Methods:

Sequencing runs:

ONT 1D R9 runs were carried out in the lab:

Species name	Sample name	Total sequences	Total size (bp)	Longest sequence	Genome size (approx, pg 2C)
Sorbus aria	Database: SCI-FEST_B_20160729_Sorbus-aria_metrichor112154.	17,928	23,196,528	66,551	0.71
Nepenthes alata	Database: SCI-FEST-A_20160729_Napenthes-alata.	51,608	68,411,676	95,184	0.28 (N.pervillei)
Silene uniflora	Database: SCI-FEST-C_20160802_Silene_1D_sf.	22,923	36,943,694	23,031	2.7 (S.latifolia)
Erycina echinata	Database: SCI-FEST-C_20160802_Erycina.	57,699	71,713,785	144,298	1.9 (E.diaphena)
Beta patula	Database: SCI-FEST-A_20160802_beta.	23,694	40,492,055	117,469	1.25; 1.28 (B.vulgaris/B.maritima)

Science festival sequencing was carried out in the Kew Science Festival marquee (with flowcells later transported back to the lab to continue running in some cases):

Confirmed Species Name	Sci fest resequencing run number	Sci fest randomised letter	Sci fest guessed ID	Reads sequenced at ID	bp sequenced at ID	Time to identification (TTI)	First sequence start	fasta file write time	(oldest TTI)
???	Sample_01	?	Nepenthes alata	1460*	1495967*		?	?	14:35*
???	Sample_02	?	Arabidopsis lyrata ssp. petraea**	818* *	603677* *		?	17:26	(next day)
Silene uniflora	Sample_03	F	Silene uniflora	1,421	2,725,090		?	14:50	15:19
Erycina echinata	Sample_04	E	Erycina echinata	1,679	2,028,975		?	16:56	22:56
Sorbus aria	Sample_05	D	Sorbus aria	210	505,043		?	14:11	14:19
Beta patula	Sample_06	B	Beta patula	3,546	7,226,106		?	16:01	16:05

Resampling in silico:

Simulations with 20 replicates were constructed by subsampling (without replacement) from the R4IDs-sequencing and festival-resequenced data sets. For each simulation replicate, the R4ID input and resequencing run were subsampled randomly for 100, 500, 1000, 5000 and 10000 reads, in combination (25 orthogonal combinations per input sample). Each R4IDs dataset was converted to a BLASTN database separately.

Sample ID via BLASTN:

Each simulated sample was matched to every BLASTN database in turn with the following parameters: `blastn -outfmt "6 sacc qacc length pident evalue" -evalue 0.01 -num_threads 6 -num_alignments 1 -max_hsps 1 -db $this_db -query $this_test_input`

Only the top hit for each read is reported in the output file. Output files for each database/sample intensity combination (5 samples * 5 databases * 5 sampling intensities * 5 database intensities; 625 total) were then combined as follows: For each input sample at a given sampling/DB intensity combination, all hits against each of the 5 databases were collected for every read. The following ID statistics were then compiled:

Stat	Definition
TP one-way	Reads only matching the correct DB
FP one-way	Reads only matching an incorrect DB
TP two-way	Reads matching the correct DB, and at least one other DB, but with the longest number of identities for the correct DB
FP two-way	Reads matching the correct DB, and at least one other DB, but with the longest number of identities for the INcorrect DB; or matching more than one DB but NOT the correct one at all

Stat	Definition
mean bias	Average of (TP hit - next-best-hit) for all reads with at least two hits, one of which is for the correct DB
two-way rate	Ratio of TP two-way : total two-way hits
two-way rate with cutoff	Ratio of hits with bias > 50bp : total two-way hits (**e.g., more stringent**)
total hits	Total number of reads with any hit to any DB

Not all replicate / sample combinations generated results owing to random error. In the plots below data were aggregated amongst replicates. 20 replicates were carried out in total.

Note: For these the following sample identities were assumed…

Sample	File	TP label
sample_1	_Volumes_SCI-FEST-A_sci-fest-data_20160823_sample_01_basecall_both-runs.poretools.fasta	N.alata
sample_2	_Volumes_SCI-FEST-A_sci-fest-data_20160823_sample_02_basecall_both-runs.poretools.fasta	? Arabidopsis (**omitted**)
sample_3	_Volumes_SCI-FEST-A_sci-fest-data_20160823_sample_03_basecall_both-runs.poretools.fasta	S.uniflora
sample_4	_Volumes_SCI-FEST-A_sci-fest-data_20160823_sample_04_basecall_both-runs.poretools.fasta	E.echinata

Sample	File	TP label
sample_5	__Volumes_SCI-FEST-A_sci-fest-data_20160823_sample_05_basecall_both-runs.poretools.fasta	S_aria
sample_6	__Volumes_SCI-FEST-A_sci-fest-data_20160823_sample_06_basecall_both-runs.poretools.fasta	B_patula

Results

```r
# set up
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```r
library(lattice)
```

```r
# read in the input for each labelling / subsampling size
rep_01 = read.table('~/Documents/all_work/programming/oddjects-sandbox/R4IDs/collected_all_replicates.t

# cbind them all into a big table, adding the 'training_intensity' and 'query_intensity' cols

# calculate TP rate for a variety of cutoffs

# plot basic
plot(sample_DB_intensity ~ species_DB_intensity, data=rep_01[rep_01$TP_species=='Beta-patula',])
plot(sample_DB_intensity ~ two_way_rate_with_cutoff, data=rep_01[rep_01$TP_species=='Beta-patula',])

# plot heatmap prediction
#rep_01[rep_01$TP_species=='Beta-patula',c(2,4,12)]
levels(rep_01$TP_species)
species = c(
  "Beta-patula",
  "Erycina-echinata",
  "Napenthes-alata",
  "Silene-uniflora",
  "Sorbus-aria")
# bias
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
```

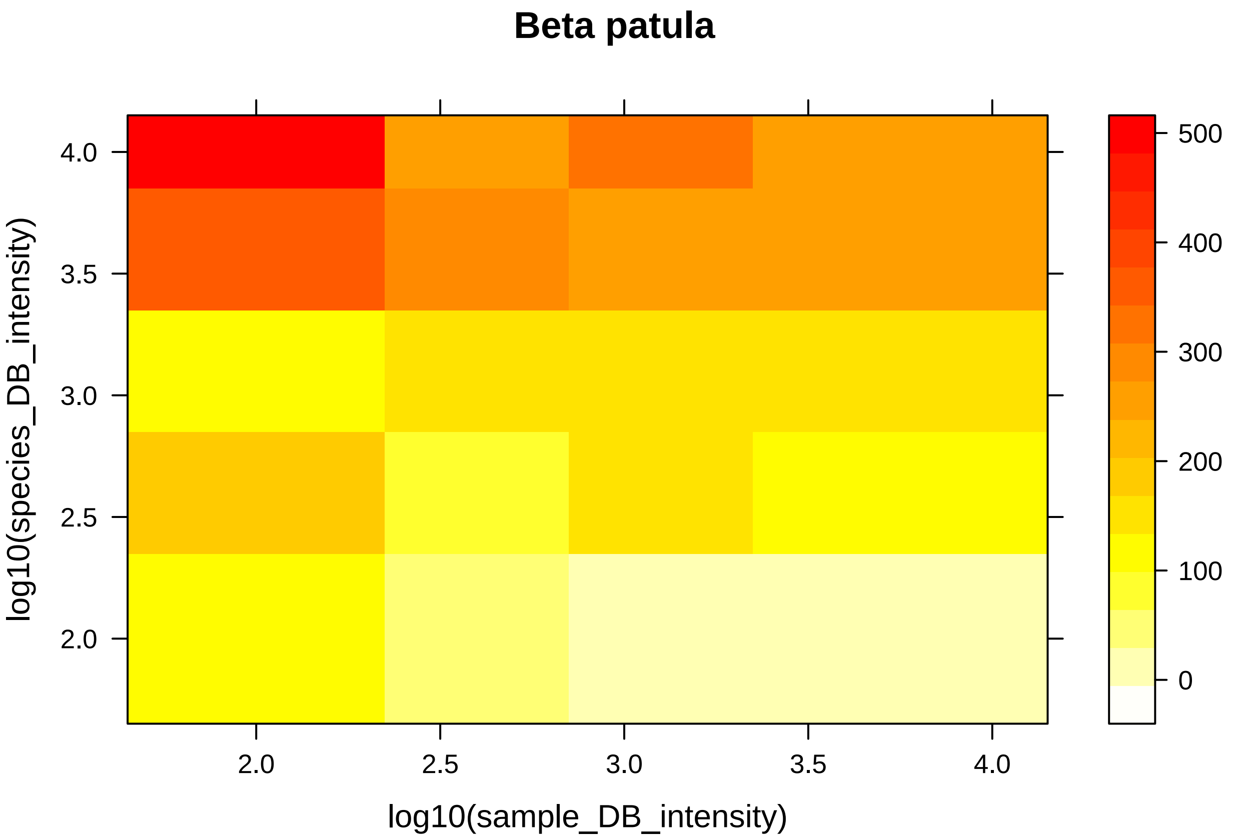
```
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01[rep_01$TP_sp
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01[rep_01$TP_sp
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01[rep_01$TP_sp
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01[rep_01$TP_sp
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01[rep_01$TP_sp
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01, col.regions
```

Now plot 2-way rate with cutoff=50:

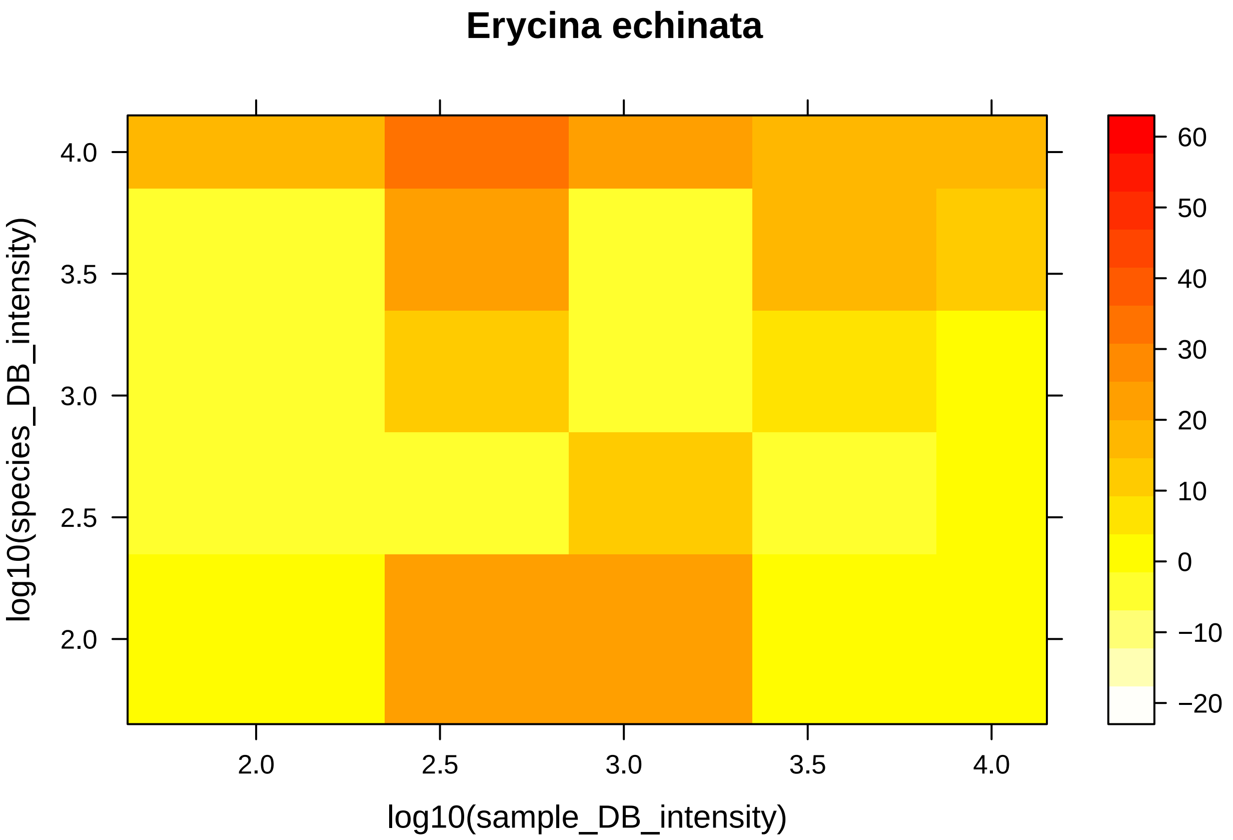
```
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=rep_01
```

Now repeat it properly for all reps. Mean bias first:

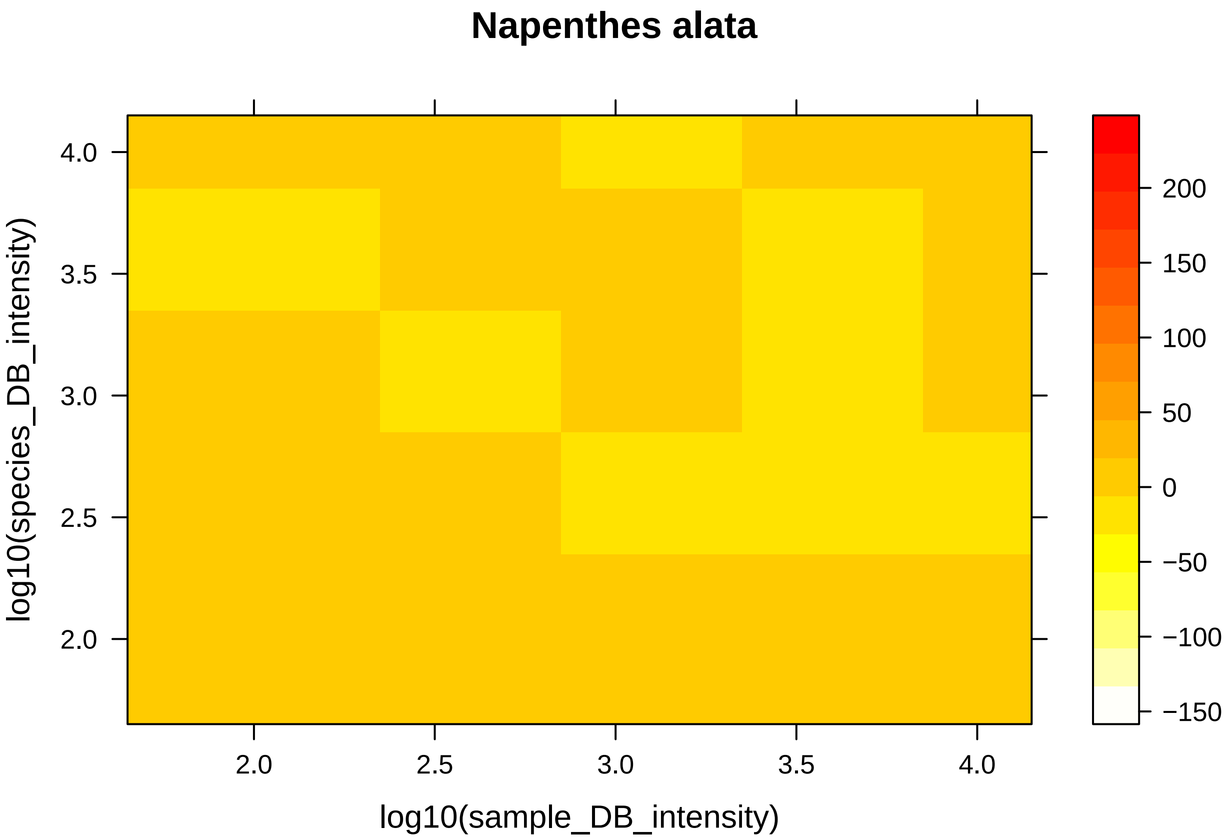
```
# read in the input for each labelling / subsampling size
all_reps = read.table('~/Documents/all_work/programming/oddjects-sandbox/R4IDs/collected_really_all_repl
# bias
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps[all_reps$TP_s
```



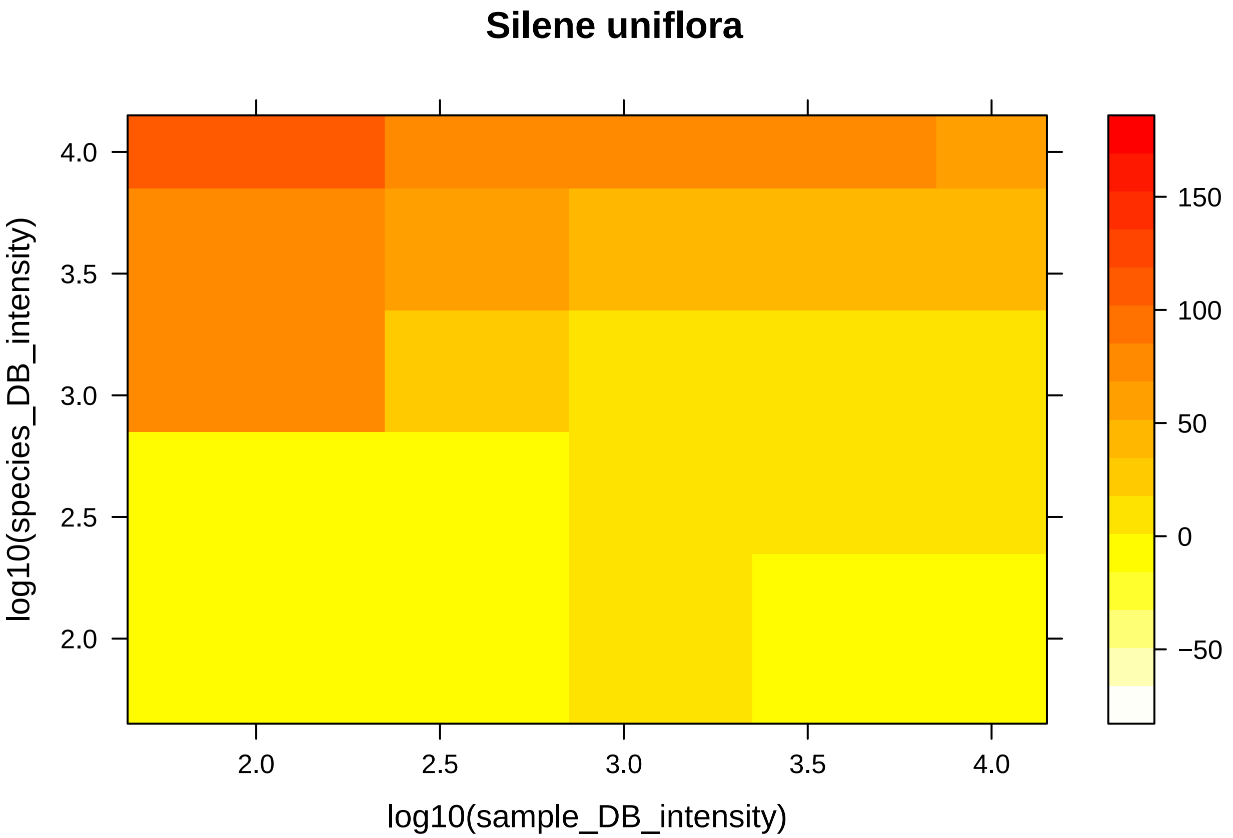
```
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps[all_reps$TP_s
```



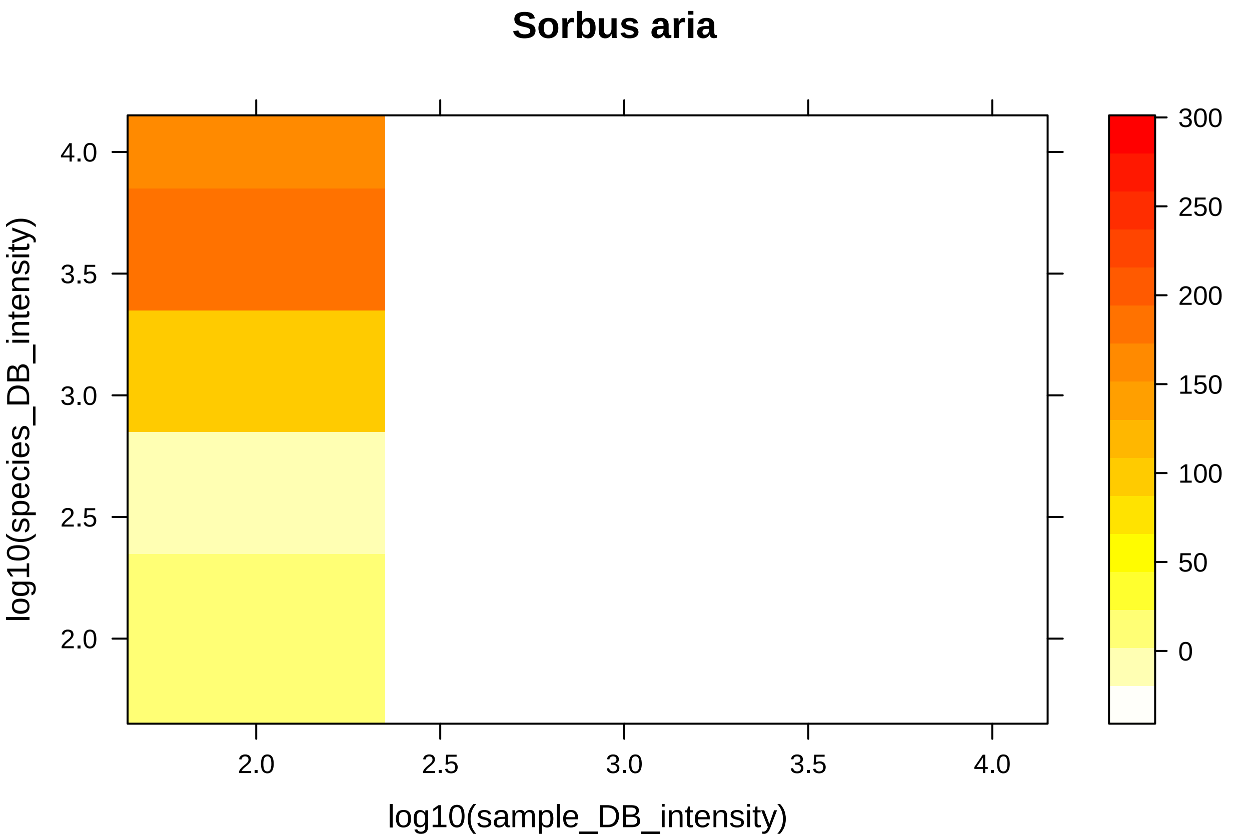
```
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps[all_reps$TP_
```



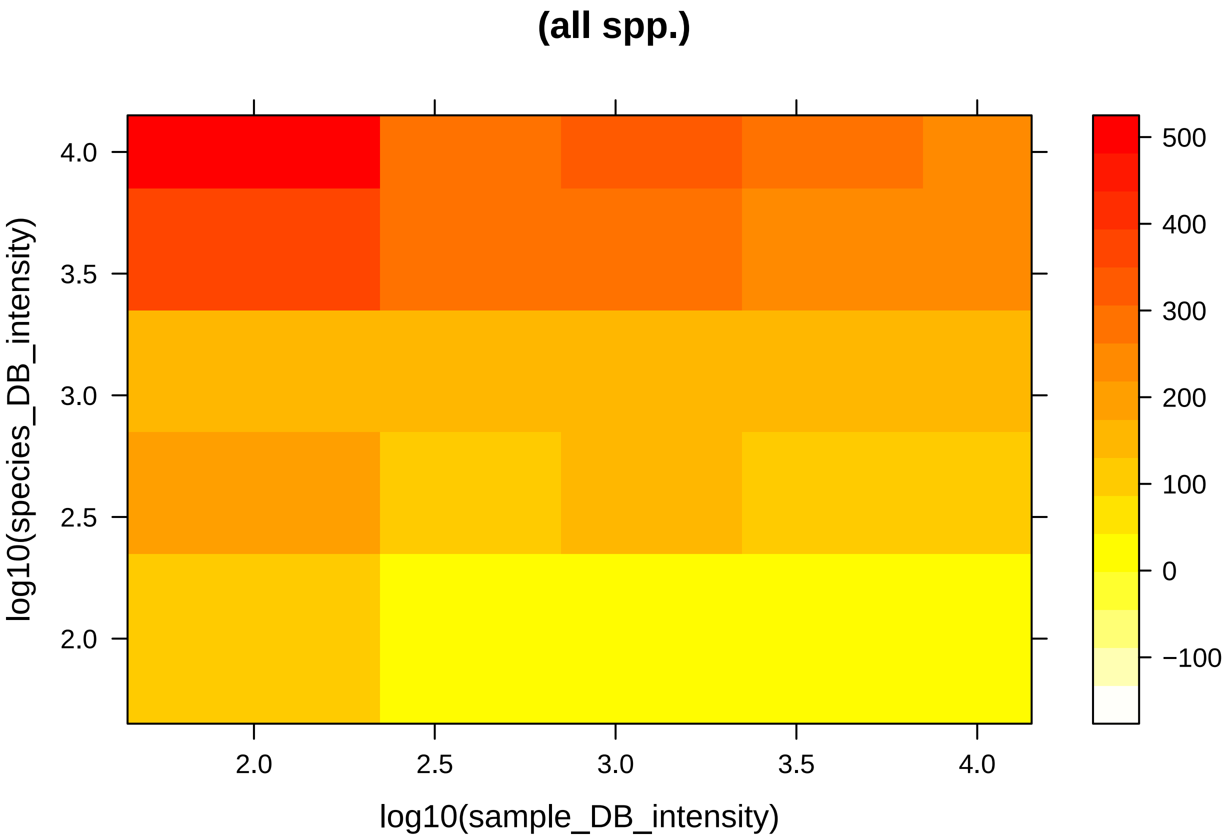
```
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps[all_reps$TP_
```



```
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps[all_reps$TP_
```

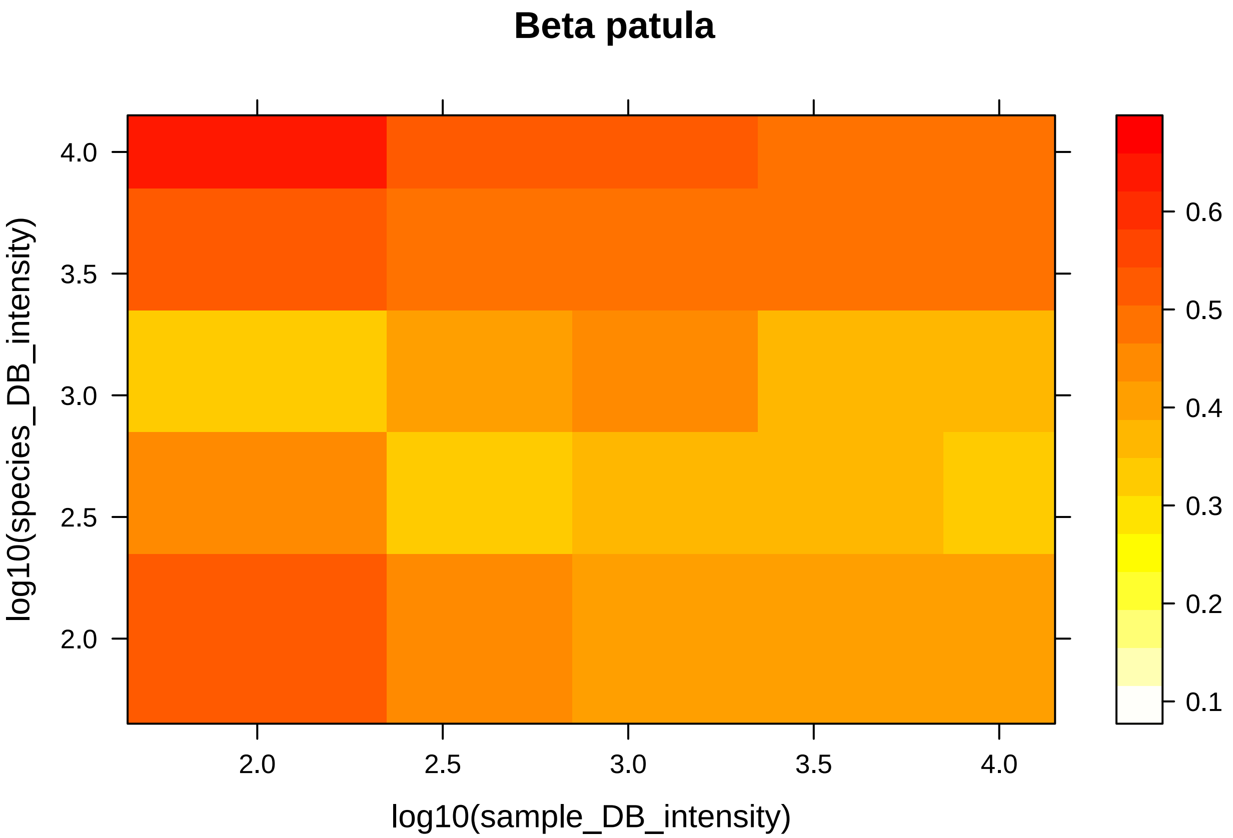


```
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps, col.regions
```

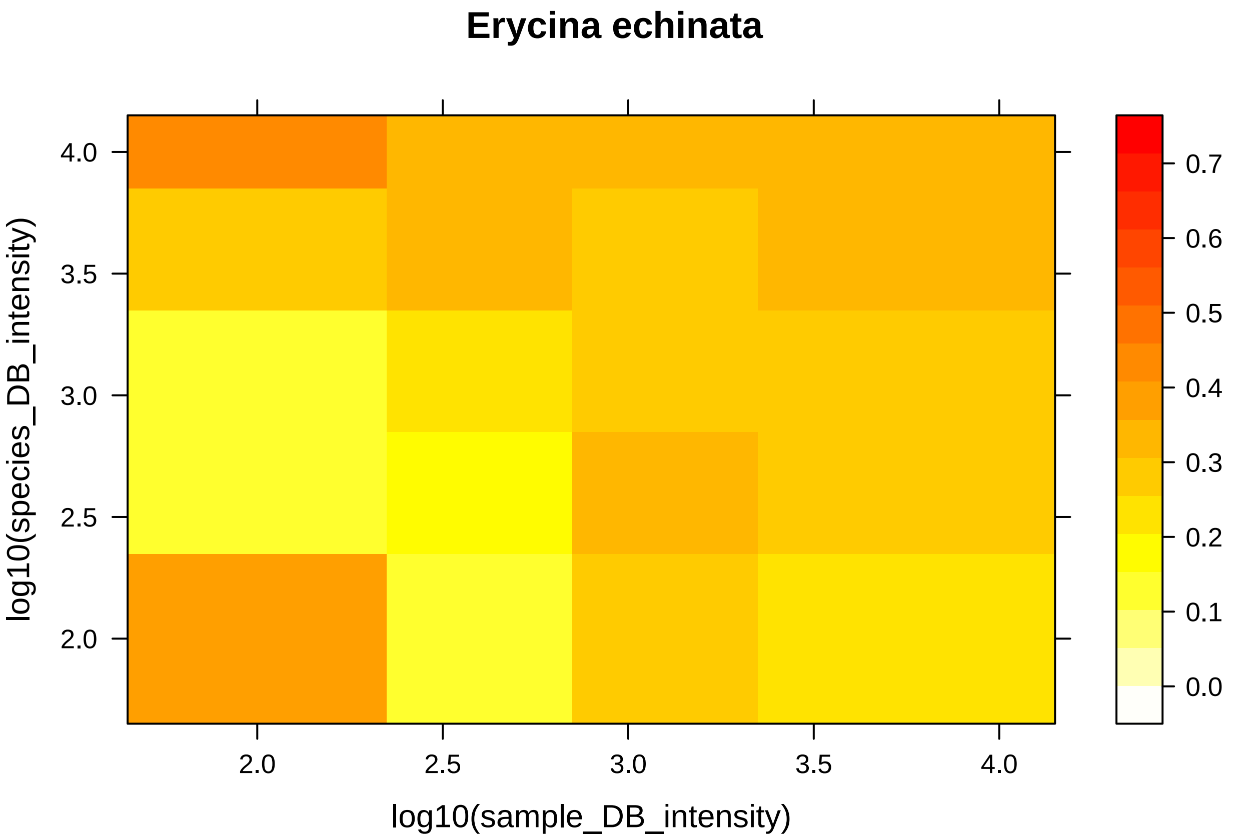


Now plot 2-way rate %:

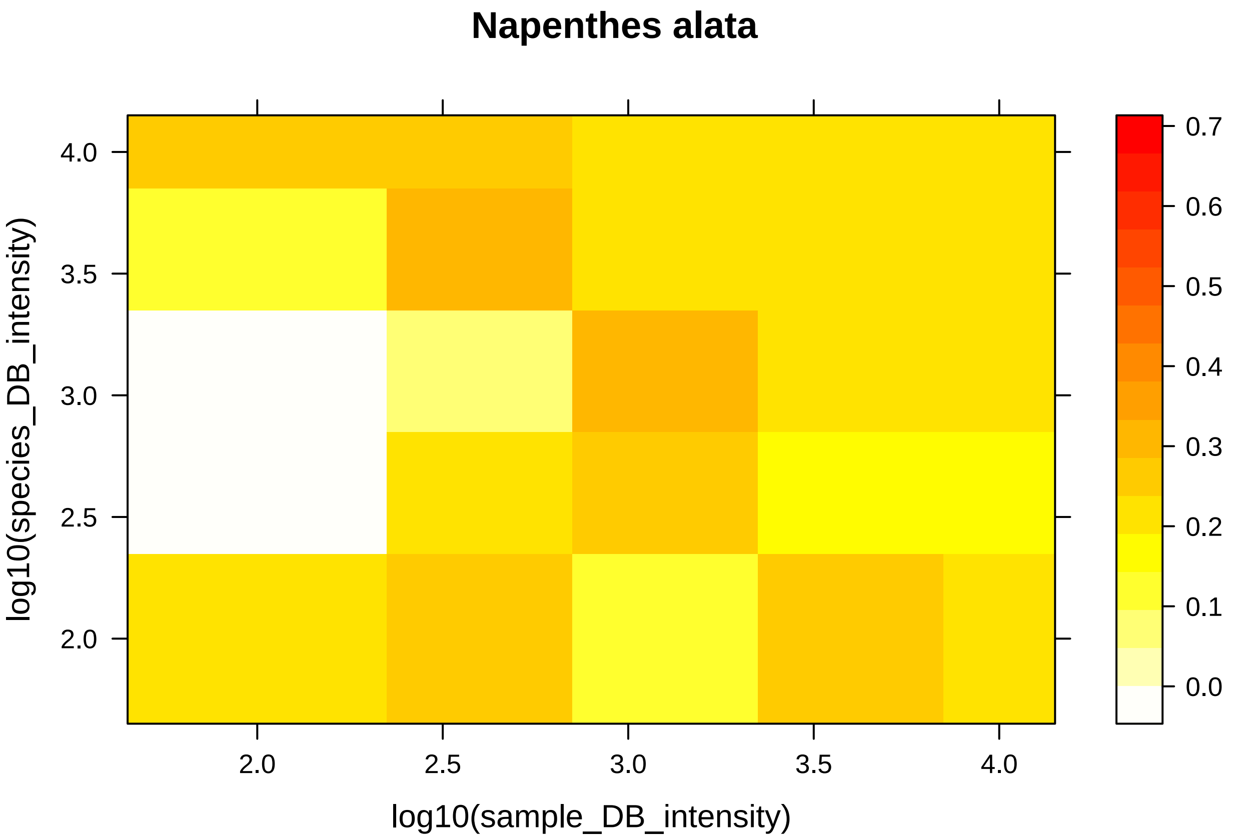
```
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps[all_reps$
```



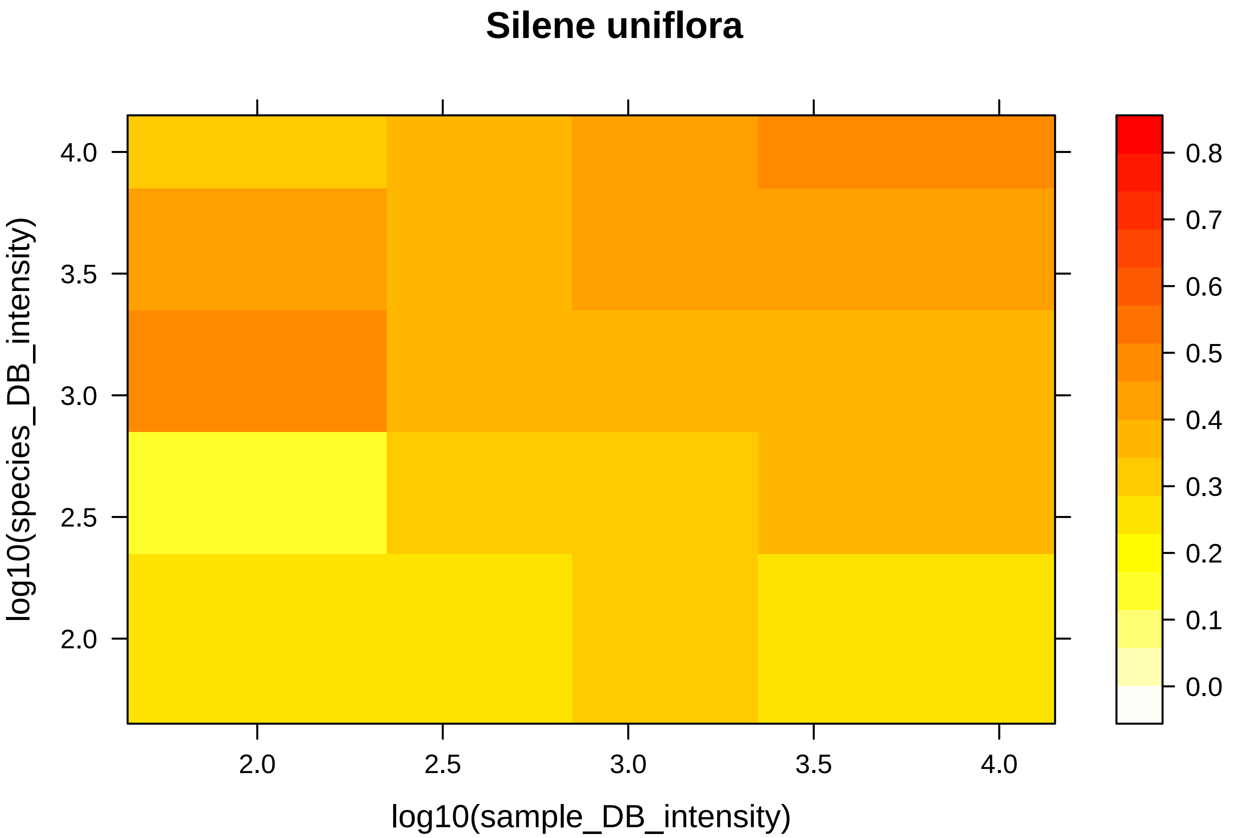
```
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps[all_reps$
```



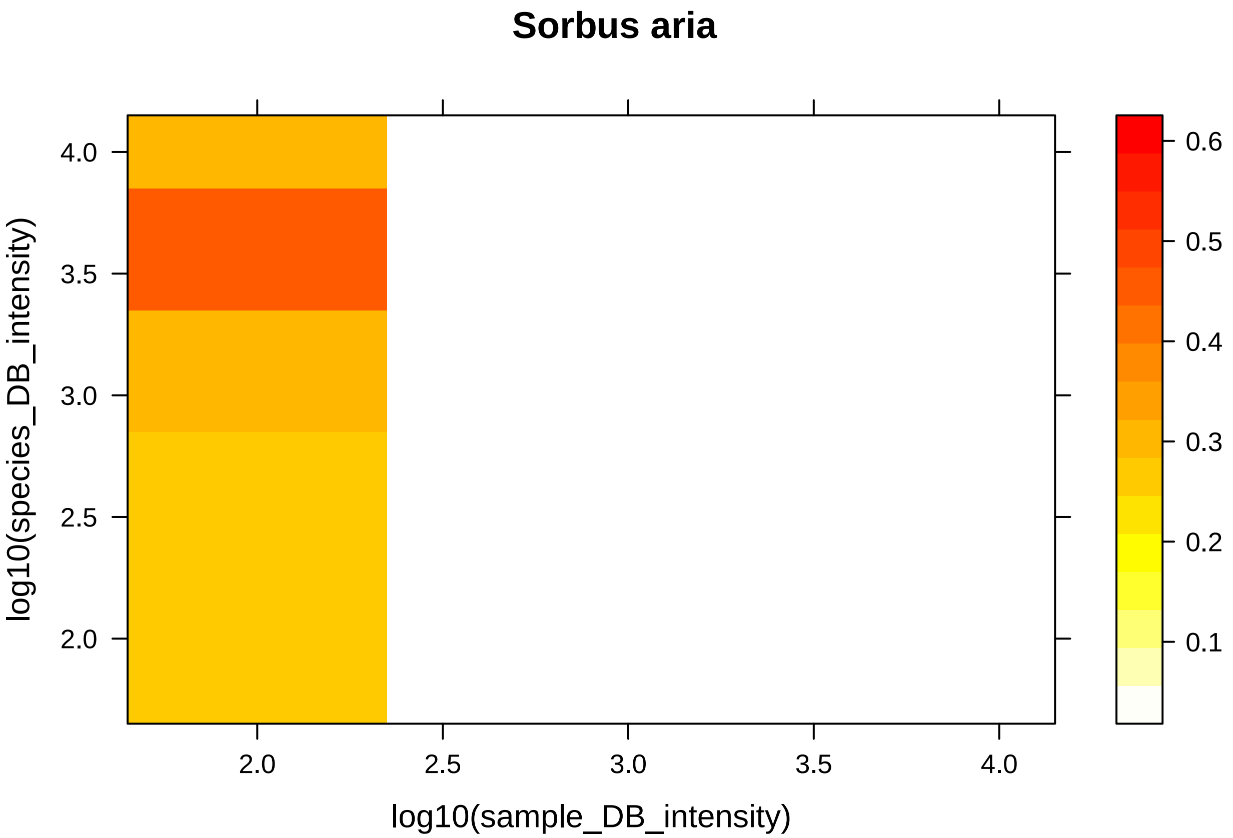
```
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps[all_reps$
```



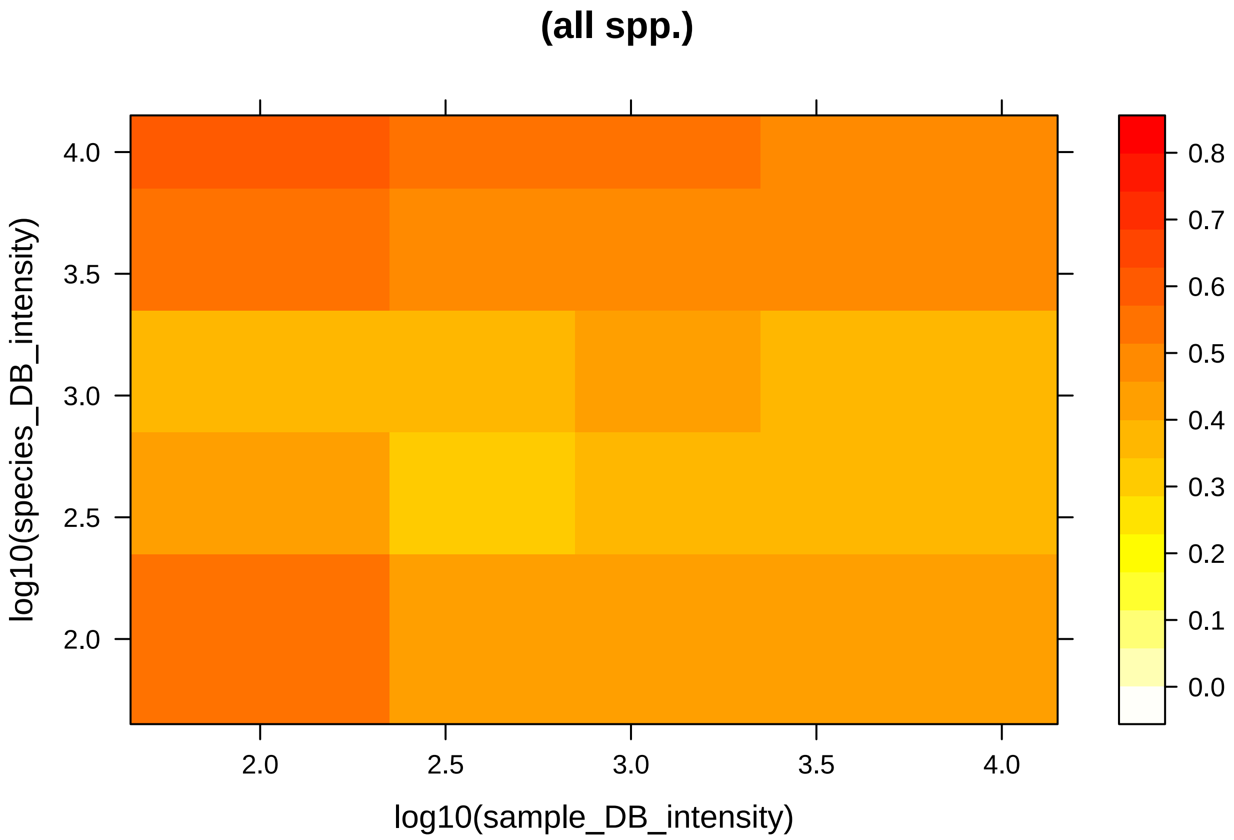
```
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps[all_reps$
```



```
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps[all_reps$
```

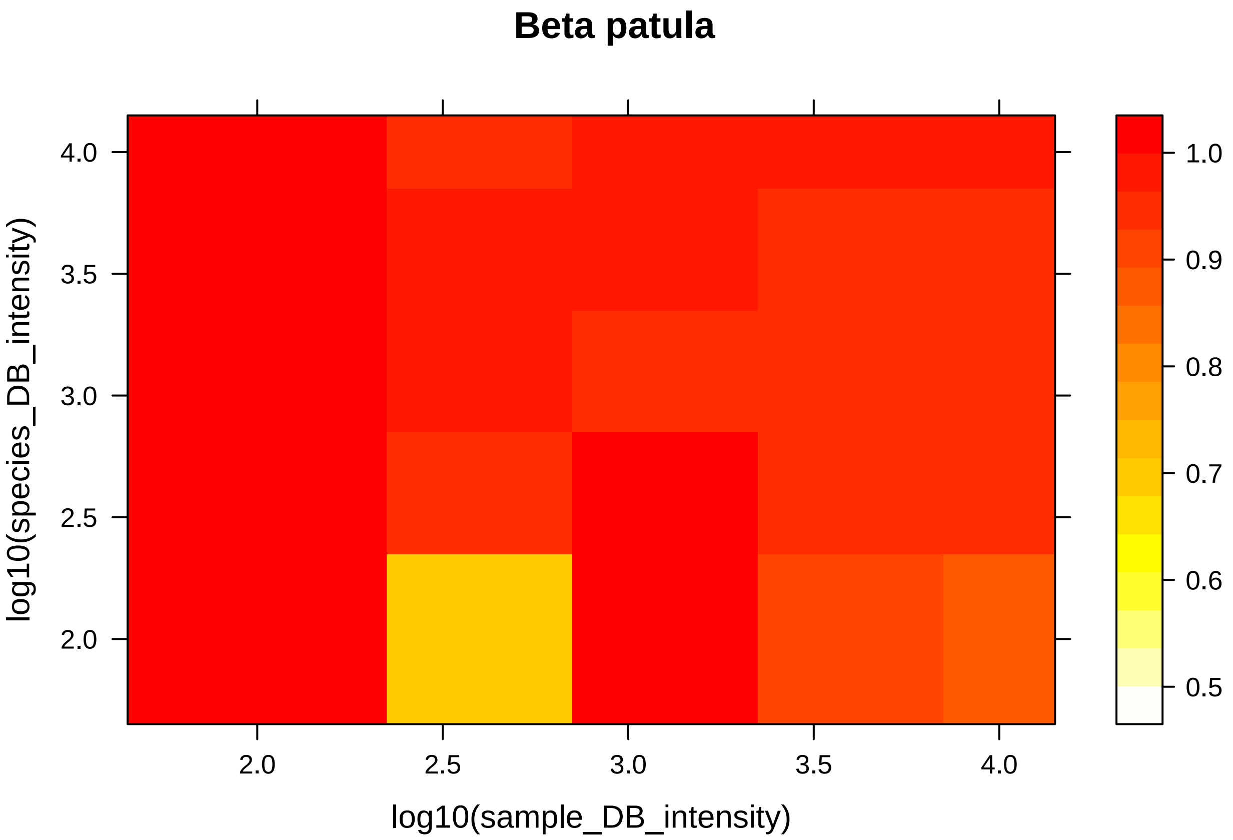


```
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps, col.regi
```



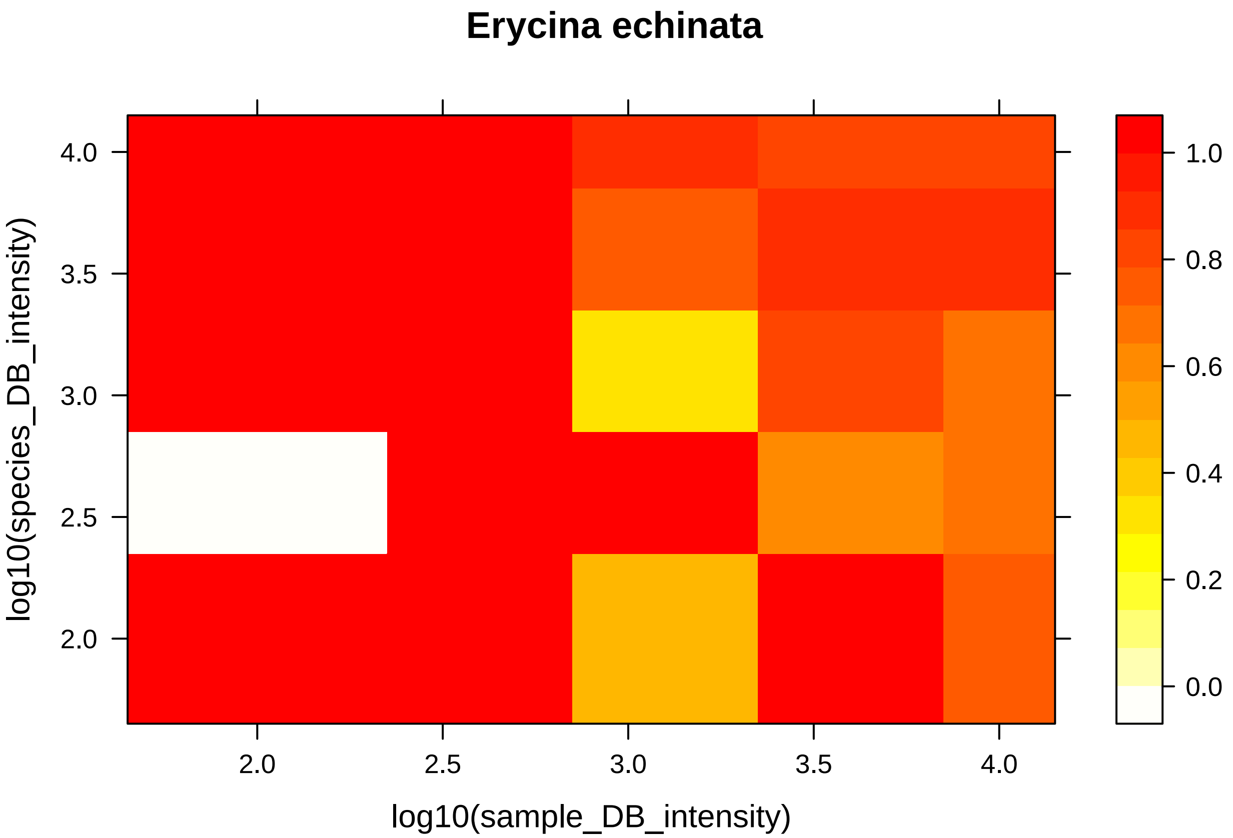
Now plot 2-way rate with cutoff=50:

```
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_rep
```

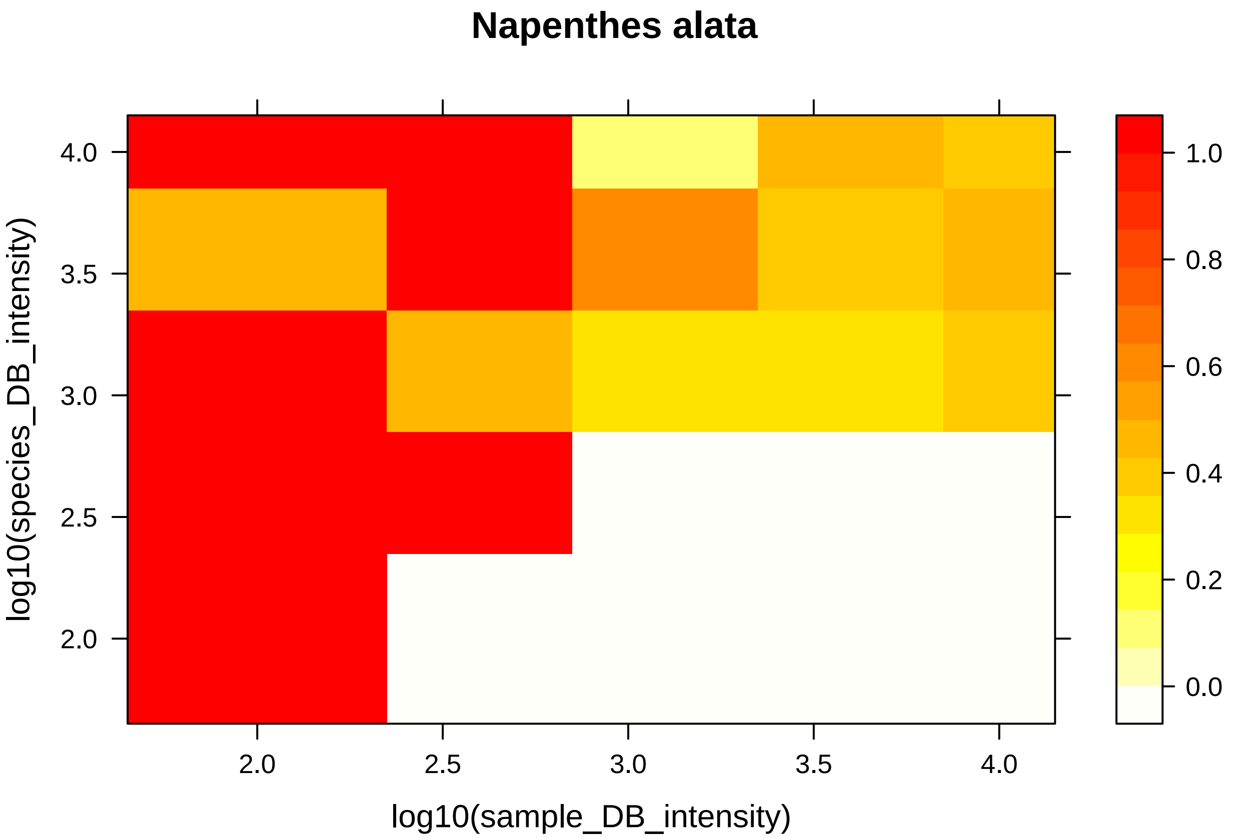


Beta patula

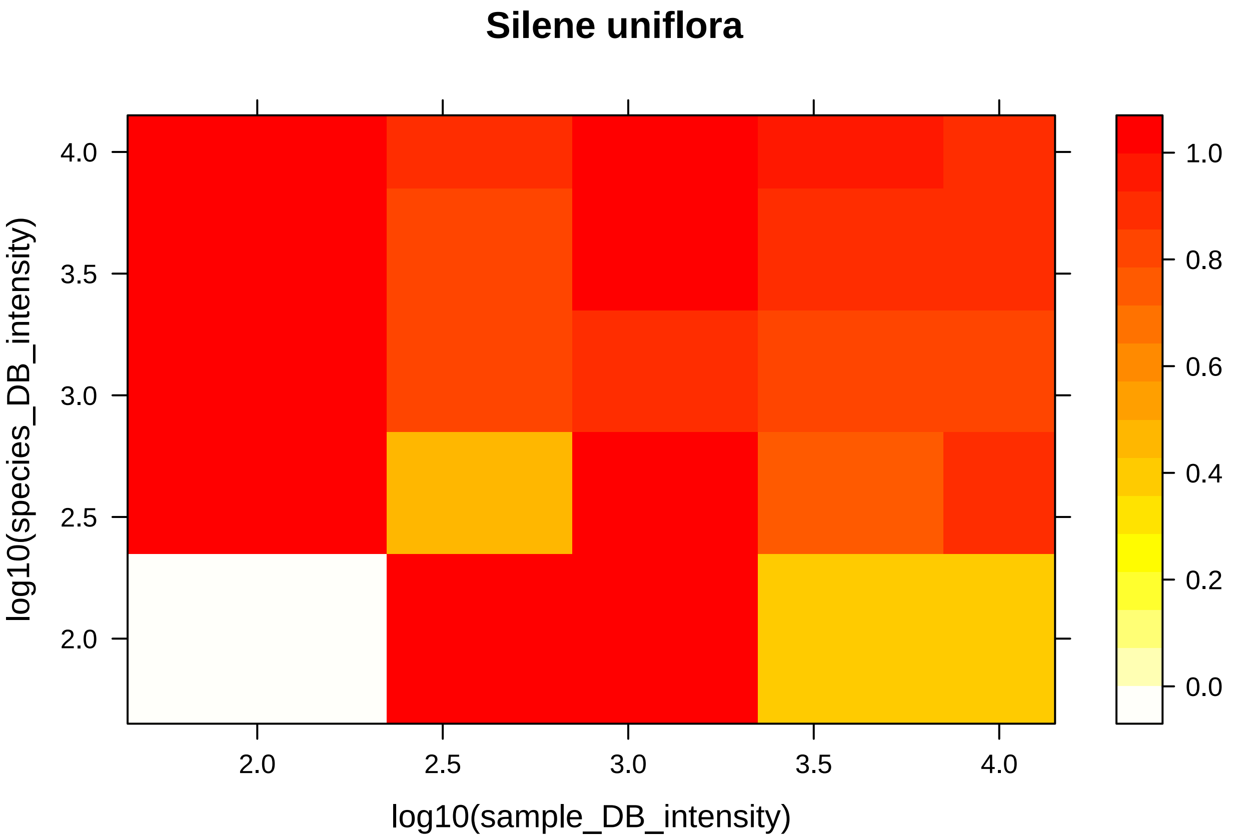
```
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_re
```



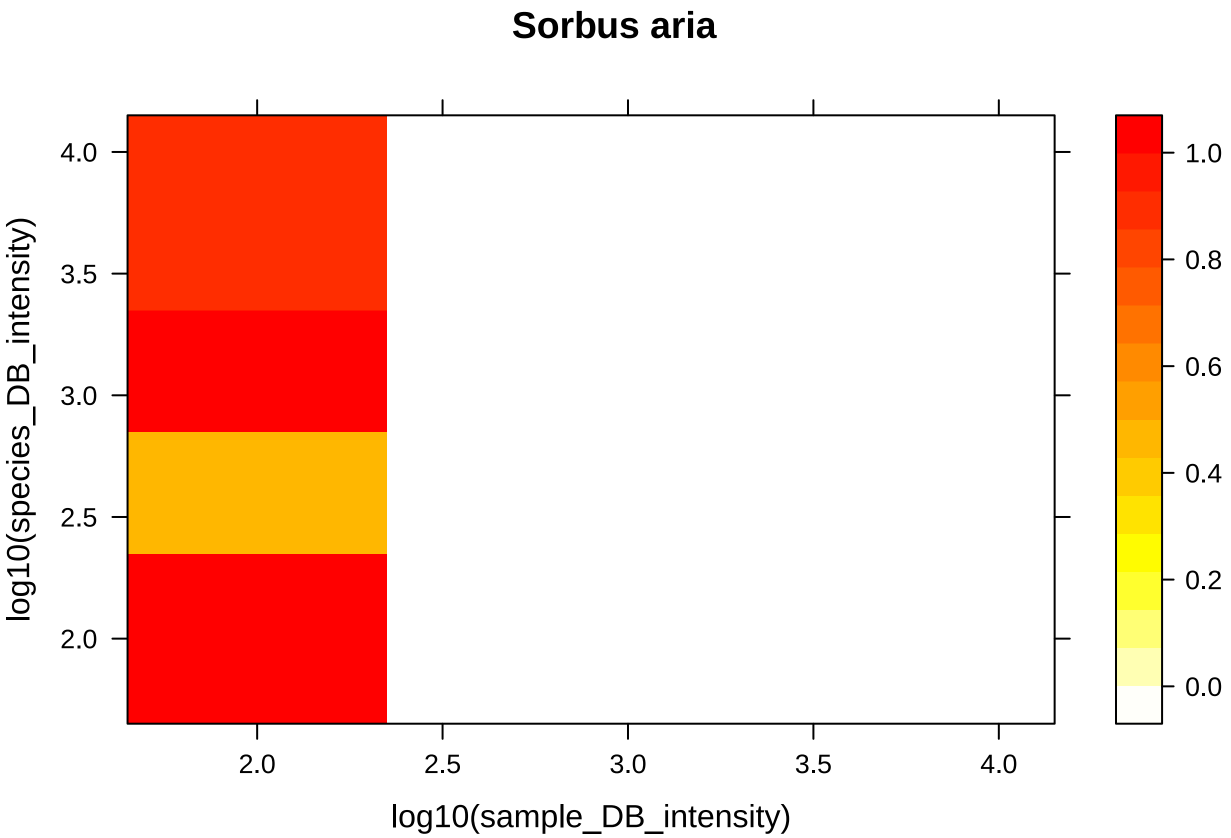
```
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_rep
```



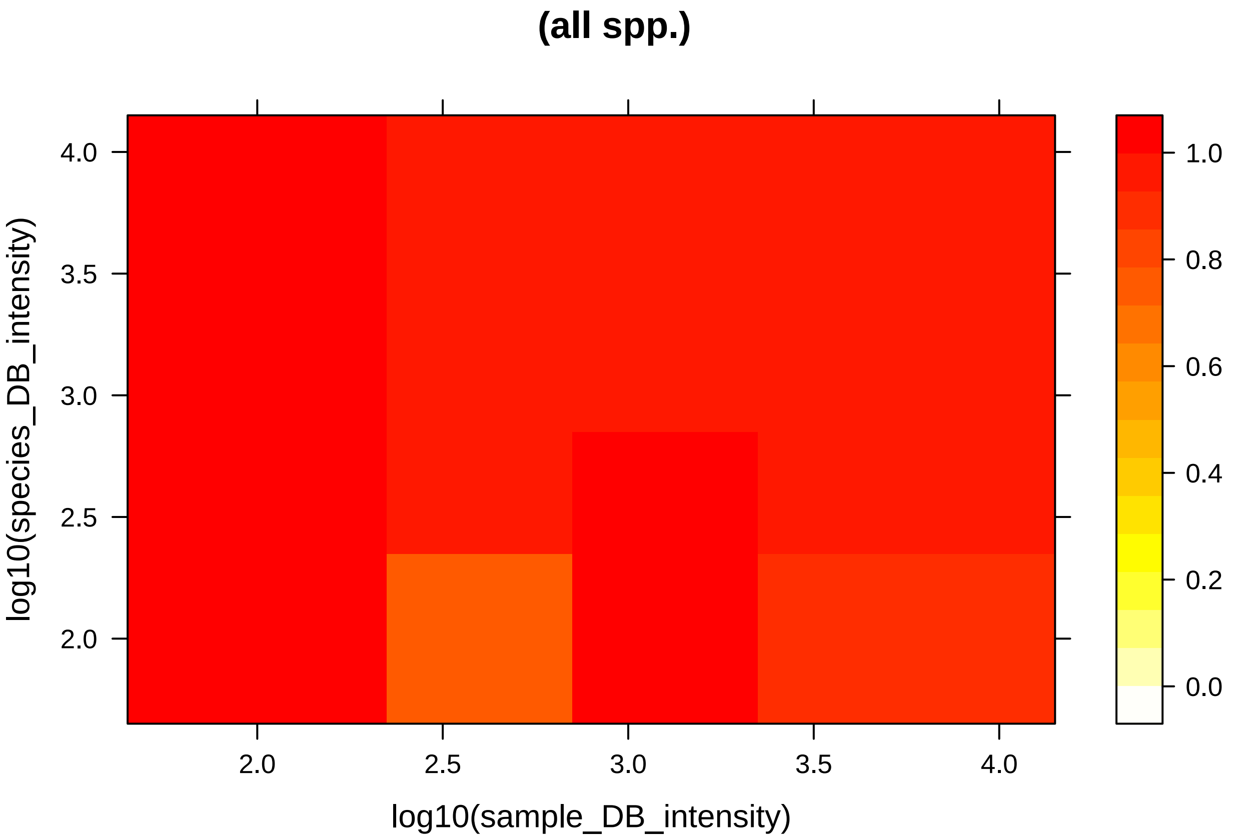
```
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_rep
```



```
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_rep
```



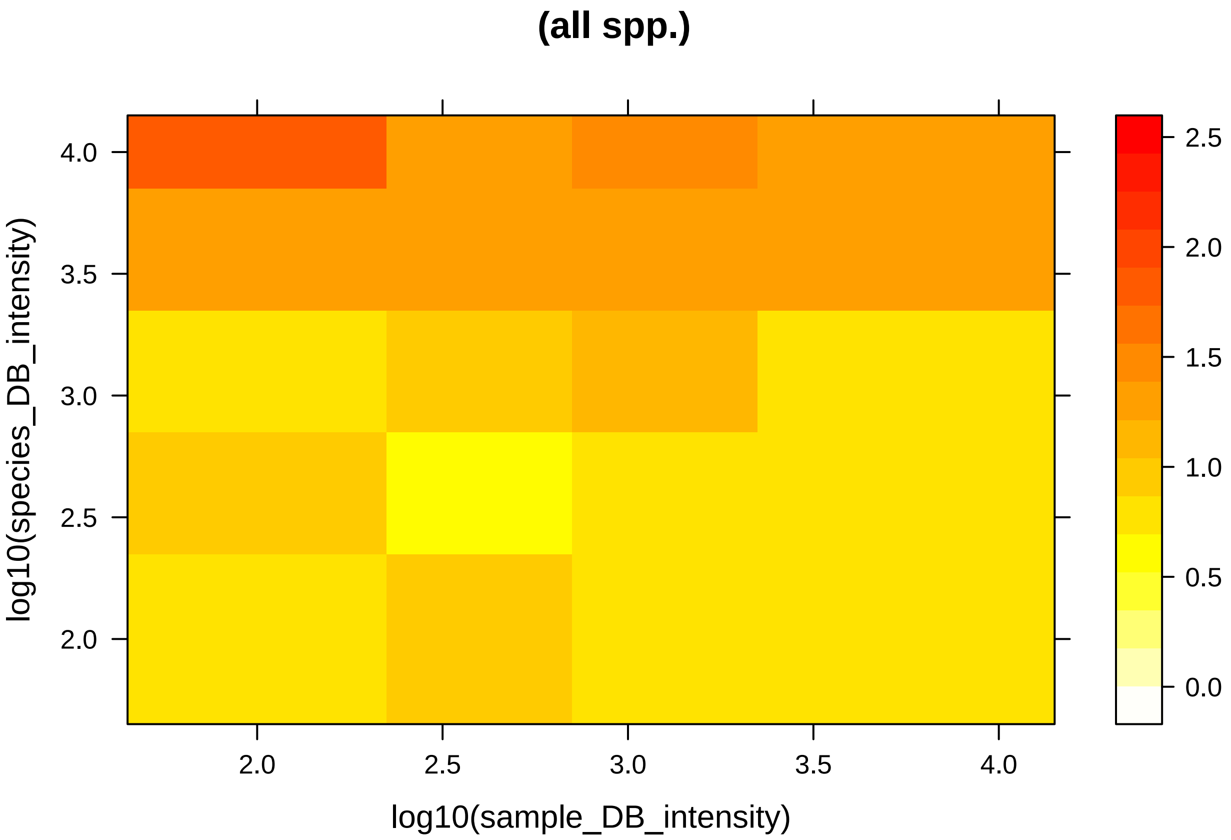
```
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_re
```



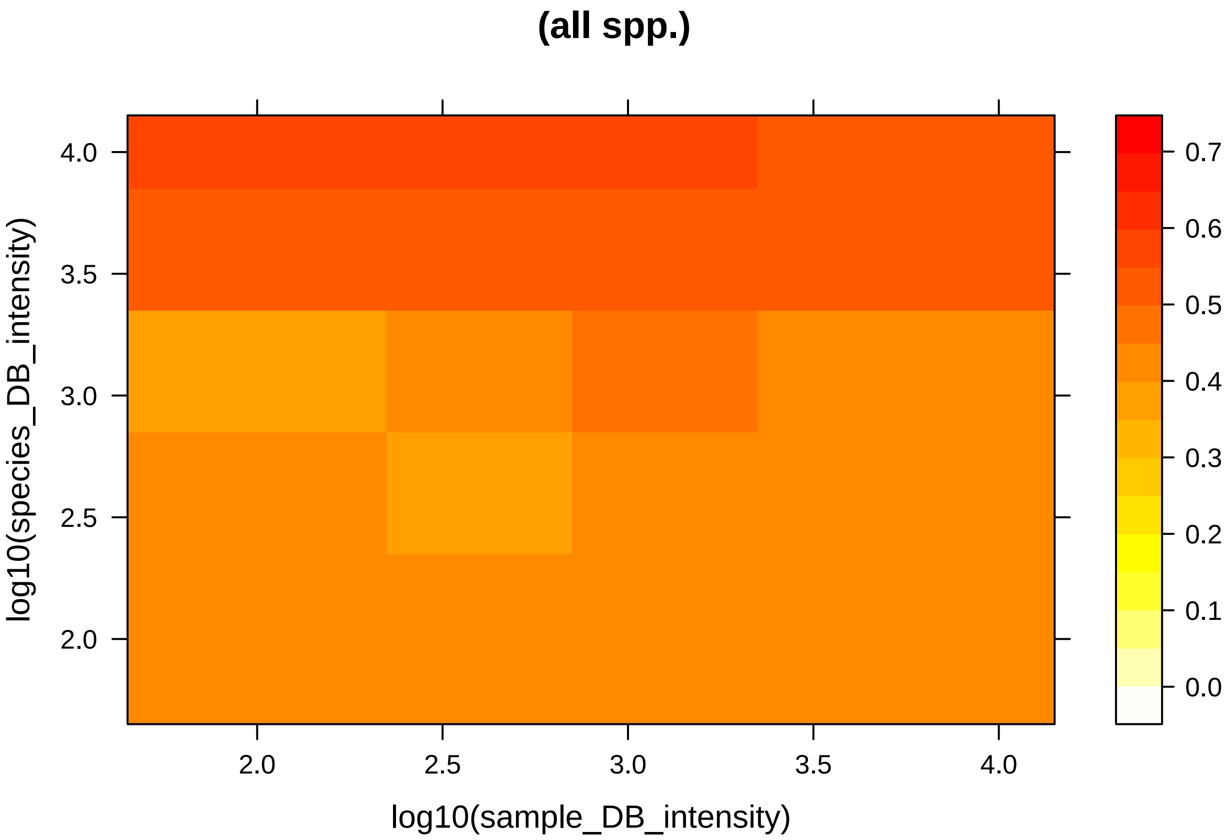
```
boxplot(mean_bias ~ species_DB_intensity * sample_DB_intensity,data=all_reps[all_reps$TP_species=='Sorbu
boxplot(two_way_rate ~ species_DB_intensity * sample_DB_intensity,data=all_reps[all_reps$TP_species=='Sc
boxplot(two_way_rate_with_cutoff ~ species_DB_intensity * sample_DB_intensity,data=all_reps[all_reps$TP_
boxplot(two_way_rate ~ species_DB_intensity * sample_DB_intensity,data=all_reps[all_reps$TP_species=='Na
boxplot(mean_bias ~ species_DB_intensity * sample_DB_intensity,data=all_reps[all_reps$TP_species=='Naper
boxplot(mean_bias ~ species_DB_intensity * sample_DB_intensity,data=all_reps[all_reps$TP_species=='Siler
boxplot(mean_bias ~ species_DB_intensity * sample_DB_intensity,data=all_reps[all_reps$TP_species=='Erycı
```

Now try adding 1- and 2-way hits

```
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(speci
```

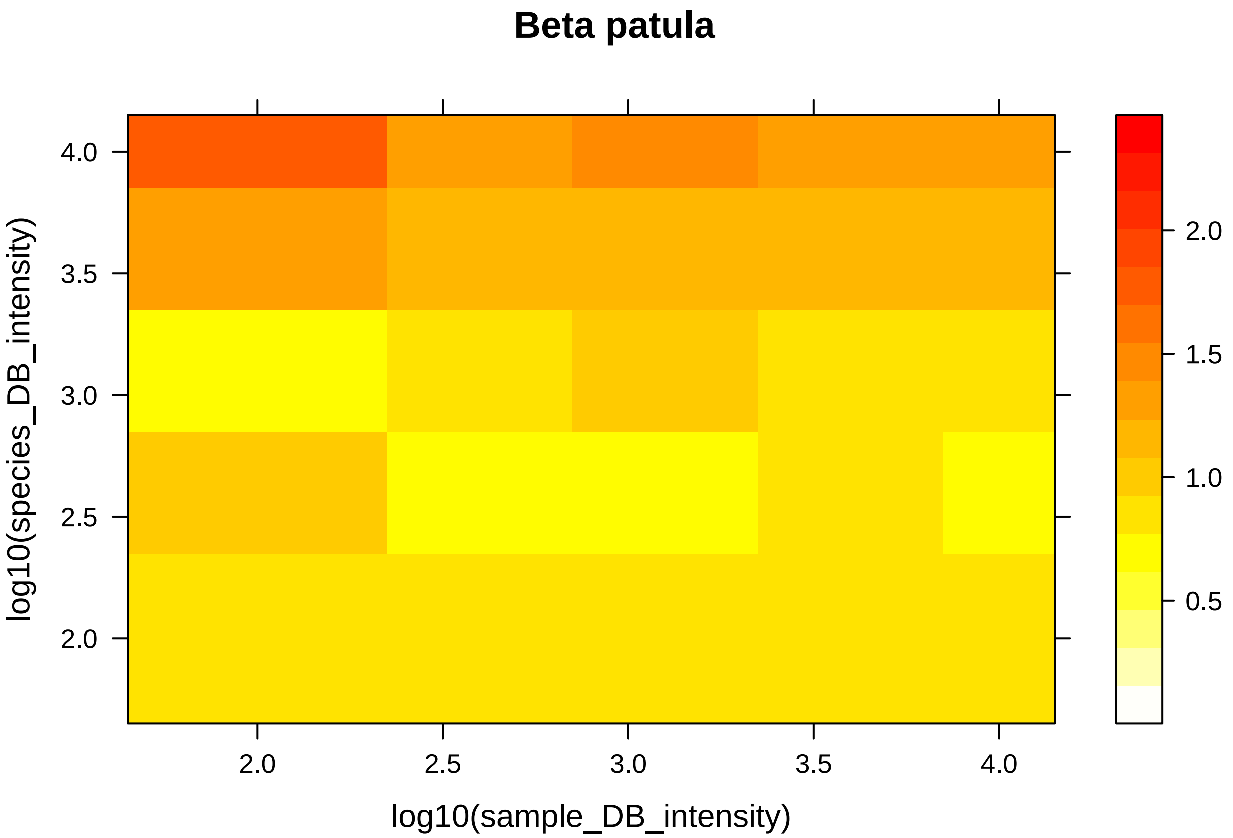


```
levelplot((one_way_TP + two_way_TP) / total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensit
```

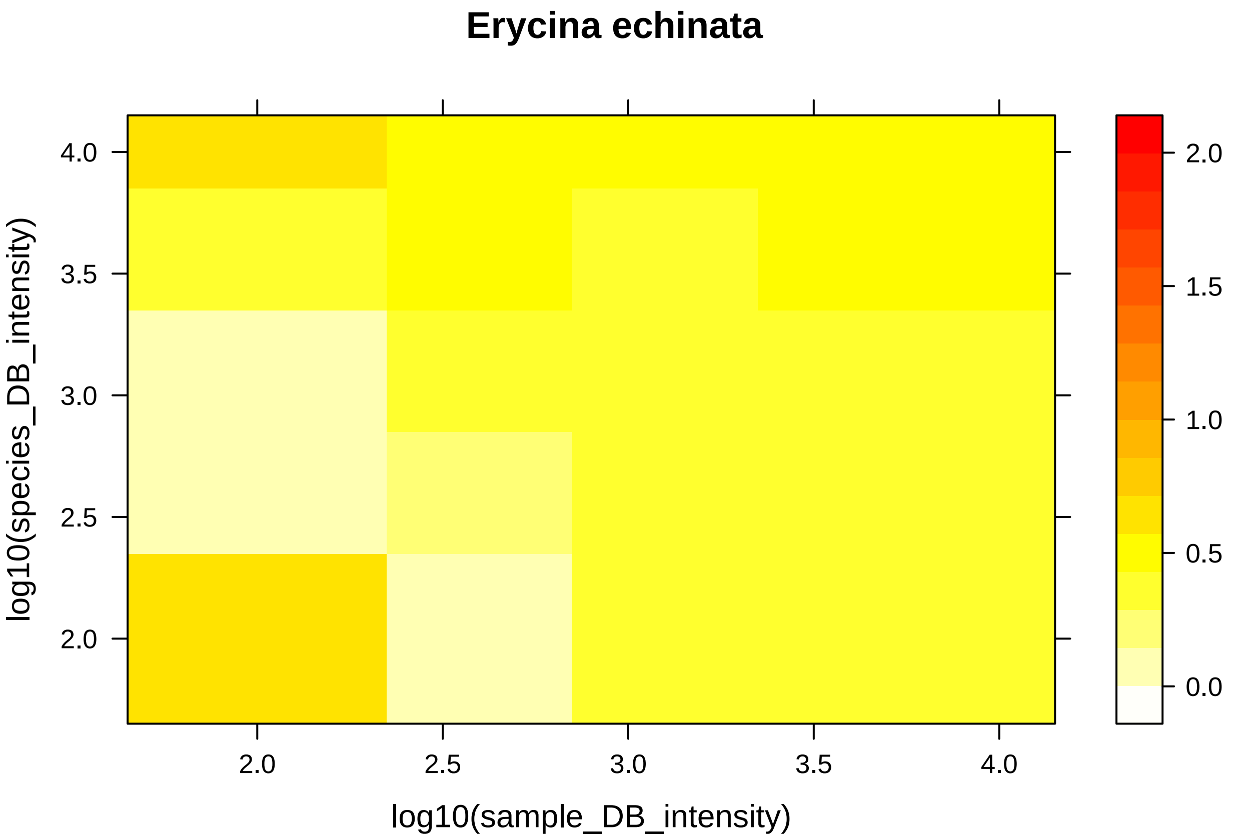


Hits ratios heatmap by species:

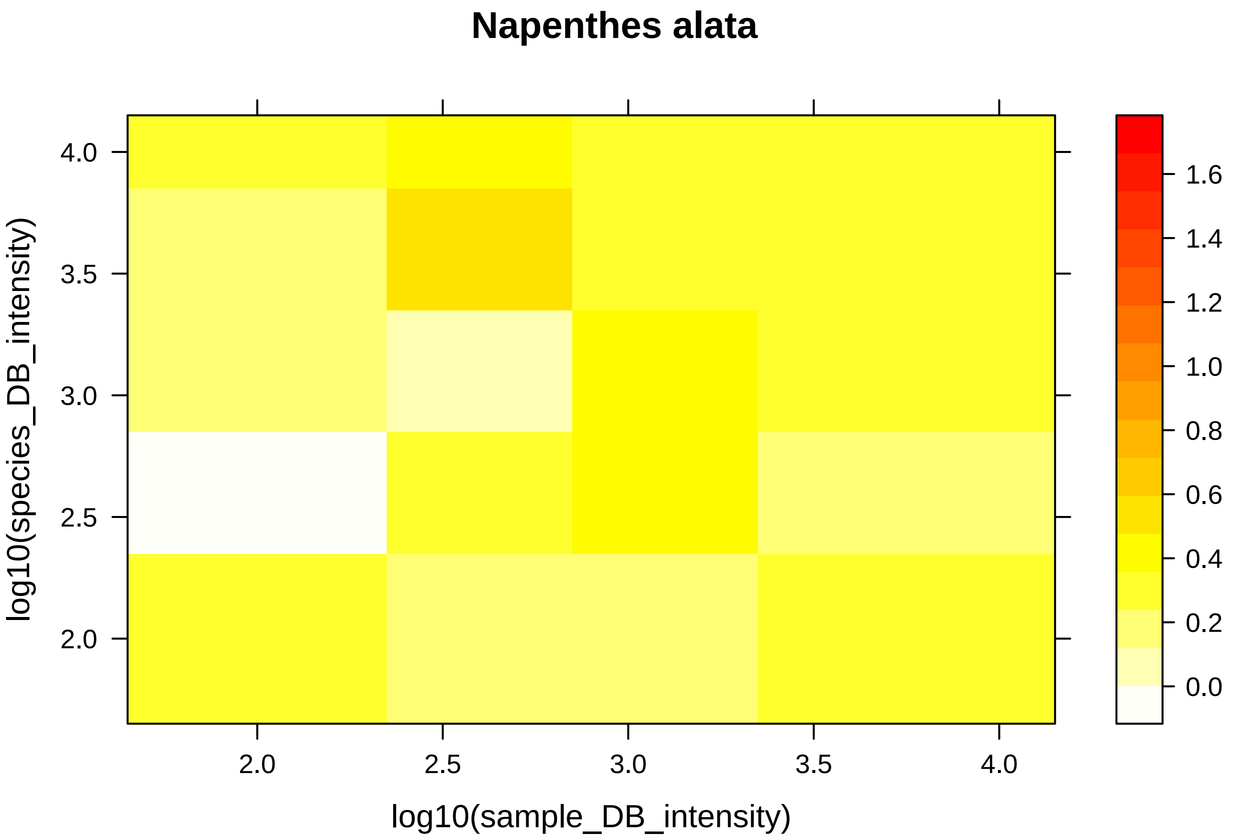
```
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(speci
```



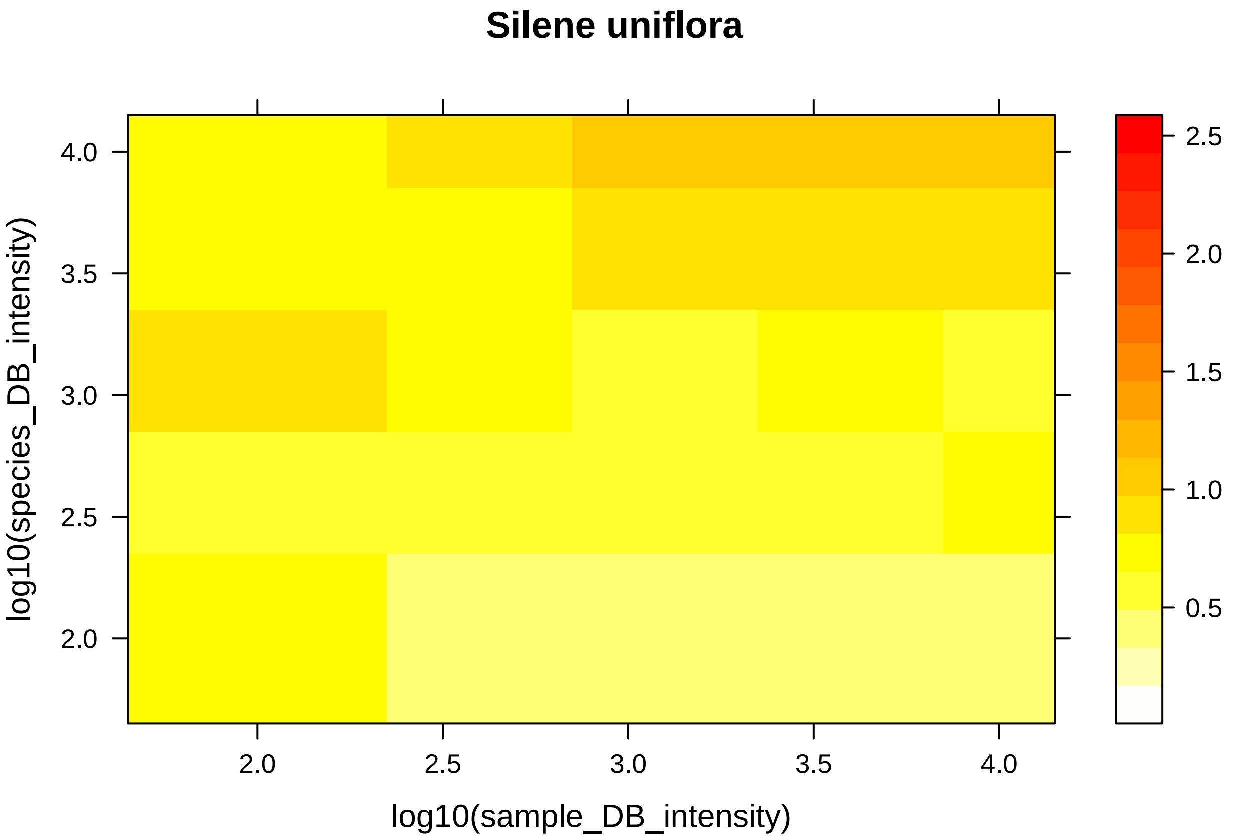
```
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(spec
```



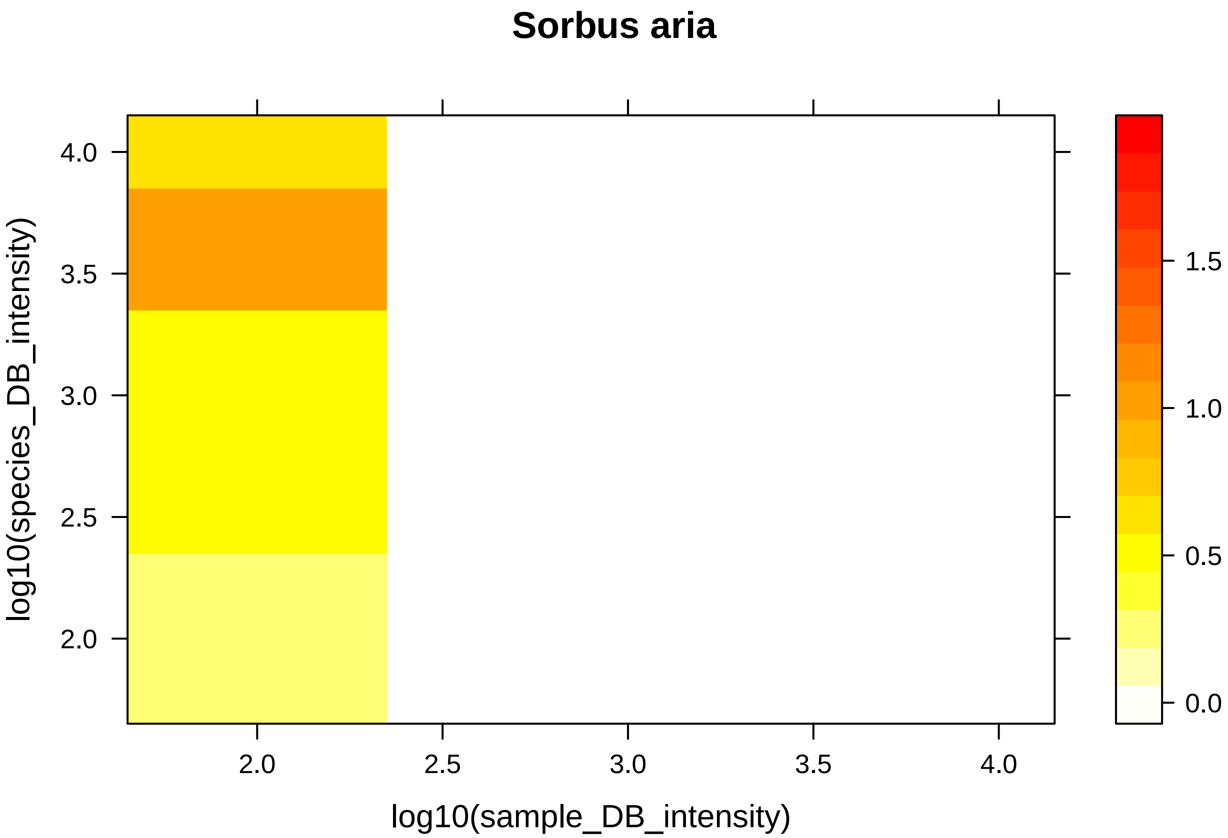
```
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(speci
```



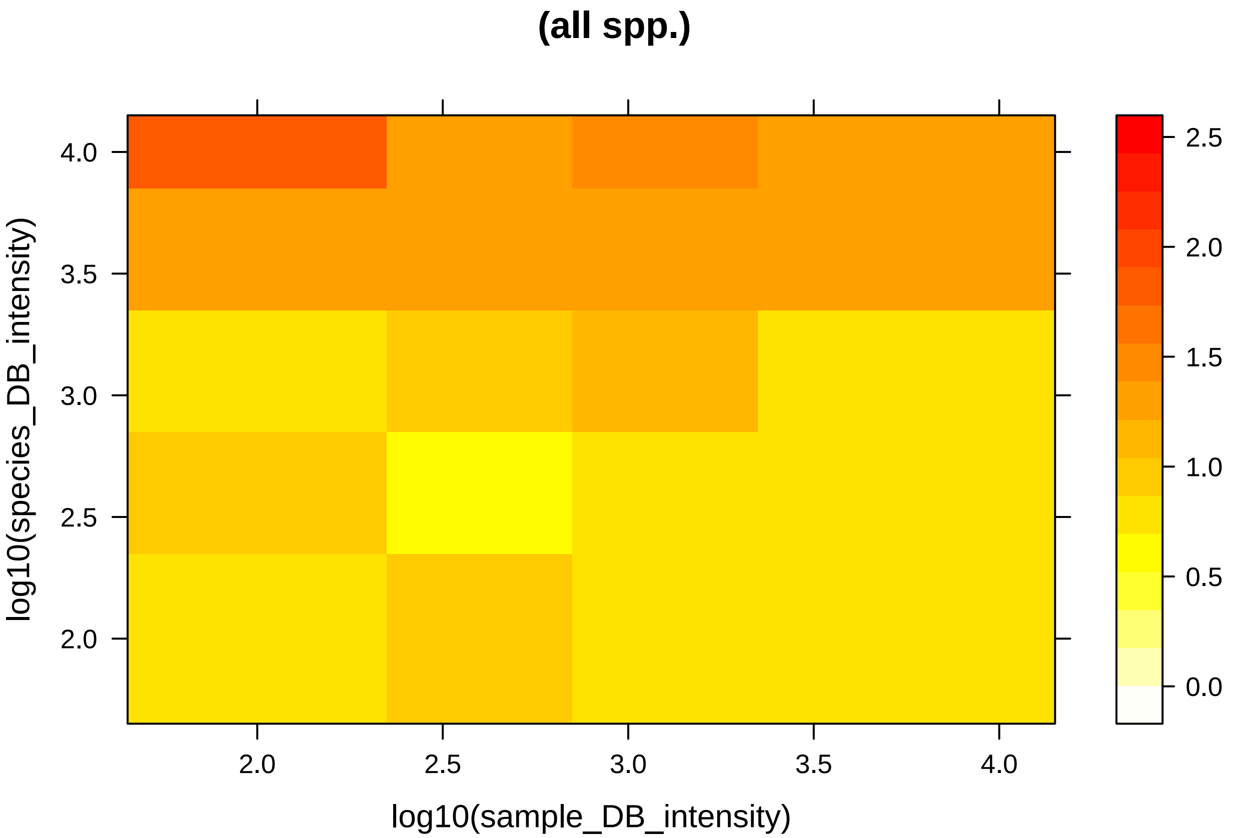
```
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(speci
```



```
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(speci
```



```
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(speci
```

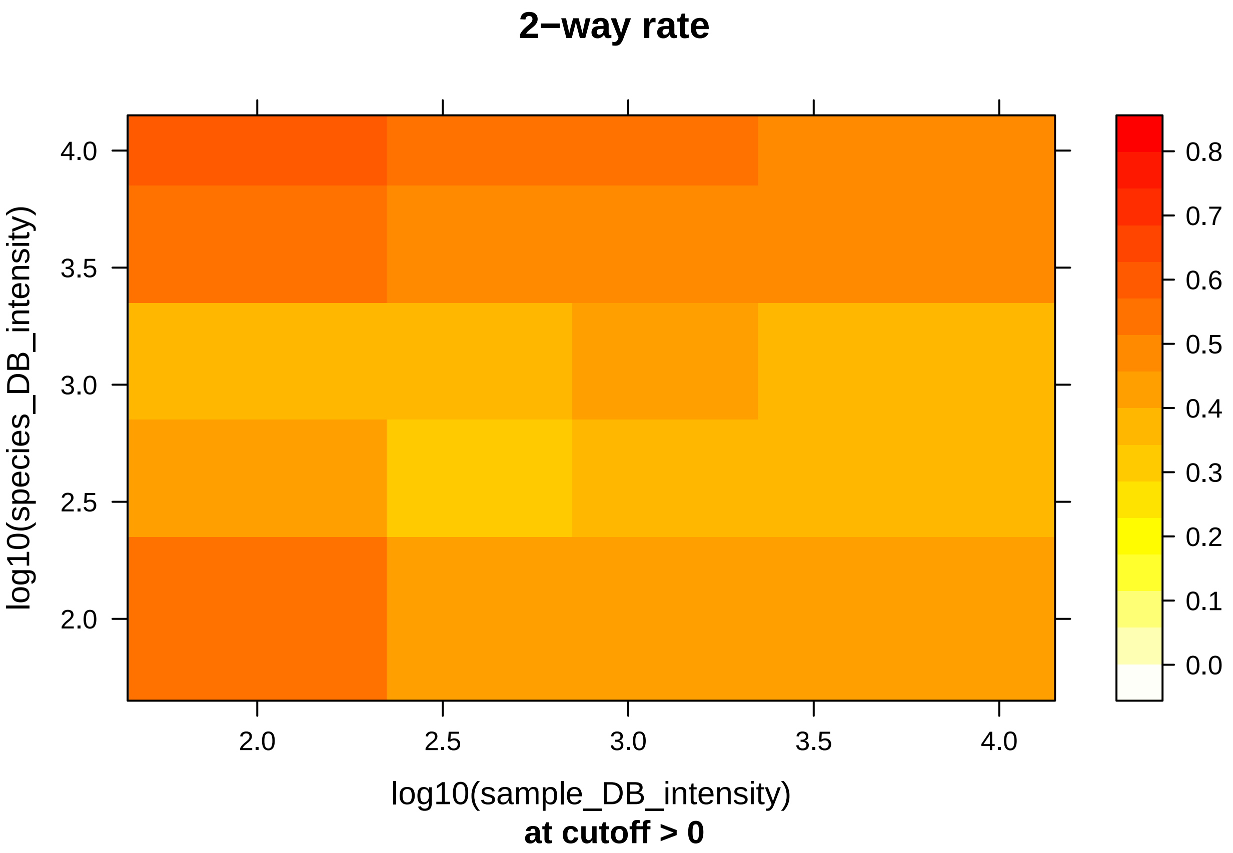


So far…

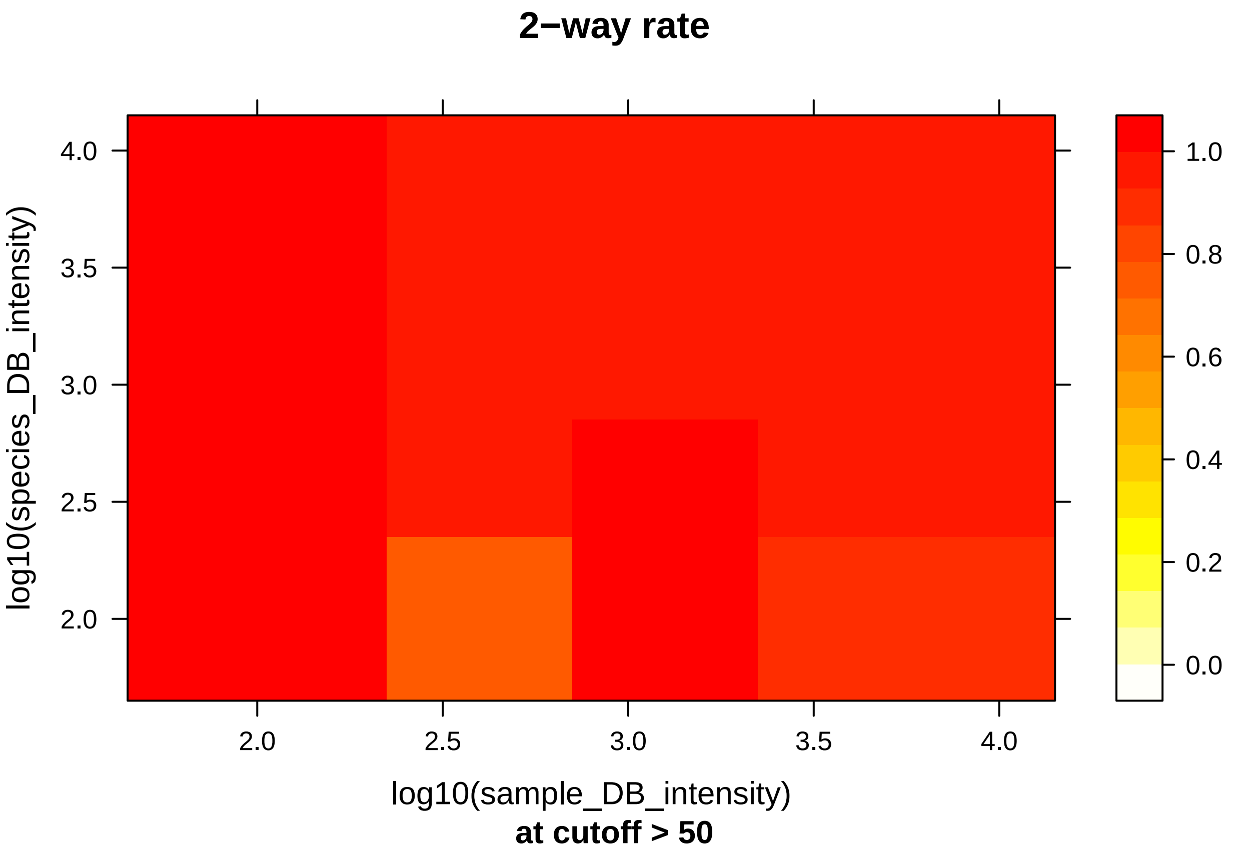
OK: something a bit odd is going on here, but basically it looks like a) it's all working OK more or less b) species DB intensity seems to make a bigger difference than sample effort/intensity..

To show this, collect all the aggregates in one:

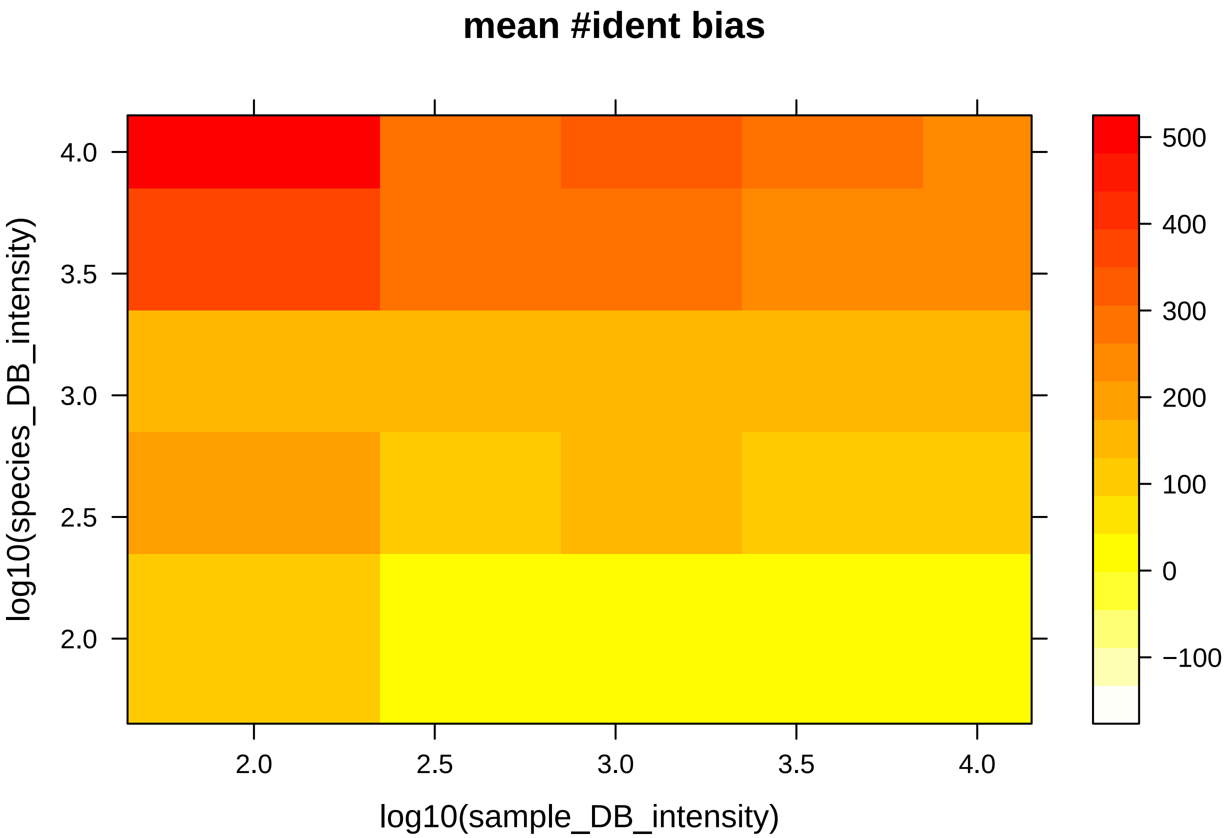
```
par(mfrow=c(2,3))
levelplot(two_way_rate ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps, col.regi
```



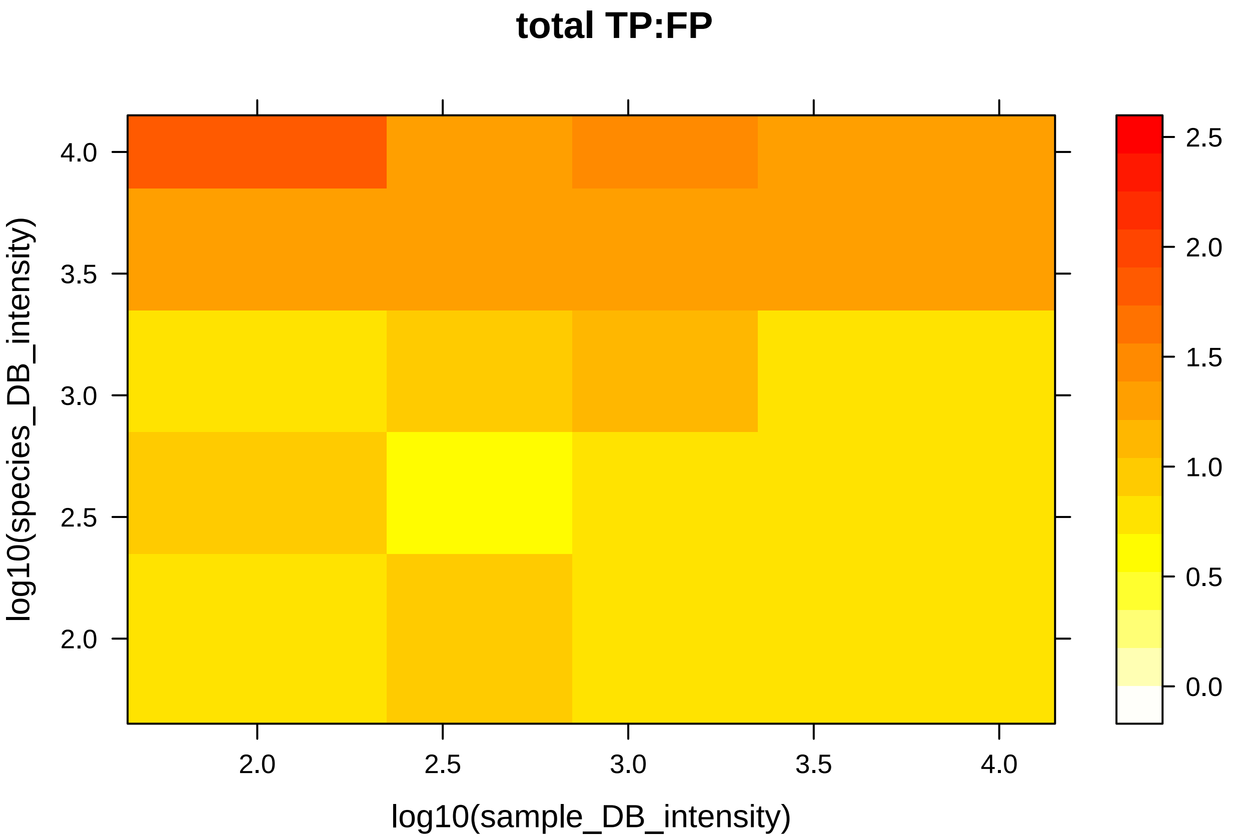
```
levelplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_rep
```



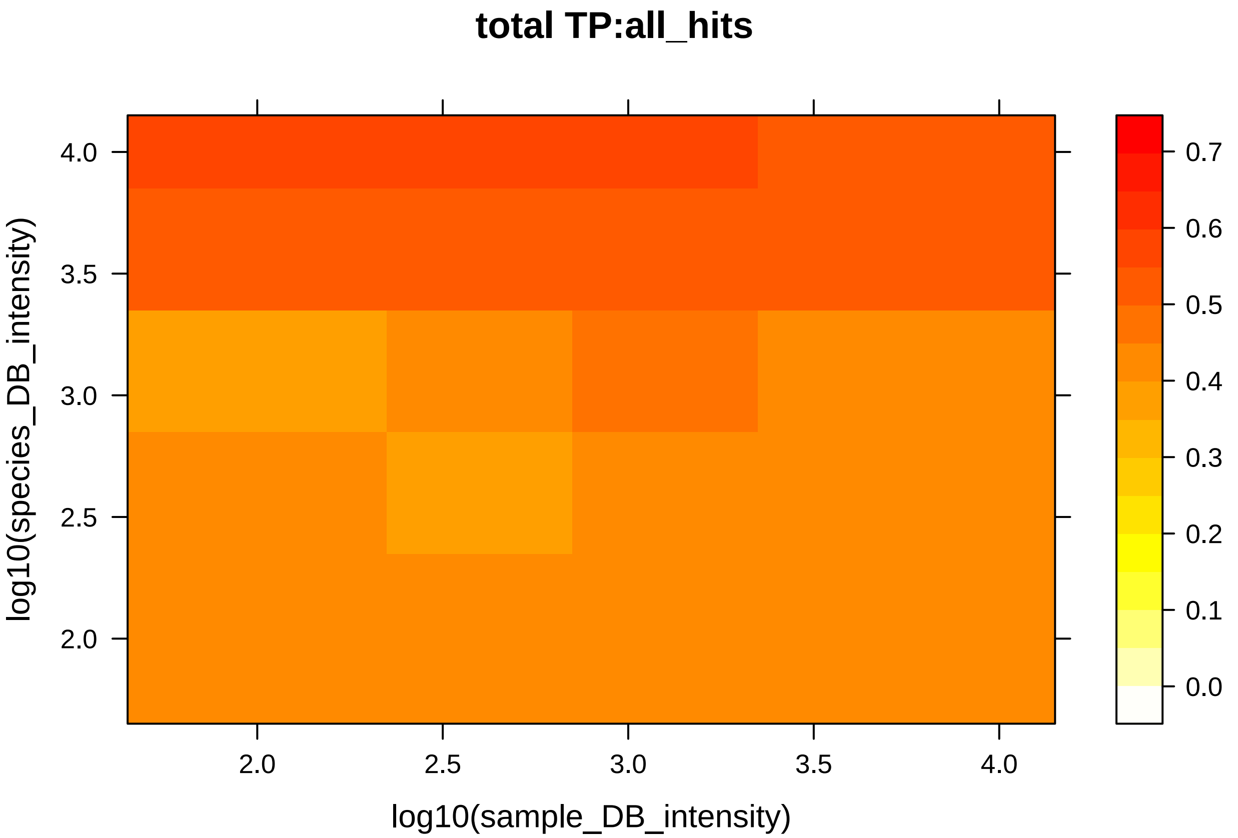
```
levelplot(mean_bias ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps, col.regions
```



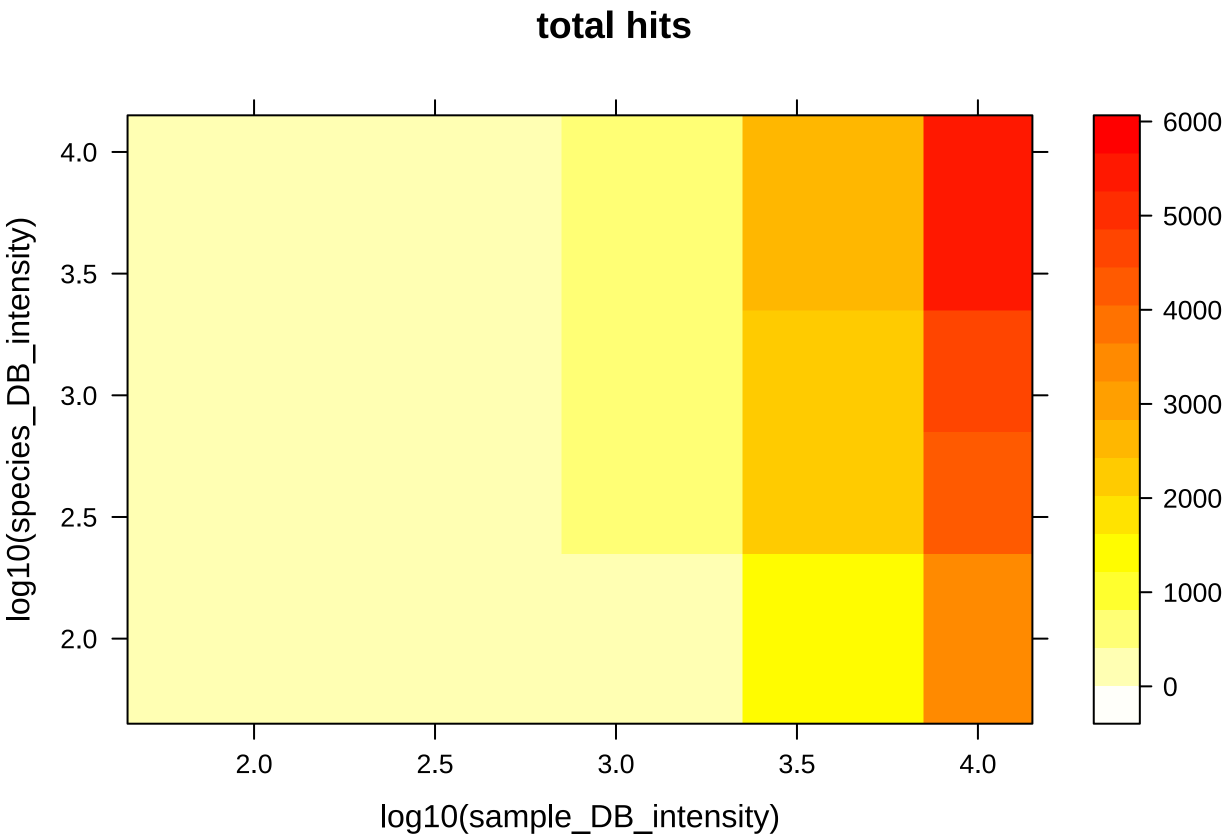
```
levelplot((one_way_TP + two_way_TP) / (one_way_FP + two_way_FP) ~ log10(sample_DB_intensity)*log10(speci
```



```
levelplot((one_way_TP + two_way_TP) / total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensit
```

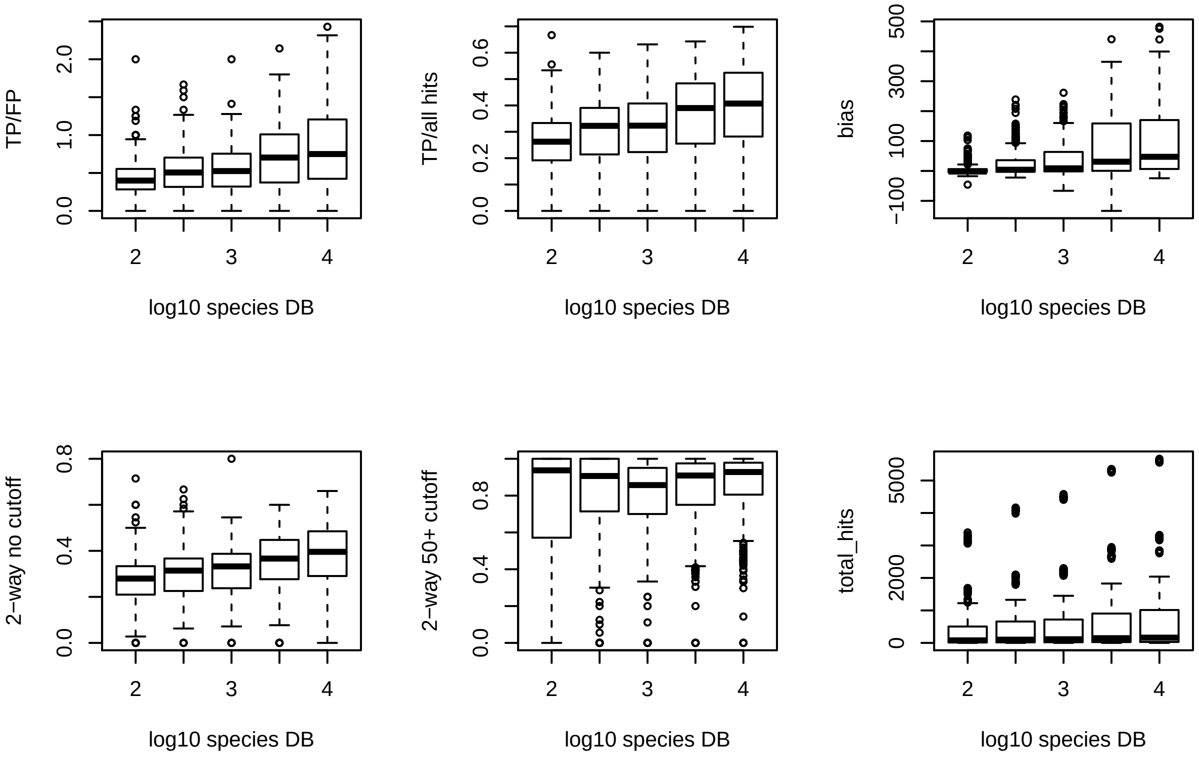


```
levelplot(total_hits ~ log10(sample_DB_intensity)*log10(species_DB_intensity),data=all_reps, col.region
```



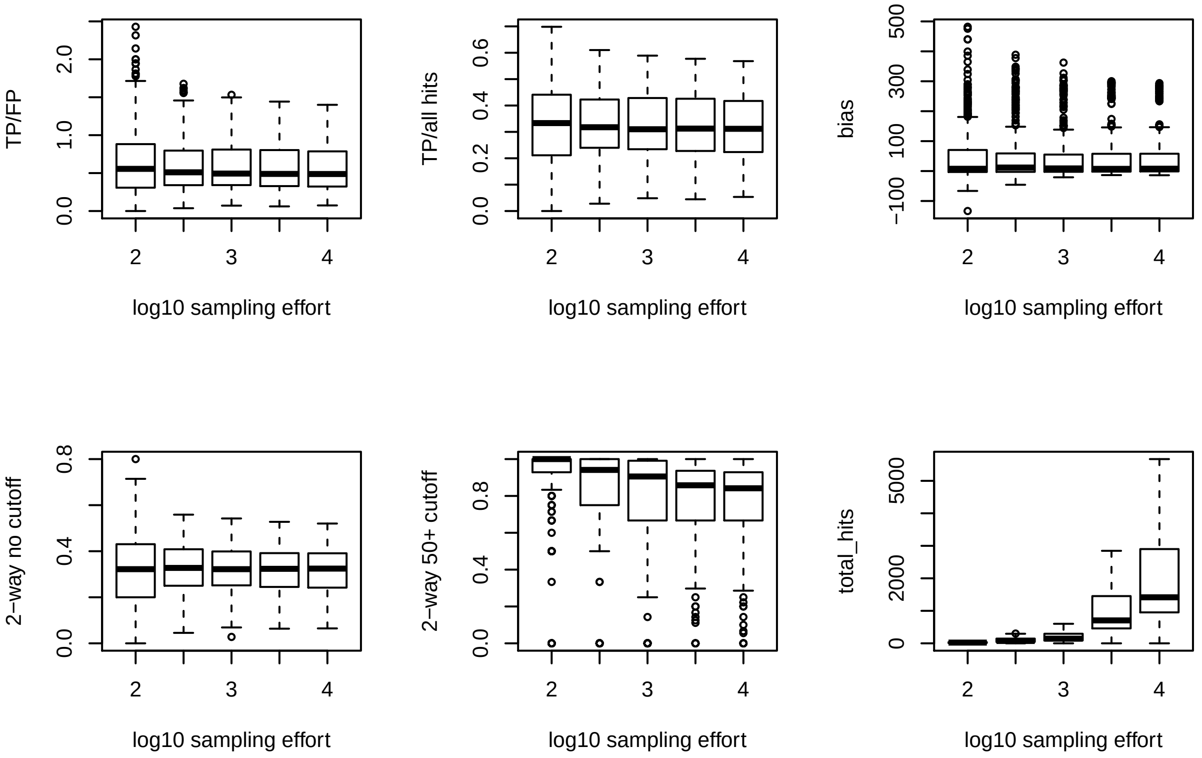
checking headline stuff; how does blast DB intensity affect outputs?:

```
par(mfrow=c(2,3))
boxplot((one_way_TP + two_way_TP) /(one_way_FP + two_way_FP) ~ log10(species_DB_intensity),data=all_reps
boxplot((one_way_TP + two_way_TP) /(total_hits) ~ log10(species_DB_intensity),data=all_reps, col.regions
boxplot(mean_bias ~ log10(species_DB_intensity),data=all_reps, col.regions = heat.colors(100)[length(hea
boxplot(two_way_rate ~ log10(species_DB_intensity),data=all_reps, col.regions = heat.colors(100)[length
boxplot(two_way_rate_with_cutoff ~ log10(species_DB_intensity),data=all_reps, col.regions = heat.colors
boxplot(total_hits ~ log10(species_DB_intensity),data=all_reps, col.regions = heat.colors(100)[length(h
```



how does resequencing (sampling) intensity affect the same?:

```
par(mfrow=c(2,3))
boxplot((one_way_TP + two_way_TP) /(one_way_FP + two_way_FP) ~ log10(sample_DB_intensity),data=all_reps
boxplot((one_way_TP + two_way_TP) /(total_hits) ~ log10(sample_DB_intensity),data=all_reps, col.regions
boxplot(mean_bias ~ log10(sample_DB_intensity),data=all_reps, col.regions = heat.colors(100)[length(hea
boxplot(two_way_rate ~ log10(sample_DB_intensity),data=all_reps, col.regions = heat.colors(100)[length(
boxplot(two_way_rate_with_cutoff ~ log10(sample_DB_intensity),data=all_reps, col.regions = heat.colors(
boxplot(total_hits ~ log10(sample_DB_intensity),data=all_reps, col.regions = heat.colors(100)[length(he
```



Conclusion

Overall it seems to be working fine, e.g. we get hits and length differences that tend to correct at low intensities (10E4 is a bloody small amount of sequencing..!)

The biggest determinant of overall success seems to be R4IDs database sampling effort, not resequencing effort. This is interesting.

The length bias cutoff (hits' length difference greater than 50 in favour of TP database for a posititve hit) seems to greatly improve accuracy at the expense of total number of hits. An optimal tradeoff will depend on the expected taxonomic distance between likely ID candidates.

Unsure what is driving the between-species differences, but the quality of the runs in terms of total yield and also q-scores is probably to blame, as well as genome size (since this affects coverage). Our target genomes here range from ~0.28 to 3.0Gbp, and our *maximum* R4IDs sequencing yields are in the 20Mbp (0.02Gbp) to 70Mbp zone; so we are sequencing much less than a 1x-coverage genome. Insofar as this works at all it is probably partly (largely?) due to bias; in the **A. thaliana** data we observed that plastid depths were ~350x even though total genome depth ~2x (max).

Note will need to know genome sizes more accurately if at all possible - only have C-values.