

# Supporting Appendix: Calculation and analysis of Relative Exon Usage Coefficients (REUCs)

Alejandro Reyes, Simon Anders, Robert J. Weatheritt,  
Toby Gibson, Lars M. Steinmetz, Wolfgang Huber

## Contents

<b>1</b>	<b>Downloading path and setting environment variables</b>	<b>2</b>
<b>2</b>	<b>Loading of the data</b>	<b>2</b>
<b>3</b>	<b>Model matrix</b>	<b>3</b>
<b>4</b>	<b>Preparations for shrinkage fitting</b>	<b>9</b>
<b>5</b>	<b>Fit coefficients with minimal shrinkage</b>	<b>11</b>
<b>6</b>	<b>Fit the dispersions</b>	<b>12</b>
<b>7</b>	<b>Fit coefficients with proper shrinkage</b>	<b>14</b>
<b>8</b>	<b>Standard errors from likelihood</b>	<b>16</b>
<b>9</b>	<b>Rearrange and summerize REUCs</b>	<b>20</b>
<b>10</b>	<b>Tissue dependence score</b>	<b>21</b>
<b>11</b>	<b>Test for conservation</b>	<b>22</b>
<b>12</b>	<b>Explained variance</b>	<b>24</b>
<b>13</b>	<b>Load all libraries for plots</b>	<b>29</b>
<b>14</b>	<b>Backgrounds for enrichments</b>	<b>30</b>

<b>15 Figures</b>	<b>31</b>
15.1 Figure 1B: visualization of REUCs in a CTDU case . . . . .	31
15.2 Figure 2A: PCA analysis . . . . .	35
15.3 Figure 2B: variance analysis . . . . .	39
15.4 Figure 2C: conserved tissue-specific vs mya . . . . .	42
15.5 Figure 2D: Tissue strength vs conservation . . . . .	44
15.6 Figure 3D: heatmap of tissue dependence . . . . .	46
15.7 Figure 3A: disordered regions . . . . .	48
15.8 Figure 3B: venn diagram . . . . .	49
15.9 Figure 3C: CTDU barchart in different cathegories and com- parison with background set of exons . . . . .	51
15.10 Figure 3E: splicing factor motif enrichments . . . . .	53
<b>16 Session Info</b>	<b>57</b>
<b>17 All genes with strictly CTDU exons.</b>	<b>58</b>

## 1 Downloading path and setting environment variables

Download the following directory:

```
http://www-huber.embl.de/pub/DEUprimates_supplement/SupplementaryFile3
```

Then, you should make the variable PRIMATES reflect the path to this directory:

```
cd ~/
wget -r -nH --cut-dirs=2 \
  --reject="index.html*" \
  http://www-huber.embl.de/pub/DEUprimates_supplement/SupplementaryFile3
export PRIMATES=~ /SupplementaryFile3
```

Then you are ready to reproduce the results from the paper:

## 2 Loading of the data

We use these packages for the analysis of REUCs

```
library( DEXSeq )
library( statmod )
library( multicore )
```

```
library( abind )
library( genefilter )
cores=15
```

The following loads an ExonCountSet with only the count values and the design but no intermediate or final results

```
setwd(Sys.getenv("PRIMATES"))
load( "inputs/ecsTRT.RData" )
```

Normalize factor levels

```
geneIDs(ecs) <- factor( geneIDs(ecs) )
```

### 3 Model matrix

We construct the model frame by taking the design frame (i.e., sample meta data) from the ecs argument and concatenate it to itself. This is because each sample appears with two rows in the model matrix, once with the counts for the exon described by the row and once with the sum of all other exons.

```
modelFrame <- cbind(
  sample = sampleNames(ecs),
  design(ecs),
  sizeFactor = sizeFactors(ecs) )
modelFrame <- rbind(
  cbind( modelFrame, exon="this" ),
  cbind( modelFrame, exon="others" ) )
rownames(modelFrame) <- paste( modelFrame$sample,
  modelFrame$exon, sep="-" )
```

The model frame now:

```
modelFrame
```

	sample	species	tissue	sex	sizeFactor	exon
ggo-br-F-1-ggo-this	ggo-br-F-1-ggo	ggo	br	F	2.0067414	this
ggo-br-M-1-ggo-this	ggo-br-M-1-ggo	ggo	br	M	1.0526627	this
ggo-cb-F-1-ggo-this	ggo-cb-F-1-ggo	ggo	cb	F	1.6221341	this
ggo-cb-M-1-ggo-this	ggo-cb-M-1-ggo	ggo	cb	M	0.8665522	this
ggo-ht-F-1-ggo-this	ggo-ht-F-1-ggo	ggo	ht	F	0.5751154	this

ggo-ht-M-1-ggo-this	ggo-ht-M-1-ggo	ggo	ht	M	0.3945627	this
ggo-kd-F-1-ggo-this	ggo-kd-F-1-ggo	ggo	kd	F	1.2493353	this
ggo-kd-M-1-ggo-this	ggo-kd-M-1-ggo	ggo	kd	M	1.3566434	this
ggo-lv-F-1-ggo-this	ggo-lv-F-1-ggo	ggo	lv	F	1.0174225	this
ggo-lv-M-1-ggo-this	ggo-lv-M-1-ggo	ggo	lv	M	1.1105119	this
hsa-br-F-1-hsa-this	hsa-br-F-1-hsa	hsa	br	F	1.6377571	this
hsa-br-M-1-hsa-this	hsa-br-M-1-hsa	hsa	br	M	1.1801184	this
hsa-br-M-2-hsa-this	hsa-br-M-2-hsa	hsa	br	M	1.2849264	this
hsa-br-M-3-hsa-this	hsa-br-M-3-hsa	hsa	br	M	0.9329287	this
hsa-br-M-4-hsa-this	hsa-br-M-4-hsa	hsa	br	M	0.8513831	this
hsa-br-M-5-hsa-this	hsa-br-M-5-hsa	hsa	br	M	0.2589920	this
hsa-cb-F-1-hsa-this	hsa-cb-F-1-hsa	hsa	cb	F	1.6327022	this
hsa-cb-M-1-hsa-this	hsa-cb-M-1-hsa	hsa	cb	M	2.6374909	this
hsa-ht-F-1-hsa-this	hsa-ht-F-1-hsa	hsa	ht	F	0.8321838	this
hsa-ht-M-1-hsa-this	hsa-ht-M-1-hsa	hsa	ht	M	1.1546257	this
hsa-ht-M-2-hsa-this	hsa-ht-M-2-hsa	hsa	ht	M	1.1002540	this
hsa-kd-F-1-hsa-this	hsa-kd-F-1-hsa	hsa	kd	F	1.4297802	this
hsa-kd-M-1-hsa-this	hsa-kd-M-1-hsa	hsa	kd	M	0.9659092	this
hsa-kd-M-2-hsa-this	hsa-kd-M-2-hsa	hsa	kd	M	1.2008835	this
hsa-lv-M-1-hsa-this	hsa-lv-M-1-hsa	hsa	lv	M	0.7943727	this
hsa-lv-M-2-hsa-this	hsa-lv-M-2-hsa	hsa	lv	M	0.8464109	this
mml-br-F-1-mml-this	mml-br-F-1-mml	mml	br	F	1.0363749	this
mml-br-M-1-mml-this	mml-br-M-1-mml	mml	br	M	1.0454692	this
mml-br-M-2-mml-this	mml-br-M-2-mml	mml	br	M	1.3040308	this
mml-cb-F-1-mml-this	mml-cb-F-1-mml	mml	cb	F	1.2432106	this
mml-cb-M-1-mml-this	mml-cb-M-1-mml	mml	cb	M	1.1965691	this
mml-ht-F-1-mml-this	mml-ht-F-1-mml	mml	ht	F	0.6214446	this
mml-ht-M-1-mml-this	mml-ht-M-1-mml	mml	ht	M	0.7366023	this
mml-kd-F-1-mml-this	mml-kd-F-1-mml	mml	kd	F	0.7966258	this
mml-kd-M-1-mml-this	mml-kd-M-1-mml	mml	kd	M	0.6366425	this
mml-lv-F-1-mml-this	mml-lv-F-1-mml	mml	lv	F	0.6254235	this
mml-lv-M-1-mml-this	mml-lv-M-1-mml	mml	lv	M	0.8636917	this
ppa-br-F-1-ptr-this	ppa-br-F-1-ptr	ppa	br	F	1.0709630	this
ppa-br-F-2-ptr-this	ppa-br-F-2-ptr	ppa	br	F	1.0990228	this
ppa-br-M-1-ptr-this	ppa-br-M-1-ptr	ppa	br	M	2.0790194	this
ppa-cb-F-1-ptr-this	ppa-cb-F-1-ptr	ppa	cb	F	1.4753921	this
ppa-cb-M-1-ptr-this	ppa-cb-M-1-ptr	ppa	cb	M	2.0698628	this
ppa-ht-F-1-ptr-this	ppa-ht-F-1-ptr	ppa	ht	F	1.1533740	this
ppa-ht-M-1-ptr-this	ppa-ht-M-1-ptr	ppa	ht	M	0.6629949	this
ppa-kd-F-1-ptr-this	ppa-kd-F-1-ptr	ppa	kd	F	1.3788909	this

ppa-kd-M-1-ptr-this	ppa-kd-M-1-ptr	ppa	kd	M	1.1443873	this
ppa-lv-F-1-ptr-this	ppa-lv-F-1-ptr	ppa	lv	F	0.6962407	this
ppa-lv-M-1-ptr-this	ppa-lv-M-1-ptr	ppa	lv	M	0.4914894	this
ppy-br-F-1-ppy-this	ppy-br-F-1-ppy	ppy	br	F	1.7079544	this
ppy-br-M-1-ppy-this	ppy-br-M-1-ppy	ppy	br	M	0.7396834	this
ppy-cb-F-1-ppy-this	ppy-cb-F-1-ppy	ppy	cb	F	1.0849149	this
ppy-ht-F-1-ppy-this	ppy-ht-F-1-ppy	ppy	ht	F	0.8499079	this
ppy-ht-M-1-ppy-this	ppy-ht-M-1-ppy	ppy	ht	M	0.4420630	this
ppy-kd-F-1-ppy-this	ppy-kd-F-1-ppy	ppy	kd	F	1.0844965	this
ppy-kd-M-1-ppy-this	ppy-kd-M-1-ppy	ppy	kd	M	1.2865658	this
ppy-lv-F-1-ppy-this	ppy-lv-F-1-ppy	ppy	lv	F	0.5737885	this
ppy-lv-M-1-ppy-this	ppy-lv-M-1-ppy	ppy	lv	M	1.0396424	this
ptr-br-F-1-ptr-this	ptr-br-F-1-ptr	ptr	br	F	1.0693073	this
ptr-br-M-1-ptr-this	ptr-br-M-1-ptr	ptr	br	M	0.7861358	this
ptr-br-M-2-ptr-this	ptr-br-M-2-ptr	ptr	br	M	0.6777453	this
ptr-br-M-3-ptr-this	ptr-br-M-3-ptr	ptr	br	M	0.6998900	this
ptr-br-M-4-ptr-this	ptr-br-M-4-ptr	ptr	br	M	1.1018289	this
ptr-br-M-5-ptr-this	ptr-br-M-5-ptr	ptr	br	M	1.0169716	this
ptr-cb-F-1-ptr-this	ptr-cb-F-1-ptr	ptr	cb	F	2.1463258	this
ptr-cb-M-1-ptr-this	ptr-cb-M-1-ptr	ptr	cb	M	1.2443930	this
ptr-ht-F-1-ptr-this	ptr-ht-F-1-ptr	ptr	ht	F	1.0107497	this
ptr-ht-M-1-ptr-this	ptr-ht-M-1-ptr	ptr	ht	M	1.0698054	this
ptr-kd-F-1-ptr-this	ptr-kd-F-1-ptr	ptr	kd	F	1.6584041	this
ptr-kd-M-1-ptr-this	ptr-kd-M-1-ptr	ptr	kd	M	2.1690272	this
ptr-lv-F-1-ptr-this	ptr-lv-F-1-ptr	ptr	lv	F	1.2792646	this
ptr-lv-M-1-ptr-this	ptr-lv-M-1-ptr	ptr	lv	M	0.5350664	this
ggo-br-F-1-ggo-others	ggo-br-F-1-ggo	ggo	br	F	2.0067414	others
ggo-br-M-1-ggo-others	ggo-br-M-1-ggo	ggo	br	M	1.0526627	others
ggo-cb-F-1-ggo-others	ggo-cb-F-1-ggo	ggo	cb	F	1.6221341	others
ggo-cb-M-1-ggo-others	ggo-cb-M-1-ggo	ggo	cb	M	0.8665522	others
ggo-ht-F-1-ggo-others	ggo-ht-F-1-ggo	ggo	ht	F	0.5751154	others
ggo-ht-M-1-ggo-others	ggo-ht-M-1-ggo	ggo	ht	M	0.3945627	others
ggo-kd-F-1-ggo-others	ggo-kd-F-1-ggo	ggo	kd	F	1.2493353	others
ggo-kd-M-1-ggo-others	ggo-kd-M-1-ggo	ggo	kd	M	1.3566434	others
ggo-lv-F-1-ggo-others	ggo-lv-F-1-ggo	ggo	lv	F	1.0174225	others
ggo-lv-M-1-ggo-others	ggo-lv-M-1-ggo	ggo	lv	M	1.1105119	others
hsa-br-F-1-hsa-others	hsa-br-F-1-hsa	hsa	br	F	1.6377571	others
hsa-br-M-1-hsa-others	hsa-br-M-1-hsa	hsa	br	M	1.1801184	others
hsa-br-M-2-hsa-others	hsa-br-M-2-hsa	hsa	br	M	1.2849264	others
hsa-br-M-3-hsa-others	hsa-br-M-3-hsa	hsa	br	M	0.9329287	others

hsa-br-M-4-hsa-others	hsa-br-M-4-hsa	hsa	br	M	0.8513831	others
hsa-br-M-5-hsa-others	hsa-br-M-5-hsa	hsa	br	M	0.2589920	others
hsa-cb-F-1-hsa-others	hsa-cb-F-1-hsa	hsa	cb	F	1.6327022	others
hsa-cb-M-1-hsa-others	hsa-cb-M-1-hsa	hsa	cb	M	2.6374909	others
hsa-ht-F-1-hsa-others	hsa-ht-F-1-hsa	hsa	ht	F	0.8321838	others
hsa-ht-M-1-hsa-others	hsa-ht-M-1-hsa	hsa	ht	M	1.1546257	others
hsa-ht-M-2-hsa-others	hsa-ht-M-2-hsa	hsa	ht	M	1.1002540	others
hsa-kd-F-1-hsa-others	hsa-kd-F-1-hsa	hsa	kd	F	1.4297802	others
hsa-kd-M-1-hsa-others	hsa-kd-M-1-hsa	hsa	kd	M	0.9659092	others
hsa-kd-M-2-hsa-others	hsa-kd-M-2-hsa	hsa	kd	M	1.2008835	others
hsa-lv-M-1-hsa-others	hsa-lv-M-1-hsa	hsa	lv	M	0.7943727	others
hsa-lv-M-2-hsa-others	hsa-lv-M-2-hsa	hsa	lv	M	0.8464109	others
mml-br-F-1-mml-others	mml-br-F-1-mml	mml	br	F	1.0363749	others
mml-br-M-1-mml-others	mml-br-M-1-mml	mml	br	M	1.0454692	others
mml-br-M-2-mml-others	mml-br-M-2-mml	mml	br	M	1.3040308	others
mml-cb-F-1-mml-others	mml-cb-F-1-mml	mml	cb	F	1.2432106	others
mml-cb-M-1-mml-others	mml-cb-M-1-mml	mml	cb	M	1.1965691	others
mml-ht-F-1-mml-others	mml-ht-F-1-mml	mml	ht	F	0.6214446	others
mml-ht-M-1-mml-others	mml-ht-M-1-mml	mml	ht	M	0.7366023	others
mml-kd-F-1-mml-others	mml-kd-F-1-mml	mml	kd	F	0.7966258	others
mml-kd-M-1-mml-others	mml-kd-M-1-mml	mml	kd	M	0.6366425	others
mml-lv-F-1-mml-others	mml-lv-F-1-mml	mml	lv	F	0.6254235	others
mml-lv-M-1-mml-others	mml-lv-M-1-mml	mml	lv	M	0.8636917	others
ppa-br-F-1-ptr-others	ppa-br-F-1-ptr	ppa	br	F	1.0709630	others
ppa-br-F-2-ptr-others	ppa-br-F-2-ptr	ppa	br	F	1.0990228	others
ppa-br-M-1-ptr-others	ppa-br-M-1-ptr	ppa	br	M	2.0790194	others
ppa-cb-F-1-ptr-others	ppa-cb-F-1-ptr	ppa	cb	F	1.4753921	others
ppa-cb-M-1-ptr-others	ppa-cb-M-1-ptr	ppa	cb	M	2.0698628	others
ppa-ht-F-1-ptr-others	ppa-ht-F-1-ptr	ppa	ht	F	1.1533740	others
ppa-ht-M-1-ptr-others	ppa-ht-M-1-ptr	ppa	ht	M	0.6629949	others
ppa-kd-F-1-ptr-others	ppa-kd-F-1-ptr	ppa	kd	F	1.3788909	others
ppa-kd-M-1-ptr-others	ppa-kd-M-1-ptr	ppa	kd	M	1.1443873	others
ppa-lv-F-1-ptr-others	ppa-lv-F-1-ptr	ppa	lv	F	0.6962407	others
ppa-lv-M-1-ptr-others	ppa-lv-M-1-ptr	ppa	lv	M	0.4914894	others
ppy-br-F-1-ppy-others	ppy-br-F-1-ppy	ppy	br	F	1.7079544	others
ppy-br-M-1-ppy-others	ppy-br-M-1-ppy	ppy	br	M	0.7396834	others
ppy-cb-F-1-ppy-others	ppy-cb-F-1-ppy	ppy	cb	F	1.0849149	others
ppy-ht-F-1-ppy-others	ppy-ht-F-1-ppy	ppy	ht	F	0.8499079	others
ppy-ht-M-1-ppy-others	ppy-ht-M-1-ppy	ppy	ht	M	0.4420630	others
ppy-kd-F-1-ppy-others	ppy-kd-F-1-ppy	ppy	kd	F	1.0844965	others

ppy-kd-M-1-ppy-others	ppy-kd-M-1-ppy	ppy	kd	M	1.2865658	others
ppy-lv-F-1-ppy-others	ppy-lv-F-1-ppy	ppy	lv	F	0.5737885	others
ppy-lv-M-1-ppy-others	ppy-lv-M-1-ppy	ppy	lv	M	1.0396424	others
ptr-br-F-1-ptr-others	ptr-br-F-1-ptr	ptr	br	F	1.0693073	others
ptr-br-M-1-ptr-others	ptr-br-M-1-ptr	ptr	br	M	0.7861358	others
ptr-br-M-2-ptr-others	ptr-br-M-2-ptr	ptr	br	M	0.6777453	others
ptr-br-M-3-ptr-others	ptr-br-M-3-ptr	ptr	br	M	0.6998900	others
ptr-br-M-4-ptr-others	ptr-br-M-4-ptr	ptr	br	M	1.1018289	others
ptr-br-M-5-ptr-others	ptr-br-M-5-ptr	ptr	br	M	1.0169716	others
ptr-cb-F-1-ptr-others	ptr-cb-F-1-ptr	ptr	cb	F	2.1463258	others
ptr-cb-M-1-ptr-others	ptr-cb-M-1-ptr	ptr	cb	M	1.2443930	others
ptr-ht-F-1-ptr-others	ptr-ht-F-1-ptr	ptr	ht	F	1.0107497	others
ptr-ht-M-1-ptr-others	ptr-ht-M-1-ptr	ptr	ht	M	1.0698054	others
ptr-kd-F-1-ptr-others	ptr-kd-F-1-ptr	ptr	kd	F	1.6584041	others
ptr-kd-M-1-ptr-others	ptr-kd-M-1-ptr	ptr	kd	M	2.1690272	others
ptr-lv-F-1-ptr-others	ptr-lv-F-1-ptr	ptr	lv	F	1.2792646	others
ptr-lv-M-1-ptr-others	ptr-lv-M-1-ptr	ptr	lv	M	0.5350664	others

To construct the model matrix, we first set "index variable", which indicate which columns of the model matrix refer to which factors.

```
sampleIdx <- data.frame( sample = levels(modelFrame$sample),
  col = seq_along( levels(modelFrame$sample) ) )
exonAvgIdx <- max(sampleIdx$col) + 1
sexIdx <- exonAvgIdx + 1
crossIdx <- expand.grid( tissue=levels(modelFrame$tissue),
  species=levels(modelFrame$species) )
crossIdx$col = sexIdx + 1:nrow(crossIdx)
```

```
head(sampleIdx)
```

	sample	col
1	ggo-br-F-1-ggo	1
2	ggo-br-M-1-ggo	2
3	ggo-cb-F-1-ggo	3
4	ggo-cb-M-1-ggo	4
5	ggo-ht-F-1-ggo	5
6	ggo-ht-M-1-ggo	6

```
exonAvgIdx
```

```
[1] 72
```

```
sexIdx
```

```
[1] 73
```

```
head(crossIdx)
```

```
  tissue species col
1     br      ggo  74
2     cb      ggo  75
3     ht      ggo  76
4     kd      ggo  77
5     lv      ggo  78
6     br      hsa  79
```

Next, fill the matrix according to the indexes

```
mm <- matrix( 0, nrow = nrow(modelFrame), ncol = max(crossIdx$col) )
colnames(mm) <- rep( NA, ncol(mm) )
colnames(mm)[sampleIdx$col] <- paste( "sample", sampleIdx$sample, sep="_" )
colnames(mm)[exonAvgIdx] <- "exonAvg"
colnames(mm)[sexIdx] <- "sex"
colnames(mm)[crossIdx$col] <- sprintf( "tissue_%s:species_%s",
  crossIdx$tissue, crossIdx$species )

for( i in 1:nrow(modelFrame) ) {
  mm[ i, sampleIdx$col[ sampleIdx$sample == modelFrame$sample[i] ] ] <- 1
  mm[ i, exonAvgIdx ] <- { if( modelFrame$exon[i] == "this" ) 1 else 0 }
  mm[ i, sexIdx ] <- {
    if( modelFrame$sex[i] == "M" && modelFrame$exon[i] == "this" ) .5 else -.5 }
  mm[ i, crossIdx$col[ crossIdx$tissue == modelFrame$tissue[i]
    & crossIdx$species == modelFrame$species[i] ] ] <-
    { if( modelFrame$exon[i] == "this" ) .5 else -.5 }
}
```

Make the table of "other" counts (i.e., of counts for all exons in the gene but the exon under consideration)

```
a <- do.call( rbind,
  tapply( 1:nrow(ecs), geneIDs(ecs), function(i) {
```



```

      sct <- counts(ecs)[i,,drop=FALSE]
      t( sapply( 1:nrow(sct), function(r)
        colSums( sct[-r,,drop=FALSE] ) ) )
    } ) )
assayData(ecs)[["otherCounts"]] <- a

```

An accessor for this count, and a look into it.

```

otherCounts <- function( ecs ) assayData(ecs)[["otherCounts"]]

str( otherCounts(ecs) )

num [1:119344, 1:71] 206 213 235 225 188 153 146 156 150 143 ...
- attr(*, "dimnames")=List of 2
..$ : NULL
..$ : chr [1:71] "ggo-br-F-1-ggo" "ggo-br-M-1-ggo" "ggo-cb-F-1-ggo" "ggo-cb-M-1-ggo"

```

## 4 Preparations for shrinkage fitting

The likelihood of the negative binomial (without terms that do not depend on muhat) and its gradient w.r.t. muhat

```

ll <- function( muhat, y, disp )
  sum( y * log( muhat ) - ( y + 1/disp ) * log( muhat + 1/disp ) )

dll <- function( muhat, y, disp )
  y / muhat - ( y + 1/disp ) / ( muhat + 1/disp )

```

The function to perform the shrinkage fit.

This function takes a model matrix (mm), a model frame (mf), initial values (beta0), a boolean vector indicating which coefficients to shrink (beta0) and a prior standard deviation as shrinkage goal (priorsd) and performs a GLM fit

```

shrinkageFit <- function( mm, counts,
  dispersions, sizeFactors,
  beta0, shrink, priorsd ) {
  ofit <- optim(
    beta0,
    function(beta) {

```

```

muhat <- exp( mm %% beta ) * sizeFactors
-ll( muhat, counts, dispersions ) + sum( beta[shrink]^2 ) / priorsd^2 / 2 } ,
  function(beta) {
muhat <- exp( mm %% beta ) * sizeFactors
-t( dll( muhat, counts, dispersions ) * muhat ) %% mm
  + beta*shrink / priorsd^2 },
  method="L-BFGS-B",
  control = list( trace=0, maxit=400, factr=1e2 ) )

if( ofit$convergence != 0 )
  warning( "L-BFGS optimization did not converge." )

ofit$par
}

```

This function does the work, with mclapply

```

fitAllExons <- function( ecs, mm, shrink, priorsd ) {
  sf <- rep( sizeFactors(ecs), 2 )
  rows <- 1:nrow(ecs)
  allCoefs <- mclapply( rows,
    function(i) {
if( i %% 1000 == 0 )
  print( i )
if( fData(ecs)$testable[i] ) {
  cnts <- c( counts(ecs)[i,], otherCounts(ecs)[i,] )
  disps <- pmax( 3e-3,
    ifelse( modelFrame$exon=="this",
      fData(ecs)$dispThis[i],
      fData(ecs)$dispOthers ) )
  beta0 <- rep( 2, ncol(mm) )
  ans <- try(
    shrinkageFit( mm, cnts,
      disps, sf, beta0, shrink, priorsd ) )
  if( !inherits( ans, "try-error" ) )
    ans
  else
    rep( NA, ncol(mm) ) }
else
  rep( NA, ncol(mm) ) },

```

```

    mc.cores = cores )
  allCoefs <- do.call( rbind, allCoefs )
  rownames(allCoefs) <- featureNames(ecs)[rows]
  colnames(allCoefs) <- colnames(mm)
  allCoefs
}

```

These coefficients should be subject to shrinkage

```
shrink <- 1:ncol(mm) %in% c( sexIdx, crossIdx$col )
```

## 5 Fit coefficients with minimal shrinkage

For the initial fit, we set all dispersions to a plausible initial value. We have two dispersion values for each exon, one for the response variable with `exon=="this"` in the model frame and one for those with `exon=="others"`.

```
fData(ecs)$dispThis <- .1
fData(ecs)$dispOthers <- .1
```

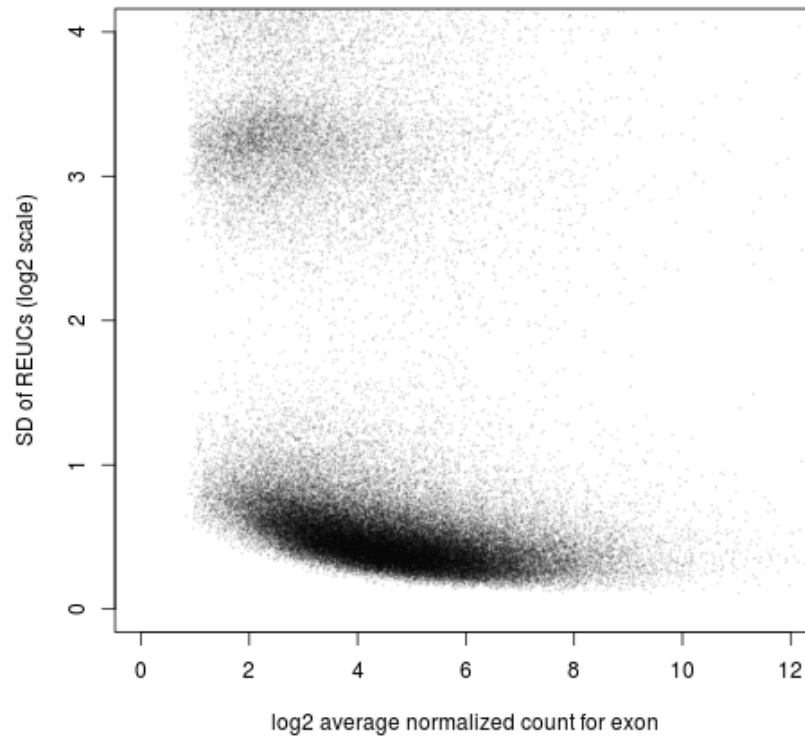
Now, run the fit.

```
allCoefsWeakShrinkage <- fitAllExons( ecs, mm, shrink, priorsd=1000 )
```

Plot the SD of the cross coefficients against the means of the sample coefficients.

```
plot(
  log2( rowMeans( counts( ecs, normalized=TRUE ) ) ),
  rowSds( allCoefsWeakShrinkage[,crossIdx$col] ) / log(2),
  xlim=c(0,12), ylim=c(0,4), pch=20, cex=.3, col="#00000010",
  xlab="log2 average normalized count for exon",
  ylab="SD of REUCs (log2 scale)" )

```



It seems that for sample effects above 8 on a log2 scale, there is only little heteroscedasticity left. Therefore, we take the square root of the mean of these variances as shrinkage prior:

```
priorsd <- sqrt( mean( rowVars( allCoefsWeakShrinkage[
  rowMeans( log2( counts( ecs, normalized=TRUE ) ) ) > 8,
  crossIdx$col ] ), na.rm=TRUE ) )
priorsd
[1] 0.3240358
```

## 6 Fit the dispersions

The Cox-Reid adjusted log likelihood

```
llCR <- function( muhat, y, disp, mm ) {
```

```

ll <- sum( sapply( seq(along=y), function(i)
  dnbinom( y[i], mu=muhat[i], size=1/disp[i], log=TRUE ) ) )
z <- log(muhat) + ( y - muhat ) / muhat
v0 <- muhat + disp * muhat^2
w <- 1 / ( ( 1 / muhat )^2 * v0 )
qrres <- qr( mm*sqrt(w) )
cr <- sum( log( abs( diag( qrres$qr ) [ seq_len(qrres$rank) ] ) ) )
ll - cr
}

```

This function maximizes the adjusted likelihood

```

estimateDispForExon <- function( muhat, y, mm, isThis,
  startDispThis=.1, startDispOthers=.1 ) {
  a <- optim(
    log( c( startDispThis, startDispOthers ) ),
    function( x ) {
  disps <- ifelse( isThis, exp(x[1]), exp(x[2]) )
-llCR( muhat, y, disps, mm ) } )
  names(a$par) <- c( "dispThis", "dispOthers" )
  exp(a$par)
}

```

Calculate the dispersions.

```

rawDisps <- do.call( rbind, mclapply( 1:nrow(ecs), function(i) {
  disps <- try(
    estimateDispForExon(
  as.vector( exp( mm %*% allCoefsWeakShrinkage[i,] ) *
    rep( sizeFactors(ecs), 2 ) ),
  c( counts(ecs)[i,], otherCounts(ecs)[i,] ),
  mm,
  modelFrame$exon == "this" ), silent=TRUE )
  if( !inherits( disps, "try-error" ) )
    disps
  else
    c( NA, NA ) },
  mc.cores=cores ) )

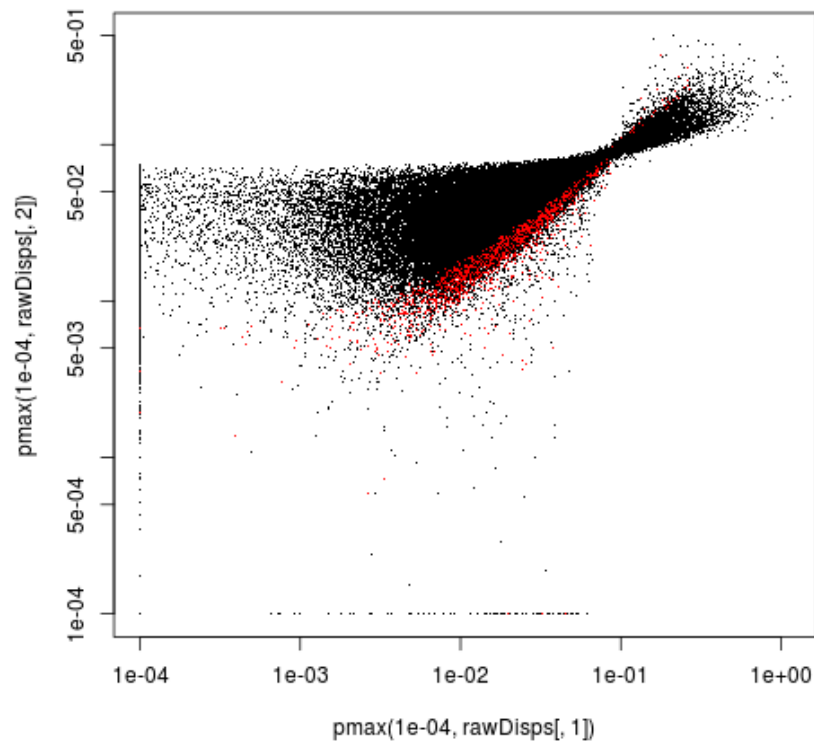
```

Plot the "others" dispersions against the "this" dispersions, highlighting stronger exons in red.

```

plot(
  pmax( 1e-4, rawDisps[,1] ),
  pmax( 1e-4, rawDisps[,2] ), log="xy", pch=".", asp=1,
  col=1+( rowMeans(log10(counts(ecs,normalized=TRUE))) > 2 ) )

```



Write the results into the ecs object.

```

fData(ecs)$dispThis <- ifelse( !is.na(rawDisps[,"dispThis"]),
  rawDisps[,"dispThis"], mean( rawDisps[,"dispThis"], na.rm=TRUE ) )
fData(ecs)$dispOthers <- ifelse( !is.na(rawDisps[,"dispOthers"]),
  rawDisps[,"dispOthers"], mean( rawDisps[,"dispOthers"], na.rm=TRUE ) )

```

## 7 Fit coefficients with proper shrinkage

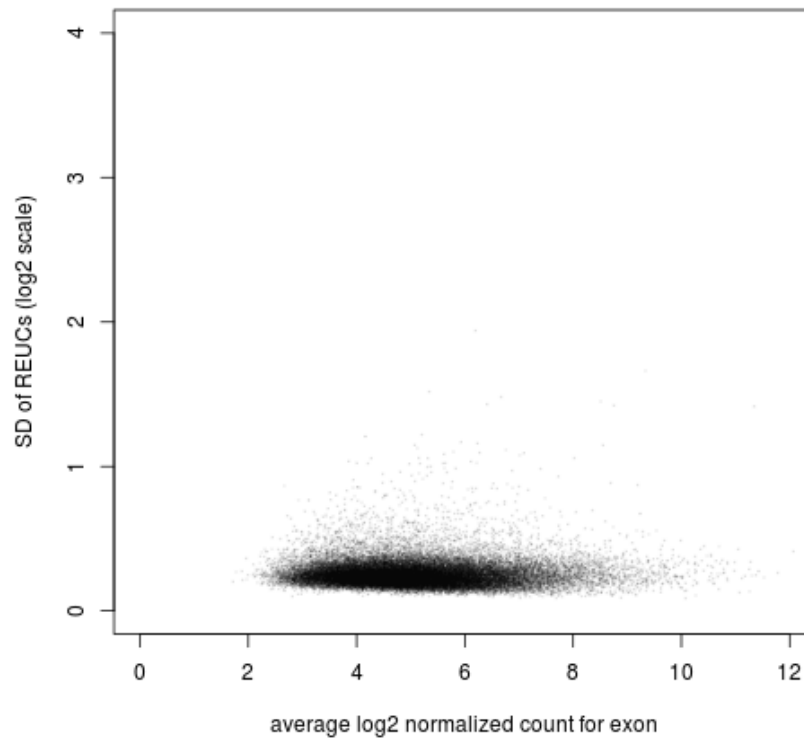
Do the fitting again, now using the priorsd and dispersion values found in the initial fit.

```
allCoefs <- fitAllExons( ecs, mm, shrink, priorsd )
```

Do the plot again.

Plot the SD of the cross coefficients against the means of the sample coefficients.

```
plot(  
  rowMeans( log2( counts( ecs, normalized=TRUE ) ) ),  
  rowSds( allCoefs[,crossIdx$col] ) / log(2),  
  xlim=c(0,12), ylim=c(0,4), pch=20, cex=.3, col="#00000010",  
  xlab="average log2 normalized count for exon",  
  ylab="SD of REUCs (log2 scale)" )
```



## 8 Standard errors from likelihood

Determine standard errors for the coefficients by calculating the second derivative of the likelihood wrt the coefficients.

```
hStep <- .1
sf <- rep( sizeFactors(ecs), 2 )
allSE <- sqrt( do.call( rbind,
  mclapply( 1:nrow(allCoefs), function(i) {
    if( i %% 1000 == 0 )
    print(i)
    if( any( is.na(allCoefs[i,]) ) )
    rep( NA, ncol(allCoefs) )
    else {
    y <- c( counts(ecs)[i,], otherCounts(ecs)[i,] )
    disps <-
      rep( c( fData(ecs)$dispThis[i], fData(ecs)$dispOthers[i] ),
        each = ncol(ecs) )
    sapply( 1:ncol(allCoefs), function(j) {
      beta <- allCoefs[i,]
      lln <- sapply( c( -hStep, 0, hStep ), function(dbeta) {
        beta[j] <- allCoefs[i,j] + dbeta
        muhat <- exp( mm %*% beta ) * sf
        -ll( muhat, y, disps ) } )
      hStep^2 / ( lln[1] - 2*lln[2] + lln[3] ) } ) } },
      mc.cores=cores ) ) )
dimnames(allSE) <- dimnames(allCoefs)
```

```
[1] 1000
[1] 2000
[1] 3000
[1] 4000
[1] 5000
[1] 6000
[1] 7000
[1] 8000
[1] 9000
[1] 10000
[1] 11000
[1] 12000
```



[1] 13000  
[1] 14000  
[1] 15000  
[1] 16000  
[1] 17000  
[1] 18000  
[1] 19000  
[1] 20000  
[1] 21000  
[1] 22000  
[1] 23000  
[1] 24000  
[1] 25000  
[1] 26000  
[1] 27000  
[1] 28000  
[1] 29000  
[1] 30000  
[1] 31000  
[1] 32000  
[1] 33000  
[1] 34000  
[1] 35000  
[1] 36000  
[1] 37000  
[1] 38000  
[1] 39000  
[1] 40000  
[1] 41000  
[1] 42000  
[1] 43000  
[1] 44000  
[1] 45000  
[1] 46000  
[1] 47000  
[1] 48000  
[1] 49000  
[1] 50000  
[1] 51000  
[1] 52000

[1] 53000  
[1] 54000  
[1] 55000  
[1] 56000  
[1] 57000  
[1] 58000  
[1] 59000  
[1] 60000  
[1] 61000  
[1] 62000  
[1] 63000  
[1] 64000  
[1] 65000  
[1] 66000  
[1] 67000  
[1] 68000  
[1] 69000  
[1] 70000  
[1] 71000  
[1] 72000  
[1] 73000  
[1] 74000  
[1] 75000  
[1] 76000  
[1] 77000  
[1] 78000  
[1] 79000  
[1] 80000  
[1] 81000  
[1] 82000  
[1] 83000  
[1] 84000  
[1] 85000  
[1] 86000  
[1] 87000  
[1] 88000  
[1] 89000  
[1] 90000  
[1] 91000  
[1] 92000

```
[1] 93000
[1] 94000
[1] 95000
[1] 96000
[1] 97000
[1] 98000
[1] 99000
[1] 100000
[1] 101000
[1] 102000
[1] 103000
[1] 104000
[1] 105000
[1] 106000
[1] 107000
[1] 108000
[1] 109000
[1] 110000
[1] 111000
[1] 112000
[1] 113000
[1] 114000
[1] 115000
[1] 116000
[1] 117000
[1] 118000
[1] 119000
```

Warning message:

```
In sqrt(do.call(rbind, mclapply(1:nrow(allCoefs), function(i) { :
  NaNs produced
```

```
str(allSE)
```

```
num [1:119344, 1:103] 0.178 0.16 0.162 0.147 0.129 ...
```

```
- attr(*, "dimnames")=List of 2
```

```
..$ : chr [1:119344] "ENSG00000000419:E005" "ENSG00000000419:E010" "ENSG00000000419:E
```

```
..$ : chr [1:103] "sample_ggo-br-F-1-ggo" "sample_ggo-br-M-1-ggo" "sample_ggo-cb-F-1-g
```

## 9 Rearrange and summerize REUCs

Rearrange cross coefficients (REUCs) into a 3D array.

```
crossCoefs <- do.call( abind, c( along=3,
  tapply( 1:nrow(crossIdx), crossIdx$tissue, function(i) {
    x <- allCoefs[,crossIdx$col[i]];
    colnames(x) <- crossIdx$species[i];
    x} ) ) )
```

```
str(crossCoefs)
```

```
num [1:119344, 1:6, 1:5] -0.368 -0.191 -0.277 -0.171 0.122 ...
```

```
- attr(*, "dimnames")=List of 3
```

```
..$ : chr [1:119344] "ENSG00000000419:E005" "ENSG00000000419:E010" "ENSG00000000419:E
```

```
..$ : chr [1:6] "ggo" "hsa" "mml" "ppa" ...
```

```
..$ : chr [1:5] "br" "cb" "ht" "kd" ...
```

Do the same for their standard errors.

```
crossCoefSEs <- do.call( abind, c( along=3,
  tapply( 1:nrow(crossIdx), crossIdx$tissue, function(i) {
    x <- allSE[,crossIdx$col[i]];
    colnames(x) <- crossIdx$species[i];
    x} ) ) )
```

```
str(crossCoefSEs)
```

```
num [1:119344, 1:6, 1:5] 0.274 0.25 0.255 0.237 0.208 ...
```

```
- attr(*, "dimnames")=List of 3
```

```
..$ : chr [1:119344] "ENSG00000000419:E005" "ENSG00000000419:E010" "ENSG00000000419:E
```

```
..$ : chr [1:6] "ggo" "hsa" "mml" "ppa" ...
```

```
..$ : chr [1:5] "br" "cb" "ht" "kd" ...
```

Calculate standard deviations across tissues:

```
spSDs <- apply( crossCoefs, 1:2, sd )
str( spSDs )
```

```
num [1:119344, 1:6] 0.144 0.172 0.273 0.218 0.135 ...
```

```
- attr(*, "dimnames")=List of 2
```

```
..$ : chr [1:119344] "ENSG00000000419:E005" "ENSG00000000419:E010" "ENSG00000000419:E
```

```
..$ : chr [1:6] "ggo" "hsa" "mml" "ppa" ...
```

Species pairs:

```
spPairs <- t( combn( levels(modelFrame$species), 2 ) )
rownames(spPairs) <- paste( spPairs[,1], spPairs[,2], sep=":" )
head( spPairs )
```

```
      [,1] [,2]
ggo:hsa "ggo" "hsa"
ggo:mml "ggo" "mml"
ggo:ppa "ggo" "ppa"
ggo:ppy "ggo" "ppy"
ggo:ptr "ggo" "ptr"
hsa:mml "hsa" "mml"
```

Calculate covariances

```
sppCovs <- apply( spPairs, 1, function(spp)
  apply( crossCoefs, 1, function(a)
    cov( a[ spp[1], ], a[ spp[2], ] ) ) )
str( sppCovs )
```

```
num [1:119344, 1:15] 0.00724 -0.00471 0.03738 0.01067 0.00687 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:119344] "ENSG00000000419:E005" "ENSG00000000419:E010" "ENSG00000000419:E011" ...
..$ : chr [1:15] "ggo:hsa" "ggo:mml" "ggo:ppa" "ggo:ppy" ...
```

and correlations

```
sppCors <- apply( spPairs, 1, function(spp)
  apply( crossCoefs, 1, function(a)
    cor( a[ spp[1], ], a[ spp[2], ] ) ) )
str( sppCors )
```

```
num [1:119344, 1:15] 0.473 -0.26 0.716 0.563 0.605 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:119344] "ENSG00000000419:E005" "ENSG00000000419:E010" "ENSG00000000419:E011" ...
..$ : chr [1:15] "ggo:hsa" "ggo:mml" "ggo:ppa" "ggo:ppy" ...
```

## 10 Tissue dependence score

Get maximum deviation from the average across tissues, within a species

```

maxDevSp <- apply( crossCoefs, 1:2, function(x) {
  xm <- x - mean(x)
  wm <- which.max(abs(xm))
  if( length(wm)!=0 ) xm[wm] else NA } )
str( maxDevSp )

num [1:119344, 1:6] 0.188 -0.225 -0.289 -0.279 0.146 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:119344] "ENSG00000000419:E005" "ENSG00000000419:E010" "ENSG00000000419:E015" ...
..$ : chr [1:6] "ggo" "hsa" "mml" "ppa" ...

```

Maximize over species

```

maxDev <- apply( maxDevSp, 1, function(x) {
  wm <- which.max(abs(x))
  if( length(wm)!=0 ) x[wm] else NA } )
str( maxDev )

Named num [1:119344] -0.426 -0.247 0.416 -0.279 -0.229 ...
- attr(*, "names")= chr [1:119344] "ENSG00000000419:E005" "ENSG00000000419:E010" "ENSG00000000419:E015" ...

```

Table

```

addmargins( table( cut( abs(maxDev), c( -Inf, 1/2, 1, Inf ) * log(2) ) ) ) )

```

(-Inf,0.347]	(0.347,0.693]	(0.693, Inf]	Sum
58033	30985	2578	91596

```

addmargins( table( cut( abs(maxDevSp[,"hsa"]), c( -Inf, 1/2, 1, Inf ) * log(2) ) ) ) )

```

(-Inf,0.347]	(0.347,0.693]	(0.693, Inf]	Sum
80032	10185	1379	91596

## 11 Test for conservation

```

pAdjCov <- apply( sppCovs, 2, function(a)
  ecdf( a )( -a ) / ( 1 - ecdf( a )( a ) ) )
rownames( pAdjCov ) <- rownames( sppCovs )
str( pAdjCov )

```

```

      ggo:hsa ggo:mml ggo:ppa ggo:ppy ggo:ptr hsa:mml hsa:ppa hsa:ppy hsa:ptr
FALSE  76677  86776  77631  86342  86679  87680  69590  84346  82793
TRUE   14918   4819  13964   5253   4916   3915  22005   7249   8802
      mml:ppa mml:ppy mml:ptr ppa:ppy ppa:ptr ppy:ptr
FALSE  88038  86802  86955  82543  76051  86710
TRUE   3557   4793   4640   9052  15544  4885

```

What are the thresholds?

```

covThrSp <- sapply( rownames(spPairs),
                   function(spp)
                     min( sppCovs[ which( padjCov[,spp] < .1), spp ] ) )
covThrSp
max(covThrSp)

```

```

      ggo:hsa  ggo:mml  ggo:ppa  ggo:ppy  ggo:ptr  hsa:mml  hsa:ppa
0.01920610 0.03548043 0.01938005 0.03092064 0.03507048 0.04659819 0.01623985
      hsa:ppy  hsa:ptr  mml:ppa  mml:ppy  mml:ptr  ppa:ppy  ppa:ptr
0.02848151 0.02774398 0.04708521 0.03461425 0.03949056 0.02379542 0.01991629
      ppy:ptr
0.03606846
[1] 0.04708521

```

The same on log2 scale

```

covThrSp / log(2)^2
max(covThrSp) / log(2)^2

      ggo:hsa  ggo:mml  ggo:ppa  ggo:ppy  ggo:ptr  hsa:mml  hsa:ppa
0.03997497 0.07384787 0.04033703 0.06435726 0.07299461 0.09698803 0.03380112
      hsa:ppy  hsa:ptr  mml:ppa  mml:ppy  mml:ptr  ppa:ppy  ppa:ptr
0.05928054 0.05774547 0.09800170 0.07204503 0.08219443 0.04952705 0.04145316
      ppy:ptr
0.07507177
[1] 0.0980017

```

Let's round this to 0.1

```

covThr <- 0.1 * log(2)^2

```

How many surpass this maximum threshold?

```
apply( sppCovs, 2, function(x) table( x > covThr ) )
```

```

      ggo:hsa ggo:mml ggo:ppa ggo:ppy ggo:ptr hsa:mml hsa:ppa hsa:ppy hsa:ptr
FALSE  87097  88378  87372  88572  88168  87796  86152  87820  86836
TRUE   4499   3218   4224   3024   3428   3800   5444   3776   4760
      mml:ppa mml:ppy mml:ptr ppa:ppy ppa:ptr ppy:ptr
FALSE  88131  88379  87980  88068  86799  88112
TRUE   3465   3217   3616   3528   4797   3484

```

And how many genes?

```

apply( sppCovs, 2, function(x)
  length( unique( substr(
    rownames(sppCovs)[ which( x > covThr ) ],
    1, 15 )
  ) ) )

```

```

ggo:hsa ggo:mml ggo:ppa ggo:ppy ggo:ptr hsa:mml hsa:ppa hsa:ppy hsa:ptr mml:ppa
2044    1487    1924    1380    1568    1643    2310    1653    2023    1584
mml:ppy mml:ptr ppa:ppy ppa:ptr ppy:ptr
1462    1617    1615    2148    1564

```

Compare conservation between human and macaque with tissue dependence score for human:

```

addmargins( table(
  TDS = cut( abs(maxDevSp[,"hsa"]),
    c( -Inf, 1/2, 1, Inf ) * log(2) ),
  cons = sppCovs[,"hsa:mml"] > covThr ) )

```

	cons		
TDS	FALSE	TRUE	Sum
(-Inf,0.347]	79607	425	80032
(0.347,0.693]	7917	2268	10185
(0.693, Inf]	272	1107	1379
Sum	87796	3800	91596

## 12 Explained variance

```

msq <- do.call( rbind, mclapply(
  1:nrow(allCoefs),

```



```

function(i) {
  reucs <- allCoefs[i,crossIdx$col]
  if( any(is.na( reucs )) )
rep( NA, 3 )
  else
anova( lm( REUC ~ species + tissue,
          data.frame( crossIdx,
                     REUC = allCoefs[i,crossIdx$col] ) ) )$'Mean Sq' },
      mc.cores=cores ) )
rownames(msq) <- rownames(allCoefs)
colnames(msq) <- c( "species", "tissue", "residual" )
save(msq, file="objects/explainedVariance.RData")
save.image("objects/REUC_analysis.rda")

gamma <- .3
plot( asinh( msq[,"tissue"] / log(2)^2 / gamma ),
      asinh( msq[,"species"] / log(2)^2 / gamma ),
      pch=20, cex=.2, xaxt="n", yaxt="n", asp=1,
      xlab="variance explained by tissue",
      ylab="variance explained by species",
      col=ifelse( rowSums(sppCovs>covThr) == 15, "#FF000060",
                 ifelse( sppCovs[,"hsa:mml"] > covThr, "#FF00FF60", "#00000060" ) ) ) )
ticks <- c( 0, .1*(1:9), 1:10, 20 )
longTicks <- c( 0, .1, 1, 10 )
for( ax in 1:2 ) {
  labeledTicks <- c( 0, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20 )
  axis( ax, asinh( labeledTicks / gamma ), labeledTicks,
        tcl=0, las=1 )
  axis( ax, asinh( c( 0, .1, 1, 10 ) / gamma ),
        FALSE, tcl=-.8 )
  axis( ax, asinh( c( .1*(2:9), 2:9, 20 ) / gamma ),
        FALSE, tcl=-.4 ) }
abline( h = asinh(0.75/gamma), v = asinh(0.75/gamma),
        col="#00000060", lty="dashed" )

```

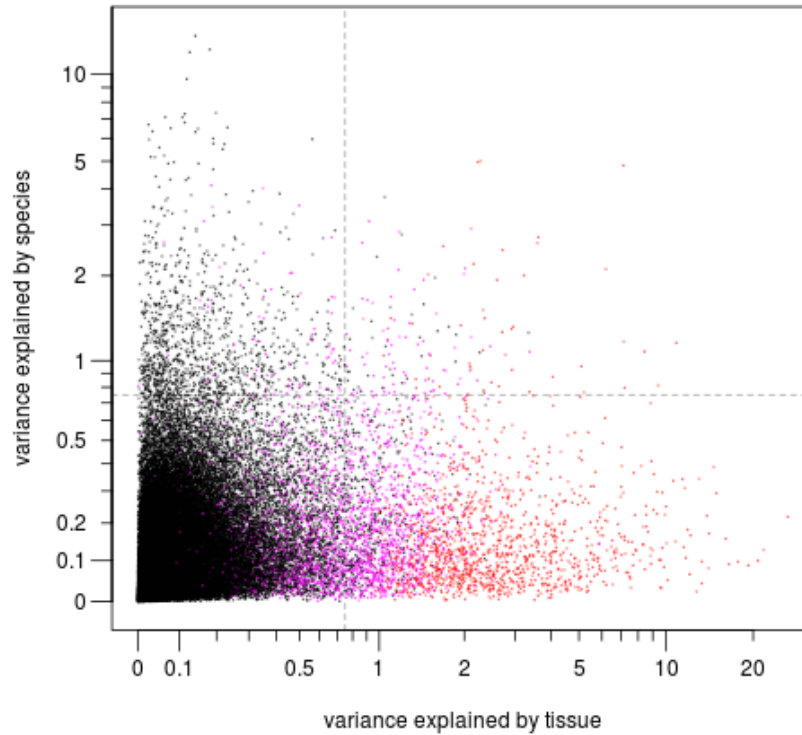


Table:

```
table( 'tissue > 0.75' = msq[,"tissue"] / log(2)^2 > .75,
      'species > 0.75' = msq[,"species"] / log(2)^2 > .75 )
```

	species > 0.75	
tissue > 0.75	FALSE	TRUE
FALSE	87591	1189
TRUE	2668	148

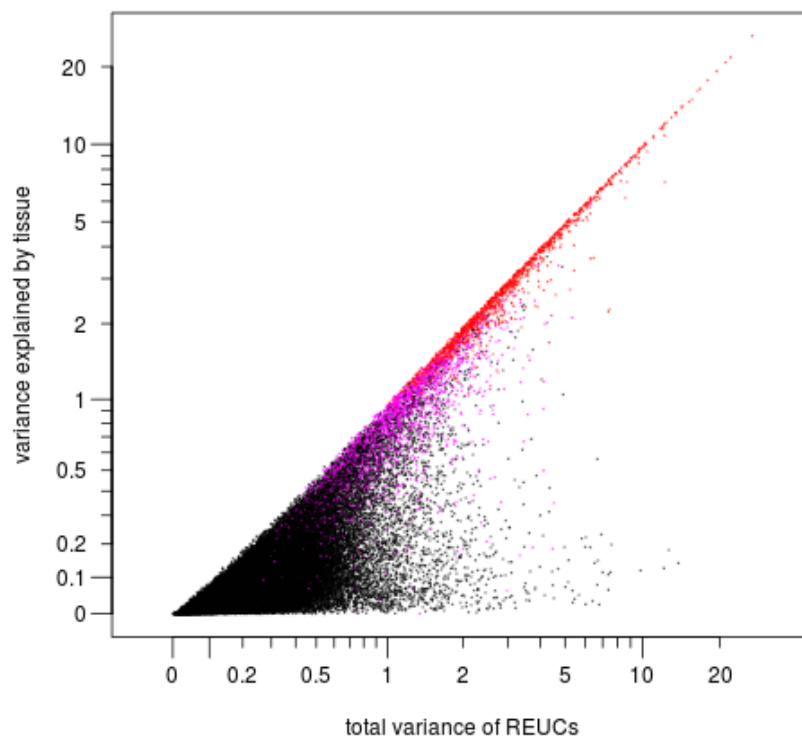
Showing tissue versus all variation

```
gamma <- .3
plot( asinh( rowSums(msq) / log(2)^2 / gamma ),
      asinh( msq[,"tissue"] / log(2)^2 / gamma ),
      pch=20, cex=.2, xaxt="n", yaxt="n", asp=1,
      xlab="total variance of REUCs",
```

```

ylab="variance explained by tissue",
col=ifelse( rowSums(sppCovs>covThr) == 15, "#FF000060",
           ifelse( sppCovs[,"hsa:mml"] > covThr, "#FF00FF60", "#00000060" ) ) )
ticks <- c( 0, .1*(1:9), 1:10, 20 )
longTicks <- c( 0, .1, 1, 10 )
for( ax in 1:2 ) {
  labeledTicks <- c( 0, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20 )
  axis( ax, asinh( labeledTicks / gamma ),
        labeledTicks, tcl=0, las=1 )
  axis( ax, asinh( c( 0, .1, 1, 10 ) / gamma ),
        FALSE, tcl=-.8 )
  axis( ax, asinh( c( .1*(2:9), 2:9, 20 ) / gamma ),
        FALSE, tcl=-.4 ) }

```

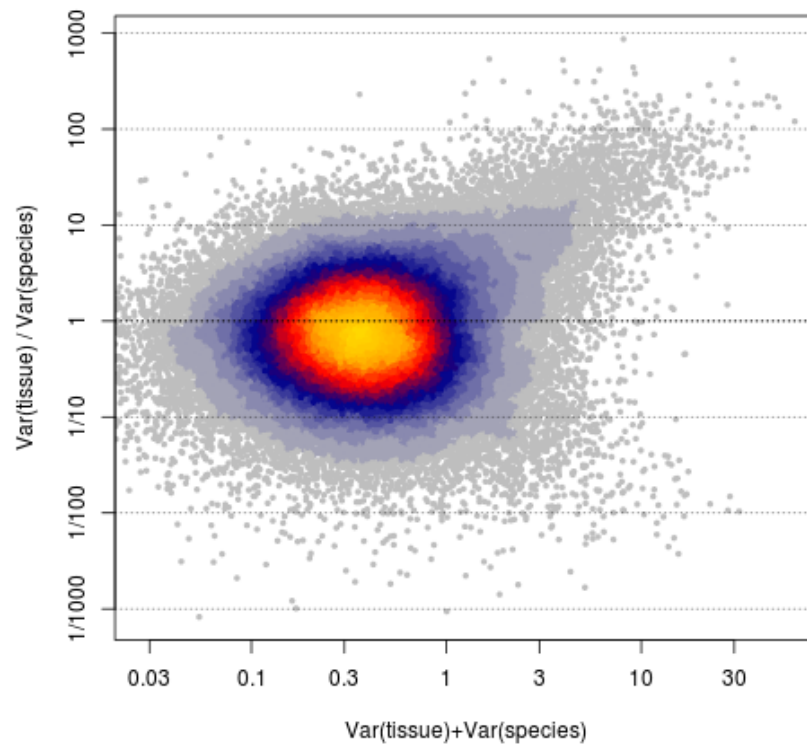


Variation explained by tissue and species plotted in an MA plot  
library(LSD)

```

heatscatter(
  log10( ( msq[,"tissue"] + msq[,"species"] ) / log(2) / gamma ),
  log10( msq[,"tissue"] / msq[,"species"] ),
  xaxt="n", yaxt="n", main="", cor=FALSE, xlim=c(-1.7, 1.9 ), xaxs="i",
  xlab = "Var(tissue)+Var(species)",
  ylab = "Var(tissue) / Var(species)" )
axis( 1, log10( c( .03, .1, .3, 1, 3, 10, 30 ) ),
  c( "0.03", "0.1", "0.3", "1", "3", "10", "30" ) )
axis( 2, (-3:3),
  c( "1/1000", "1/100", "1/10", "1", "10", "100", "1000" ) )
abline( h = -3:3, col="#00000080", lty="dotted" )
abline( h = 0, lty="dotted" )

```



### 13 Load all libraries for plots

```
library(lattice)
library(latticeExtra)
library(DEXSeq)
library(DESeq)
library(parallel)
library(RColorBrewer)
library(MatchIt)
library(geneplotter)
library( plotrix )
library(gplots)
library(Vennerable)
library(ggplot2)
library(GenomicRanges)
library(GenomicFeatures)
## PRIMATES should reflect the path to supplementaryFile3
setwd(Sys.getenv("PRIMATES"))
colSpecies <- brewer.pal(7, "Set1")[-6]
load("objects/REUC_analysis.rda")
load("inputs/ecsTRT.RData")
```

Change the object types for easier acces in further plots

```
species <- dimnames( crossCoefs[ 1, , ] )[[1]]
tissues <- dimnames( crossCoefs[ 1, , ] )[[2]]
allEffects <- lapply( species, function(y){
  vect <- lapply( tissues, function(x){ crossCoefs[ , y , x] } )
  vect <- do.call( cbind, vect )
  colnames( vect ) <- paste(y, tissues, sep="_" )
  vect
} )
names( allEffects ) <- species
allEffects <- do.call( cbind, allEffects )
save(allEffects, file="objects/allEffects.RData")
ctsr <- names( which( rowSums( sppCovs > max( covThr ) )
  == ncol( sppCovs ) ) )
save(ctsr, file="objects/ctsr.RData")
```

## 14 Backgrounds for enrichments

Exon backgrounds

```
load("inputs/ecsTRT.RData")
load("objects/ctsr.RData")

df <- data.frame(
  significant=as.numeric(featureNames(ecs) %in% ctsr),
  mean=rowMeans(counts(ecs)),
  length=fData(ecs)$end-fData(ecs)$start,
  dispersion=fData(ecs)$dispersion
)

df <- df[!is.na( df$significant ),]
back <- c()
for( i in 1:4 ){
  mmcts <- matchit(
    significant ~ mean + length + dispersion,
    df, distance="mahalanobis")
  back <- c(back, mmcts$match.matrix[,1])
  df <- df[-which( rownames(df) %in% back ),]
}

save(back, file="objects/back-exon.RData")

df <- data.frame(
  significant=as.numeric(featureNames(ecs) %in% ctsr),
  mean=rowMeans(counts(ecs)),
  length=fData(ecs)$end-fData(ecs)$start,
  dispersion=fData(ecs)$dispersion
)

df <- df[!is.na( df$significant ),]
domains <- read.delim("inputs/mappedExons_proteinscores.txt",
  header=TRUE, comment.char="#")
domains$exonID <- sub(":", "E", domains$exonID)
df <- df[which( rownames( df ) %in% domains$exonID ),]
```

```

back <- c()
for( i in 1:4 ){
  mmcts <- matchit(
    significant ~ mean + length + dispersion,
    df, distance="mahalanobis")
  back <- c(back, mmcts$match.matrix[,1])
  df <- df[-which( rownames(df) %in% back ),]
}
save(back, file="objects/back-coding.RData")

```

## 15 Figures

### 15.1 Figure 1B: visualization of REUCs in a CTDU case

A shingle plot with an example of CTDU exons, and a representation of a species specific exon usage case. We first prepare the data:

```

geneIDs <- sapply(
  strsplit( dimnames(crossCoefs)[[1]], ":" ), "[(", 1)

getGeneExonLevelDF <- function( gn, zlim=2.5 ) {

  x <- crossCoefs[ which(geneIDs %in% gn), , ]
  zlim=2.5
  # Makes a shingles plot from the a cross-coefficients array

  dimCC <- dim(x)
  names( dimCC ) <- c("exon", "species", "tissue")
  names( dimnames(x) ) <- c("exon", "species", "tissue")
  dimnames(x)$exon <- sub("\\S+:", "", dimnames(x)$exon)

  crossCoefsDF <- data.frame(
exon = factor(
  rep( dimnames(x)$exon,
        each = dimCC["species"] * dimCC["tissue"] ) ),
  levels = c( dimnames(x)$exon ),
species = factor(
  rep( rep( dimnames(x)$species,
            each=dimCC["tissue"] ), dimCC["exon"] ),
  levels = dimnames(x)$species ),

```

```

tissue = factor(
  rep( dimnames(x)$tissue,
        dimCC["exon"]* dimCC["species"] ),
  levels = dimnames(x)$tissue ),
effect = NA_real_ )

for( ex in unique( crossCoefsDF$exon ) ){
  gex <- paste(gn, ex, sep=":")
  anyZero <- tapply( counts(ecs)[gex,],
    list(design(ecs)$specie, design(ecs)$tissue), sum ) == 0
  w <- which( anyZero, arr.ind=TRUE )
  if( length(w) > 0 ){
for( i in 1:nrow(w) ){
  x[ex,w[,"row"][i],w[,"col"][i]] <- -100
}
}
}

for( r in 1:nrow(crossCoefsDF) )
crossCoefsDF$effect[r] <- x[ as.character(crossCoefsDF$exon[r]),
  as.character(crossCoefsDF$species[r]),
  as.character(crossCoefsDF$tissue[r]) ]

crossCoefsDF$effect <- pmin( zlim,
  pmax( -zlim, crossCoefsDF$effect ) )
crossCoefsDF[which( is.na( crossCoefsDF$effect ) ),"effect"]
<- -zlim

cbind( crossCoefsDF, geneID=gn )
}

colors <- colorRampPalette(brewer.pal(3, name="YlGnBu"))(100)
colors <- c("#F8F8F8", colors)

forcts <- getGeneExonLevelDF( "ENSG00000171992" )
forcts <- forcts[forcts$exon %in% tail( levels( forcts$exon ), 5 ),]
forcts$exon <- factor( forcts$exon )
levels(forcts$geneID) <- "SYNPO"

```



```

forcts$species <- factor( forcts$species,
  levels=c("hsa", "ppa", "ptr", "ggo", "ppy", "mml"))

forss <- getGeneExonLevelDF( "ENSG00000136628" )
forss <- forss[forss$exon %in% levels( forss$exon )[2:6],]
forss$exon <- factor( forss$exon )
levels( forss$geneID ) <- "EPRS"
forss$species <- factor( forss$species,
  levels=c("hsa", "ppa", "ptr", "ggo", "ppy", "mml"))

lp <- levelplot(
  effect ~ tissue * species | exon + geneID,
  forcts,
  col.regions=colors,
  as.table=TRUE,
  strip=strip.custom(bg="#E8E8E8"),
  at=seq( -2.5, 2.5, length.out=100 ),
  scales=list(x=list(cex=1.3,
    alternating=c(2, 0, 2, 0, 2, 0, 2, 0)),
    y=list(cex=1.3, alternating=2 ) ),
  ylab=list(label=""),
  xlab=list(label=""),
  colorkey=NULL,
  par.strip.text=list(cex=1.3),
  between=list(x=0, y=1),
)
pr1 <- useOuterStrips( lp,
  strip=strip.custom(bg=c("#E8E8E8", "#E8E8E8") ),
  strip.left=strip.custom(bg=c("#E8E8E8", "#E8E8E8") )
)

lp <- levelplot(
  effect ~ tissue * species | exon + geneID,
  forss,
  col.regions=colors,
  as.table=TRUE,
  at=seq( -2.5, 2.5, length.out=100 ),
  strip=strip.custom(bg="#E8E8E8"),

```

```

scales=list(x=list(cex=1.3,
  alternating=c(0, 1, 0, 1,0, 1, 0, 1)),
  y=list(cex=1.3, alternating=2 ) ),
ylab=list(label=""),
xlab=list(label=""),
colorkey=NULL,
par.strip.text=list(cex=1.3),
between=list(x=0, y=1),
)
pr2 <- useOuterStrips( lp,
  strip=strip.custom(bg=c("#E8E8E8", "#E8E8E8") ),
  strip.left=strip.custom(bg=c("#E8E8E8", "#E8E8E8") ))

```

Warning message:

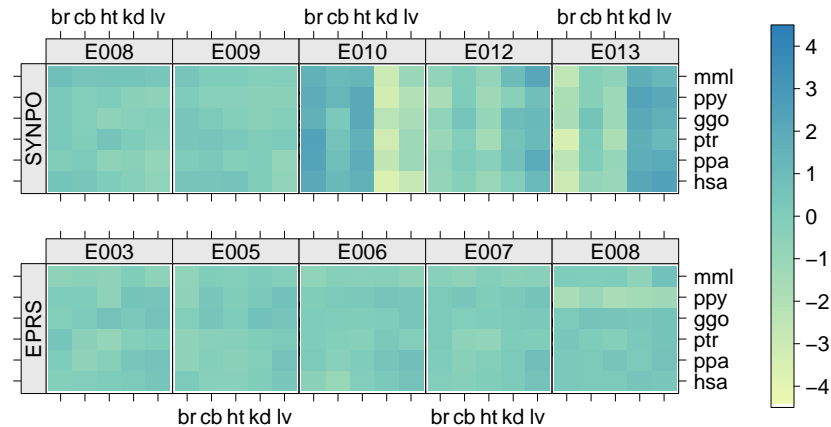
In max(i) : no non-missing arguments to max; returning -Inf

Now we plot it in a shingle matrix heatmap representation, to make figure 1B:

```

print( pr1, position=c(0, 0.45, 0.9, 1), more=TRUE )
print( pr2, position=c(0, 0, 0.9, 0.55), more=TRUE )
draw.colorkey(
  key=list(
    col=colors,
    at=seq( -4.5, 4.5, length.out=100), labels=list(cex=1.3)
  ),
  draw=TRUE,
  vp=grid::viewport(
    x=grid::unit(0.95, "npc"),
    y=grid::unit(0.5, "npc"),
    height=grid::unit(0.8, "npc")
  )
)

```



## 15.2 Figure 2A: PCA analysis

Principal component analysis of REUCs with the full data set and two subsets restricting to specific species or tissues. A comparison to gene expression levels was also done:

```
load("objects/allEffects.RData")
sds <- apply( allEffects, 1, sd, na.rm=TRUE)
threshold <- quantile(sds,
  probs=seq(0, 1, 0.1), na.rm=TRUE)["90%"]
newany <- names( which(sds > threshold) )
allEffects <- allEffects[rownames(allEffects) %in% newany,]

pca <- prcomp( t( allEffects ) )
cmp <- rownames(pca$x)
sp <- sapply( strsplit( cmp, "_" ), "[", 1)
ts <- sapply( strsplit( cmp, "_" ), "[", 2)
fullNames <-
  c("gorilla", "human", "rhesus monkey", "bonobo", "orangutan", "chimpanzee")
names(fullNames) <-
  c("ggo", "hsa", "mml", "ppa", "ppy", "ptr")
fullNamesTS <- c("brain", "cerebellum", "heart", "kidney", "liver")
names(fullNamesTS) <- c("br", "cb", "ht", "kd", "lv")
pcaData <- data.frame(
  PC1=pca$x[,"PC1"], PC2=pca$x[,"PC2"],
  which="Exon usage (all)",
```

```

species=fullNames[sp],
tissue=fullNamesTS[ts])

allEffects2 <- allEffects[,!grepl("br|cb|mml", colnames( allEffects ))]
pca <- prcomp( t( allEffects2 ) )
cmp <- rownames(pca$x)
sp <- sapply( strsplit( cmp, "_" ), "[", 1)
ts <- sapply( strsplit( cmp, "_" ), "[", 2)
pcaData <- rbind( pcaData,
  data.frame(
    PC1=pca$x[, "PC1"],
    PC2=pca$x[, "PC2"],
    which="Exon usage (subset 1)",
    species=fullNames[sp],
    tissue=fullNamesTS[ts]))

allEffects3 <- allEffects[,!grepl("mml|cb|br|ht|ppy", colnames( allEffects ))]
pca <- prcomp( t( allEffects3 ) )
cmp <- rownames(pca$x)
sp <- sapply( strsplit( cmp, "_" ), "[", 1)
ts <- sapply( strsplit( cmp, "_" ), "[", 2)
pcaData <- rbind( pcaData,
  data.frame(
    PC1=pca$x[, "PC1"],
    PC2=pca$x[, "PC2"],
    which="Exon usage (subset 2)",
    species=fullNames[sp],
    tissue=fullNamesTS[ts]))

load("inputs/ecsTRT.RData")
geneCounts <- geneCountTable( ecs )
cds <- newCountDataSet( as.matrix(geneCounts), conditions=design(ecs))
cds <- estimateSizeFactors( cds )
norCounts <- counts(cds, normalized=TRUE)

forPCA <- sapply( rownames( norCounts ), function(gn){
  singleGene <- tapply(norCounts[gn,],
    INDEX=list( pData(cds)$specie, pData(cds)$tissue ),
    mean)

```

```

singleGeneVec <- as.vector( singleGene )
names(singleGeneVec) <- as.vector(
  sapply(colnames(singleGene),
    function(x){
      paste(x, rownames(singleGene), sep="_")
    } ) )
singleGeneVec
})

forPCA <- t( forPCA )
forPCA <- log( forPCA )
forPCA[ which( is.infinite( forPCA ) ) ] <- NA
NAS <- apply( forPCA, 1, function(x){!any(is.na(x))})
forPCA <- forPCA[NAS,]

thesholdExp <- quantile(apply( forPCA, 1, sd ),
  seq(0, 1, 0.1))["90%"]
forPCA <-
  forPCA[which( apply( forPCA, 1, sd ) > thesholdExp ),]
pca <- prcomp( t( forPCA ) )
cmp <- rownames(pca$x)
sp <- sapply( strsplit( cmp, "_" ), "[", 2)
ts <- sapply( strsplit( cmp, "_" ), "[", 1)
pcaData <- rbind(data.frame(
  PC1=pca$x[, "PC1"],
  PC2=pca$x[, "PC2"],
  which="Gene expression",
  species=fullNames[sp],
  tissue=fullNamesTS[ts]), pcaData)
save(pcaData, file="objects/pcaData.RData")

pcaData$species <- factor(pcaData$species,
  levels=c("human", "bonobo",
    "chimpanzee", "gorilla", "orangutan", "rhesus monkey"))
signs <- c("B", "C", "H", "K", "L")
colSpecies <- brewer.pal(7, "Set1")[-6]

pcaData[pcaData[,3] == "Gene expression",c(1,2)] <-
  pcaData[pcaData[,3] == "Gene expression",c(1,2)] * 0.5

```

We plot the different PCA panels in the same panel

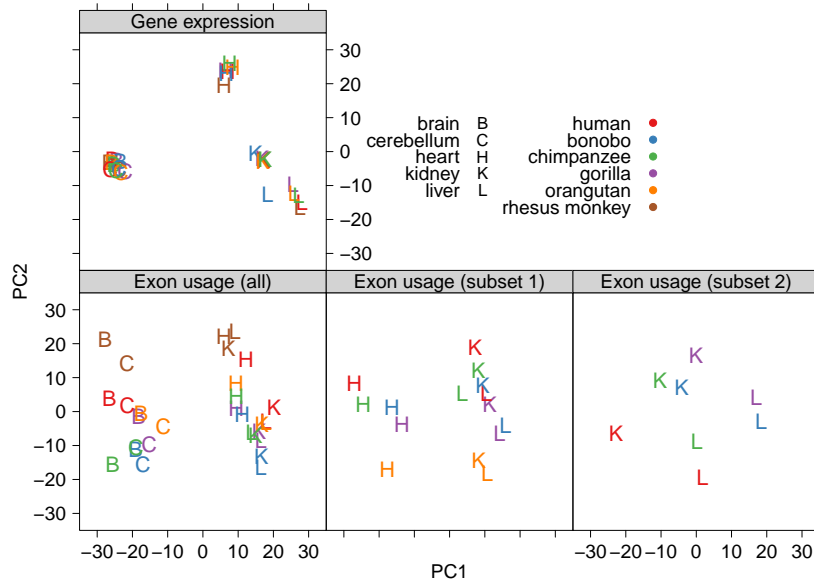
```
theme.novpadding <-
  list(layout.heights =
list(top.padding = 0,
      main.key.padding = 0,
      key.axis.padding = 0,
      axis.xlab.padding = 0,
      xlab.key.padding = 0,
      key.sub.padding = 0,
      bottom.padding = 0.1),
layout.widths =
list(left.padding = 0.2,
      key.ylab.padding = 0,
      ylab.axis.padding = 0,
      axis.key.padding = 0,
      right.padding = 0))

with( pcaData,
      xyplot( PC2 ~ PC1 | which, layout=c(3,2), cex=1,
              panel=function(x, y, ..., subscripts){
                pch<-signs[tissue[subscripts]];
                col=colSpecies[species[subscripts]];
                panel.xyplot(x, y, pch=pch, col=col, cex=2.3 )
              },
              xlim=c(-35, 35),
              ylim=c(-35, 35),
              index.cond=list(c(2, 3, 4, 1)),
              ylab=list(label="PC2", cex=1.3),
              xlab=list(label="PC1", cex=1.3),
              scales=list(x=list(cex=1.3), y=list(cex=1.3)),
              par.strip.text=list(cex=1.3),
              strip=strip.custom(bg="lightgray"),
              par.settings = theme.novpadding,
              panel.error=NULL,
              key=list(
                x=0.38, y=0.80, adj=1,
                text=list( levels(tissue), cex=1.3 ),
                points = list( pch=signs, cex=1.7 ),
                text = list( levels(species), cex=1.3),
```

```

    points = list( pch=20, col=colSpecies, cex=1.5),
    rep=FALSE)
  )
)

```



### 15.3 Figure 2B: variance analysis

We plotted the variance explained by tissues vs the variance explained by species, as in figure 2B:

```

gamma <- .3
par(mar=c(5, 5, 1, 1))
plot(
  asinh( msq["tissue"] / log(2)^2 / gamma ),
  asinh( msq["species"] / log(2)^2 / gamma ),
  pch=16, cex=1, xaxt="n", yaxt="n", asp=1,
  xlab="variance explained by tissue",
  ylab="variance explained by species",
  col=ifelse( rowSums(sppCovs>covThr) == 15, "#FF000030",
    ifelse( sppCovs["hsa:mml"] > covThr, "#FF00FF30", "#00000030" ) ), cex.lab=2)
ticks <- c( 0, .1*(1:9), 1:10, 20 )
longTicks <- c( 0, .1, 1, 10 )
for( ax in 1:2 ) {

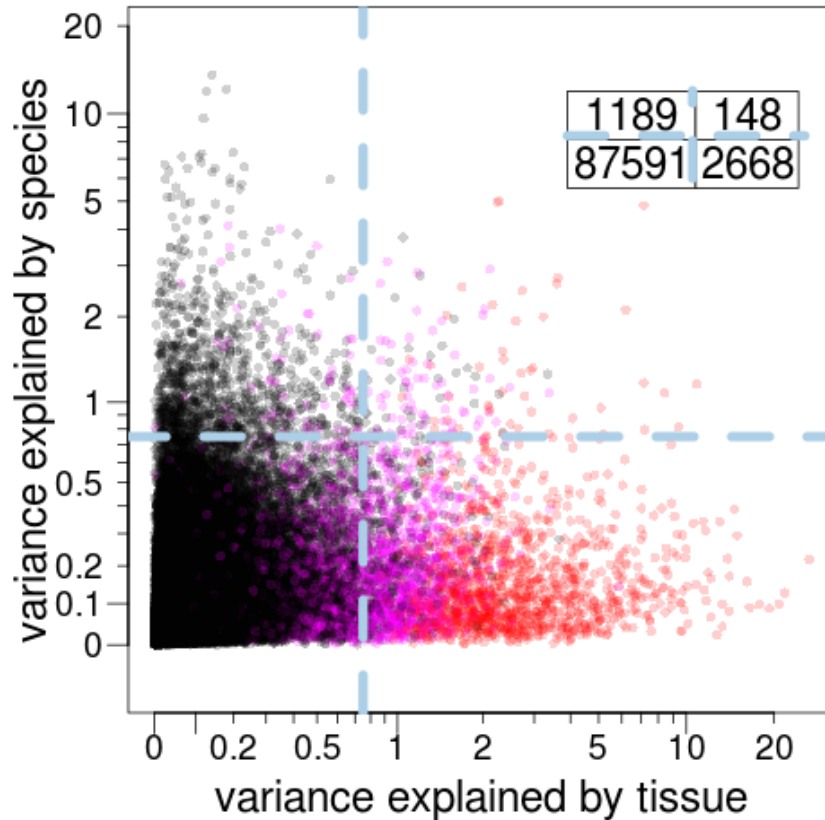
```

```

labeledTicks <- c( 0, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20 )
axis( ax,
      asinh( labeledTicks / gamma ),
      labeledTicks, tcl=0, las=1, cex.axis=1.6)
axis( ax,
      asinh( c( 0, .1, 1, 10 ) / gamma ),
      FALSE, tcl=-.8, cex.axis=1.6)
axis( ax, asinh( c( .1*(2:9), 2:9, 20 ) / gamma ),
      FALSE, tcl=-.4, cex.axis=1.6) }
abline( h = asinh(0.75/gamma),
        v = asinh(0.75/gamma),
        col="#ACCEE6", lty="dashed",
        lwd=7)
mat <- as.matrix (
  table(
    asinh( msq[,"species"] / log(2)^2 / gamma )
    > asinh(0.75/gamma),
    asinh( msq[,"tissue"] / log(2)^2 / gamma )
    > asinh(0.75/gamma)) ) [2:1,]
addtable2plot(x=asinh(13), y=asinh(18.5), table=mat,
  display.colnames=FALSE, cex=2, lwd=0,
  hlines=FALSE, vlines=FALSE, box.col="blue")
segments(asinh(35), asinh(20),
  asinh(35), asinh(40), lty="dashed", lwd=7, col="#ACCEE6" )
segments(asinh(13),
  asinh(28), asinh(85), asinh(28),
  lty="dashed", lwd=7, col="#ACCEE6" )

```





As in figure S3, we plot the histogram of the ratio between the variance explained by tissue and the variance explained by species:

```

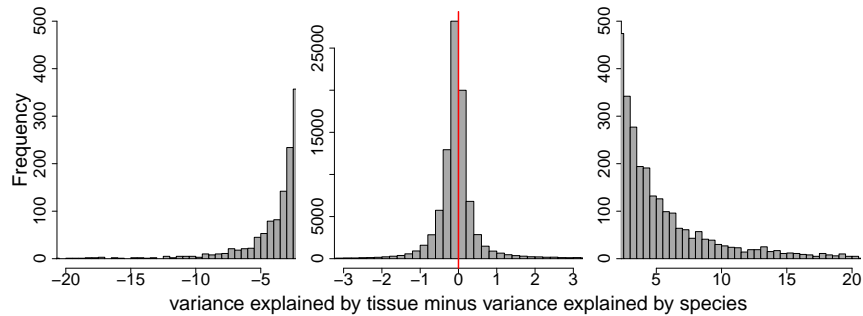
par(mfrow=c(1, 3))
par(mar=c(6, 5, 1, 0.1))
hist(
  (msq["tissue"] / log(2)^2/0.3 ) -
  ( msq["species"] / log(2)^2 /0.3 ),
  breaks=450, xlim=c(-20, -3), cex.axis=2.2, main="",
  ylim=c(0, 500), xlab="", cex.lab=2.5, col="darkgray")
par(mar=c(6, 4, 1, 0.1))
hist(
  (msq["tissue"] / log(2)^2/0.3 ) -
  ( msq["species"] / log(2)^2 /0.3 ),
  breaks=900, xlim=c(-3, 3), cex.axis=2.2, main="", ylab="",
  xlab="", col="darkgray")

```

```

text(x=0, y=-5500,
     label="variance explained by
           tissue minus variance explained by species",
     xpd=NA, cex=2.5)
abline( v=0, col="red", lwd=2)
par(mar=c(6, 4, 1, 1))
hist( (msq[,"tissue"] / log(2)^2/0.3 ) -
      ( msq[,"species"] / log(2)^2 /0.3 ),
      breaks=450, xlim=c(3, 20), cex.axis=2.2,
      main="", ylim=c(0, 500),
      ylab="", xlab="", col="darkgray")

```



#### 15.4 Figure 2C: conserved tissue-specific vs mya

We plot the number of CTDU exons between human and the rest of the primates vs divergence time in million years (according to Israfil et al, 2011). We do some data preparation:

```

spPairsDist <- c( 8.6, 33.6, 8.6,
                 17.5, 8.6, 33.6, 6.1, 17.5, 6.1,
                 33.6, 33.6, 33.6, 17.5, 3.9, 17.5 )
names( spPairsDist ) <- colnames( sppCovs )
human <- sppCovs[,which( grepl("hsa", colnames( padjCov )) )]
numb <- colSums( human > covThr, na.rm=TRUE )
extra <- numb[c("hsa:ppa", "ggo:hsa")]
good <- numb[c("hsa:ptr", "ggo:hsa", "hsa:ppy", "hsa:mml")]

```

Now we can do the plot from figure 2B. In which we can see that the number of CTDU exons decreases with evolutionary time.

```

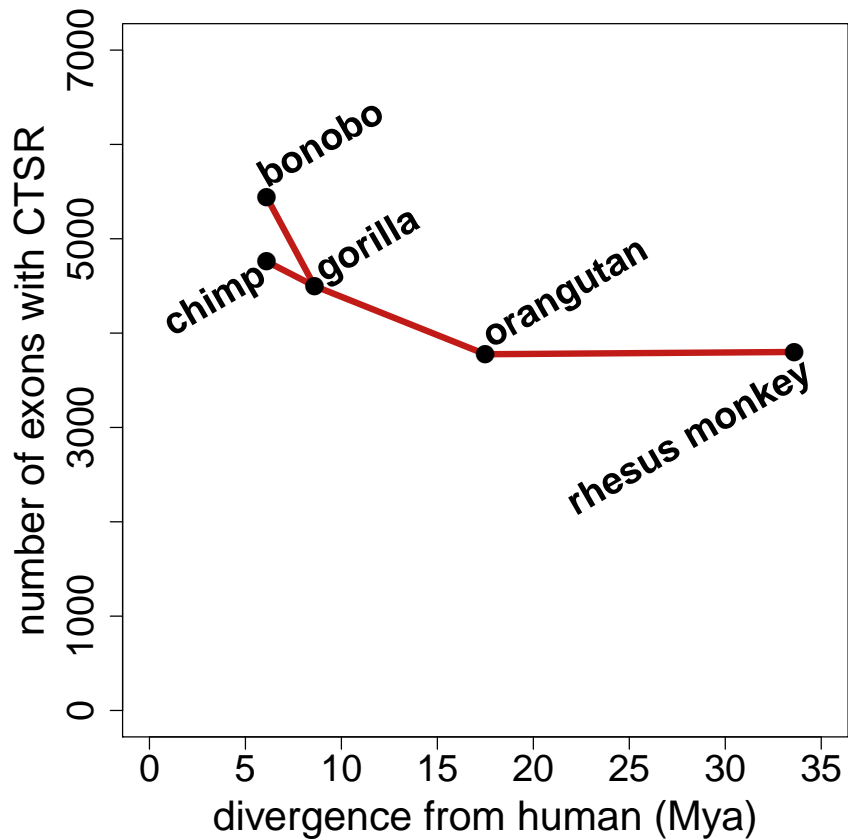
par(mar=c(5, 5, 1, 0.5))

```

```

plot( spPairsDist[names(good)],
      good, lwd=5, cex.lab=2, ylab="number of exons with CTDU",
      xlab="divergence from human (Mya)",
      cex.axis=1.8, xlim=c(0, 35),
      ylim=c(0, 7000), type="l", col="#C11B17")
lines( spPairsDist[names(extra)],
      extra, type="l", lwd=5, col="#C11B17")
points( spPairsDist[names(numb)], numb, cex=2, pch=16)
text(spPairsDist[names(extra)[1]],
     extra[1]+250, label="bonobo", adj=0.05,
     cex=1.8, srt=30, font=2 )
text(spPairsDist[names(good)[1]]-4.9,
     good[1]-680, label="chimp", adj=0.05,
     cex=1.8, srt=30, font=2 )
text(spPairsDist[names(good)[2]]+0.5,
     good[2]+150, label="gorilla",
     adj=0.05, cex=1.8, srt=30, font=2 )
text(spPairsDist[names(good)[3]]+0.5, good[3]+190,
     label="orangutan", adj=0.05, cex=1.8, srt=30, font=2 )
text(spPairsDist[names(good)[4]]-11,
     good[4]-1580, label="rhesus monkey",
     adj=0.05, cex=1.8, srt=30, font=2 )

```



### 15.5 Figure 2D: Tissue strength vs conservation

In order to see the relation between conservation of the regulation and strength of tissue-dependent usage, we plot these two variables. This results in figure 2D.

```

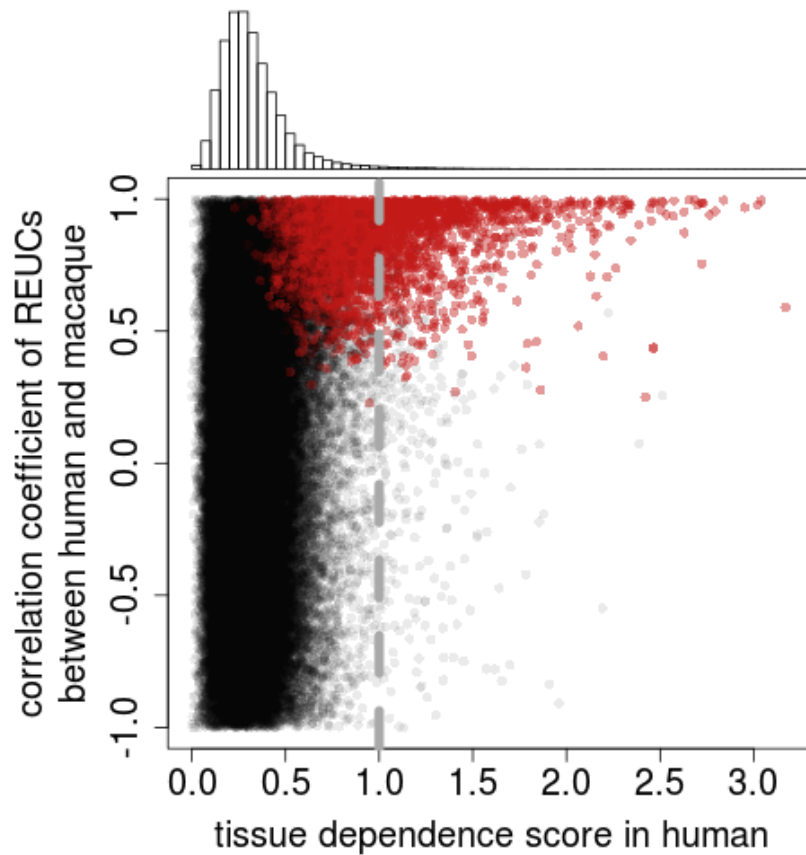
layout(mat=as.matrix(c(1, 2), ncol=2),
       heights=c(1, 4), widths=c(1, 1))
par(mar=c(0, 8, 0.1, 0.3))
hist(abs(maxDevSp[,"hsa"])/log(2),
     100, xlim=c(0, 3.2), xaxt="n",
     main="", cex.axis=1.8, cex.lab=1.8,
     ylab="", yaxt="n", lwd=3)
par(mar=c(4.5, 8, 0.1, 0.3))
cols <- ifelse( sppCovs[,"hsa:mml"] < covThr,

```

```

"#00000015", "#C11B1770")
plot(
  abs(maxDevSp[,"hsa"])/log(2),
  sppCors[,"hsa:mml"], pch=16,
  cex=1, cex.lab=1.65, cex.axis=1.65,
  xlim=c(0, 3.2),
  ylab="correlation coefficient of REUCs
  \nbetween human and macaque",
  xlab="tissue dependence score in human",
  col=cols, lwd=3)
abline(v=1, col="darkgray", lty=2, lwd=7)

```



## 15.6 Figure 3D: heatmap of tissue dependence

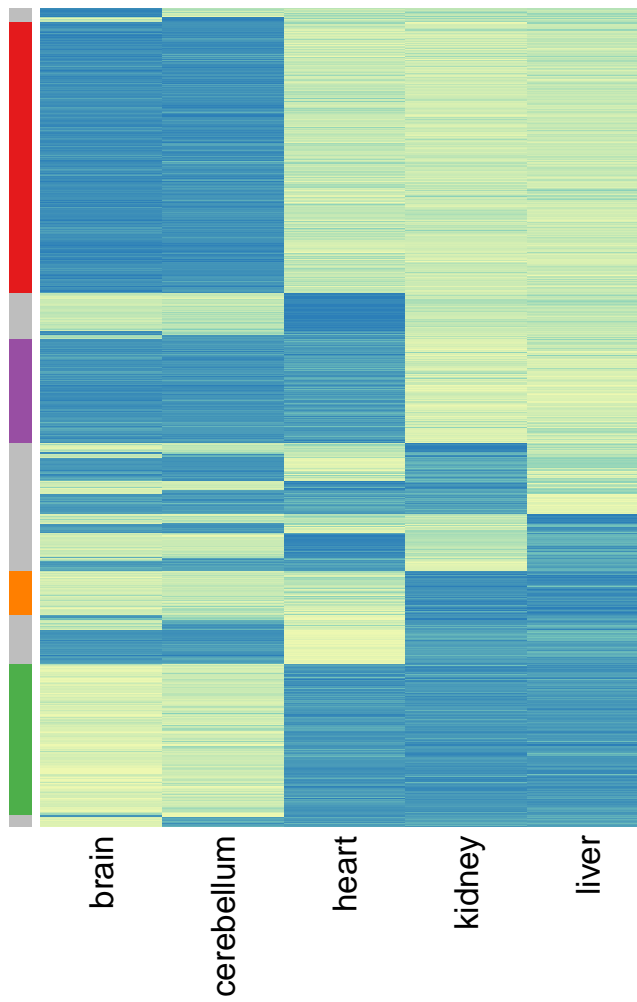
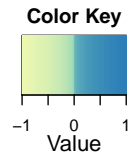
We classify our set of strictly conserved exons according to their mean REUC across species. We represent this in a heatmap, which we present in figure 3.

```
load("objects/allEffects.RData")
load("objects/ctsr.RData")
meanTiCoefs <- apply( crossCoefs, c( 1, 3 ), mean )
forClust <- meanTiCoefs[ctsr,]
tiDirs <- do.call( expand.grid,
  sapply( colnames(meanTiCoefs),
    function(ti) c( -1, 1 ),
    simplify=FALSE ) )
save( tiDirs, file="objects/classesDefinition.RData")
tiDirClass <- apply(
  meanTiCoefs[ ctsr, ] %*% t( as.matrix(tiDirs) ),
  1, which.max )
save(tiDirClass, file="objects/CDUT-class.RData")
sortedClasses <- sort( tiDirClass )
forClust <- forClust[names( sort( tiDirClass ) ),]
colors <- colorRampPalette(
  brewer.pal(3, name="YlGnBu"))(100)
colors2 <- brewer.pal(n=5, "Set1")
colors2 <- colors2[-2]
cols <- rep("gray", nrow(forClust))
goodClasses <- names( head( sort(
  sapply( split( names( sortedClasses ), sortedClasses ),
    length ), decreasing=TRUE ), 4 ) )
goodClasses <- as.numeric( goodClasses )
for( i in 1:4 ) {
  cols[which( sortedClasses == goodClasses[i] )] <-
    colors2[i]
}
```

We make the heatmap

```
heatmap.2( forClust, col=colors,
  breaks=seq( -1, 1, length.out=101)^3,
  trace="none", dendrogram="none", Rowv=FALSE,
  keysize=1.3, labRow=rep("", nrow(forClust)),
```

```
labCol=c("brain", "cerebellum", "heart", "kidney", "liver"),
densadj=1, Colv=FALSE, margins=c(10, 1), RowSideColors=cols,
lhei=c(2, 10), cexCol=2.3, density.info="none")
```



## 15.7 Figure 3A: disordered regions

TDU is known to be enriched in protein disordered regions that are often mediate protein-protein interactions. We verify this in our set of CTDU exons and strengthen this result.

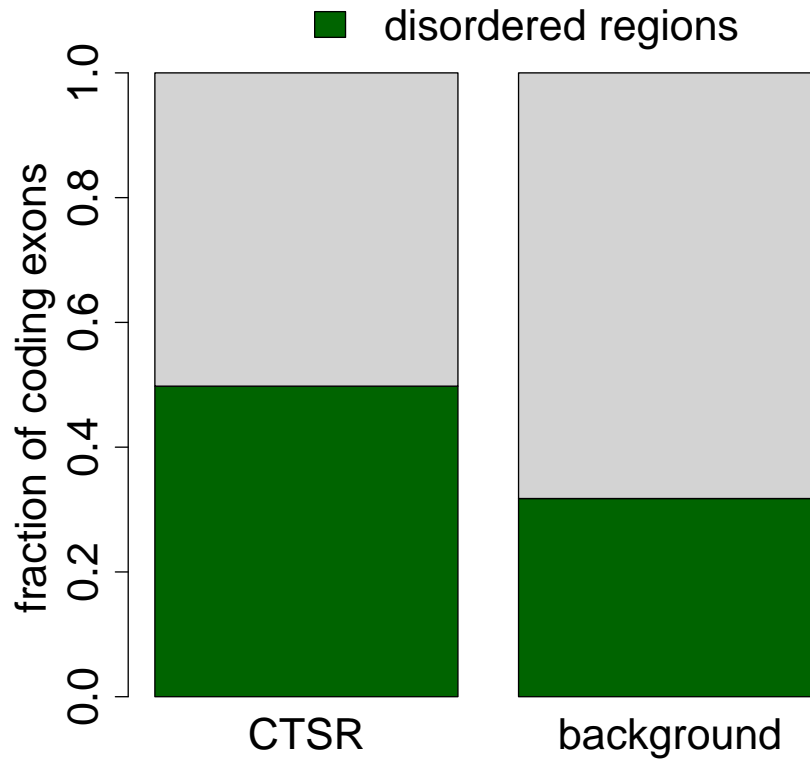
We first load and prepare all the relevant data. Including the predictions from IUPRED.

```
protFeatures <- read.table(
  "inputs/mappedExons_proteinscores.txt",
  header=TRUE, stringsAsFactors=FALSE)
rownames( protFeatures ) <-
  gsub(":", ":E", as.character(protFeatures$exonID))
load("objects/ctsr.RData")
load("inputs/ecsTRT.RData")
load("objects/CDUT-class.RData")
load("objects/back-coding.RData")
goodmat <- rbind(
  ctsr=table(
    protFeatures
    [which(rownames(protFeatures) %in% ctsr),
    "IUPred_disorder.0.4"] ),
  back=table(
    protFeatures
    [which(rownames(protFeatures) %in% back),
    "IUPred_disorder.0.4"] ))[,2:1]
goodData <- t( goodmat / rowSums( goodmat ) )
```

We can then do a barchart to see how it compares to our set of CTDU exons.

```
par(mar=c(3, 5, 4, 0.5))
barplot(goodData,
  col=c("darkgreen", "lightgray"),
  names=c("CTSR", "background"),
  cex.axis=2, ylab="fraction of coding exons",
  cex.lab=2, xpd=FALSE, cex.names=2)
legend(x=0.6, y=1.16,
  legend="disordered regions",
  fill="darkgreen", xpd=TRUE,
  cex=2, box.lty=0)
```





15.8 Figure 3B: venn diagram

```

load("inputs/ecstrt.RData")
load("inputs/utrRanges.RData")
codexons <- read.table("inputs/frames.txt", header=FALSE)
coding <- unique( sub(":", ":E", codexons$V1) )

exons <- GRanges(
  seqname=fData(ecs)$chr,
  ranges=IRanges(start=fData(ecs)$start,
    end=fData(ecs)$end, names=featureNames(ecs)),
  strand=fData(ecs)$strand)
fiveExons <- unique(
  names( exons[queryHits(
    findOverlaps( exons, fiveRanges ) ] ] ) )
threeExons <- unique(

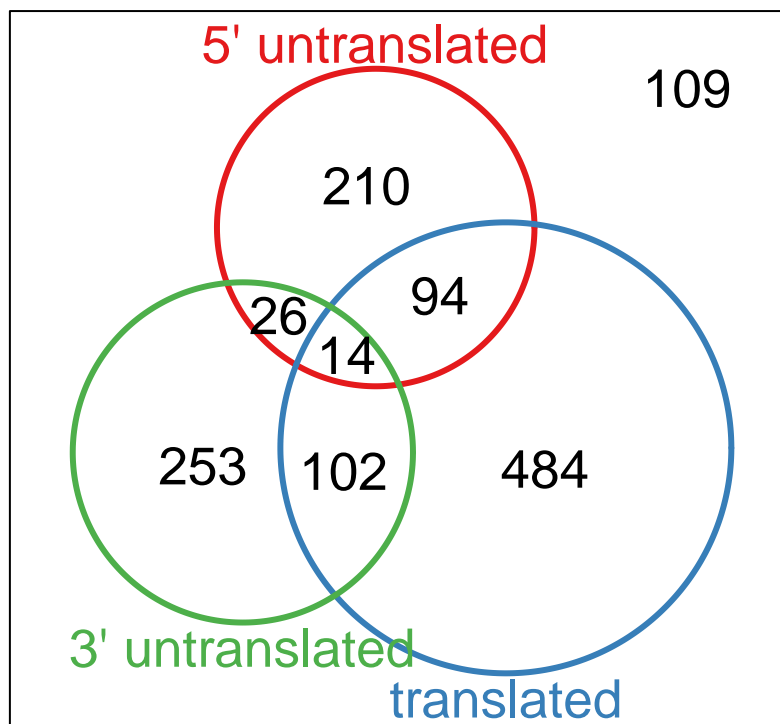
```

```

names( exons[queryHits(
  findOverlaps( exons, threeRanges ) ] ) )
load("objects/ctsr.RData")

Vtypes <- Venn(
  list(
    'all'=ctsr, '5' UTR'=ctsr[ctsr %in% fiveExons],
    'coding exon'=ctsr[ctsr %in% coding],
    '3' UTR'=ctsr[ctsr %in% threeExons] ))
Vtypes <- Vtypes[,2:4]
colnames( Vtypes@IndicatorWeight )[1:3] <-
  c("5' untranslated", "translated", "3' untranslated")
plot( Vtypes, show=list(Faces=FALSE) )

```



## 15.9 Figure 3C: CTDU barchart in different categories and comparison with background set of exons

```
load("inputs/ecsTRT.RData")
exons <- GRanges(
  seqnames=fData(ecs)$chr,
  ranges=IRanges(
    fData(ecs)$start,
    fData(ecs)$end,
    names=rownames(fData(ecs)) ),
  strand=fData(ecs)$strand )
load("objects/back-exon.RData")
load("objects/ctsr.RData")

makeContTable <- function(utrs, exons ){
  a <- length( intersect( exons, utrs ) )
  b <- length( intersect( back, utrs ) )
  mat <- rbind(
    fore=c( UTR=a, nonUTR=abs( a-length( ctrs ) ) ),
    back=c( UTR=b, nonUTR=abs(b - length( back ) ) )
  )
  return( mat )
}

load("inputs/codingornot.RData")
utr3 <- makeContTable( goodthreeExons, ctrs )
utr5 <- makeContTable( goodfiveExons, ctrs )
coding <- makeContTable( goodcoding, ctrs )

fisher.test( rbind( utr3[,1], utr5[,1], coding[,1]) )

nums <- c( (utr5[,1] / rowSums( utr5 )),
  c(coding[,1]/rowSums(coding)),
  (utr3[,1] / rowSums( utr3 ) ) )
df <- data.frame(
  fraction=nums, set=names(nums),
  region=
  c("5' UTR", "5' UTR",
    "translated", "translated",
    "3' UTR", "3' UTR" ) )
```

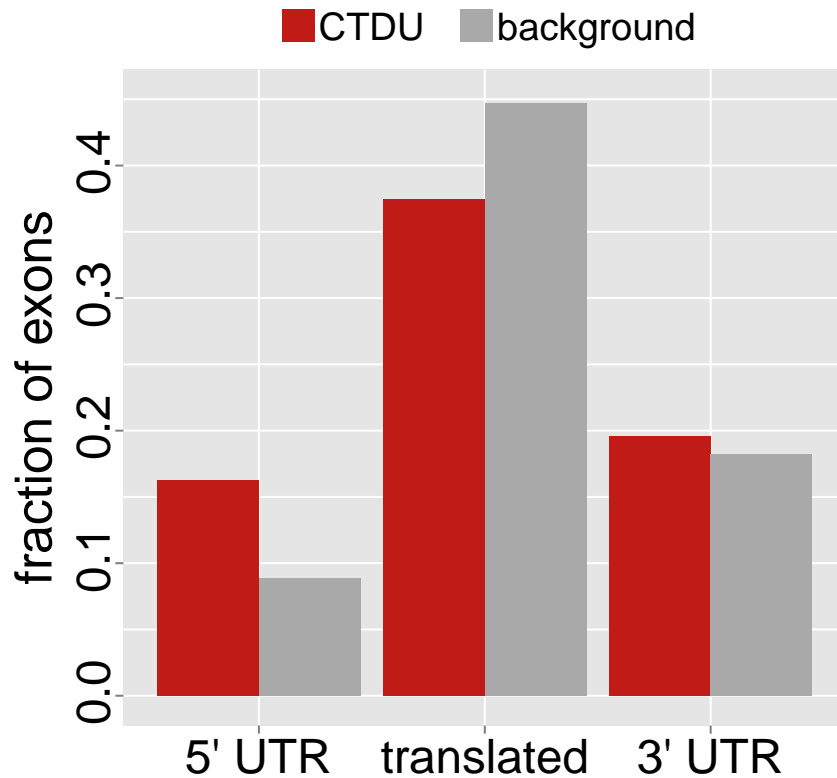
```
df$region <- factor(
  df$region,
  levels=levels(df$region)[c(2,3,1)] )
df$set <- factor(
  df$set,
  levels=c("fore", "back"))
```

#### Fisher's Exact Test for Count Data

```
data: rbind(utr3[, 1], utr5[, 1], coding[, 1])
p-value = 2.697e-14
alternative hypothesis: two.sided
```

We plot the result in a barplot

```
bp <- qplot(
  region, data=df, geom="bar",
  fill=set, weight=fraction,
  position="dodge" ) +
  scale_fill_manual(name="",
    values=c("#C11B17", "darkgray"),
    labels=c("CTDU ", "background")) +
  scale_x_discrete("") +
  theme(
    axis.title.y = element_text(size=27, vjust=.3),
    axis.text.x=element_text(size=25, colour="black"),
    axis.text.y=element_text(size=25, colour="black",
    angle=90, hjust=0.45),
    legend.text=element_text(size=20, colour="black"),
    legend.position="top", legend.direction="horizontal",
    panel.grid.minor=element_line(size=0.5, color="white"))
  + scale_y_continuous(limits=c(0, .45) )
bp$labels$y <- "fraction of exons"
bp
```



### 15.10 Figure 3E: splicing factor motif enrichments

For each of the groups we calculate the ratio of the number of predicted binding motifs for each of the splicing factors. We also calculate 95% confidence intervals.

```
load("inputs/ecstrt.RData")
load("objects/CDUT-class.RData")
load("objects/classesDefinition.RData")
goodCategories <- names(
  head( sort(table( tiDirClass ), decreasing=TRUE), 4) )
classes <- split( names( tiDirClass ), tiDirClass )

sf <- read.table(
  "inputs/sfmapTable.tab", header=TRUE, row.names=1)
```

```

sf <- split(sf,
  sapply( strsplit( rownames( sf ), "-" ), "[", 1))
rn <- sapply( strsplit( rownames(sf[[1]]), "-"), "[", 2 )
rn <- sapply( strsplit( rownames(sf[[1]]), "-"), "[", 2 )
rownames( sf[[1]] ) <- rn
rownames( sf[[2]] ) <- rn
sf <- sf[[1]] + sf[[2]]
load("objects/ctsr.RData")
load("objects/back-exon.RData")
goodCategories <- names(
  head( sort(table( tiDirClass ), decreasing=TRUE), 4) )

exons <- classes[[goodCategories[1]]]

confidences <- mclapply( goodCategories, function(x){
  exons <- classes[[x]]
  exonRows <- which( rownames(sf) %in% exons )
  backRows <- which( rownames(sf) %in% back )
  foreboot <- replicate(10000,
    colMeans( sf[sample( exonRows, replace=TRUE),] ) )
  backboot <- replicate(10000,
    colMeans( sf[sample( backRows, replace=TRUE),] ) )
  real <- log2(colMeans( sf[exonRows,] )
    / colMeans( sf[backRows,] ))
  conf <- t( sapply(
    rownames(foreboot), function(x){
      quantile( log2( foreboot[x,] / backboot[x,] ),
        c(0.05, 0.95) ) } ) )
  pval <- sapply(colnames(sf), function(x){
    wilcox.test( sf[backRows,x], sf[exonRows,x] )$p.value } )
  res <- cbind( real, conf, pval )
  return( res )
}, mc.cores=1 )

confidences <- lapply(confidences,
  function(x){
    x[-which(rownames(x) %in% c("SRp55","hnRNPAB")),] } )

df <- confidences[[1]]
num <- nrow( df )

```

```

colors2 <- brewer.pal(n=5, "Set1")
colors2 <- colors2[-2]
cols <- colors2
goodNames <- gsub("X9G8",
  "9G8",
  gsub("SF2ASF",
    "ASF/SF2",
    gsub("G\\.B", "G-B",
      gsub("H\\.F", "H/F",
        rownames(confidences[[1]]))))))
# goodorder <- 1:18
goodorder <- c(11, 16, 1, 3, 6, 10, 12, 13, 15, 2, 8, 4, 14, 7, 9, 5)

goodorder <- order( confidences[[1]][,"real"], decreasing=TRUE )
confidences2 <- lapply(confidences,
  function(x) x[goodorder,] )
confidences2 <- lapply(confidences2,
  as.data.frame )
confidences2 <- lapply( seq(along=confidences2),
  function(x) cbind(confidences2[[x]], class=x) )
confidences2 <- lapply( seq(along=confidences2),
  function(x) cbind(confidences2[[x]],
    splicingFactor=goodNames[goodorder] ))
confidences2 <- lapply( 1:length(confidences2),
  function(x) cbind(confidences2[[x]],
    color=as.character(cols[x])) )

confidences2 <- do.call(rbind, confidences2)
confidences2$splicingFactor <- factor(
  confidences2$splicingFactor, levels=goodNames[goodorder])

```

After calculating what is necessary and doing some data ordering and formating, we can finally make a plot with this results:

```

prepanel.ci <- function(x, y, ly, uy,
  subscripts, ...)
{
  y <- as.numeric(y)
  ly <- as.numeric(ly[subscripts])

```

```

    uy <- as.numeric(uy[subscripts])
    list(ylim = range(y, uy, ly, finite = TRUE))
}

panel.ci <- function(x, y, ly, uy, cols,
  subscripts, pch = 16, ...)
{
  x <- as.numeric(x)
  y <- as.numeric(y)
  ly <- as.numeric(ly[subscripts])
  uy <- as.numeric(uy[subscripts])
  colorsA <- cols[subscripts]
  panel.arrows(x, ly, x, uy, col = colorsA,
length = 0, unit = "native",
angle = 90, code = 3, lwd=3)
  panel.xyplot(x, y, pch = pch, ...)
}

trellis.par.set(superpose.symbol=
  list(col=as.character(unique(confidences2$color))))
with( confidences2,
  dotplot( real ~ splicingFactor|class,
  index.cond=list(c(2, 4, 3, 1)),
  uy='95%',
  ly='5%',
  cols=as.character(color),
  groups=class,
  prepanel=prepanel.ci,
  panel=function(x, y, ...){
    panel.abline(v = unique( as.numeric(x) ), col="#D8BFD870" )
    panel.abline( h=0, col="grey", lwd=2)
    panel.superpose(x, y, ... )
  },
  panel.groups=panel.ci,
  pch=16,
  strip=FALSE,
  layout=c(1, 4),
  between=list(y=0.5),
  scales=list(x=list(rot=90, cex=1.2), y=list(cex=1.2)),
  ylab=list(

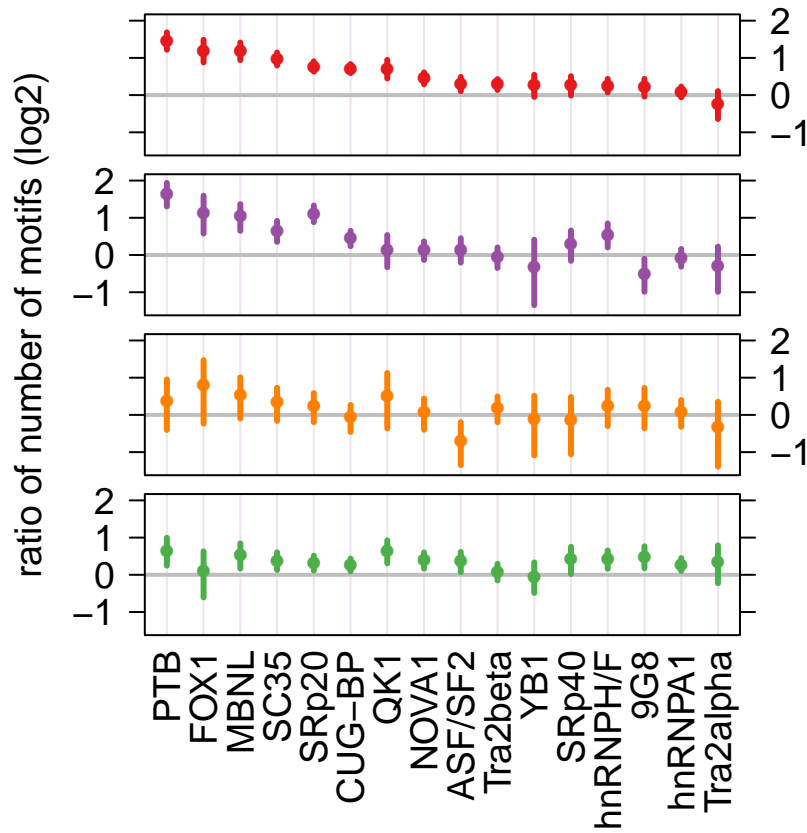
```



```

label="ratio of number of motifs (log2)", cex=1.2)
))

```



## 16 Session Info

```
sessionInfo()
```

```
R Under development (unstable) (2013-02-06 r61845)
Platform: x86_64-unknown-linux-gnu (64-bit)
```

```
locale:
```

```
[1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
```

```
[3] LC_TIME=en_US.UTF-8      LC_COLLATE=en_US.UTF-8
[5] LC_MONETARY=en_US.UTF-8  LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=C                LC_NAME=C
[9] LC_ADDRESS=C              LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
```

attached base packages:

```
[1] grid      parallel  stats      graphics  grDevices  utils      datasets
[8] methods  base
```

other attached packages:

```
[1] GenomicFeatures_1.12.2 GenomicRanges_1.12.4 IRanges_1.18.1
[4] ggplot2_0.9.3.1       Vennerable_2.2       xtable_1.7-1
[7] reshape_0.8.4         plyr_1.8              RBGL_1.36.2
[10] graph_1.38.2          gplots_2.11.3        KernSmooth_2.23-10
[13] caTools_1.14          gdata_2.13.2         plotrix_3.4-8
[16] geneplotter_1.38.0    annotate_1.38.0       AnnotationDbi_1.22.6
[19] MatchIt_2.4-21        DESeq_1.12.0         locfit_1.5-9.1
[22] latticeExtra_0.6-24   lattice_0.20-15      LSD_2.5
[25] ellipse_0.3-8         schoolmath_0.4       colorRamps_2.3
[28] RColorBrewer_1.0-5    gtools_3.0.0         MASS_7.3-27
[31] genefilter_1.42.0     abind_1.4-0          multicore_0.1-7
[34] statmod_1.4.17        DEXSeq_1.6.0         Biobase_2.20.1
[37] BiocGenerics_0.6.0    BiocInstaller_1.10.2
```

loaded via a namespace (and not attached):

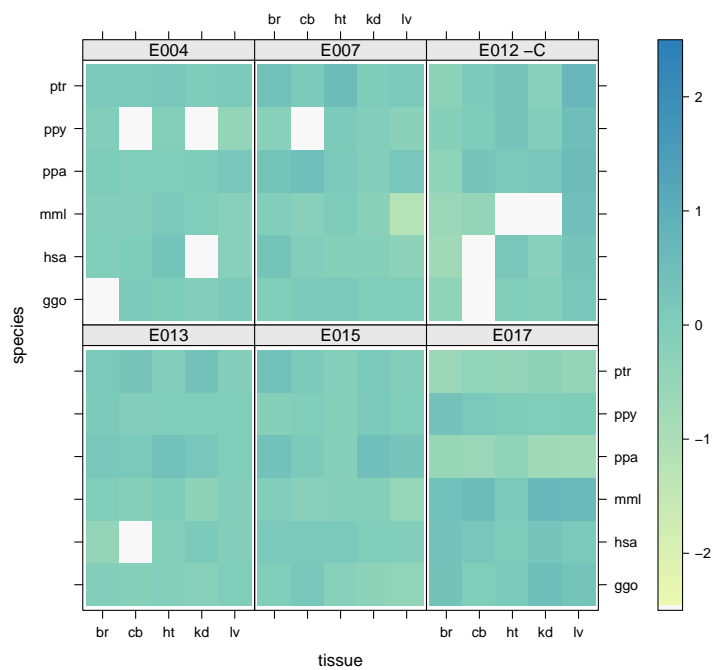
```
[1] biomaRt_2.16.0        Biostrings_2.28.0    bitops_1.0-5         BSgenome_1.28.0
[5] colorspace_1.2-2     DBI_0.2-7            dichromat_2.0-0      digest_0.6.3
[9] gtable_0.1.2         hwriter_1.3          labeling_0.2         munsell_0.4
[13] proto_0.3-10         RCurl_1.95-4.1       reshape2_1.2.2      Rsamtools_1.12.3
[17] RSQLite_0.11.4       rtracklayer_1.20.2   scales_0.2.3         splines_3.0.0
[21] stats4_3.0.0         stringr_0.6.2        survival_2.37-4      tools_3.0.0
[25] XML_3.98-1.1         zlibbioc_1.6.0
```

## 17 All genes with strictly CTDU exons.

In the next pages, the heatmap plots (as in Figure 1) of each gene that contained strictly CTDU exons are presented. Check Figure 1 for legends.

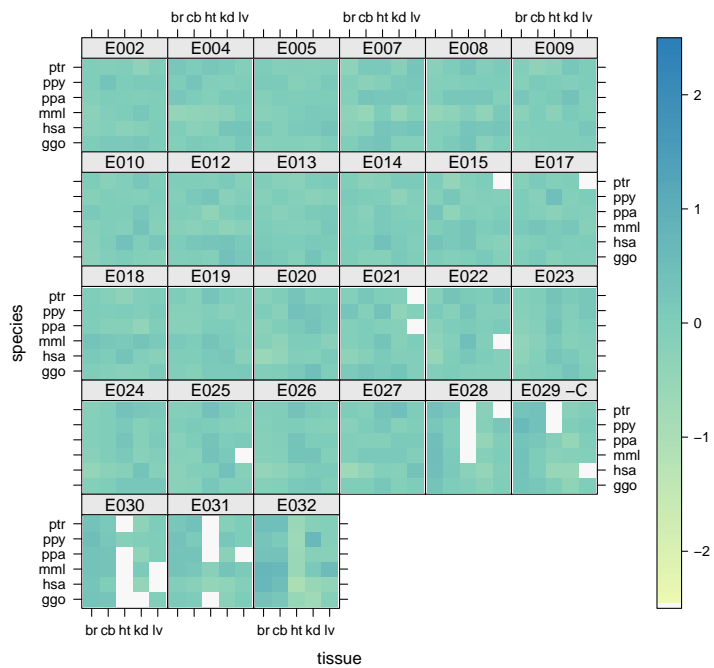


ENSG0000003989

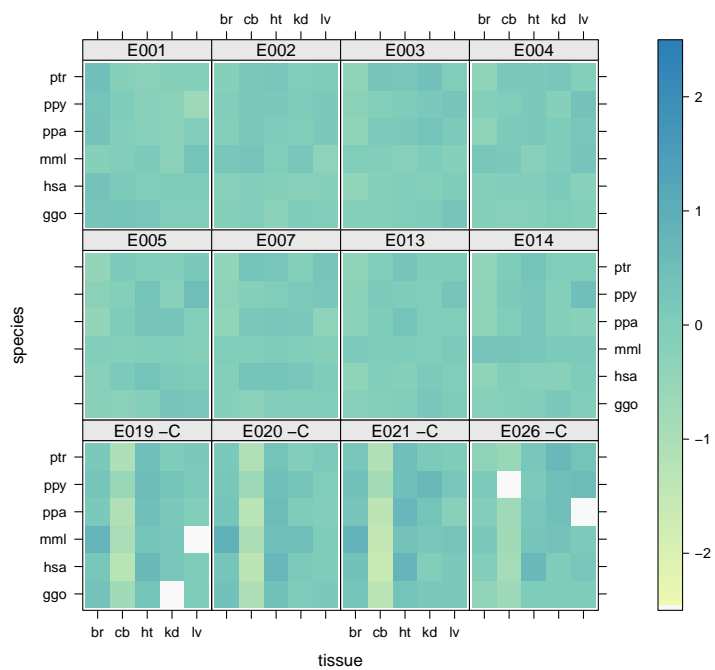




ENSG0000005379



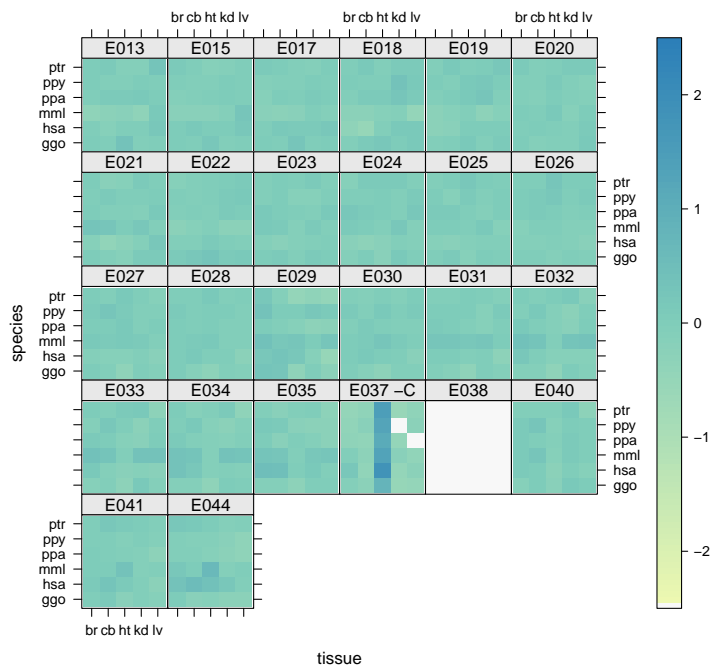
ENSG0000007237



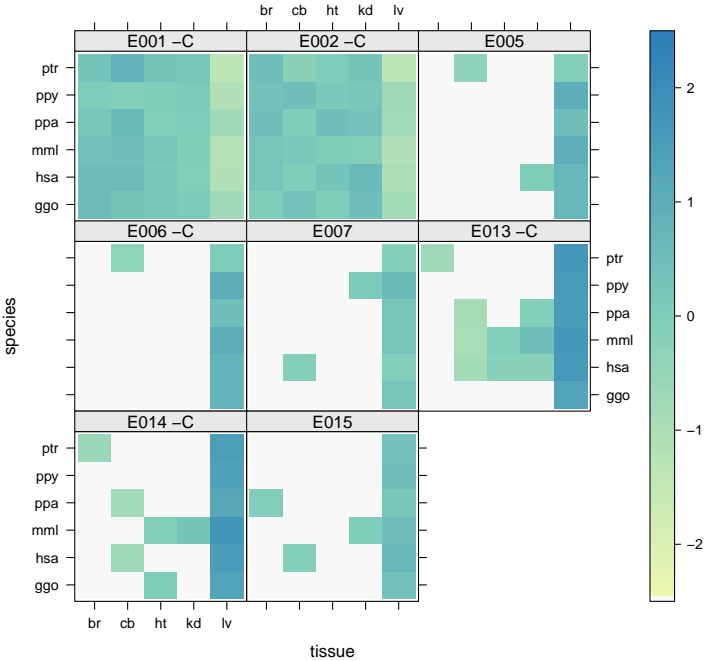




ENSG0000009307

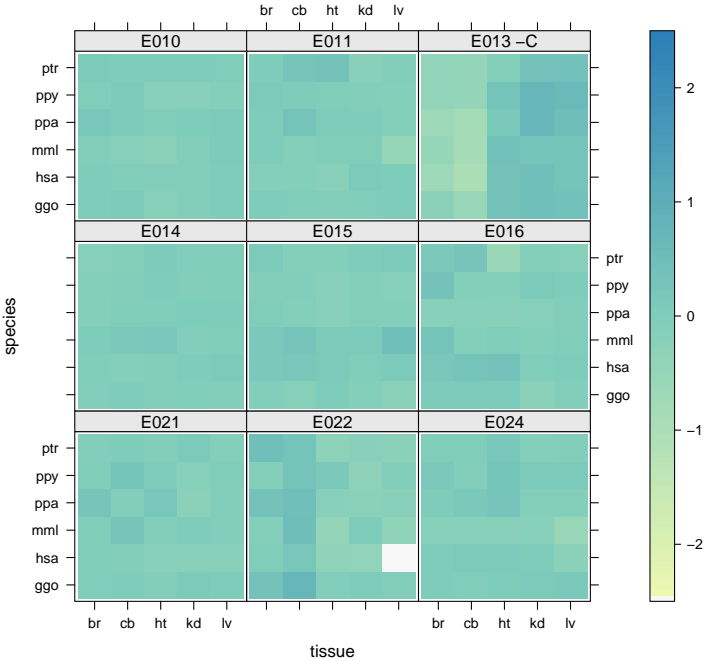


ENSG0000009724





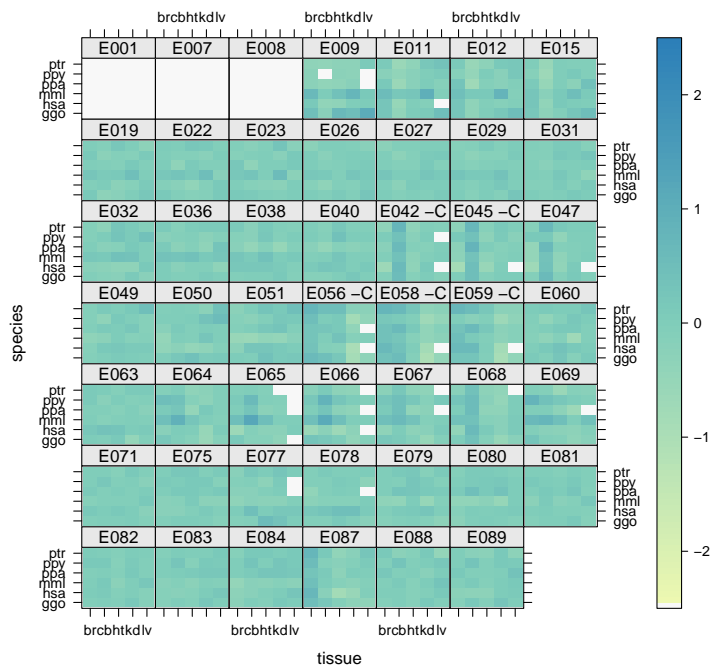
ENSG00000010295



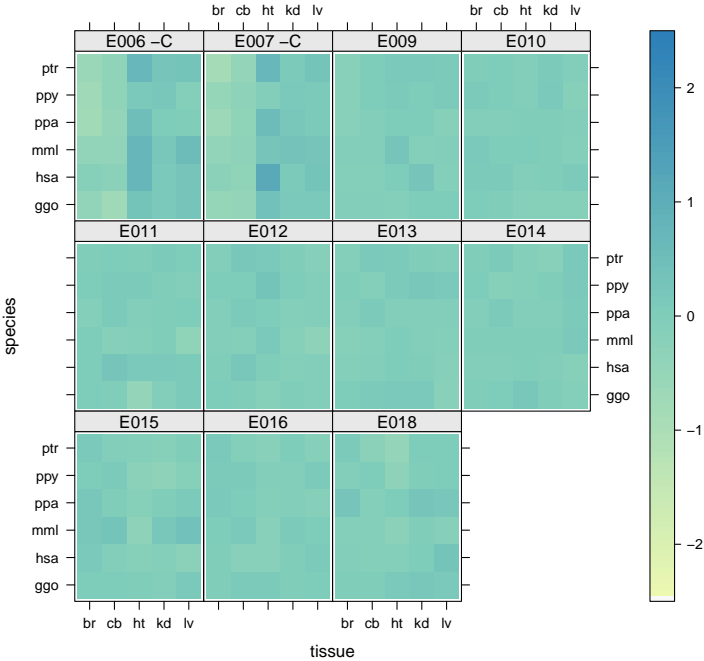




ENSG0000019144



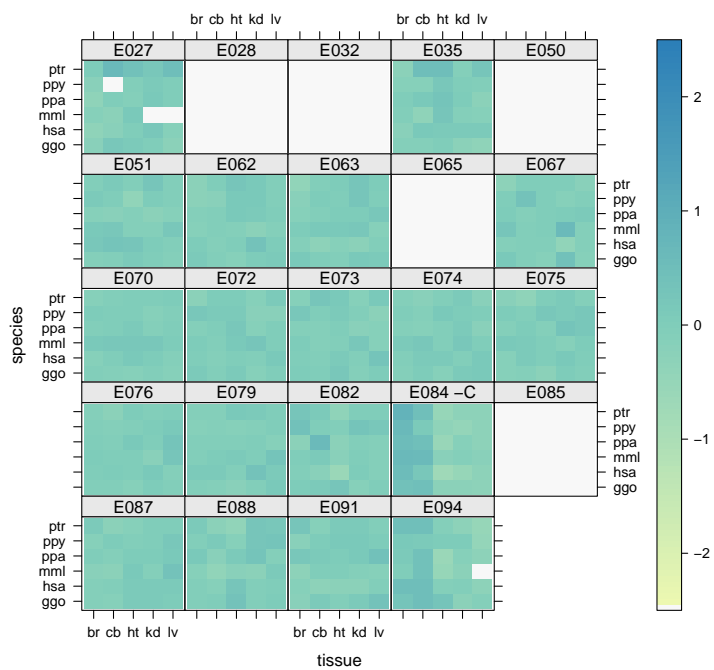
ENSG00000020129



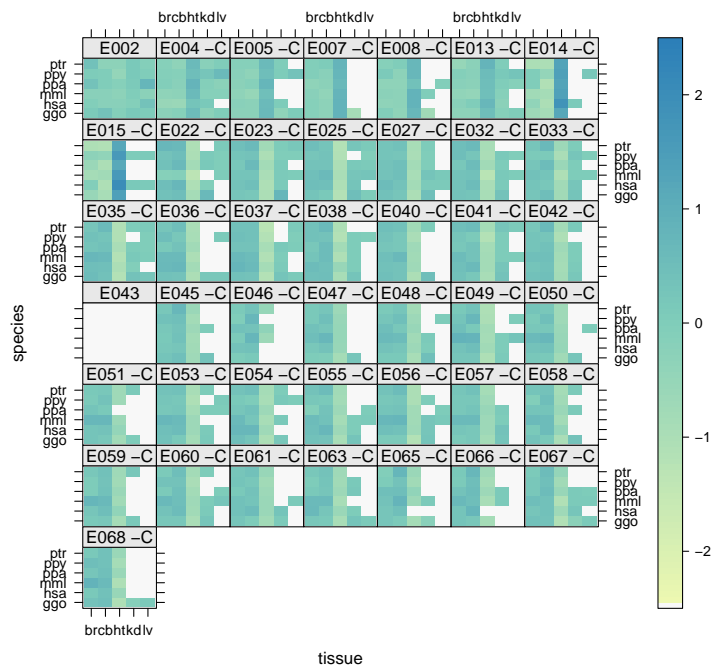




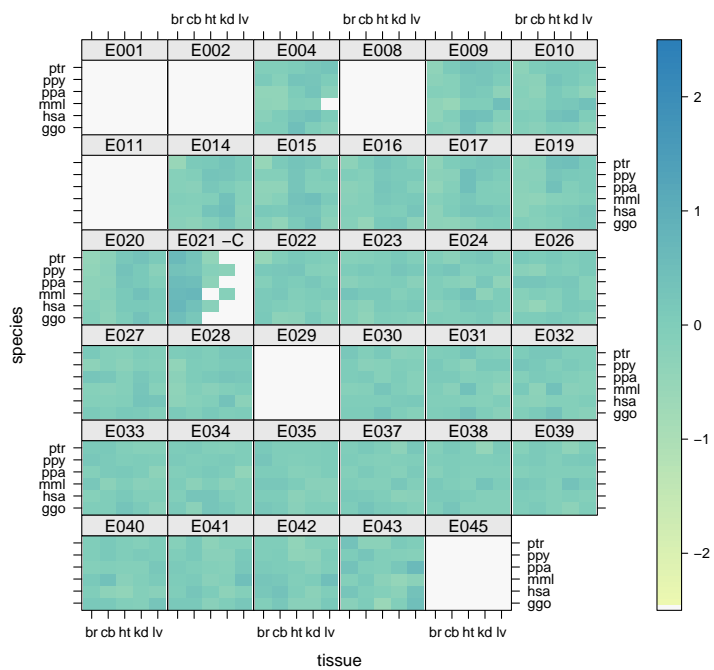
ENSG0000028203



ENSG0000029534



ENSG0000032219





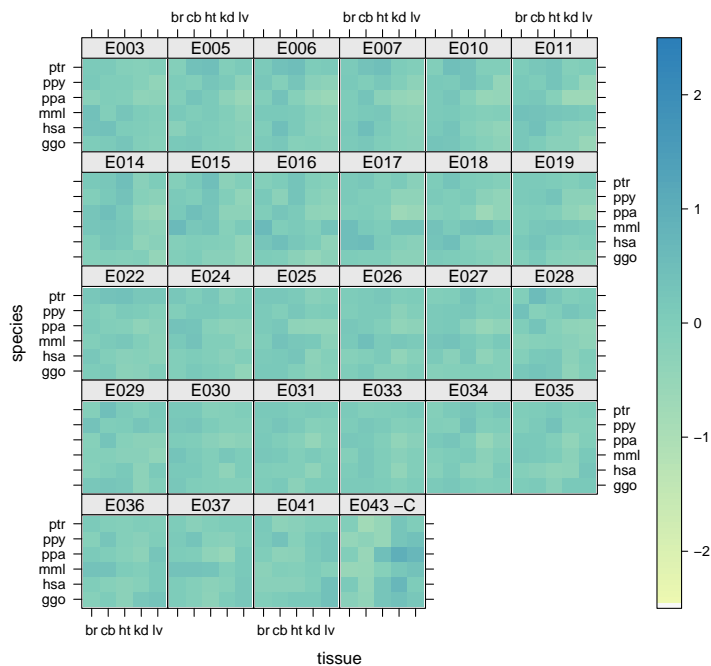




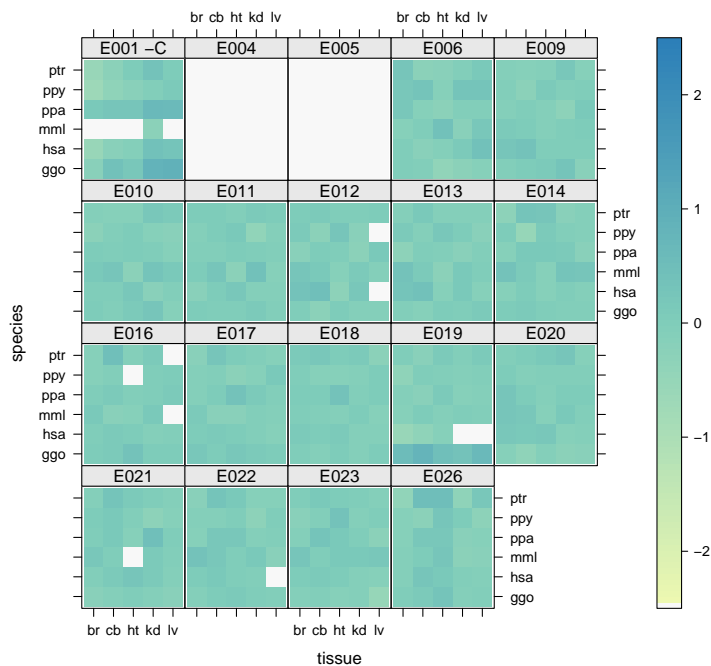




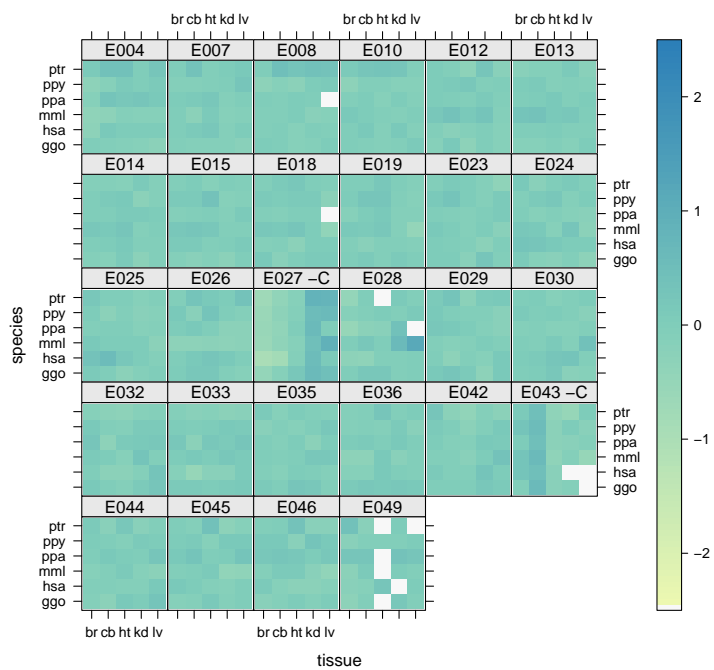
ENSG0000039123



ENSG0000040199



ENSG0000040933

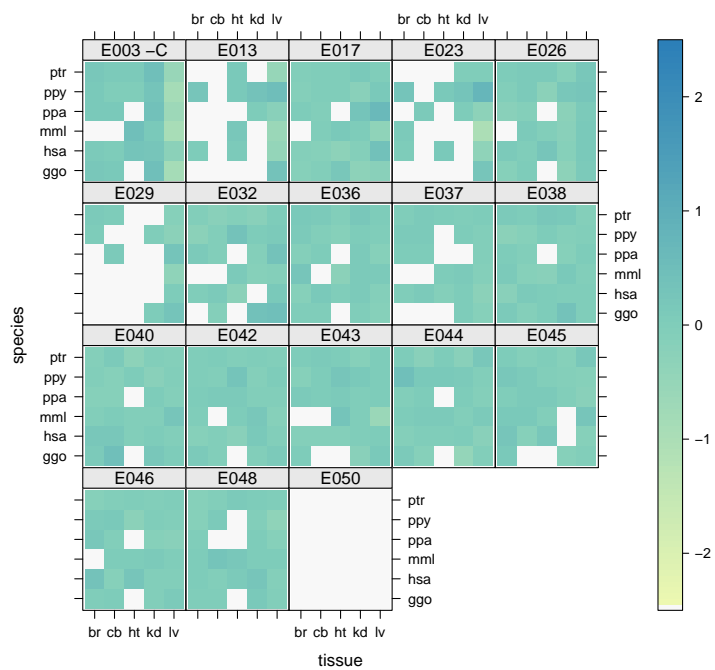








ENSG00000047457

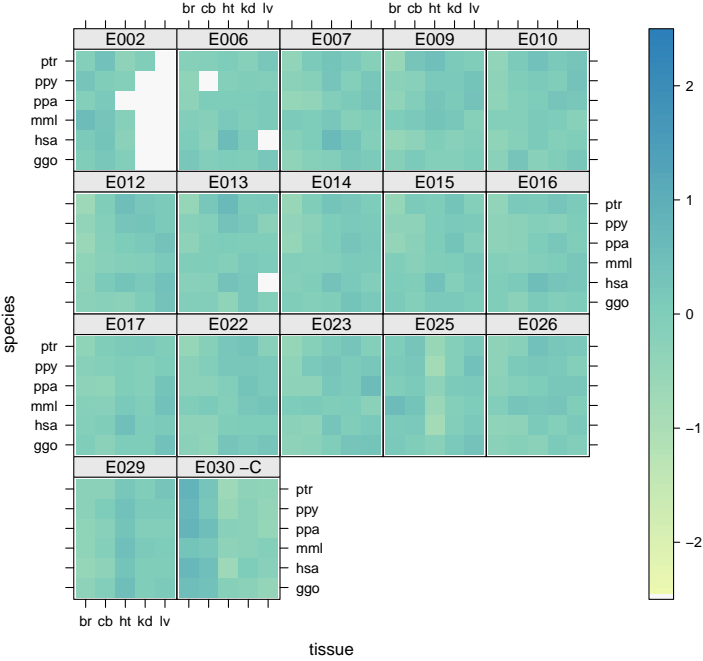




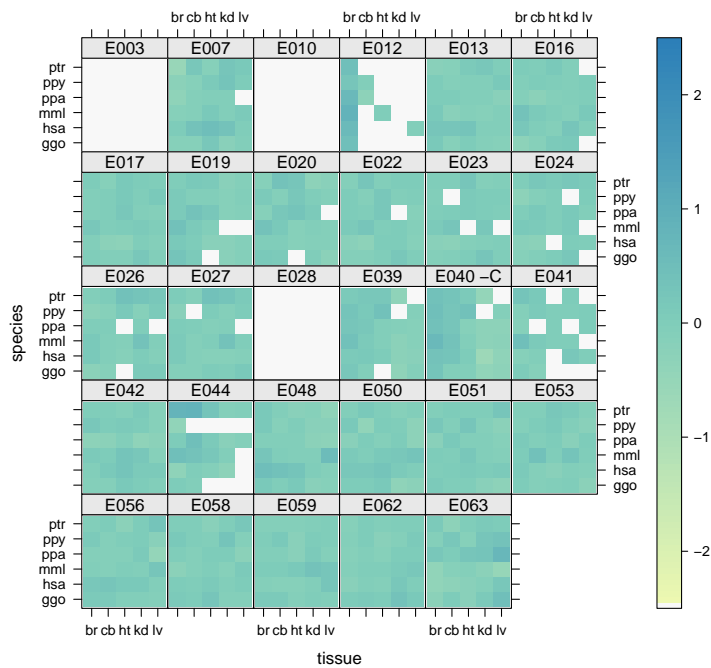




ENSG00000048740



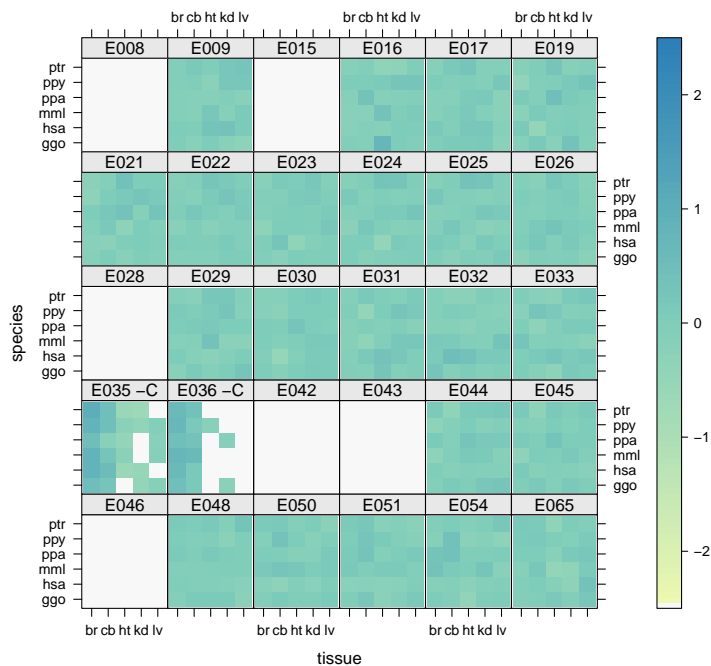
ENSG0000048991



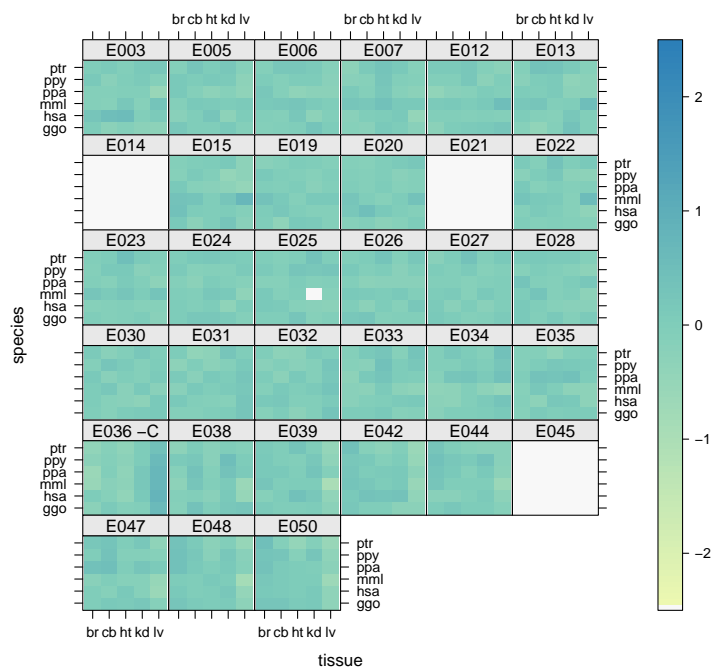




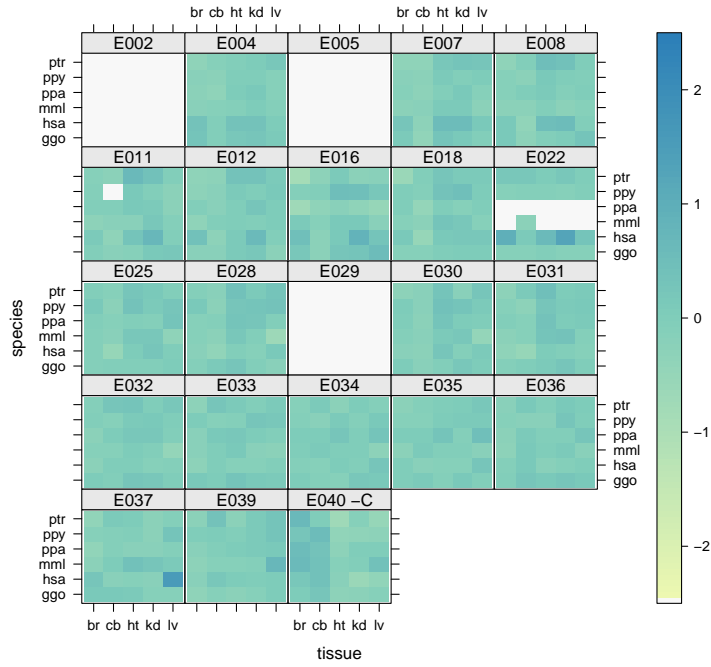
ENSG0000052126



ENSG0000052841



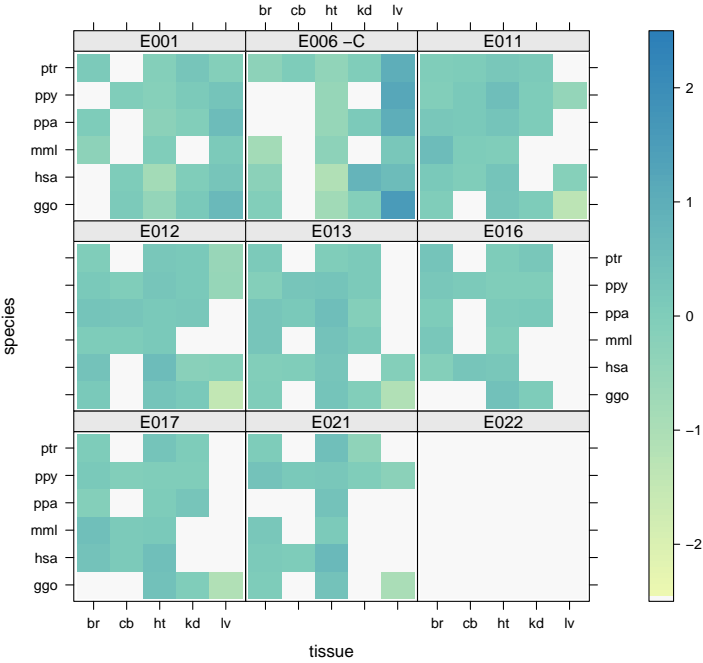
ENSG0000054282



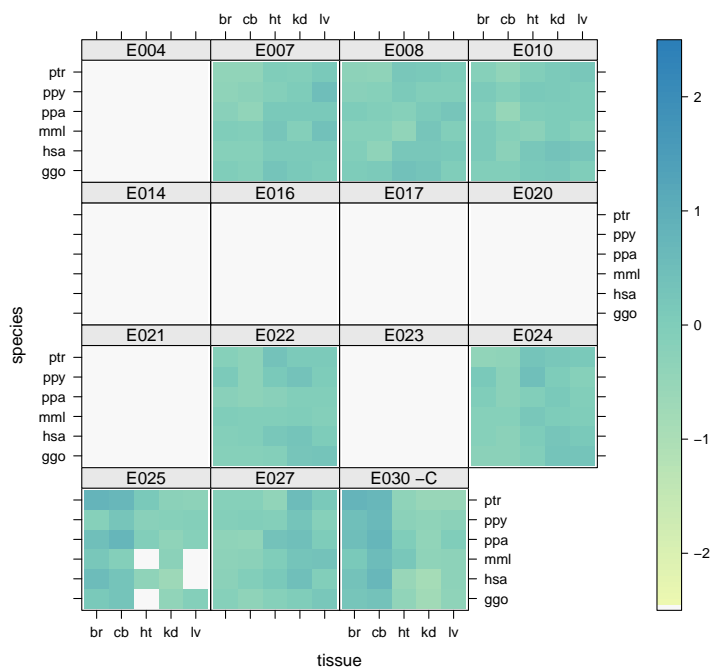




ENSG0000054938

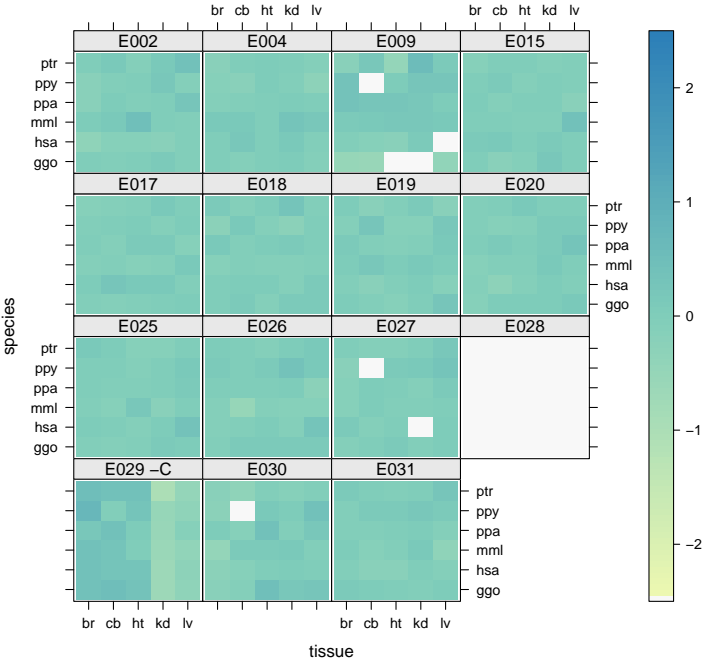


ENSG0000056972



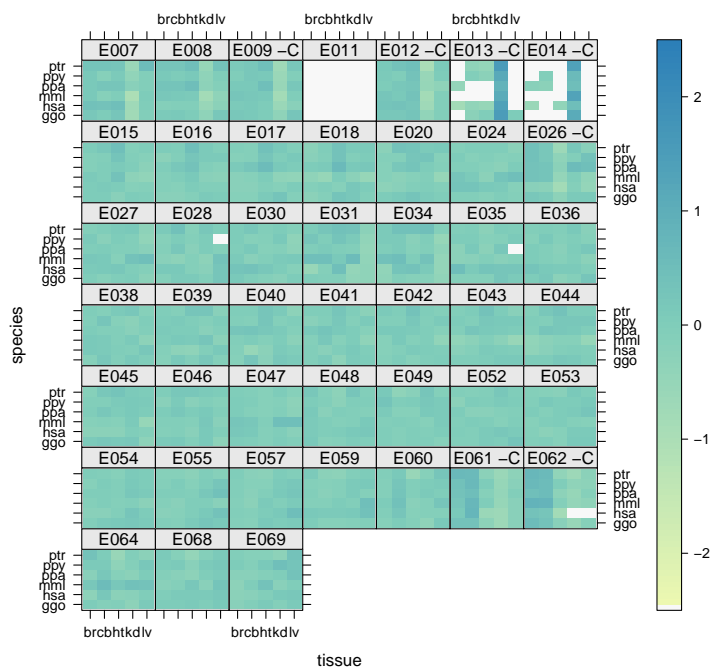


ENSG0000058668





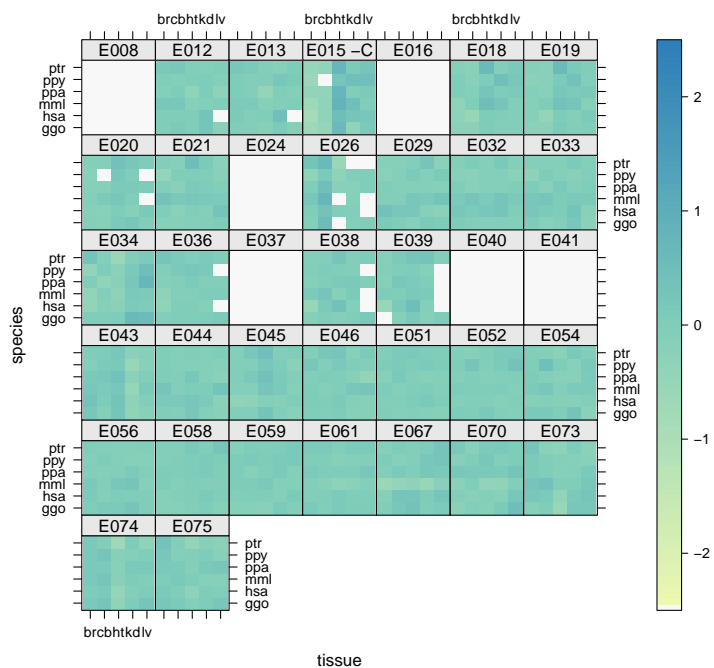
ENSG0000060237



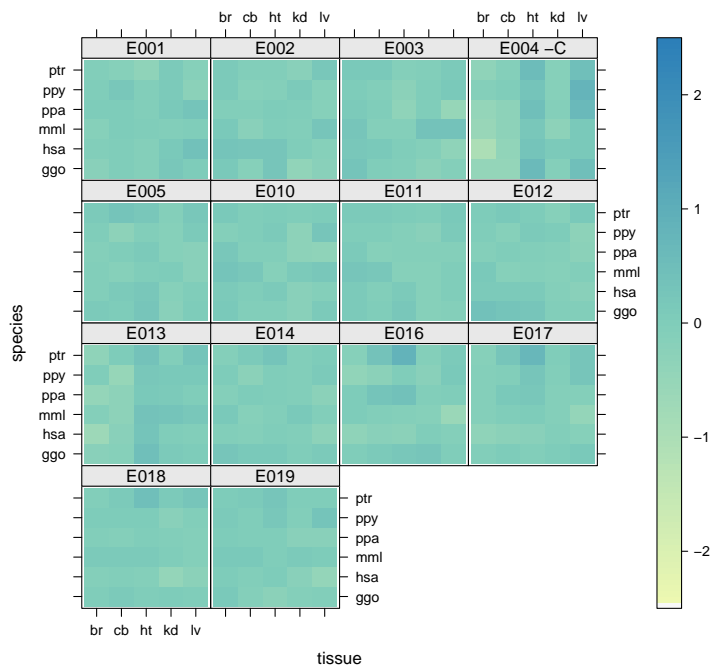




ENSG0000064042



ENSG0000064393



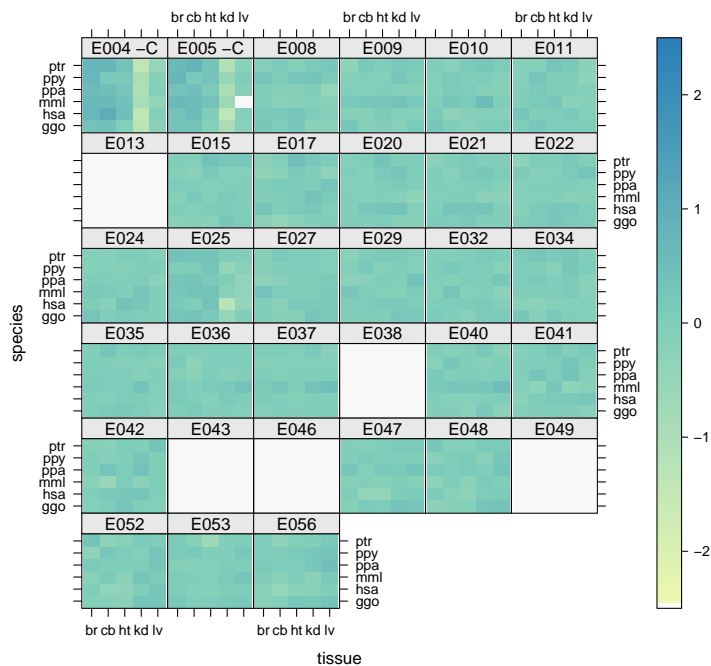








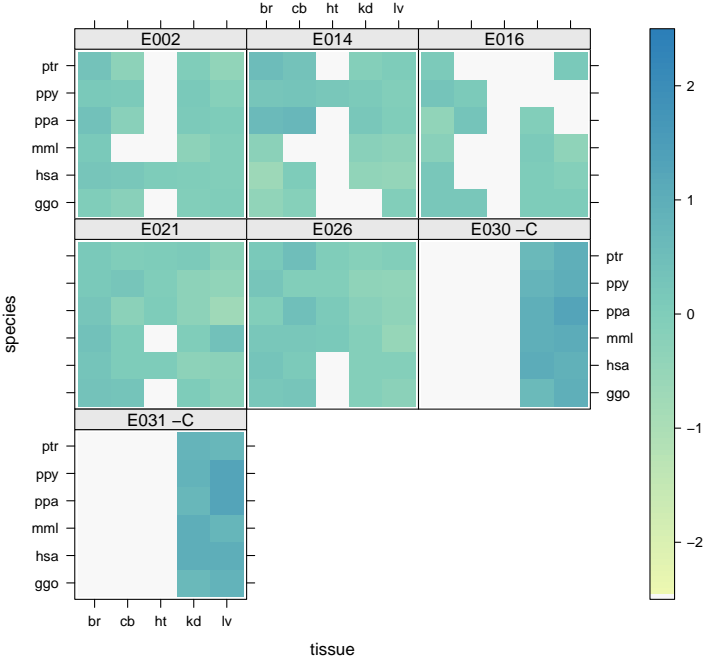
ENSG0000065882



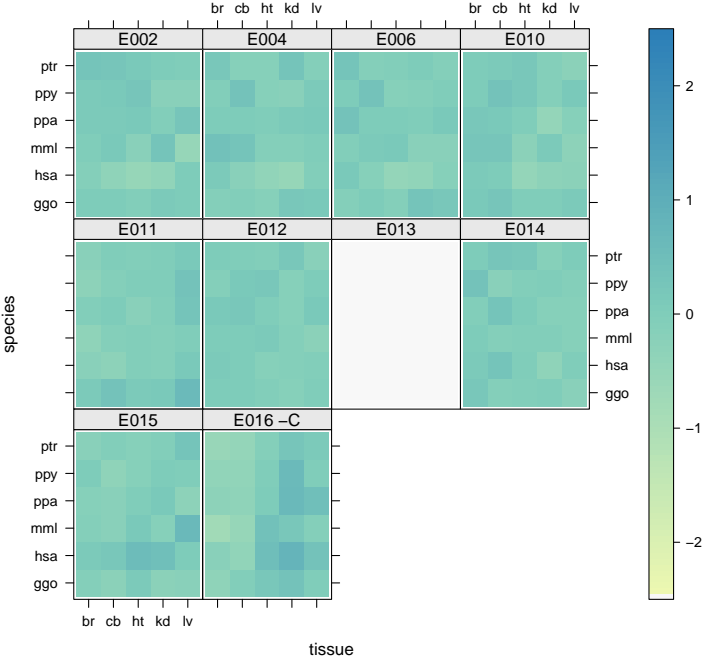




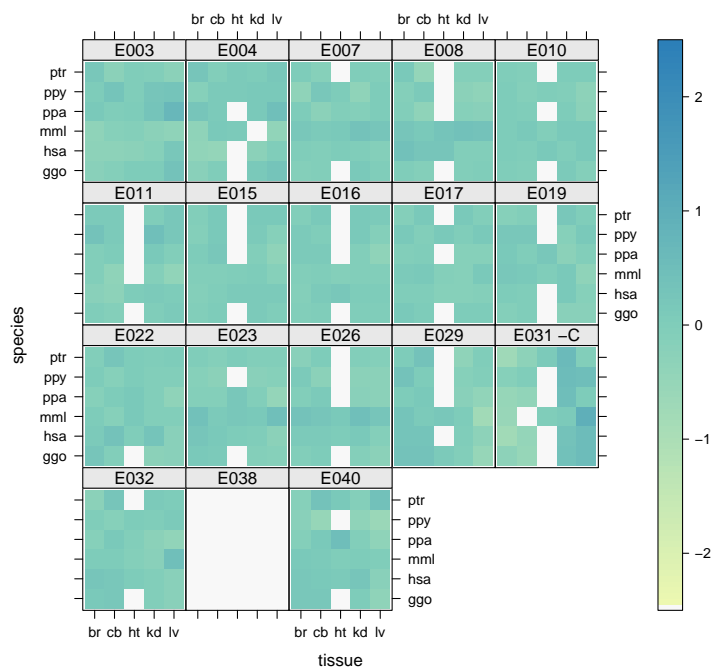
ENSG0000066248



ENSG0000066422

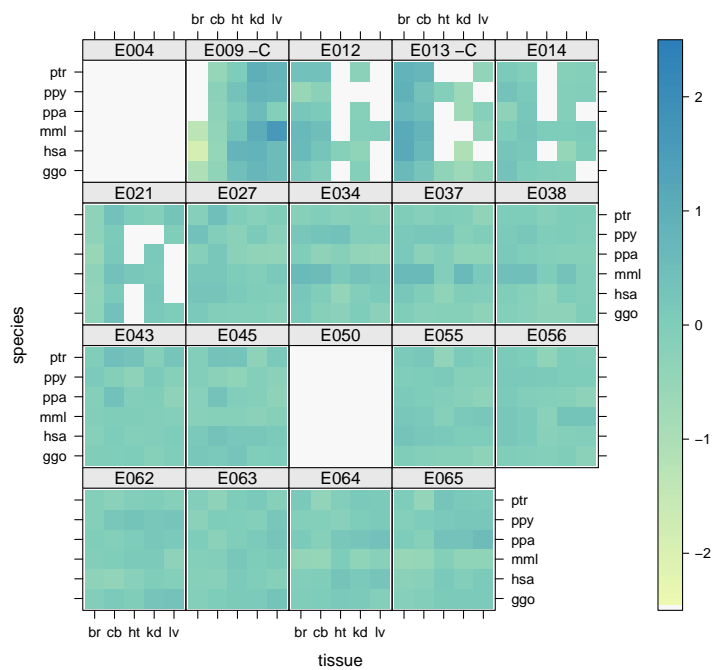


ENSG00000066468



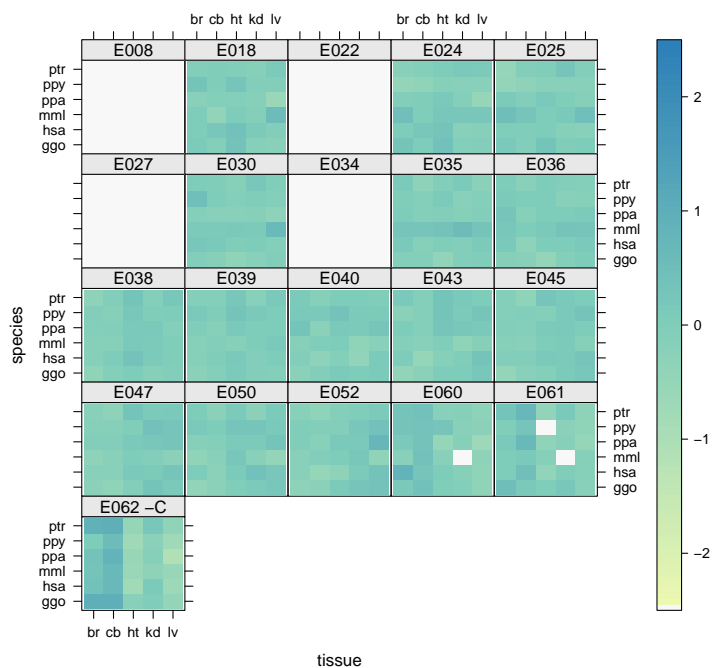


ENSG0000067606

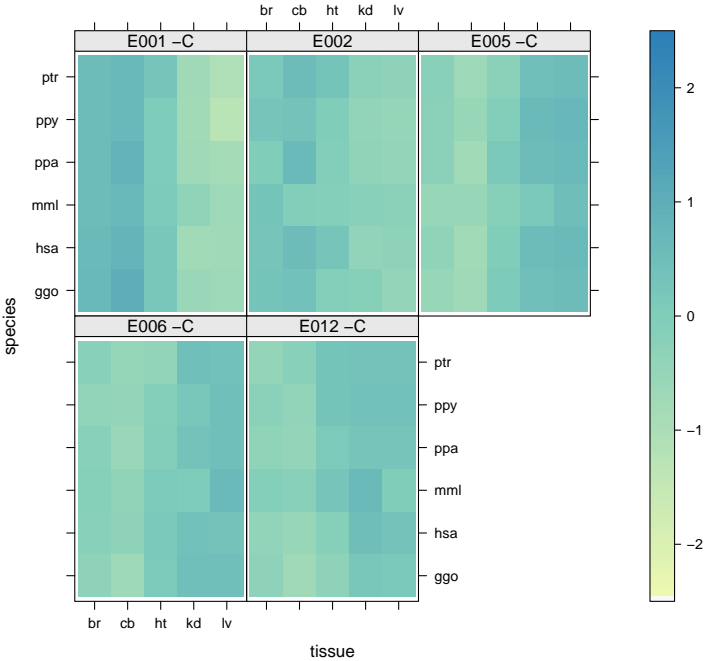




ENSG0000070081

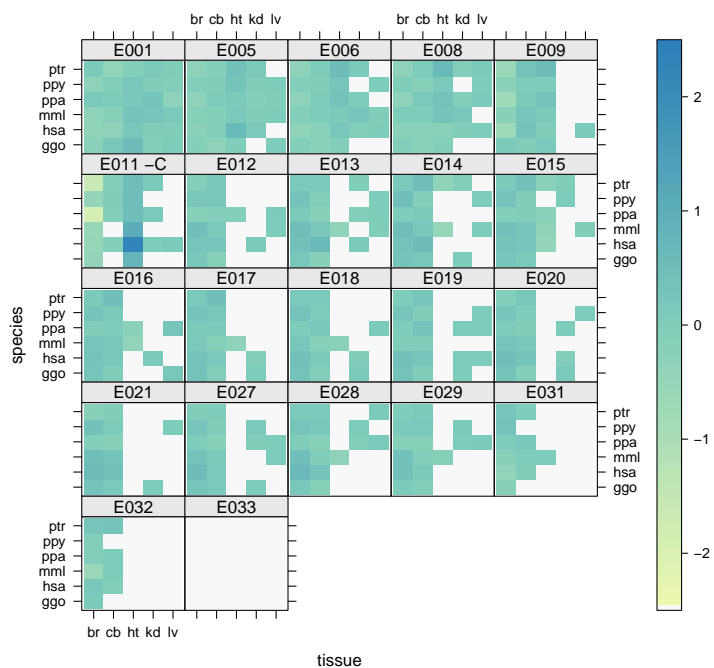


ENSG00000070495

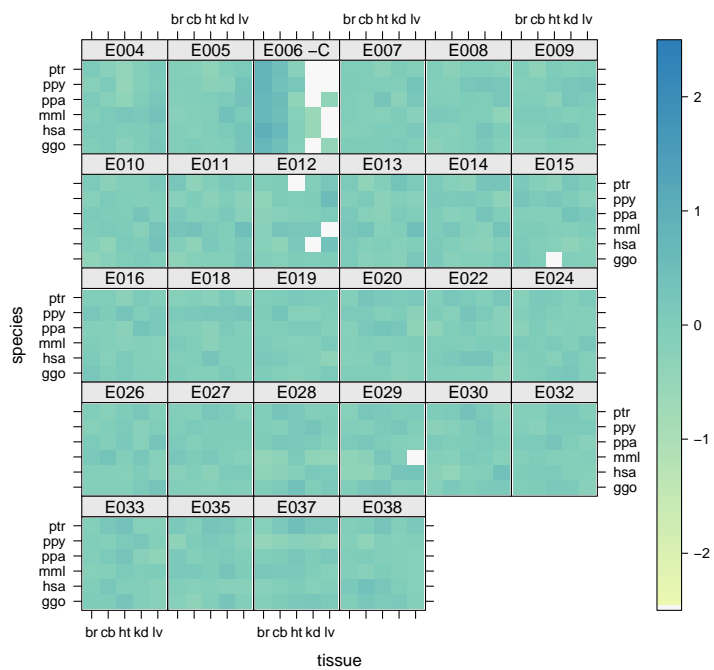




ENSG0000070808

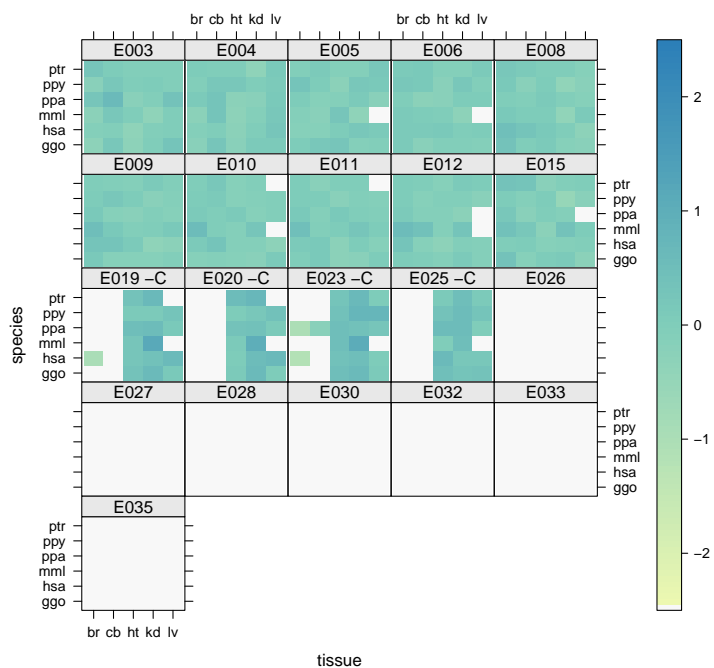


ENSG0000070961



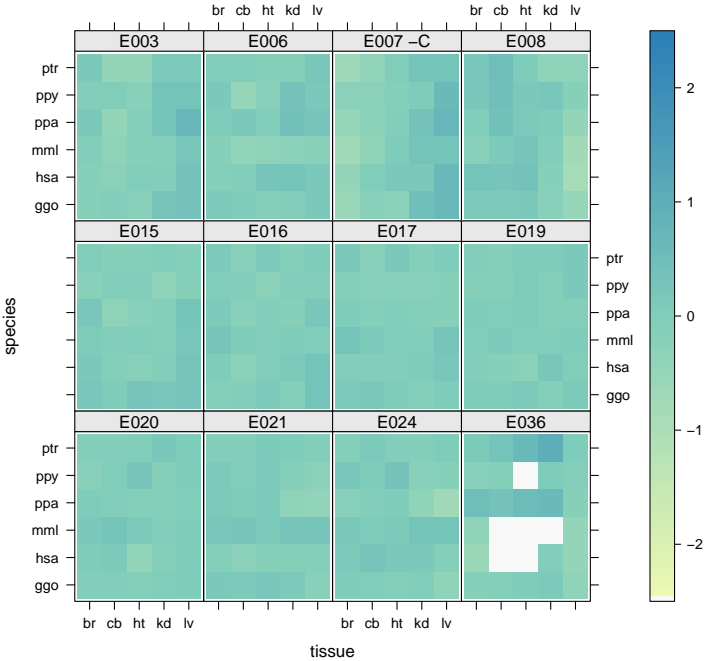


ENSG00000072201

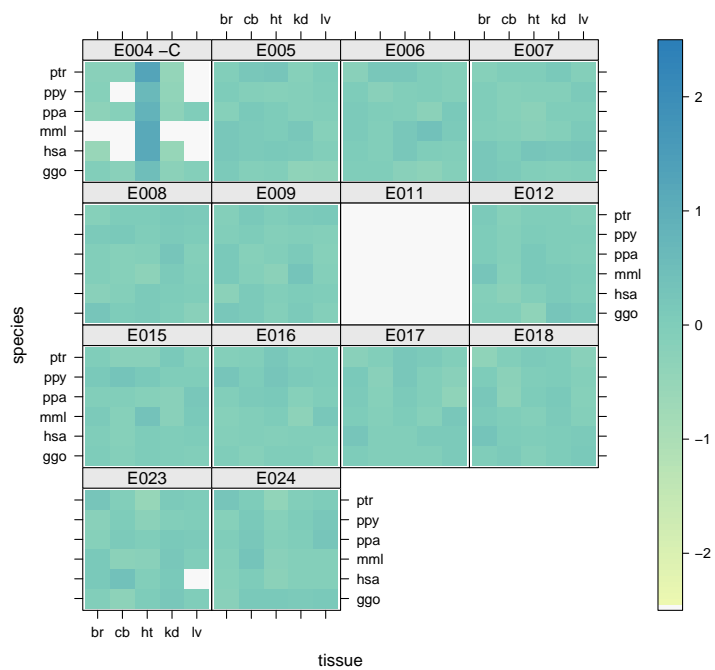




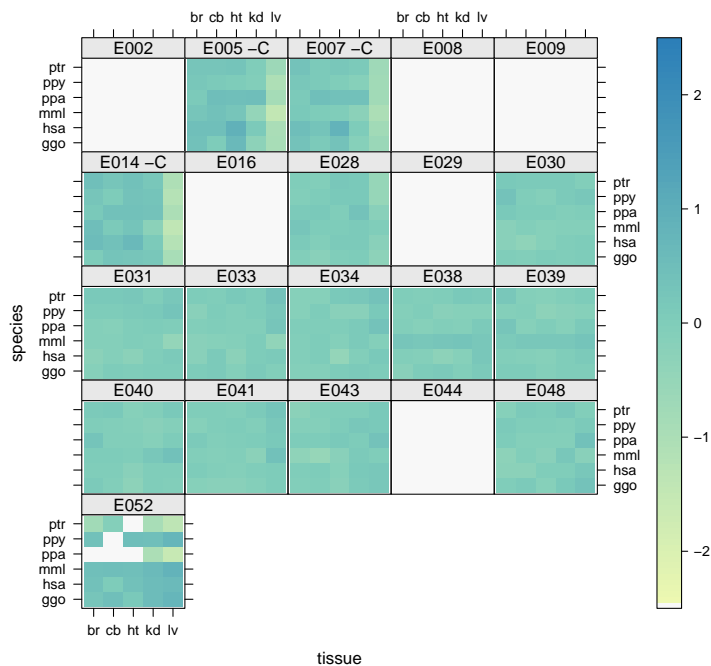
ENSG0000073060



ENSG0000073711



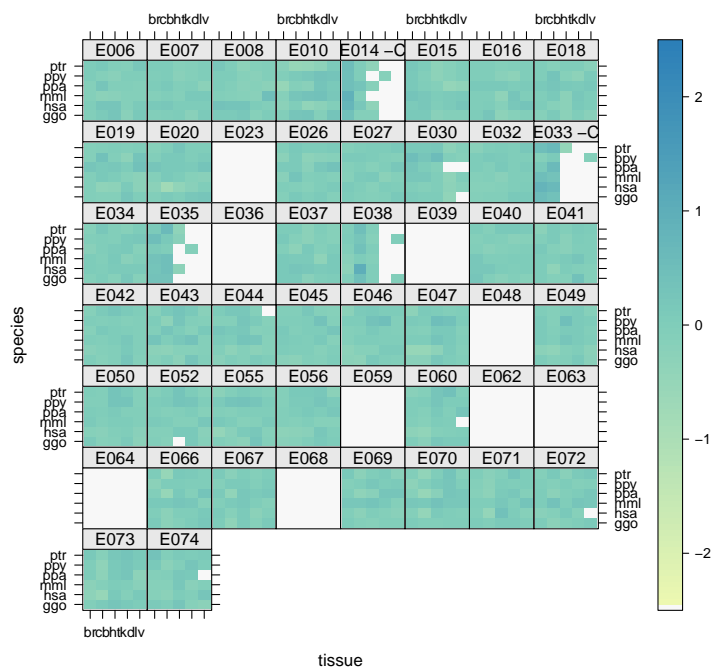
ENSG0000073849







ENSG0000074054



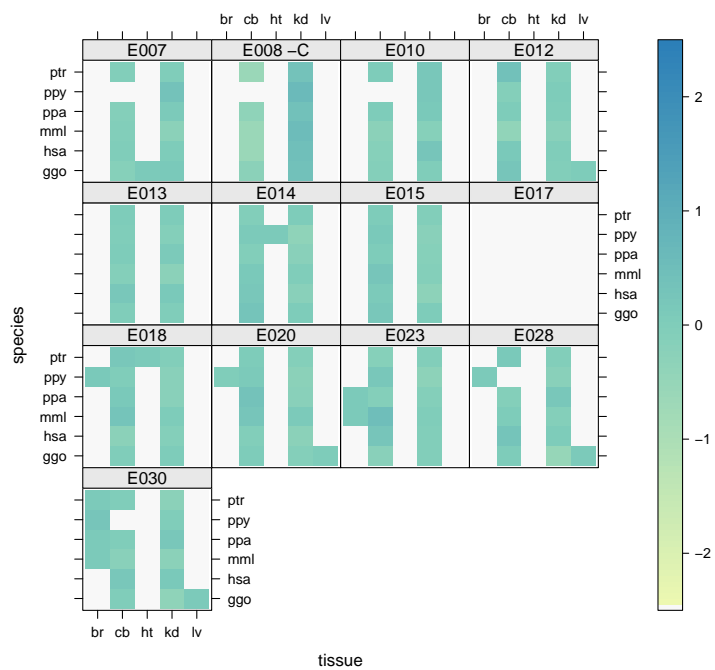




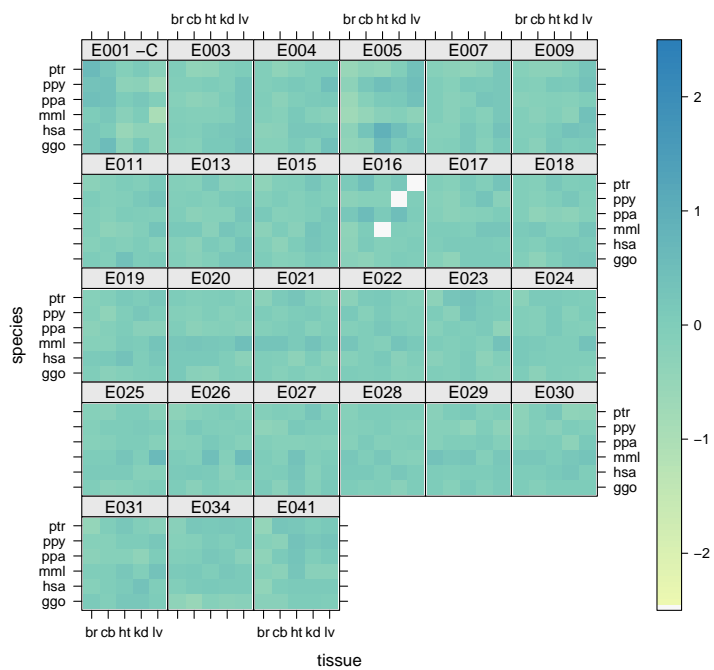




ENSG00000075891

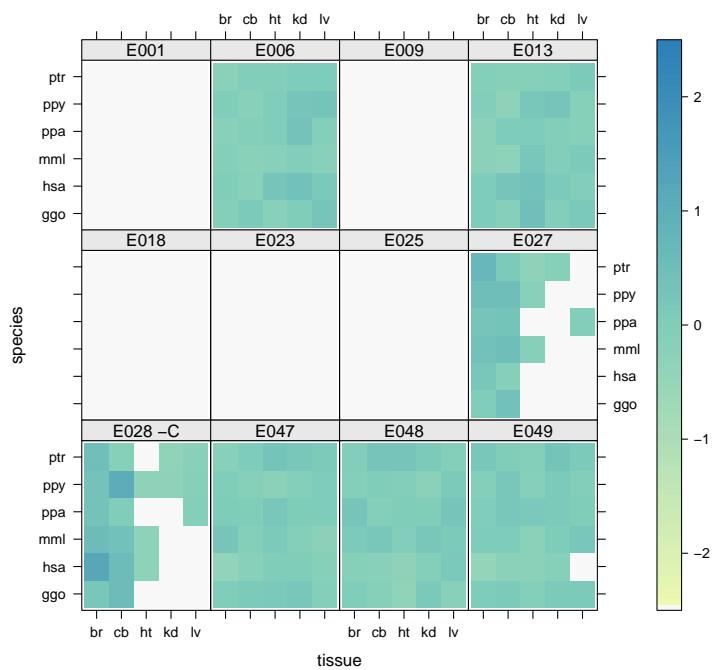


ENSG0000076356

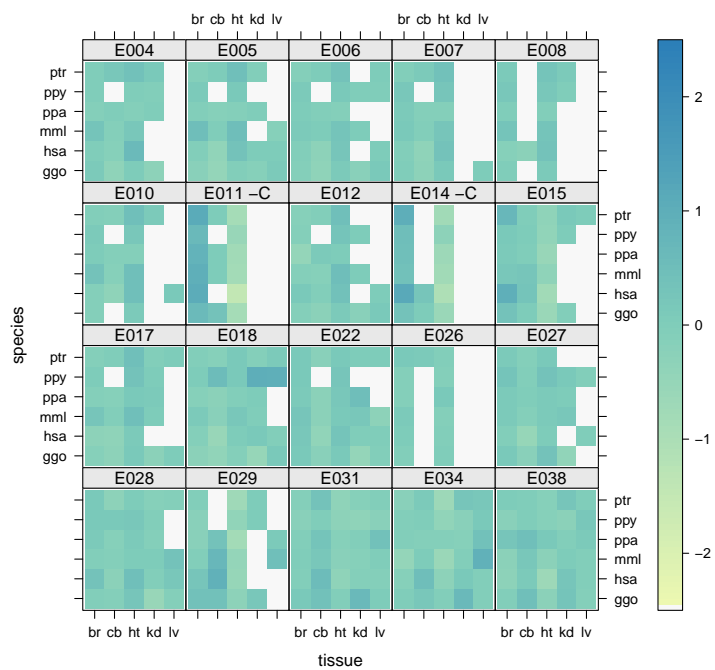




ENSG0000077380



ENSG0000077522





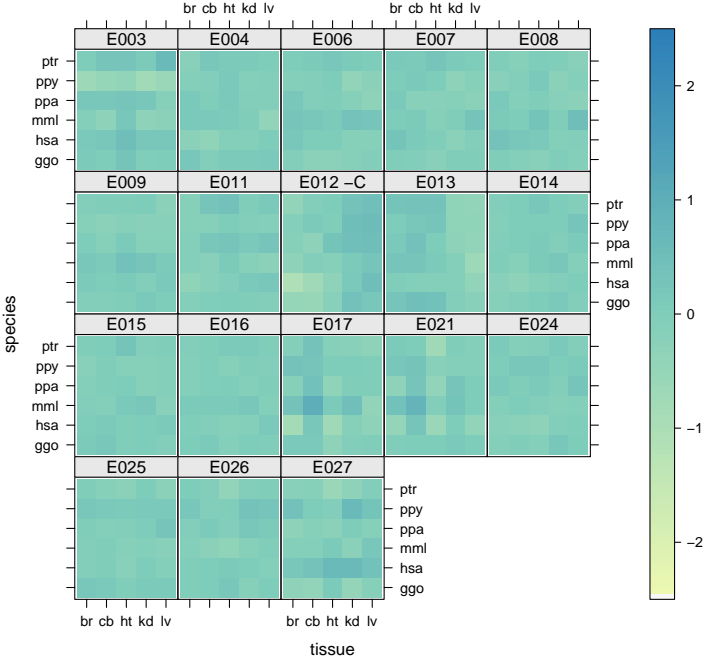






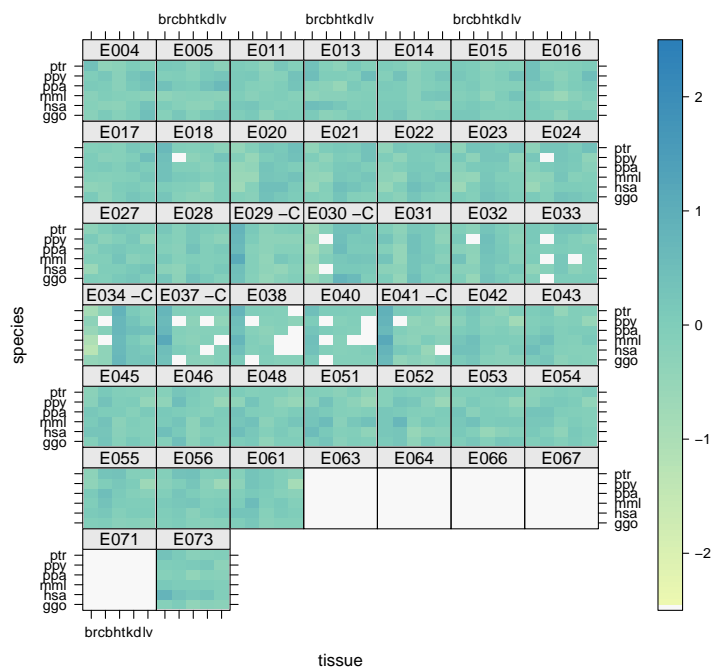


ENSG00000079805

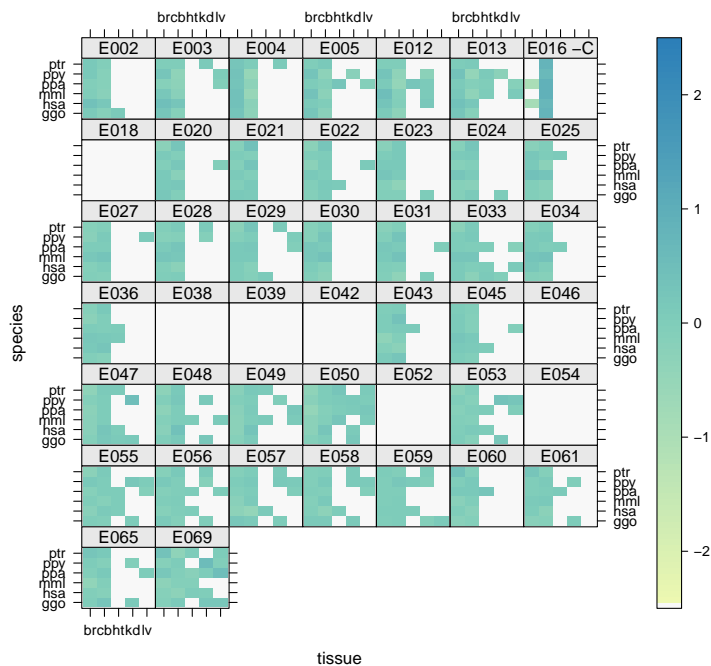




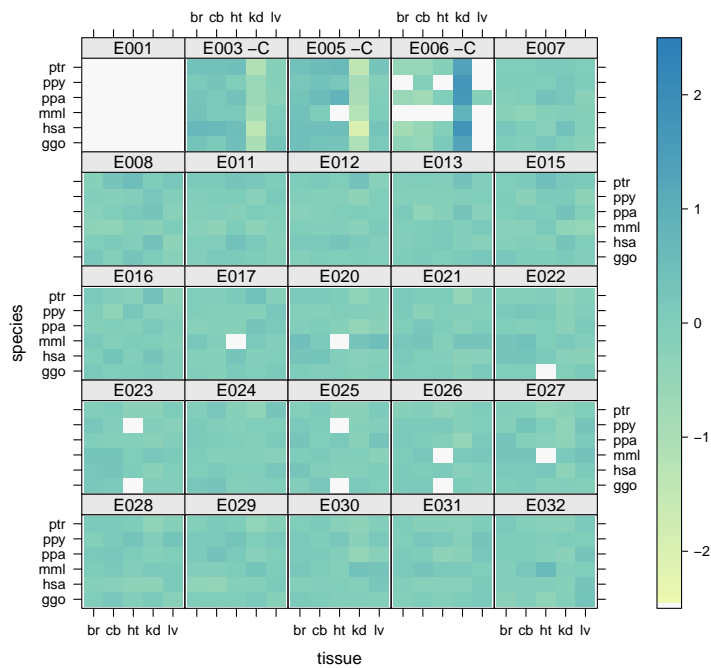
ENSG0000079819



ENSG0000079841

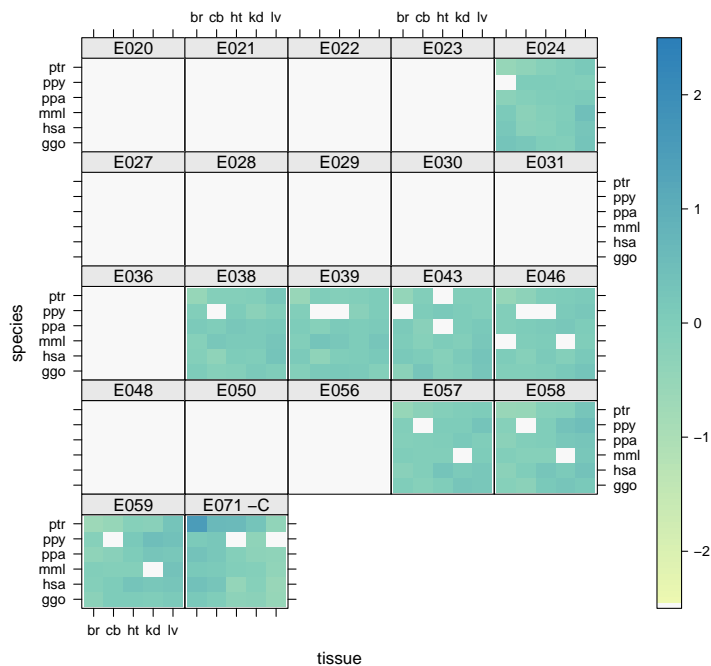


ENSG0000080493

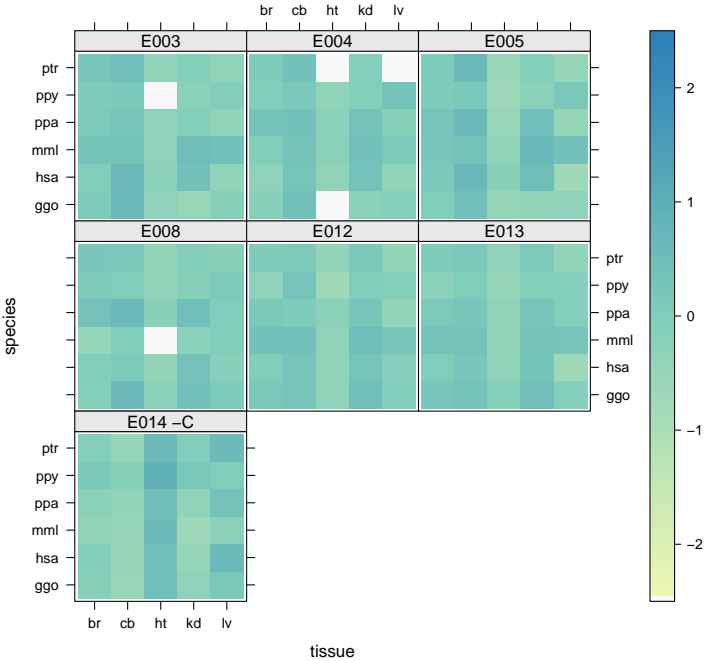




ENSG0000081059



ENSG0000081181







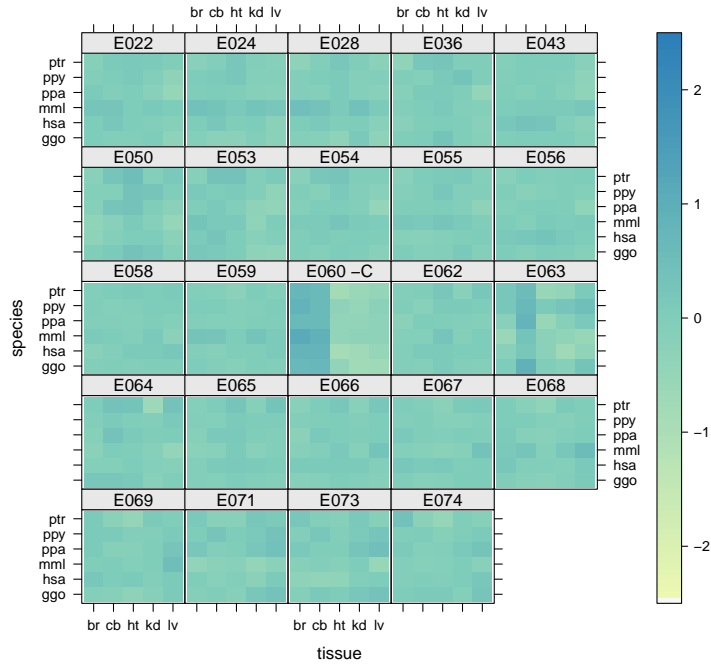




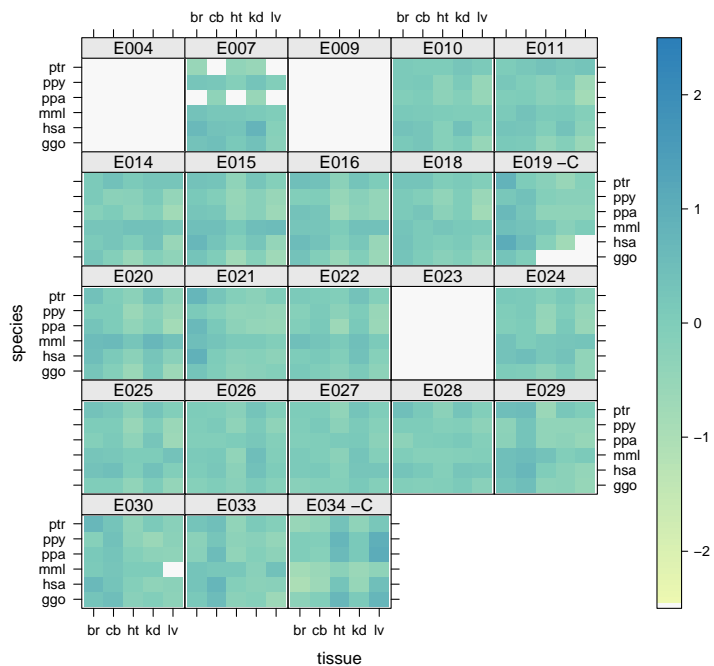




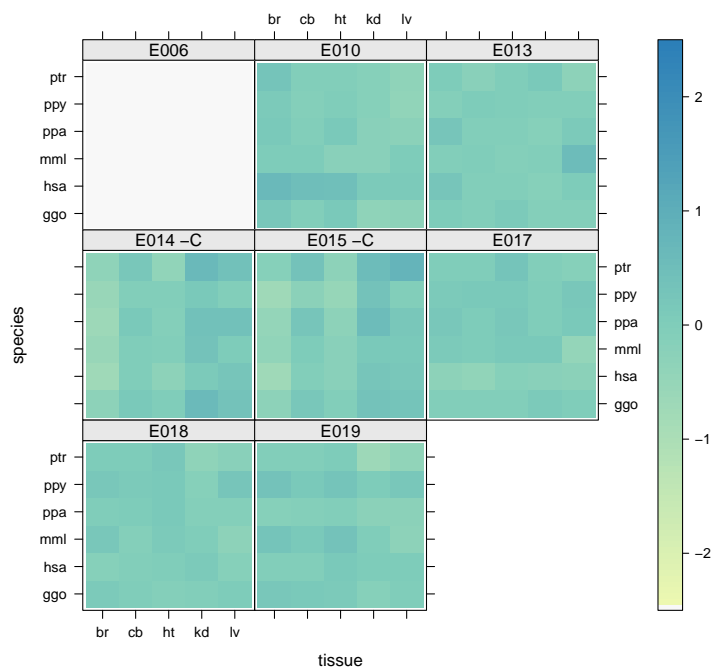
ENSG0000084234



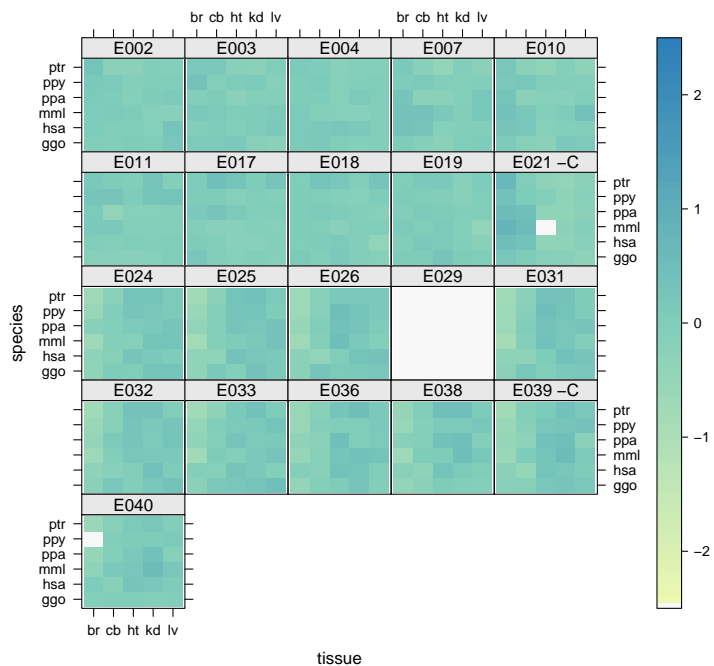
ENSG0000084693



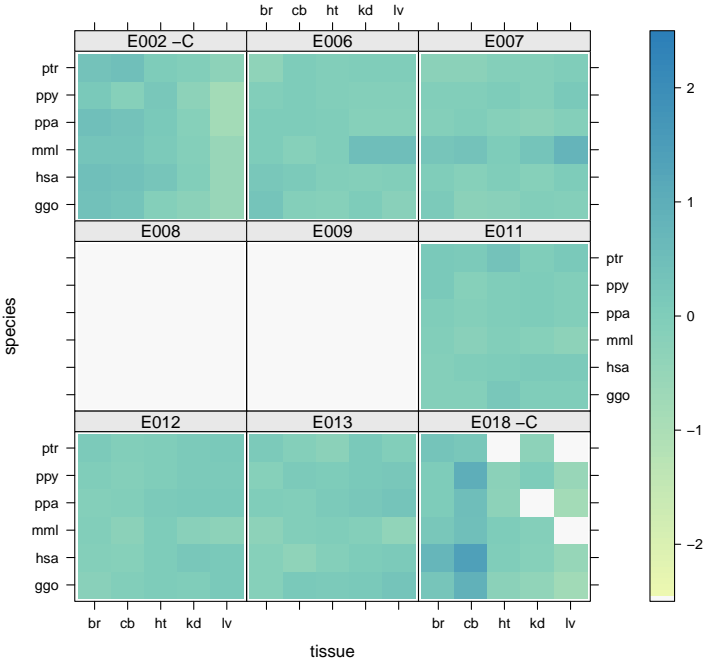
ENSG0000084764



ENSG00000085832

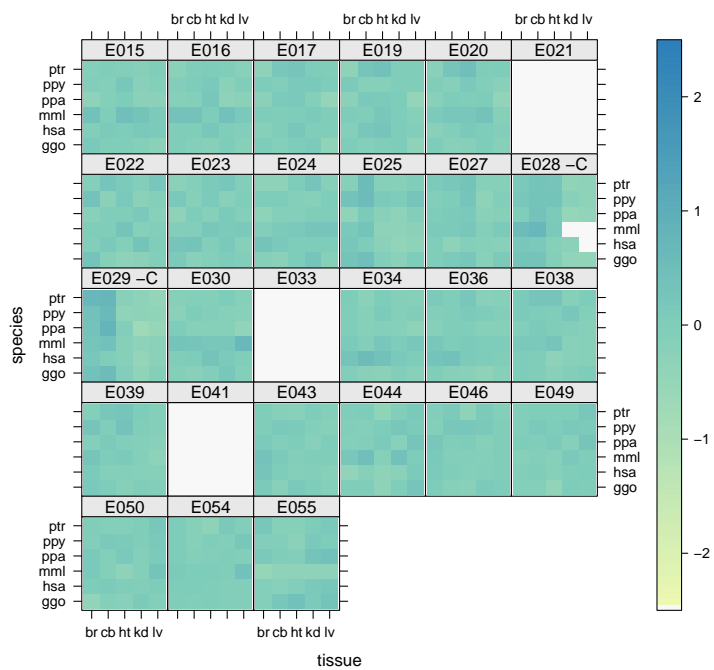


ENSG0000085871

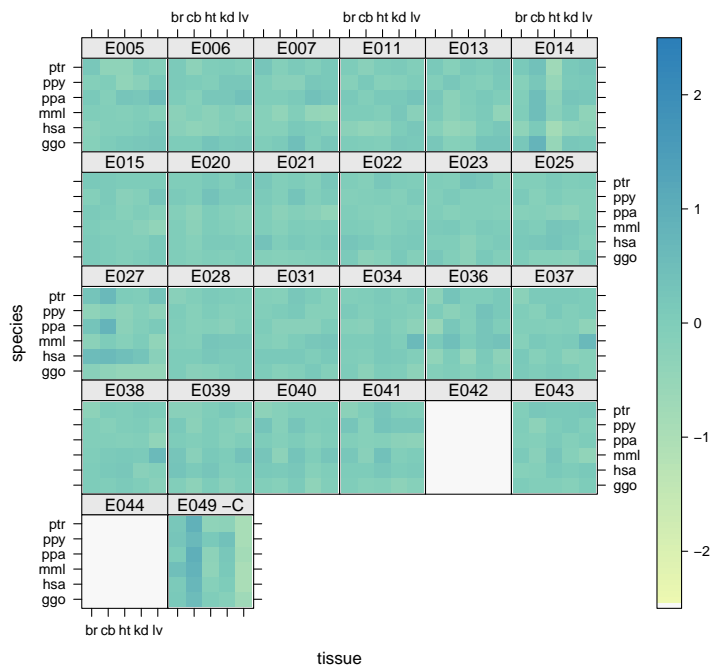




ENSG0000085978

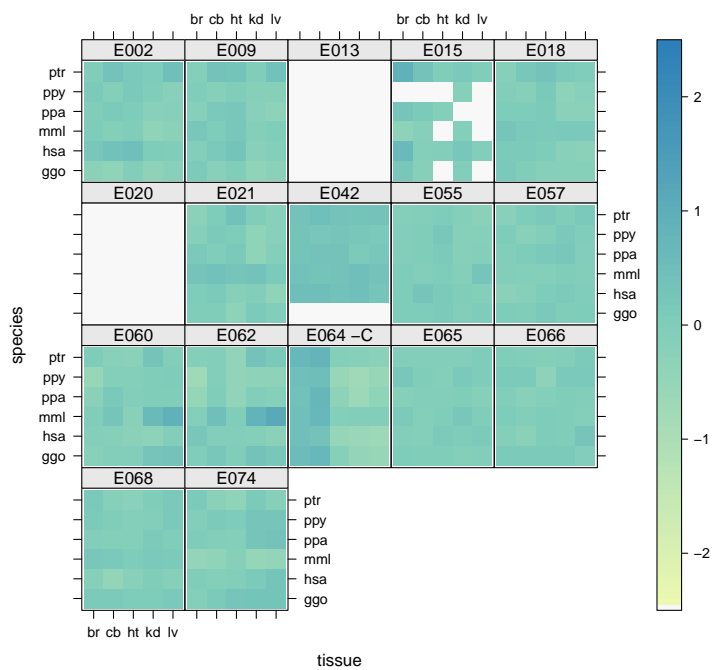


ENSG0000085998



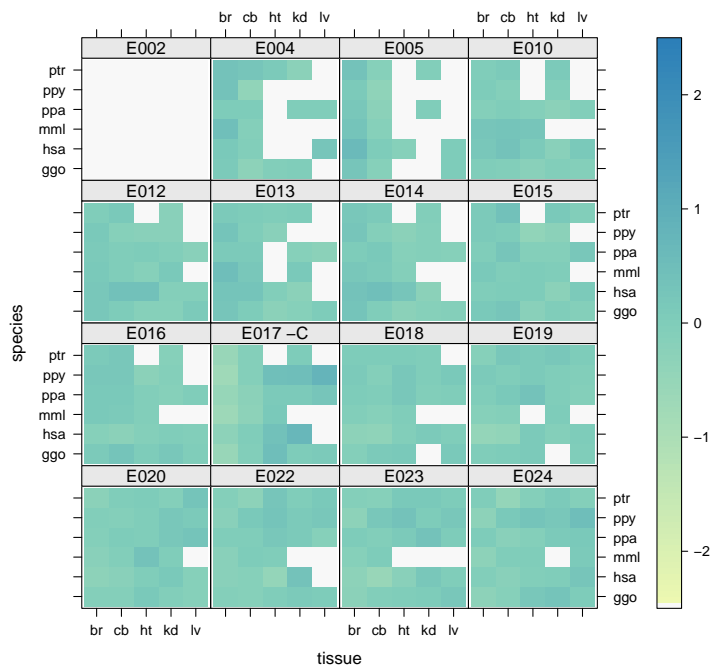


ENSG0000087274

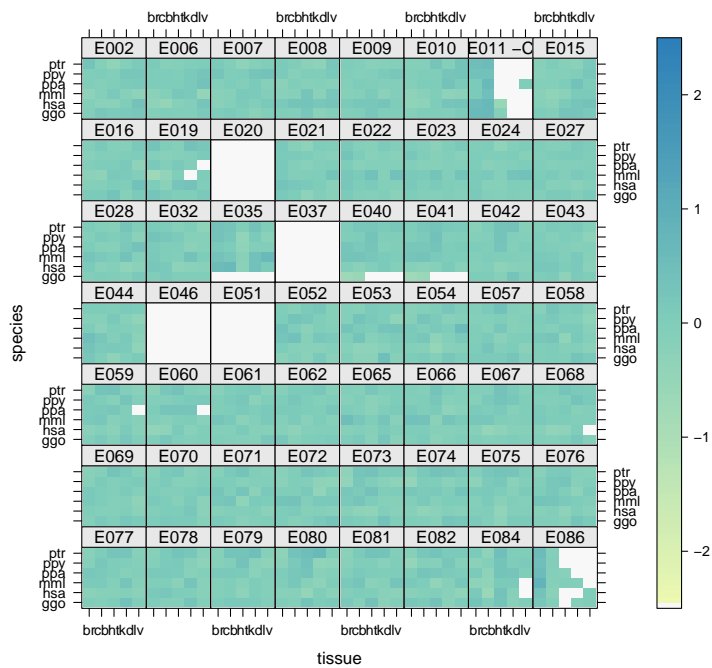




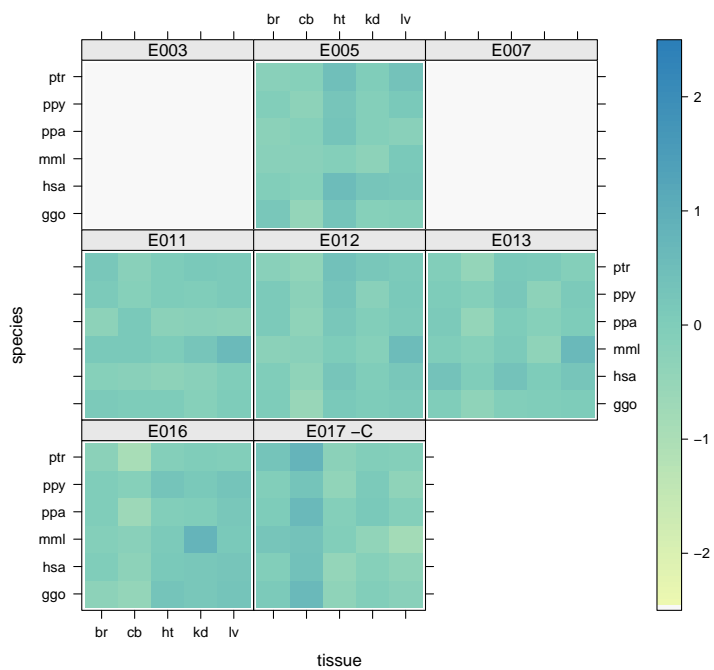
ENSG0000087495



ENSG0000088387



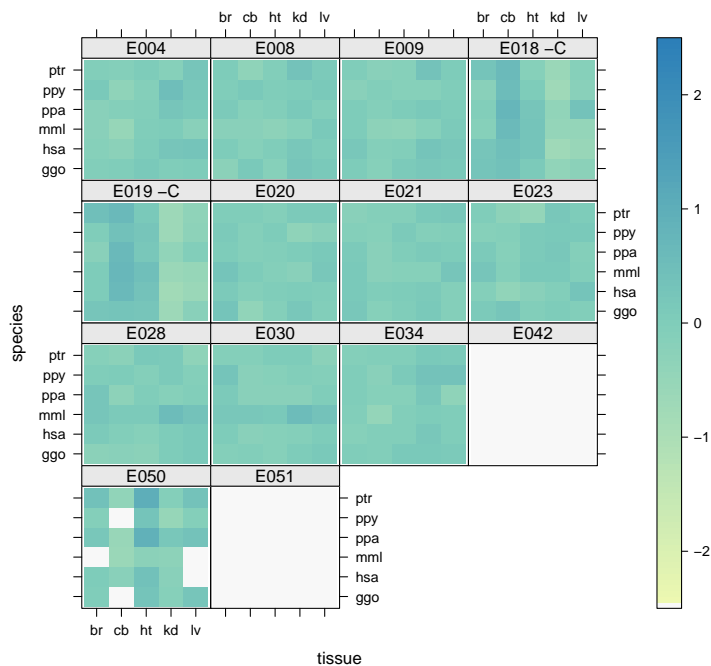
ENSG0000088766



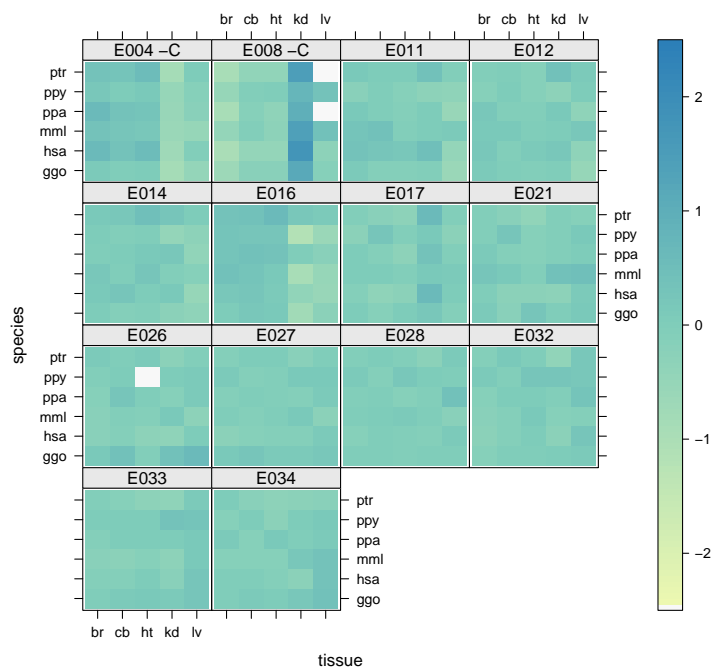




ENSG0000089159

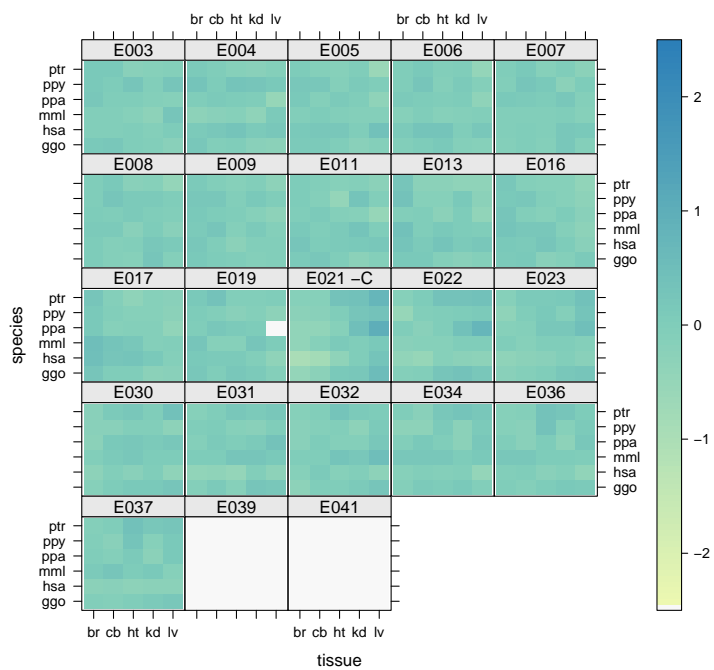


ENSG00000090565

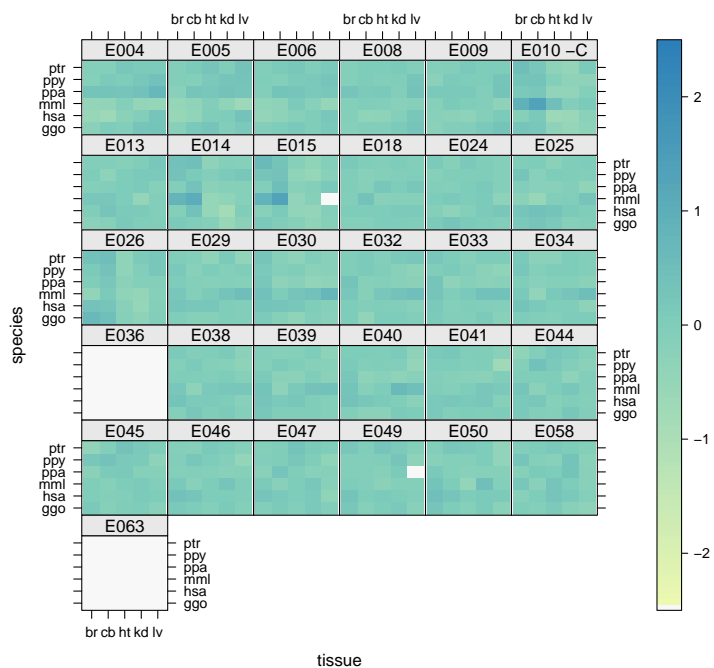




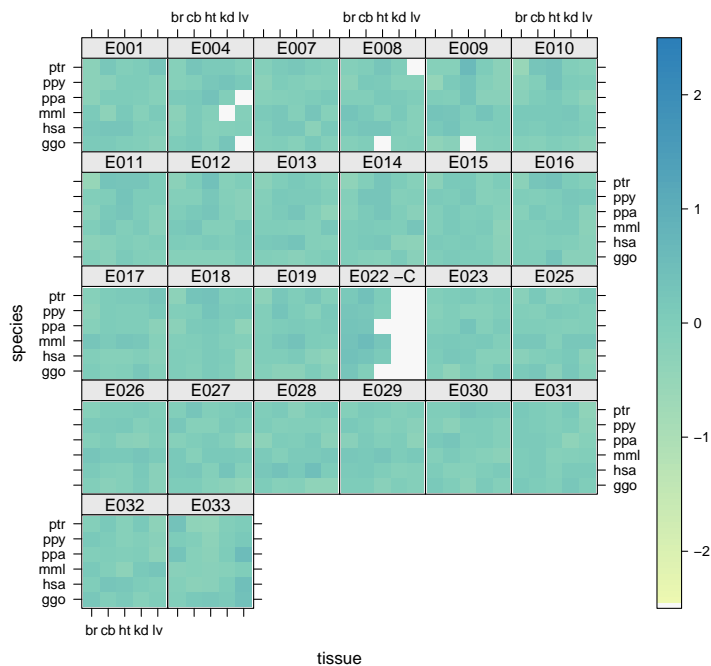
ENSG00000090975



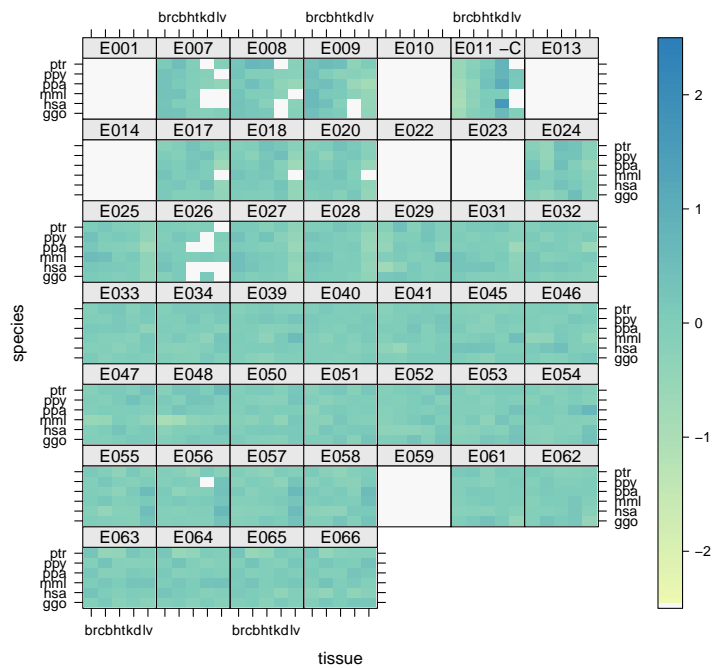
ENSG0000091136



ENSG0000091157



ENSG0000091428





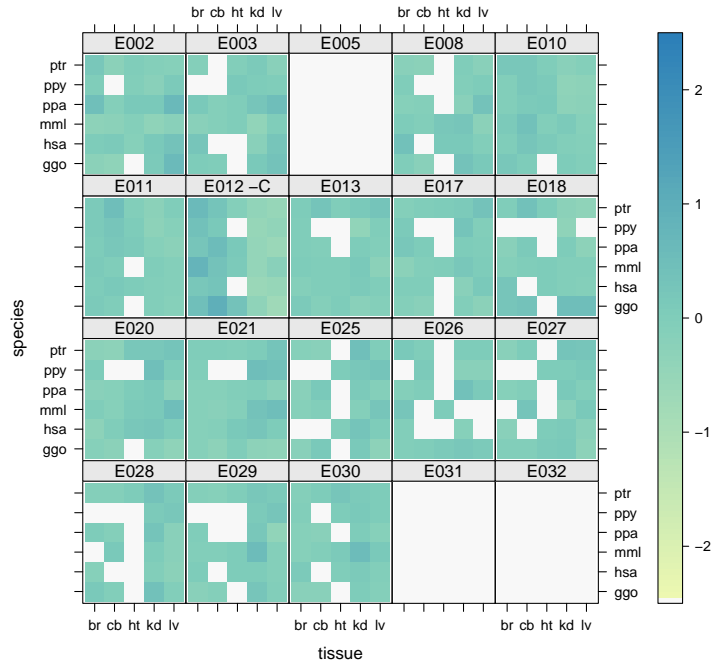




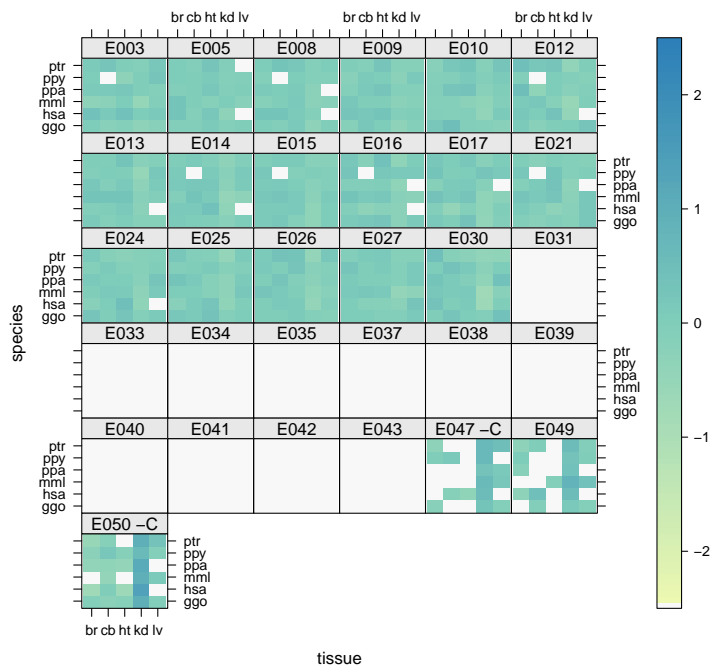




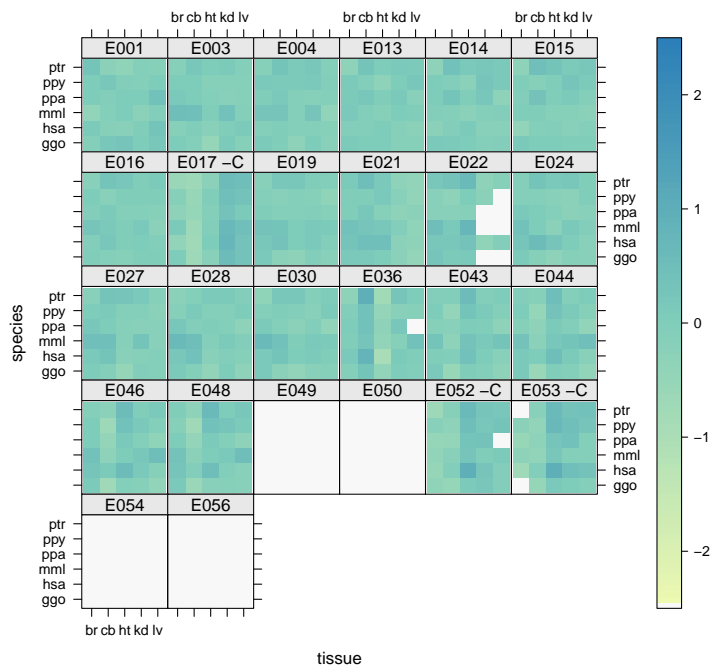
ENSG00000095585



ENSG0000099139



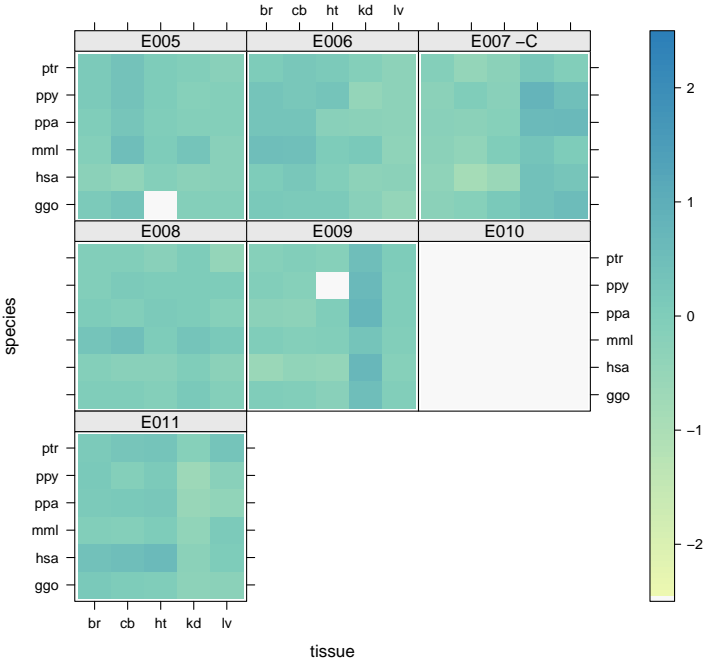
**ENSG0000099204**



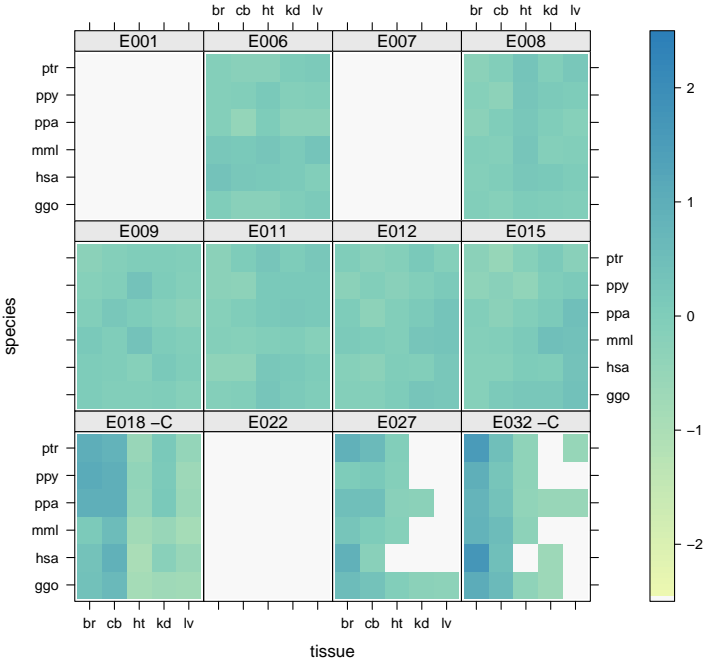




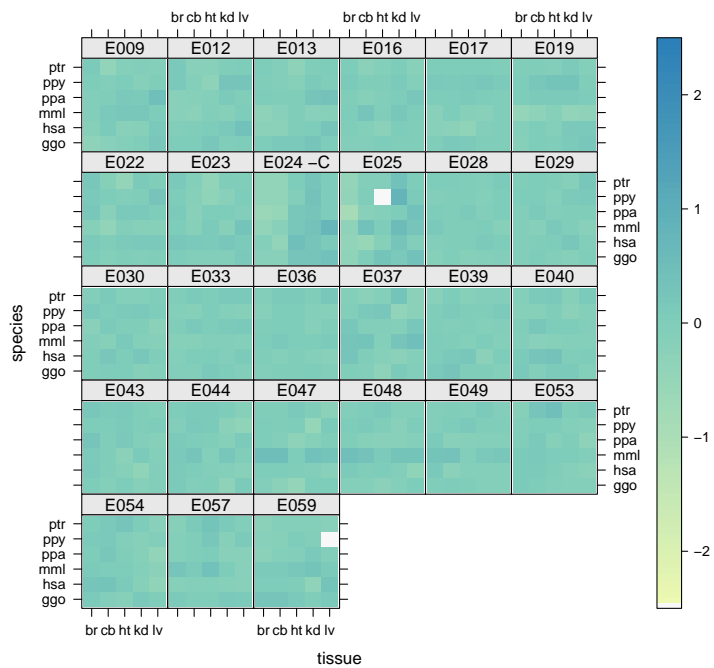
ENSG00000100105



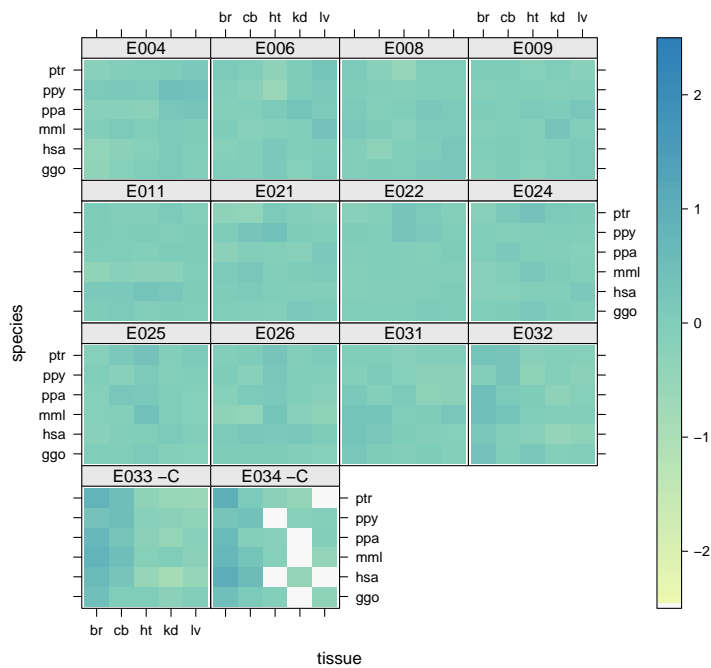
ENSG00000100142



ENSG0000100241

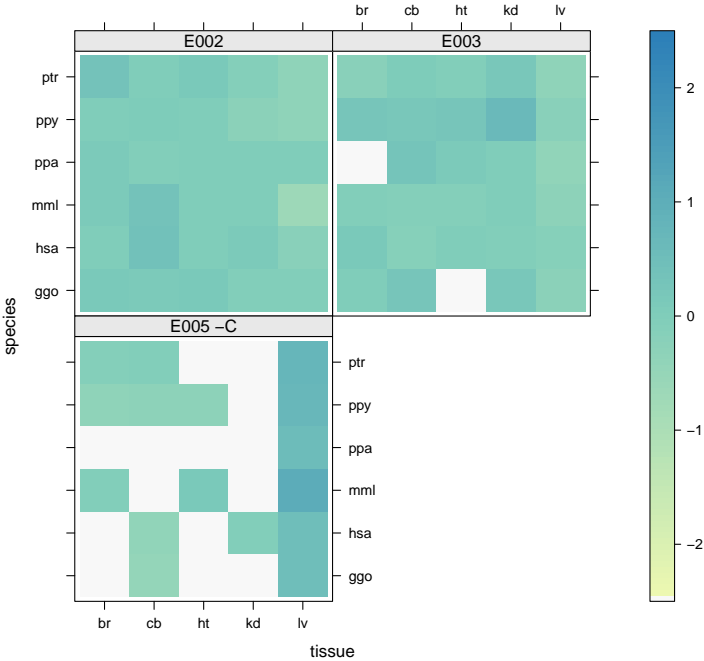


ENSG0000100364





ENSG00000100652







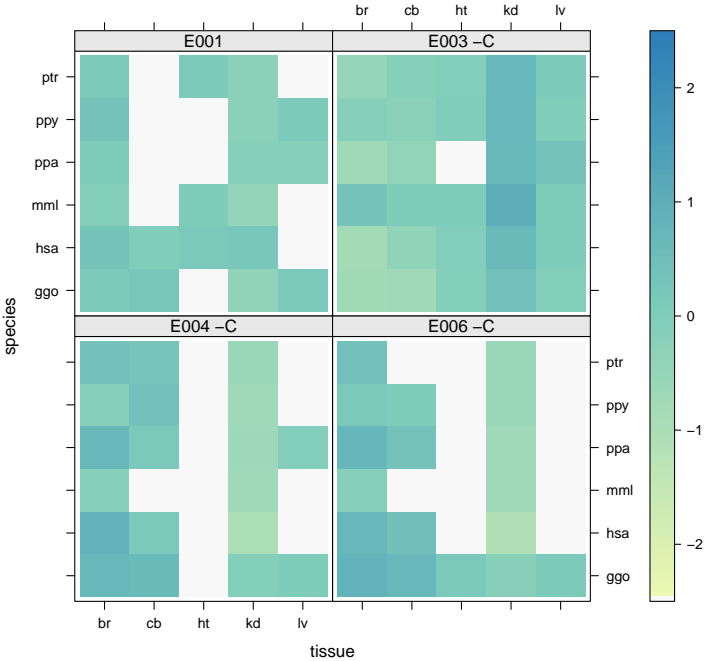




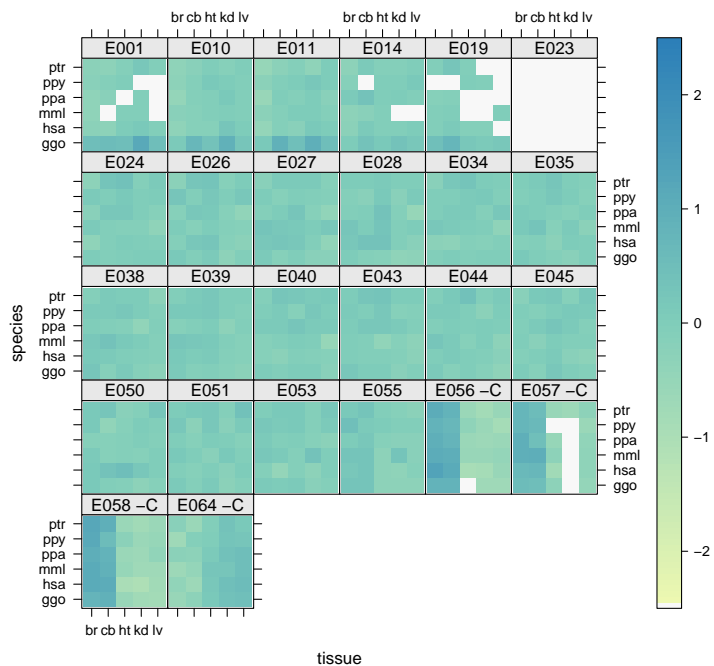




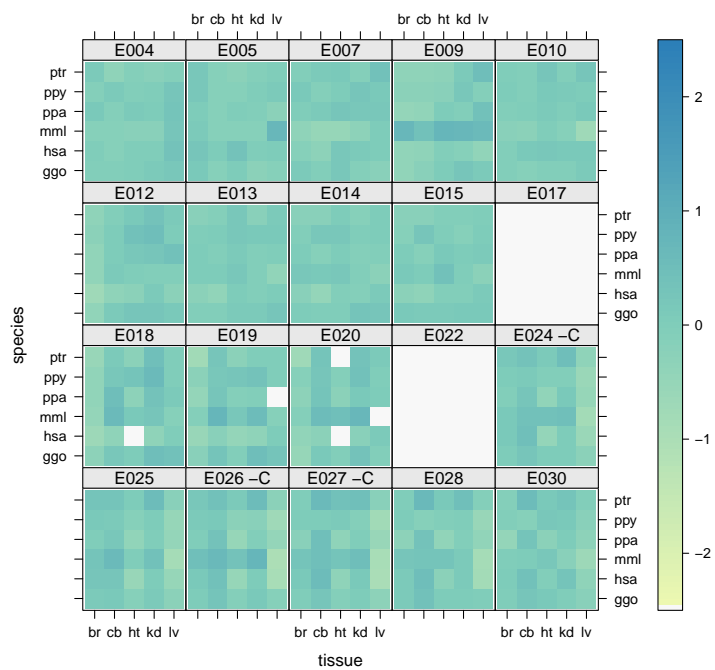
ENSG00000101443



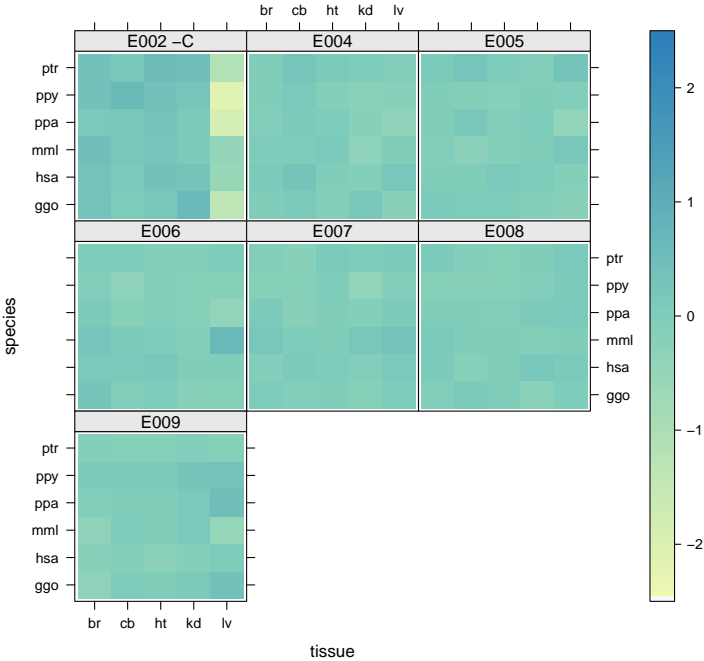
ENSG0000102606



ENSG00000102804



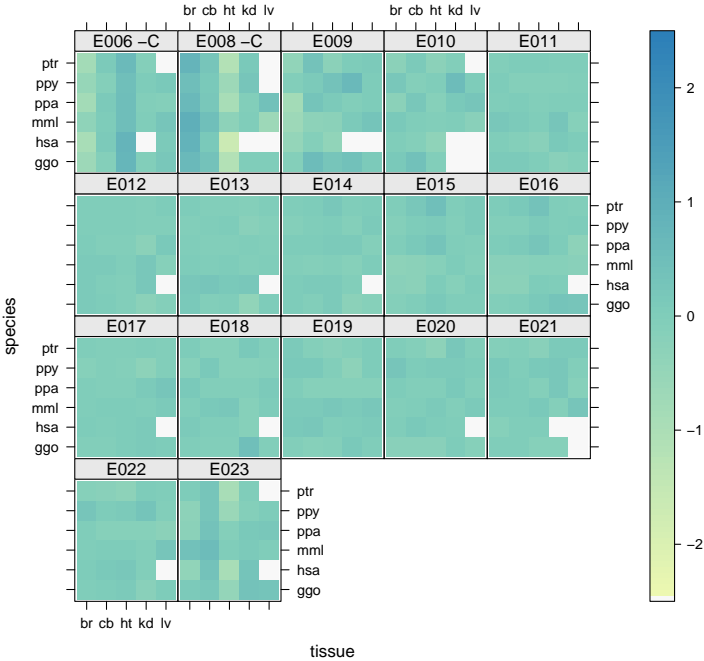
ENSG00000102967



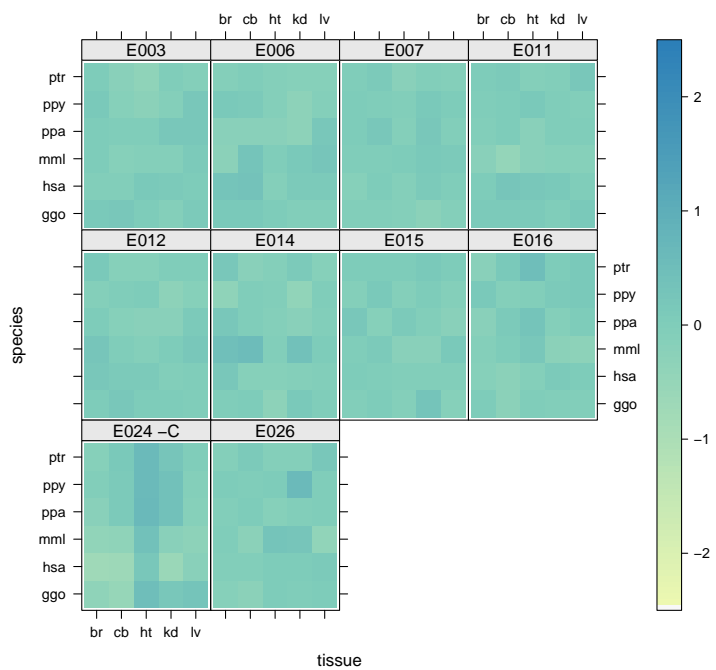




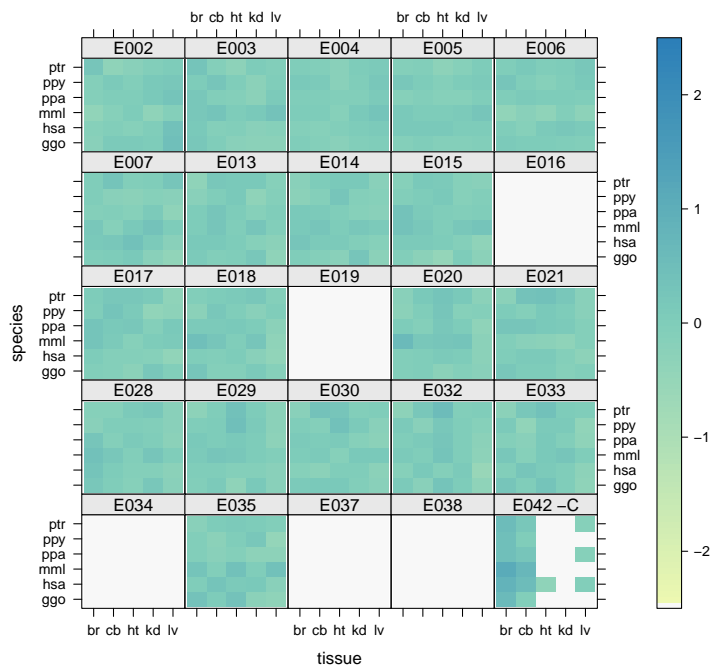
ENSG00000103034



ENSG00000103168

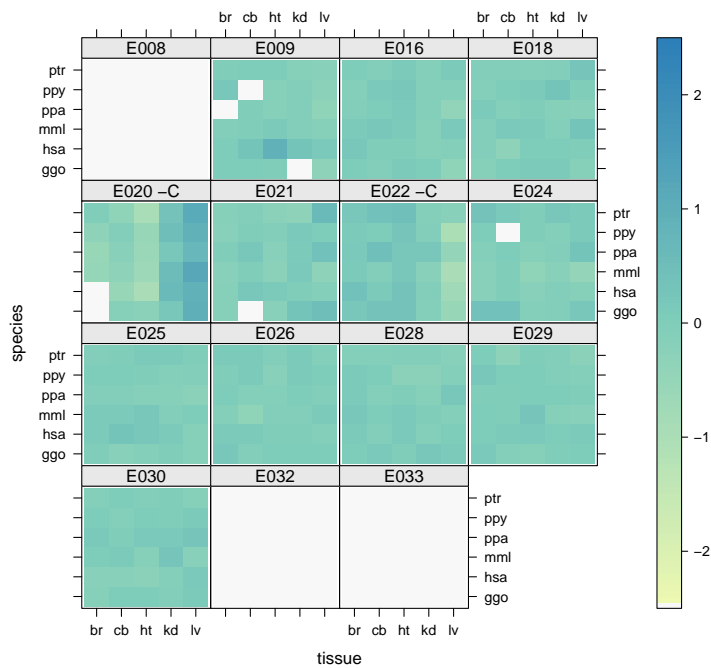


ENSG00000104067





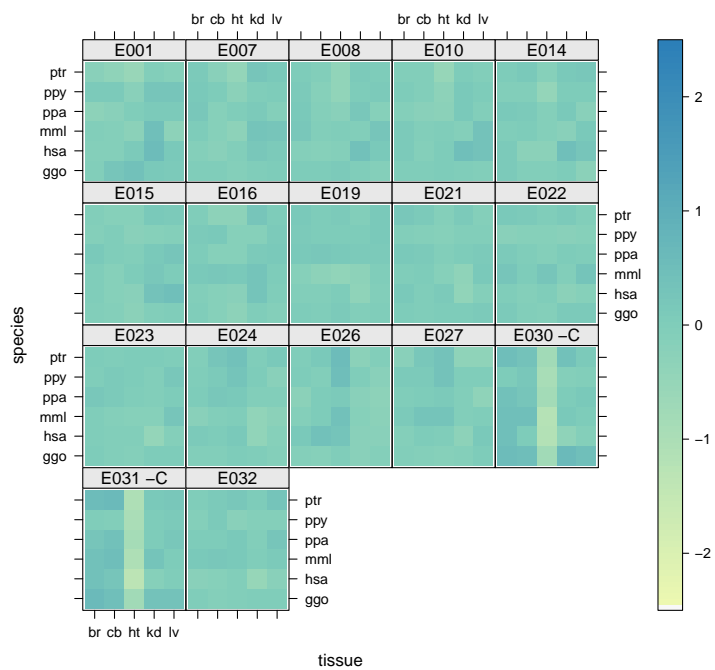
ENSG00000104635







ENSG00000104936

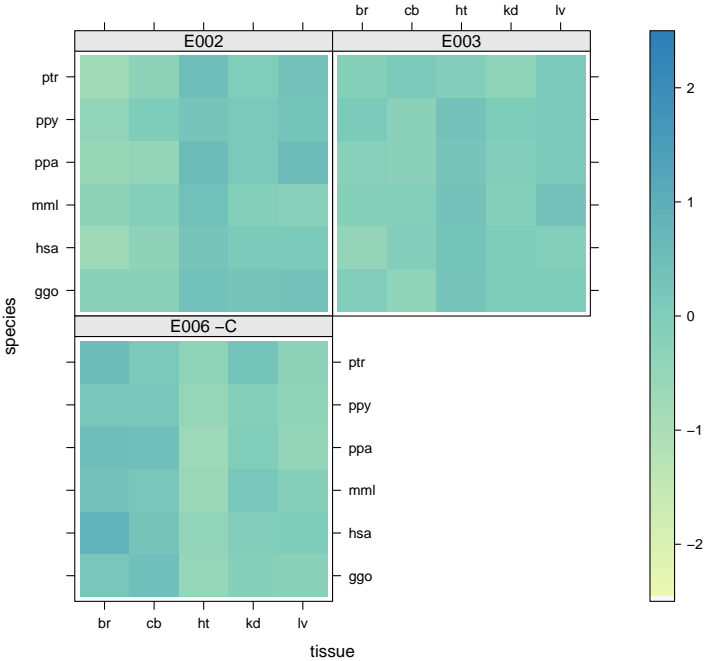




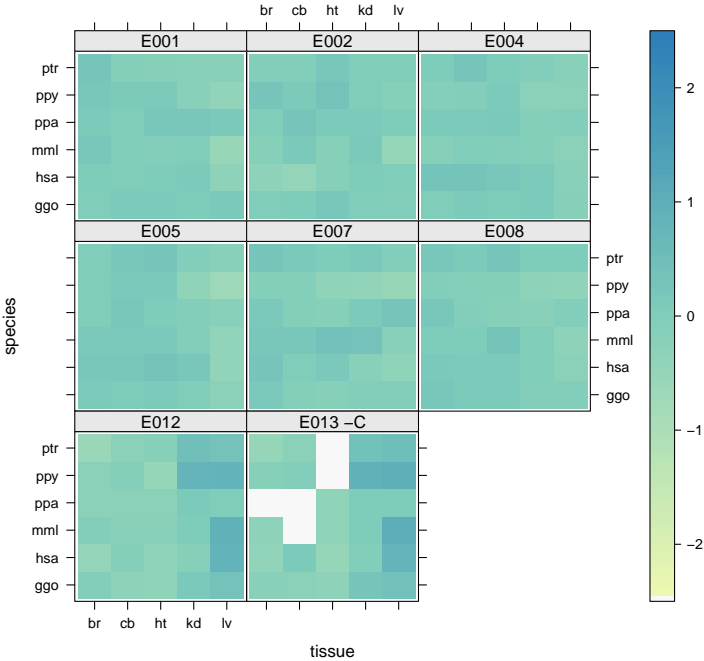




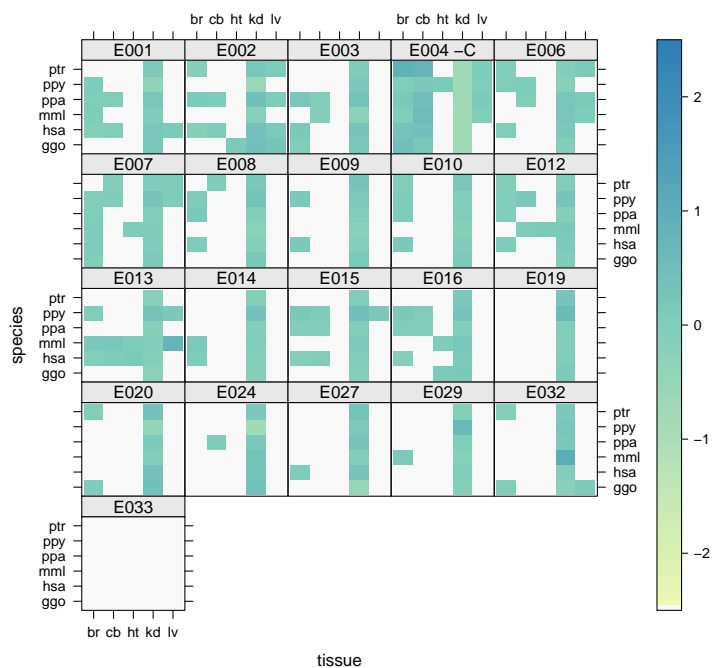
ENSG00000105472



ENSG00000105552



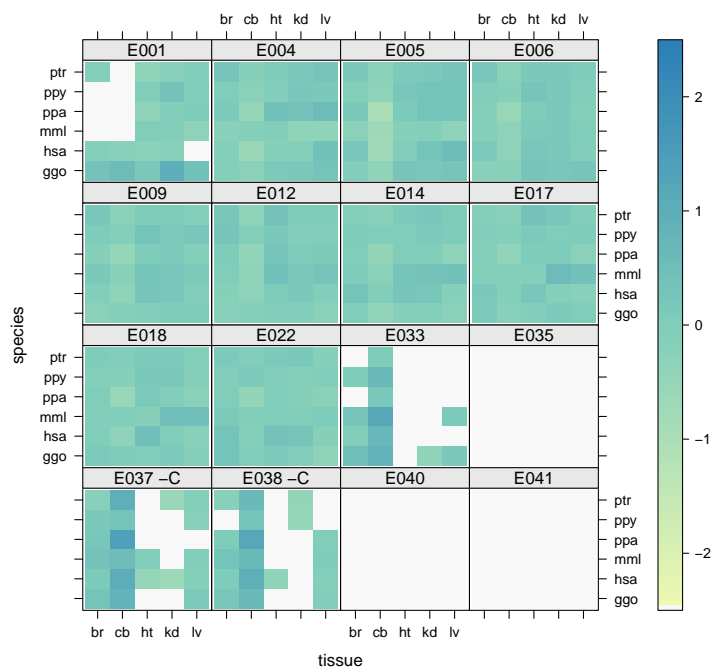
ENSG00000105929







ENSG0000106066

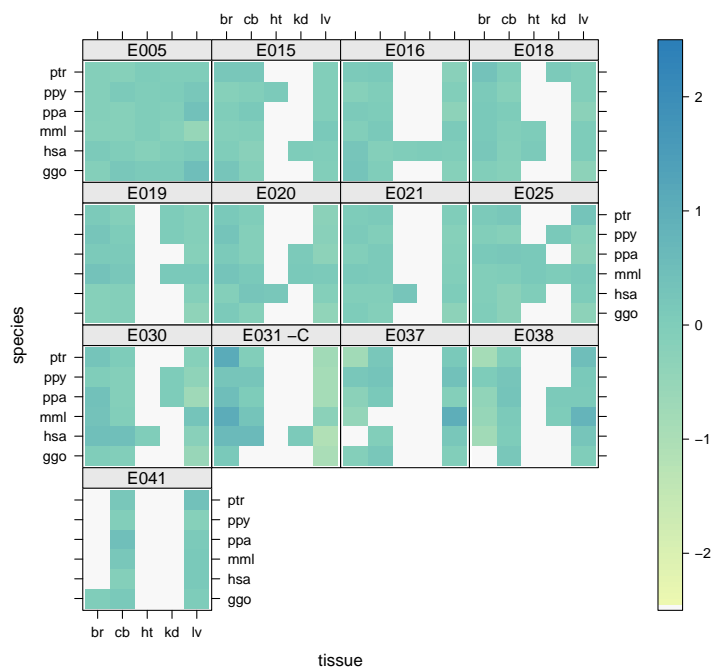








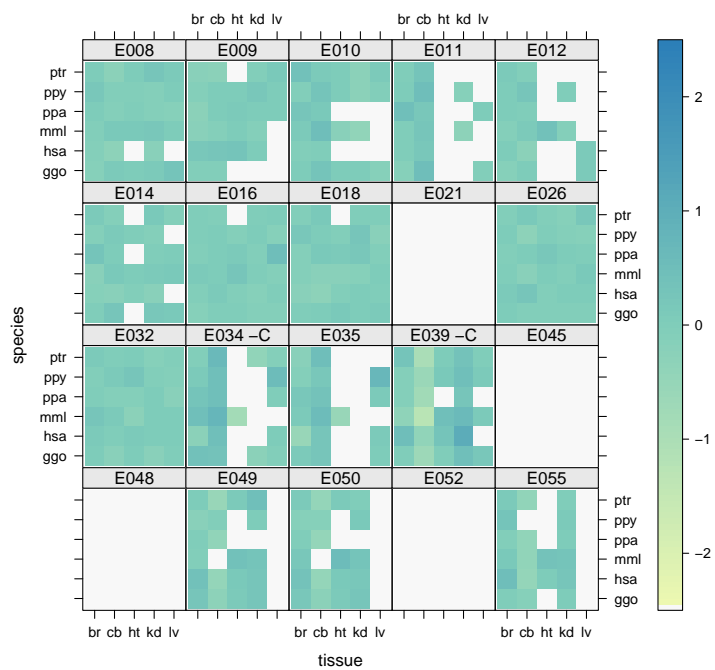
ENSG00000106327



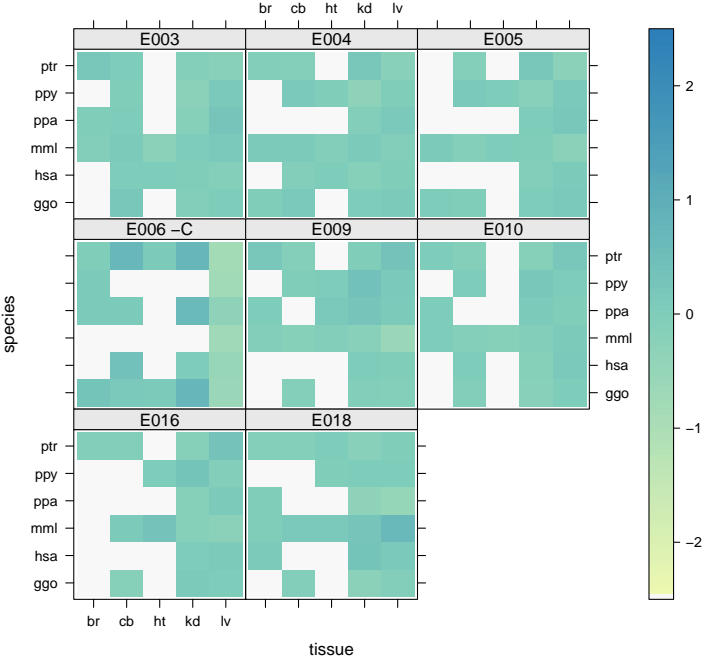




ENSG00000106772



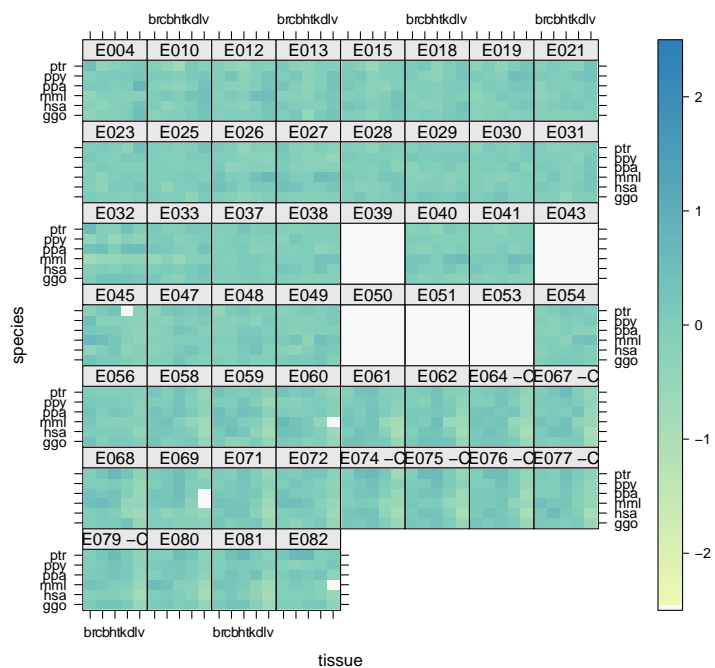
ENSG00000106927



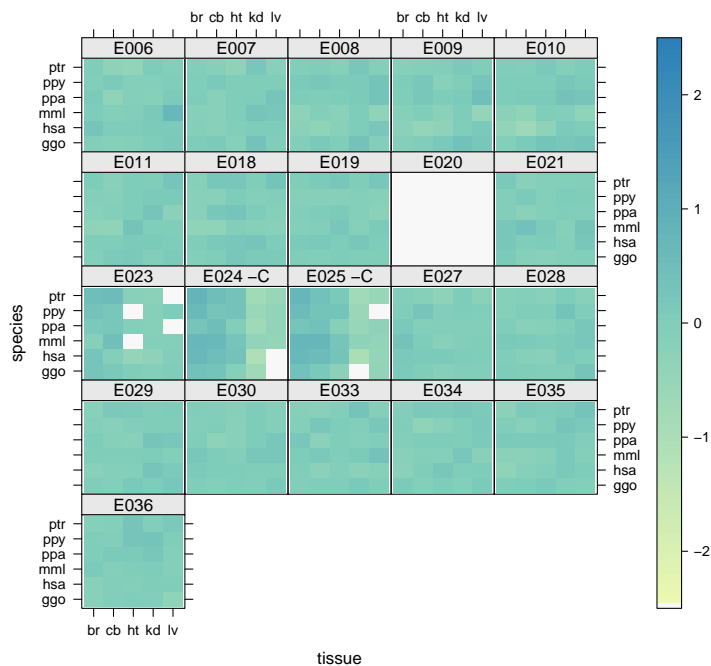




ENSG0000107186

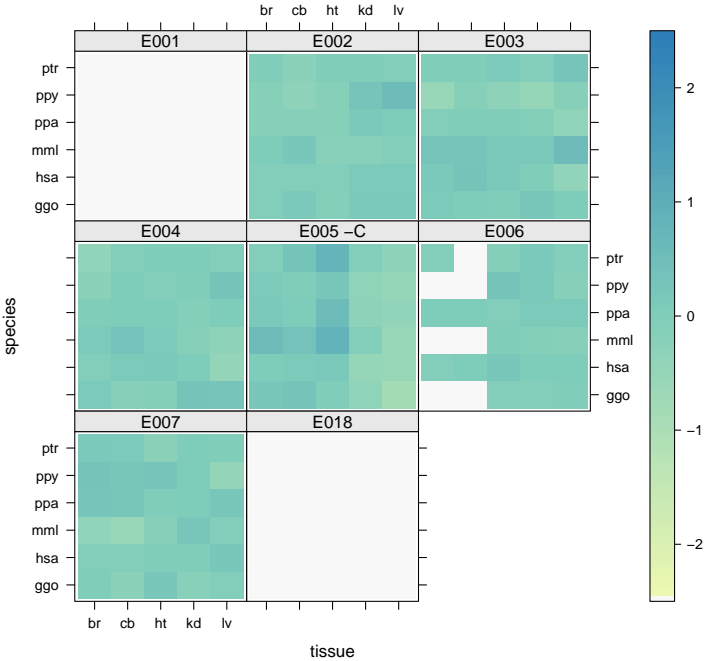


ENSG00000107263

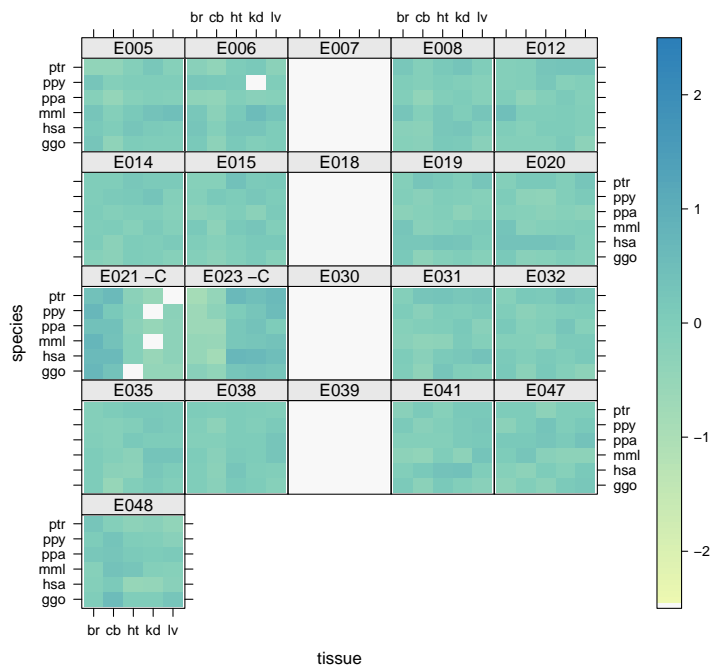




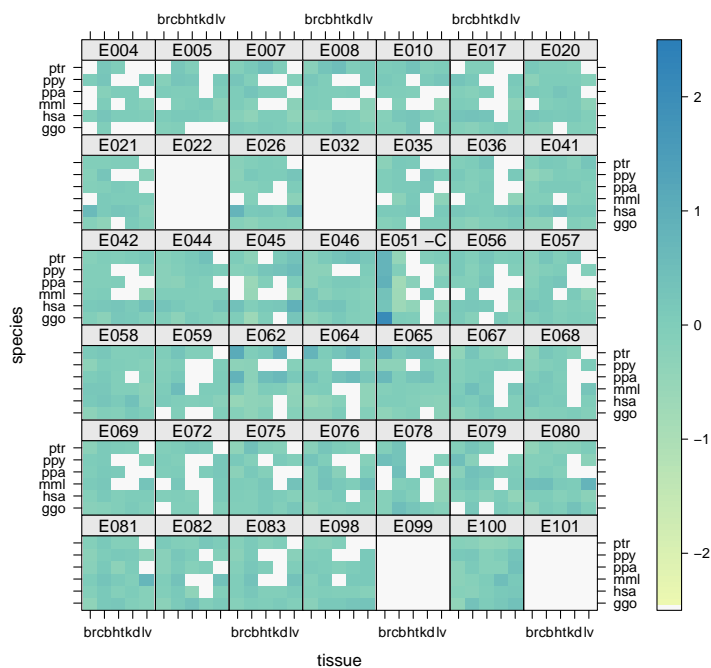
ENSG00000107562



ENSG00000107643

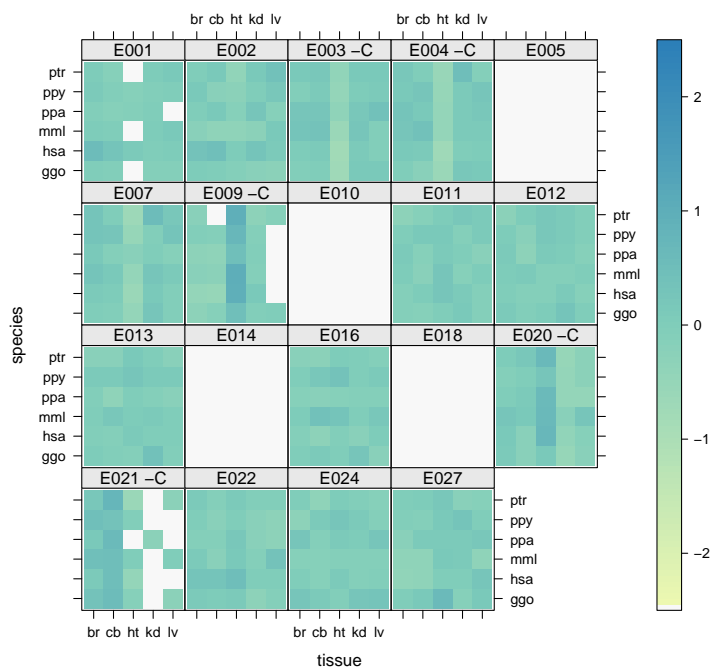


ENSG0000107736



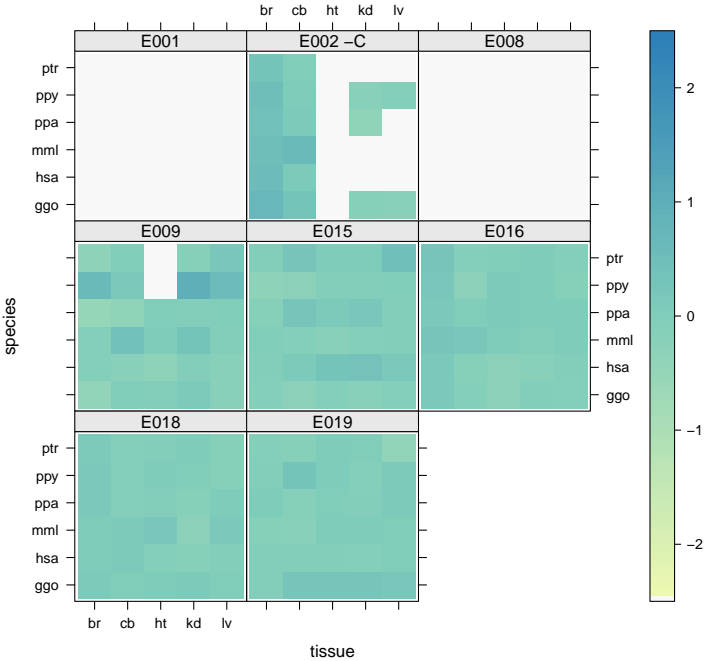


ENSG00000107771

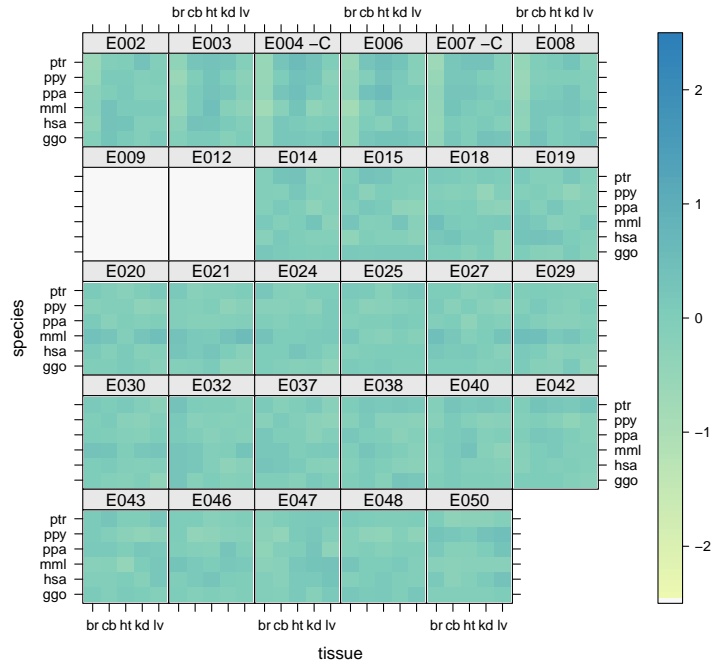




ENSG00000107872

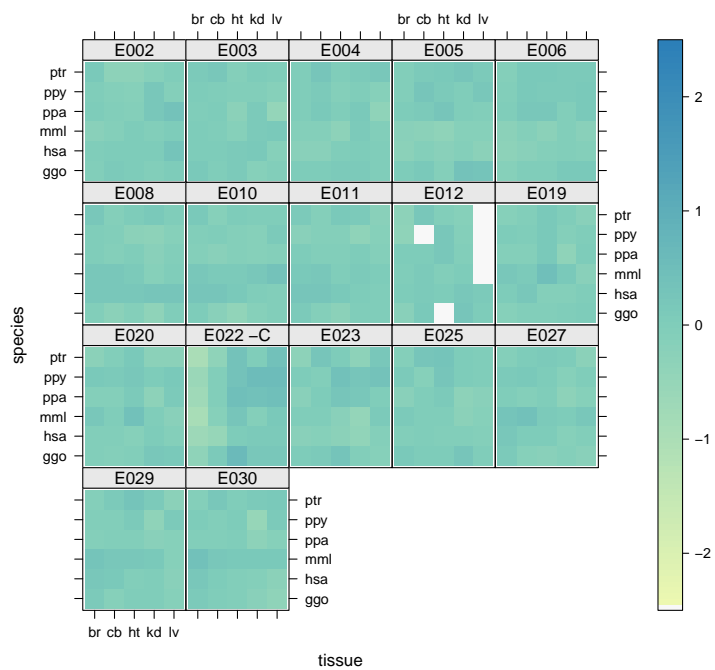


ENSG0000108175

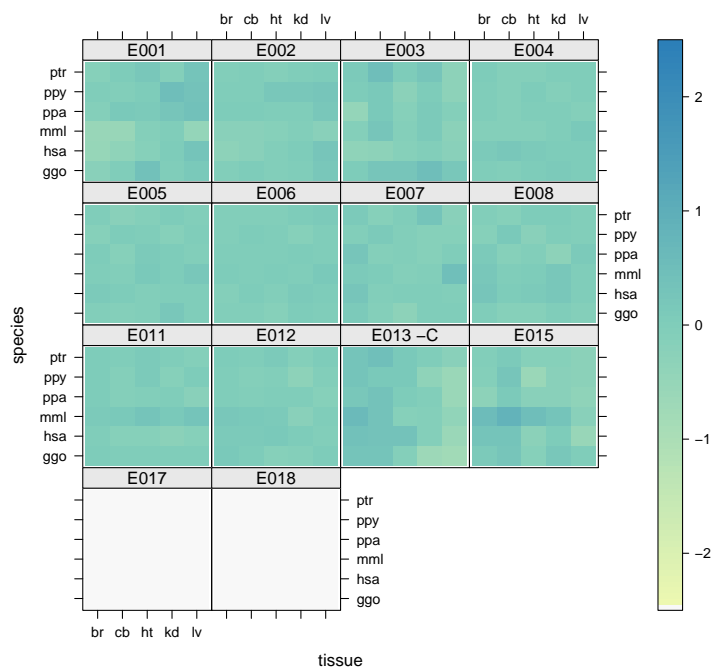




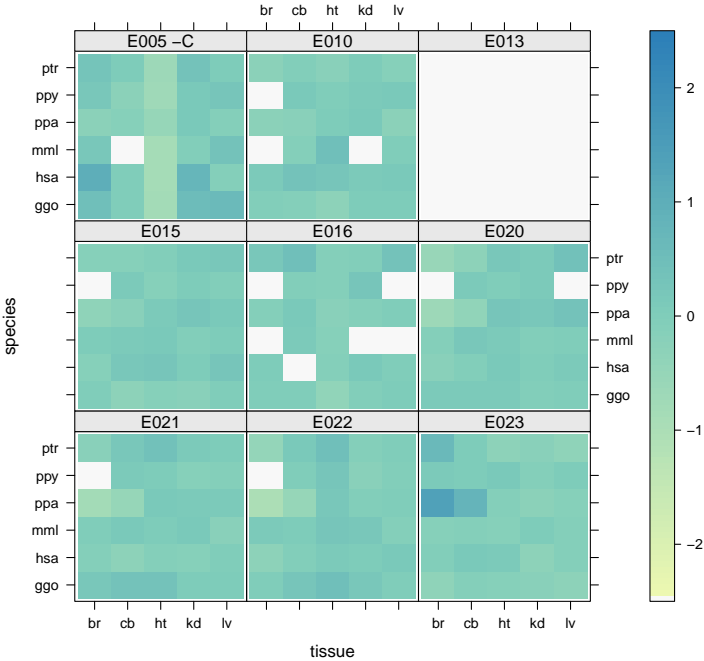
ENSG00000108262



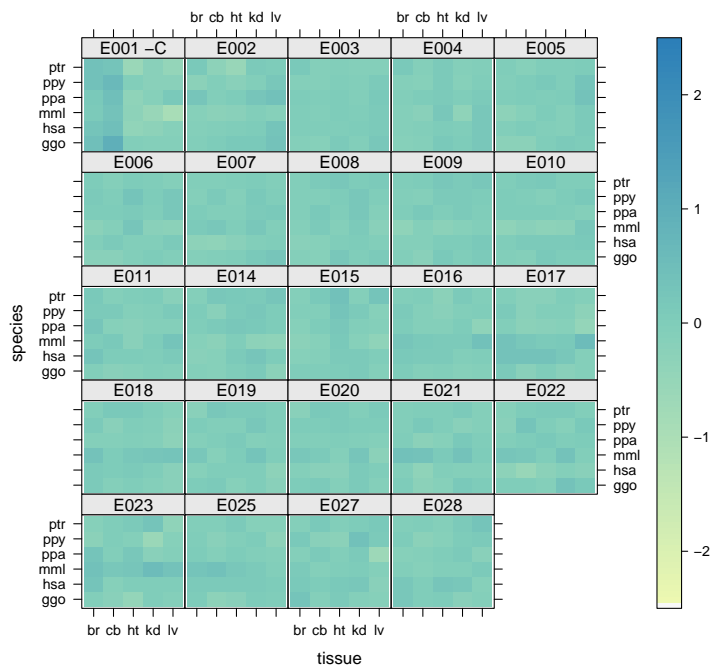
ENSG0000108387



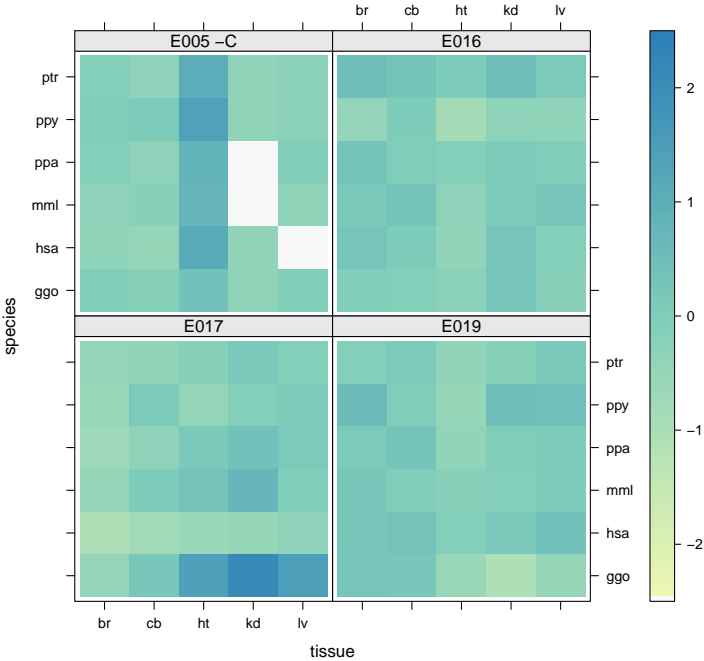
ENSG0000108515



ENSG00000108840

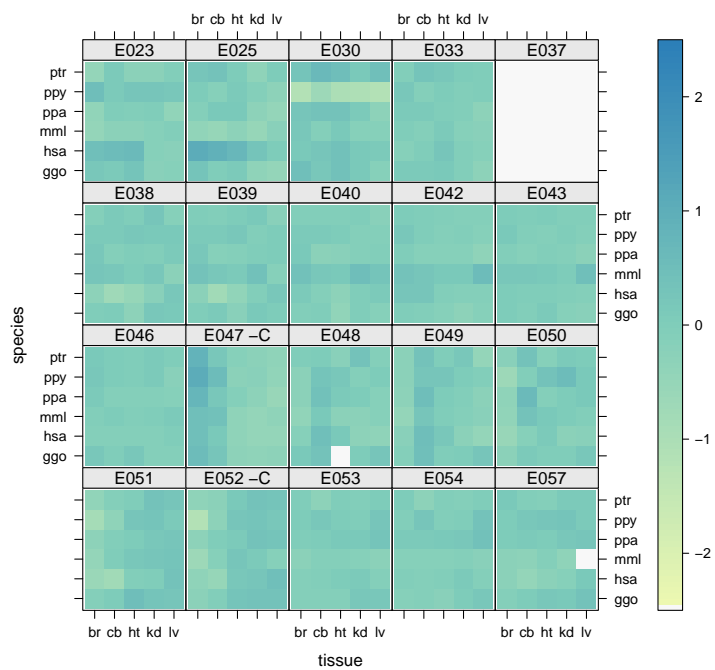


ENSG00000108953





ENSG00000109180



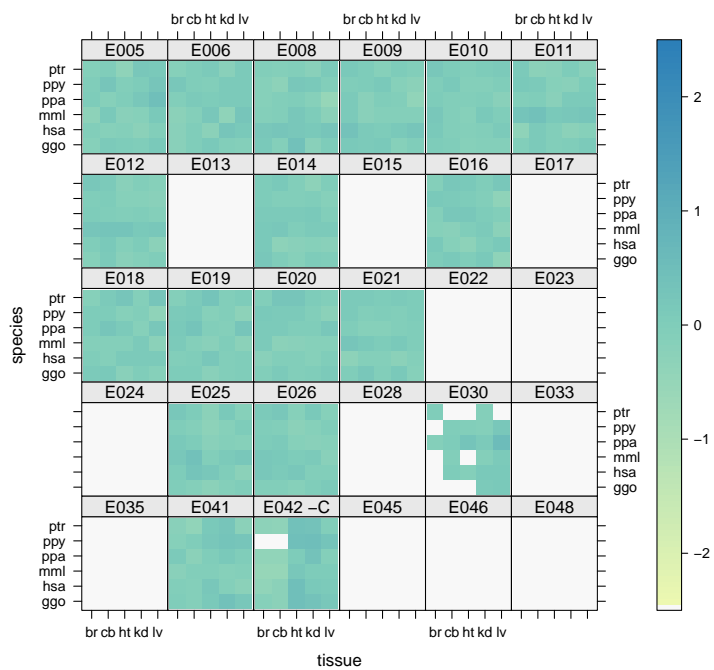








ENSG0000109819

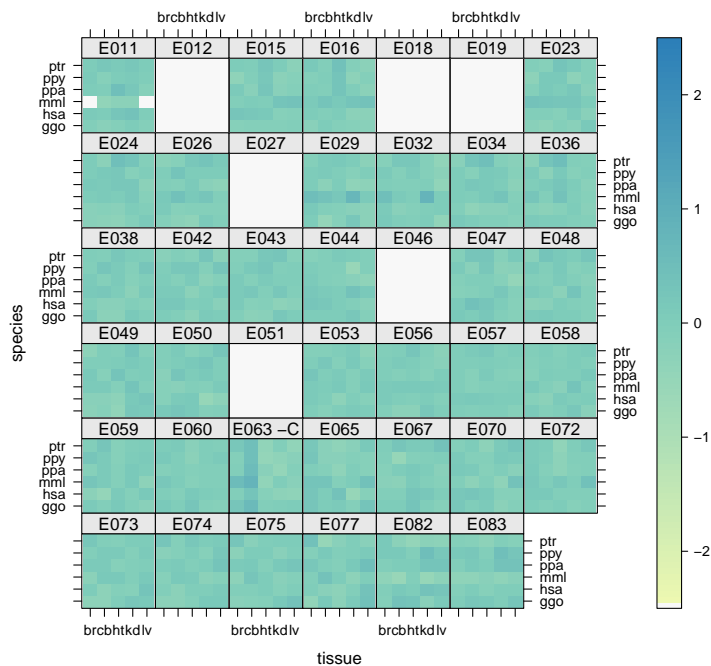






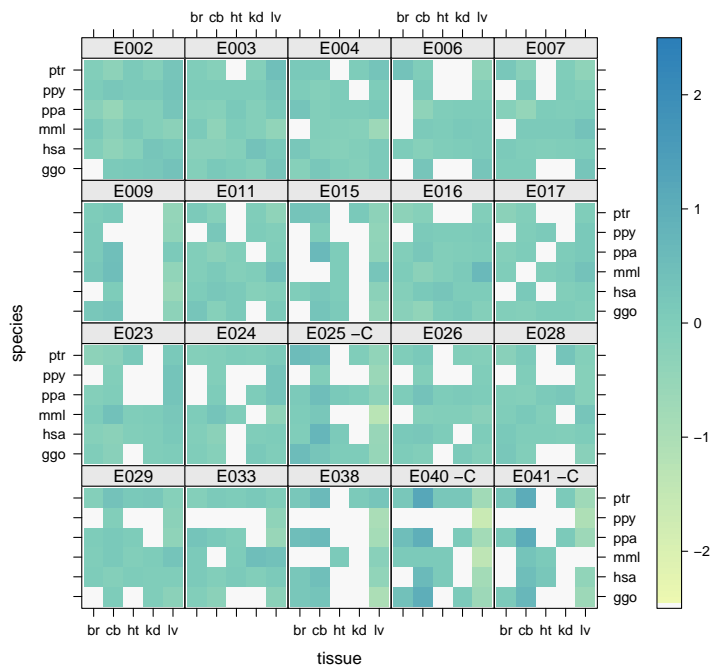


ENSG0000110075



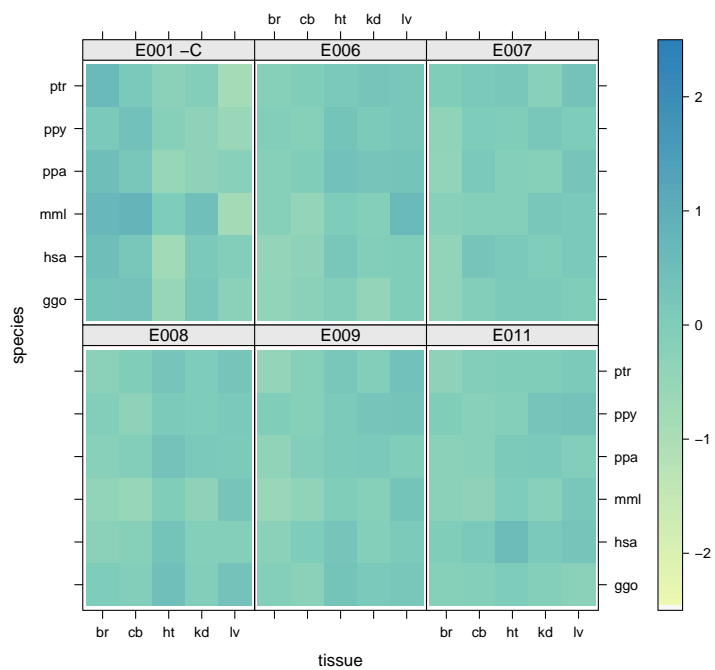


ENSG00000110169



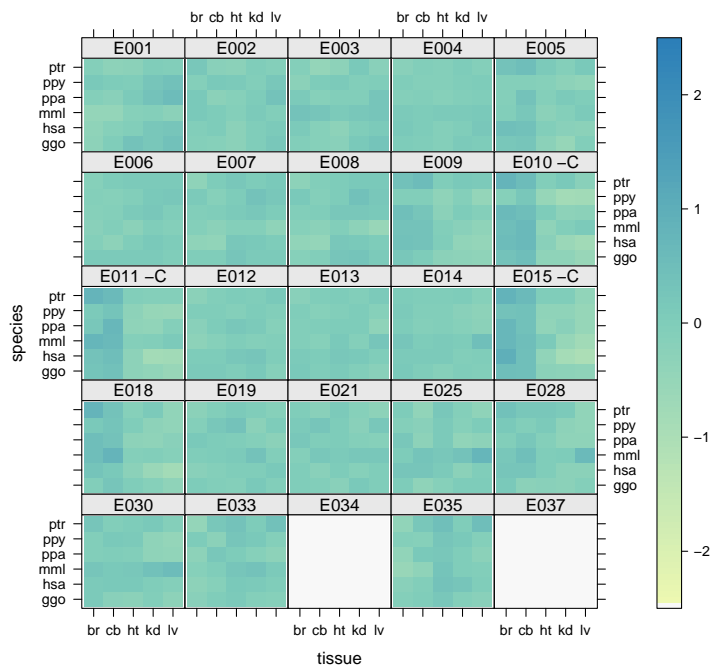


ENSG00000111412

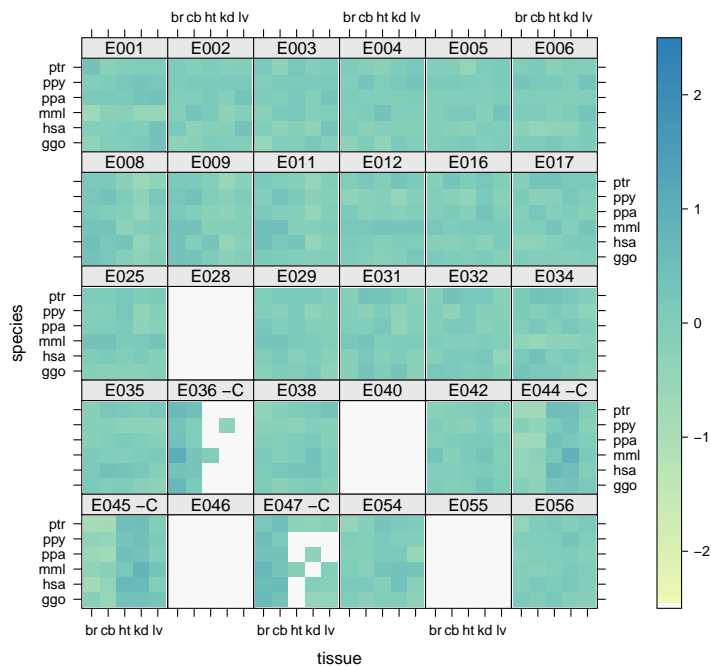




ENSG00000111684

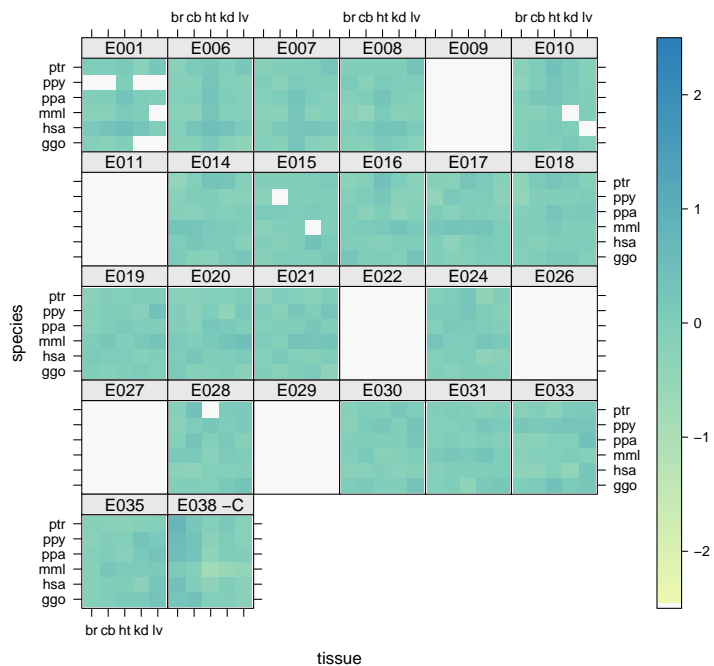


ENSG0000111731

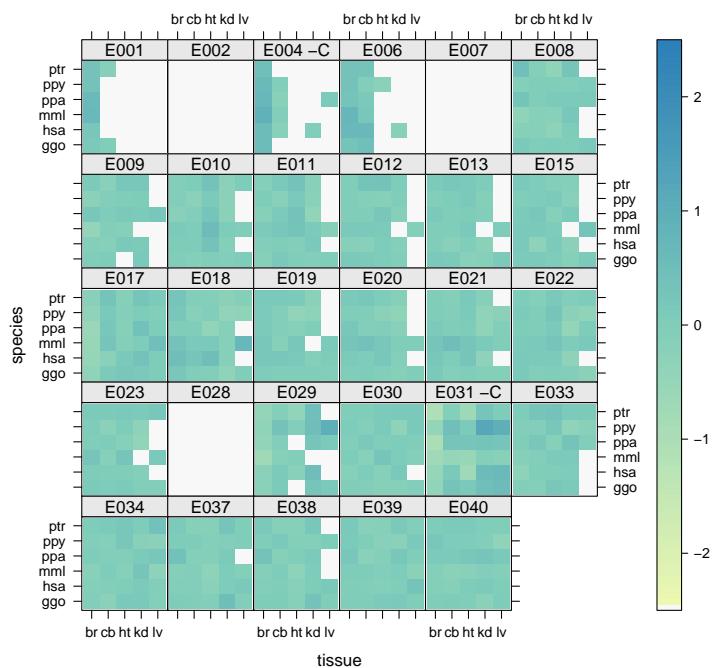




ENSG0000111785

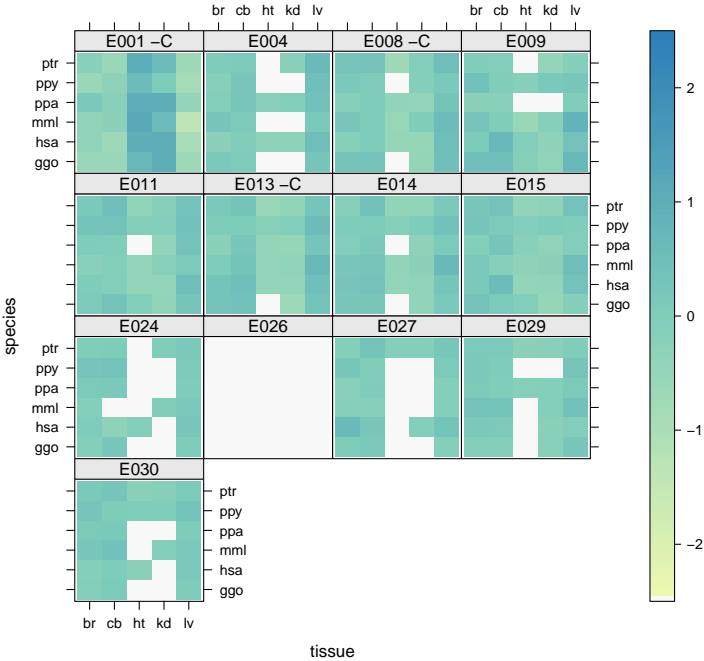


ENSG0000112137

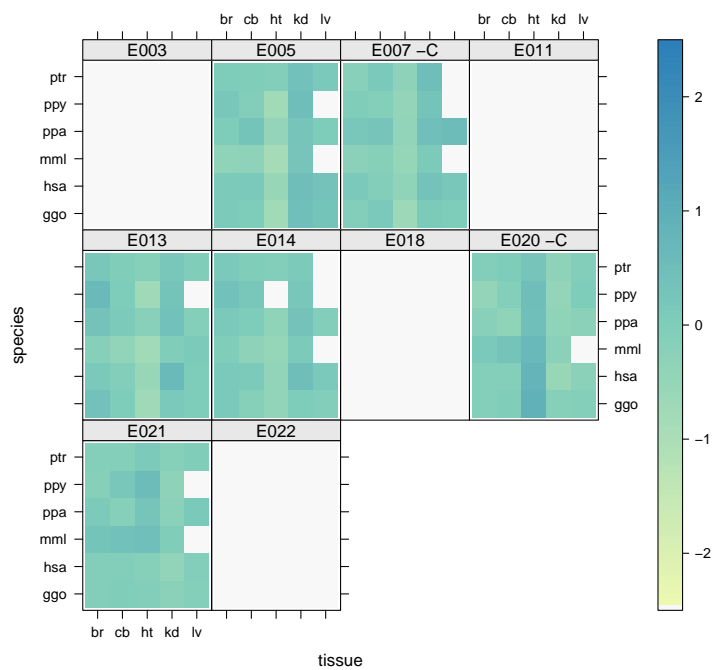




ENSG00000112293



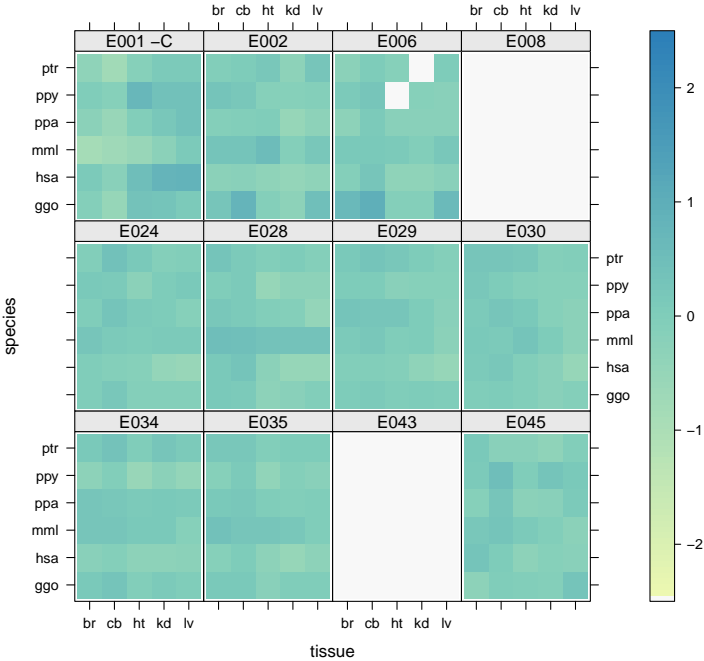
ENSG00000112530





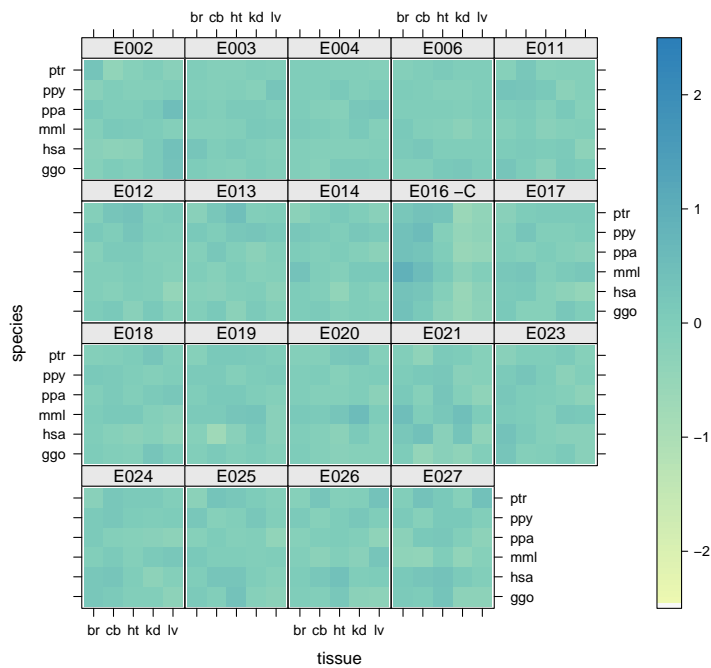


ENSG00000113141



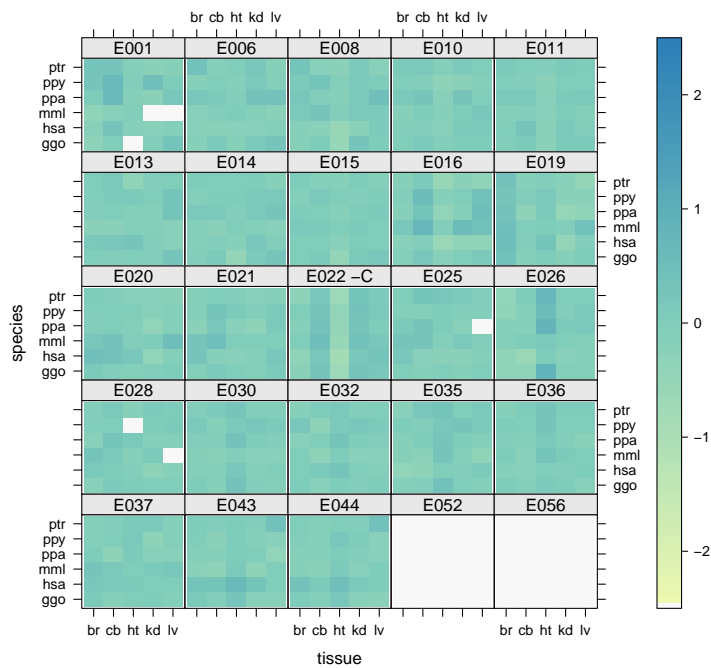


ENSG00000113163

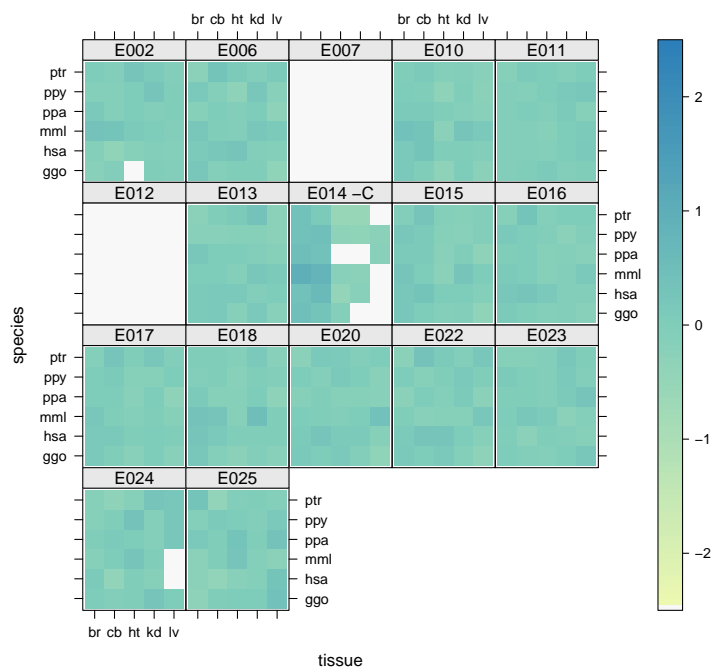




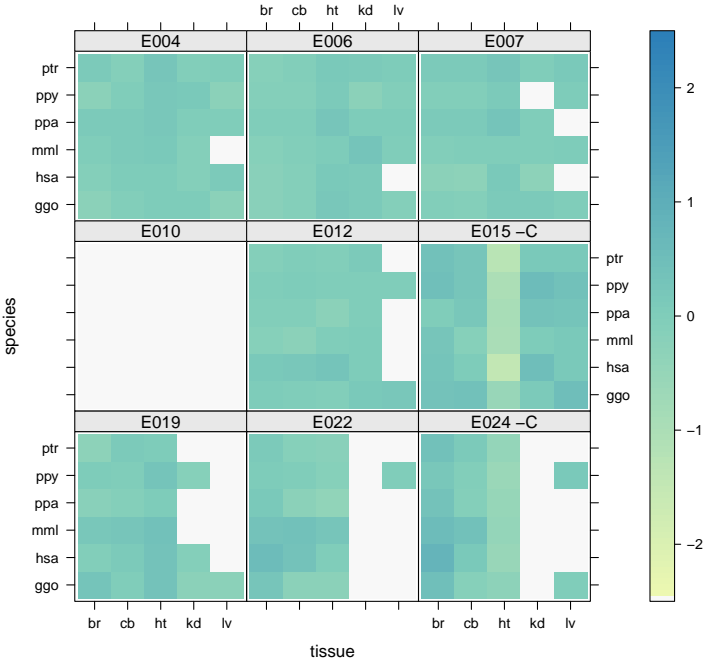
ENSG00000113648



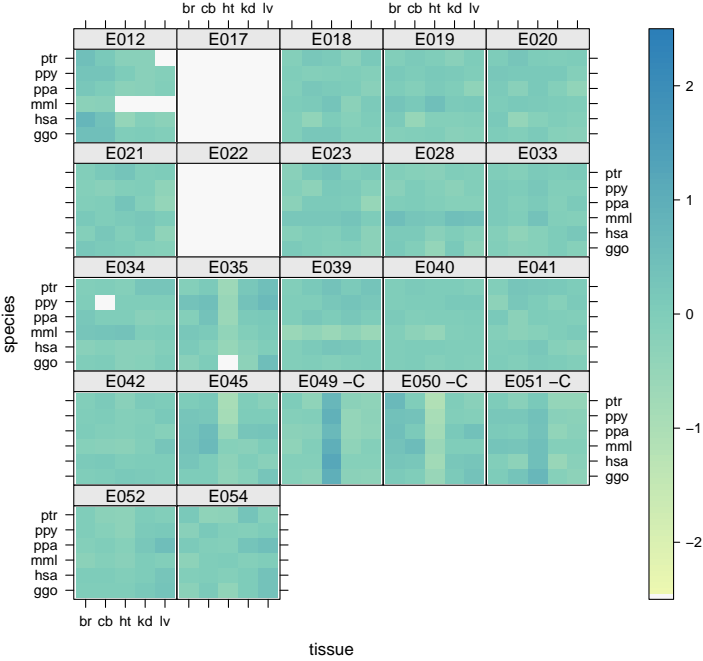
ENSG00000113742



ENSG00000114279

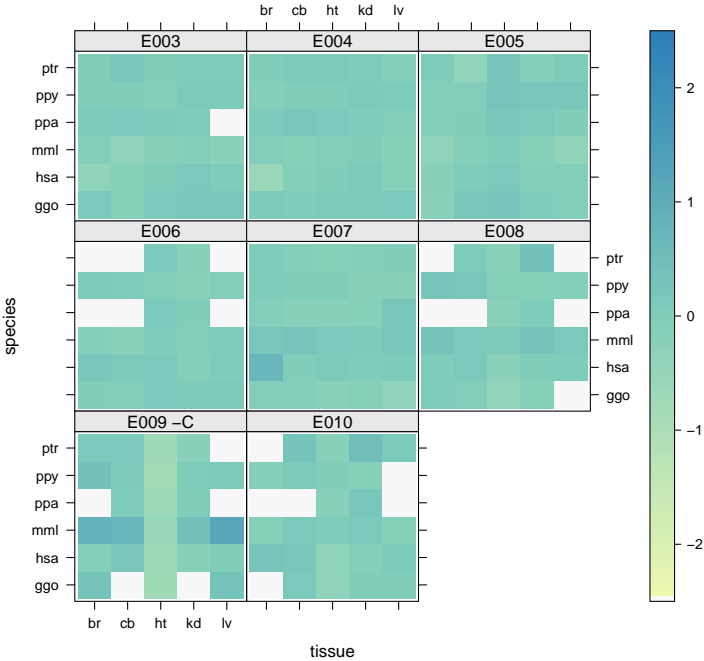


ENSG00000114416





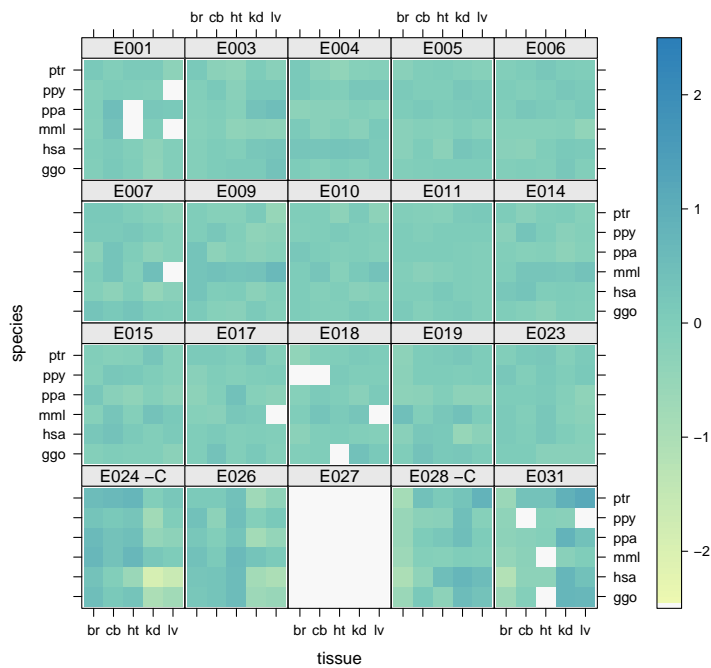
ENSG00000114854



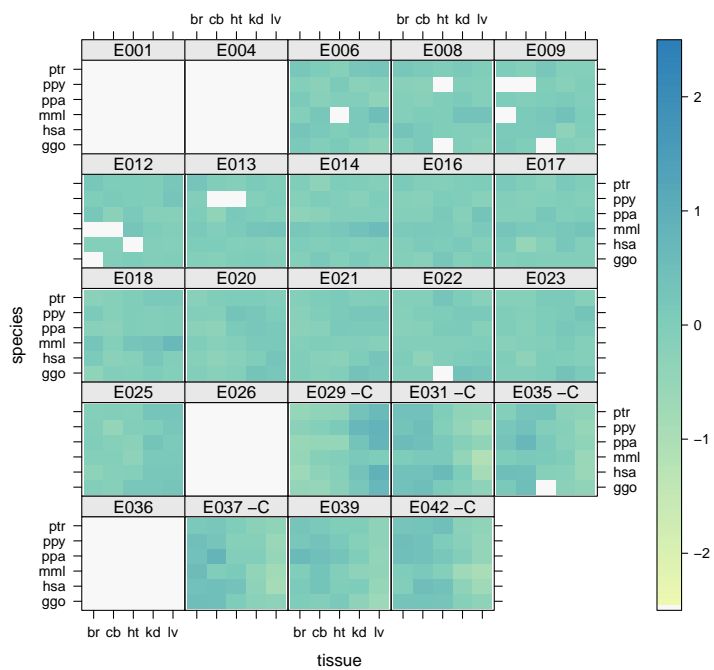




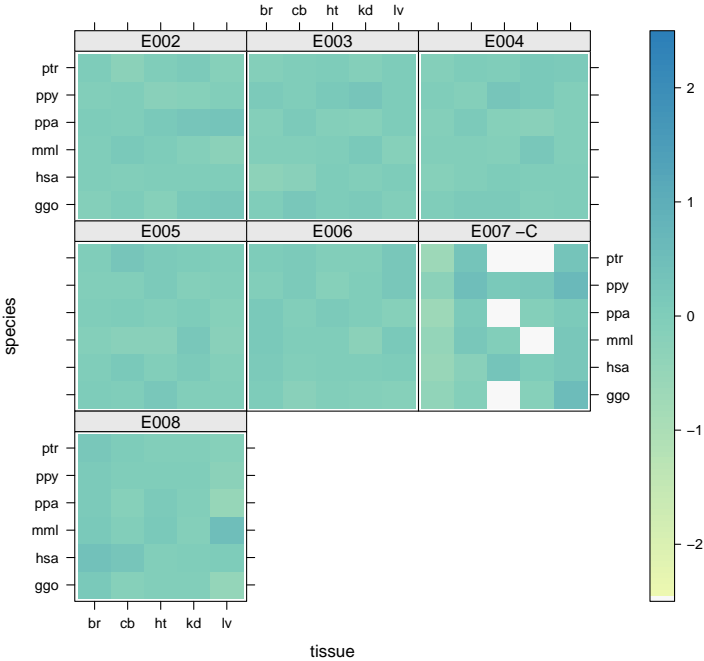
ENSG00000114993



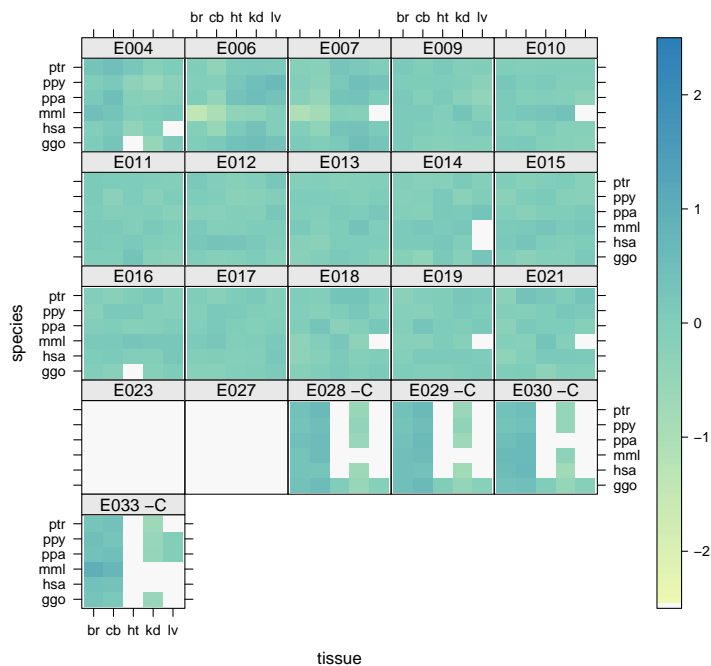
ENSG00000115109



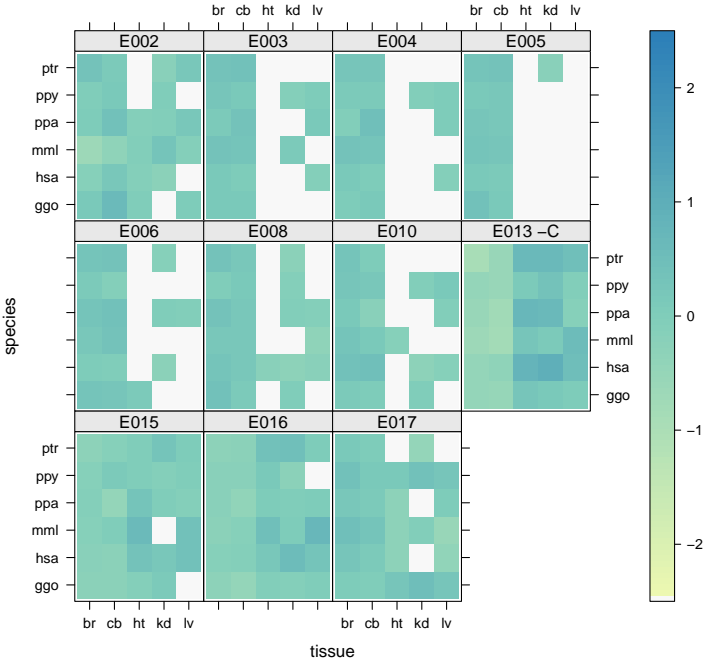
ENSG00000115226



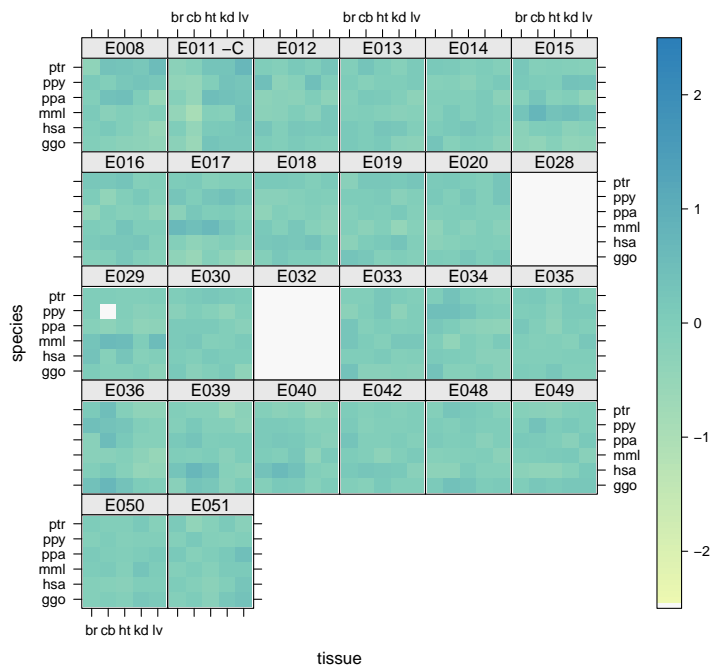
ENSG00000115252



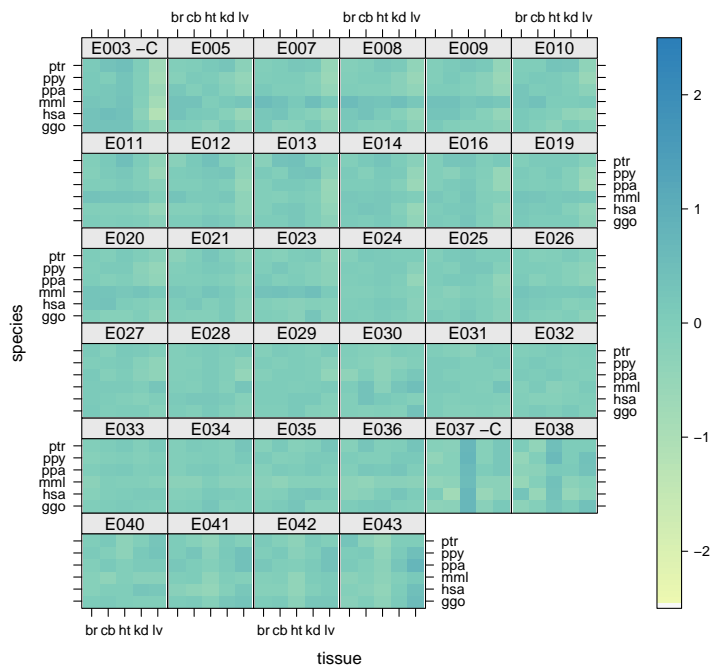
ENSG00000115266



ENSG00000115282



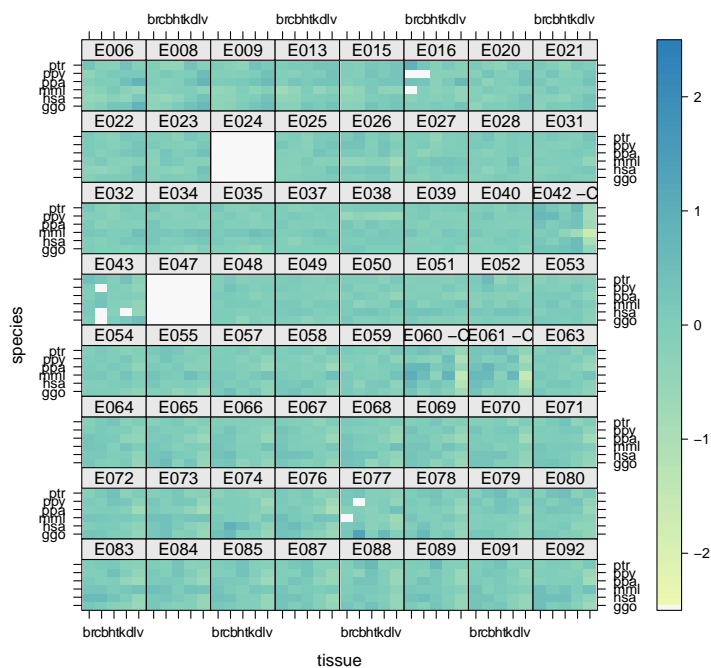
ENSG00000115306



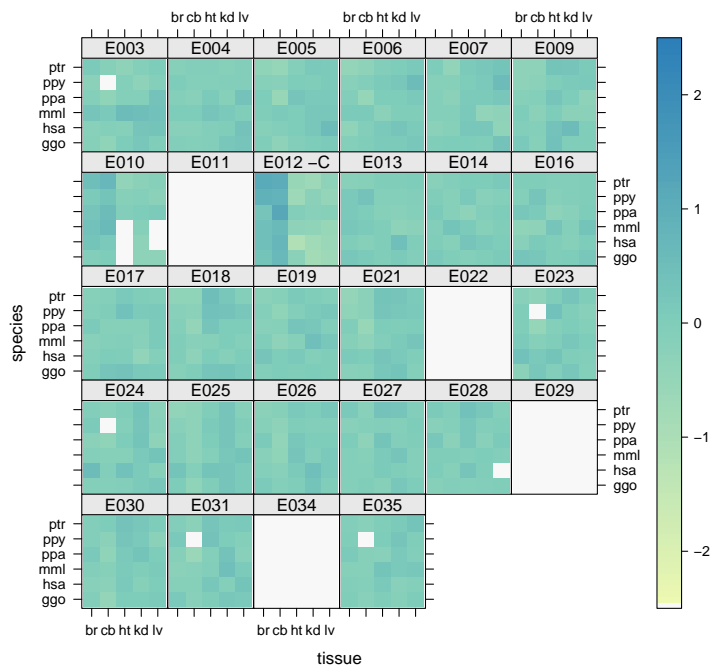




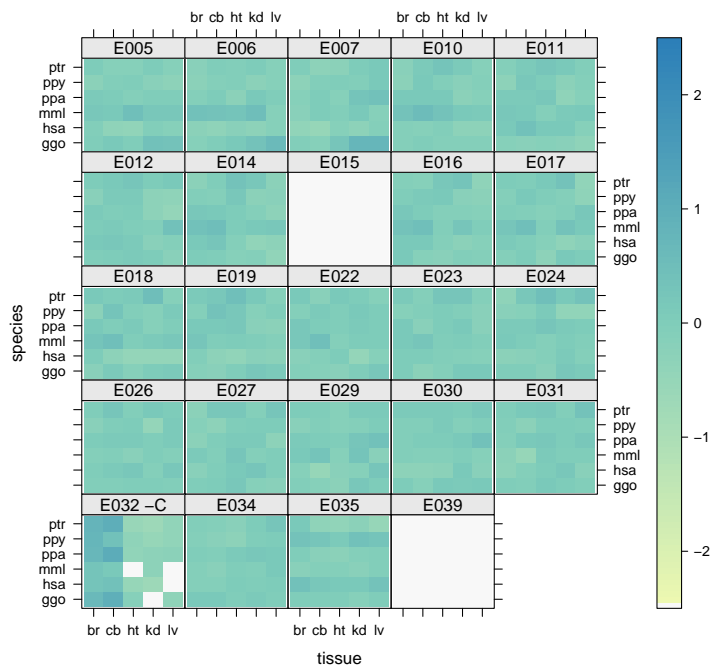
ENSG0000115414



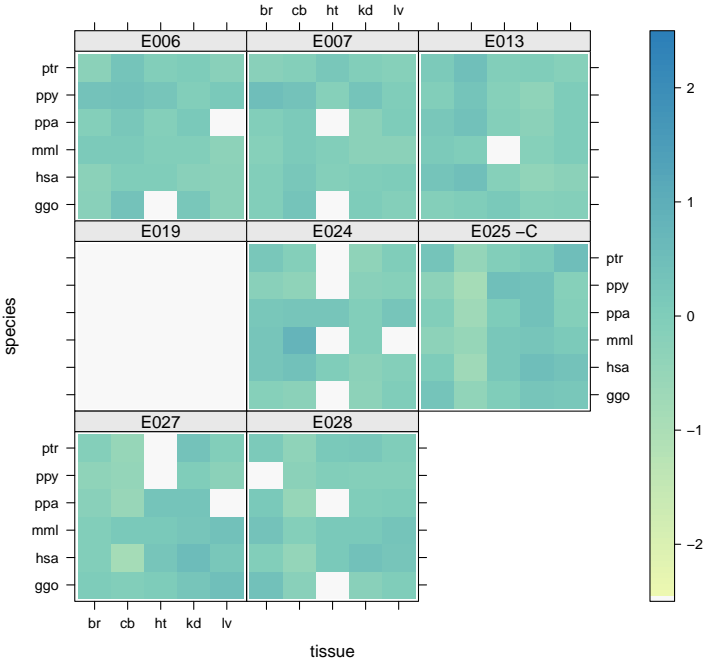
ENSG00000115825



ENSG00000115839



ENSG00000116151



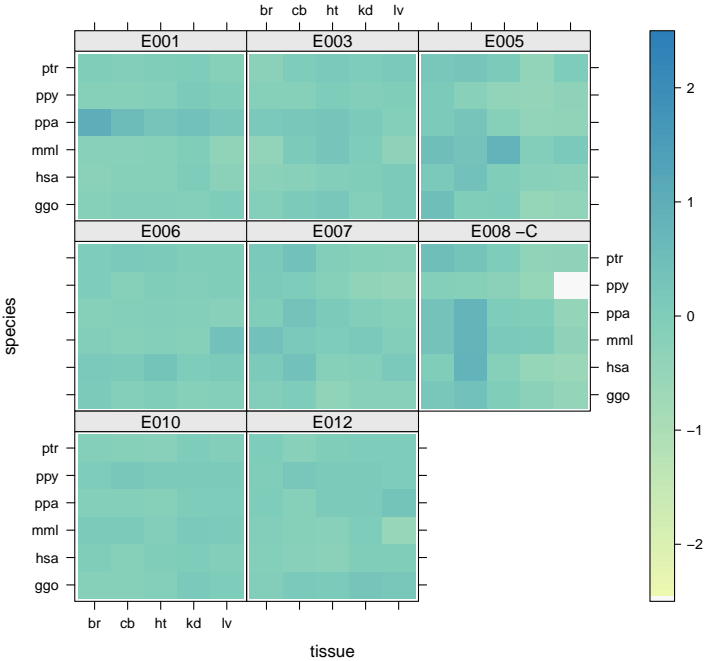




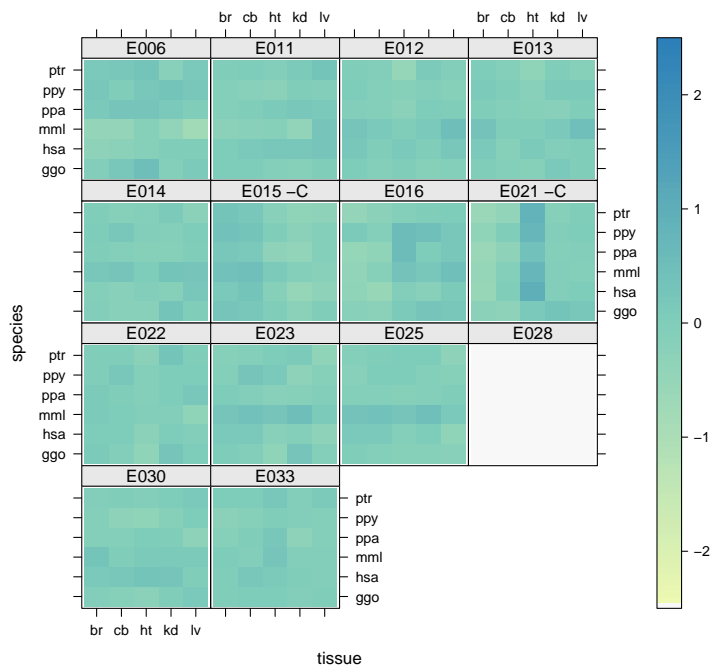




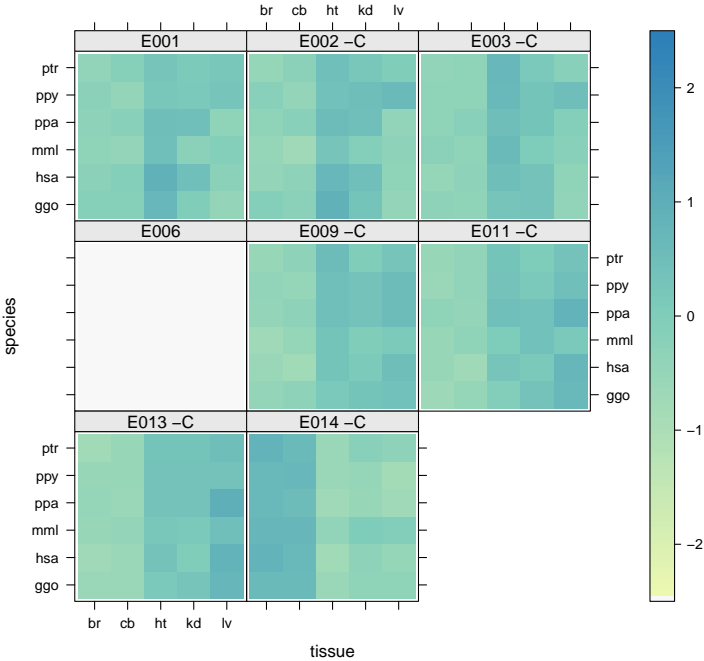
ENSG00000116586



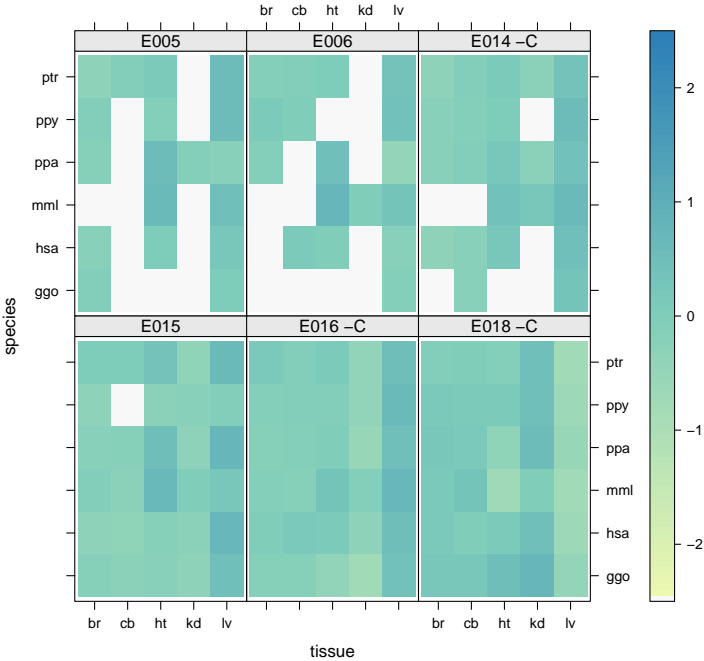
ENSG00000116604



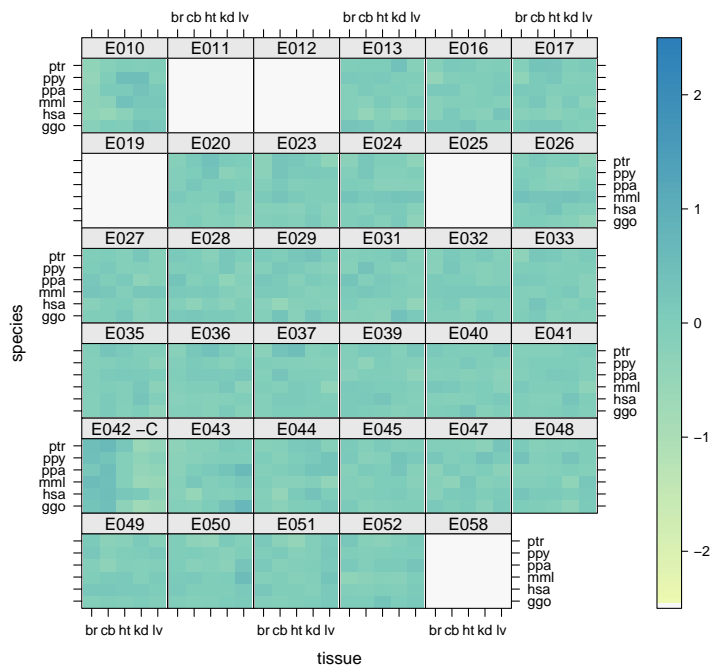
ENSG00000116667



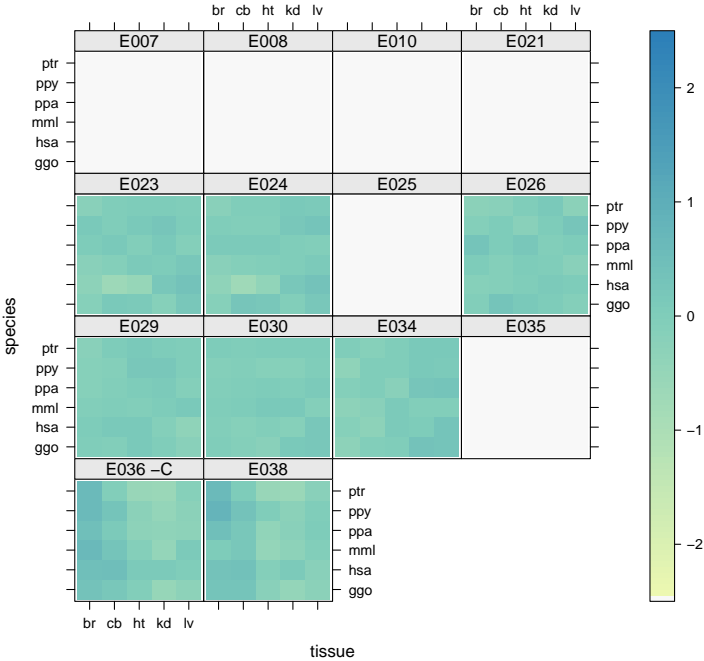
ENSG00000116690



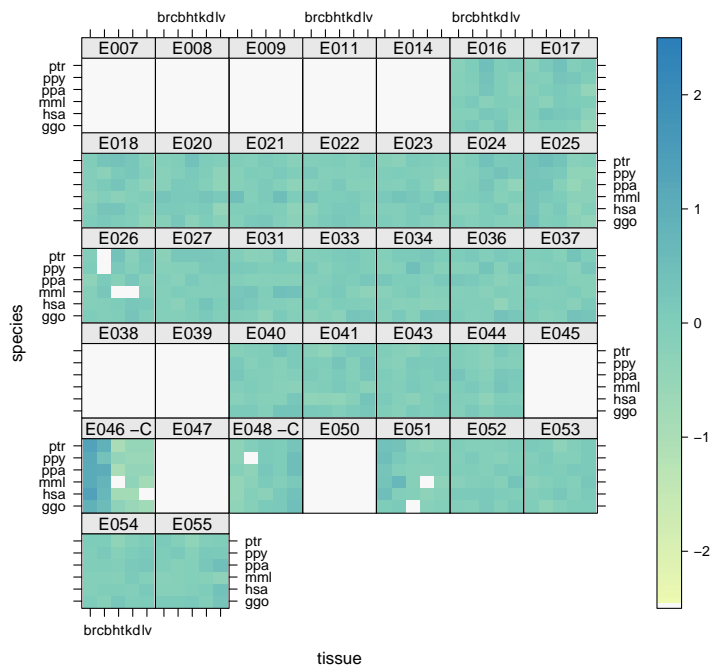
ENSG00000116698



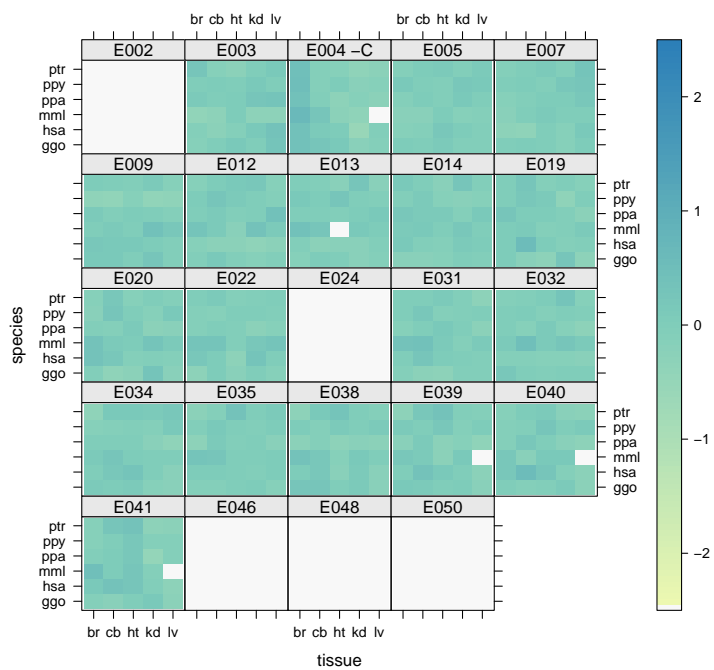
ENSG00000116731



### ENSG0000117114



ENSG00000117139





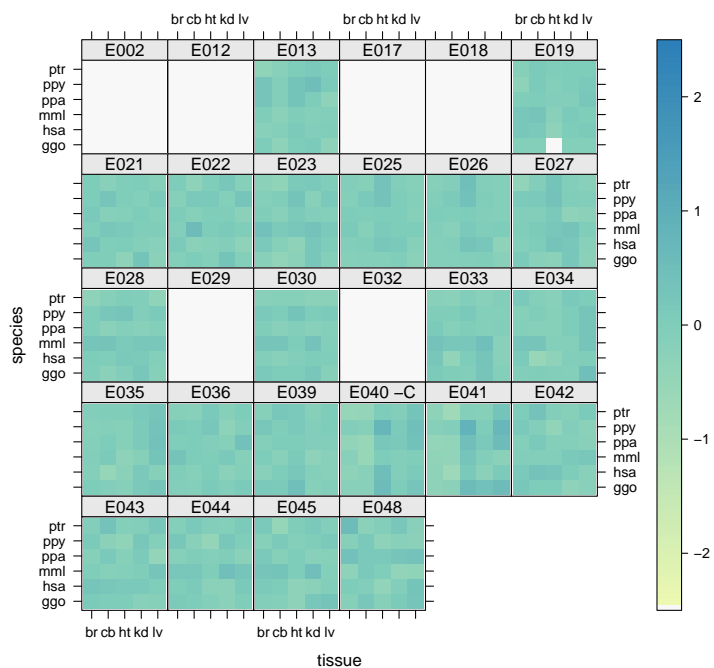




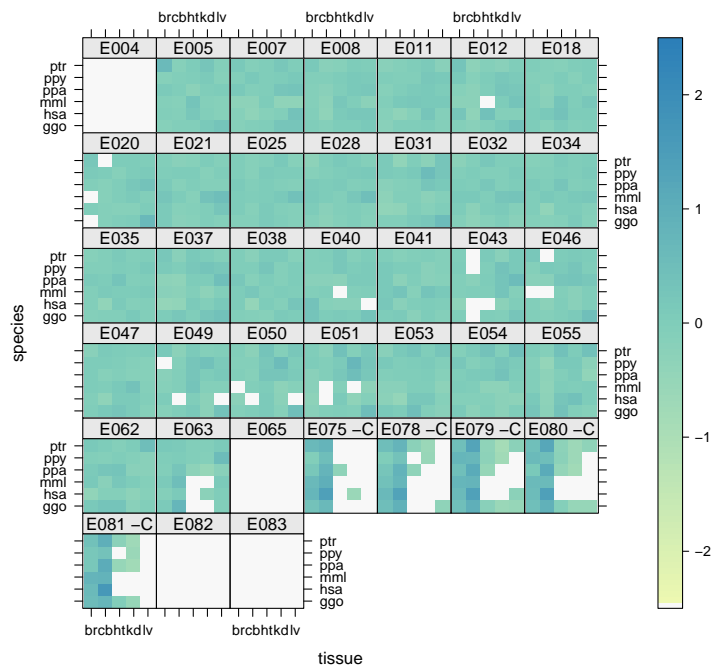




ENSG00000117625



ENSG0000118515

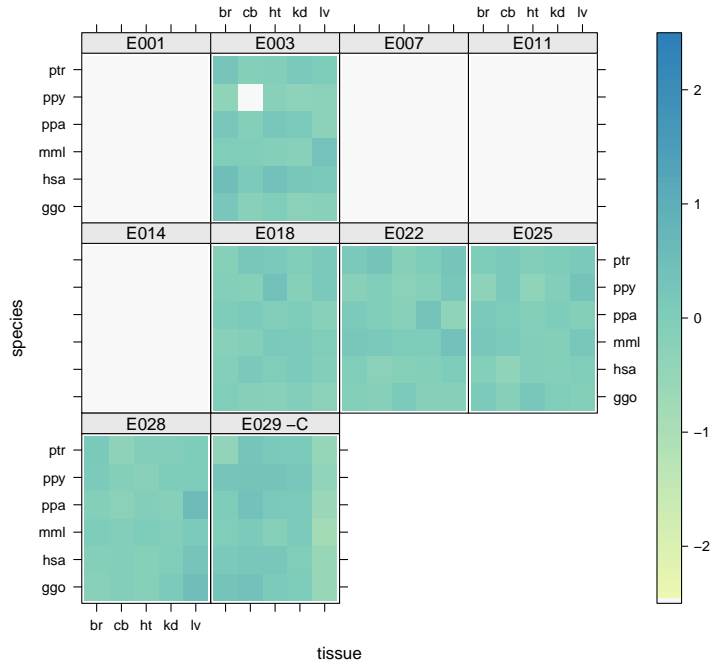




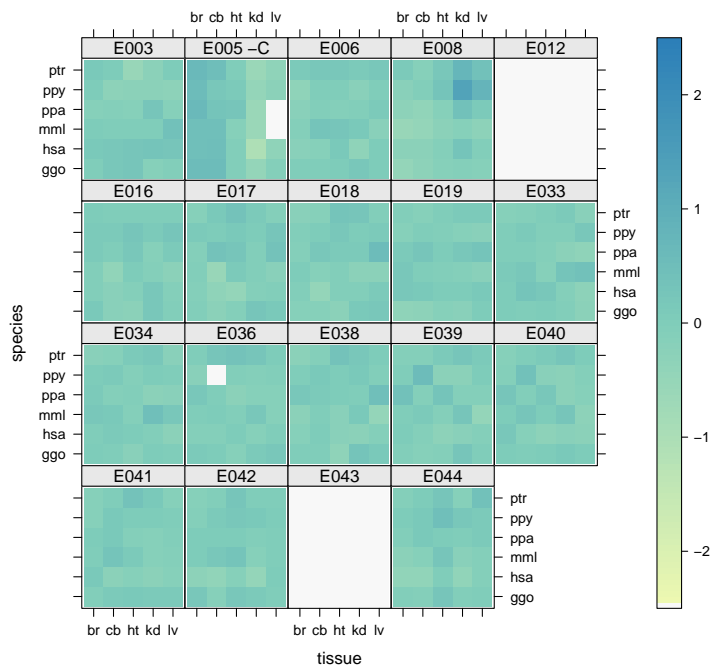




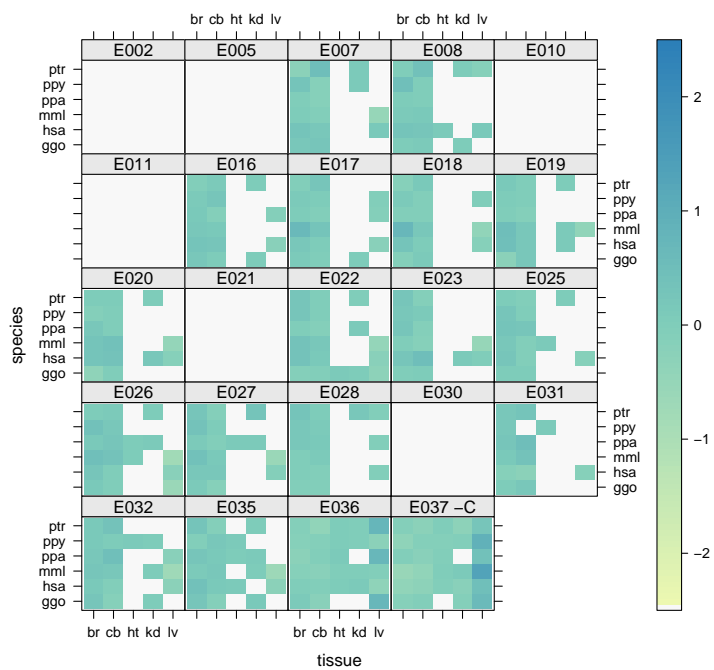
ENSG00000119408



ENSG00000119522



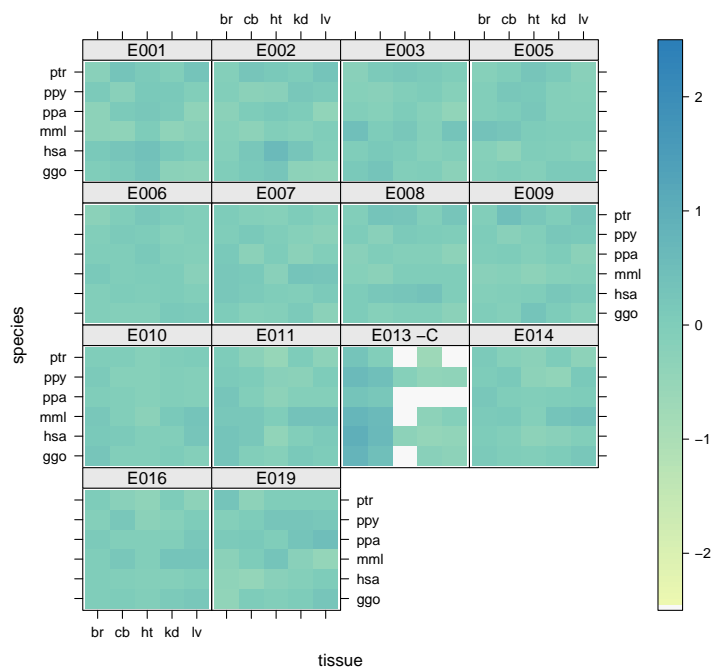
ENSG00000119698



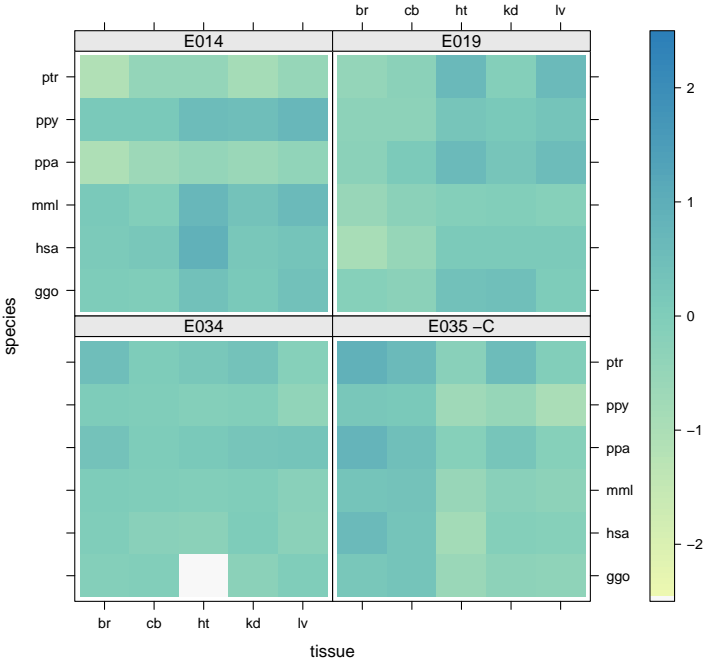




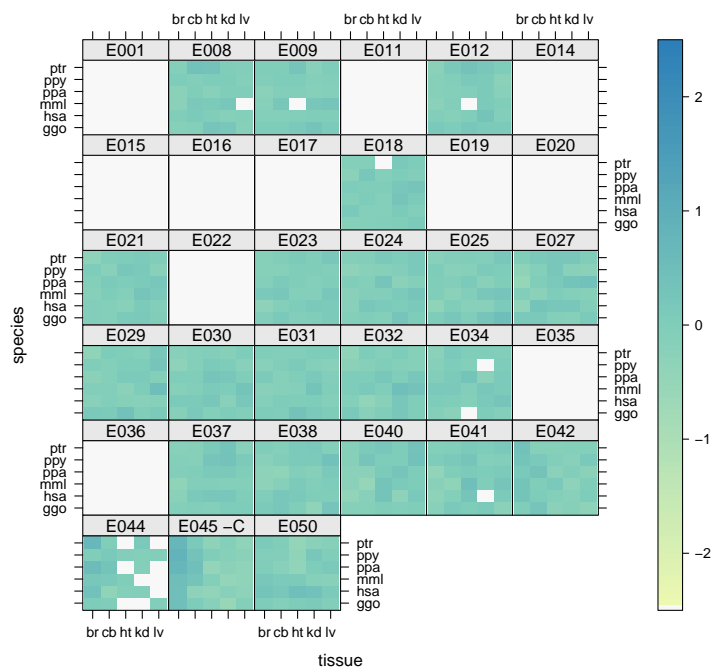
ENSG00000119844



ENSG00000120533



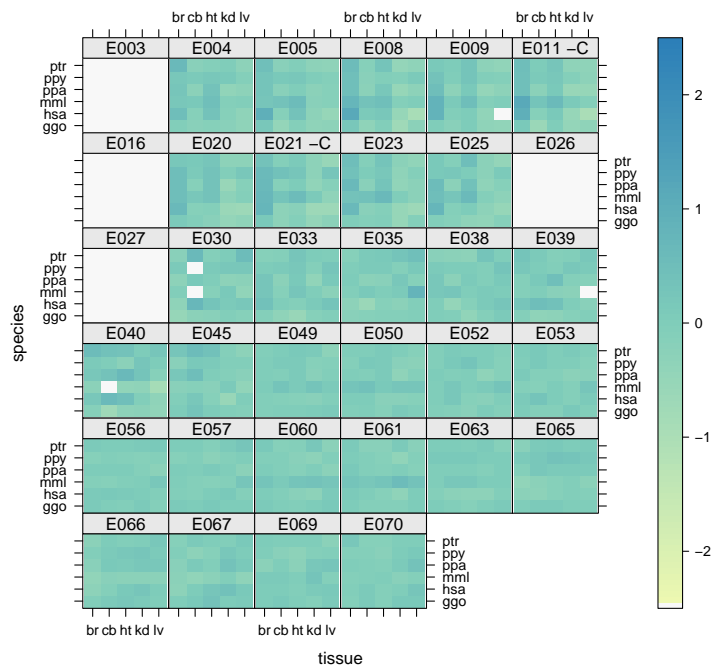
ENSG0000120868



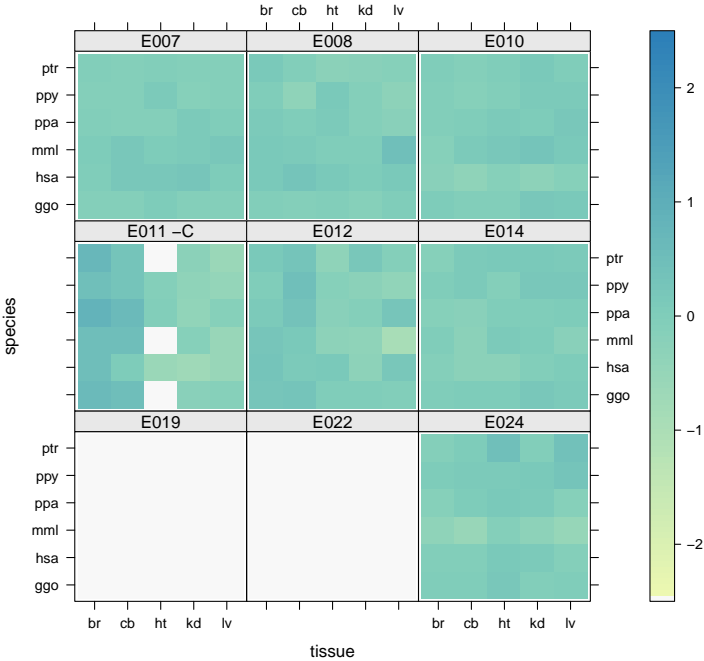




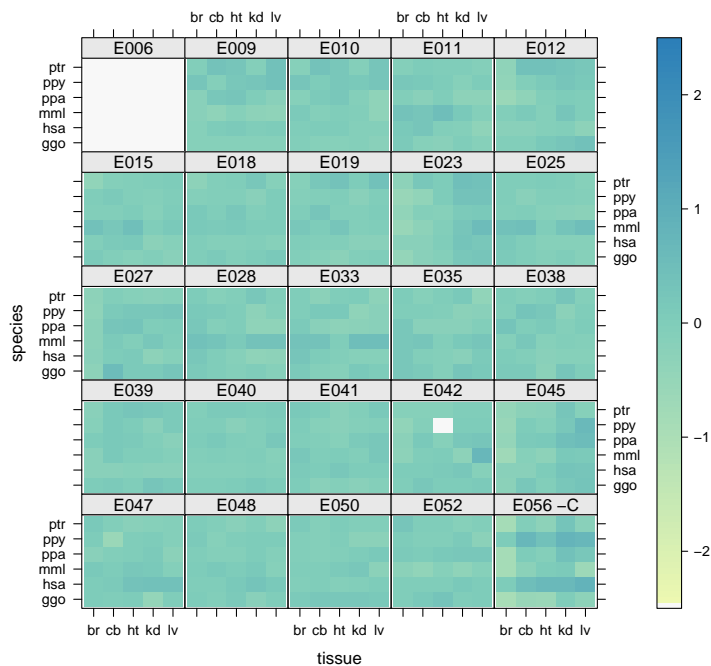
ENSG0000120896



ENSG00000122490

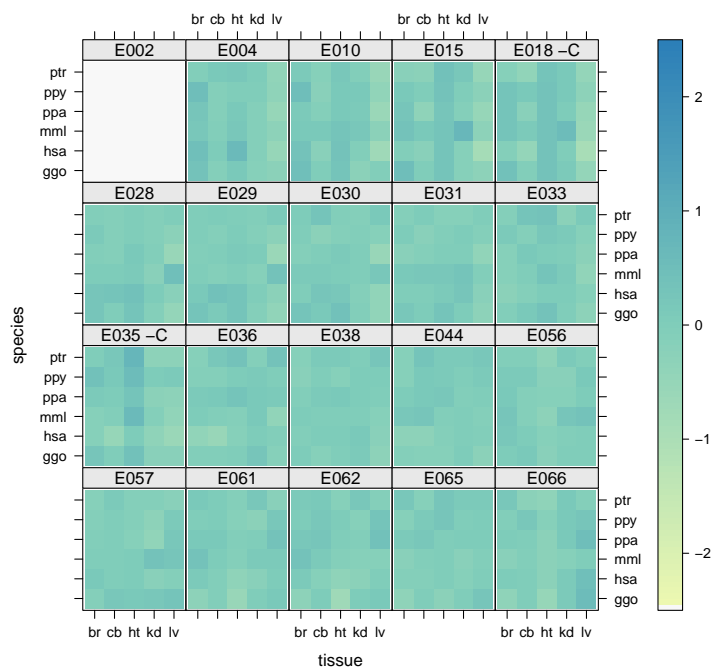


ENSG00000122515

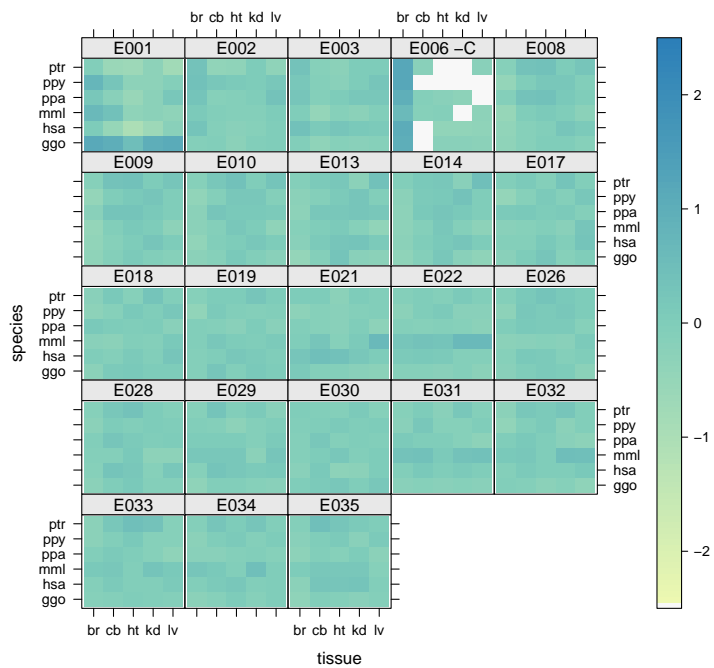




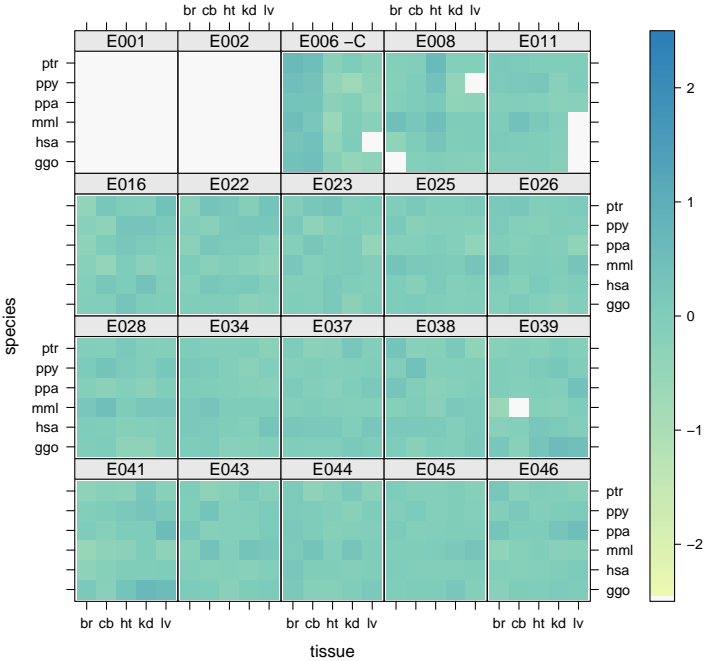
ENSG00000122786



ENSG00000123064

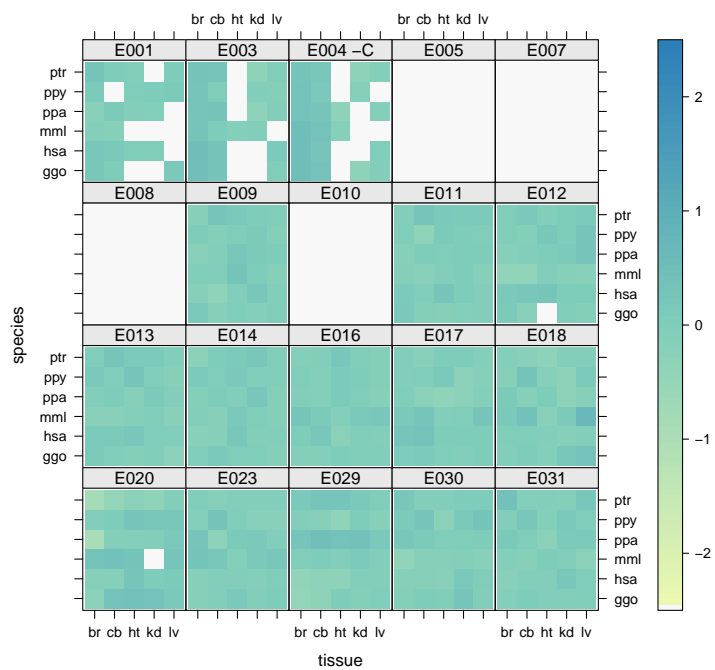


ENSG00000123358

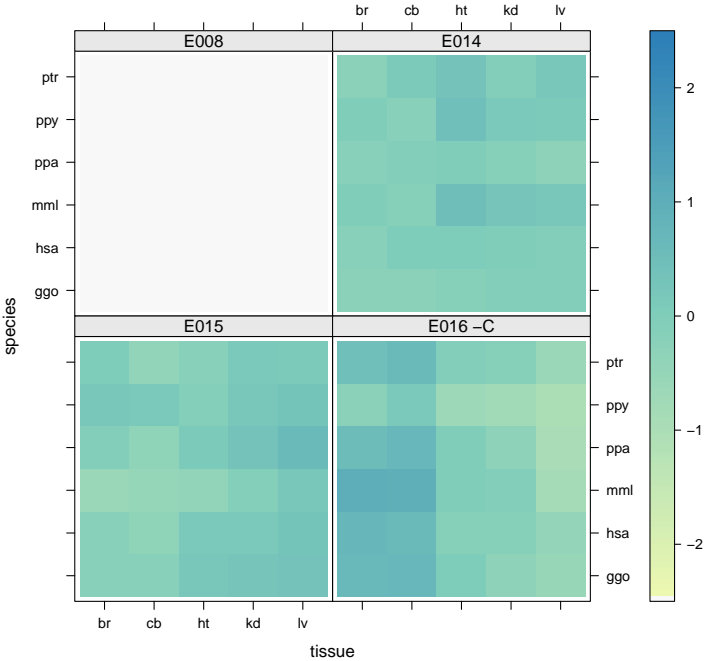




ENSG00000123360



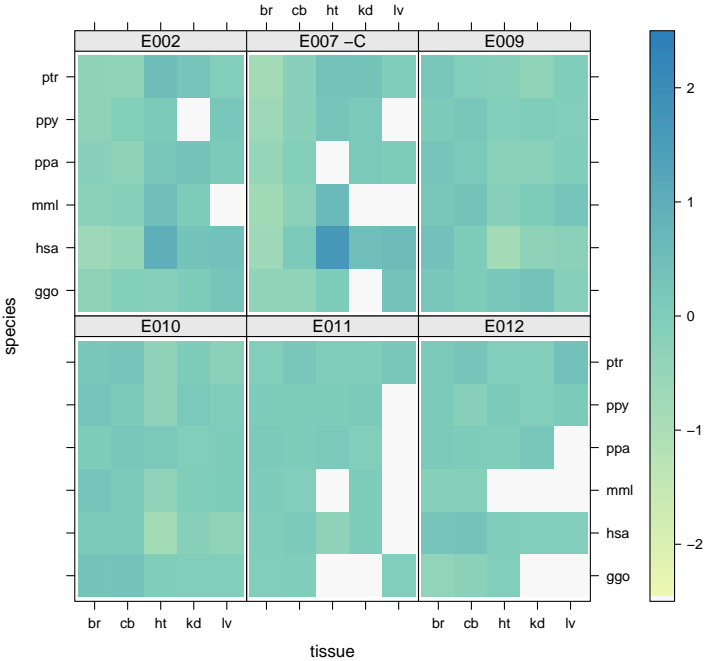
ENSG00000124164



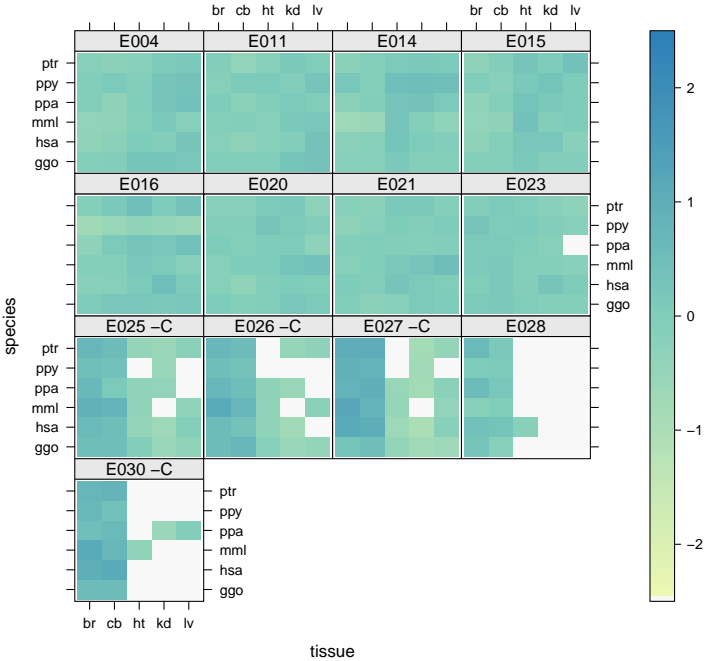




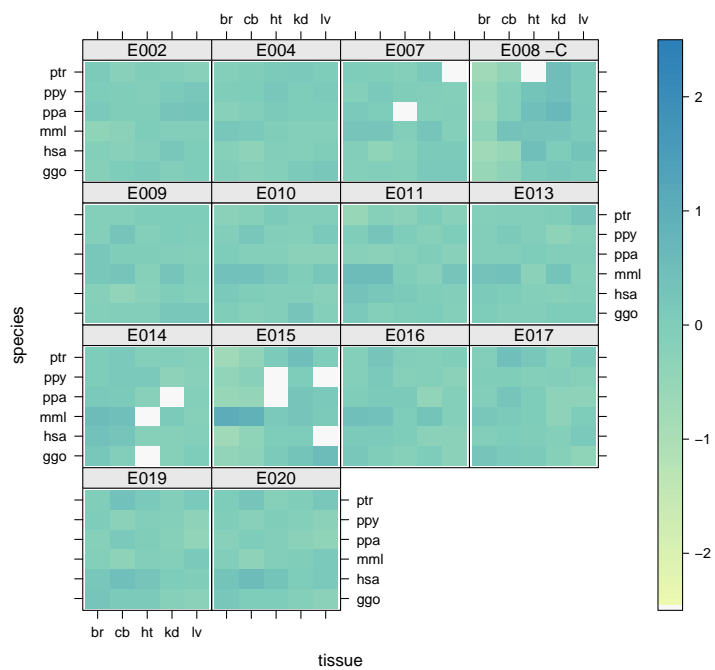
ENSG00000125744



ENSG00000125746



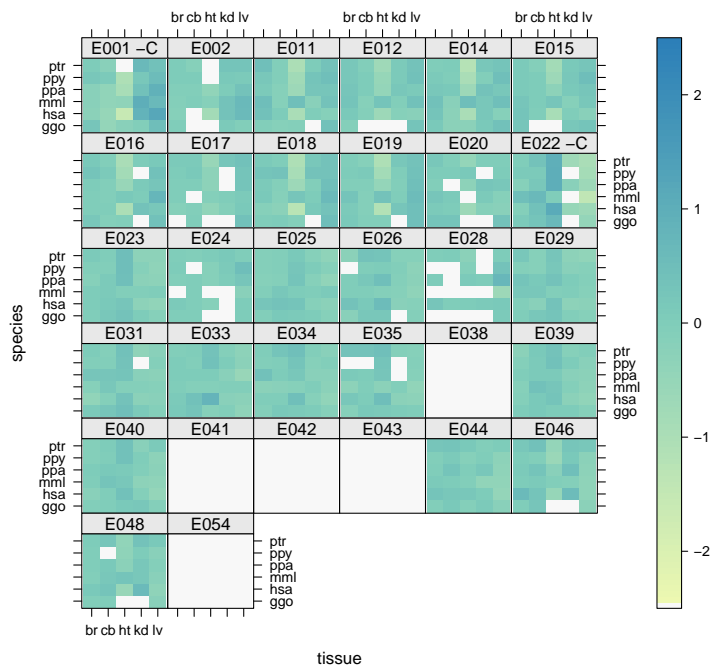
ENSG00000125814



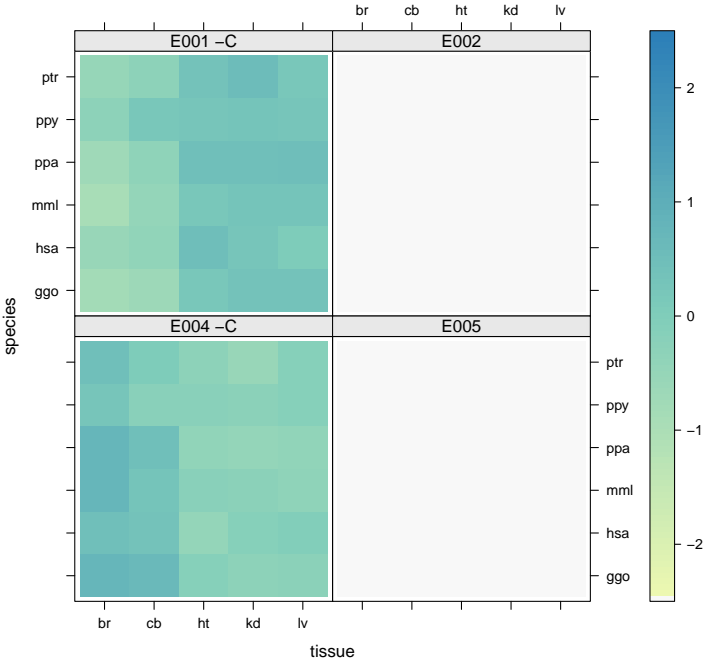




ENSG00000127241



ENSG00000127423

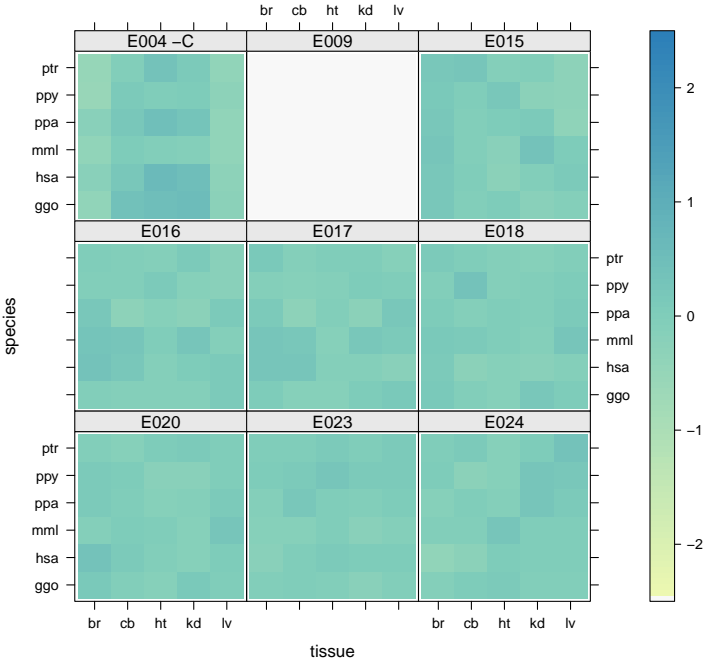






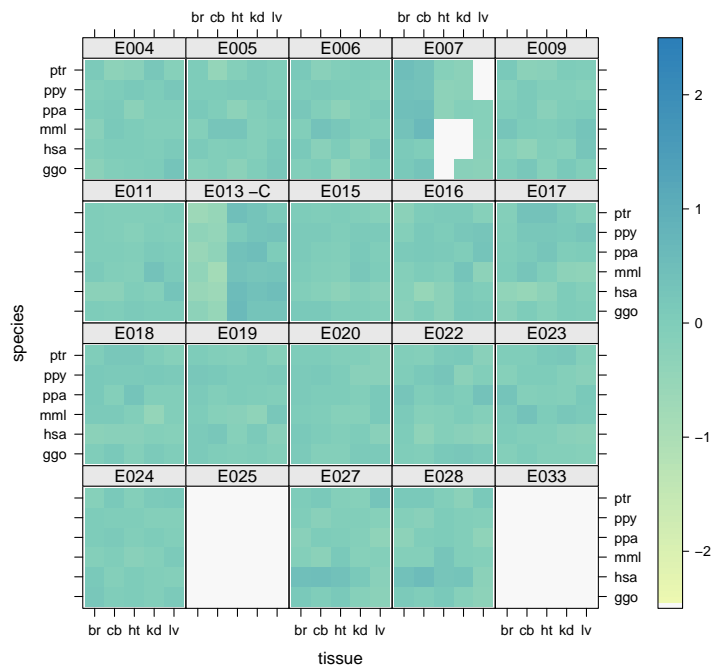


ENSG00000127838



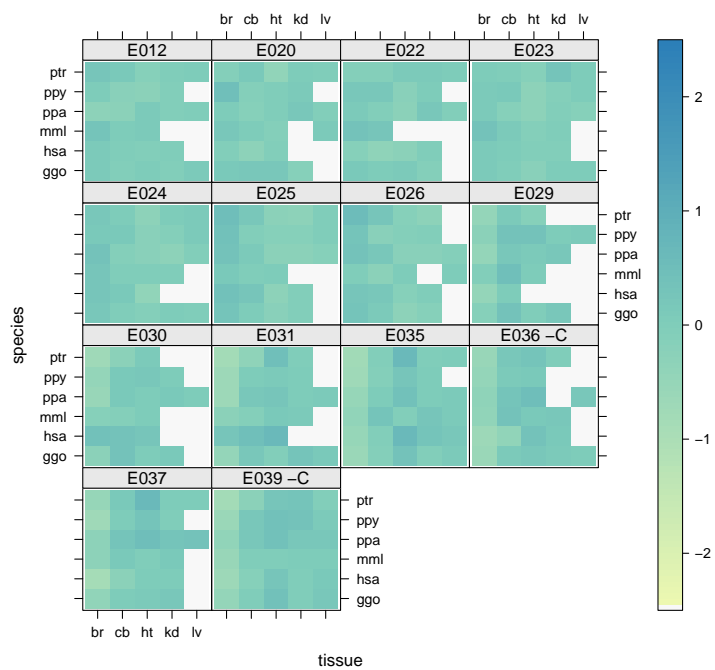


ENSG00000127990

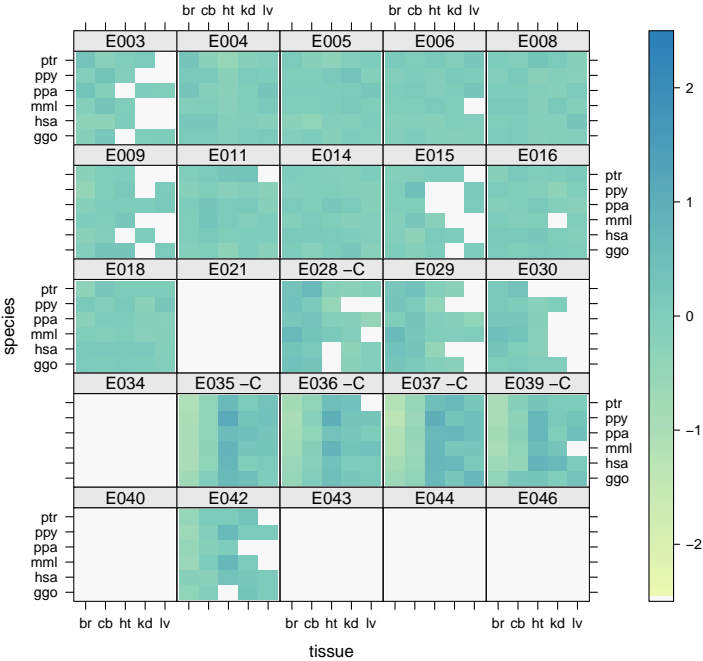




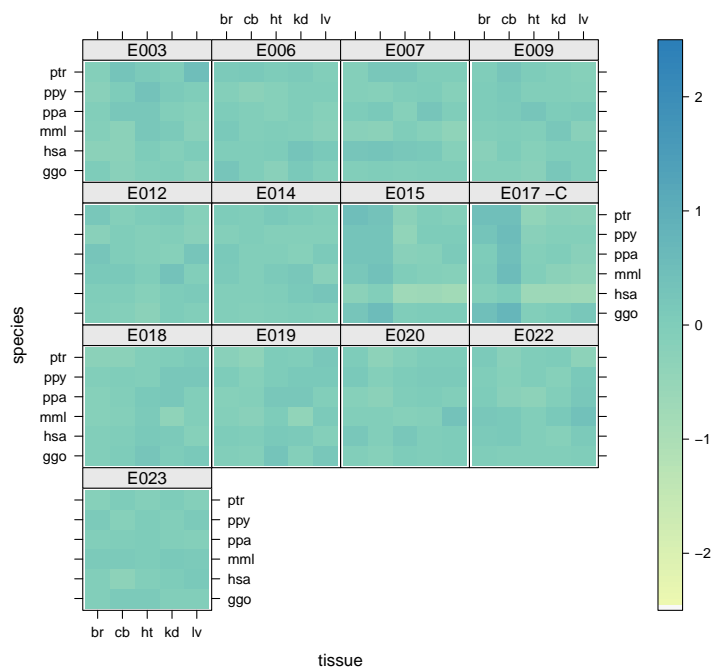
ENSG00000128596



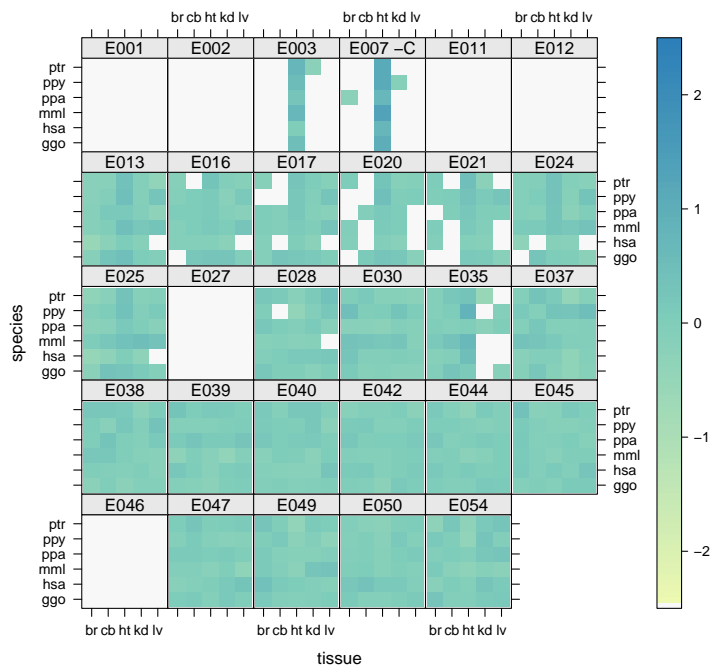
ENSG00000128656



ENSG00000128928



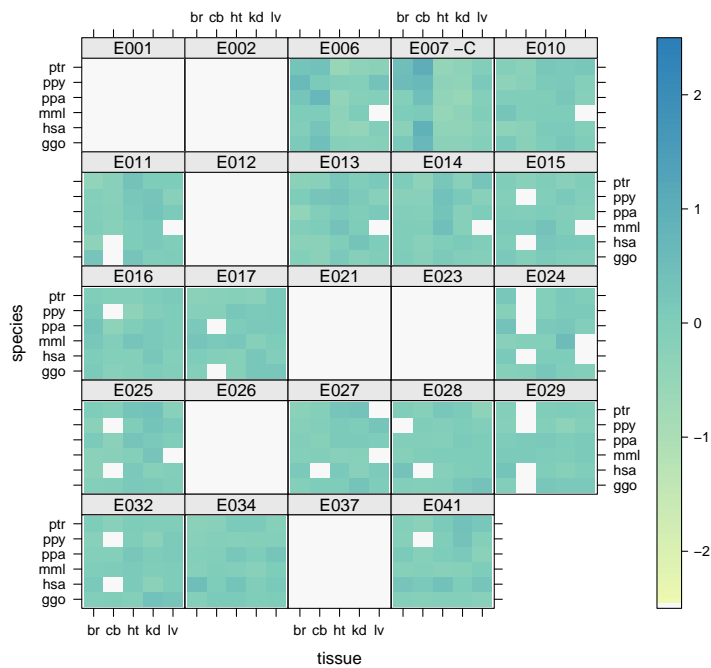
ENSG0000129116



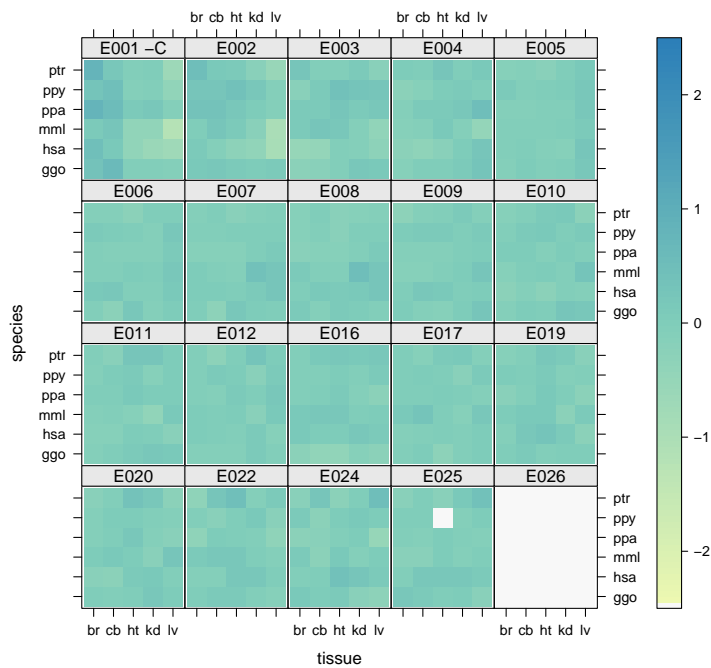




ENSG00000129595

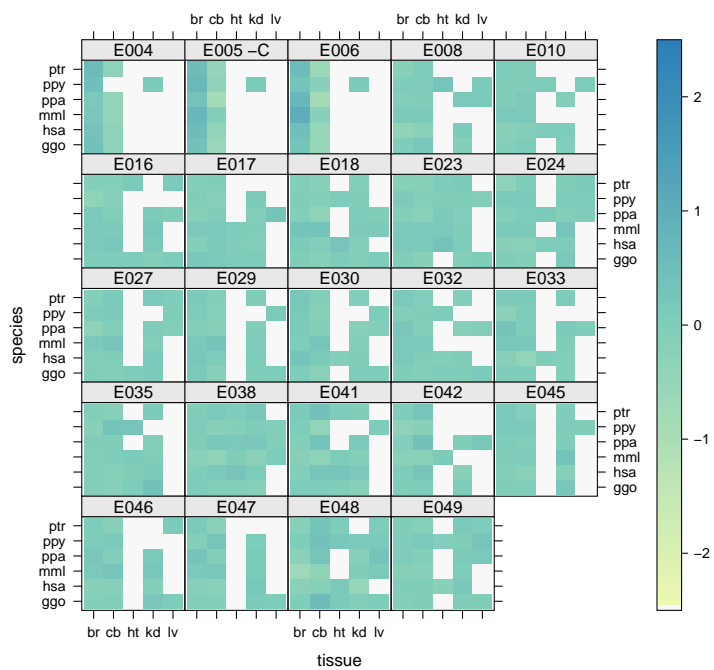


ENSG00000129636



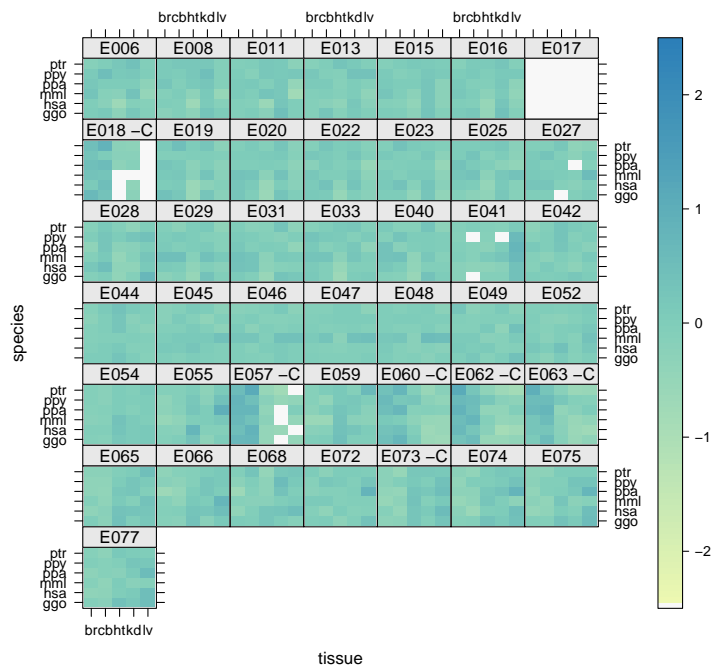


ENSG00000130226

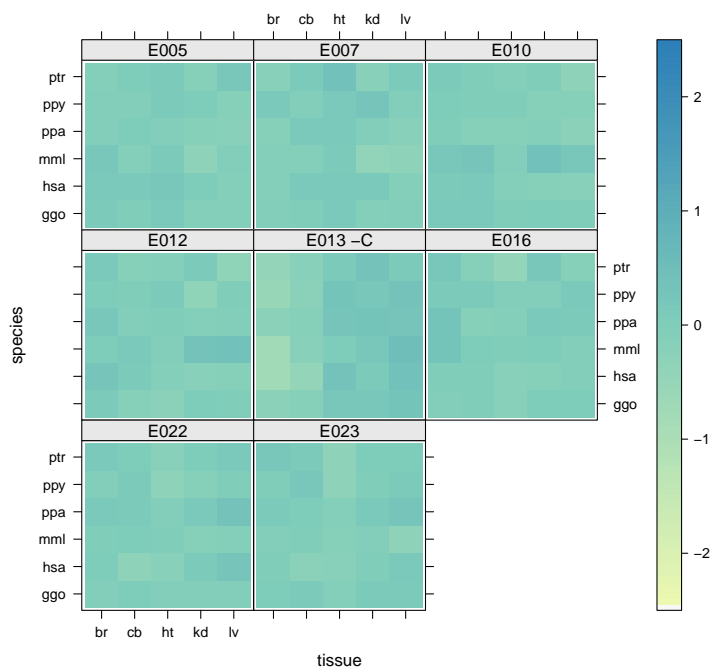




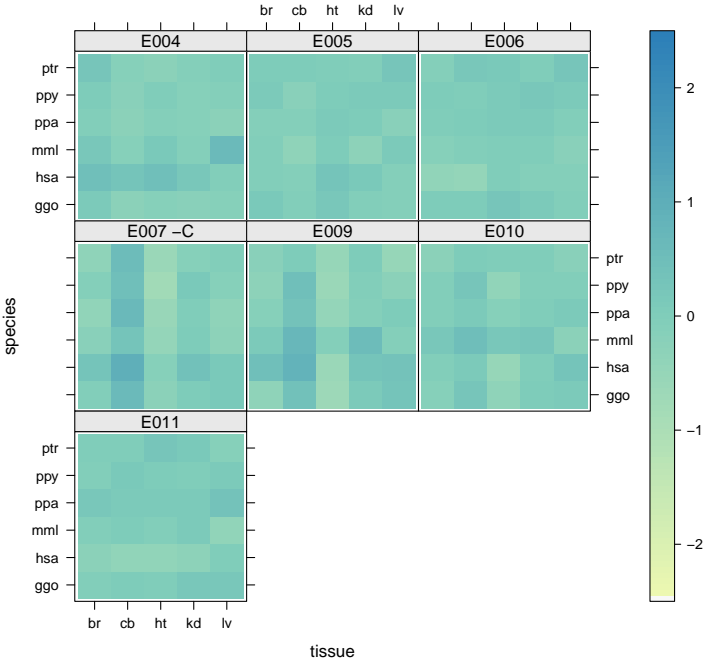
ENSG0000130396



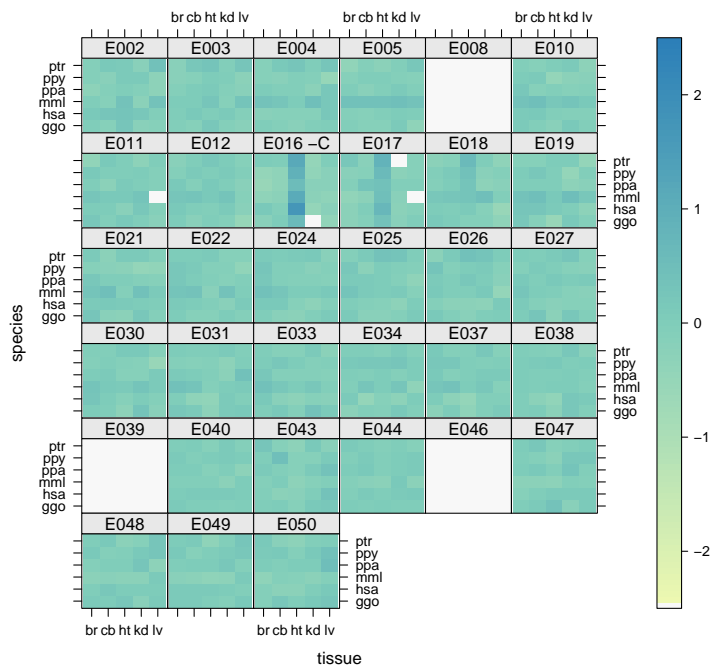
ENSG00000130402



ENSG00000130770

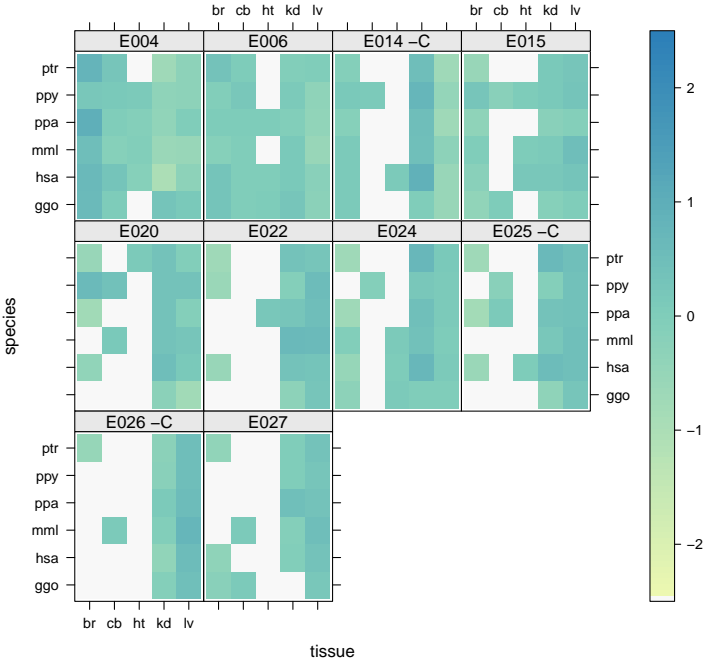


ENSG0000130939



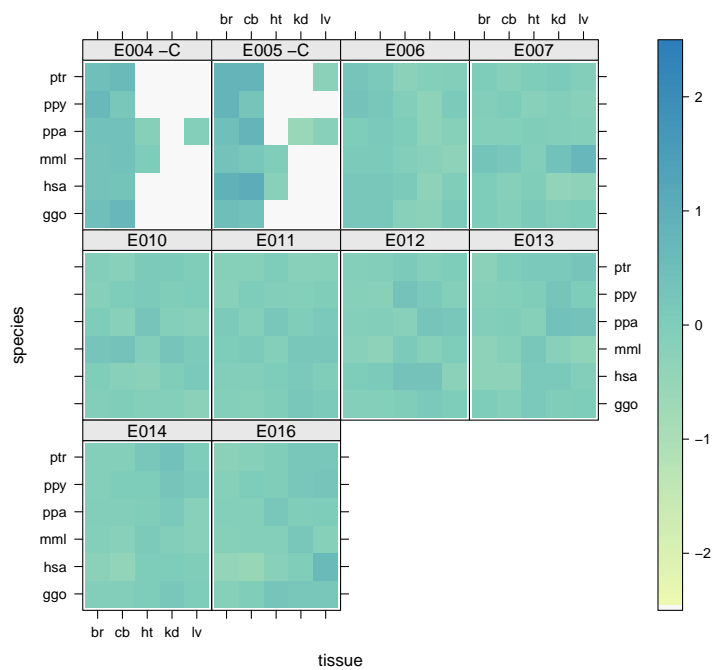


ENSG00000131187



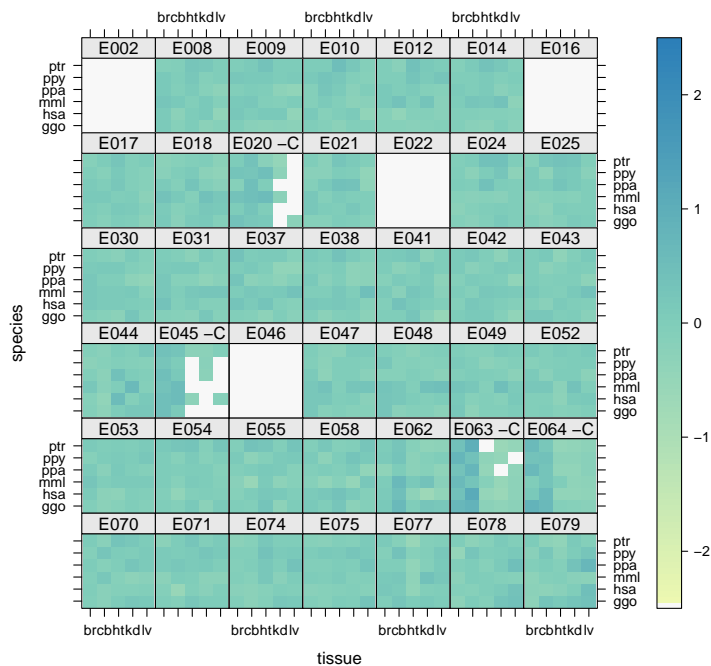


ENSG00000131242

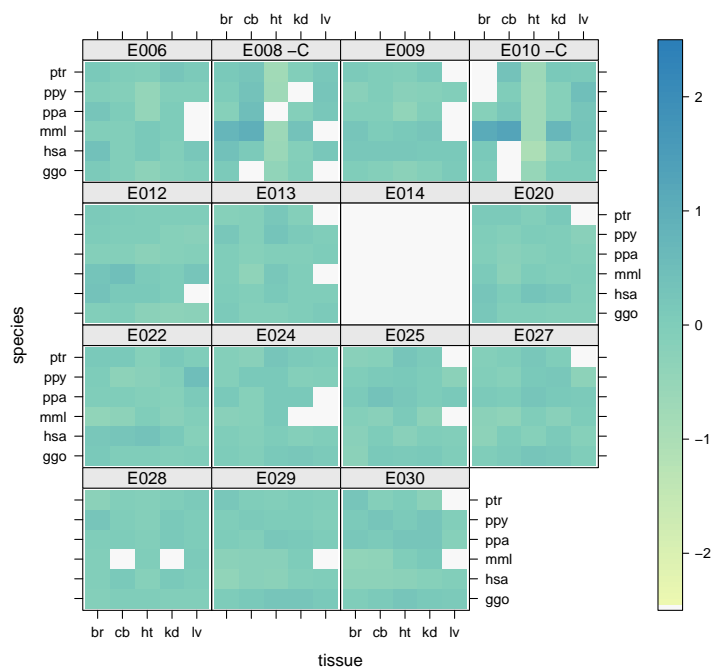




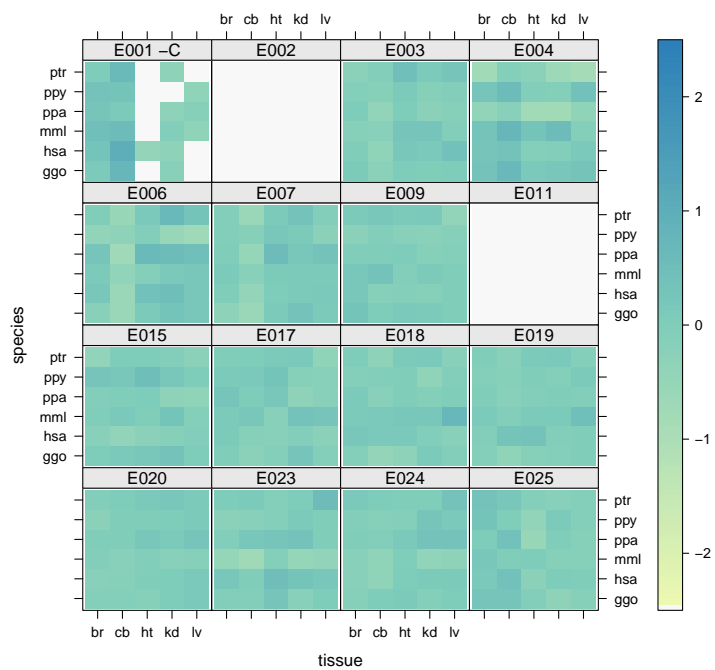
ENSG0000131626



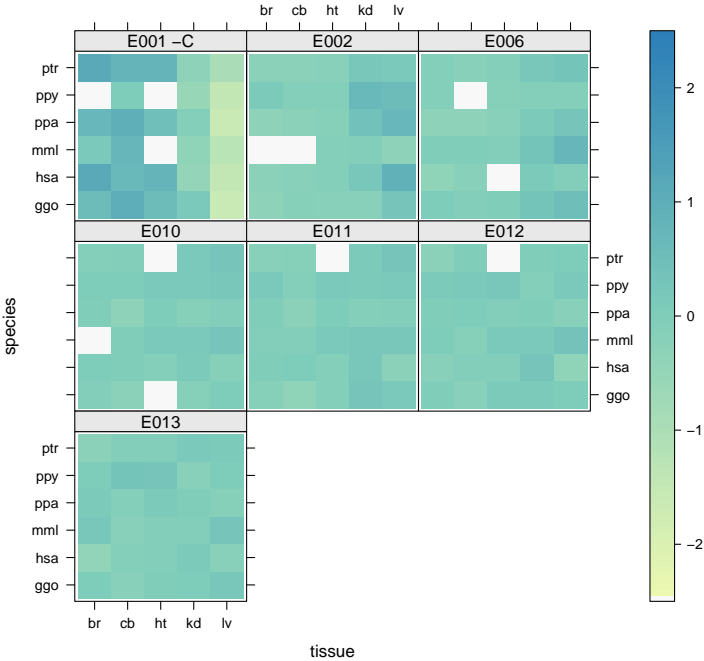
ENSG00000131730



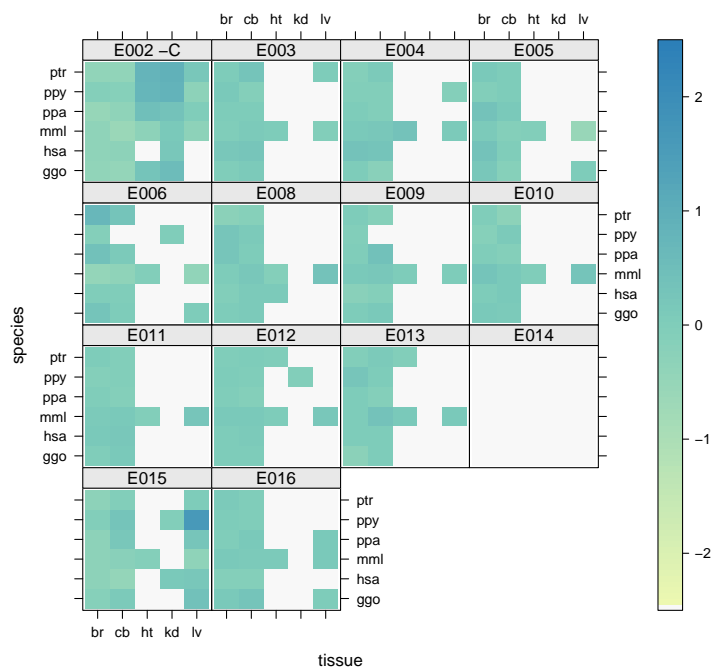
ENSG00000131759



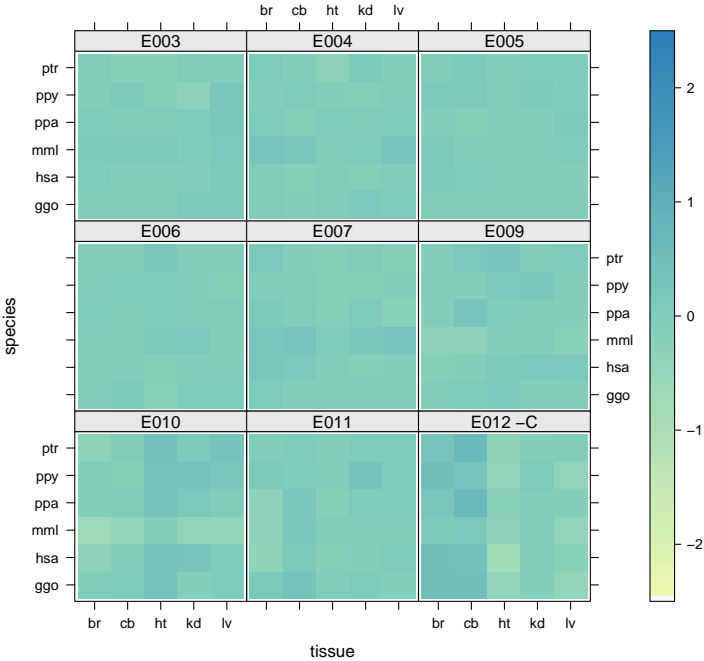
ENSG00000131781



ENSG00000132164

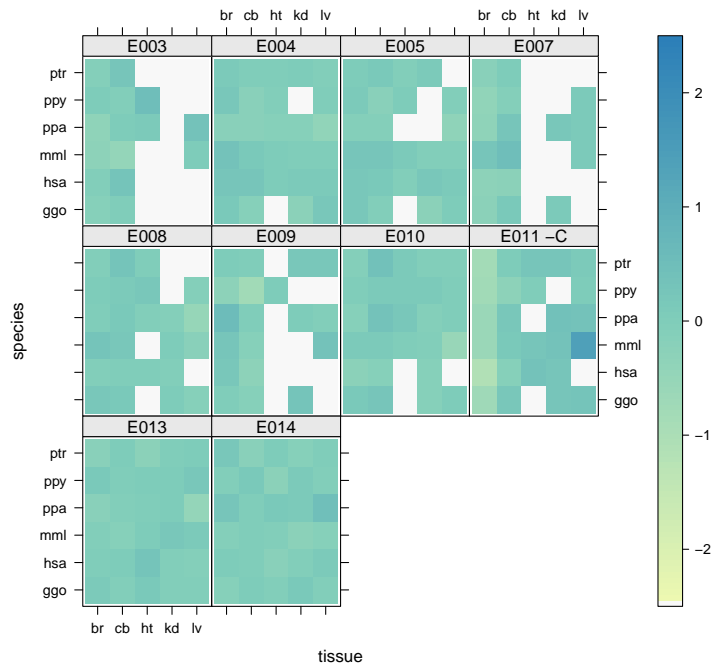


ENSG00000132589

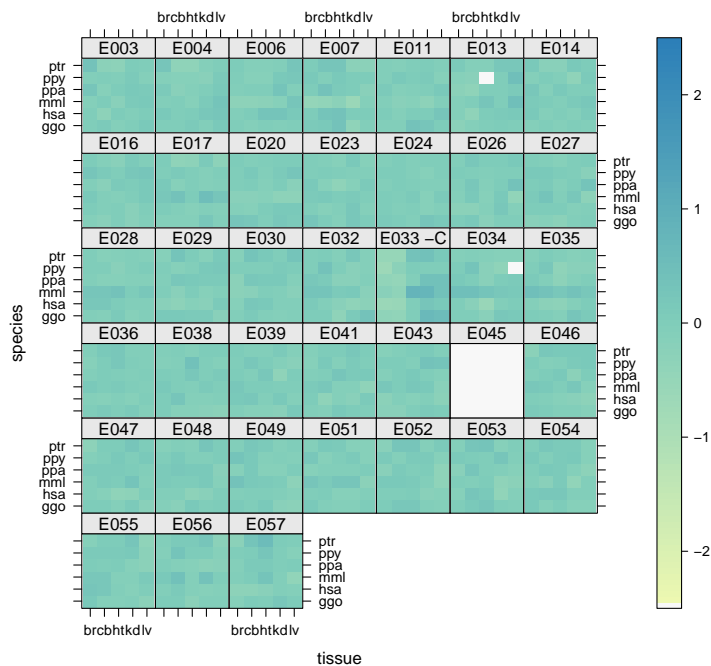




ENSG00000132639



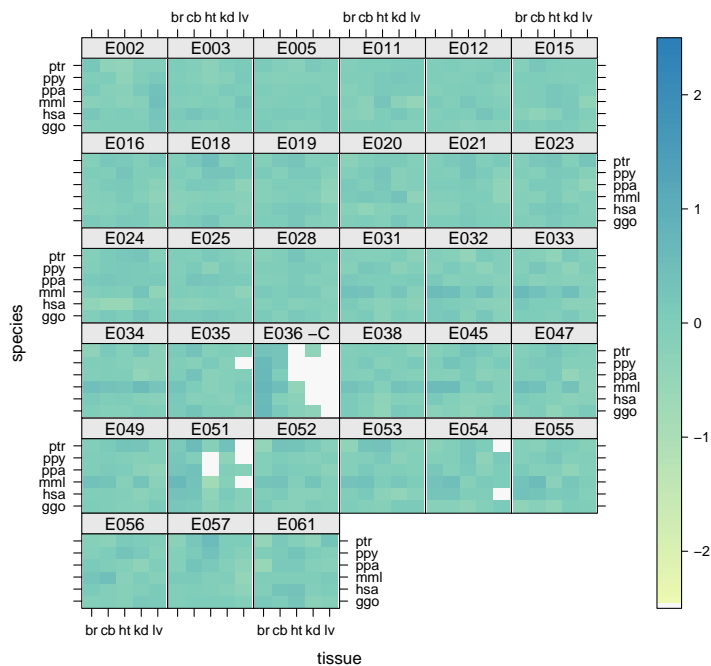
ENSG0000132694



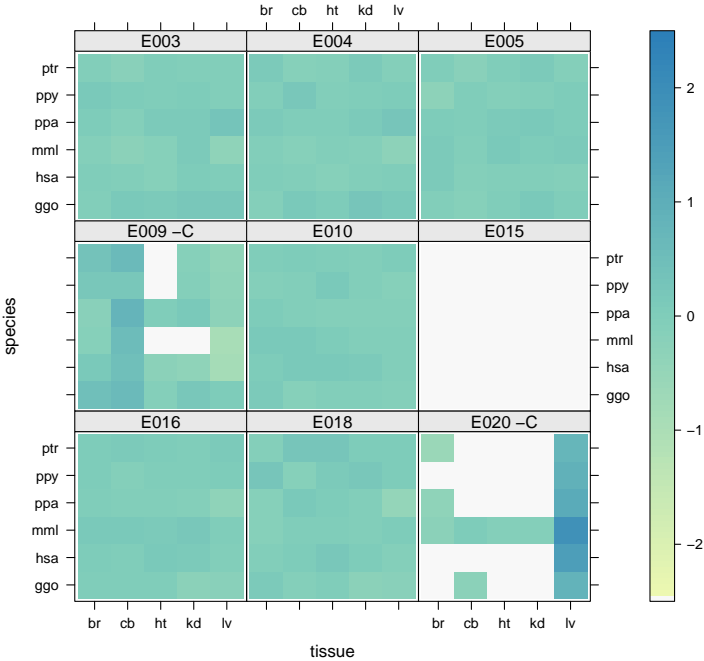




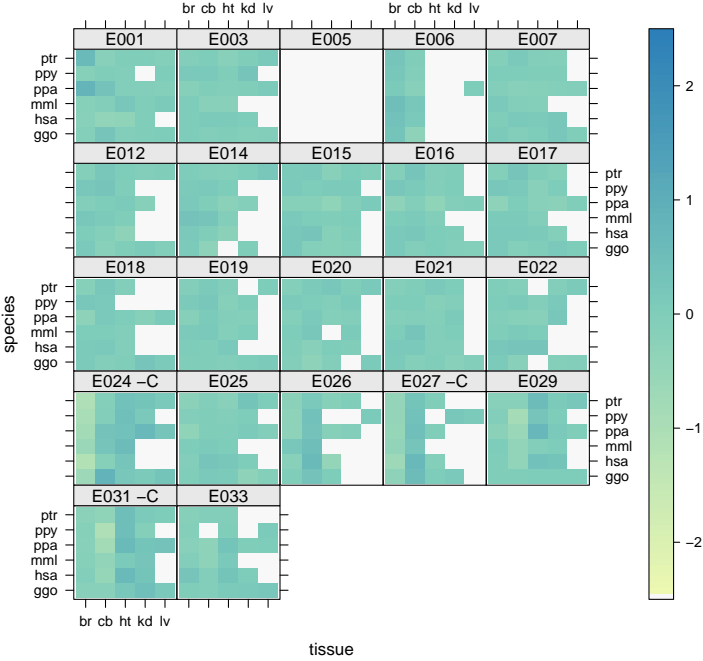
ENSG0000133026



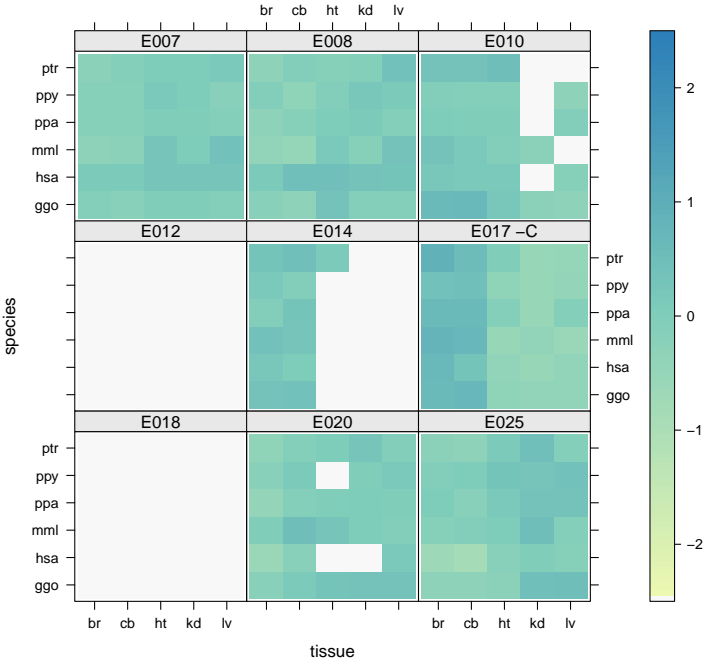
ENSG00000133027



ENSG00000133083

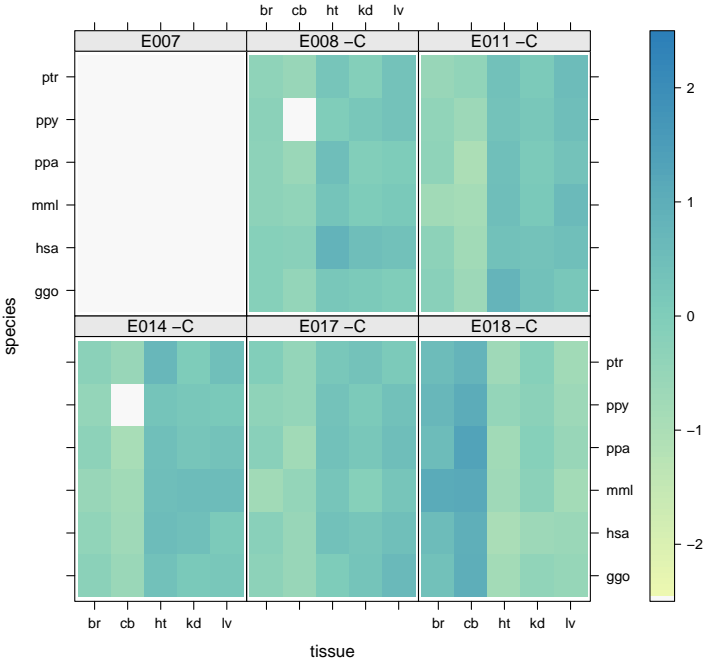


ENSG00000133318

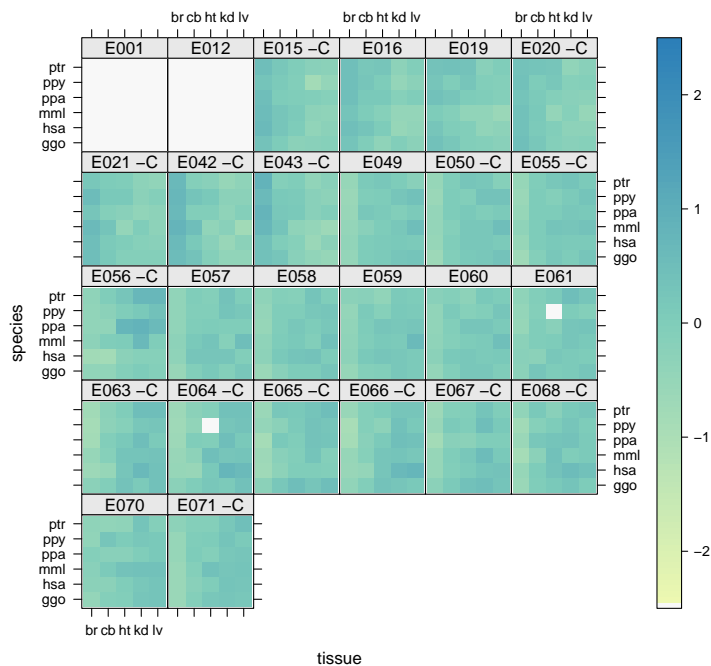




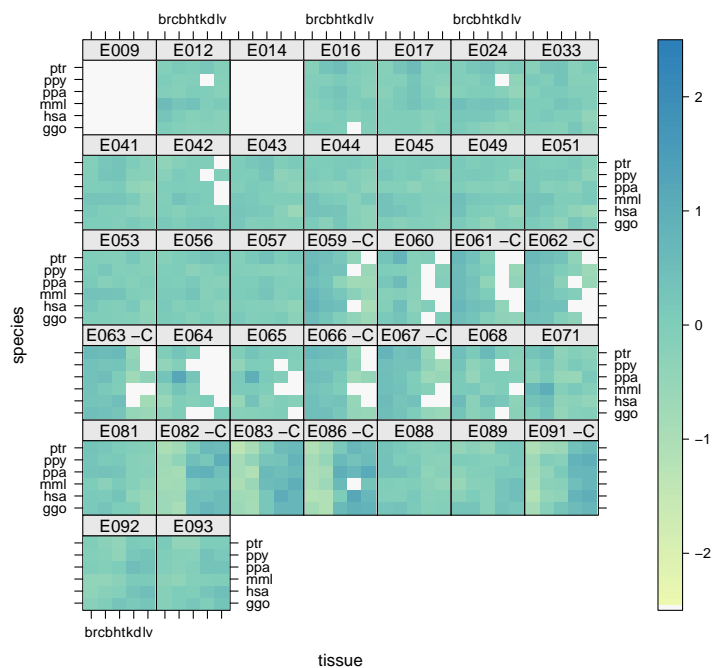
ENSG00000133574



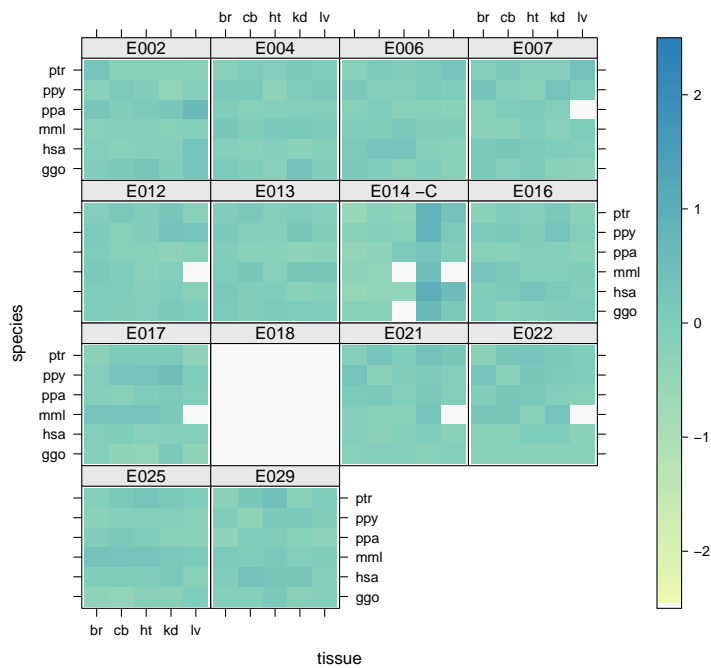
ENSG0000133612



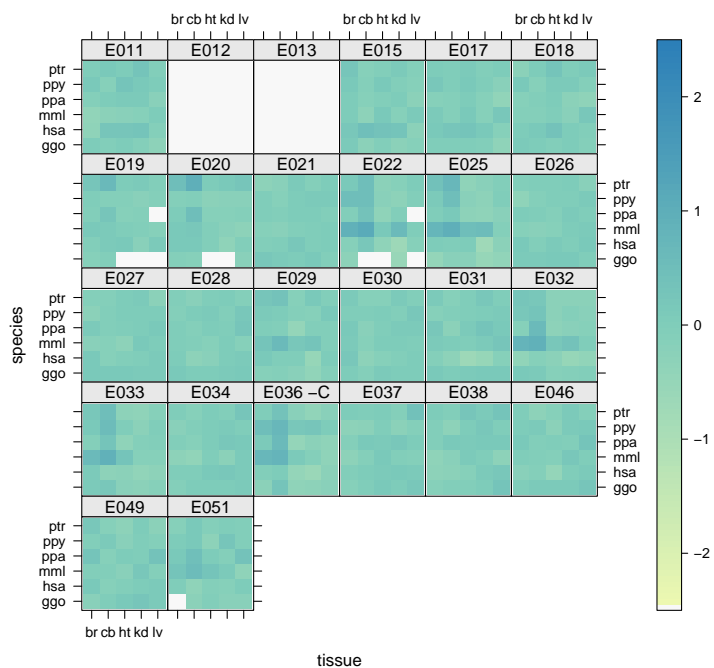
ENSG0000133816



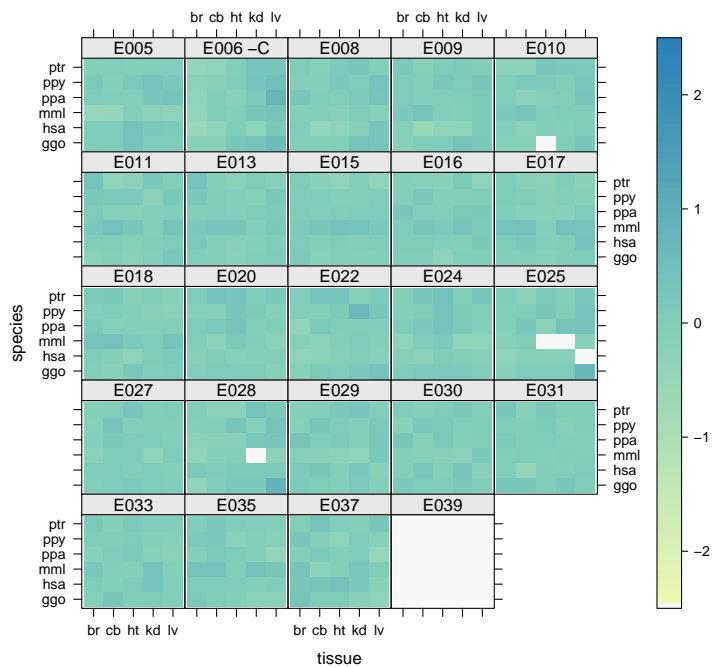
ENSG00000134278



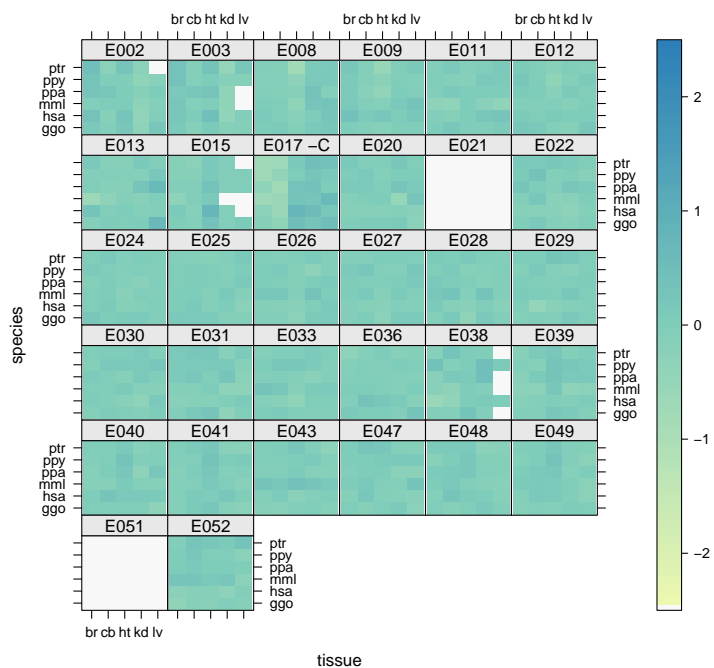
**ENSG00000134291**



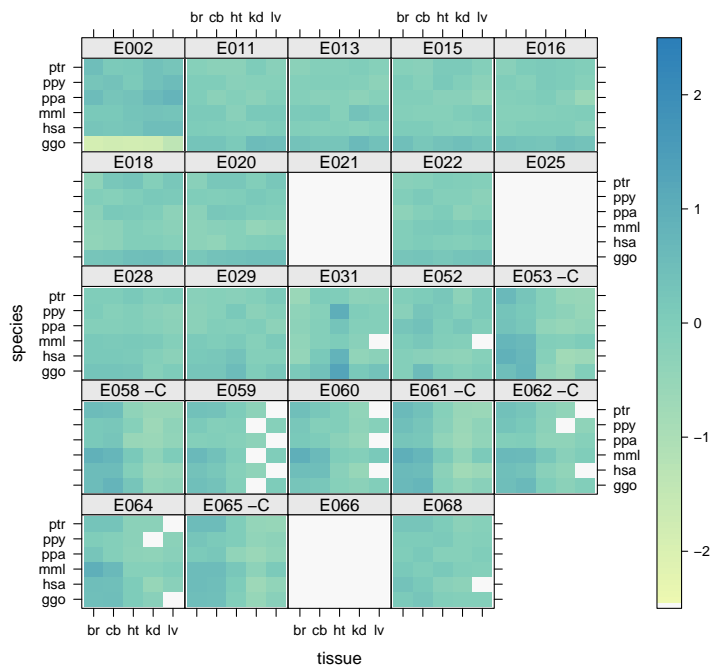
ENSG00000134294



ENSG0000134313



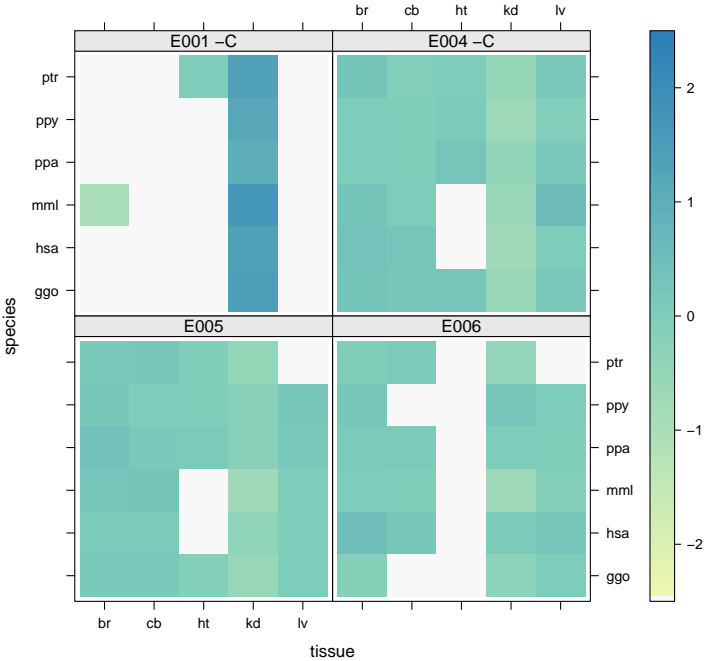
ENSG00000134686





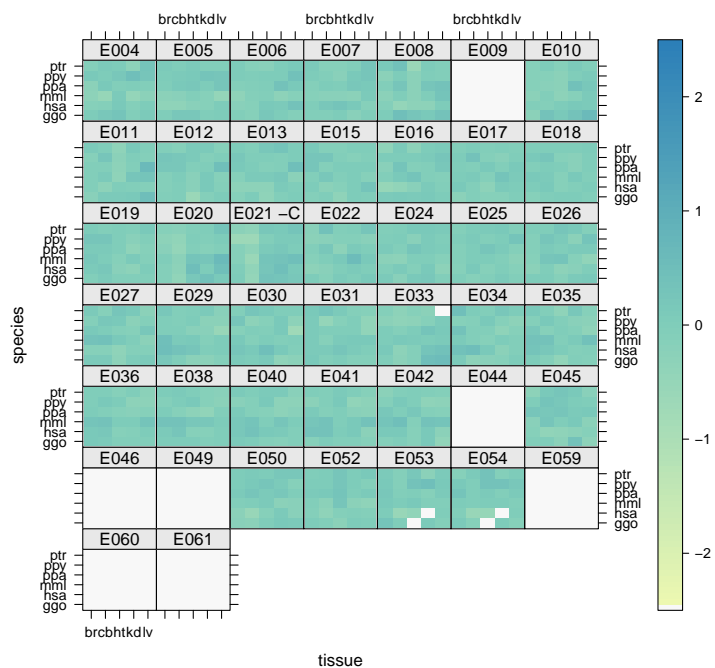


ENSG00000134873

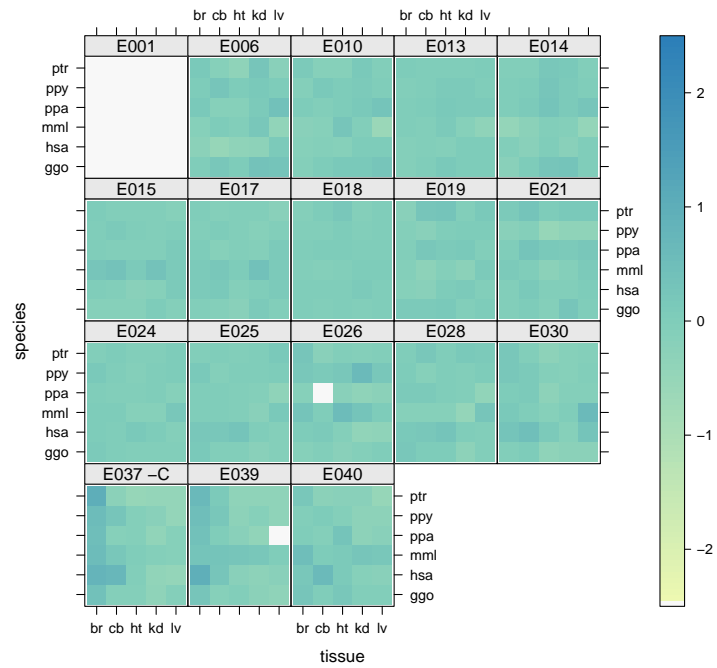




ENSG0000135365



**ENSG00000135404**





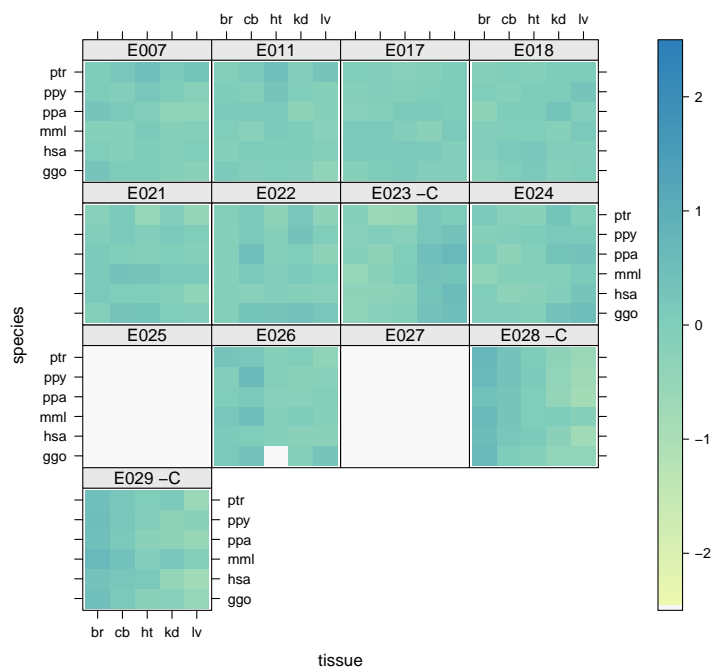








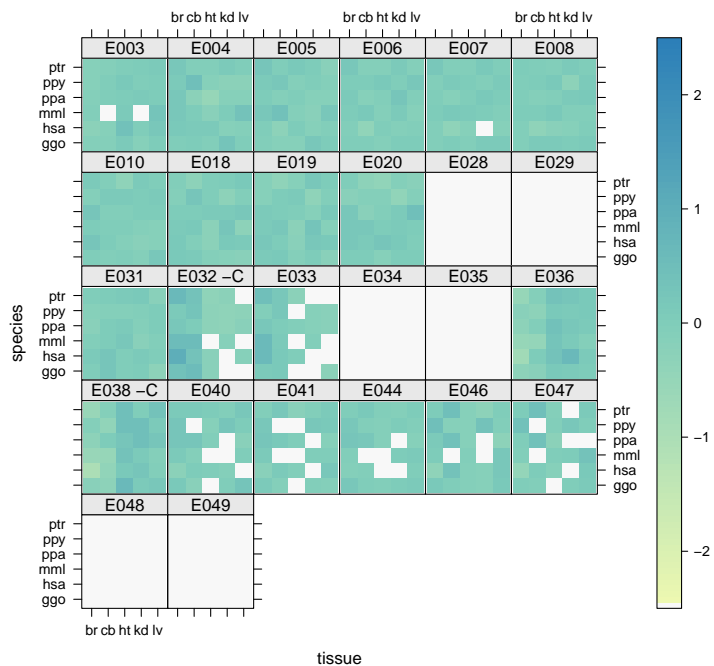
ENSG00000135930





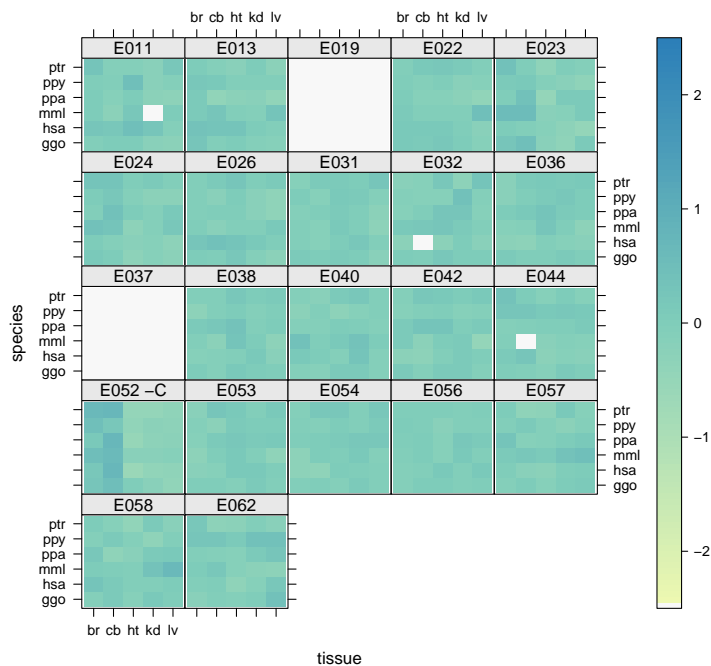


ENSG0000136237





ENSG00000136436







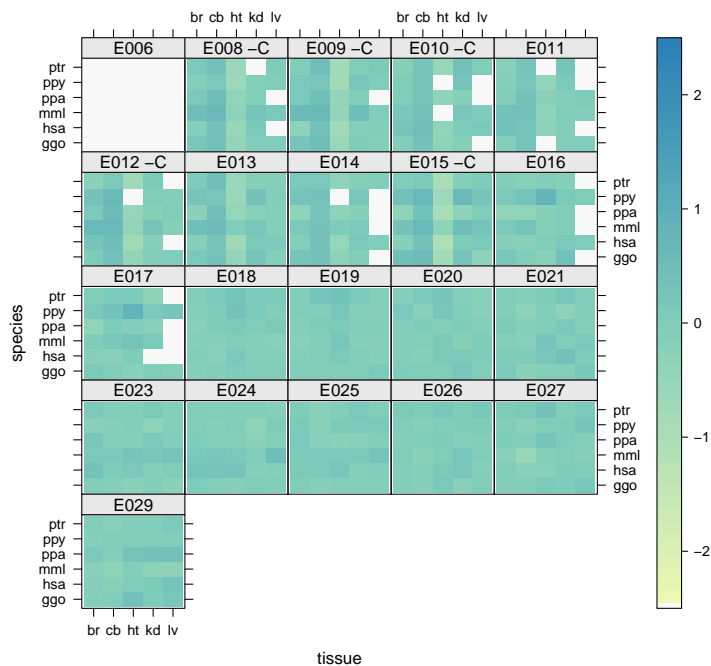




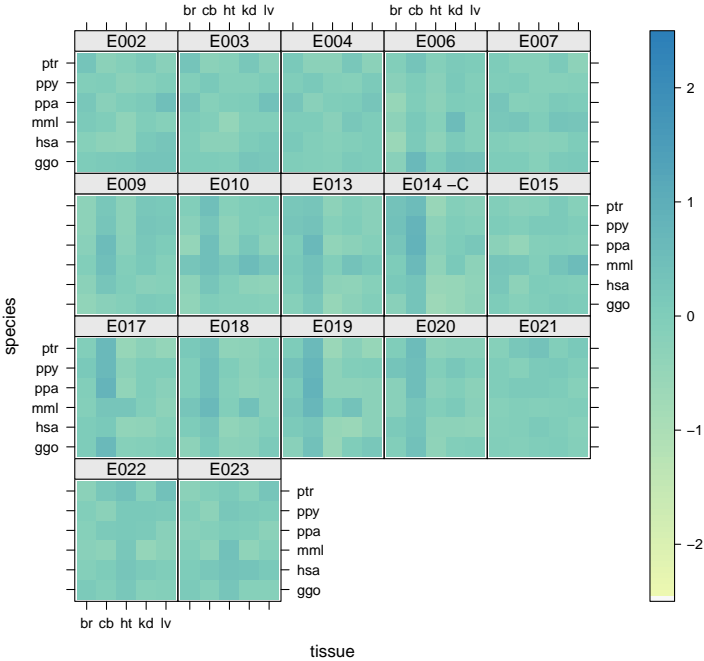




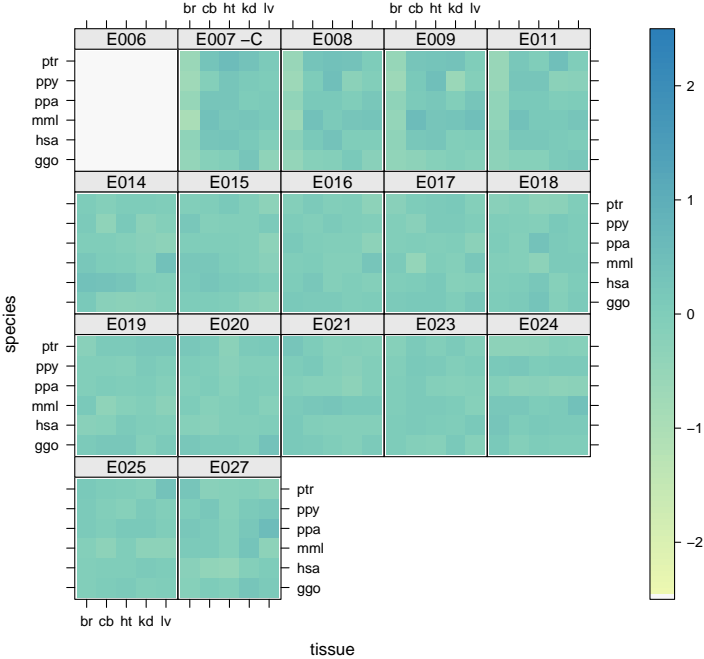
ENSG00000137094



ENSG00000137100



ENSG00000137166

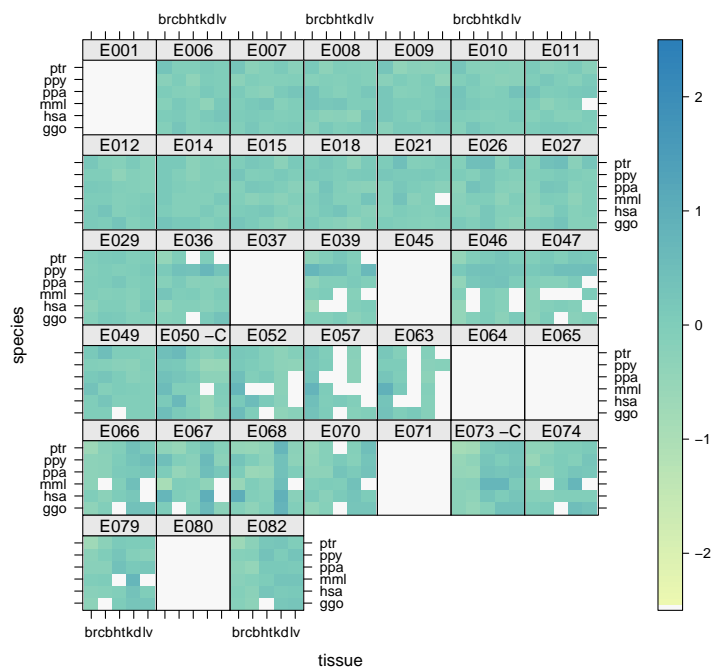






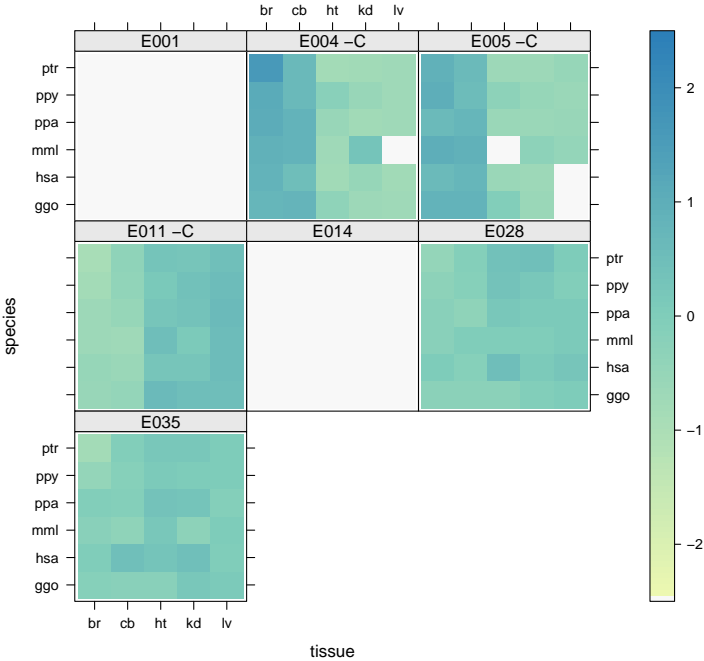


ENSG0000137501

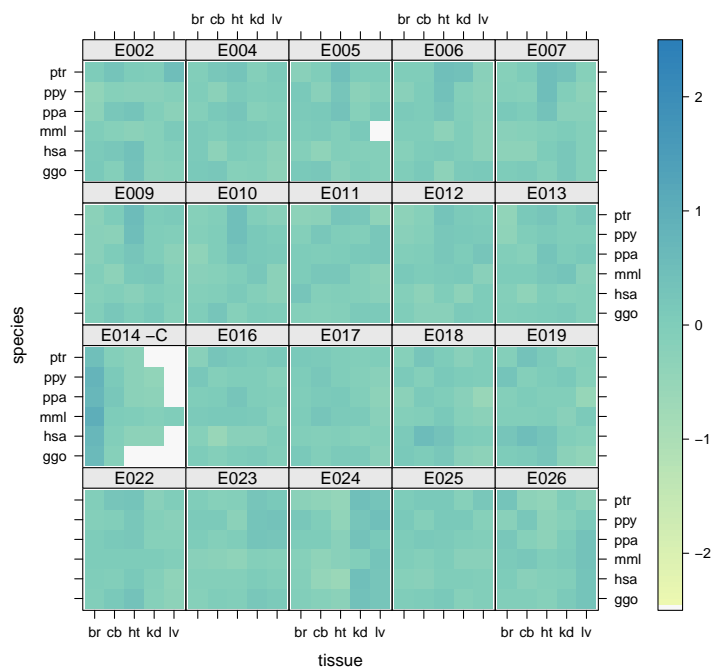




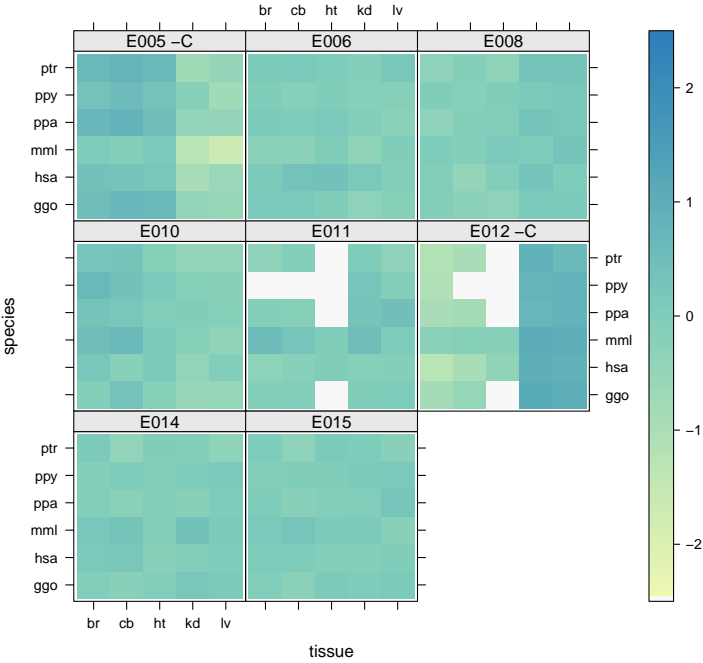
ENSG00000137710



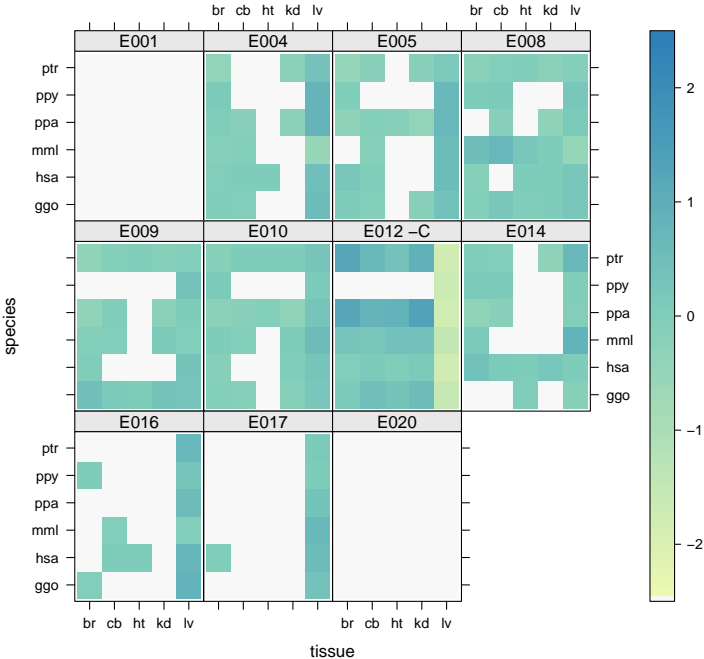
ENSG00000137942



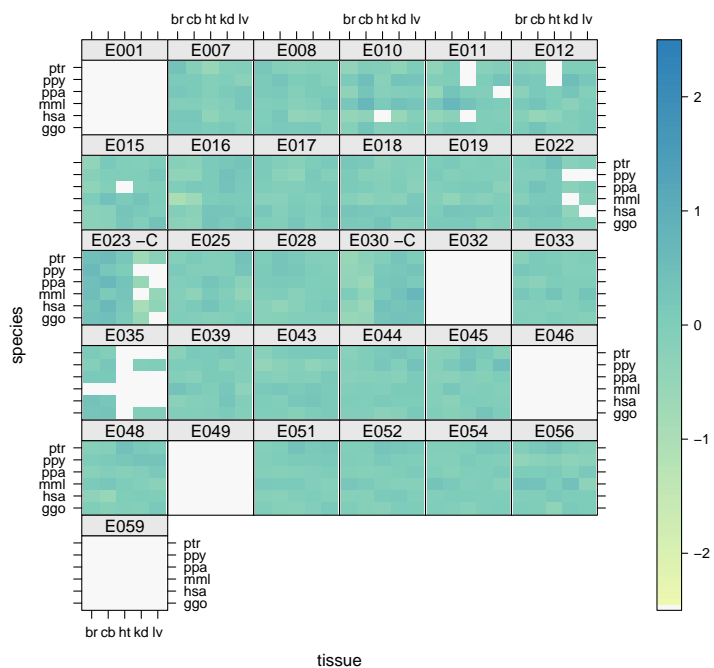
ENSG00000138030



ENSG00000138075



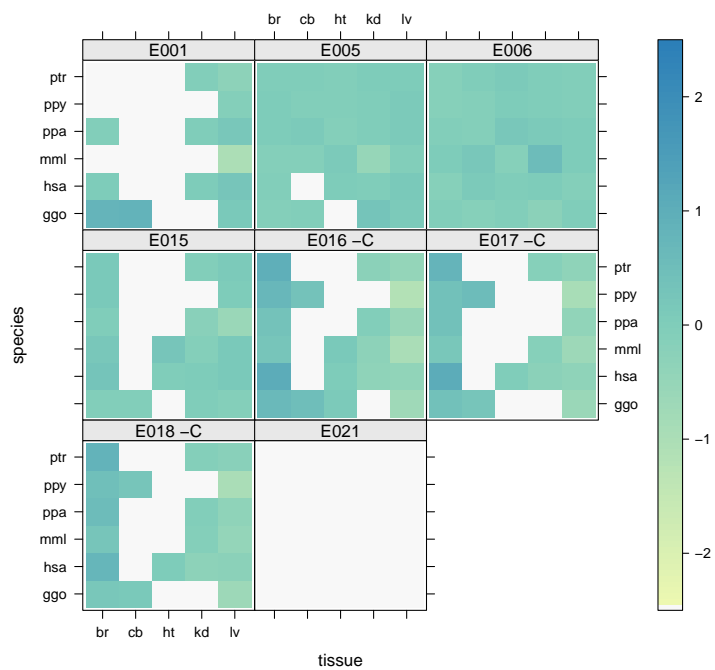
ENSG0000138101



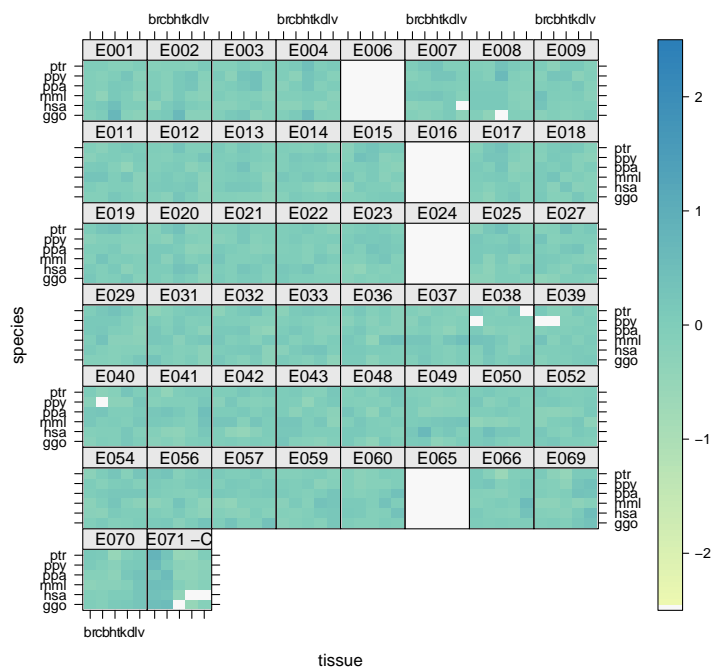




ENSG00000138207



ENSG0000138246

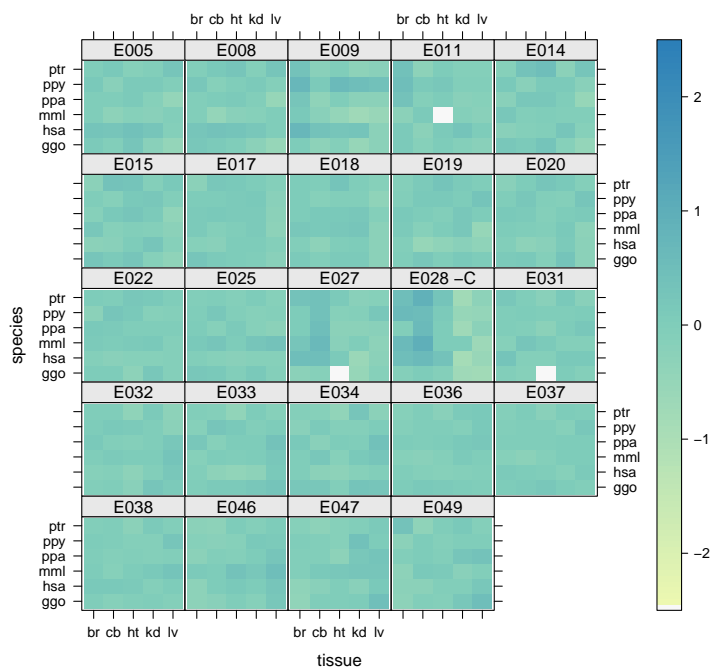




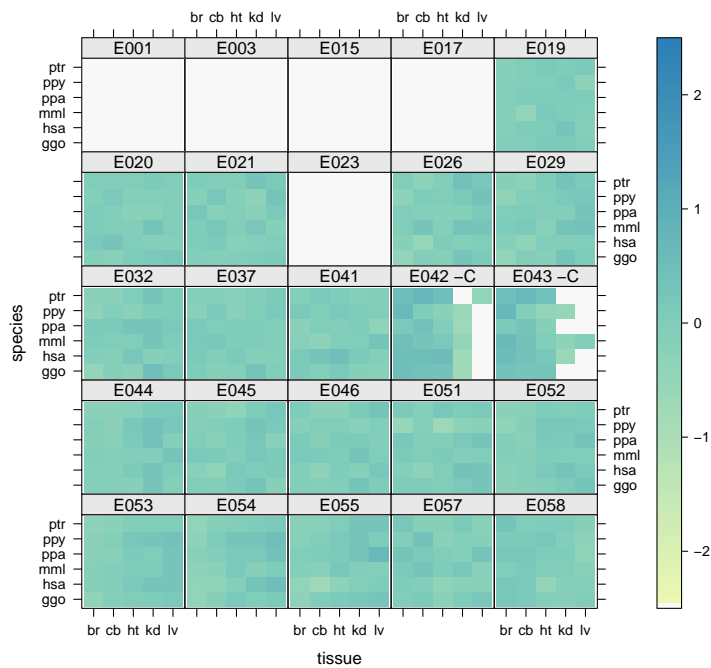




ENSG00000138434

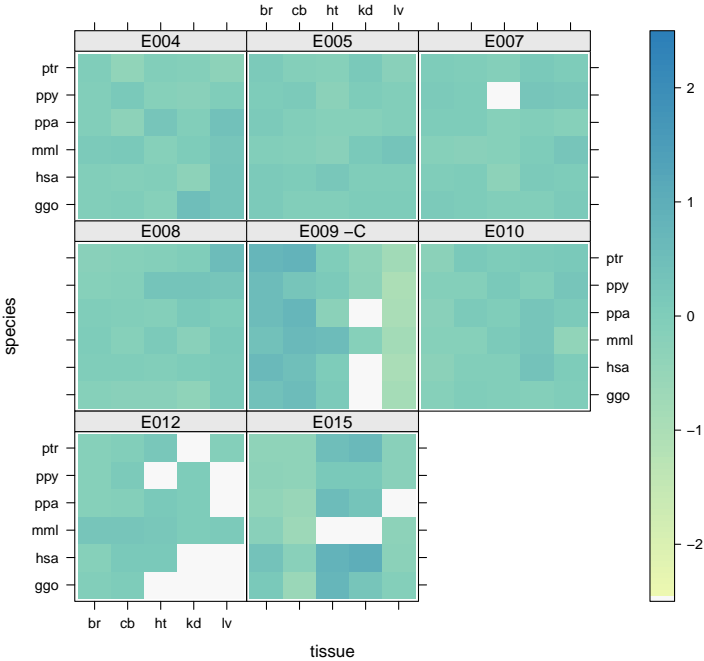


ENSG00000138443

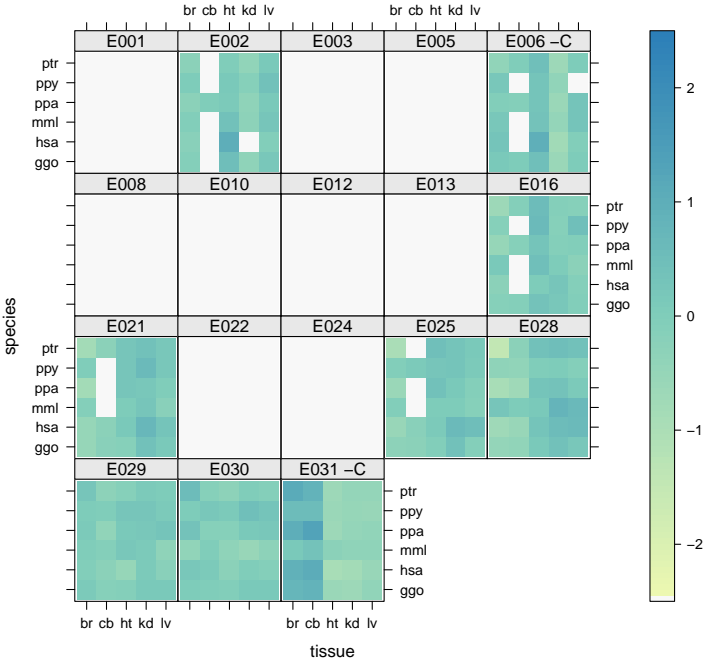




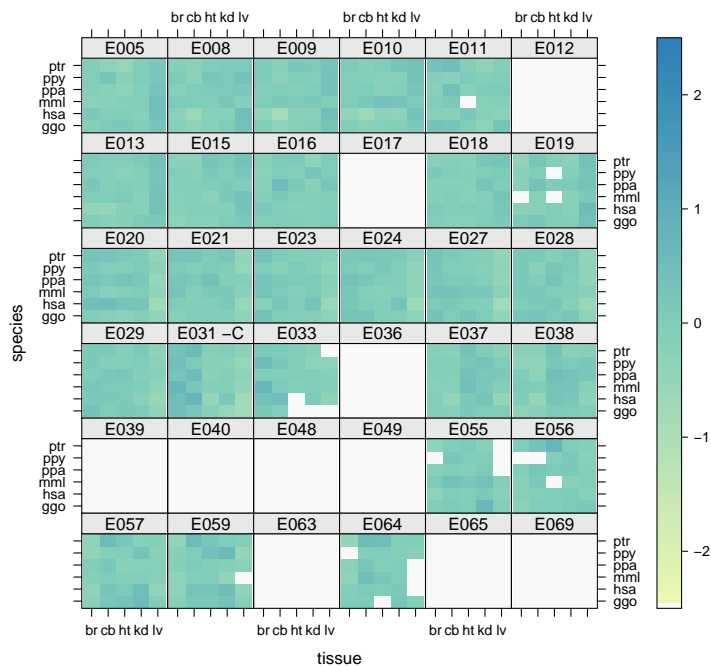
ENSG00000138606



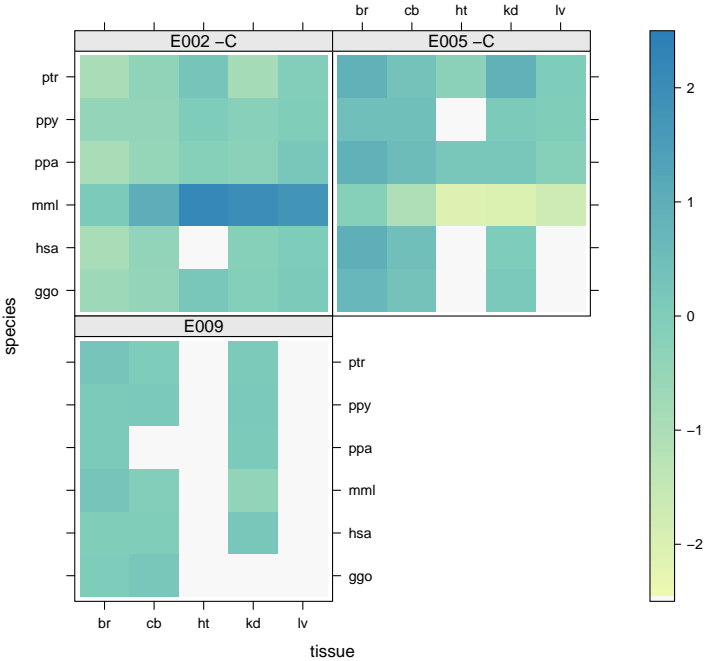
ENSG00000138639



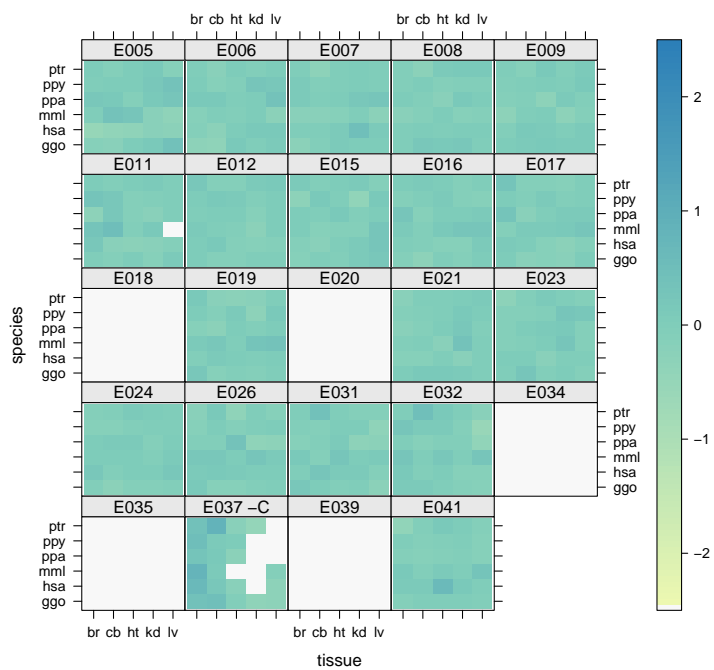
ENSG0000138640



ENSG00000138650



ENSG00000138750



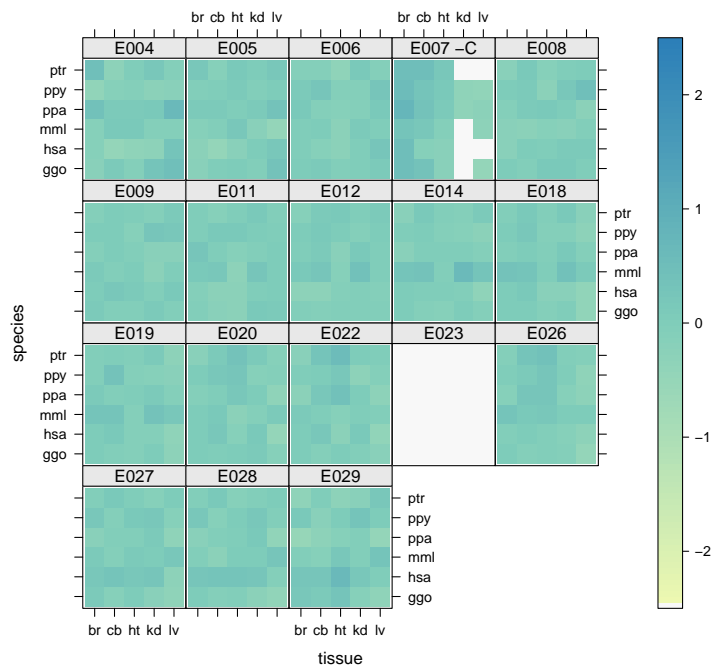




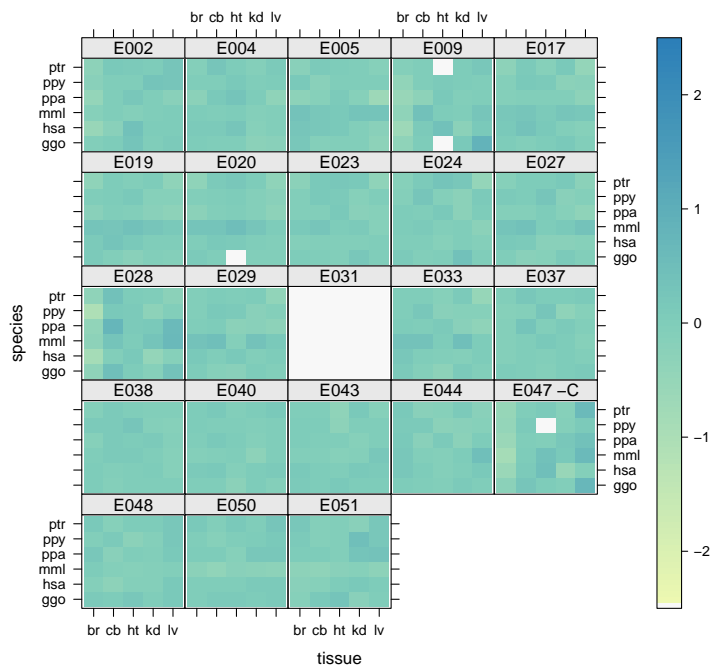




ENSG00000138814

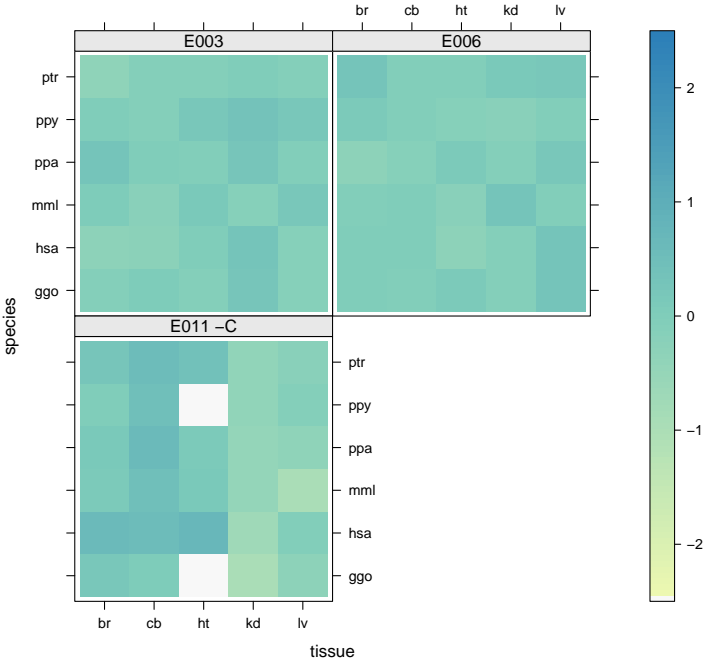


ENSG00000139182

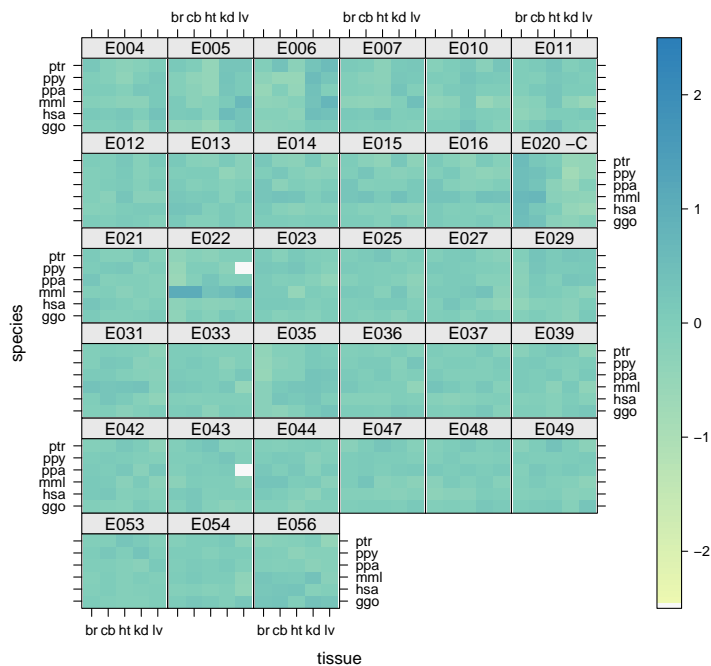




ENSG00000139194



ENSG0000139613



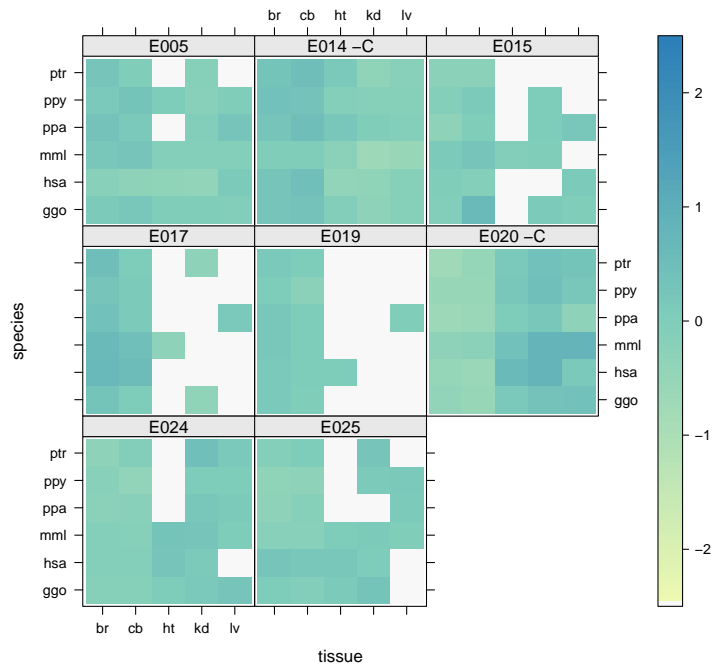






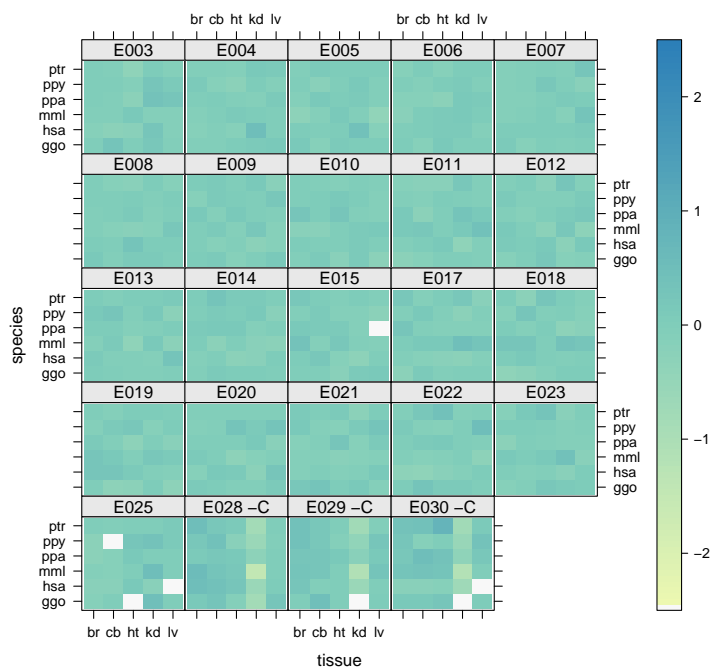


ENSG00000139970



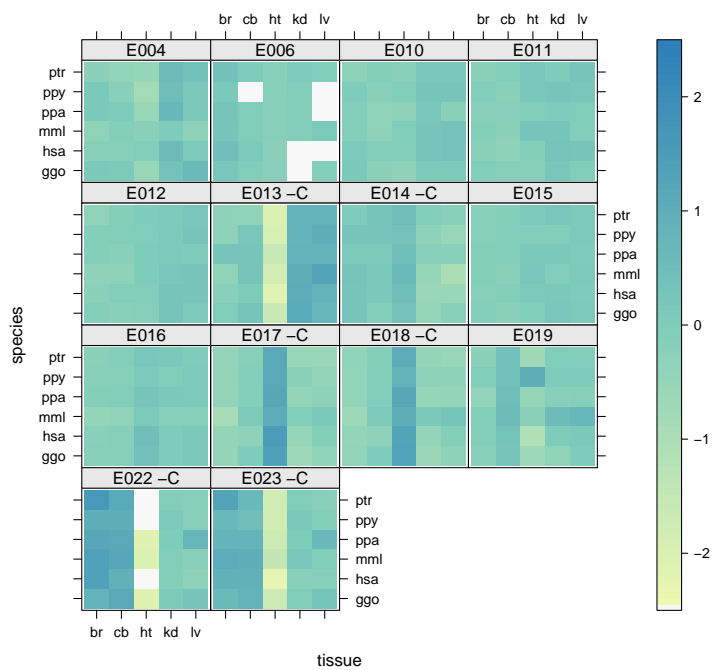


ENSG00000140199



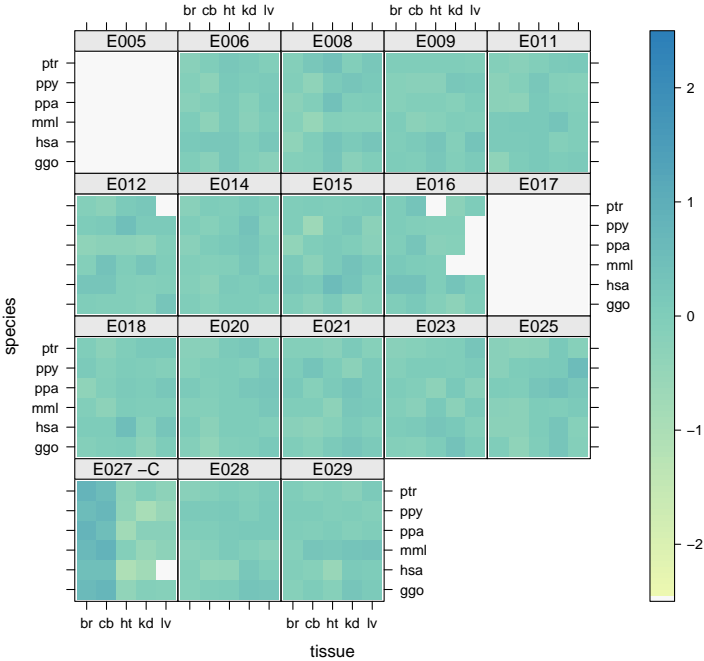


ENSG00000140416

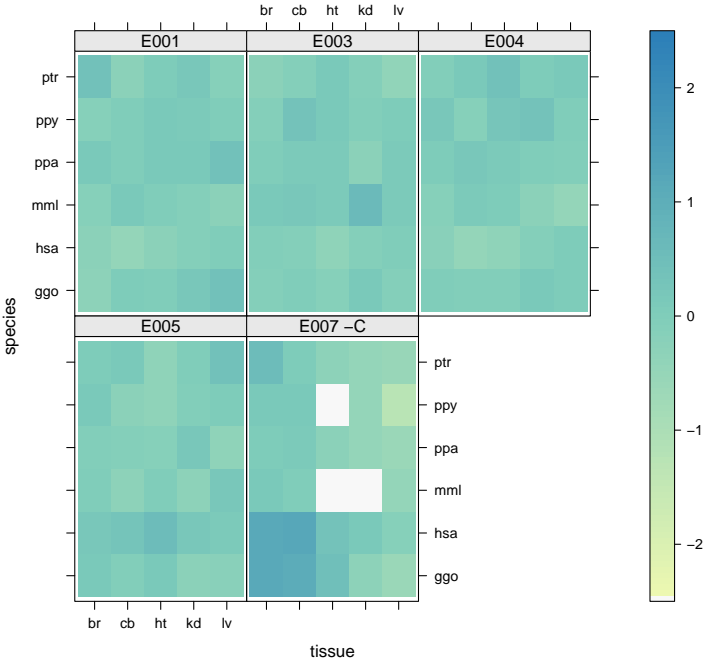




ENSG00000141068



ENSG00000141232

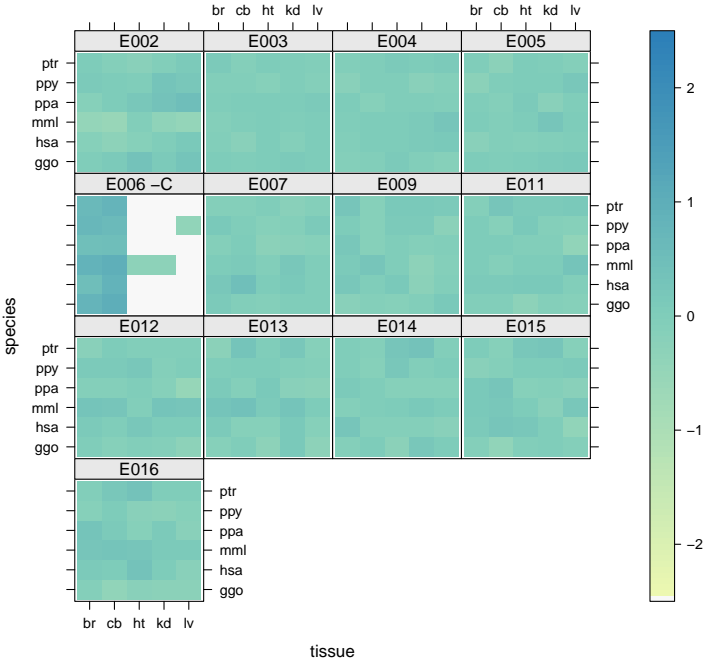






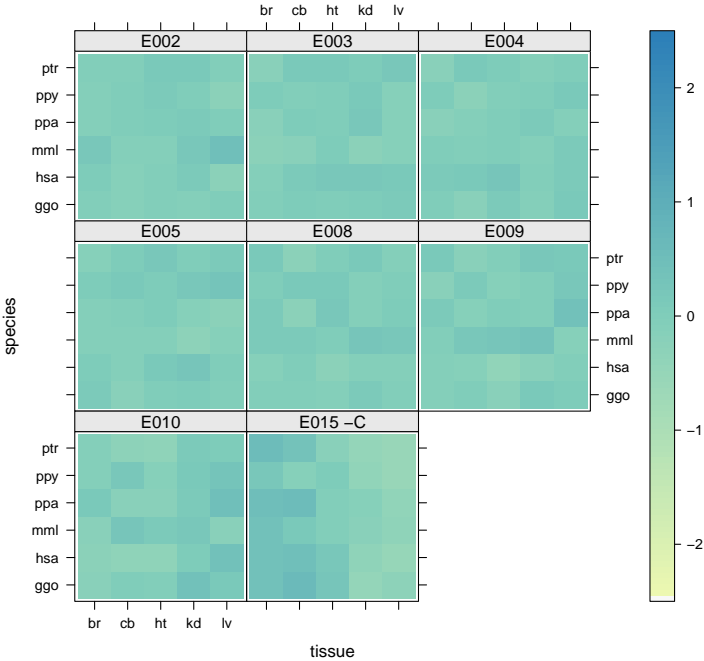


ENSG00000141627

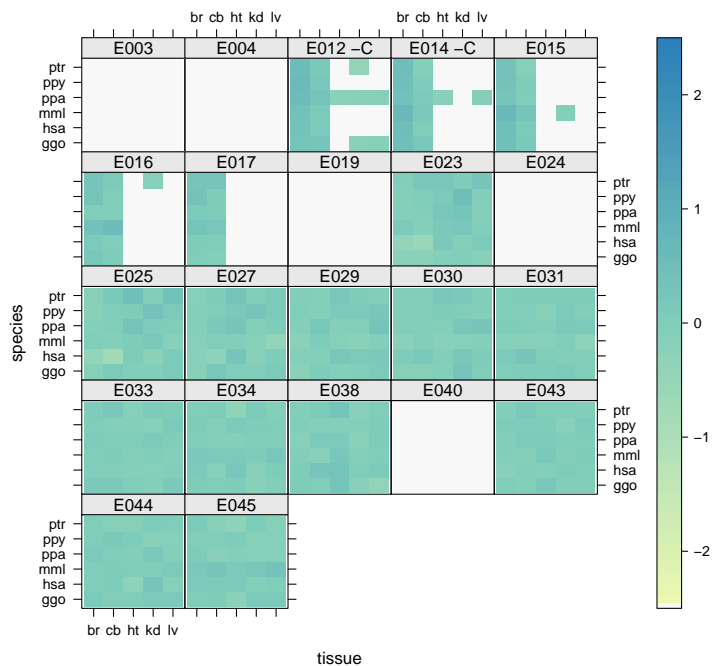




ENSG00000142453

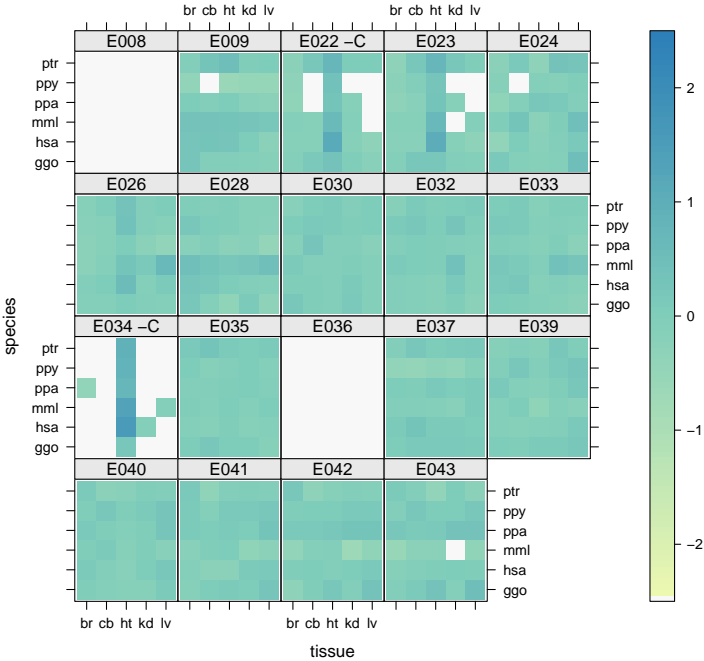


ENSG00000142875



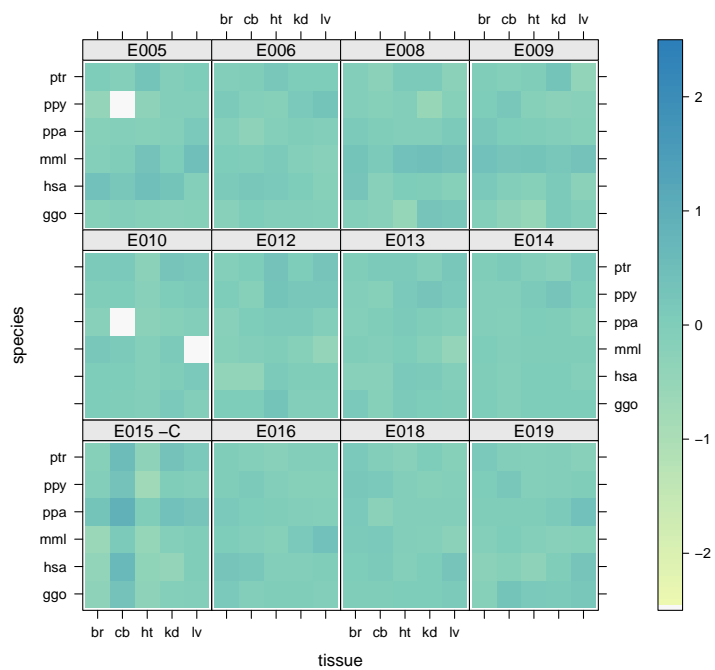


ENSG00000143164



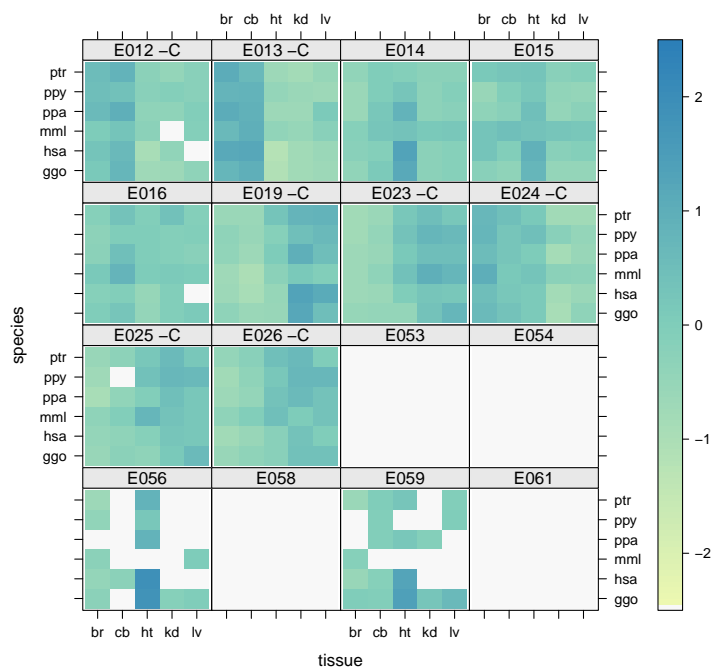


ENSG00000143499

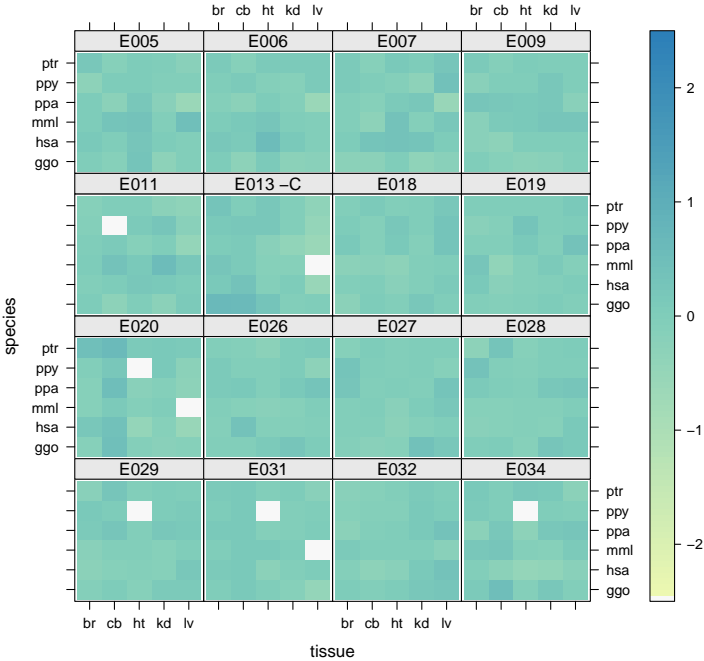




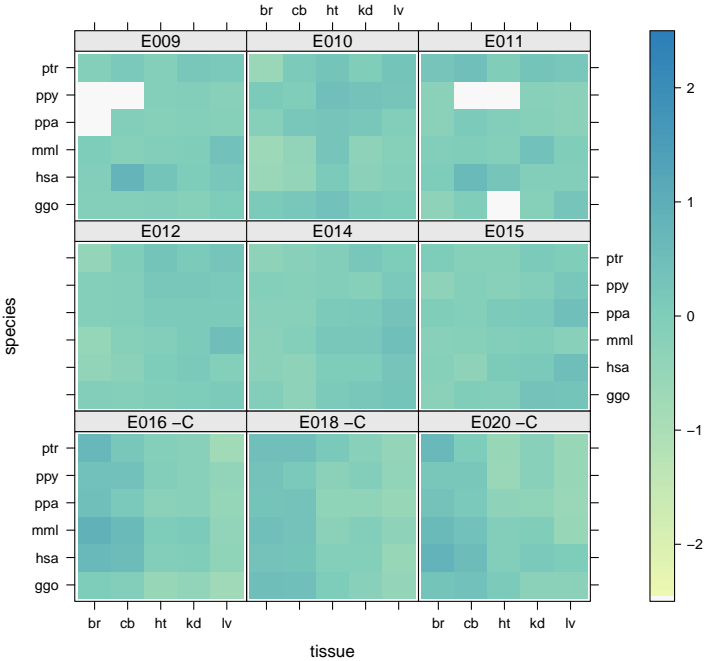
ENSG00000143549



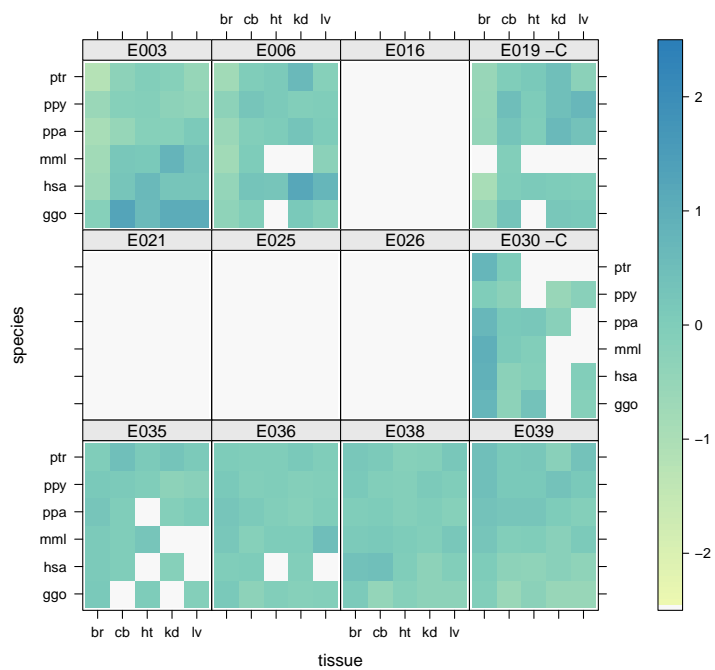
ENSG00000143554



ENSG00000143771

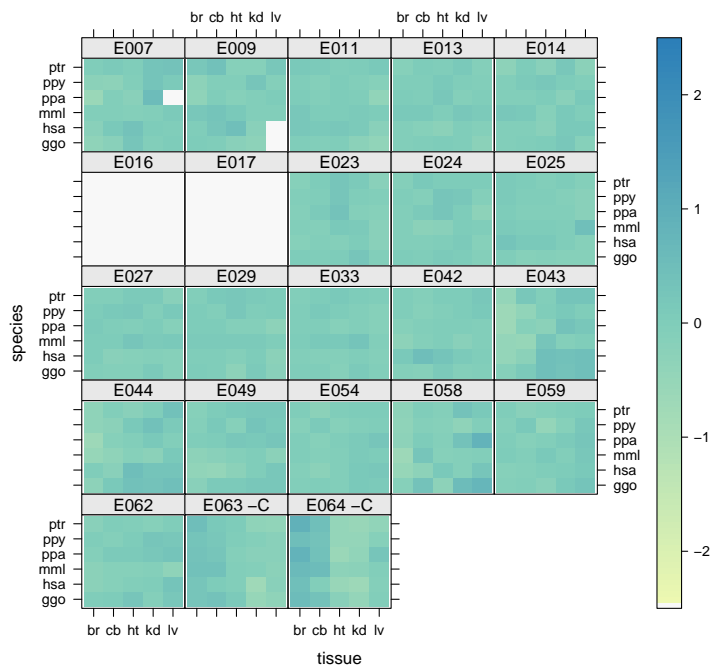


ENSG00000143786





ENSG00000144283



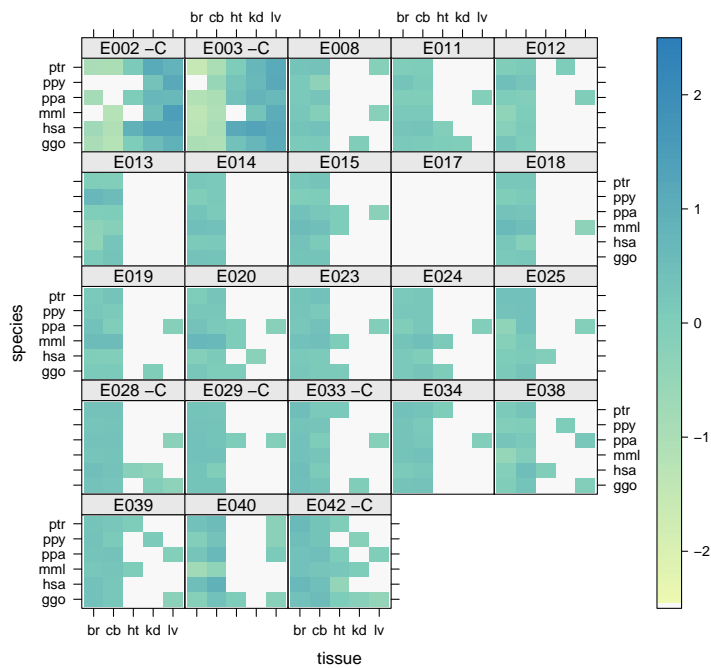




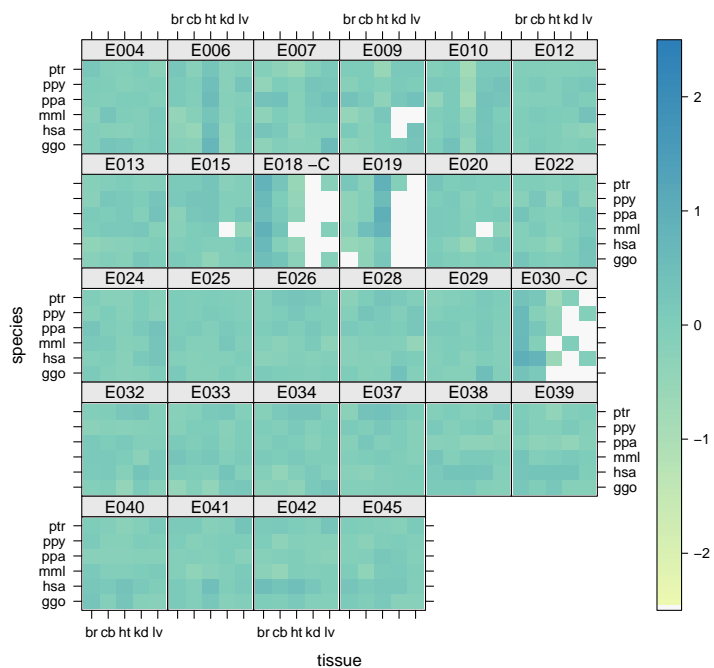




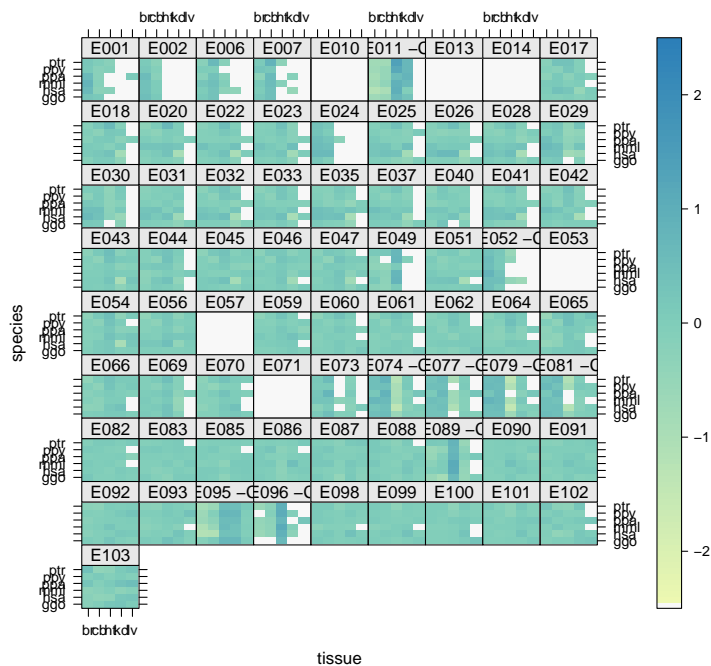
ENSG00000145194



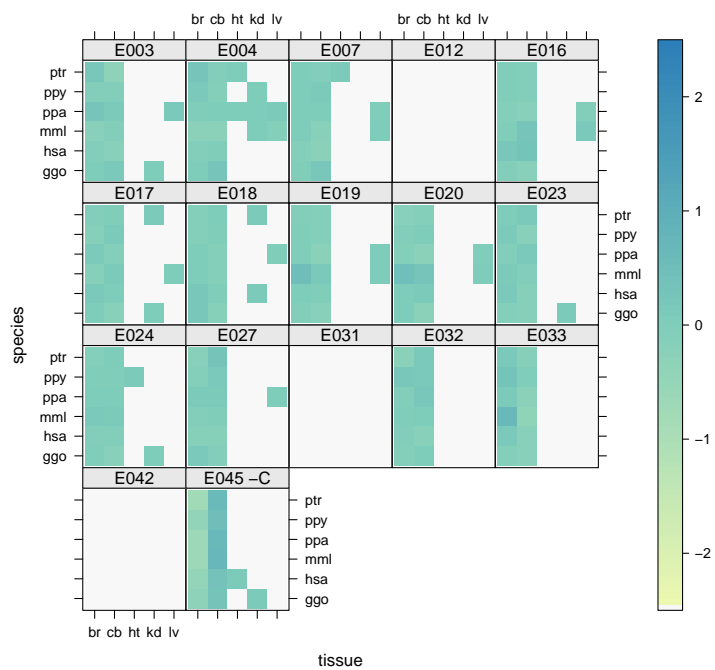
ENSG00000145349



ENSG0000145362



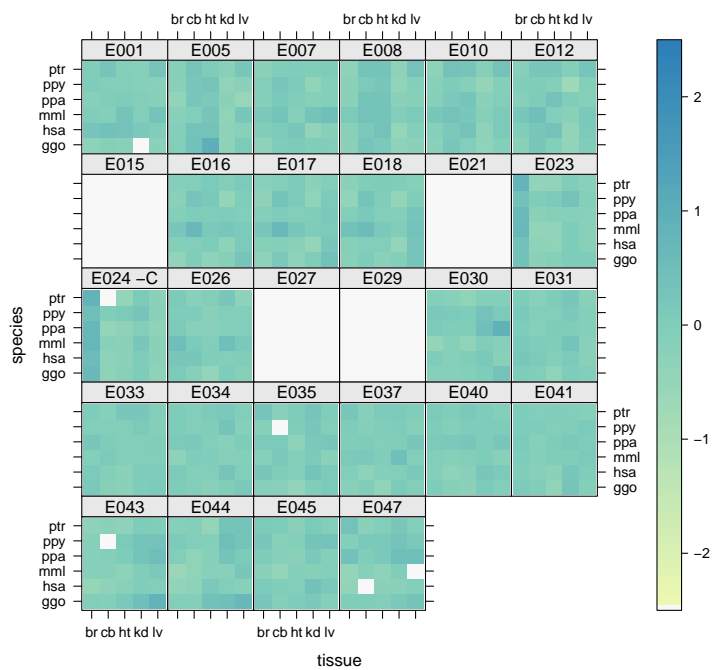
ENSG00000145526



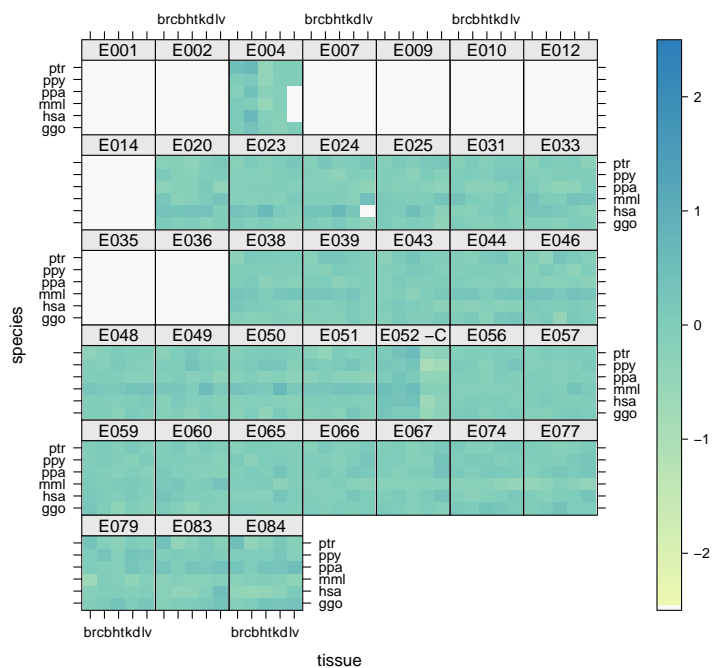




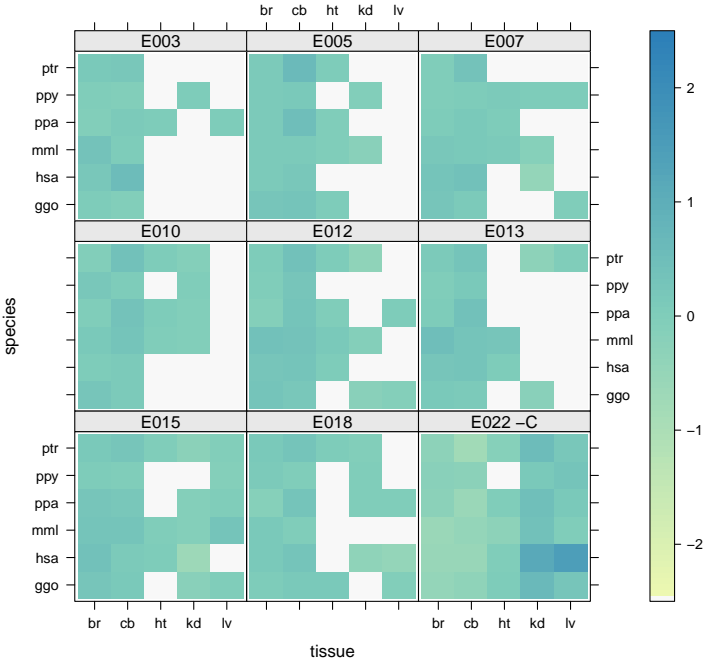
ENSG00000145675



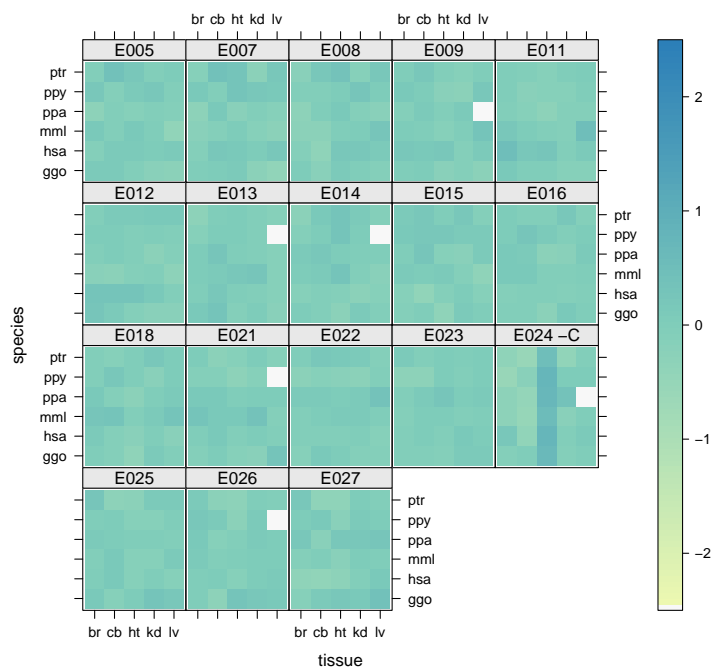
ENSG00000145730



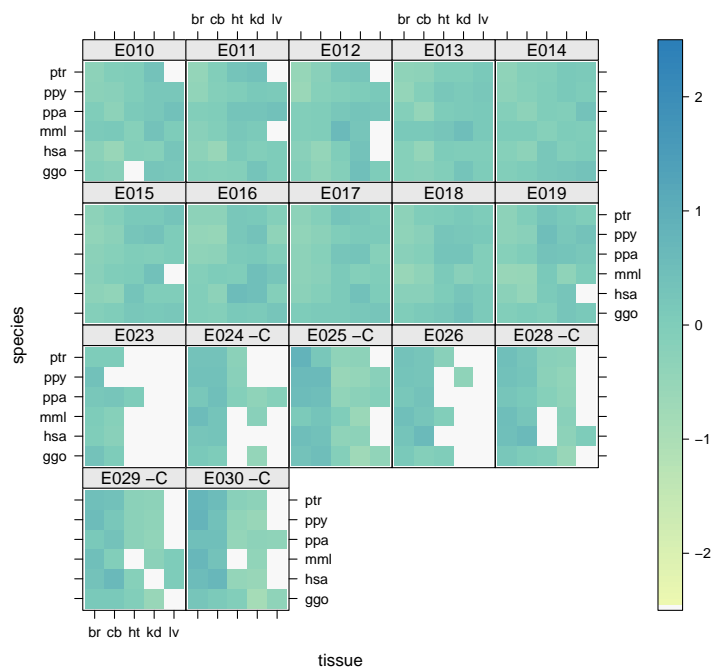
ENSG00000146216



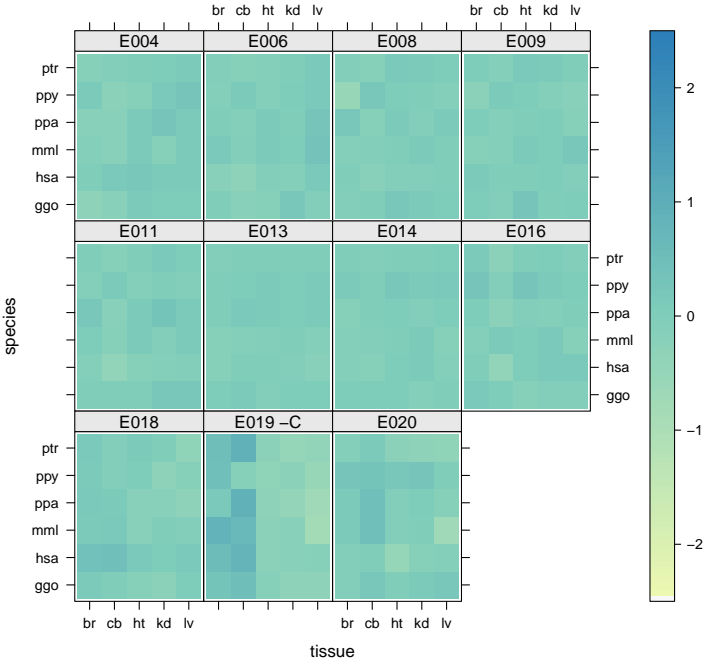
ENSG00000147852



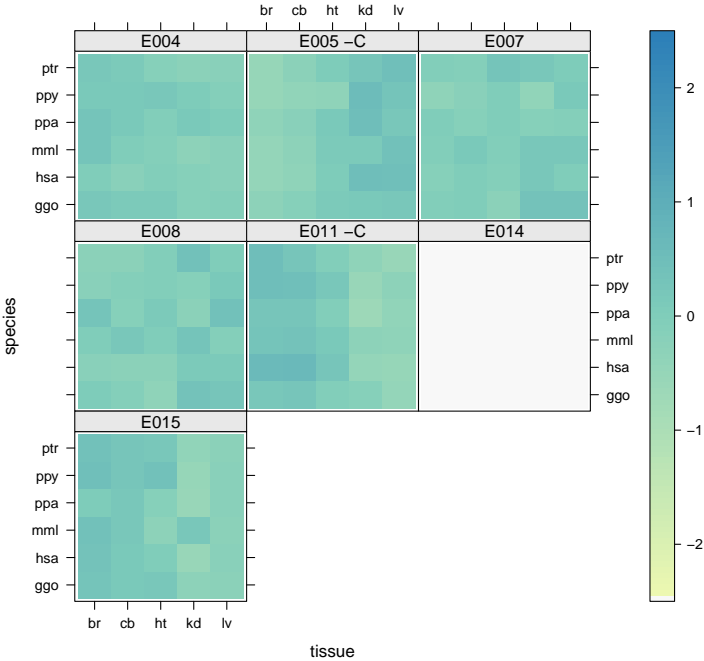
ENSG00000148053



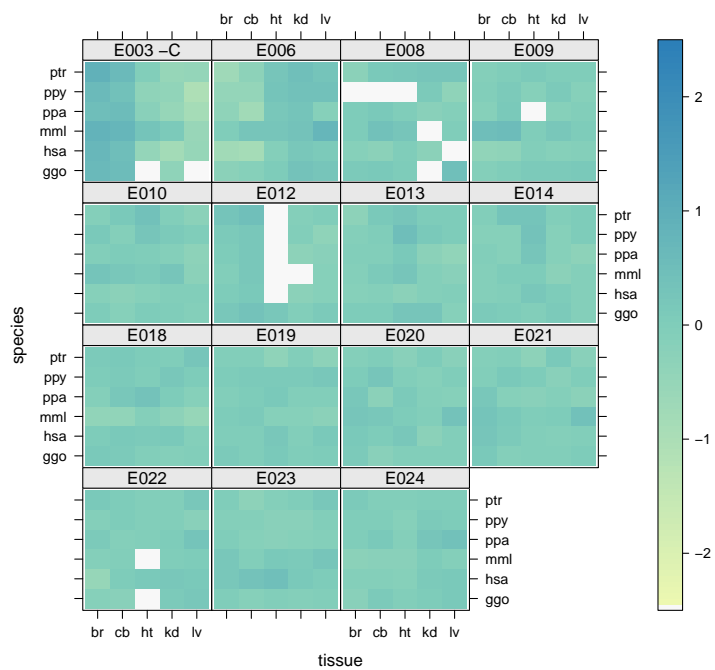
ENSG00000148218



ENSG00000148297



ENSG00000148339

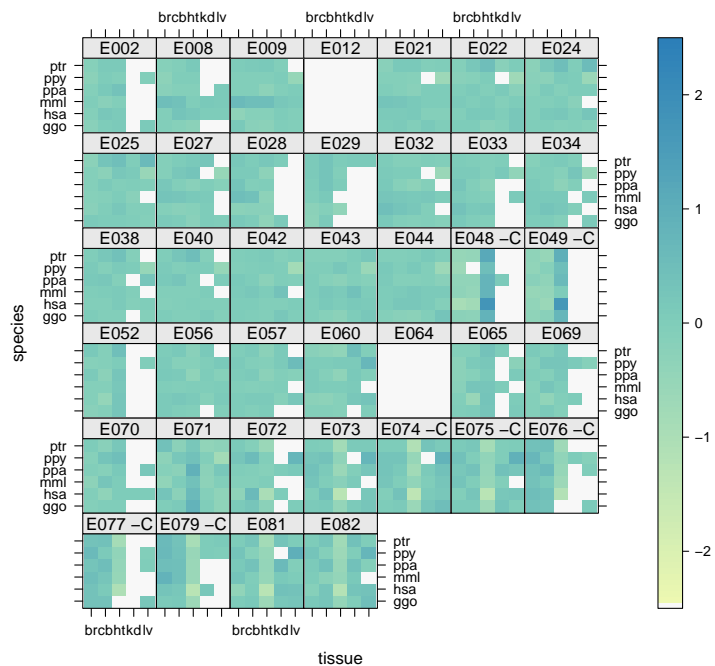






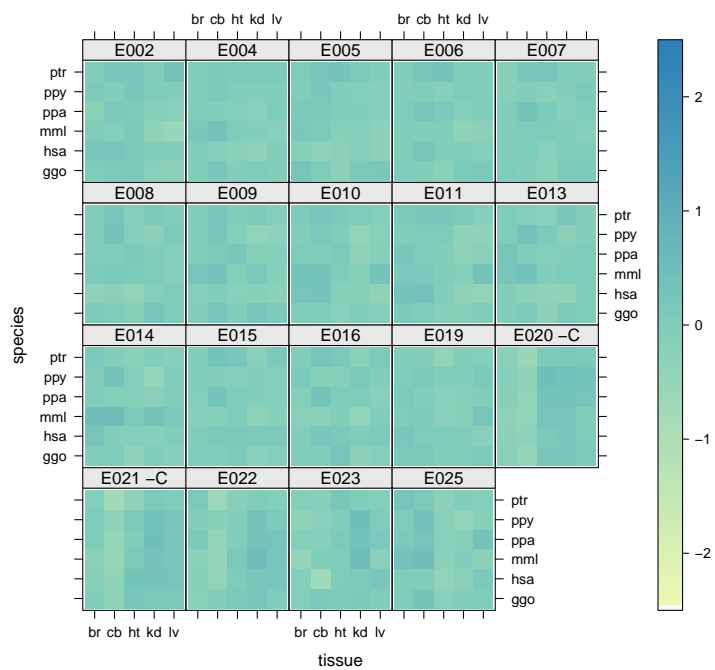


ENSG0000149294

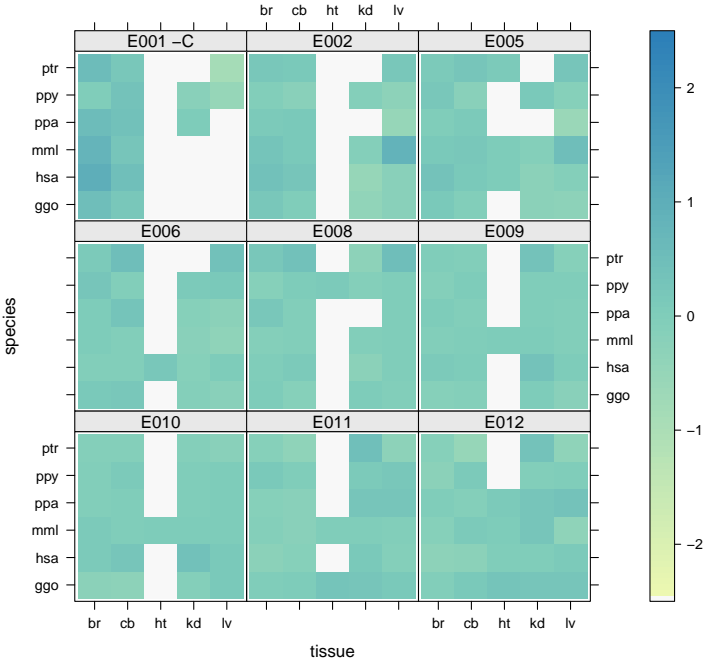




ENSG00000149930

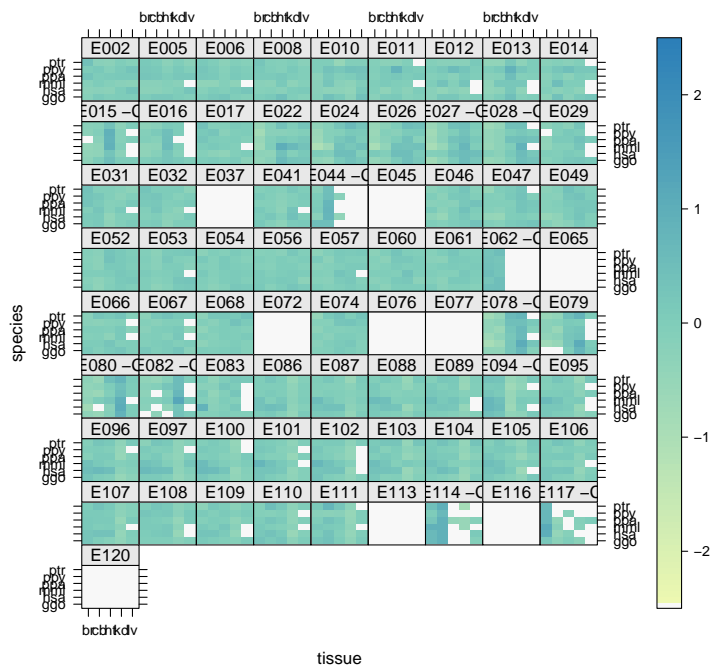


ENSG00000150656





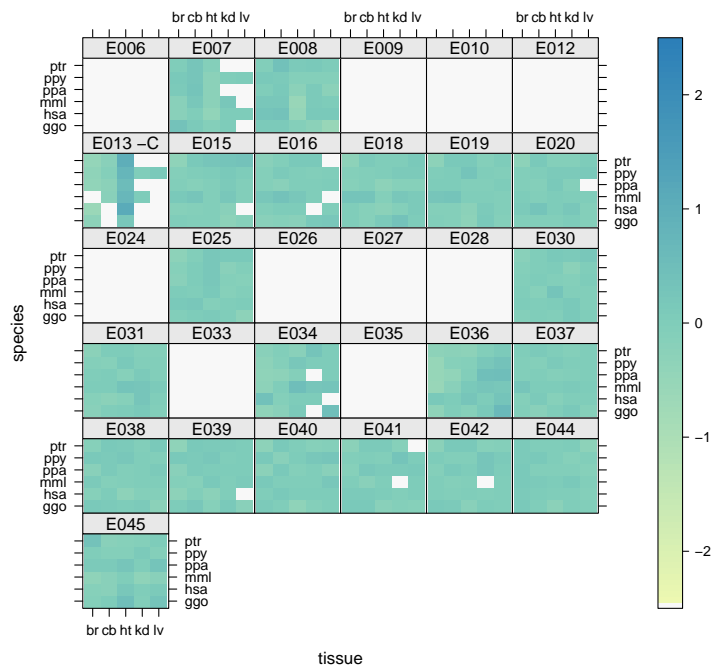
ENSG0000151150







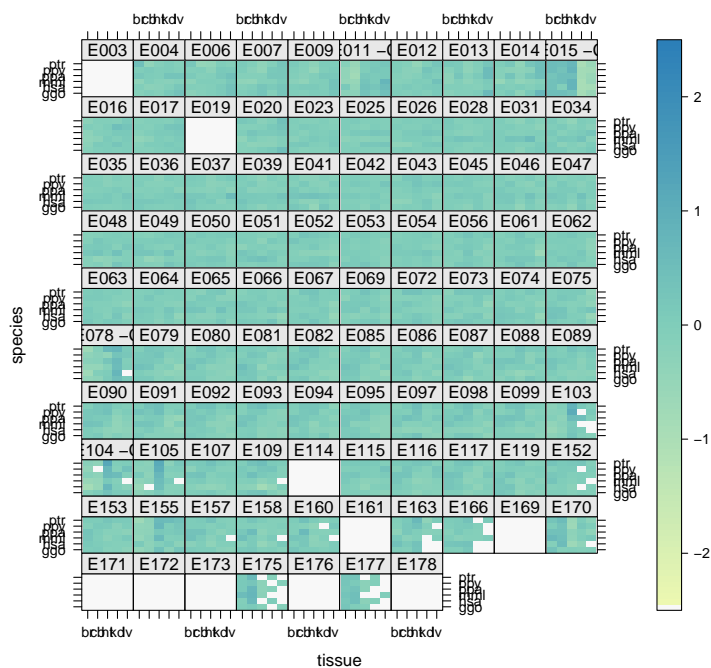
ENSG00000151320



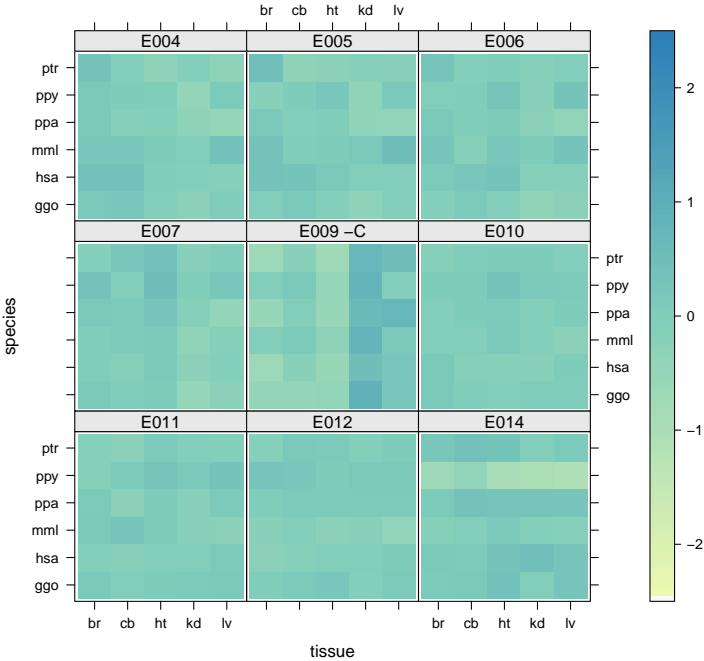




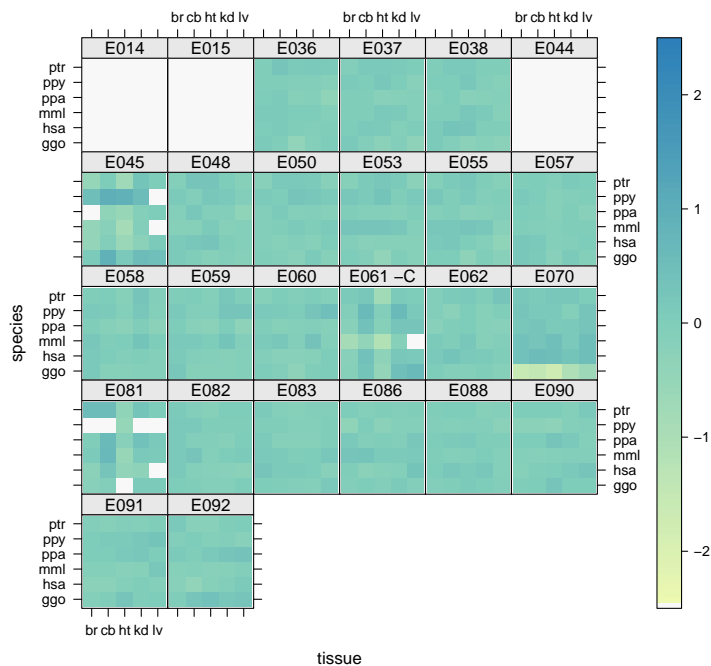
ENSG0000151914



ENSG00000152492



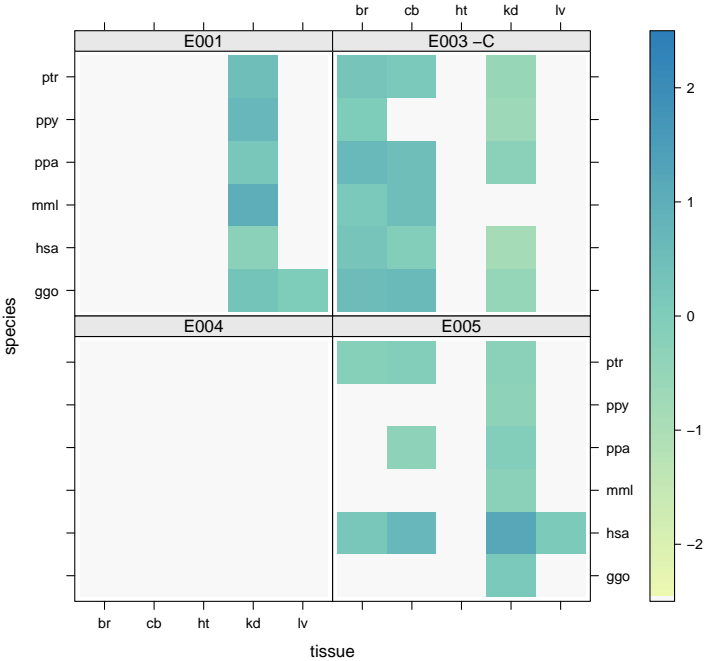
ENSG00000152556







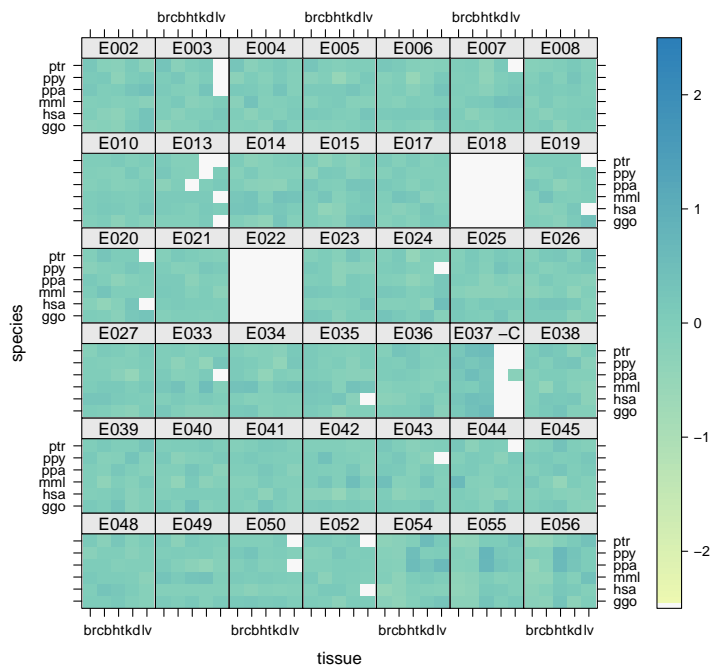
ENSG00000153822





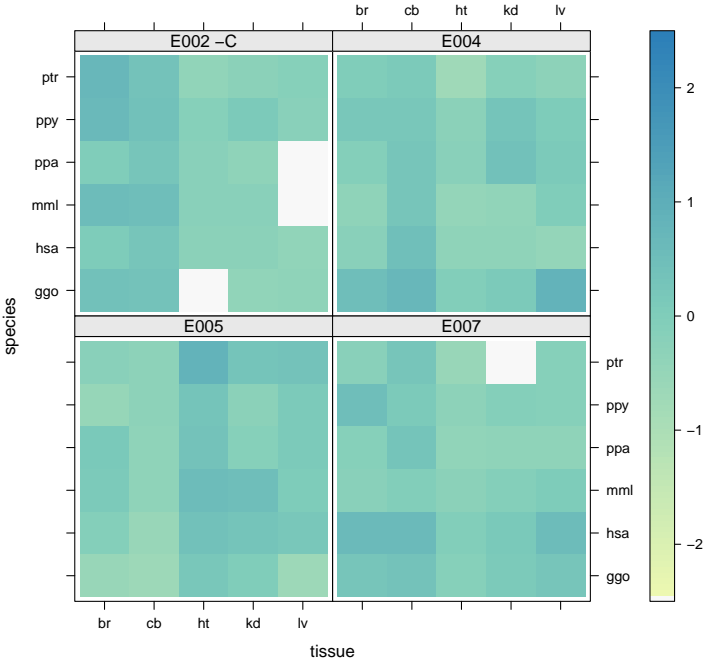


ENSG0000154310



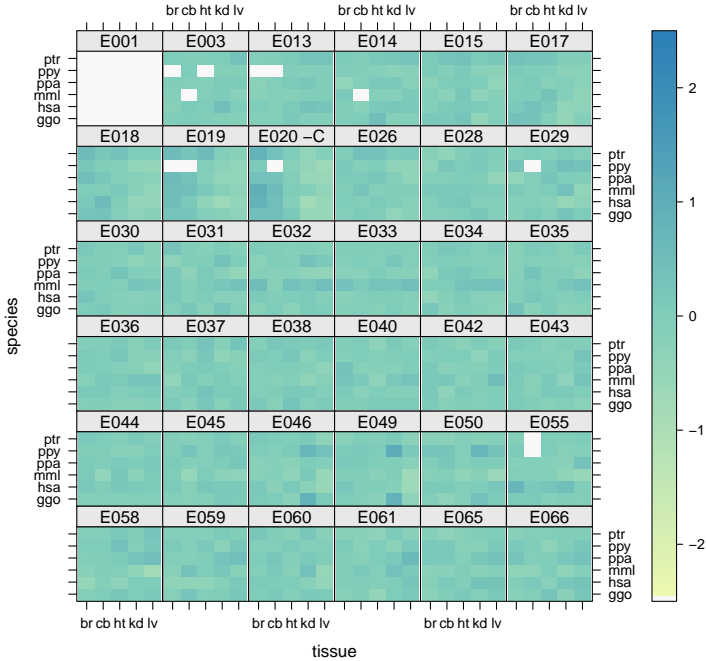


ENSG00000154723



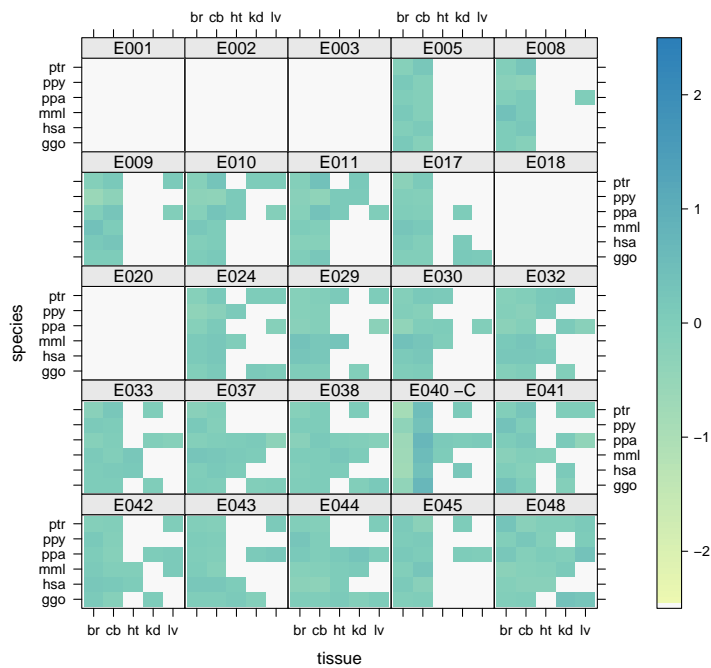


ENSG0000155363

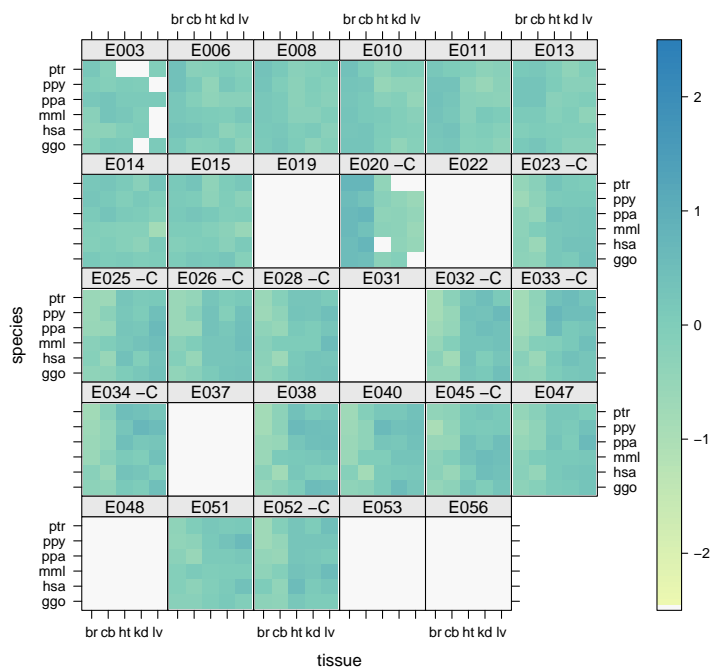




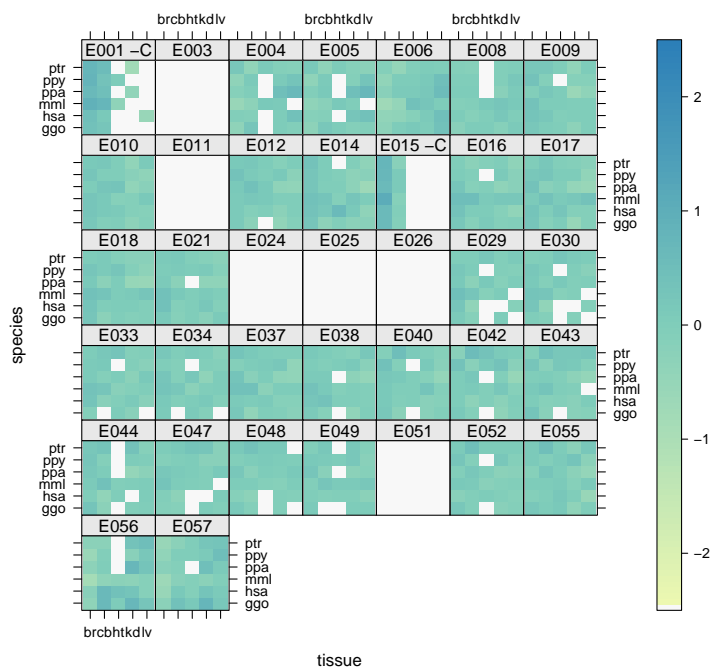
ENSG00000155511



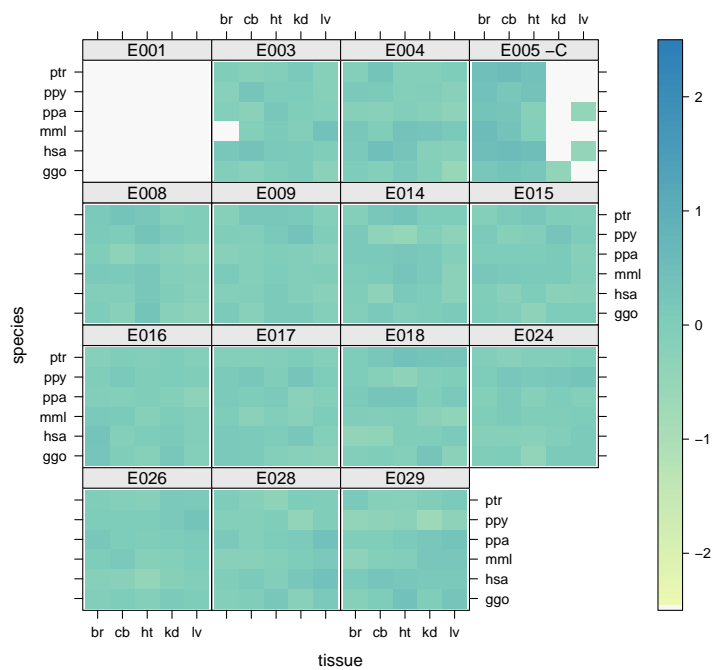
ENSG0000155849



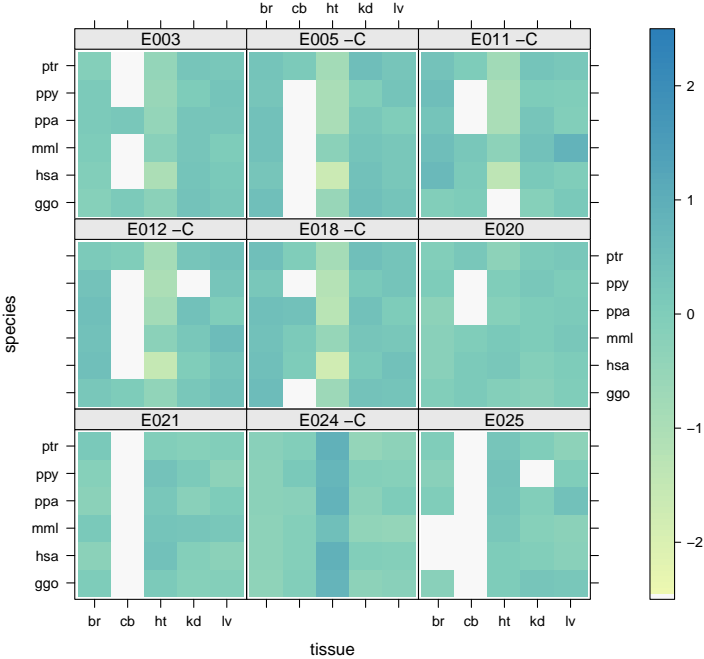
ENSG0000156113



ENSG00000156256

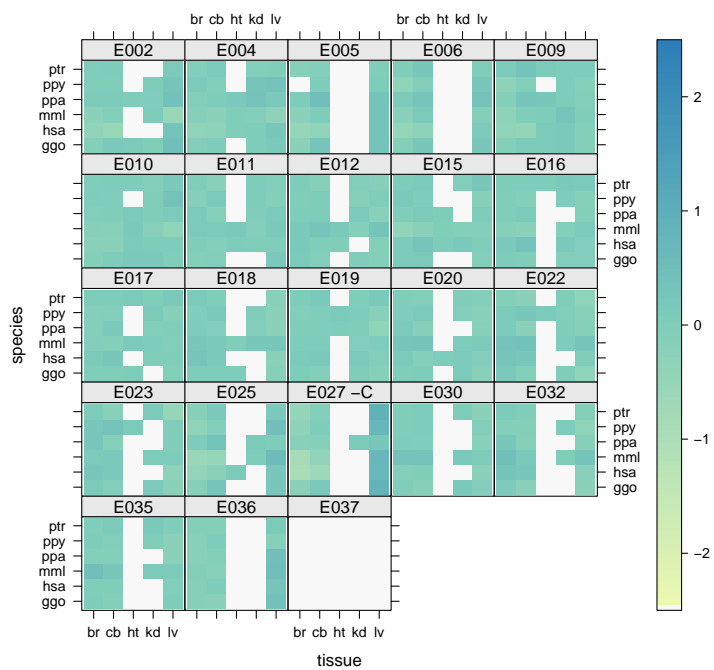


ENSG00000156463





ENSG00000157087



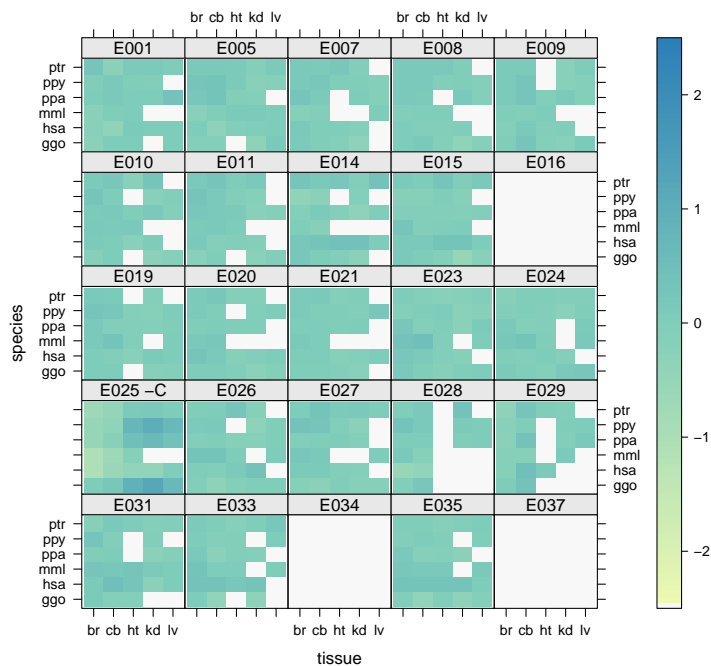








ENSG00000157193

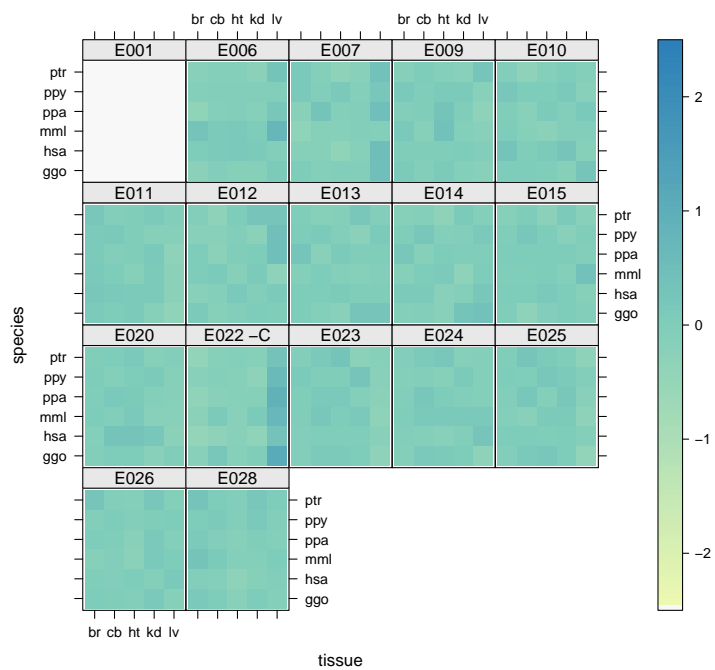








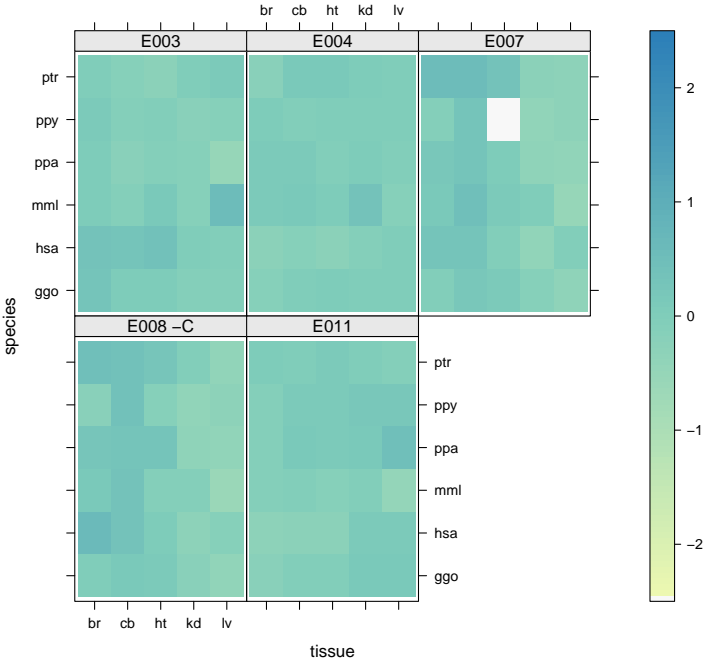
ENSG00000158882



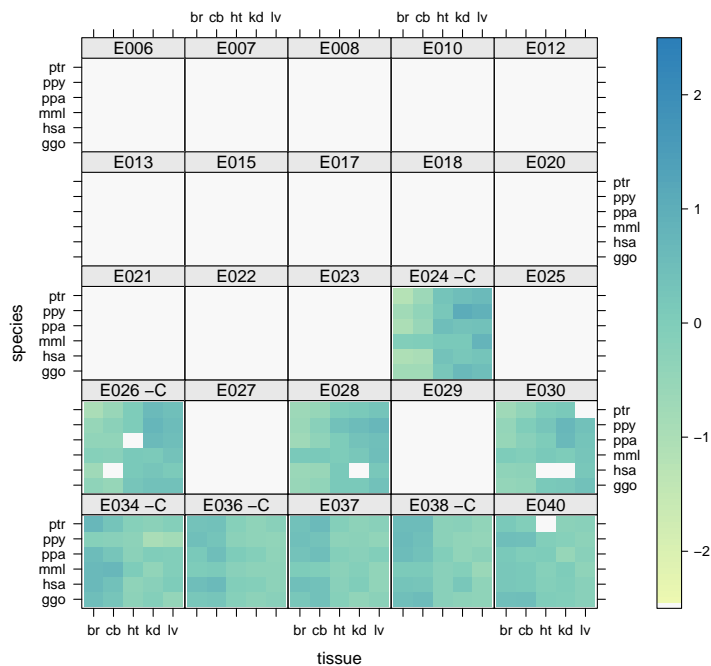




ENSG00000159228



ENSG00000159753

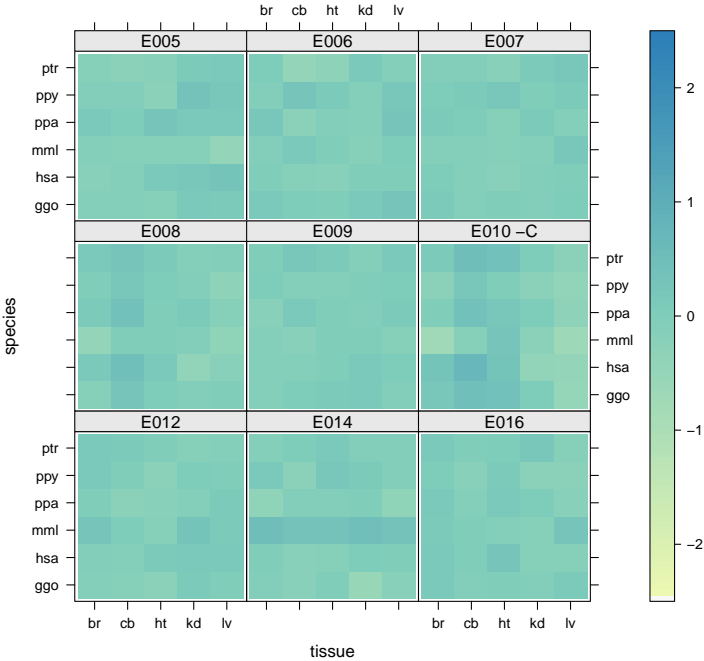




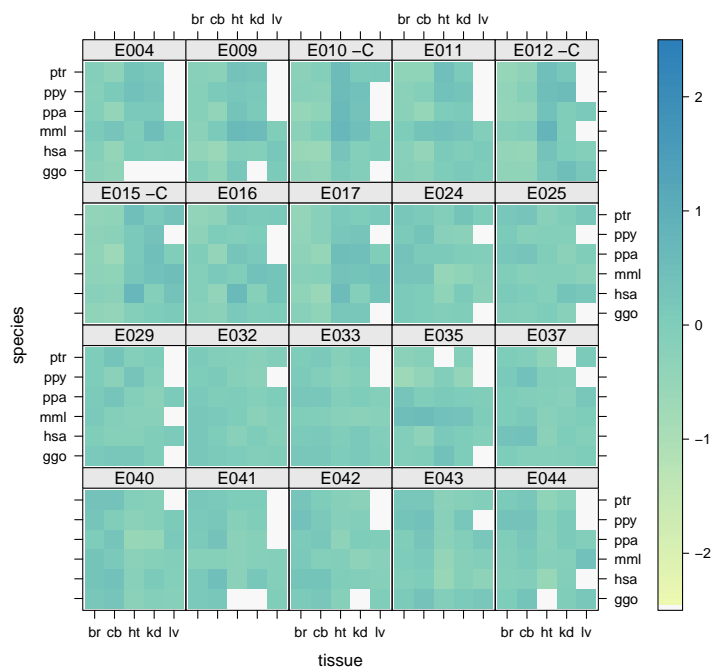




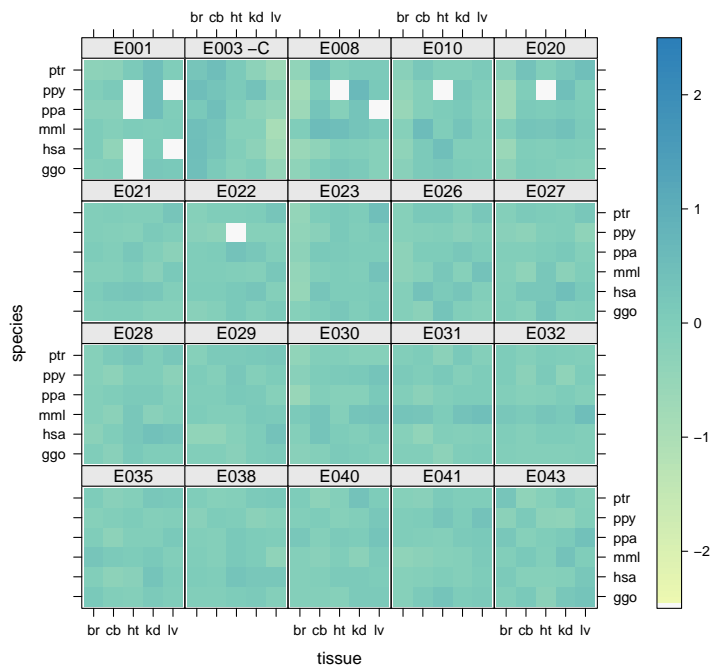
ENSG00000160404



ENSG00000160460



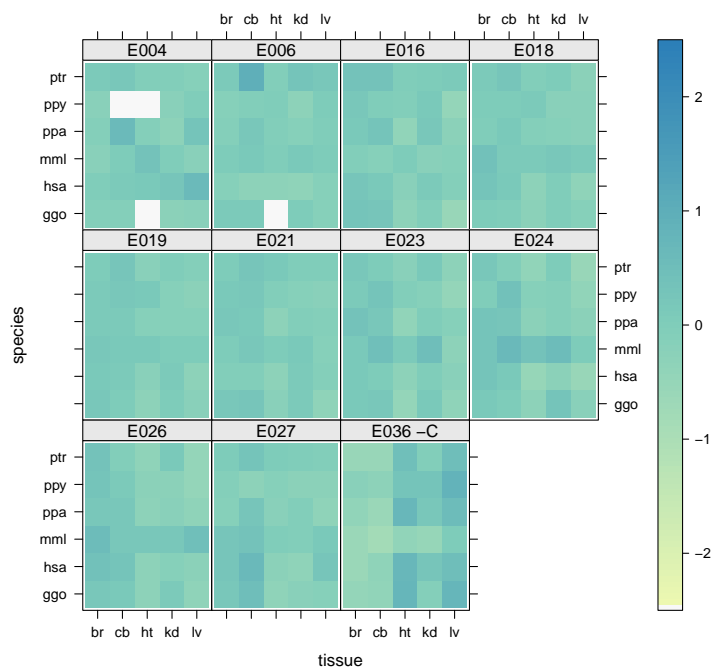
ENSG00000160753







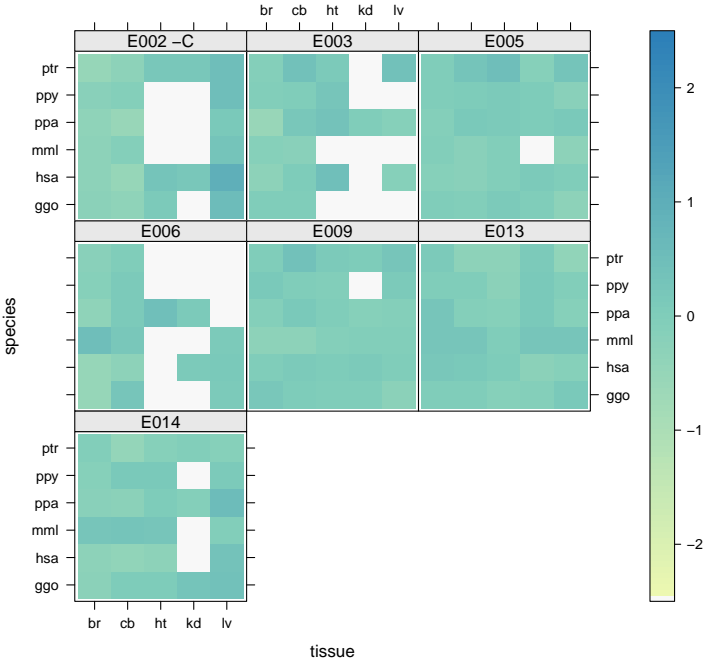
ENSG00000160972





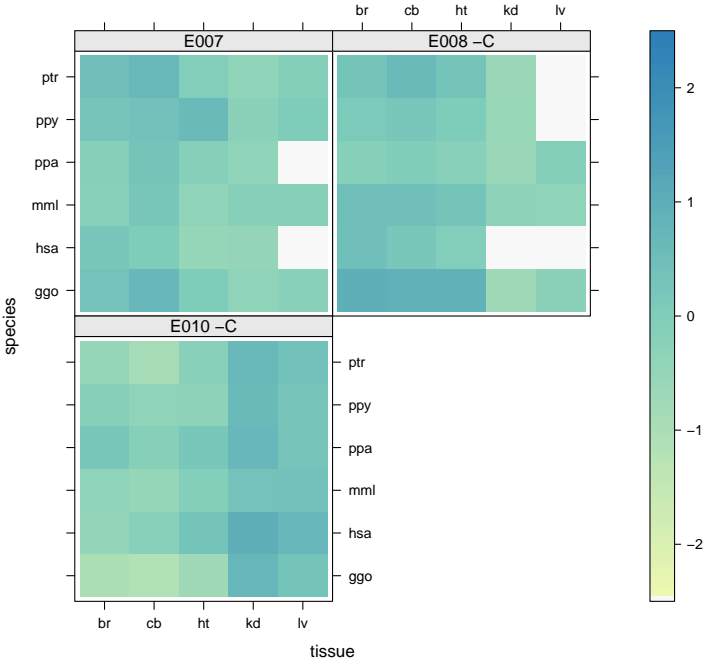


ENSG00000163032

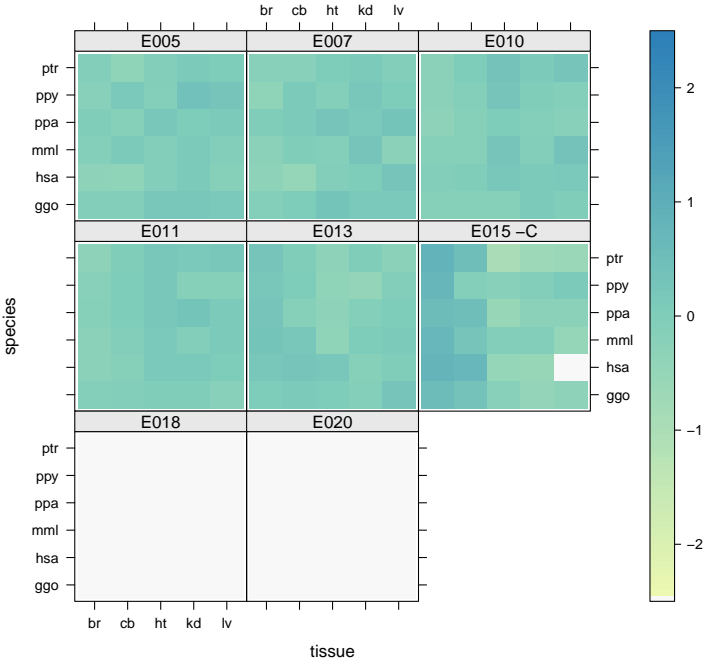




ENSG00000163170



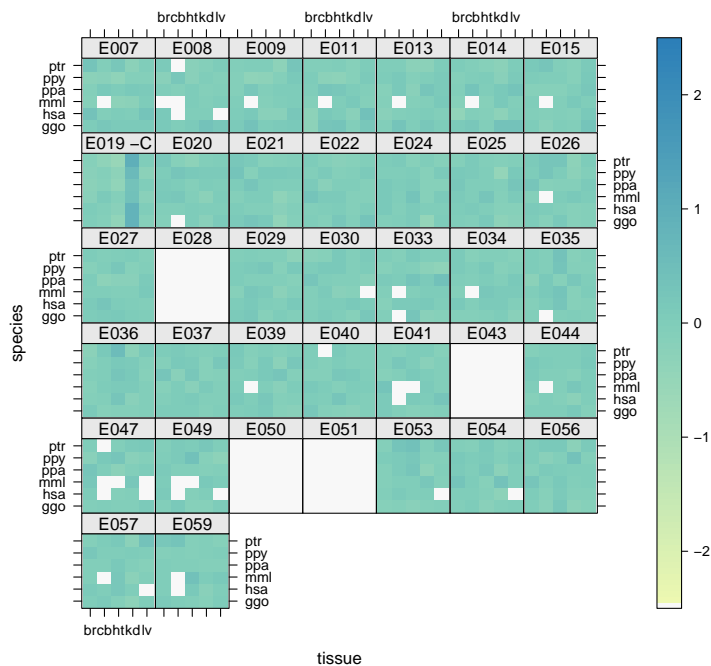
ENSG00000163412







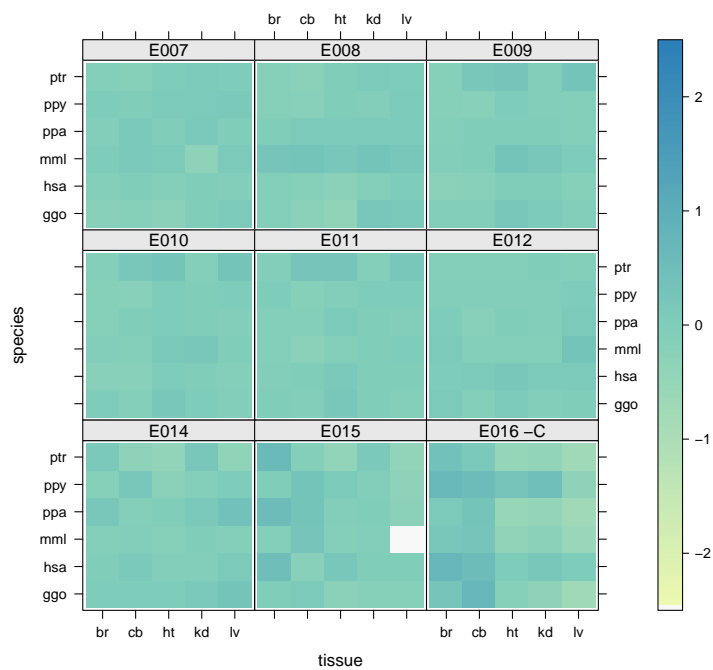
ENSG0000163638





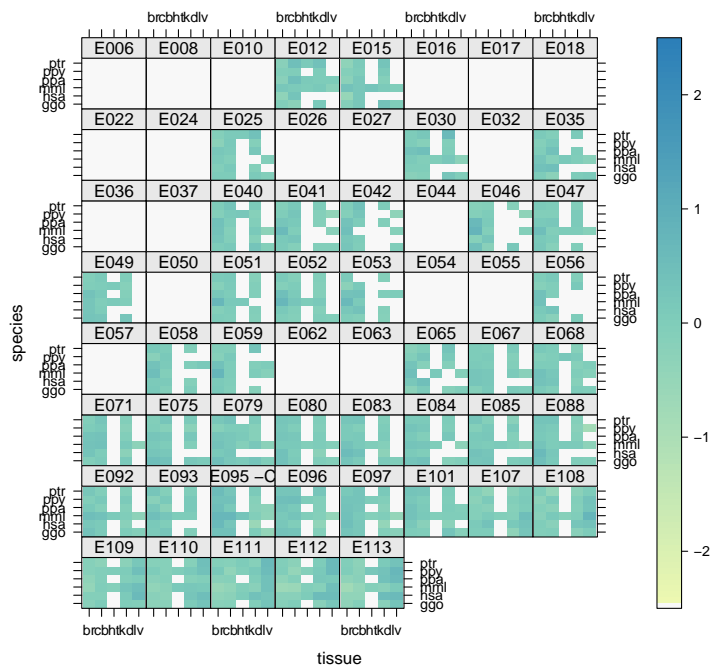


ENSG00000164022

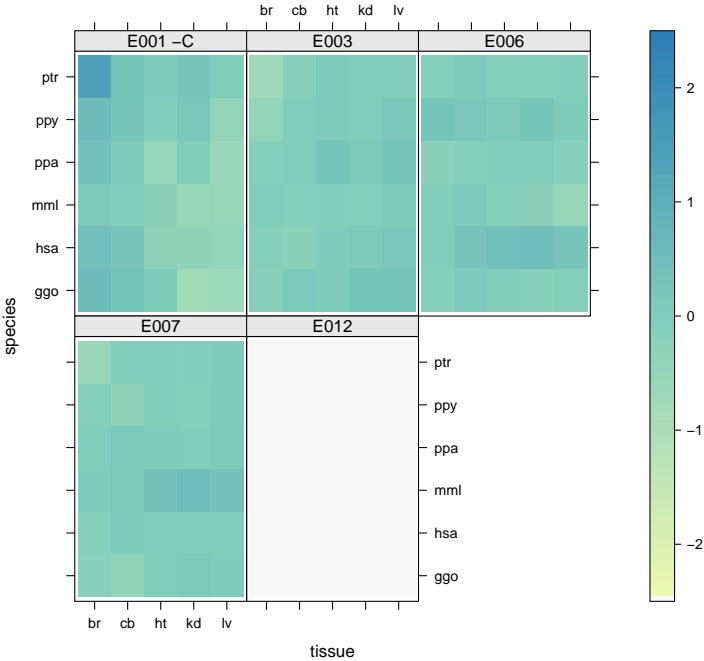




ENSG0000164199



ENSG00000164237









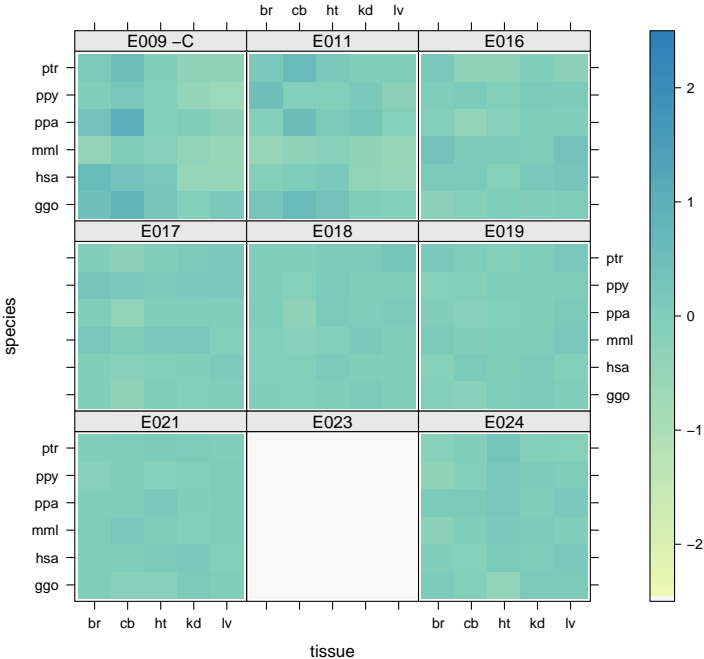




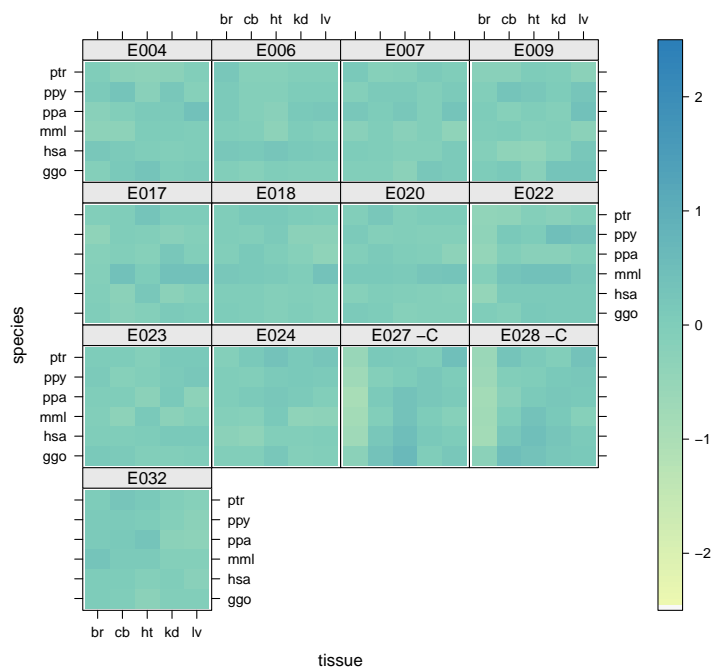




ENSG00000165609



ENSG00000165802

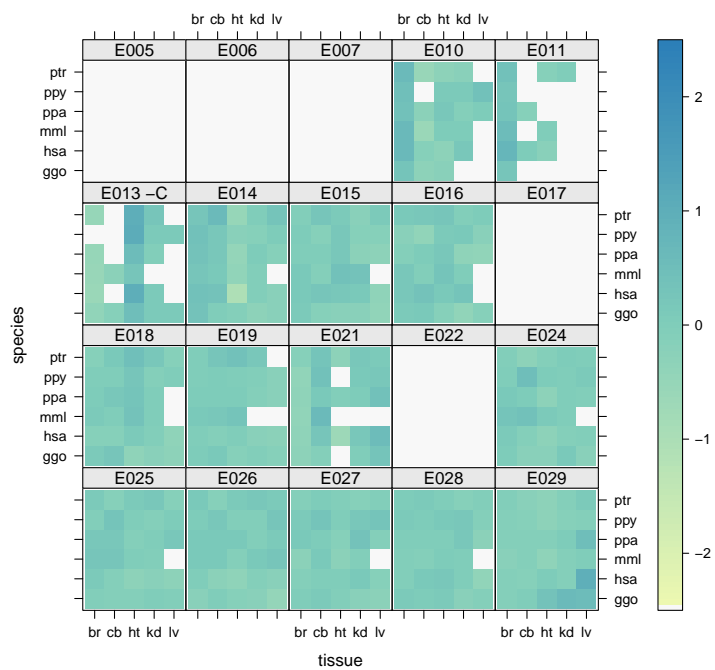




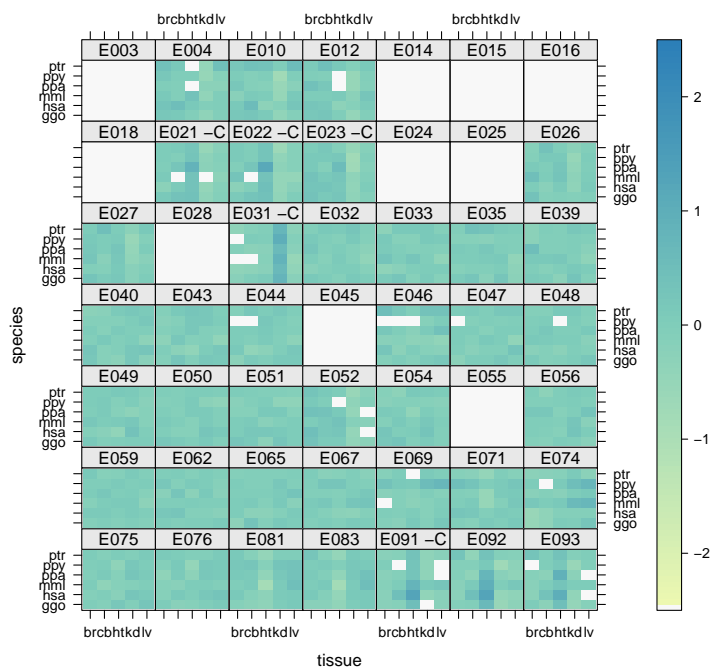




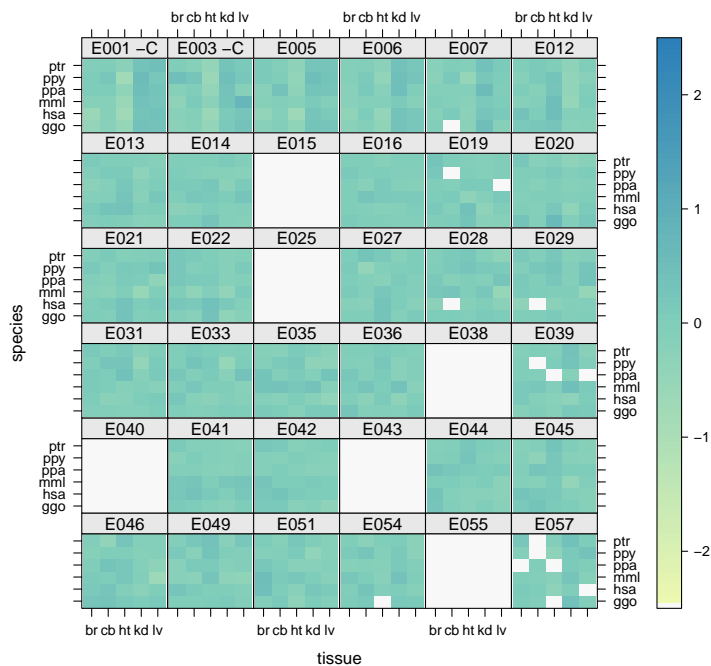
ENSG00000165995



ENSG0000166387

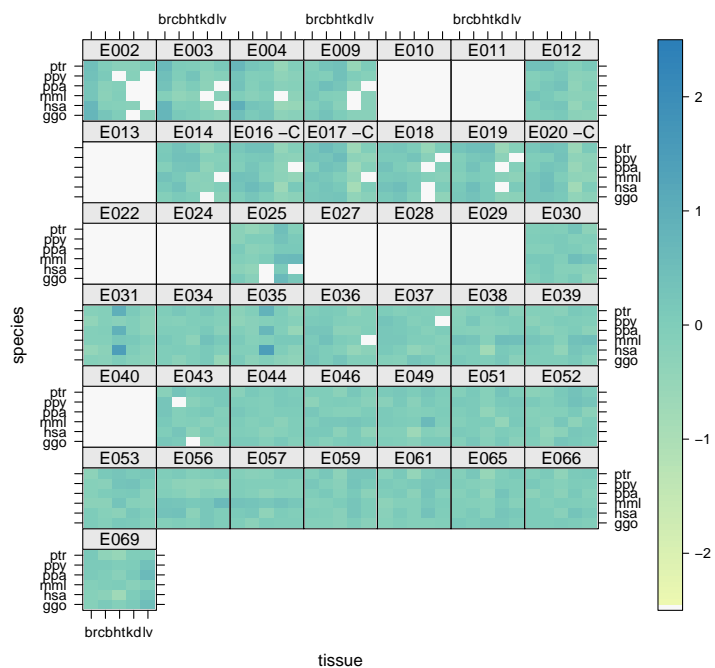


ENSG0000166689

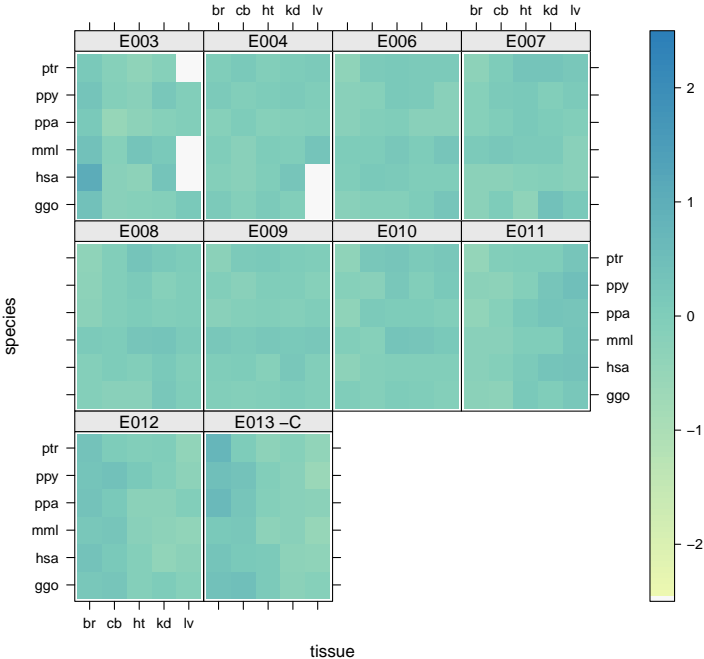




ENSG0000166833



ENSG00000166974

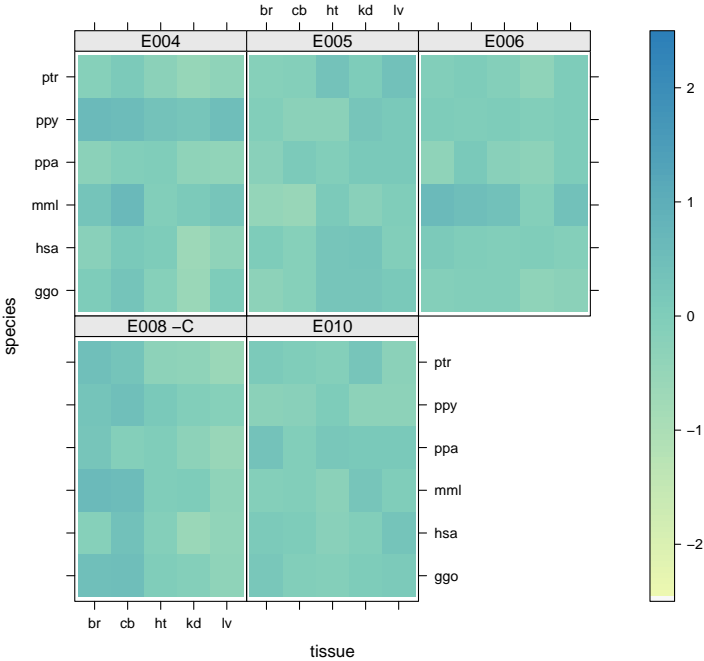






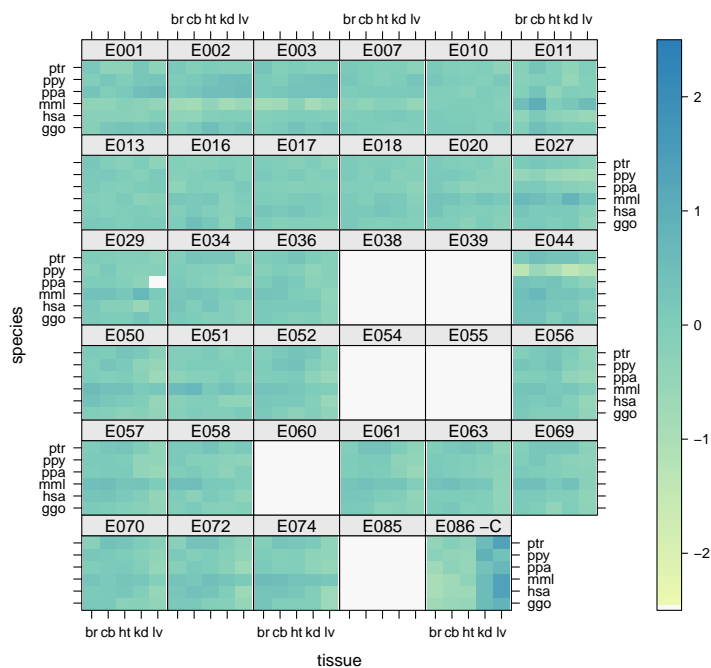


ENSG00000167700

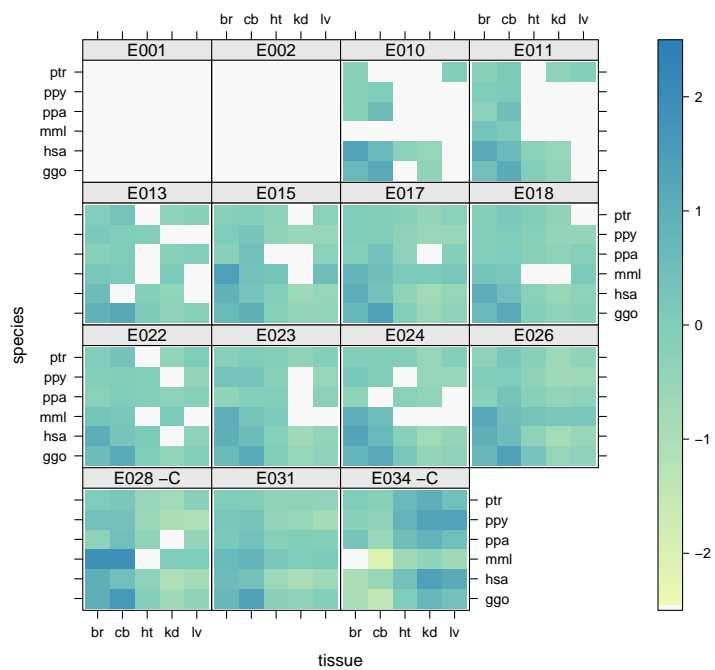




ENSG0000167986

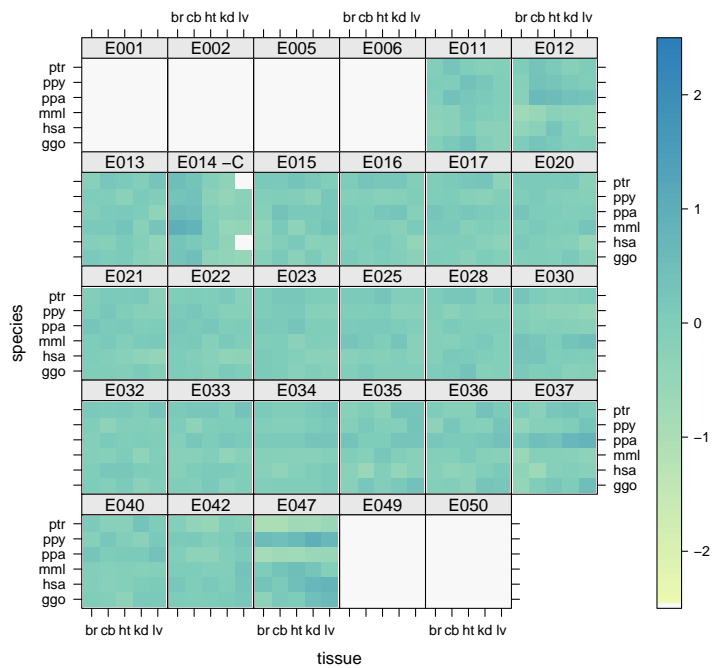


ENSG00000167995



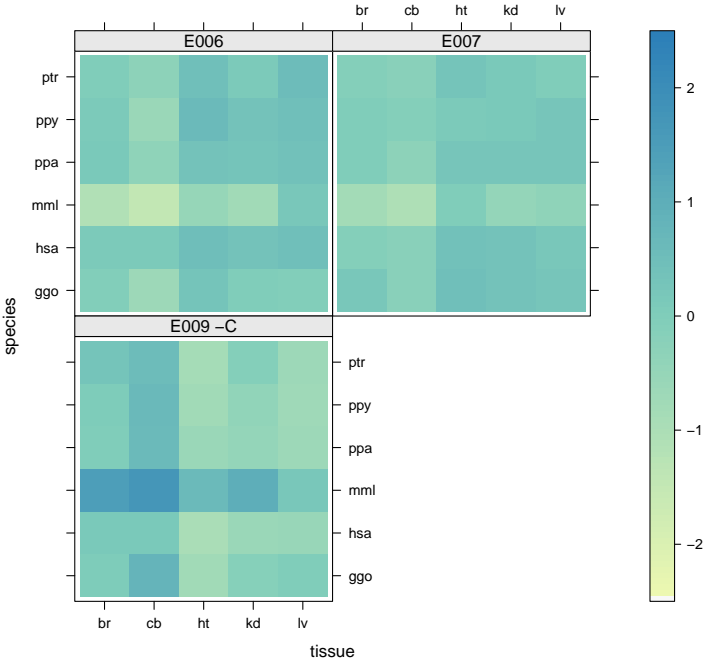


ENSG0000168036

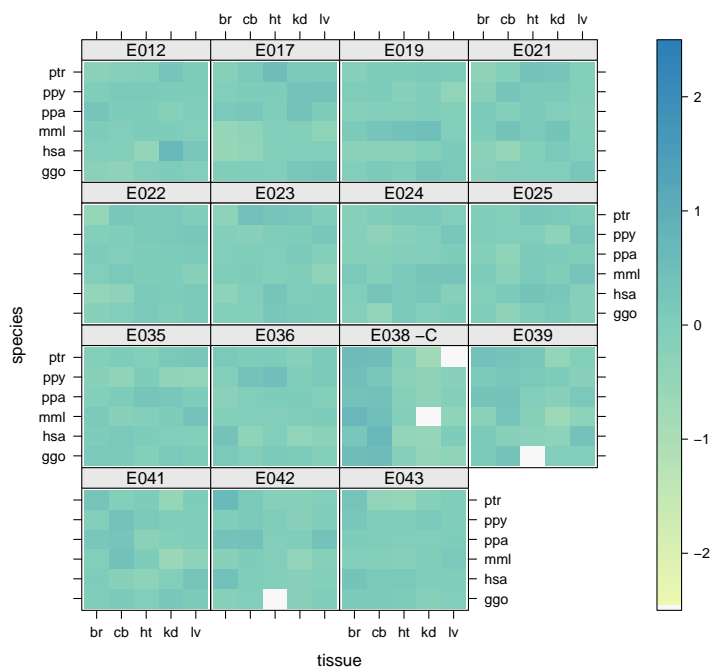




ENSG00000168273

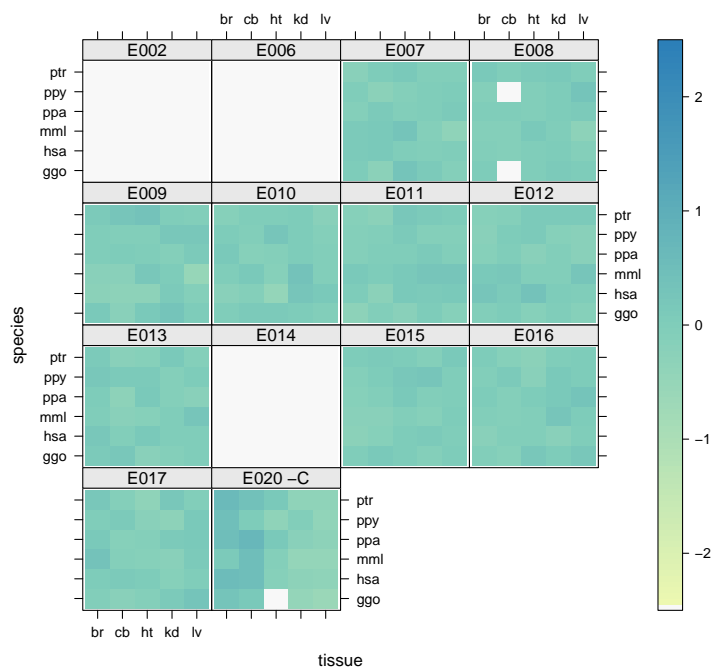


ENSG00000168297



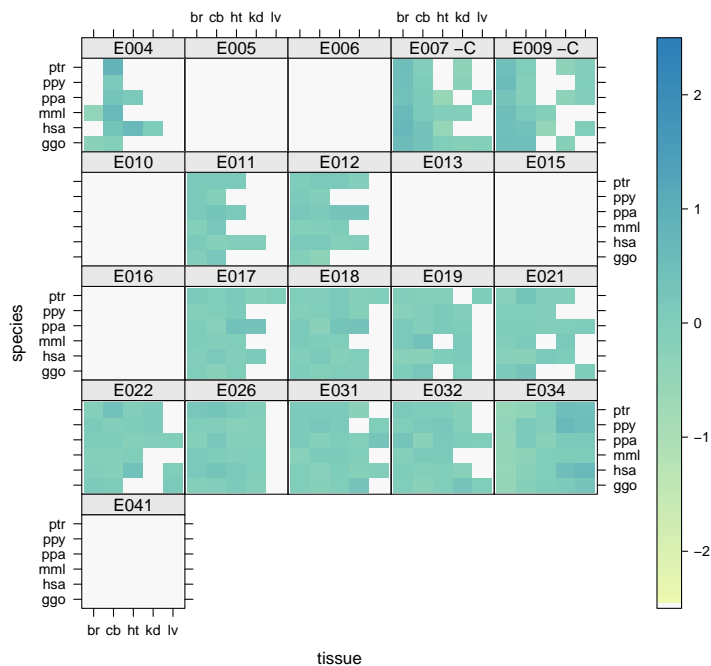


ENSG0000168904

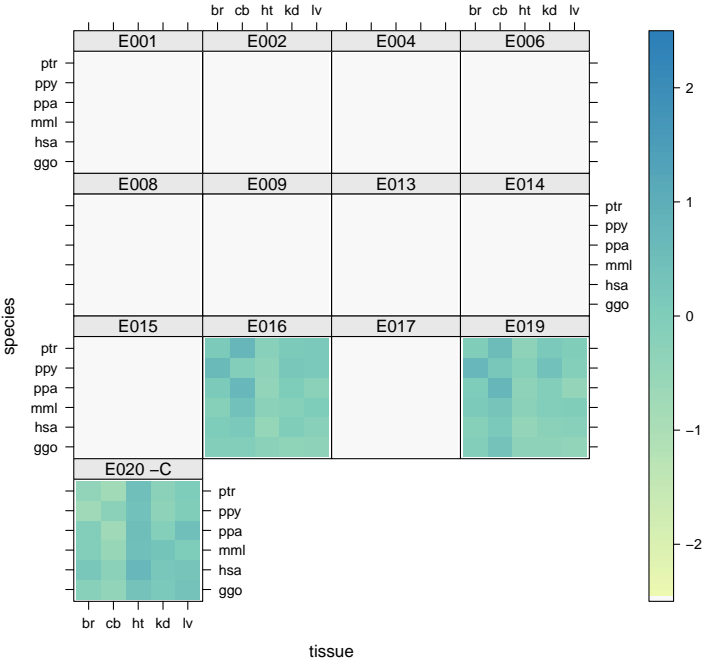




ENSG00000169282



ENSG00000169314

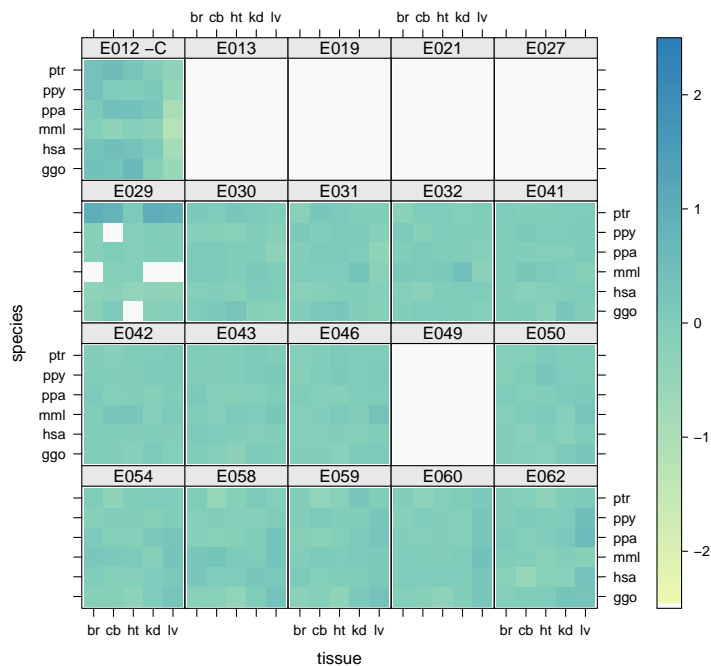








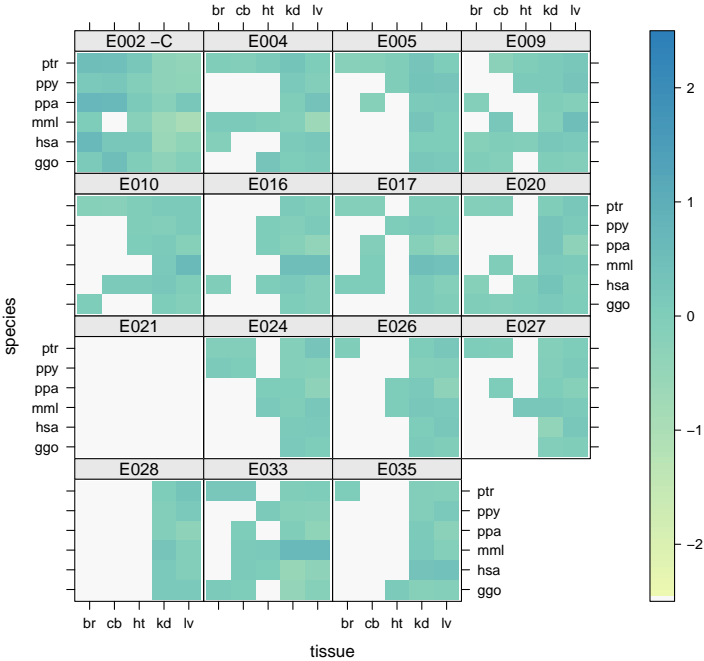
ENSG00000169764







ENSG00000170482



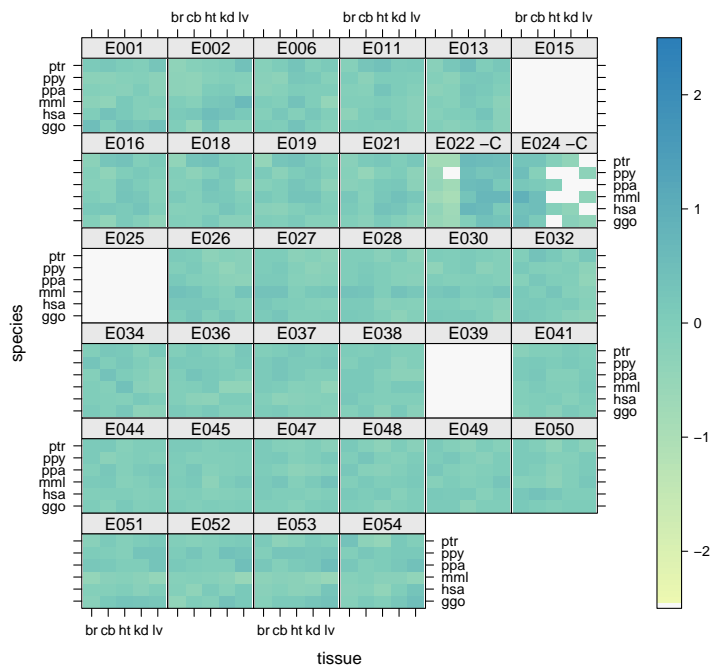






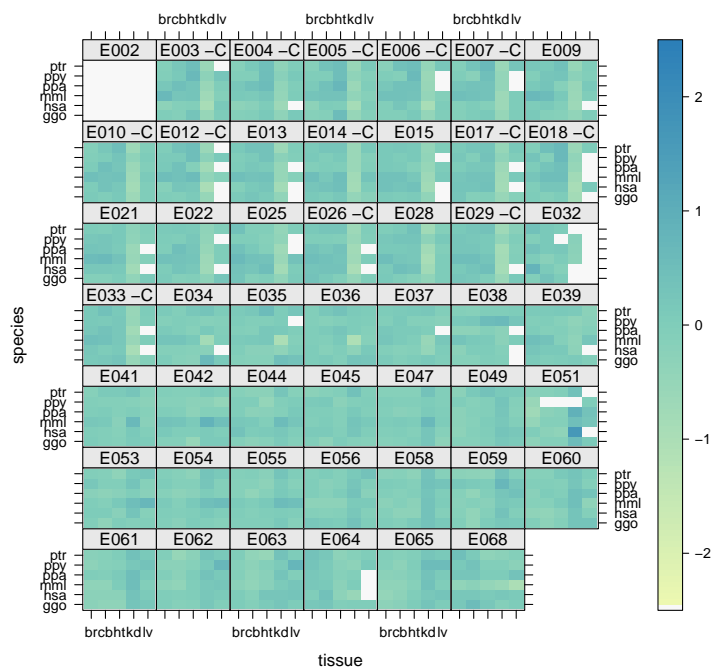


ENSG00000171723

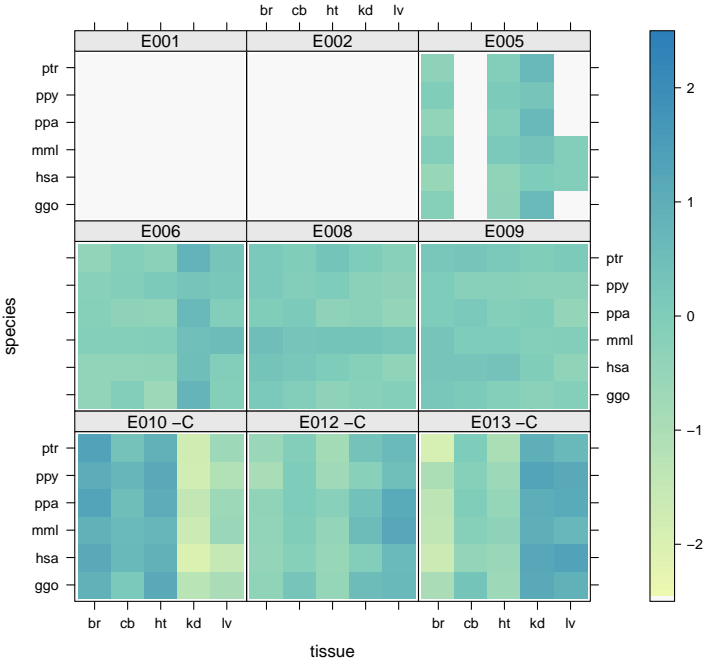




ENSG0000171914

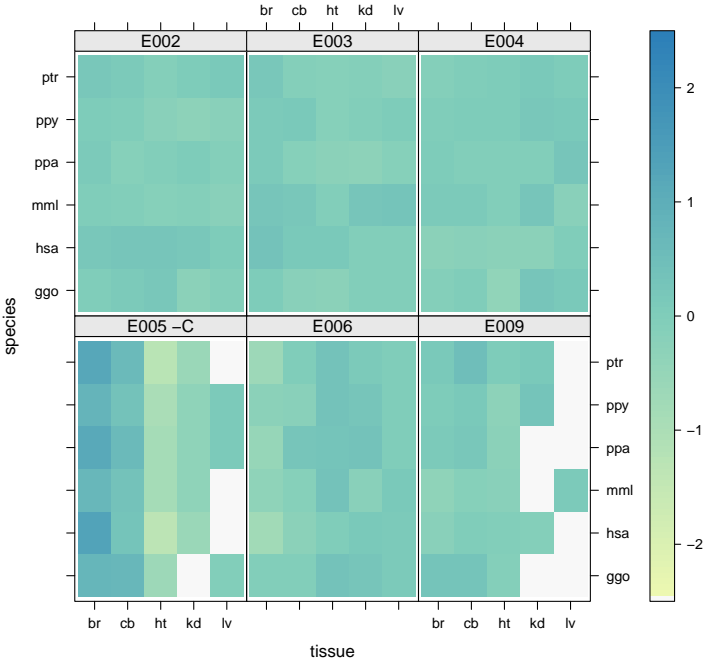


ENSG00000171992

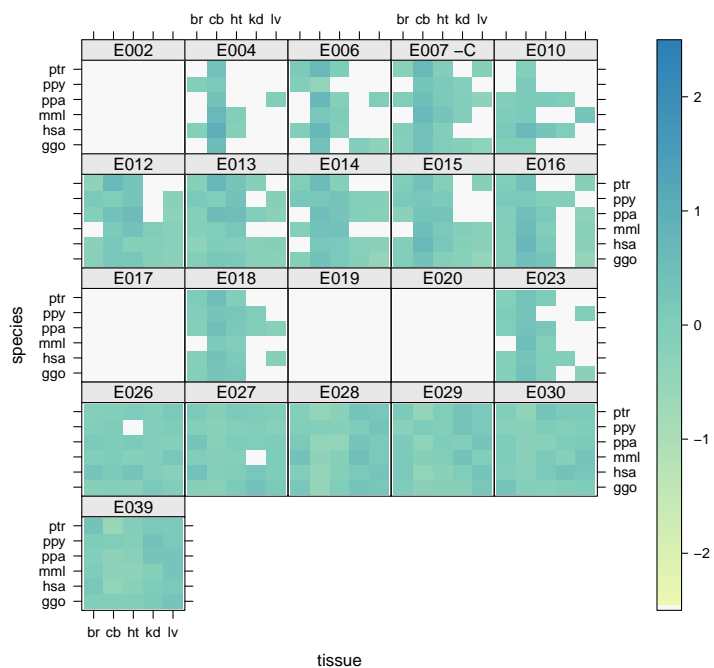




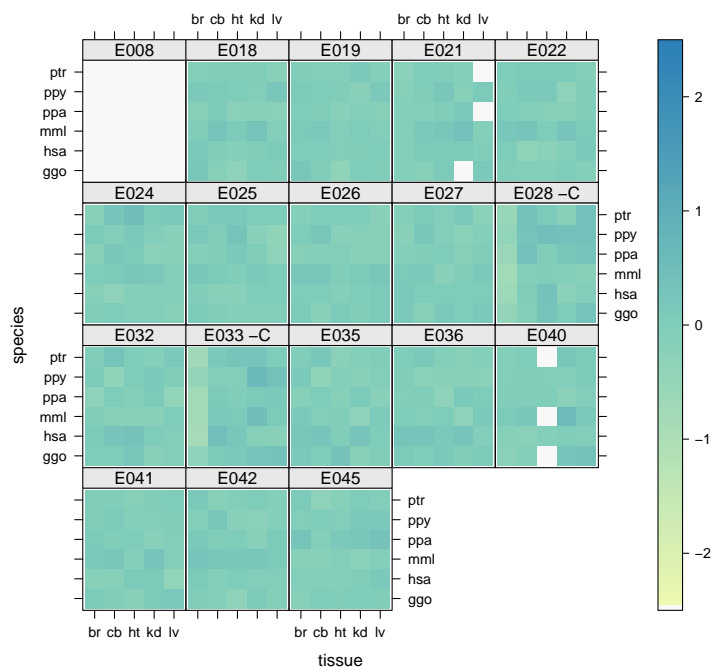
ENSG00000172348



ENSG00000172349

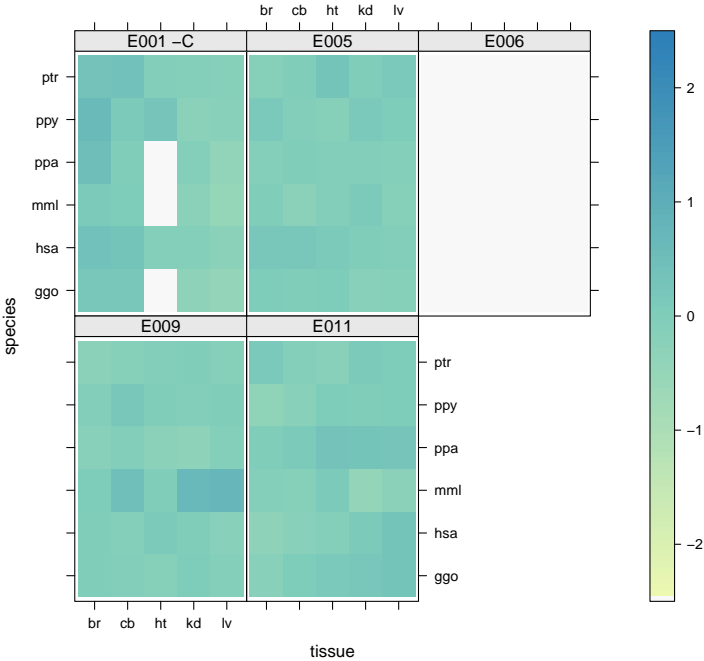


ENSG00000172375

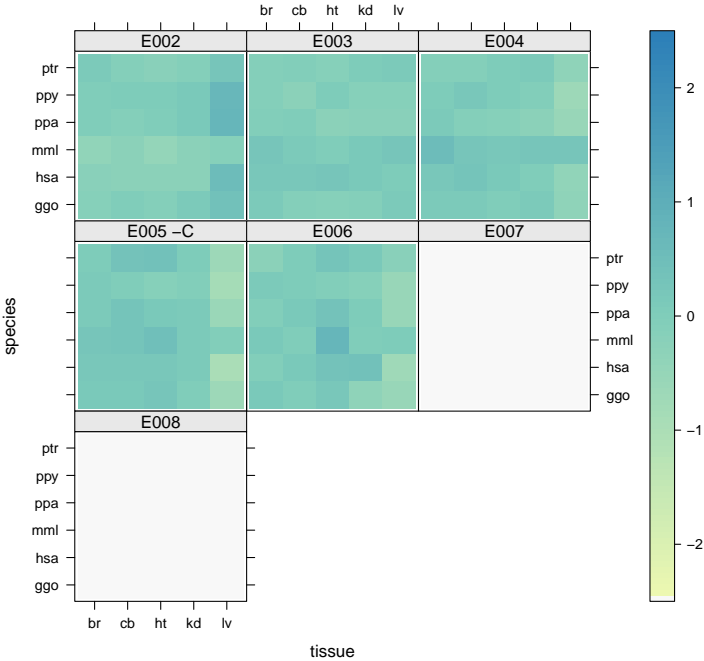




ENSG00000172594

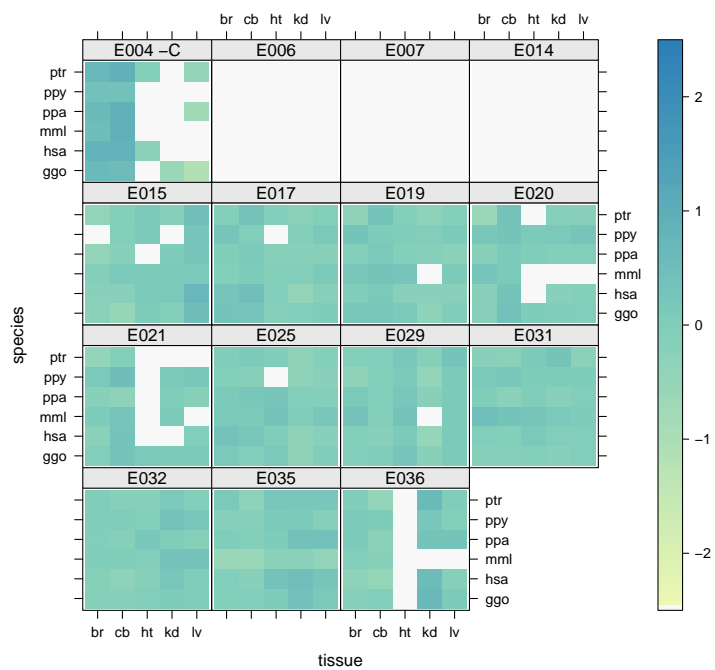


ENSG00000172731

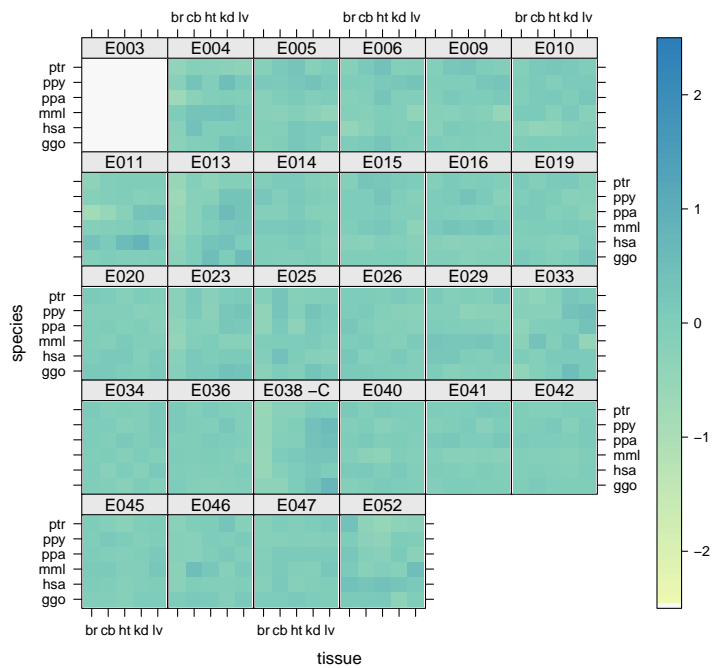




ENSG00000172794

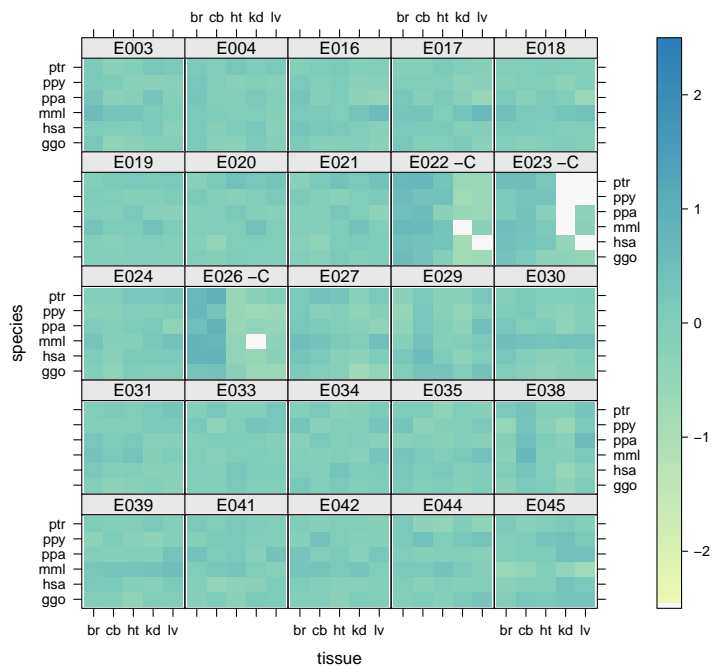


ENSG00000173020





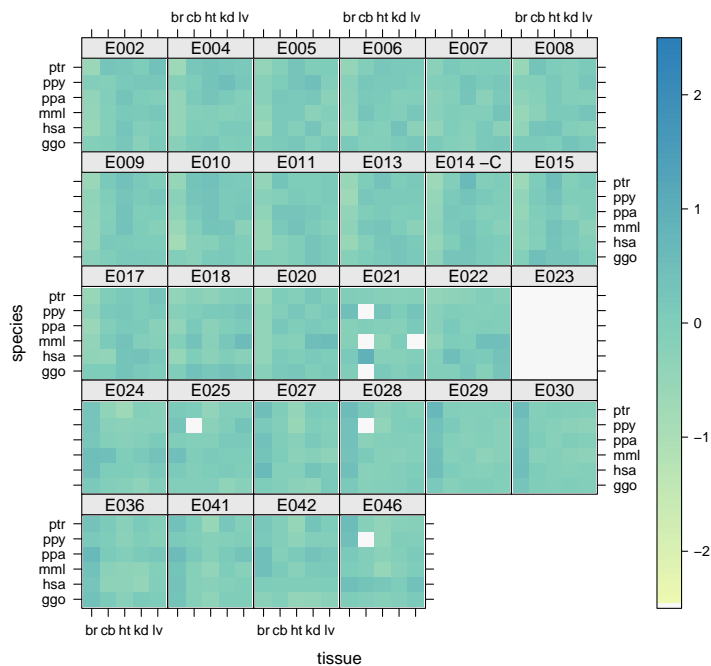
ENSG00000173210



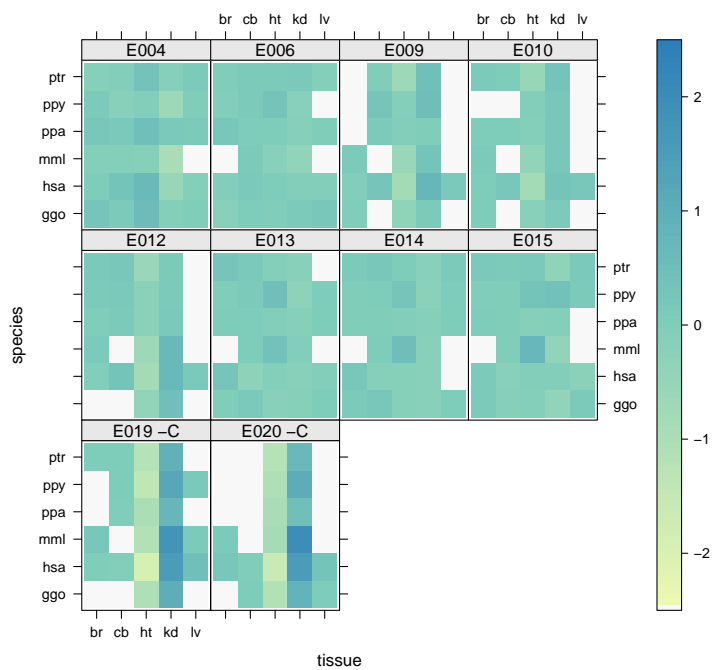




ENSG00000173442

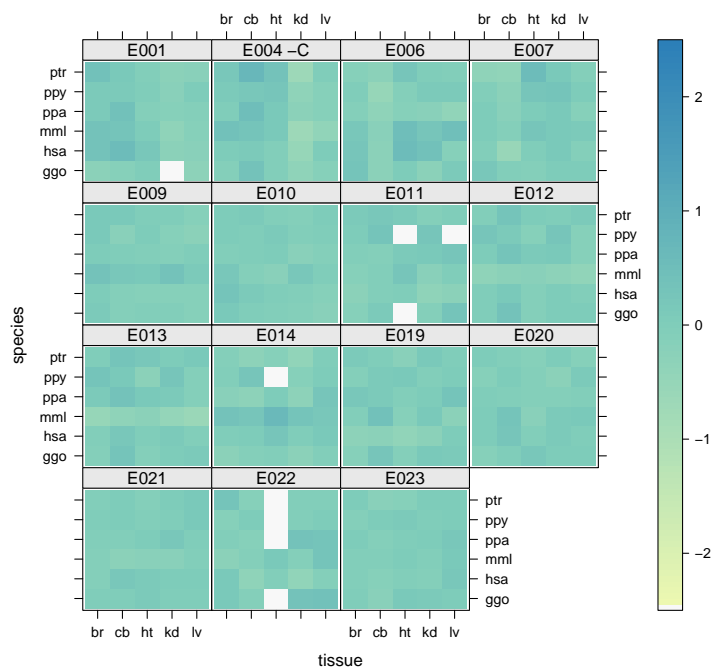


ENSG00000173641



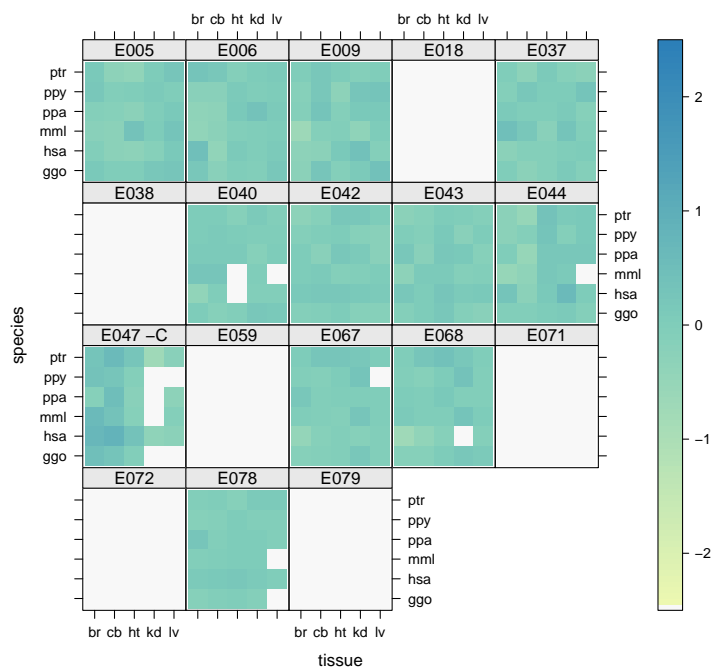


ENSG00000173848



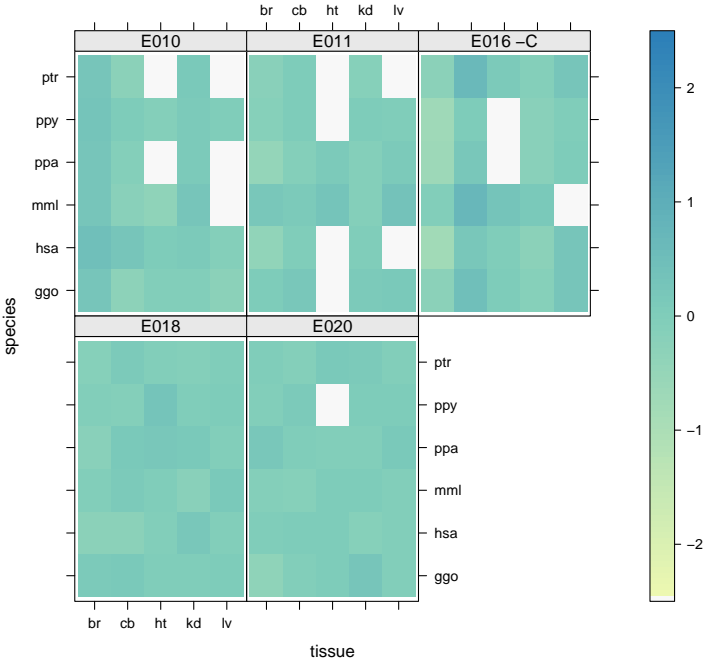


ENSG00000174373





ENSG00000174514



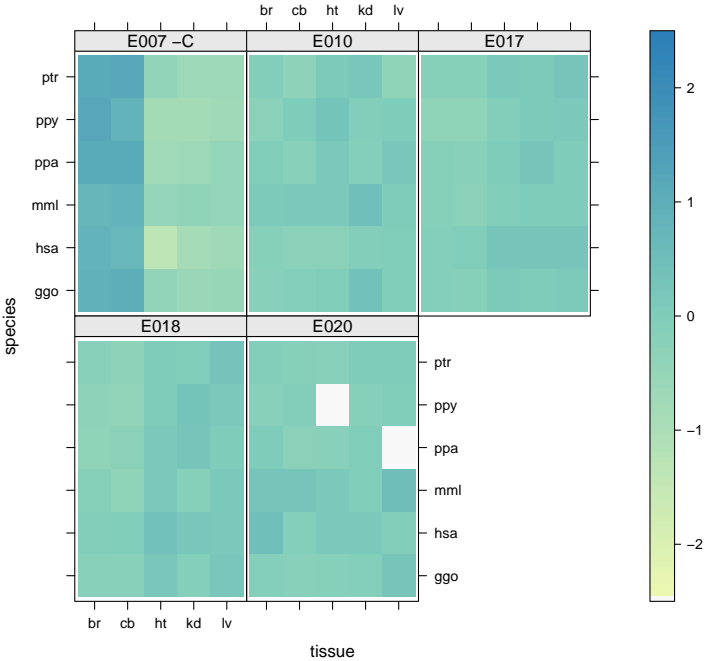








ENSG00000175416









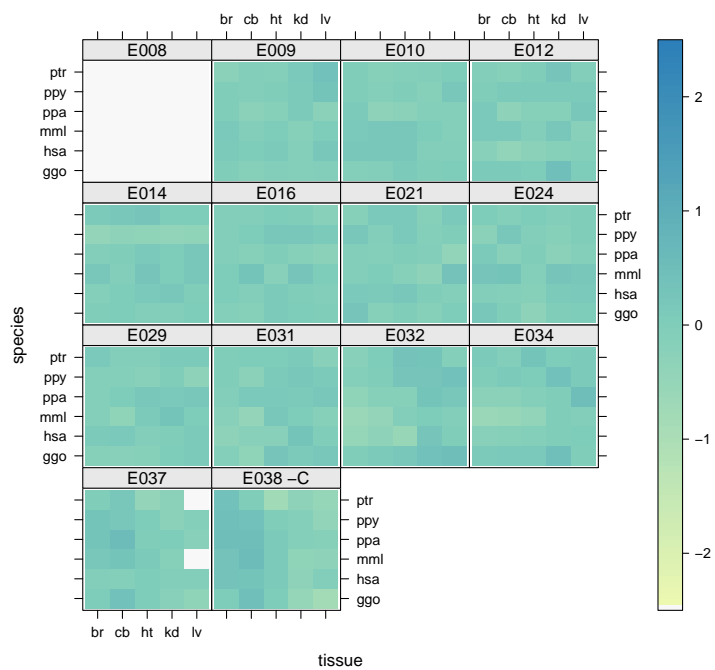




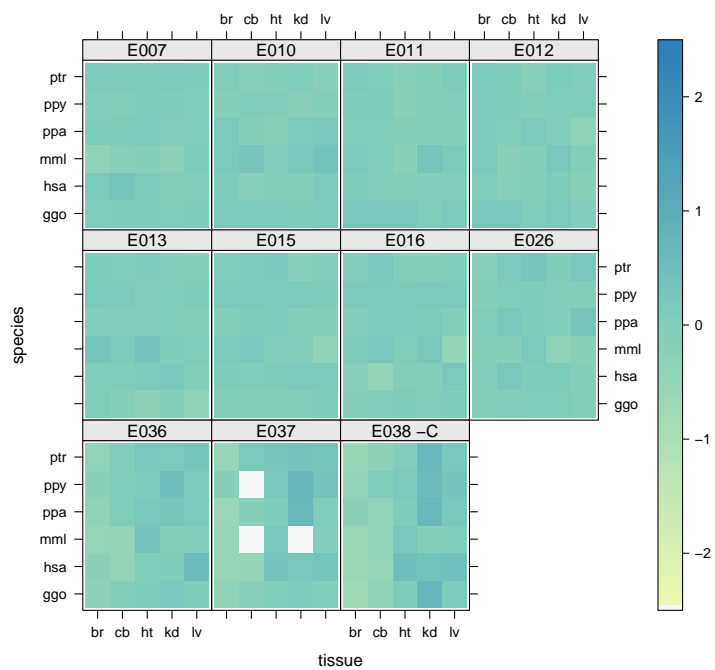




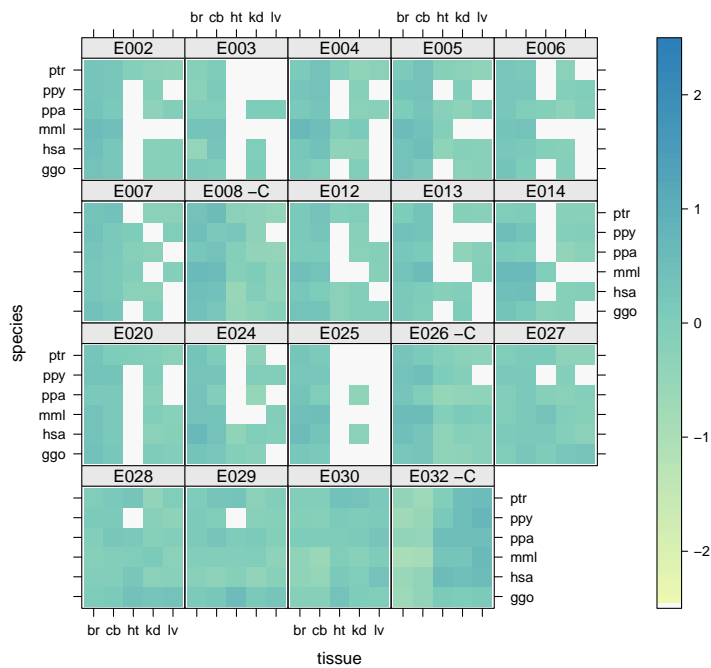
ENSG00000176463



ENSG00000177030



ENSG00000177380



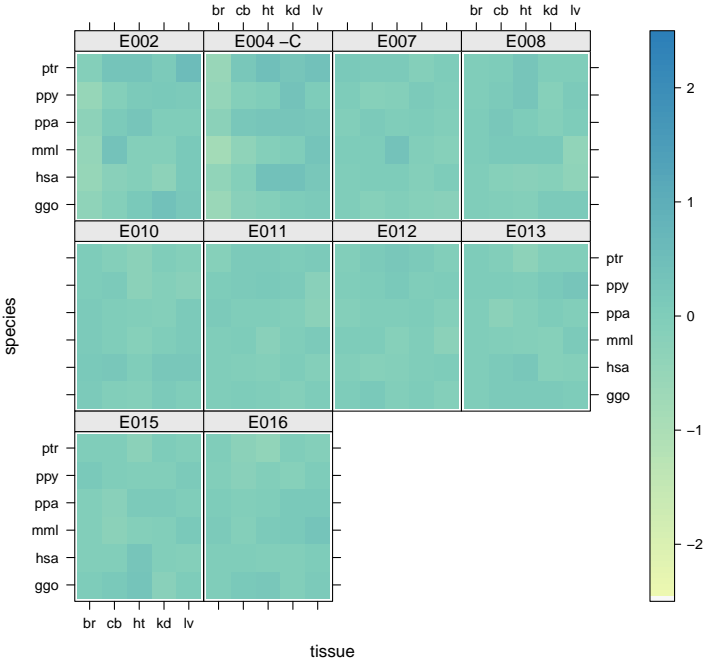








ENSG00000179134

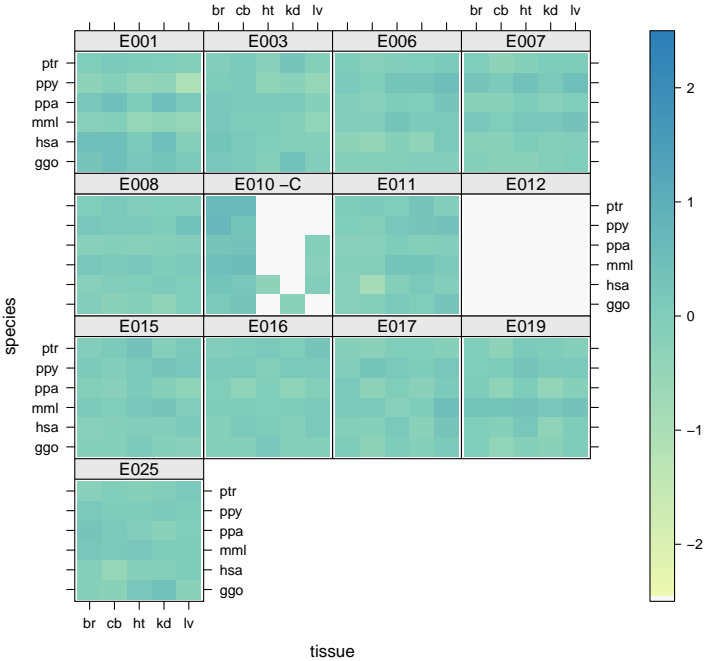




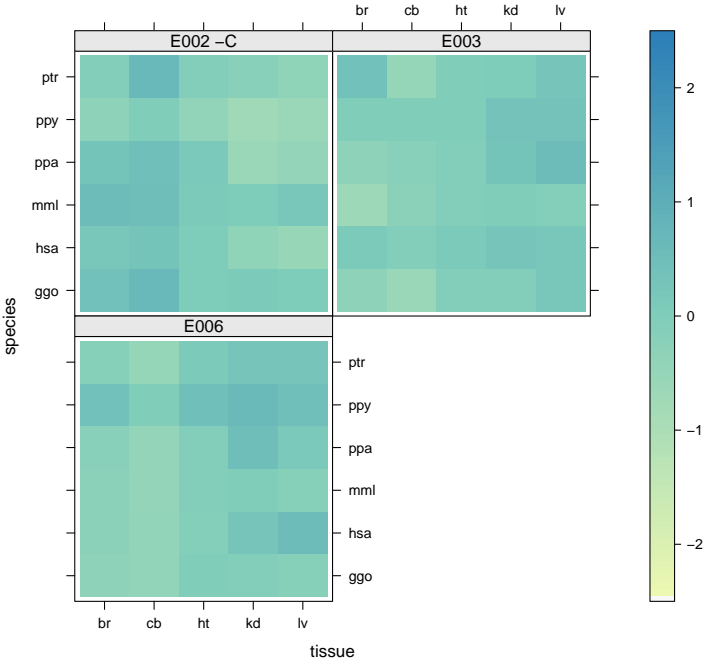




ENSG00000181982

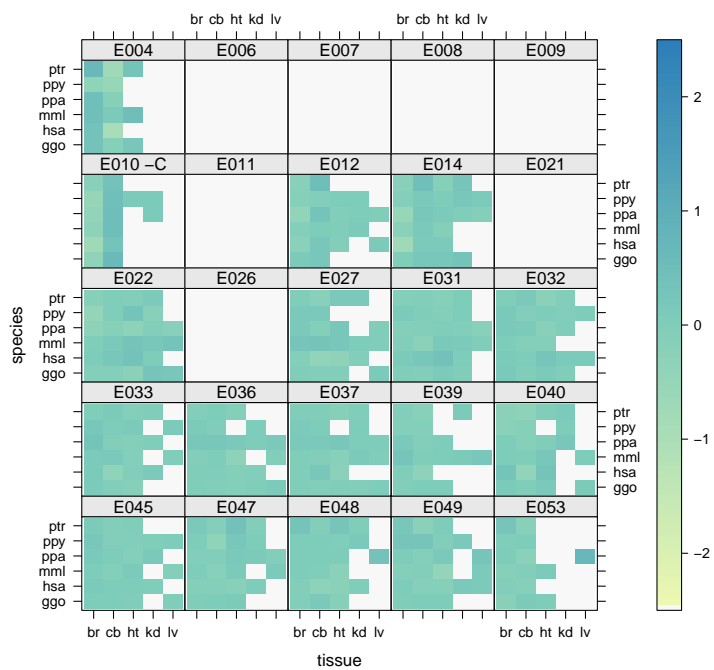


ENSG00000182359



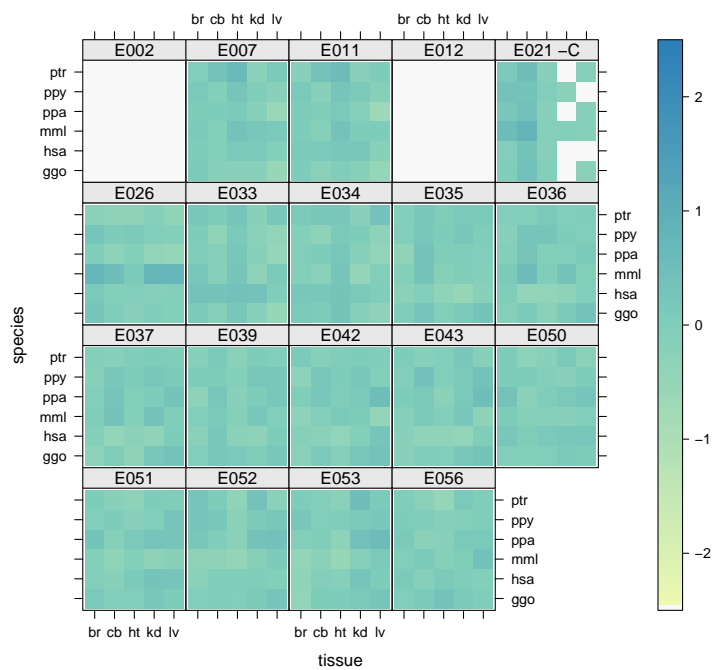


ENSG00000182667

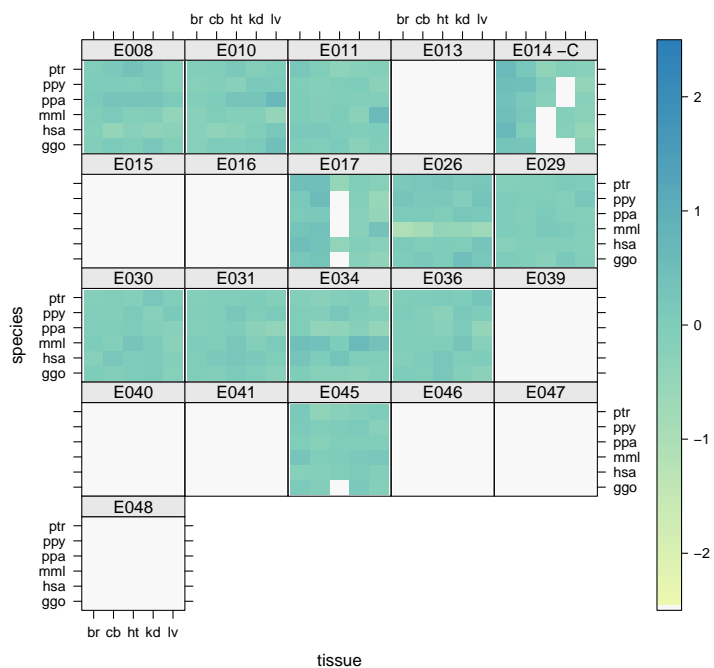




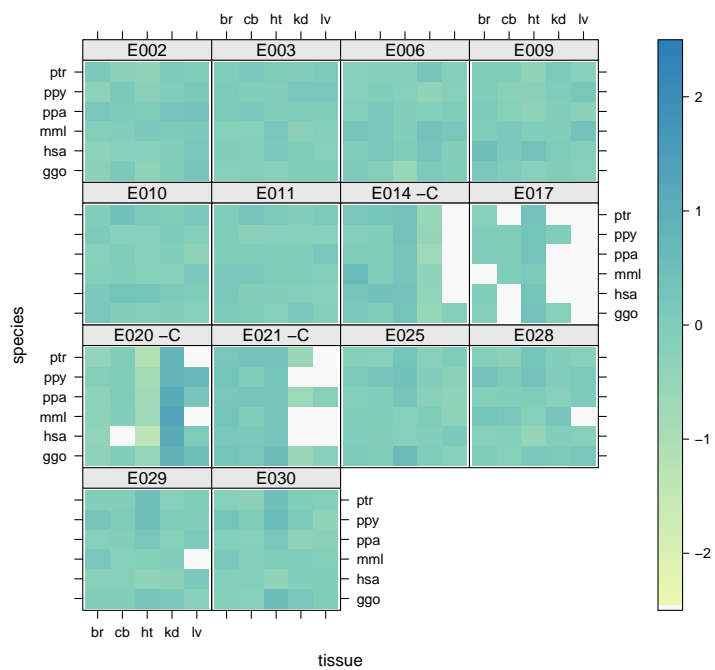
ENSG00000182944



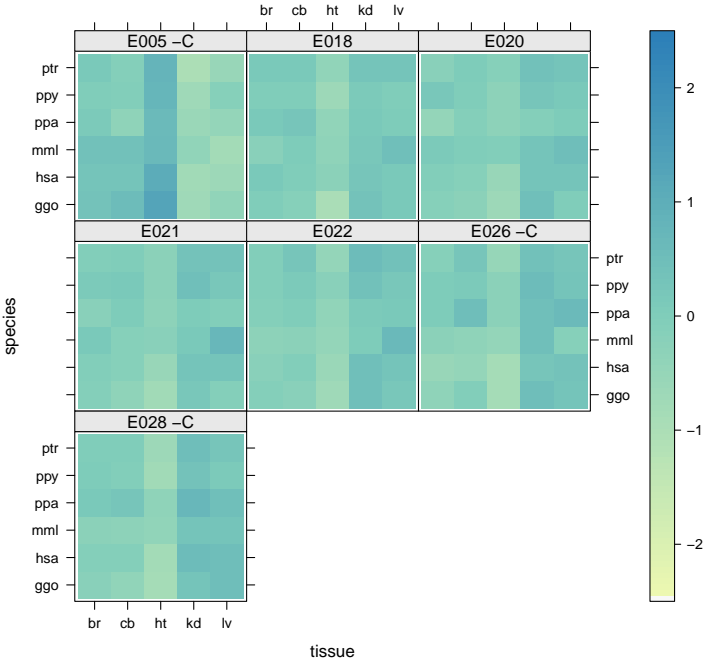
ENSG00000182985



ENSG0000183023

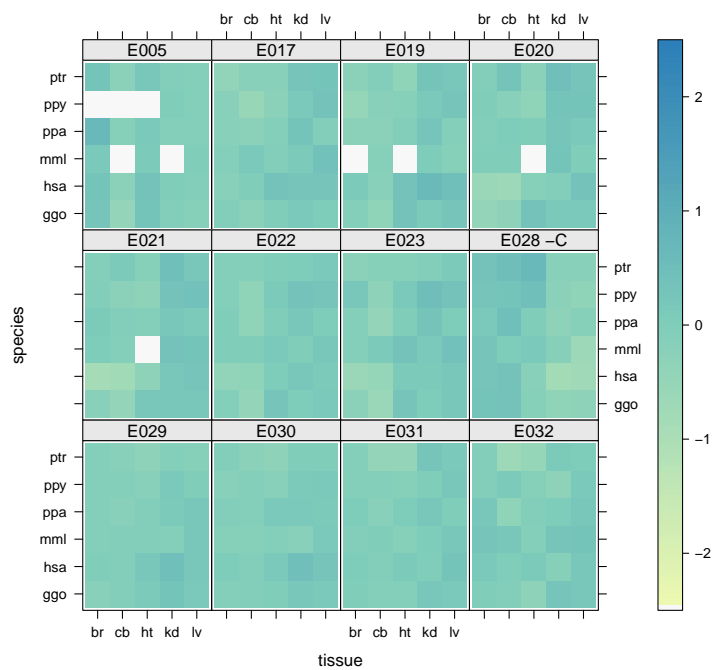


ENSG00000183048

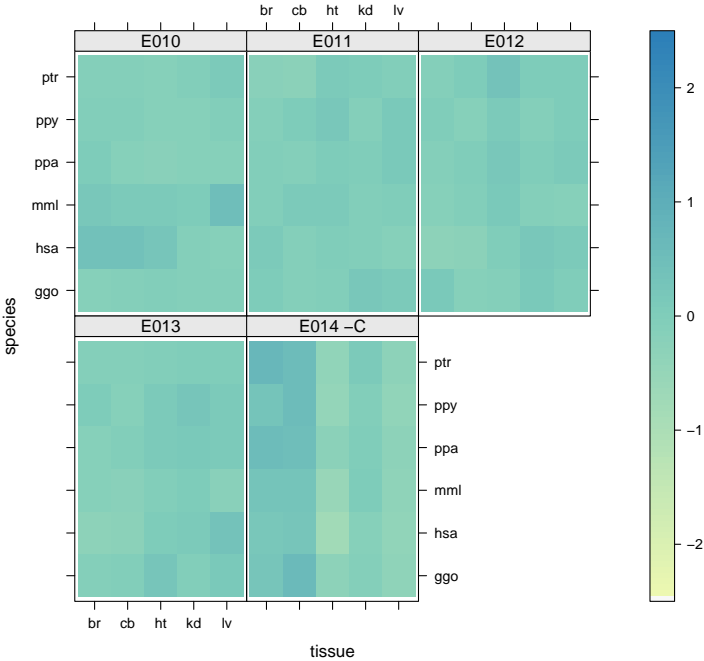




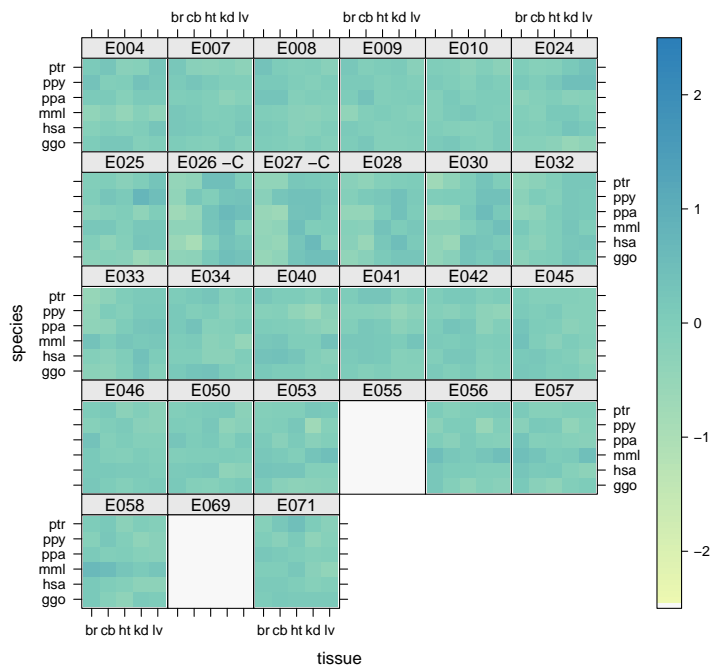
ENSG00000183696



ENSG00000183726

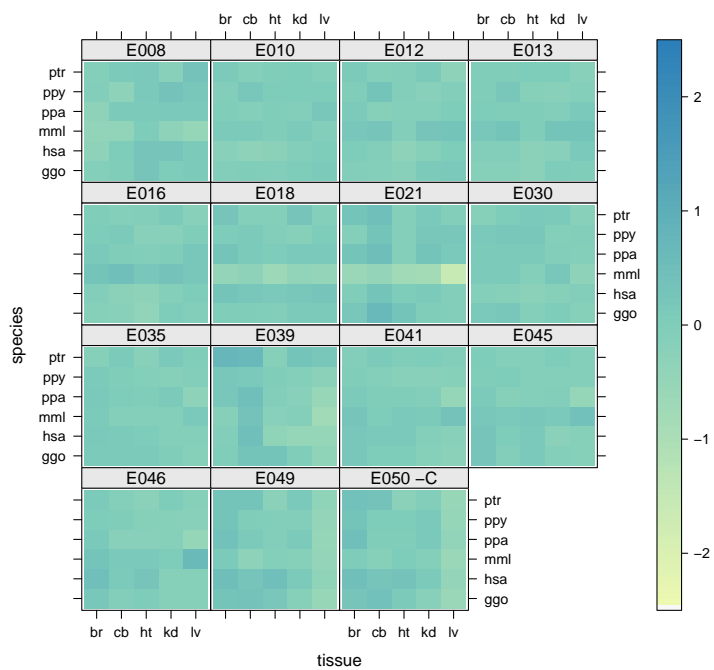


ENSG00000184381

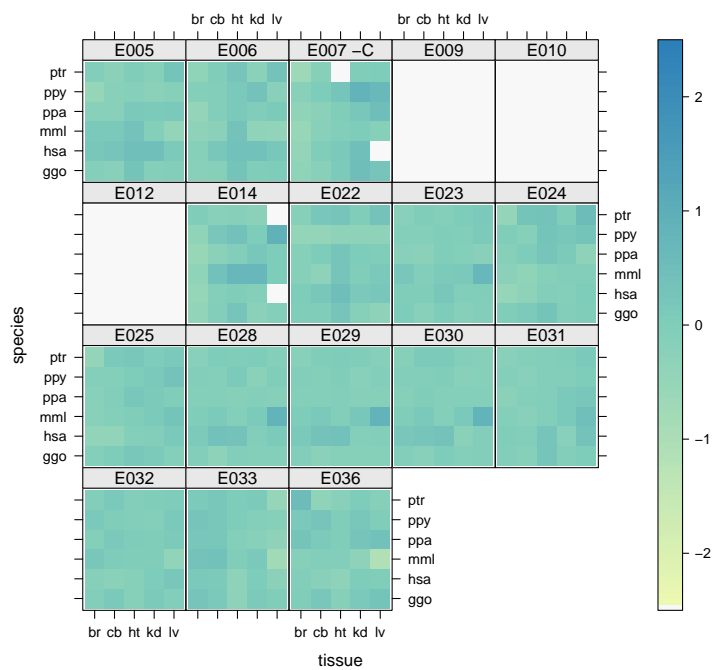




ENSG0000185624

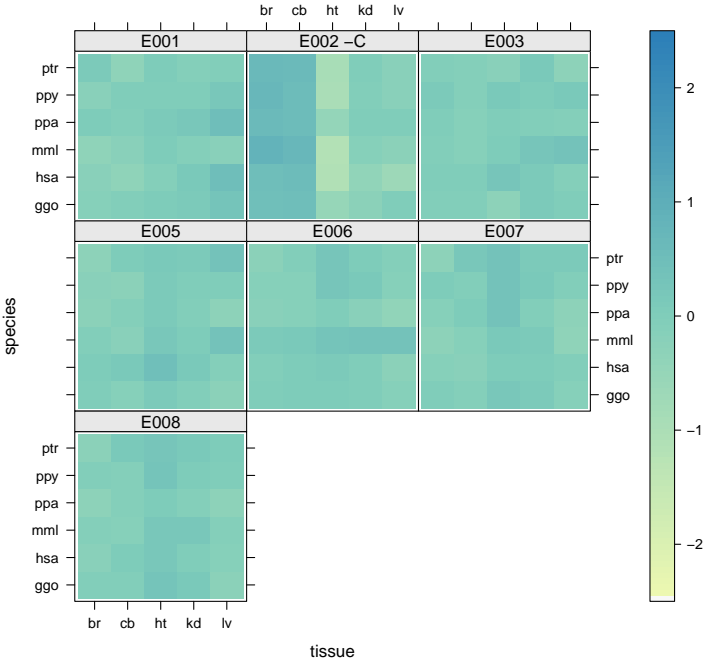


ENSG00000185630

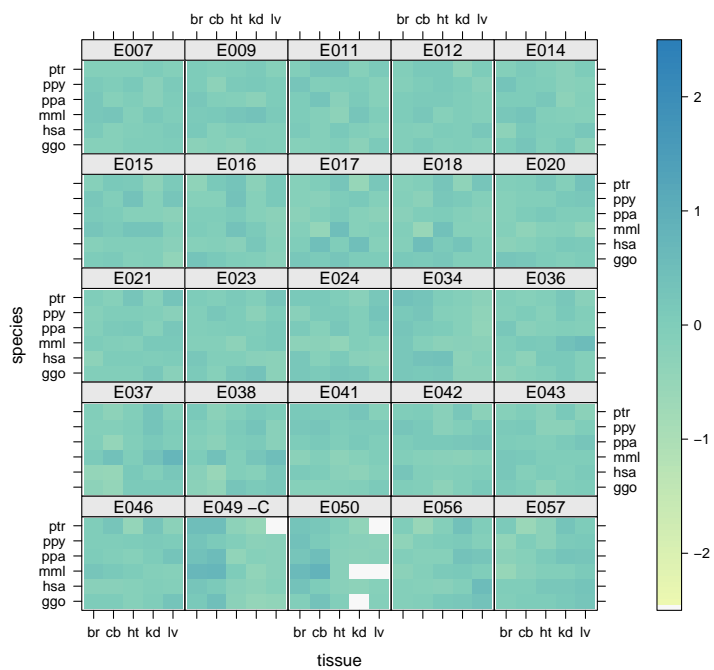




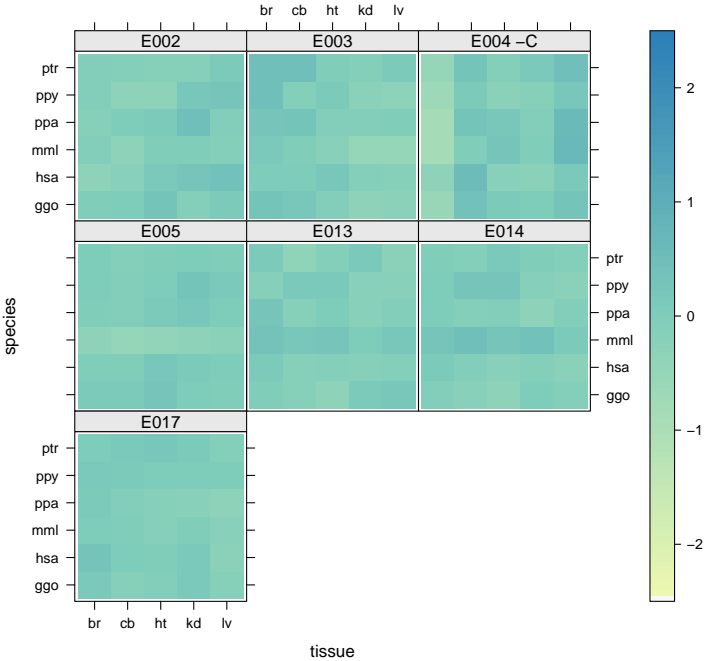
ENSG00000185963



ENSG00000186001

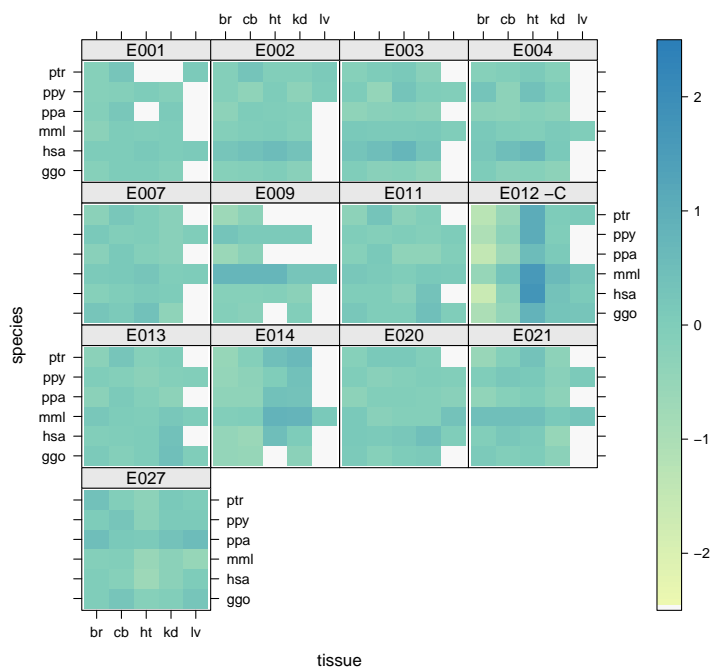


ENSG00000186111





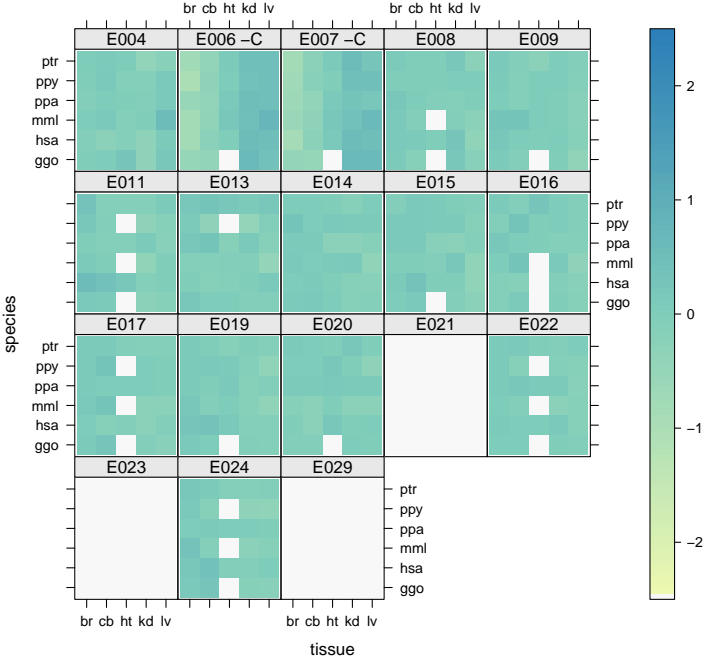
ENSG00000186868







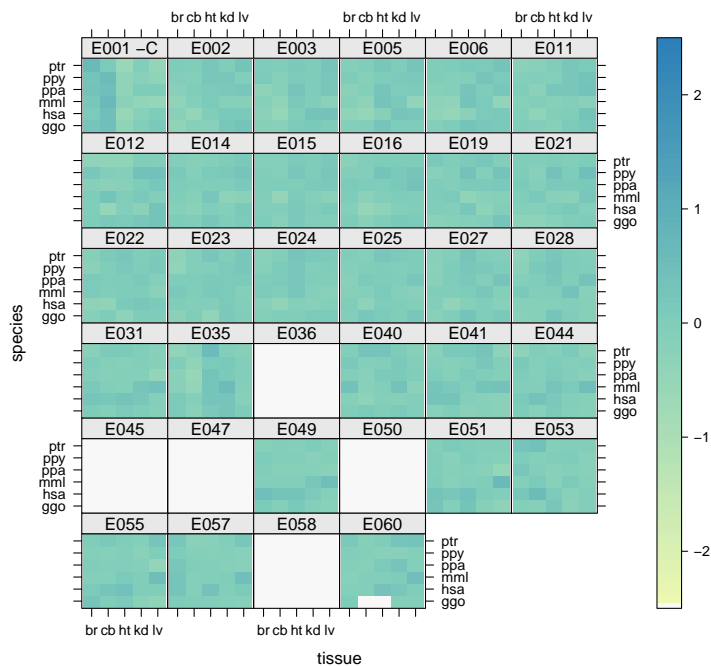
ENSG00000187164



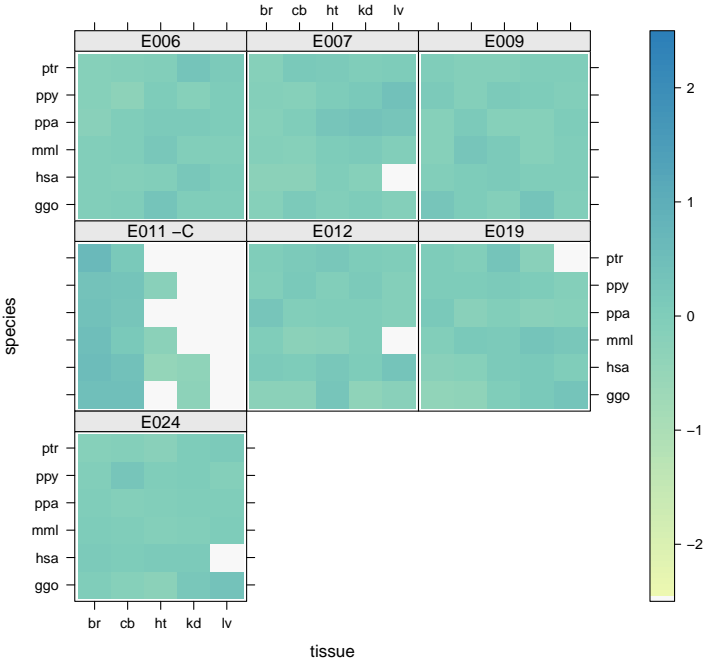




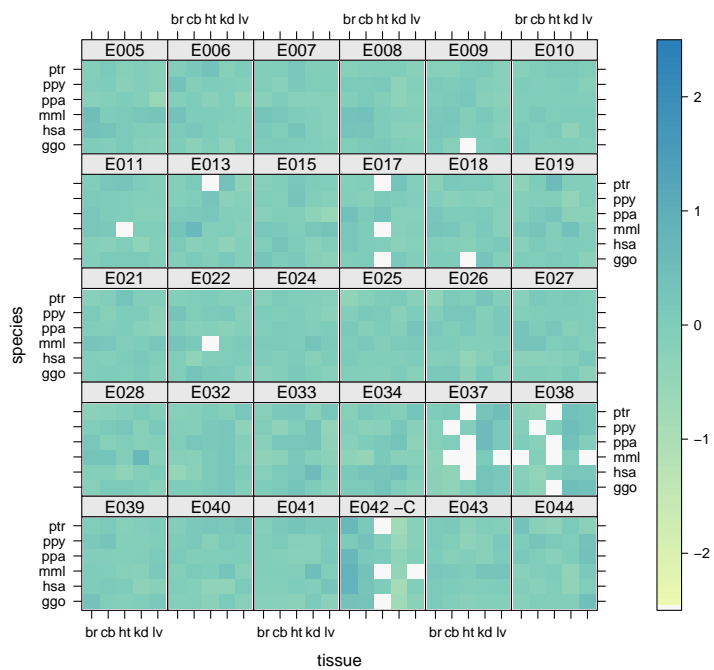
ENSG0000196504



ENSG00000196526



ENSG0000196586

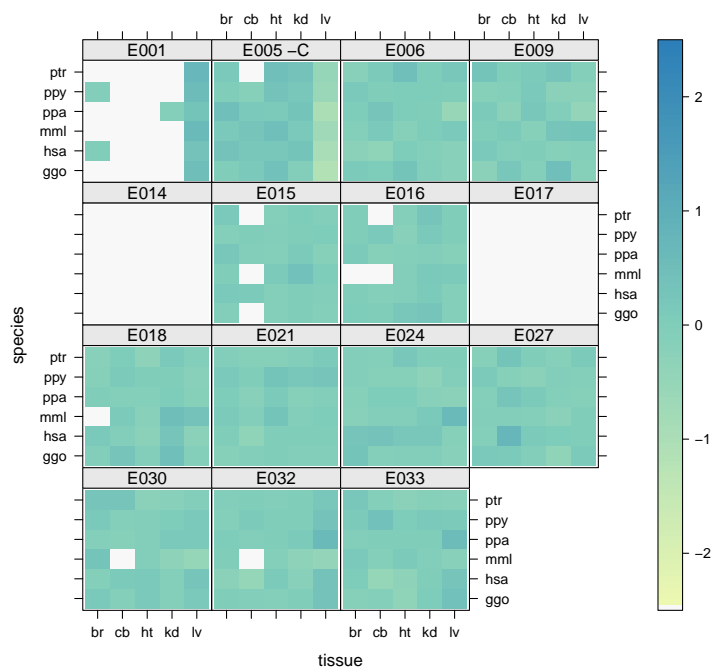




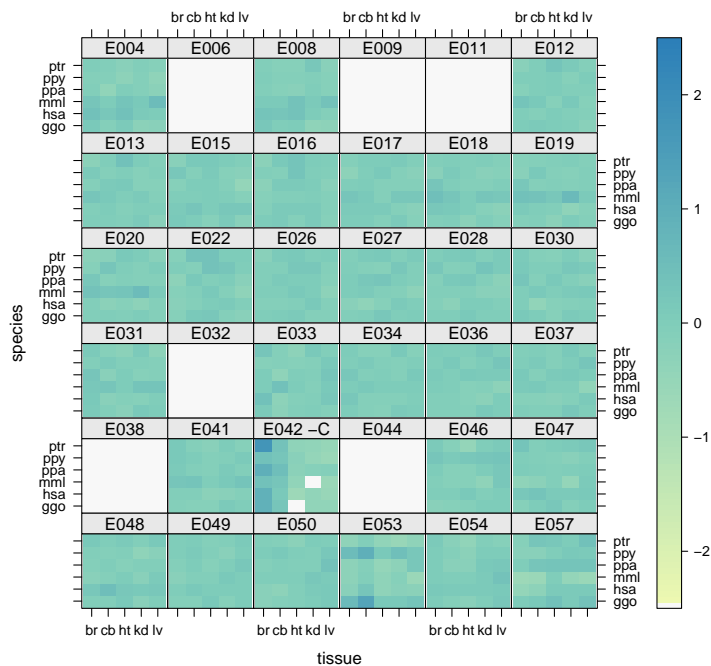




ENSG00000197142

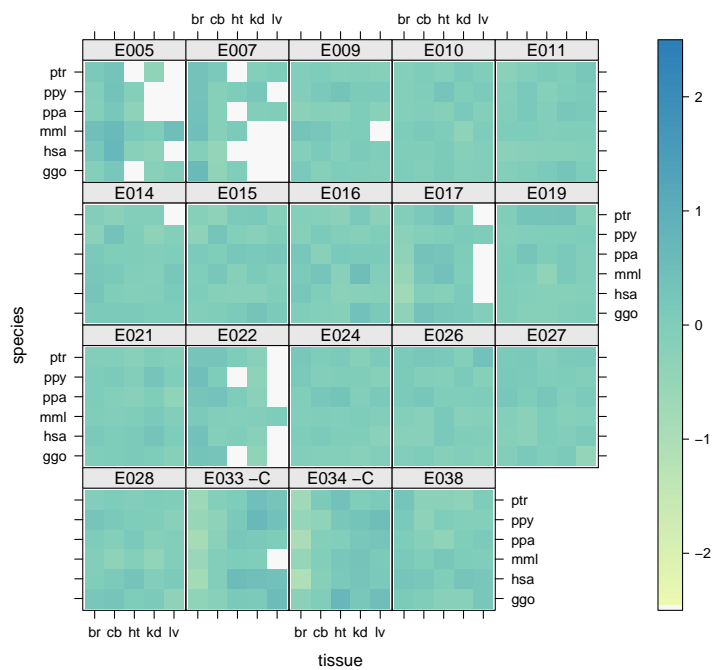


ENSG00000197157

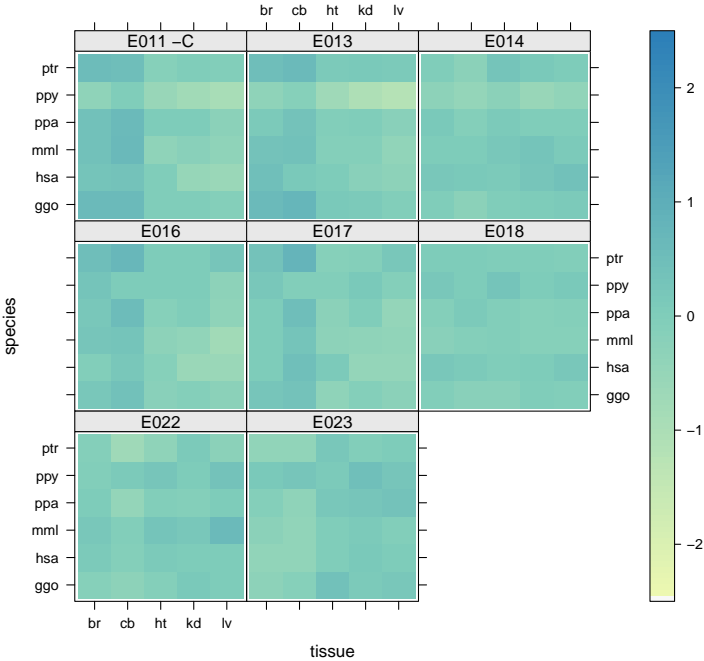




ENSG00000197283

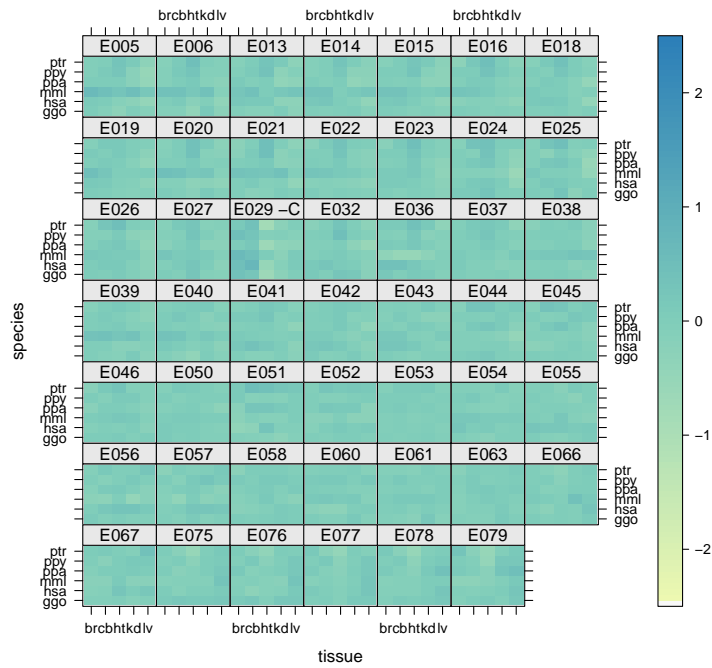


ENSG00000197448



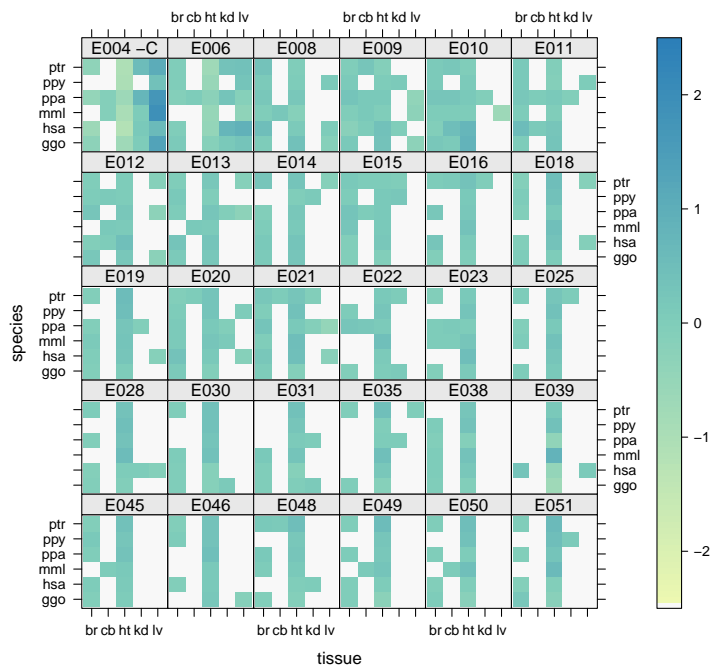


ENSG0000197694

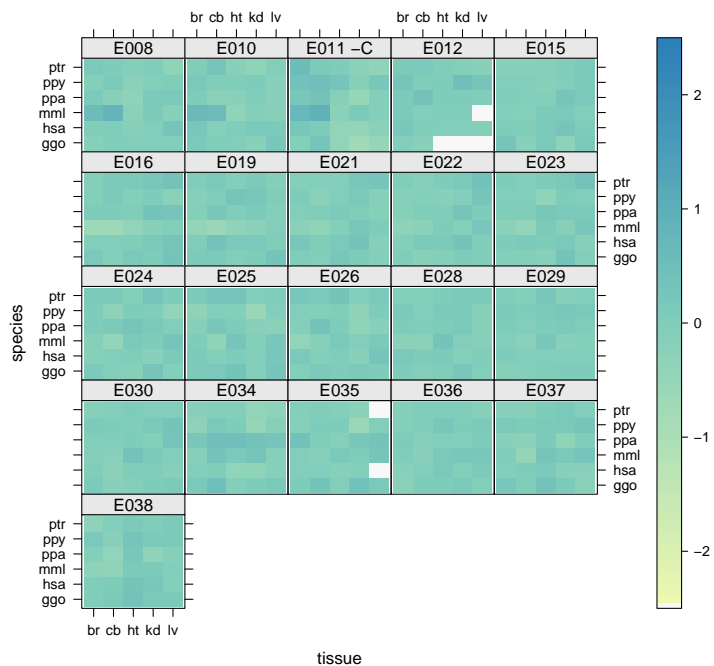




ENSG00000197893

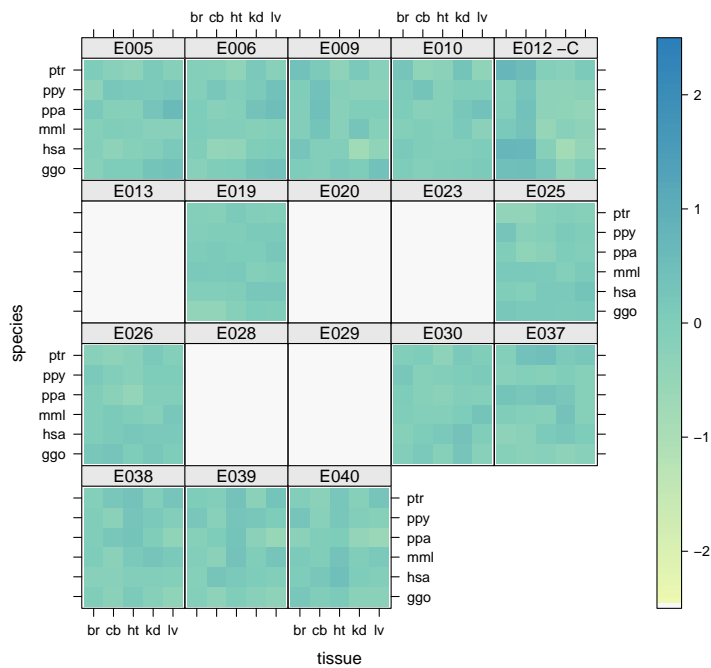


ENSG00000197948

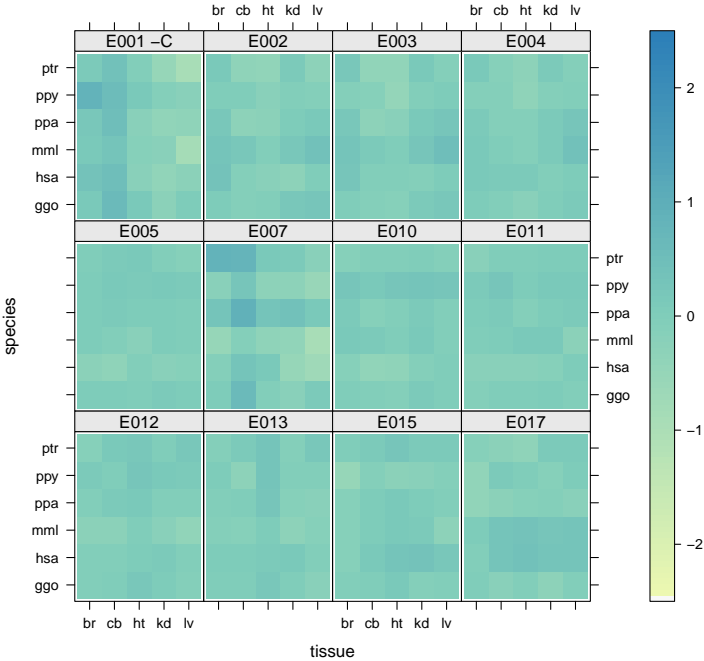




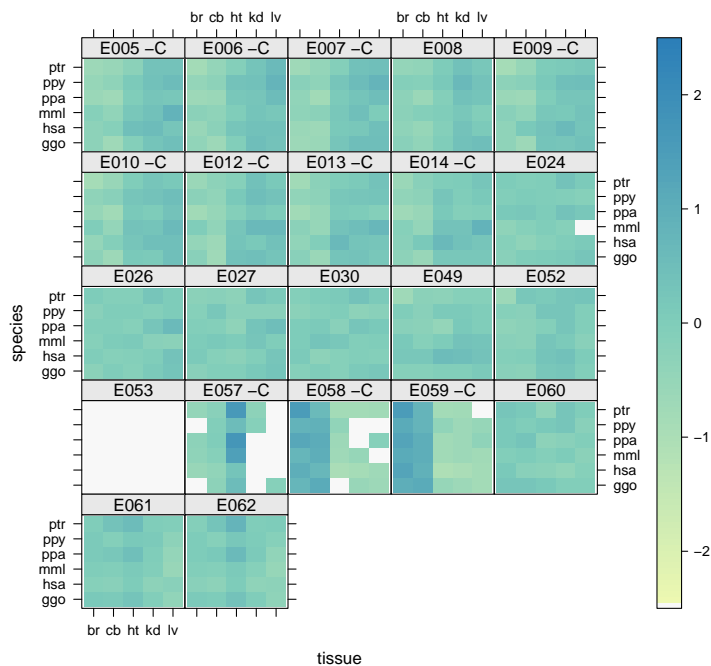
ENSG00000198130



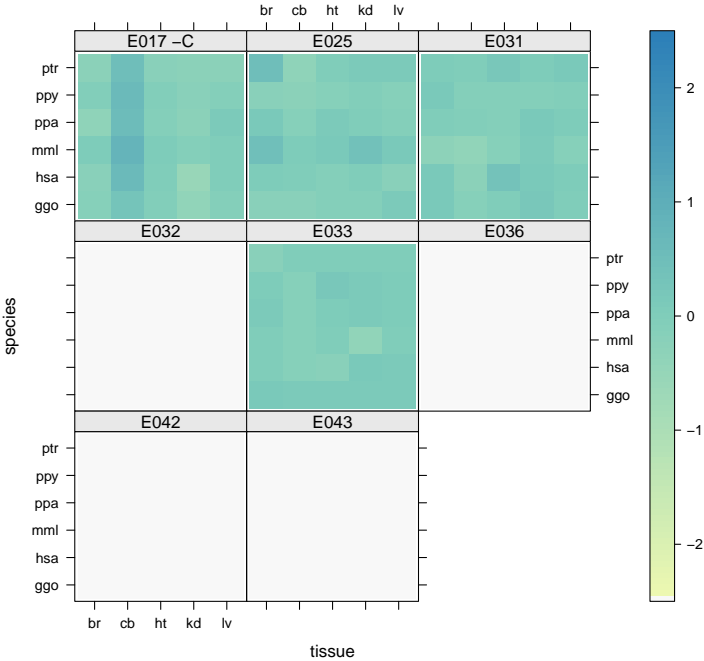
ENSG00000198171



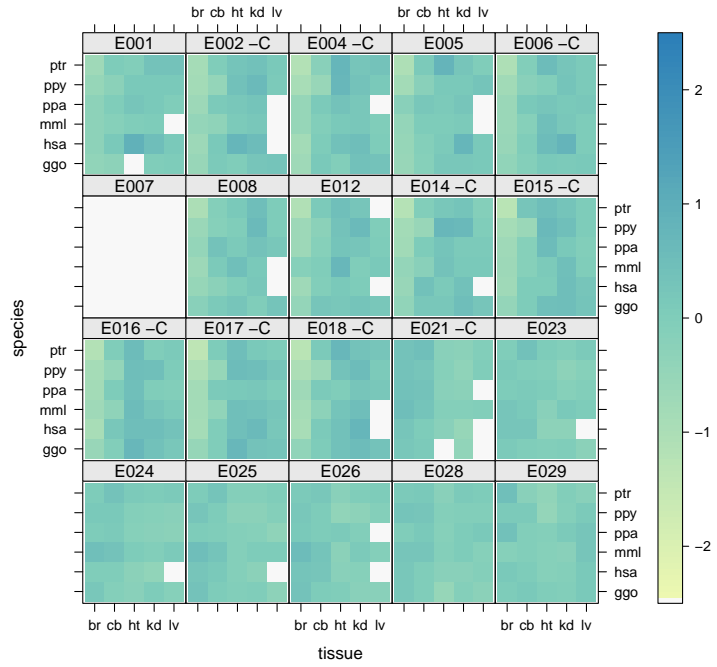
ENSG00000198363



ENSG00000198586



ENSG00000198825

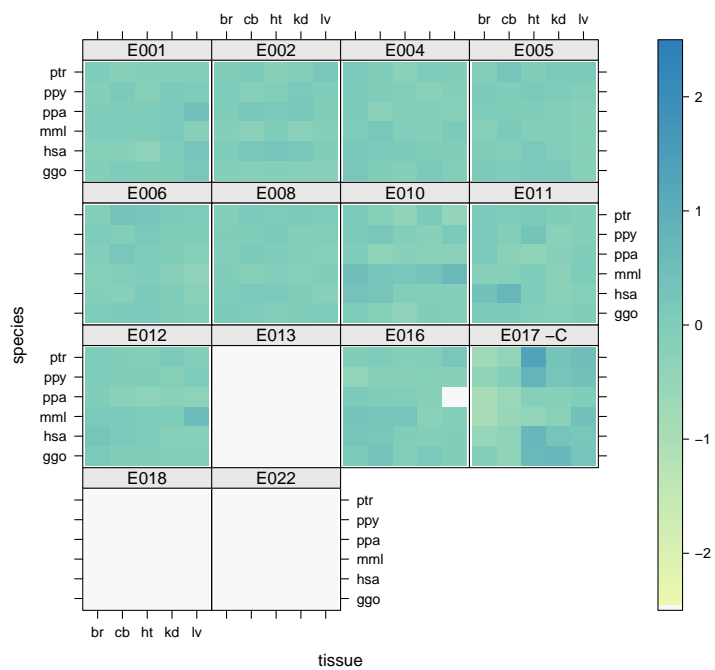




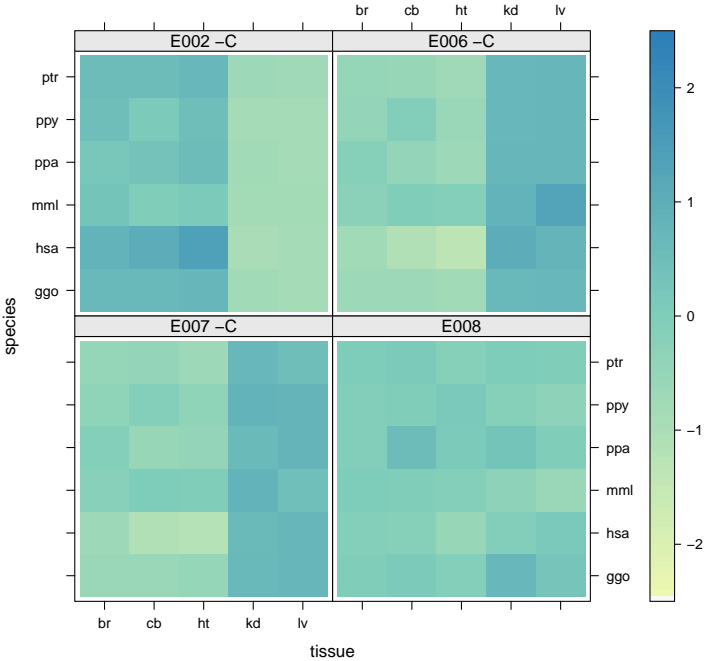




ENSG00000204310

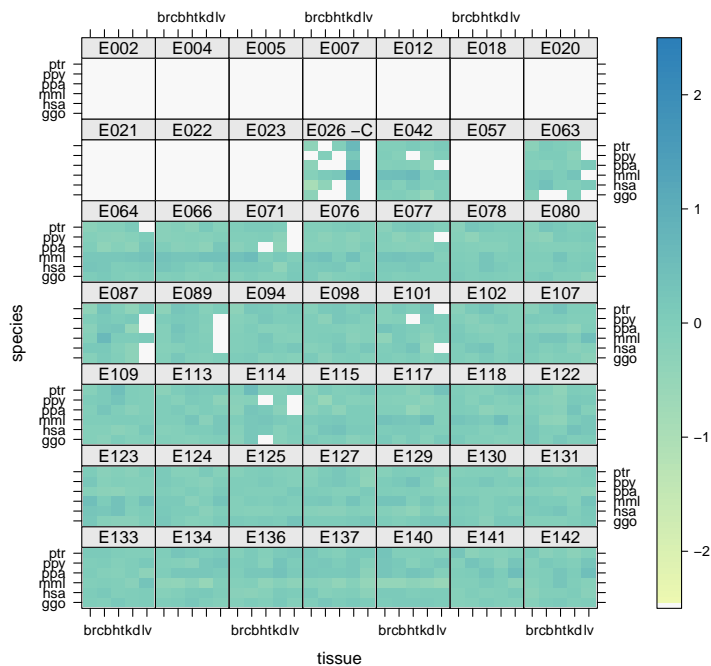


ENSG00000204444





ENSG0000204580

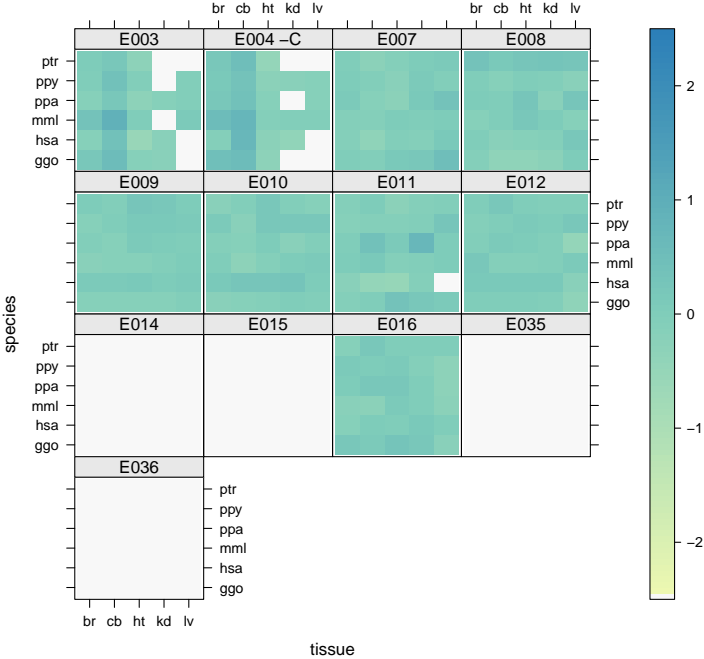






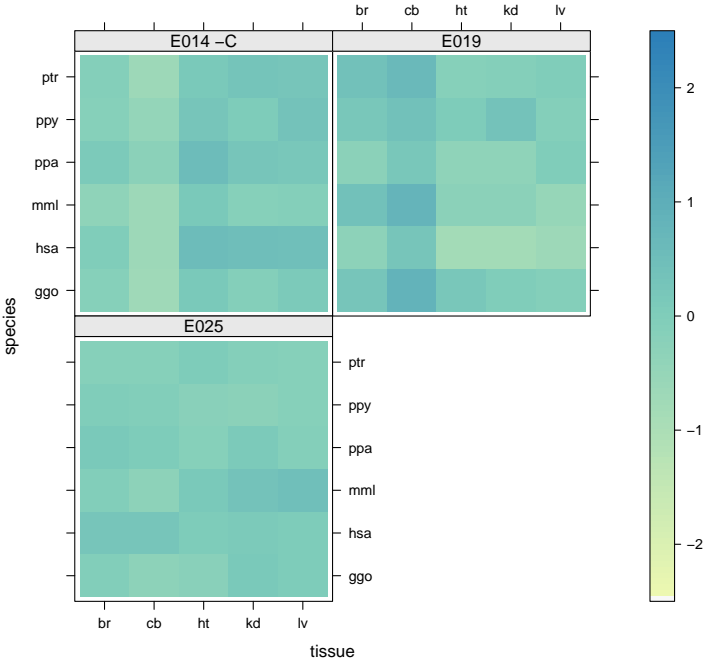


ENSG00000205683

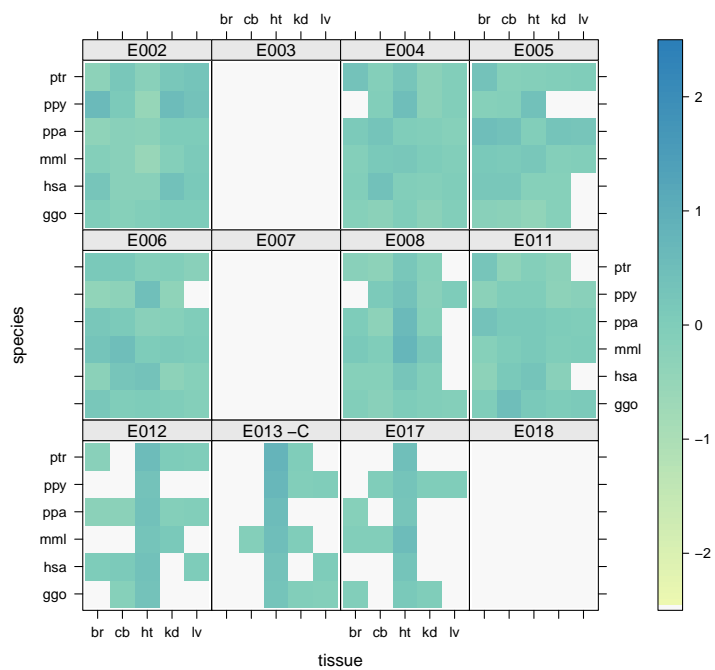




ENSG00000214063



ENSG00000239388



ENSG00000241878

