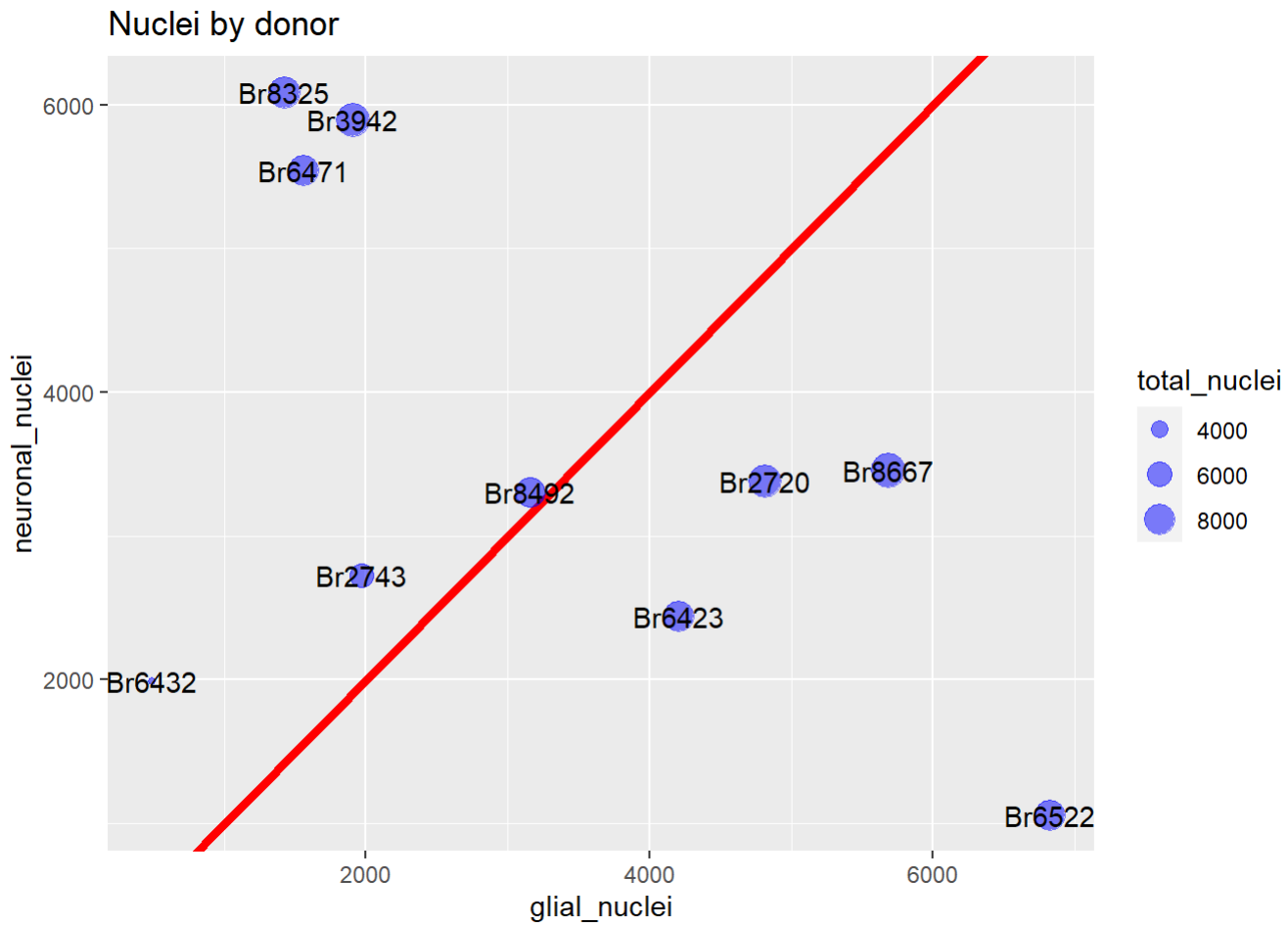


Scatterplot of glial versus neurons, by donor

```
dfp <- do.call(rbind, lapply(dv, function(donori){
  sefi <- sef[,sef[[dvarname]]==donori]
  data.frame(glial_nuclei = length(which(sefi[[gvarname]]==T)),
             neuronal_nuclei = length(which(
               sefi[[kvarname]] %in% c("Inhib", "Excit"))),
             total_nuclei = ncol(sefi), donor = donori)
}))
```

```
ggplot(dfp, aes(x = glial_nuclei, y = neuronal_nuclei)) +
  geom_abline(intercept = 0, slope = 1, col = "red", lwd = 1.5) + ggtitle("Nuclei by donor") +
  geom_point(aes(size = total_nuclei), alpha = 0.5, color = "blue") +
  geom_text(aes(label = donor), position = "dodge", color = "black")
```

```
## Warning: Width not defined. Set with `position_dodge(width = ?)`
```



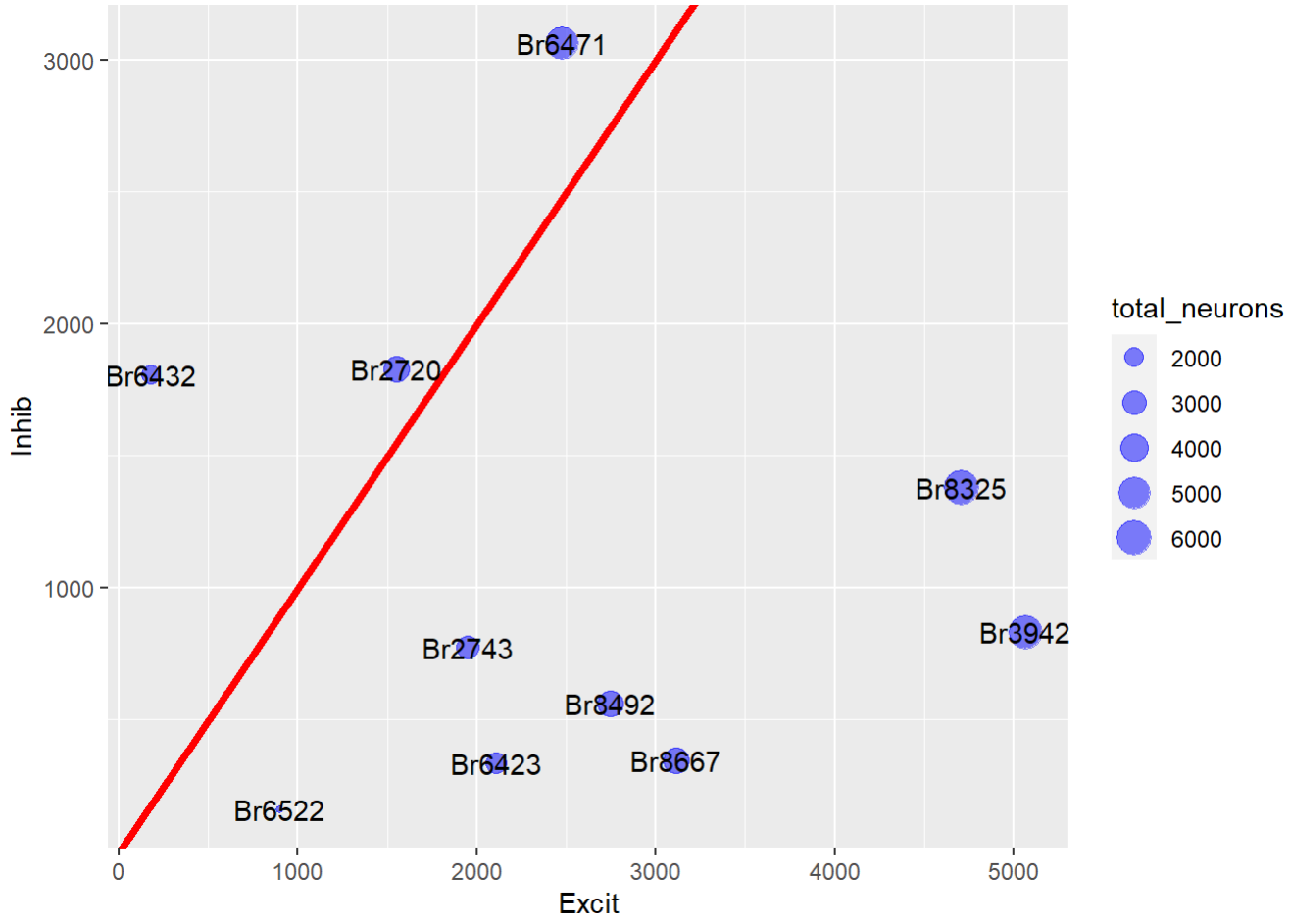
Scatterplot of neuronal types

Plotting Excit versus Inhib neuron abundances.

```
sefi <- sef[,sef[[kvarname]] %in% c("Inhib", "Excit")]
dfp <- as.data.frame(table(sefi[[kvarname]], sefi[[dvarname]]))
dfp <- cbind(dfp[dfp[,1]=="Excit",], dfp[dfp[,1]=="Inhib",])[,c(2,3,6)]
colnames(dfp) <- c("donor", "Excit", "Inhib")
dfp$total_neurons <- dfp$Excit+dfp$Inhib
```

```
ggplot(dfp, aes(x = Excit, y = Inhib)) +
  geom_point(aes(size = total_neurons), alpha = 0.5, color = "blue") +
  geom_abline(intercept = 0, slope = 1, col = "red", lwd = 1.5) +
  geom_text(aes(label = donor), position = "dodge", color = "black")
```

```
## Warning: Width not defined. Set with `position_dodge(width = ?)`
```

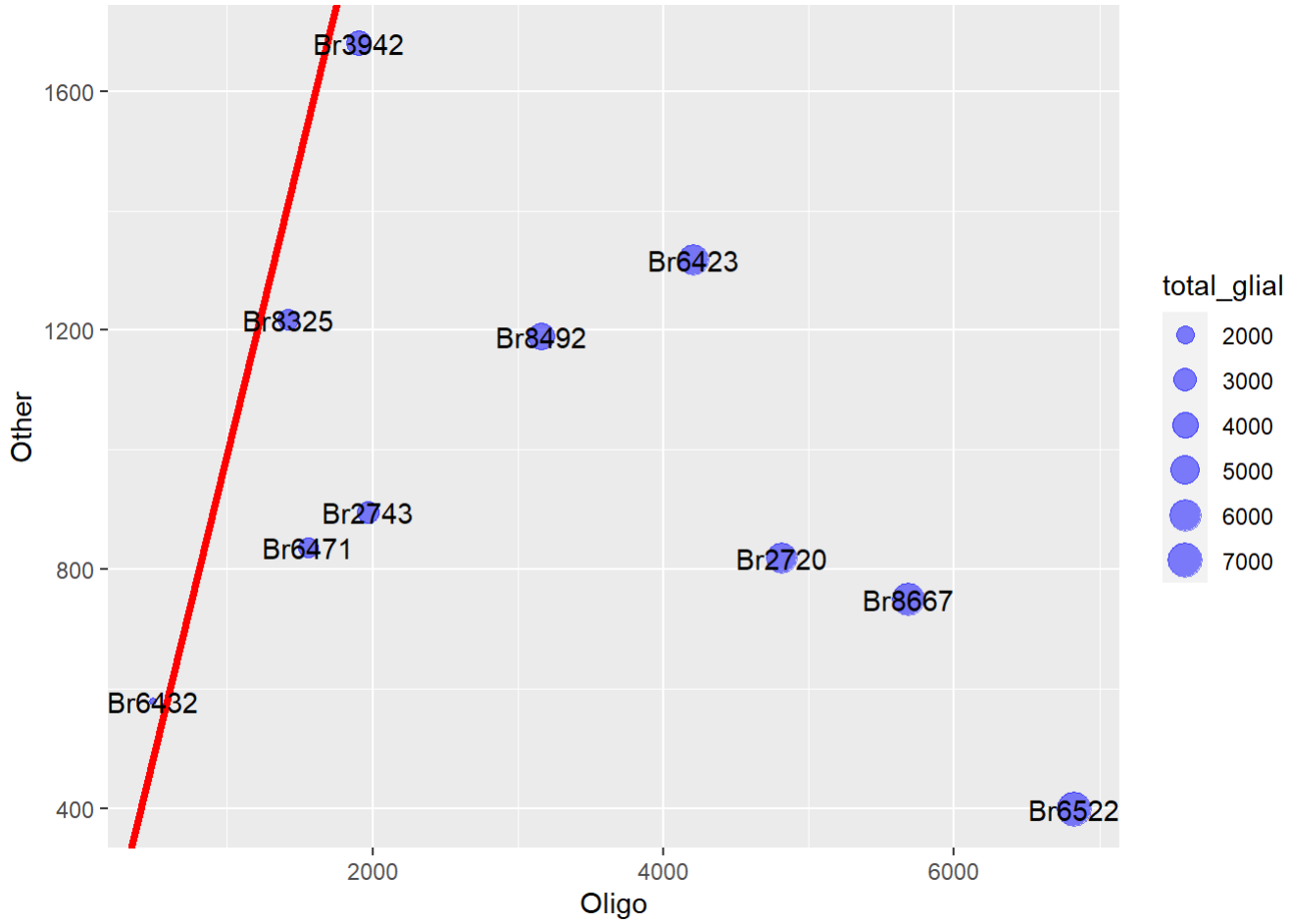


Scatterplot of Oligo versus non-Oligo cell types

```
sefi <- sef[,!sef[[kvarname]] %in% c("Inhib", "Excit")]
dfp <- as.data.frame(table(sefi[[kvarname]], sefi[[dvarname]]))
dfp <- cbind(dfp[dfp[,1]=="Oligo",], dfp[dfp[,1]=="other",])[,c(2,3,6)]
colnames(dfp) <- c("donor", "Oligo", "Other")
dfp$total_glial <- dfp$Oligo+dfp$Other
```

```
ggplot(dfp, aes(x = Oligo, y = Other)) +
  geom_point(aes(size = total_glial), alpha = 0.5, color = "blue") +
  geom_abline(intercept = 0, slope = 1, col = "red", lwd = 1.5) +
  geom_text(aes(label = donor), position = "dodge", color = "black")
```

```
## Warning: Width not defined. Set with `position_dodge(width = ?)`
```



Marker genes plots

Upset plot of marker gene identities

Expression at marker genes

Composite violin plots

Make composite plots with `grid.arrange` .

Get expression distributions across variable scales, grouped by cell types.

```
# expression data
ct <- assays(sef)$counts
sf <- 2^rnorm(ncol(sef))
sf <- sf/mean(sf)
nc <- t(t(ct)/sf)
lc <- log2(nc+1)

# get mean ratio expression -- by donor,type
knamev <- unique(sef[[kvarname]])
mr.kd <- do.call(cbind, lapply(knamev, function(ki){
  sefi <- sef[,sef[[kvarname]]==ki] # type subset
  dfpi <- do.call(cbind, lapply(dv, function(donori){
    sefii <- sefi[,sefi[[dvarname]]==donori] # donor, type subset
    numerv <- rowMeans(assays(sefii[,sefii[[kvarname]]==ki])$counts)
    ldenom <- lapply(knamev[!knamev==ki], function(kj){
      filtj <- sef[[kvarname]]==kj & sef[[dvarname]]==donori
      rowMeans(assays(sef[,filtj])$counts)+1e-6
    })
    mrkdv <- rep(numerv, length(knamev)-1)/unlist(ldenom)
    rowMeans(matrix(mrkdv, nrow = length(numerv)))
  }))
  colnames(dfpi) <- paste0(dv, ";", ki)
  rownames(dfpi) <- rownames(sef)
  dfpi
}))
dfp.mrkd <- reshape::melt(mr.kd)
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

```
colnames(dfp.mrkd) <- c("gene_name", "donor;type", "value")
dfp.mrkd$cell_type <- gsub(".*;", "", dfp.mrkd$`donor;type`)
dfp.mrkd <- dfp.mrkd[,c(1,4,3)]

# get mean ratio expression -- by type
mr.k <- do.call(cbind, lapply(knamev, function(ki){
  which.ki <- grepl(ki, colnames(mr.kd)); rowMeans(mr.kd[,which.ki])
}))
colnames(mr.k) <- knamev
rownames(mr.k) <- rownames(mr.kd)
dfp.mrk <- reshape::melt(mr.k)
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

```
colnames(dfp.mrk) <- c("gene_name", "cell_type", "value")
```

```
# get violin plots of means
lmc <- get_lggvp(ct, yaxis.name = "mean_counts", xaxis.text = F, kvarname = kvarname) # counts
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

```
lnc <- get_lggvp(nc, yaxis.name ="mean_normcounts", xaxis.text = F, kvarname = kvarname) # norm
 counts
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

```
llc <- get_lggvp(lc, yaxis.name = "mean_logcounts", xaxis.text = F, kvarname = kvarname) # log c
ounts
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

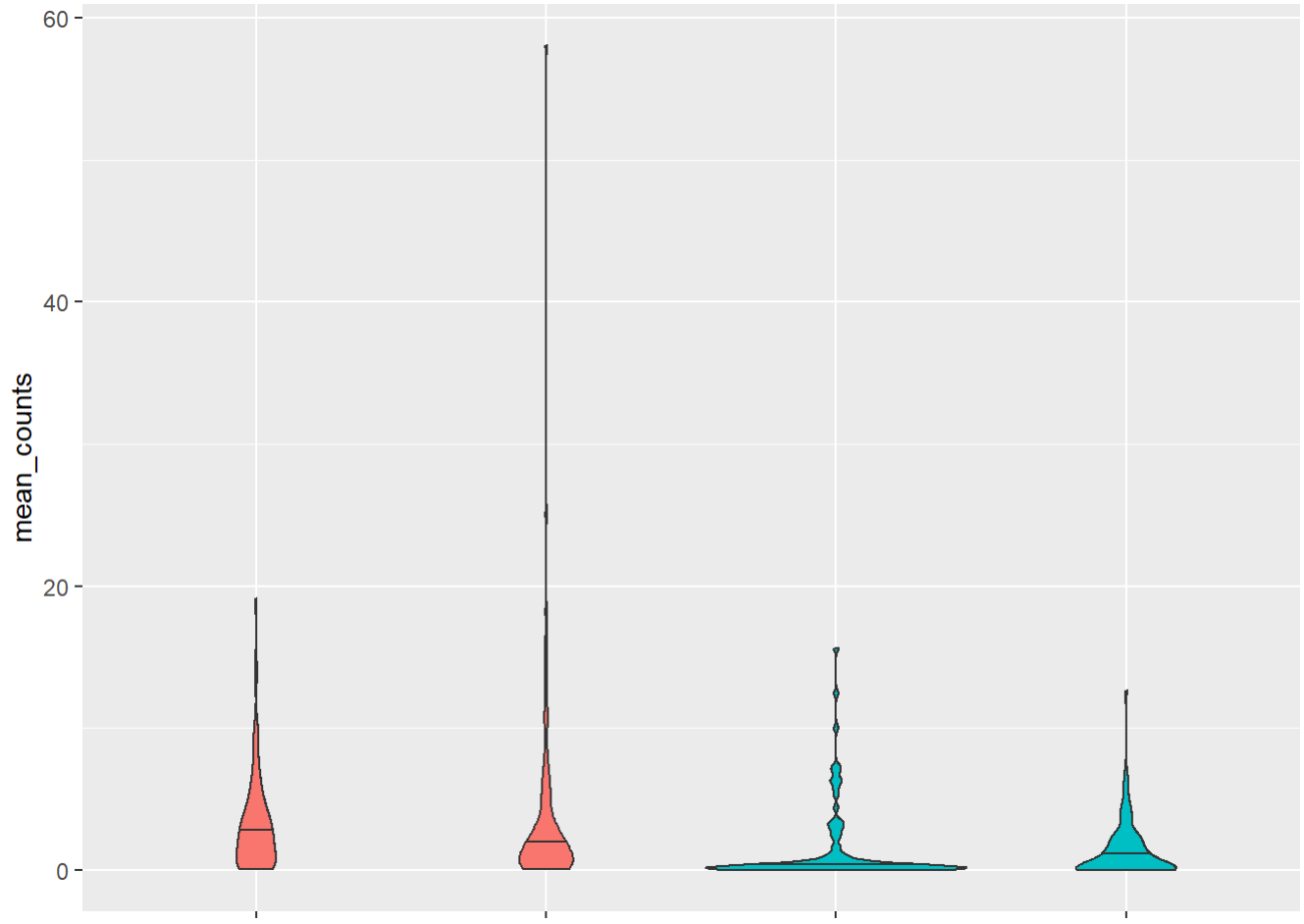
```
# mr by k,d
lmrkd <- get_lggvp(dfp.mrkd, make.dfp = F, yaxis.name = "mean_ratio_kd",
                   xaxis.text = T, kvarname = kvarname)
# mr by k
lmrk <- get_lggvp(dfp.mrk, make.dfp = F, yaxis.name = "mean_ratio_k",
                  xaxis.text = T, kvarname = kvarname)
```

Individual violin plots

Counts

```
lmc$ggvp
```

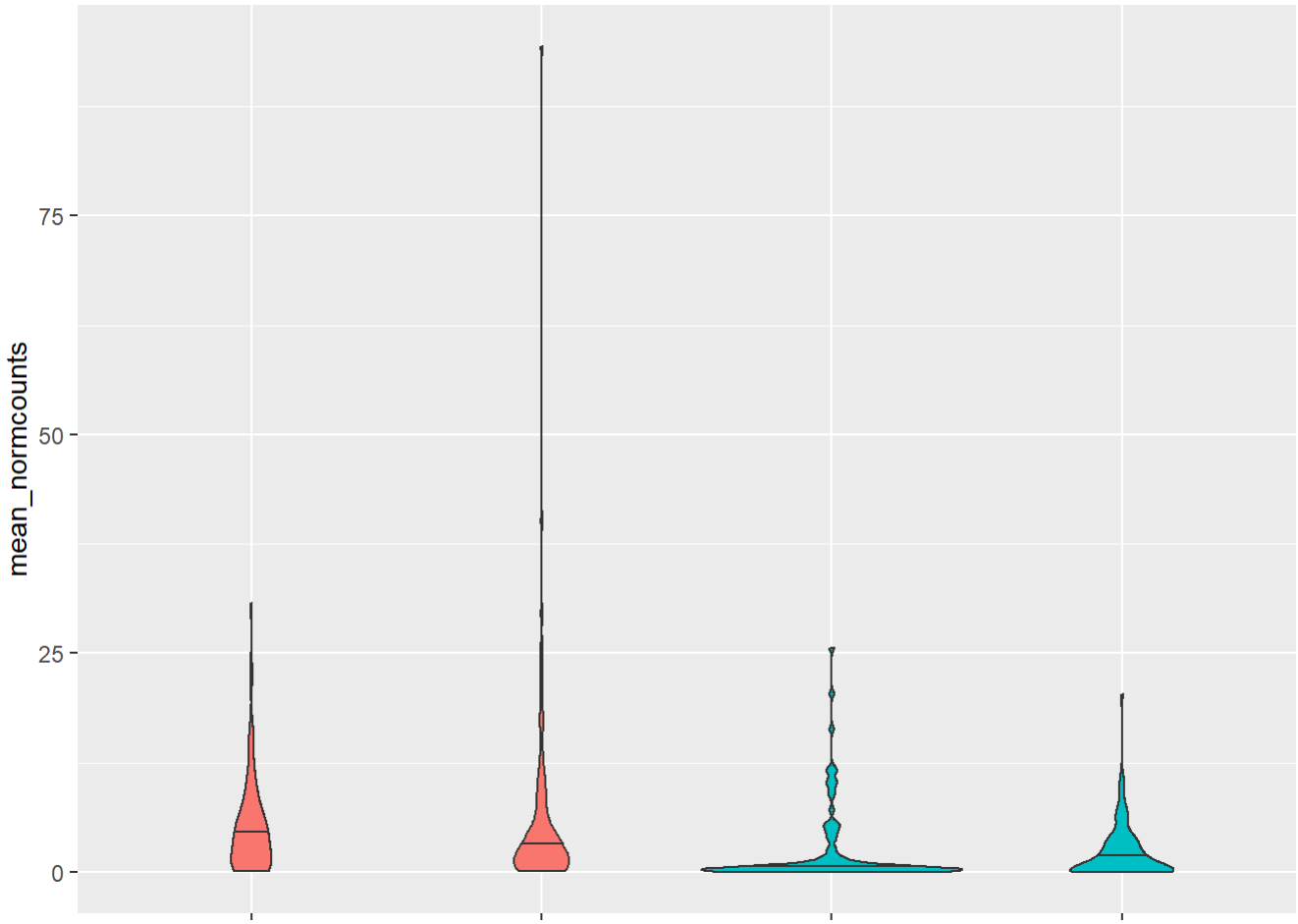
```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



Normalized counts

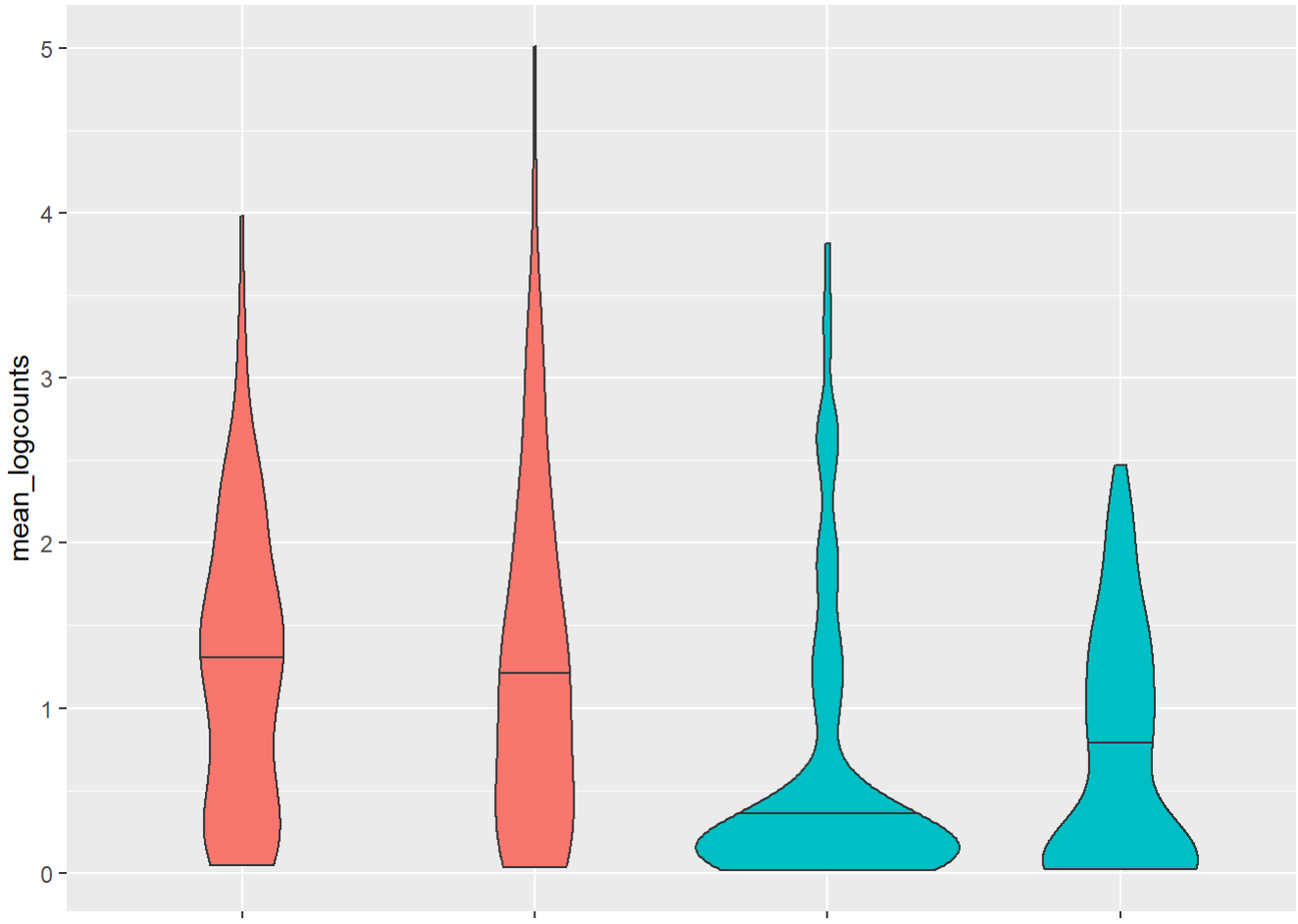
```
lnc$ggvp
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



Log counts

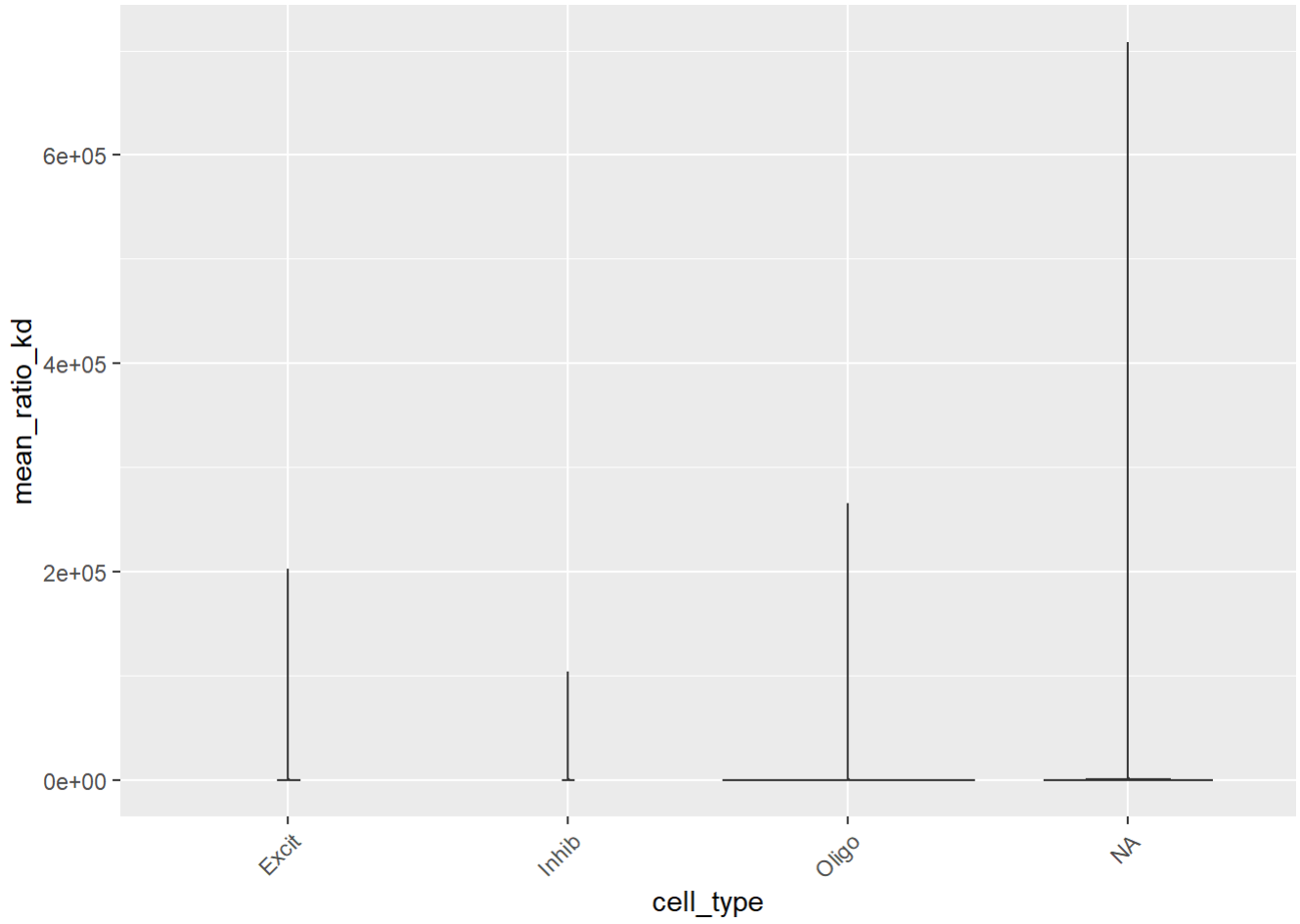
```
llc$ggvp
```



Mean ratio by cell type (k) and donor (d), e.g. “mr_kd”

```
lmrkd$ggvp
```

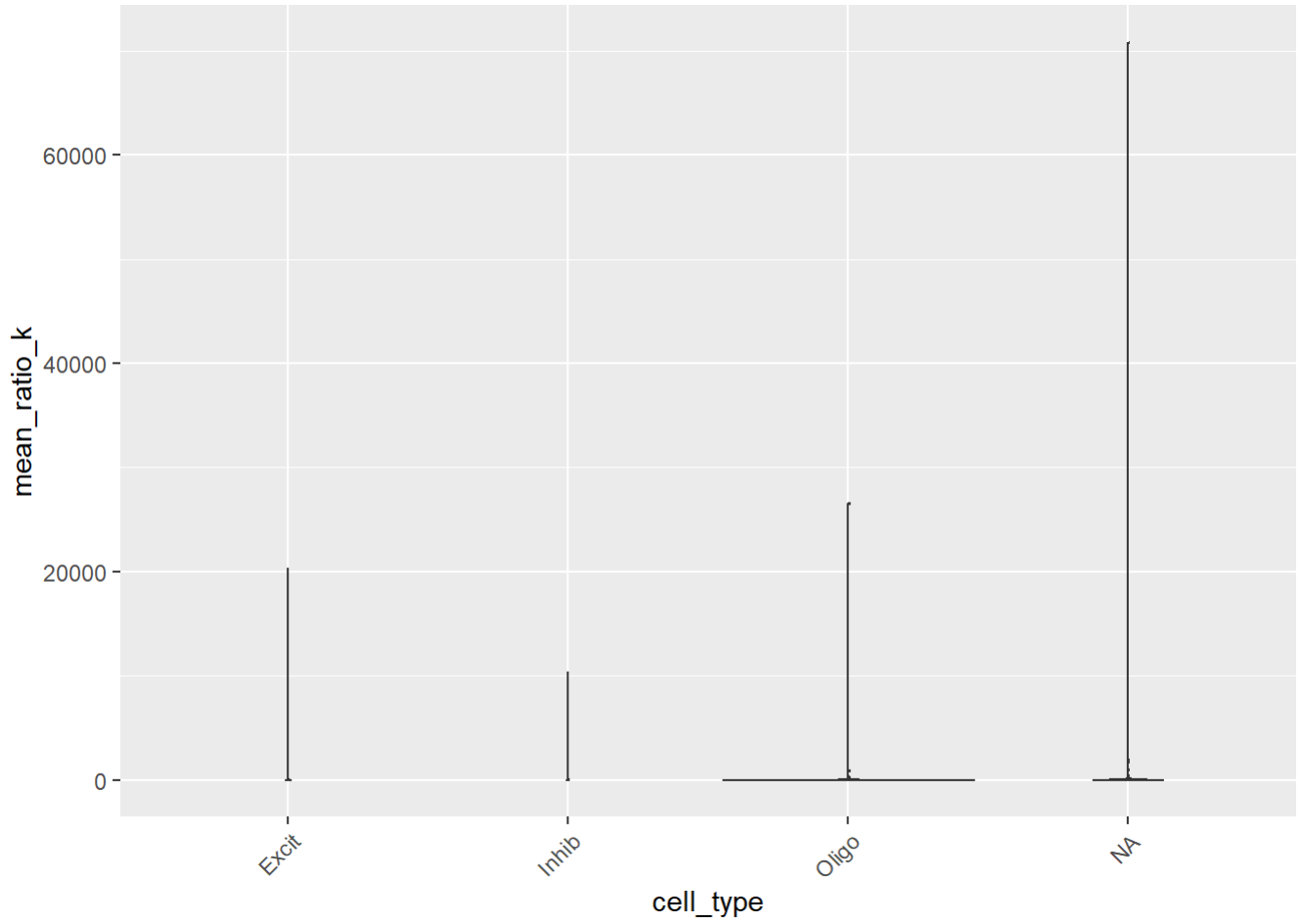
```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



Mean ratio by cell type (k)

```
lmrk$ggvp
```

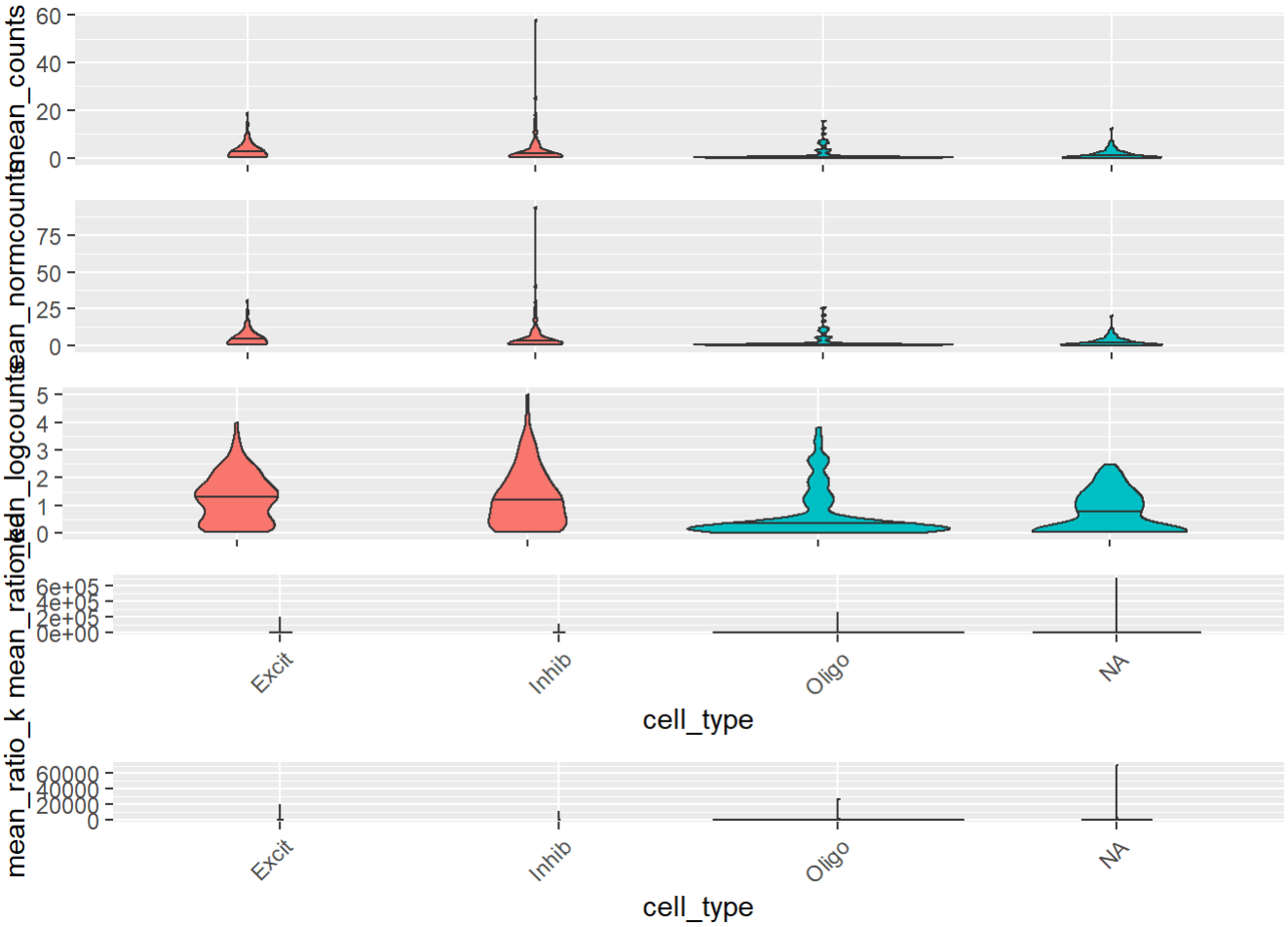
```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



Composite – all scales

```
# make composite plot
grid.arrange(lmc$ggvp, lnc$ggvp, llc$ggvp, lmrkd$ggvp, lmrk$ggvp, ncol = 1)
```

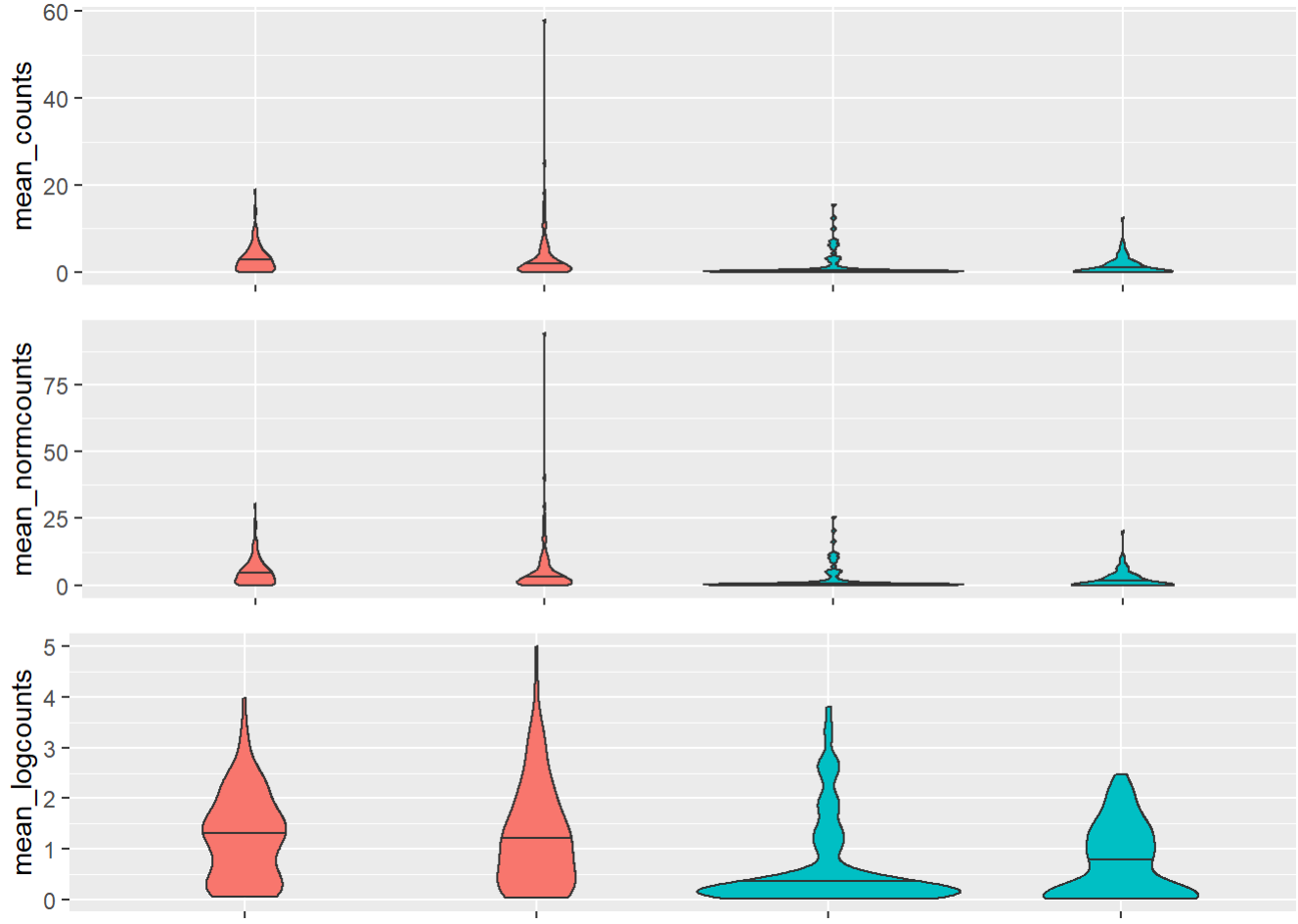
```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



Composite – counts without mean ratios

```
grid.arrange(lmc$ggvp, lnc$ggvp, llc$ggvp, ncol = 1)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



Heatmaps of mean ratios

Prep heatmap params

Make annotations

```
dfm <- as.data.frame(lz$top.marker.data)
dfm <- dfm[order(match(dfm$gene, rownames(sef))),]
cond <- identical(dfm$gene, rownames(sef))
if(!cond){stop("couldn't match gene names.")}
# cell type annotations
rowData(sef)$cell_type <- dfm$celltype.target
rowData(sef)$top_mean_ratio <- dfm$ratio
ra <- rowAnnotation(cell_type = rowData(sef)$cell_type,
                    top_mean_ratio = rowData(sef)$top_mean_ratio)
ca.kd <- columnAnnotation(cell_type = gsub(".*;", "", colnames(mr.kd)))
ca.k <- columnAnnotation(cell_type = gsub(".*;", "", colnames(mr.k)))
```

Make axis labels

```
ylab <- paste0("Genes (G = ",nrow(mr.kd),")")
xlab.kd <- paste0("Donors, types (k;d = ",ncol(mr.kd),")")
xlab.k <- paste0("Type (K = ",ncol(mr.k),")")
```

Prep heatmap data

Unscaled

```
set.seed(0)
hm.kd <- Heatmap(mr.kd, top_annotation = ca.kd,
                 show_row_names = F, show_column_names = F,
                 row_title = ylab, column_title = xlab.kd,
                 name = "MR_kd_unscaled")
```

```
## The automatically generated colors map from the 1^st and 99^th of the
## values in the matrix. There are outliers in the matrix whose patterns
## might be hidden by this color mapping. You can manually set the color
## to `col` argument.
##
## Use `suppressMessages()` to turn off this message.
```

```
hm.k <- Heatmap(mr.k, top_annotation = ca.k, right_annotation = ra,
                show_row_names = F, show_column_names = F,
                row_title = ylab, column_title = xlab.k,
                name = "MR_k_unscaled")
```

```
## The automatically generated colors map from the 1^st and 99^th of the
## values in the matrix. There are outliers in the matrix whose patterns
## might be hidden by this color mapping. You can manually set the color
## to `col` argument.
##
## Use `suppressMessages()` to turn off this message.
```

Scaled

```
set.seed(0)
hm.kds <- Heatmap(scale(mr.kd), top_annotation = ca.kd,
                  show_row_names = F, show_column_names = F,
                  row_title = ylab, column_title = xlab.kd,
                  name = "MR_kd_scaled")
```

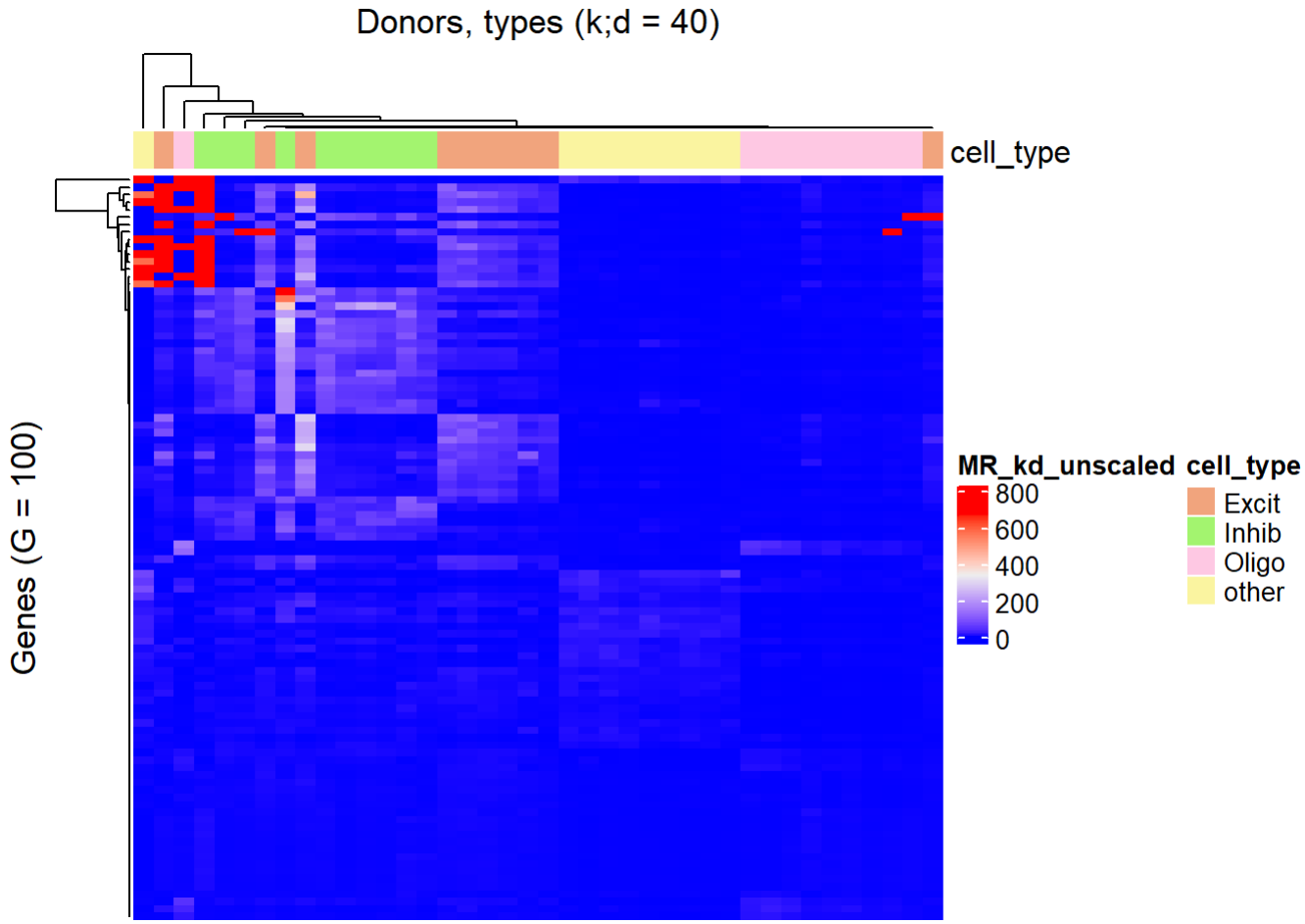
```
## The automatically generated colors map from the 1^st and 99^th of the
## values in the matrix. There are outliers in the matrix whose patterns
## might be hidden by this color mapping. You can manually set the color
## to `col` argument.
##
## Use `suppressMessages()` to turn off this message.
```

```
hm.ks <- Heatmap(scale(mr.k), top_annotation = ca.k, right_annotation = ra,
                 show_row_names = F, show_column_names = F,
                 row_title = ylab, column_title = xlab.k,
                 name = "MR_k_scaled")
```

Plot heatmaps

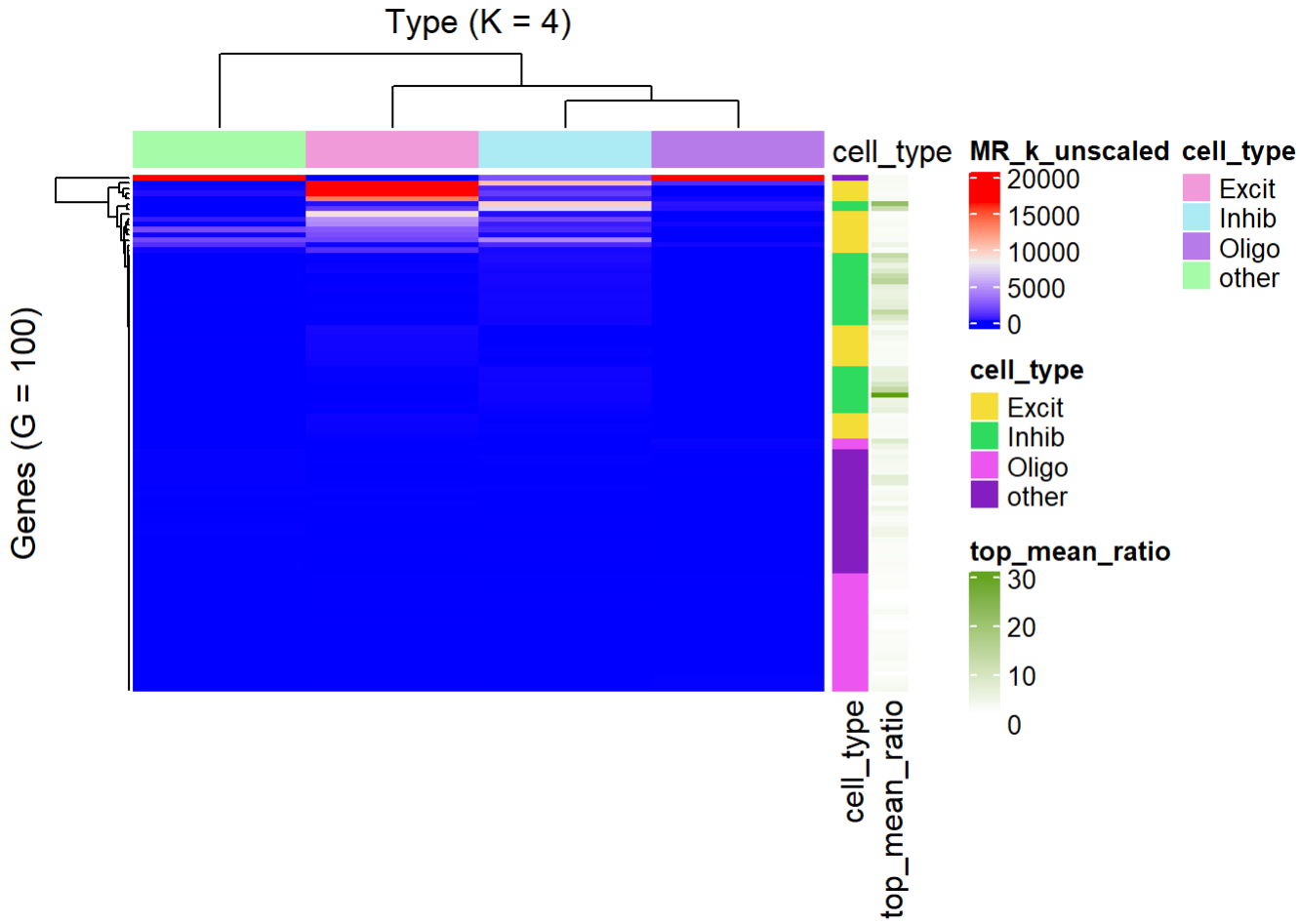
Heatmap of MRKD

```
hm.kd
```



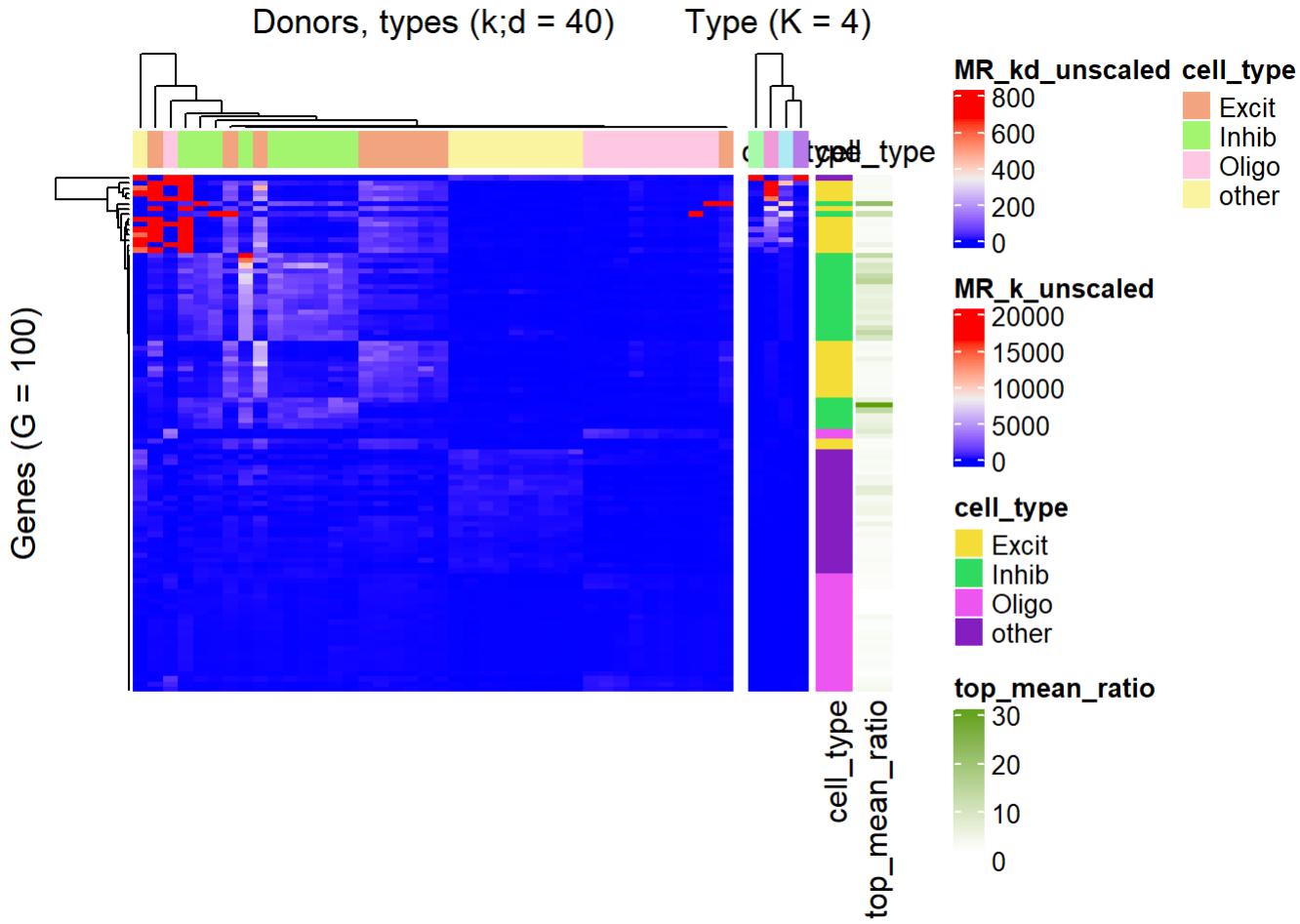
Heatmap of MRK

```
hm.k
```



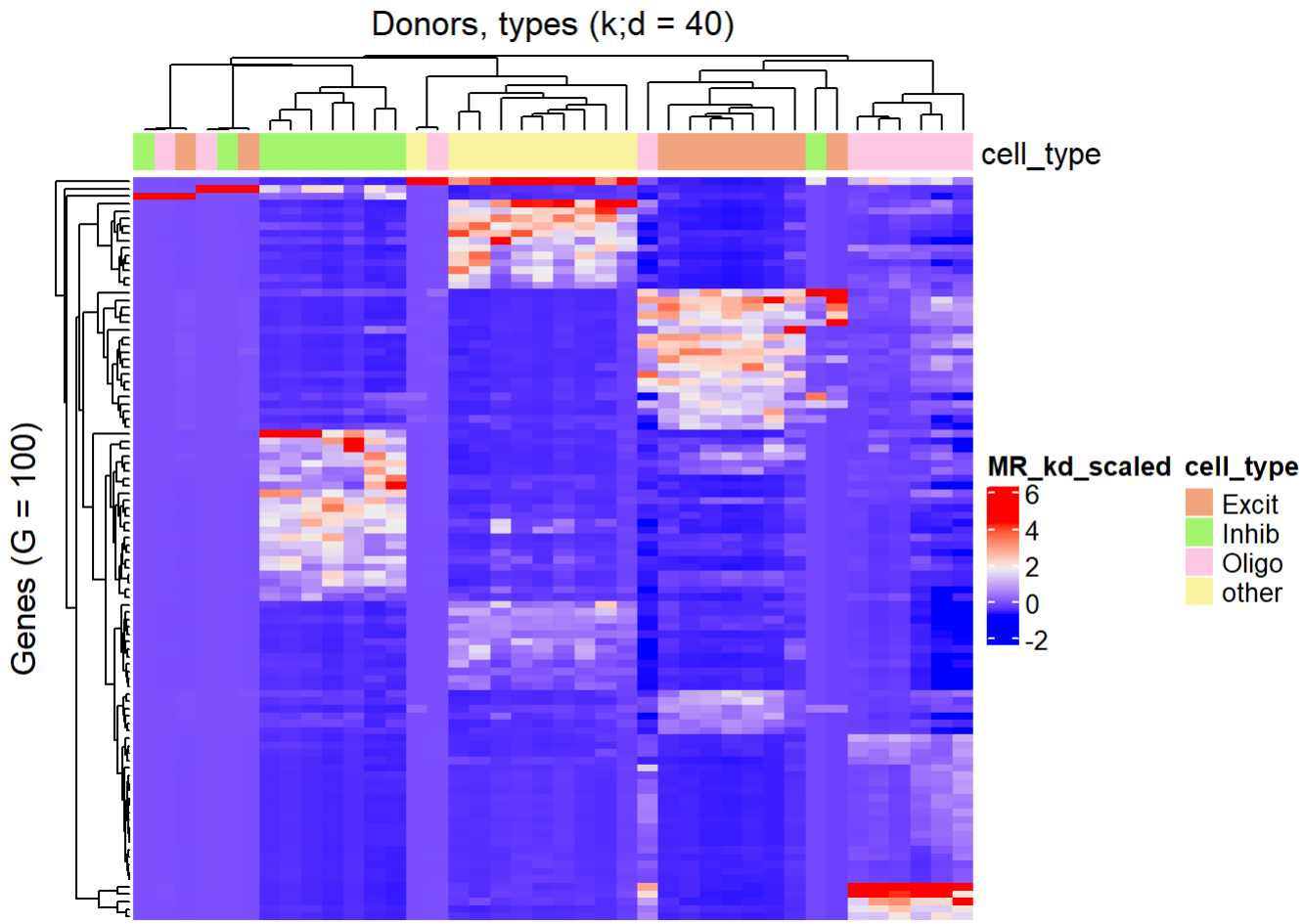
Composite heatmap

```
hm.kd + hm.k
```

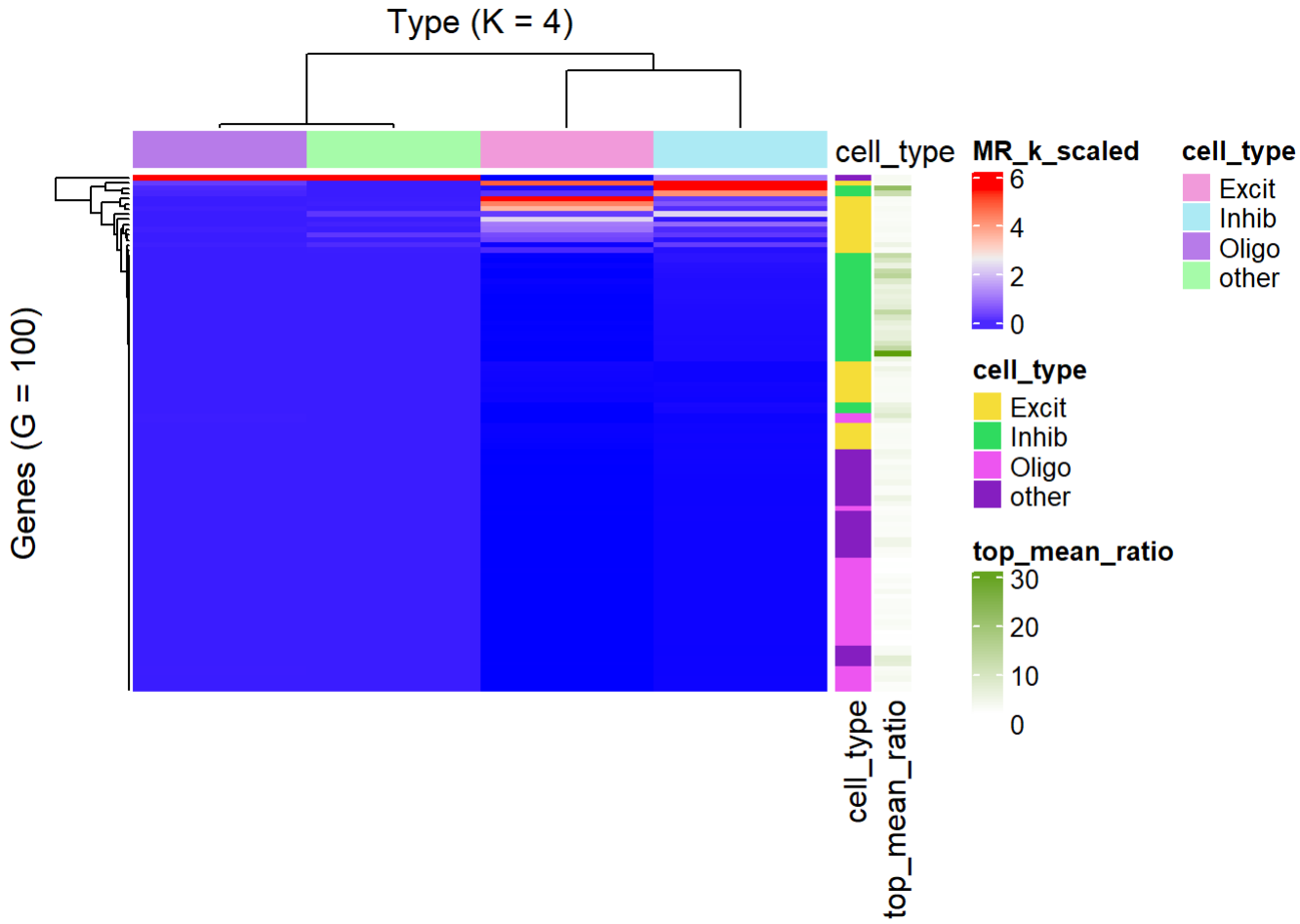


Scaled

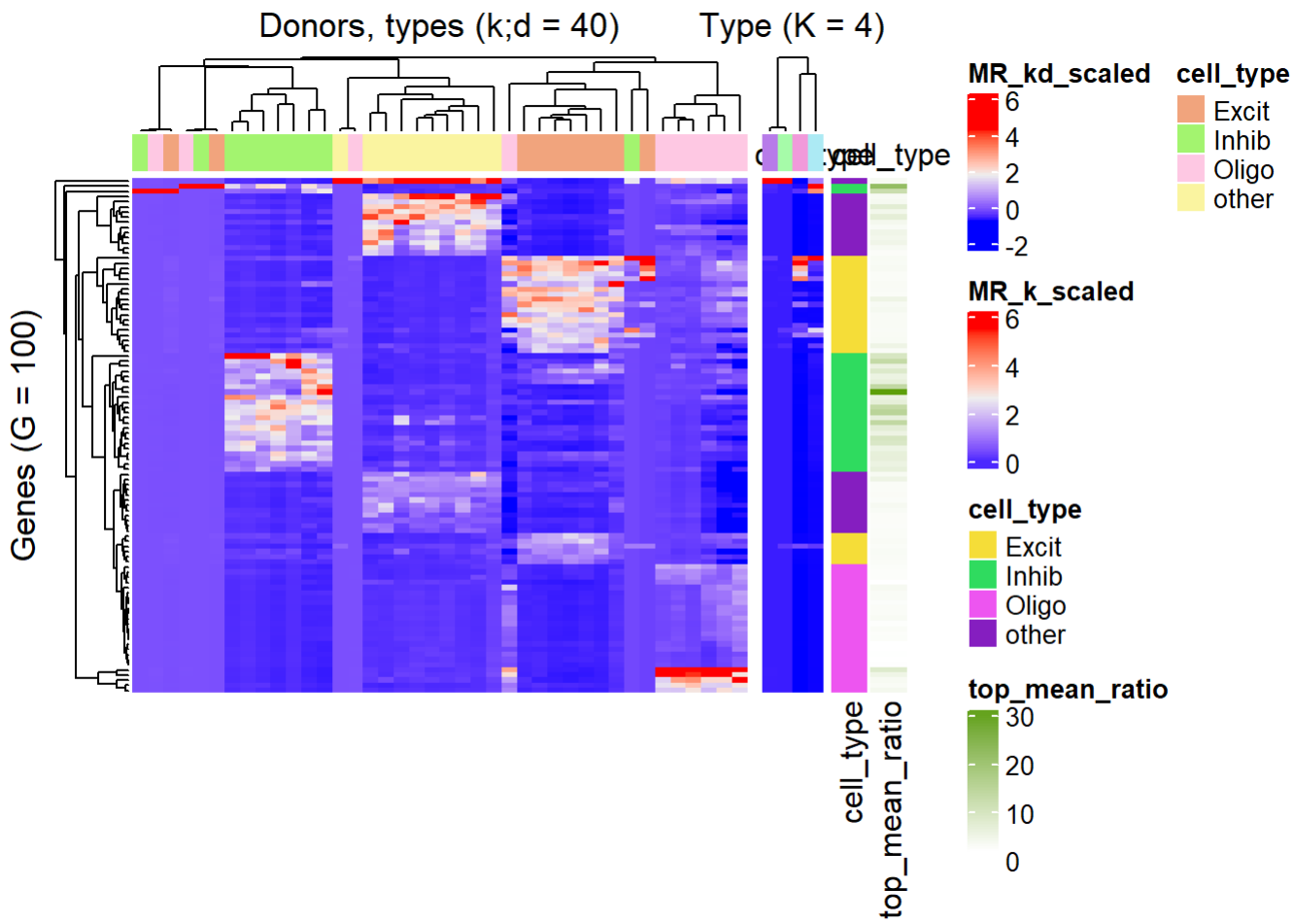
```
hm.kds
```


Donors, types (k;d = 40)
cell_type
Genes (G = 100)
MR_kd_scaled
6
4
2
0
-2
cell_type
Excit
Inhib
Oligo
other

```
hm.ks
```


Type (K = 4)
Genes (G = 100)
cell_type
top_mean_ratio
MR_k_scaled
6
4
2
0
cell_type
Excit
Inhib
Oligo
other
cell_type
Excit
Inhib
Oligo
other
top_mean_ratio
30
20
10
0

```
hm.kds + hm.ks
```


Donors, types (k;d = 40)
Type (K = 4)
Genes (G = 100)
cell_type
top_mean_ratio
MR_kd_scaled
6
4
2
0
-2
cell_type
Excit
Inhib
Oligo
other
MR_k_scaled
6
4
2
0
cell_type
Excit
Inhib
Oligo
other
top_mean_ratio
30
20
10
0