entrynyt

Differential gene expression under various stressful conditions in *Eulimnogammarus verrucosus* (Gerstfeldt, 1858), *E. cyaneus* (Dybowsky, 1874) and *Gammarus lacustris* Sars, 1863 assessed with RNA sequencing

Work in progress report

Polina Drozdova

February 8, 2019

Contents

1 Pipeline overview

Before we start, here is an overview of the pipeline used in the analysis presented below.

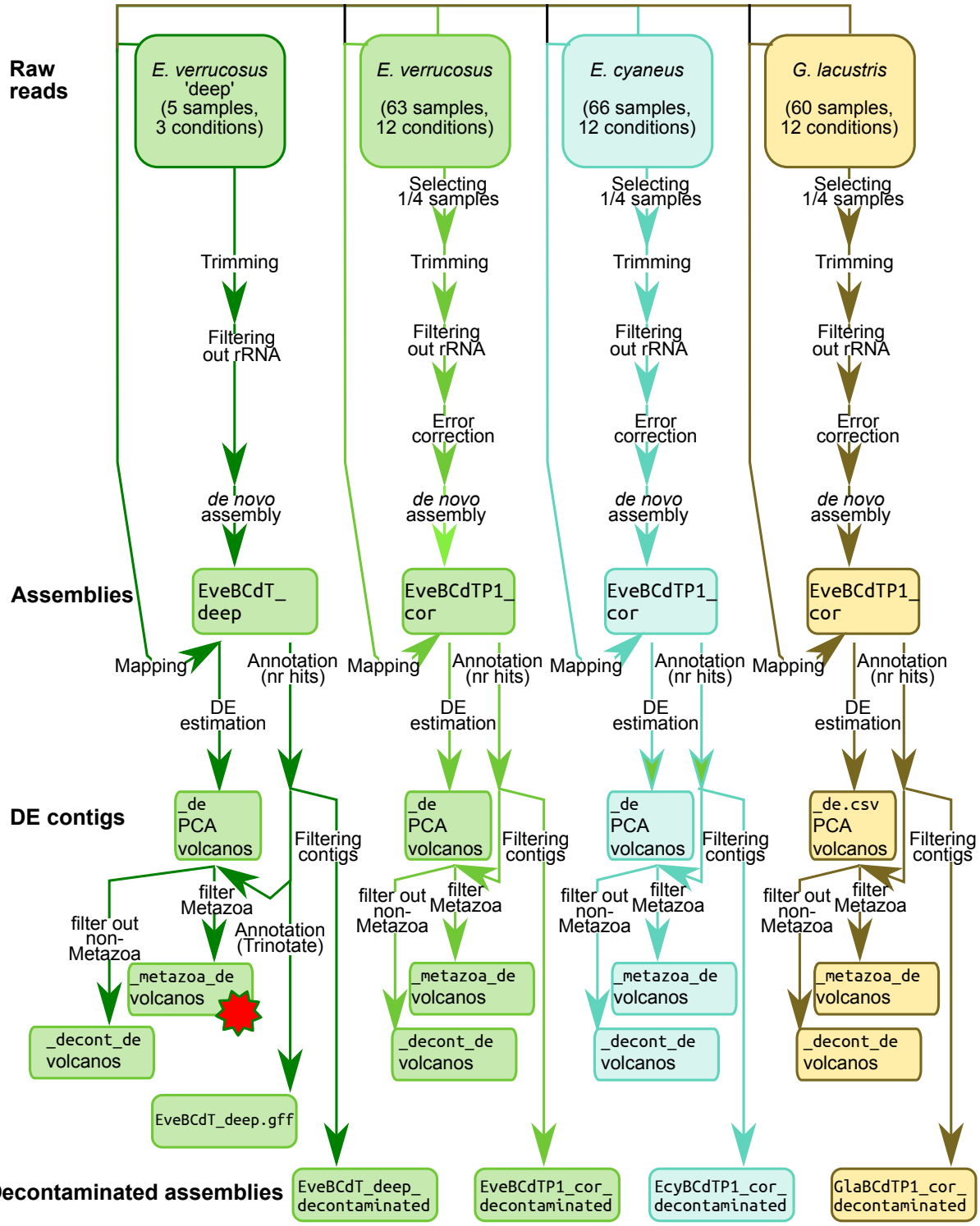

Figure 1: Pipeline overview (details below). The red asterisk marks the data in the attached tables.

2 Software

Disclaimer. I decided to put all information about the software used here for the sake of keeping all version information in one place. Thereafter in the text, the code used to run particular analyses is shown for reproducibility (at least an example), while most of the code used to create figures or tables is omitted for the sake of clarity. I do not think the following passage is an interesting reading, so please feel free to skip to Section 3.

Read quality control:

- FastQC (**FastQC**) v0.11.5;
- MuliQC (**MultiQC**) v1.2.

Trimming:

- trim_galore (**trim˙galore**) v0.4.1.

Read error correction:

- rcorrector (**Song2015**) (cloned from Github on Dec 18, 2017).

Assembly:

- Trinity (**Grabherr2011**) v2.5.1.

Assembly quality control, annotation and comparison:

- TransRate (**Smith-Unna2016**) v1.0.3;
- BUSCO (**Simao2015**) v3.0.2.
- Transdecoder 5.0.1.
- Trinotate v3.0.2.

Alignment and mapping:

- bowtie (**Langmead2012**) v2.1.0;
- standalone NCBI BLAST+ (**Camacho2009**) v2.2.28+;
- diamond (**Buchfink2014**) v0.9.10.111 and NCBI non-redundant protein sequence database of 10 October 2017;
- Salmon (**Patro2017**) v0.9.1.

Differential expression:

- DESeq2 (**Love2014**) v1.14.1;
- tximport (**Soneson2015**) v1.2.0.

Miscellaneous / presentation:

- faidx (**Shirley2015**) v0.5.3.1;
- R (**RCoreTeam2017**) v3.3.2;
- R packages: ggplot (**Wickham2009**) v2.2.1.

3 Data

The next sections are more technical. Biological results start later (section 7).

Here is an overview of the data we have. It is shown in Fig. 2 and Table 1, which basically summarize the same data but in a different way. So, we have deep sequencing data (hundreds millions reads) for *E. verrucosus* (control, Cd treatment and temperature) and 'usual' data (tens millions reads per sample) for multiple conditions (with appropriate controls).

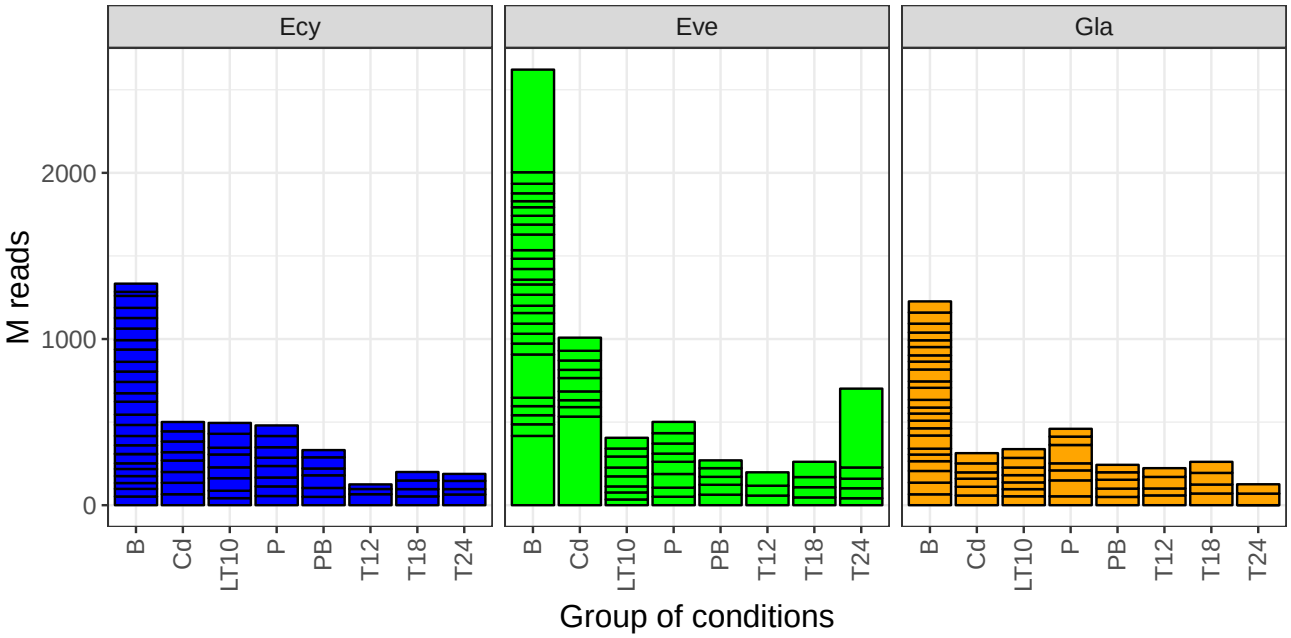

Figure 2: Amount of reads in all samples grouped by condition groups. Each sample is boxed. See Table 1 for abbreviations.

Please note that B stands for the control samples (6 °C in Baikal water). The samples labeled B12, B18 and (partly) B24 are parallel controls for the long-term exposure to rising temperature (0.8 °C per day); B3 and the remaining B24 samples are controls for 3h- and 24-long exposures to Cd (LC10) or temperature (LT10).

Table 1: Summary statistics for raw reads prior to any manipulations.

Species	Condition	Code	M reads	# samples
E. cyaneus	Control, 3h	B3	264	4
— // —	Control, zero	B6	207	4
— // —	Control for T12	B12	175	4
— // —	Control for T18	B18	185	5
— // —	Control for T24 / 24h	B24	440	8
— // —	Cd (LC10), 3h	Cd3	232	4
— // —	Cd (LC10), 24h	Cd24	234	4
— // —	Phenantrene, 3h	P3	244	4
— // —	Phenantrene, 24h	P24	236	4
— // —	Phenantrene control, 3h	PB3	152	3
— // —	Phenantrene control, 24h	PB24	180	3
— // —	Temperature, 12 degrees	T12	125	3
— // —	Temperature, 18 degrees	T18	200	4
— // —	Temperature, 24 degrees	T24	188	4
— // —	Temperature (LT10), 3h	10LT3	267	4
— // —	Temperature (LT10), 24h	10LT24	228	4
E. verrucosus	Control, 1h?	B1	677	2
— // —	Control, 3h	B3	258	4
— // —	Control, zero	B6	210	4
— // —	Control for T12	B12	230	4
— // —	Control for T18	B18	250	4
— // —	Control for T24 / 24h	B24	378	6
— // —	Cd (LC10), 1h?	Cd1	533	1
— // —	Cd (LC10), 3h	Cd3	243	4
— // —	Cd (LC10), 24h	Cd24	232	4
— // —	Phenantrene, 3h	P3	240	4
— // —	Phenantrene, 24h	P24	261	4
— // —	Phenantrene control, 3h	PB3	99	2
— // —	Phenantrene control, 24h	PB24	171	4
— // —	Temperature control	T6	617	1
— // —	Temperature, 12 degrees	T12	198	3
— // —	Temperature, 18 degrees	T18	261	4
— // —	Temperature, 24 degrees	T24	701	5
— // —	Temperature (LT10), 3h	10LT3	232	4
— // —	Temperature (LT10), 24h	10LT24	174	4
G. lacustris	Control, 3h	B3	176	4
— // —	Control, zero	B6	234	4
— // —	Control for T12	B12	265	4
— // —	Control for T18	B18	156	3
— // —	Control for T24 / 24h	B24	398	8
— // —	Cd (LC10), 3h	Cd3	154	3
— // —	Cd (LC10), 24h	Cd24	160	4
— // —	Phenantrene, 3h	P3	207	4
— // —	Phenantrene, 24h	P24	252	4
— // —	Phenantrene control, 3h	PB3	90	2
— // —	Phenantrene control, 24h	PB24	153	3
— // —	Temperature, 12 degrees	T12	223	4
— // —	Temperature, 18 degrees	T18	261	4
— // —	Temperature, 24 degrees	T24	126	3(?)
— // —	Temperature (LT10), 3h	10LT3	200	4
— // —	Temperature (LT10), 24h	10LT24	132	3

— // —, the same as above.

M reads, total number of sequence reads / 10^6.

samples, number of samples for this condition (deeply sequenced among them).

4 Quality control

The first thing that should be done with any NGS reads is subjecting them to quality control. All samples were subjected to FastQC analysis (quite a standard procedure, `fastqc Sample/*fq.zip -o .`) and then summarized with MultiQC (`multiqc .`). The results shown here come from concatenated R1/R2 files for each Sample + condition and only for *E. verrucosus* deep sequencing reads, but they adequately represent the overall picture.

Let us look at metrics that reflect purely technical characteristics of the library, quality scores and adapter content. The quality is very good, no need to adjust anything (the greener the better, and our data do not even touch the yellow region; Fig. 3A). All sequences were 100 base pairs long (not shown). So far, so good, but if we look at the adapter content (Fig. 3B), we'll see a significant elevation starting at *ca.* 30 bp and then rising to almost 20% towards the end of the reads. It should not have happened in ideal case (compare with a good FastQC report example). The only explanation I came up with is that the insert size turned out to be less than expected (Fig. 3C). So, we need to get rid of these adapter fragments, or they can significantly slow down and worsen *de novo* assembly.

Figure 3: Characteristics of *E. verrucosus* sequencing reads. Shown is MultiQC summary of FastQC analysis.

There are other characteristics that reflect more 'biological' parameters of the data. The plot of GC content shows a clearly multimodal distribution (Fig. 4A), which is quite expected for an RNA sequencing library (while for genome sequencing we would expect a quasi-normal distribution). But GC-poor peaks (for example, the one around 30% GC) might be giving away the presence of bacteria. We'll check it later (Sections 5.2 and 7). Finally, FastQC can estimate

the amount of repeated (overrepresented) sequences. In R1 reads, there are 5-10% of those, while in R2 reads, about one quarter of the. This difference between R1 and R2 reads suggests that the data are strand specific (but better check it later). In addition, FastQC provides a list of top 25 overrepresented sequences. They are not shown for the sake of clarity, but quick web BLAST identified most sequences checked as Gammaridea rRNAs (including mitochondrial mRNAs). So, we definitely need to remove rRNA sequences before the assembly.

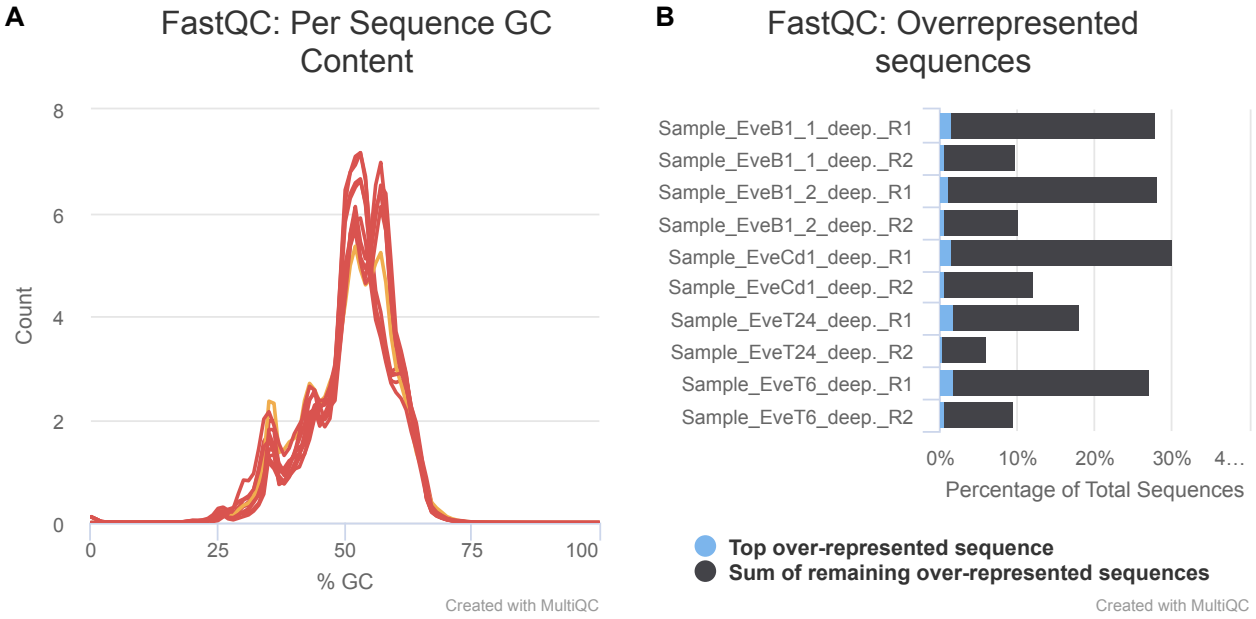

Figure 4: Characteristics of *E. verrucosus* sequencing reads continued. Shown is MultiQC summary of FastQC analysis.

We can finally start working on the data.

5 Preparing reads for *de novo* assembly

5.1 Trimming

Trimming was done with `trim_galore` (with a bit of bash tricks):

```
find  -wholename "*deep*/*_R1.fastq.gz" | cut -d "R" -f1 | parallel -j 10 \
$apps/trim_galore_zip/trim_galore --illumina --paired --fastqc --trim1    \
-o trimmed / {}R1.fastq.gz {}R2.fastq.gz
```

Data for *E. verrucosus* deep sequencing reads are shown in this figure and thereafter, unless indicated otherwise. However, all the other reads were processed with the same pipeline, and results were similar.

In this command, `trim_galore` runs fastqc. According to MultiQC summary, *no samples found with any adapter contamination >0.1%*. Thus, trimming worked, let's go on.

5.2 Filtering

We have solved the adapter problem, but rRNA sequences are still there. To get rid of rRNA I tried sortmerna (**Kopylova2012**), which should be tailored right for this purpose, but without much success. To be honest, I couldn't get it running on my data and was deeply unsatisfied with the manual.

So, I used bowtie2 (**Langmead2012**). The latest version of bowtie2 is a very powerful tool. But before this, we need a database of (ideally) rRNA sequences of our species. Of course, we do not have one, so the following set rRNA sequences were used to sort out their *Eulimnogammarus* or *Gammarus* homologs:

- KF690638.1: *Eulimnogammarus verrucosus* mitochondrion, complete genome: 14410–13791 (12S);
- KF690638.1 *Eulimnogammarus verrucosus* mitochondrion, complete genome: 12757–13738 (16S);
- AF419224.1 *Eulimnogammarus obtusatus* 18S small subunit ribosomal RNA gene, complete sequence;
- AF419223.1 *Parapallasea lagowski* 18S small subunit ribosomal RNA gene, complete sequence
- AY926773.1 *Eulimnogammarus verrucosus* 18S ribosomal RNA gene, partial sequence (1–747);
- EF582973.1 *Gammarus nekkensis* voucher 511 28S ribosomal RNA gene, partial sequence (1–1333);
- FJ422963.1 *Gammarus wilkitzkii* 18S ribosomal RNA gene, partial sequence; internal transcribed spacer 1, 5.8S ribosomal RNA gene, and internal transcribed spacer 2, complete sequence; and 28S ribosomal RNA gene, partial sequence (1–374);
- JF965799.1 *Gammarus sp.* 12 ZH-2011 voucher SLOTAD2 28S ribosomal RNA gene, partial sequence (1–1265);
- KF586544.1 *Dorogostaiskia parasitica hanajevi* voucher MZH:53197 28S ribosomal RNA gene, partial sequence (1–965);
- KY617162.1 *Niphargus dimorphus* isolate NA146 28S ribosomal RNA gene, partial sequence (1–1161);
- KT180191.1 *Jesogammarus hebeiensis* voucher IZCASIA1291 28S ribosomal RNA gene, partial sequence (1–1268);
- DQ464759.1 *Hyalella azteca* haplotype 118 28S ribosomal RNA gene, partial sequence (1–1389).

By the way, is it true that there are no complete 28S sequences from amphipods in the public databases? 1.4 kb of the *H. azteca* sequence was the longest I was able to find.

These sequences were joined in a multifasta file and indexed as a bowtie2 database: `bowtie2-build Gammaridae_rRNA.fa Gamm_rrna_db`, and then all deep sequencing reads were aligned to this reference:

```
for sample in `ls $reads/trimmed/*R1*.fq.gz`
do dir="$reads/trimmed/"; \
base=$(basename $sample "R1_val_1.fq.gz"); \
bowtie2 -x $DE/Gamm_rrna_db \
-1 ${dir}/${base}R1_val_1.fq.gz \
-2 ${dir}/${base}R2_val_2.fq.gz \
--very-sensitive-local --met-file clean4/${base}_metrics.txt \
--un-conc-gz clean4/${base}_clean.fastq.gz --al-conc-gz \
clean4/${base}.rrna.fastq.gz -p 11 -S clean4/${base}.sam
done
```

After this, a new FastQC check clearly shows that the number of overrepresented sequences dropped (at Fig. 5, clean reads are shown in darker colors while data for raw reads are shown in lighter colors). However, I was not able to remove all repeated sequences. The remaining repeats still blast to Gammaridae rRNA, but couldn't get much better (the results shown are after three rounds of expanding the rRNA database with new sequences), so I decided to stop at this point and try assembling.

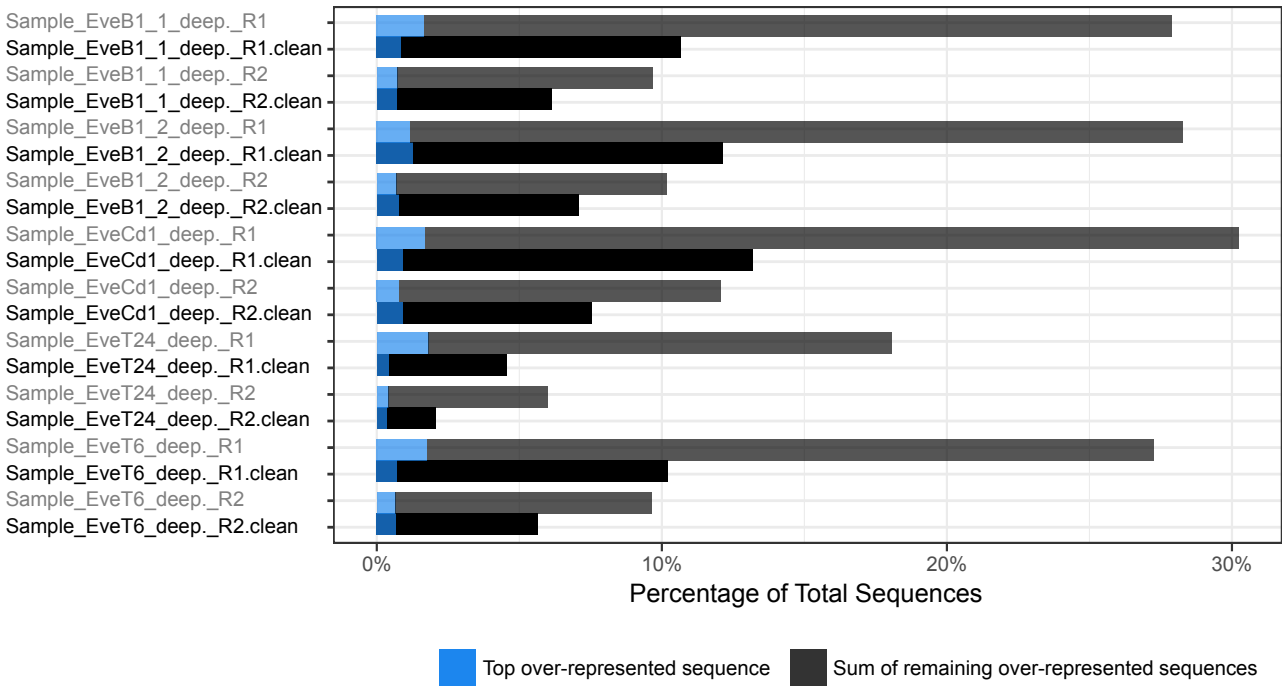

Figure 5: Overrepresented sequences in *E. verrucosus* sequencing reads after trimming and sorting out rRNA. This plot was reproduced with the ggplot2 package for R based on the MultiQC summary of FastQC analysis and designed to roughly reproduce the style for easier comparison with Fig. 4A.

It is worth noting that after filtering out rRNA, the low C/G peaks are almost gone, and the distribution became more symmetric (Fig. 6).

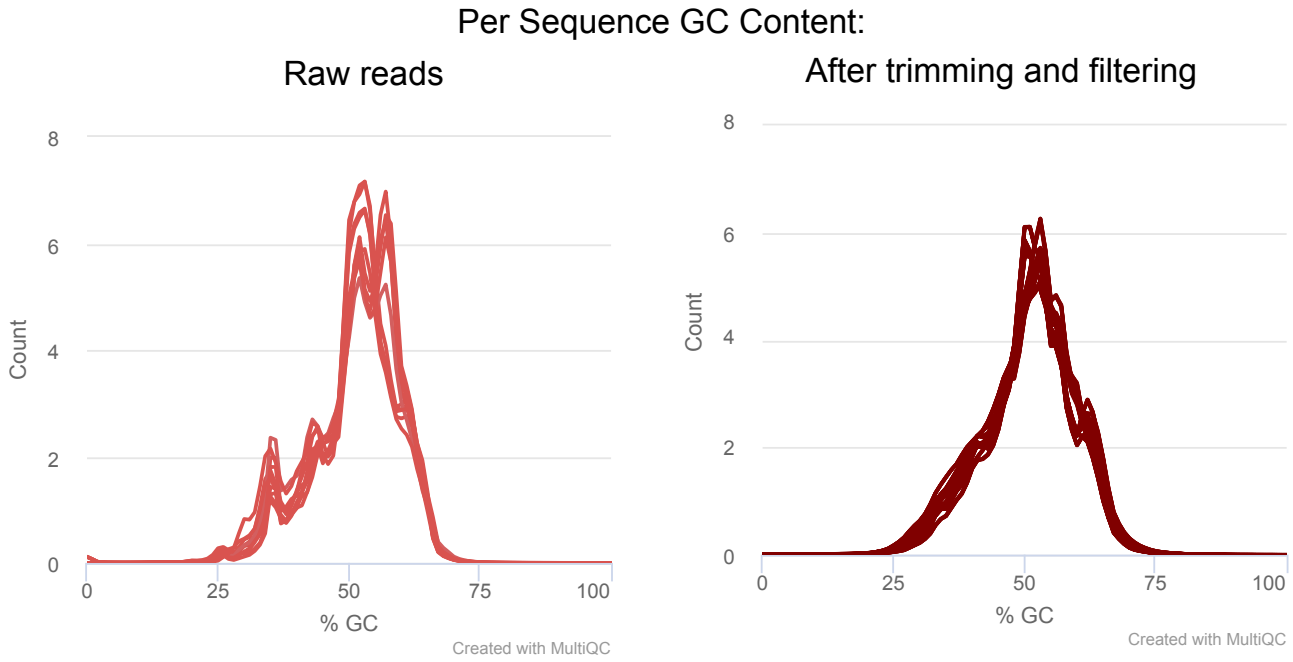

Figure 6: Distribution of G/C content in *E. verrucosus* sequencing reads before and after trimming and sorting out rRNA. The plots were created with MultiQC. A reproduces 4A.

Now we have less data but I believe that they are now better suited for assembly. Absence of adapters and less rRNA should speed up the assembly and probably improve its quality. The first assumption can be easily proved, as assembly of raw reads repeatedly failed with resources I have access to (`12 cores, 64 Gb RAM, Ubuntu 14.04.1, kernel 4.4.0-81-generic`). For this reason, the second assumption could not be tested. To be honest, failure was observed with the previous version of Trinity (2.2), but do not really want to waste time for this check...

5.3 Read error correction

In the case of separate assemblies for each species (`EveBCdTP1_cor, EcyBCdTP1_cor`, GlaBCdTP1_cor), I applied read error correction; otherwise, the assembly run >2 weeks. Here is an example command:

```
for sample in ‘ls $reads/clean4/Sample_Ecy*clean.fastq.1.gz‘
do dir="$reads/clean4/"
base=$(basename $sample "1.gz")
perl $apps/rcorrector/run_rcorrector.pl \
-1 ${dir}/${base}1.gz -2 ${dir}/${base}2.gz \
-t 9 -od $reads/corr
done
```

6 *De novo* transcriptome assembly

6.1 Assembly based on the deep *E. verrucosus* sequencing reads

The main *de novo* assembly was based on *E. verrucosus* 5 deep-sequenced samples (3 condition groups: control, Cd, elevated temperature; see Table 1). I selected these data to make the first assembly, as it should be less heterogeneous than an assembly of all the remaining data (5 animals *vs.* ∼60 ones).

The assembly was done with Trinity.

Now, let's return to the clear difference in the number of repeated sequences between R1 and R2 reads. These data suggests that the library was generated with a strand specific protocol but do not prove it. However, Trinity provides a nice script to check the strandedness of the assembly (https://github.com/trinityrnaseq/trinityrnaseq/wiki/Examine-Strand-Specificity).

So, I assembled the data without specifying library type:

```
$apps/trinityrnaseq-Trinity-v2.4.0/Trinity --seqType fq \ --max_memory 50G \
--left $reads/clean4/Vr*clean.fastq.1.gz --right $reads/clean4/Vr*clean.fastq.2.gz \
--CPU 11 --output Ever_deep_trinity --bypass_java_version_check
```

Then, the assembly to the analysis of library type (Fig. 7A). Comparison of this result with three examples provided by Trinity (Fig. 7B-D) clearly shows that our data are probably strand-specific but assembled as non-strand-specific.

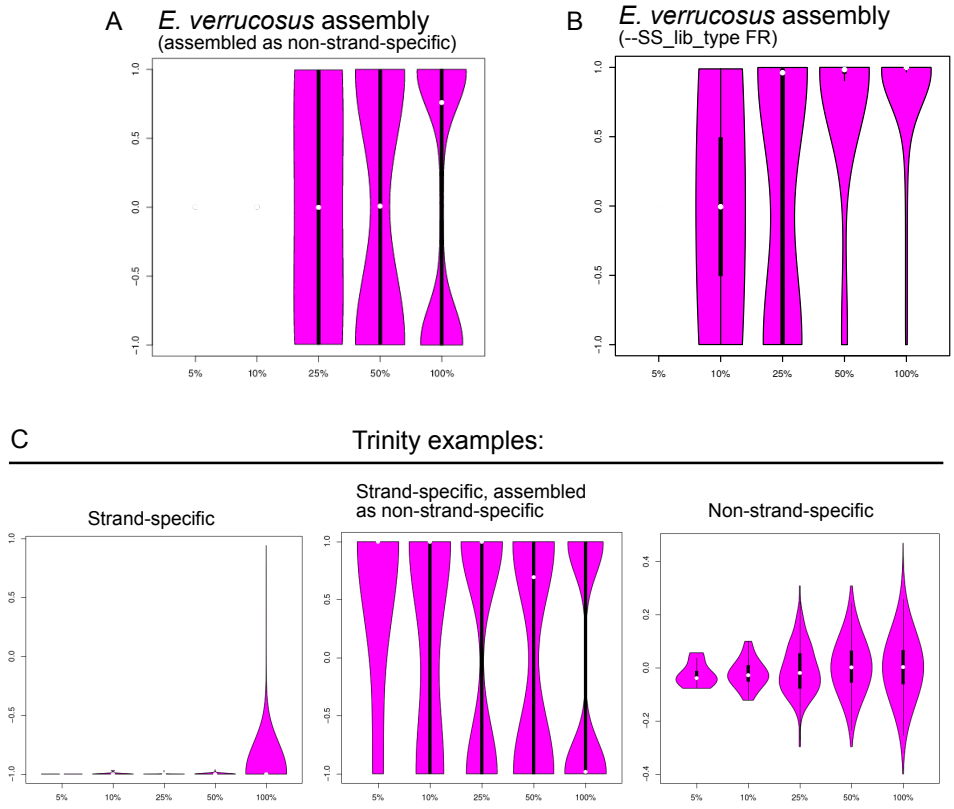

Figure 7: Comparison of *E. verrucosus* assemblies and examples provided by Trinity by distribution of read orientation (the statistics is) `(plus_strand - minus_strand) / total`. Top cumulative values are shown along the horizontal axis. For more information on the script please see the help page.

Thus, our data should be assembled as strand-specific. Illumina protocols should correspond to FR library type for Trinity.

```
$apps/trinityrnaseq-Trinity-v2.4.0/Trinity --seqType fq  --max_memory 50G \
 --left $reads/clean4/Vr*clean.fastq.1.gz --right $reads/clean4/Vr*clean.fastq.2.gz \
 --CPU 11 --output Ever_deep_trinity --bypass_java_version_check \
 --SS_lib_type FR
```

Now the distribution of reads (Fig. 7B) looks more like it should (even though it does not look as clean as the example provided at Figure 7C, and I don't know why; the only idea I have is that I used only one set of the initial R1/R2 reads (not all 5 used for the assembly) for alignment to save time).

6.2 Separate assemblies for each species (without deep sequencing reads)

We made separate assemblies for each of the 3 species with about 1/4 of the reads for each species (the first sample for each condition was selected). Reads were subjected to rcorrector to reduce complexity (see section 5.3). An example command for *E. cyaneus* is shown here (analogous commands were run for each species)

```
$apps/trinityrnaseq-Trinity-v2.4.0/Trinity --seqType fq --max_memory 50G \
 --left $reads/corr/Sample_Ecy*_1*fastq.1.cor.fq.gz \
 --right $reads/corr/Sample_Ecy*_1*fastq.2.cor.fq.gz \
 --CPU 9 --output EcyBCdTPB1_cor_trinity --bypass_java_version_check \
 --SS_lib_type FR
```

6.3 Assembly quality control

Basic assembly metrics were estimated with TransRate and BUSCO (Table 2). For each species, our assemblies look better than the published ones (**Naumenko2017**), as they contain more genes of the Arthropoda gene set. In addition, our assemblies are approximately 10x large, which is quite expected, as we had more raw data.

However, we should note that the published assemblies were filtered for contigs with top blast hits outside of Arthropoda (**Naumenko2017**). To make proper comparison, we should similarly filter out contamination and perform the comparison again. Filtering was done on the basis of annotation results (see Section 7) with bash / R / python faidx. The results are included into Table 2.

Table 2: Summary statistics of assemblies obtained compared with the published *E. verrucosus*, *E. cyaneus* and *G. lacustris* transcriptome assemblies (`gam42.1`, `gam42.3` and `gam3.1`, respectively; **Naumenko2017**).

Assembly	n_seqs	largest	n_bases	%GC	%BUSCO		
					Complete	Fragmented	Missing
EveBCdT_deep	960,103	22,898	404M	45.1	89.2	8.6	2.2
EveBCdT_deep_decontaminated	861,420	22,898	365M	45.4	88.8	8.9	2.3
EveBCdTPB1_cor	790,102	22,553	359M	43.2	91.8	6.4	1.8
EveBCdTP1_cor_decontaminated	691,272	22,553	316M	43.6	91.2	6.8	2
gam42.1	102,163	16,836	67M	45.5	71	12.9	16.1
EcyBCdTPB1_cor	1,070,909	23,639	482M	43.5	92.3	5.6	2.1
EcyBCdTP1_cor_decontaminated	958,457	23,639	430M	43.8	92	5.8	2.2
gam42.3	81,926	15,873	54M	44.5	64	14.1	21.9
GlaBCdTPB1_cor	875,485	23,666	404M	43.8	92.9	5.8	1.3
GlaBCdTP1_cor_decontaminated	825,150	23,666	379M	44	92.5	6	1.5
gam3.1	73,340	13,612	56M	45.1	68	14.1	17.9

n_seqs, total number of contigs; largest, length of the larges contig (bases); n_bases, total number of bases in the assembly; mean_len, total contig length; % BUSCO, percent of conservative genes from `arthropoda_odb9` database found in the corresponding assembly with BUSCO.

7 Transcriptome annotation

7.1 Diamond

To get information about the proteins encoded by the contigs, I annotated the assembly with diamond (it is functionally analogous to blast but runs incomparably faster).

```
diamond blastx --db $databases/nrd.dmnd \
--taxonmap $databases/prot.accession2taxid.gz
--query $assemblies/EveBCd6T24T_deep_FR.fasta \
--out EveBCd6T24T_deep_FR.diamond.tsv --threads 9 \
--sensitive --max-target-seqs 1 \
-f 6 qseqid sseqid pident length mismatch gapopen qstart qend \
sstart send evalue bitscore staxids stitle salltitles
```

It is worth noting that the newest diamond versions allow not only blast-like alignment but also quite easy taxonomic assignation. About 22% of the sequences were assigned to some hits from NCBI with this methods. Then, it was possible to summarize taxonomic assignations with the NCBI Taxonomy browser (https://www.ncbi.nlm.nih.gov/Taxonomy/TaxIdentifier/tax_identifier.cgi) and build an approximate representation of the organisms to which the

contigs probably belong. For this, one of the three highest taxonomy levels was chosen for each contig with an in-house R script.

Most of the contigs (>93%) belong to Eukaryota (Fig. 8, leftmost bar); thus, bacterial contamination is quite low. Quite surprisingly, though, about 40% of the sequences belong to Alveolata / Ciliophora, depending on the taxonomic level we cut at (middle and right bars, respectively), meaning that almost half of the identifiable sequences comes from ciliates.

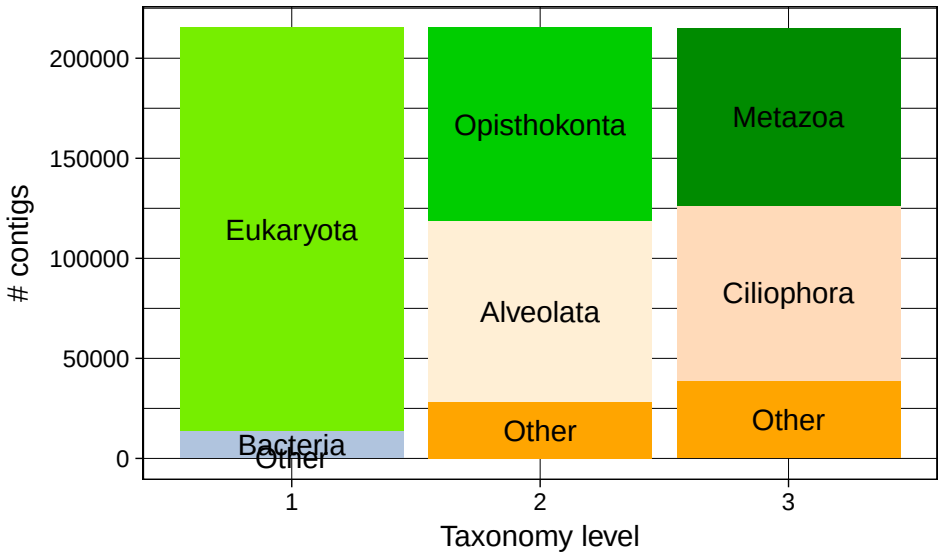

Figure 8: Taxonomic assignation of contigs of the `EveBCdT_deep` assembly. Groups with <5% representation were combined to 'Other'.

Other *Eulimnogammarus* assemblies are similar in terms of contamination, while the *G. lacustris* assembly has a smaller proportion of Ciliophora reads (Fig. 9). Of course, these data do not allow us to draw any conclusions, but if this difference (larger proportion of ciliates in Baikal species) reflects some natural tendency based on, say, body size or ecological characteristics, it might be quite interesting.

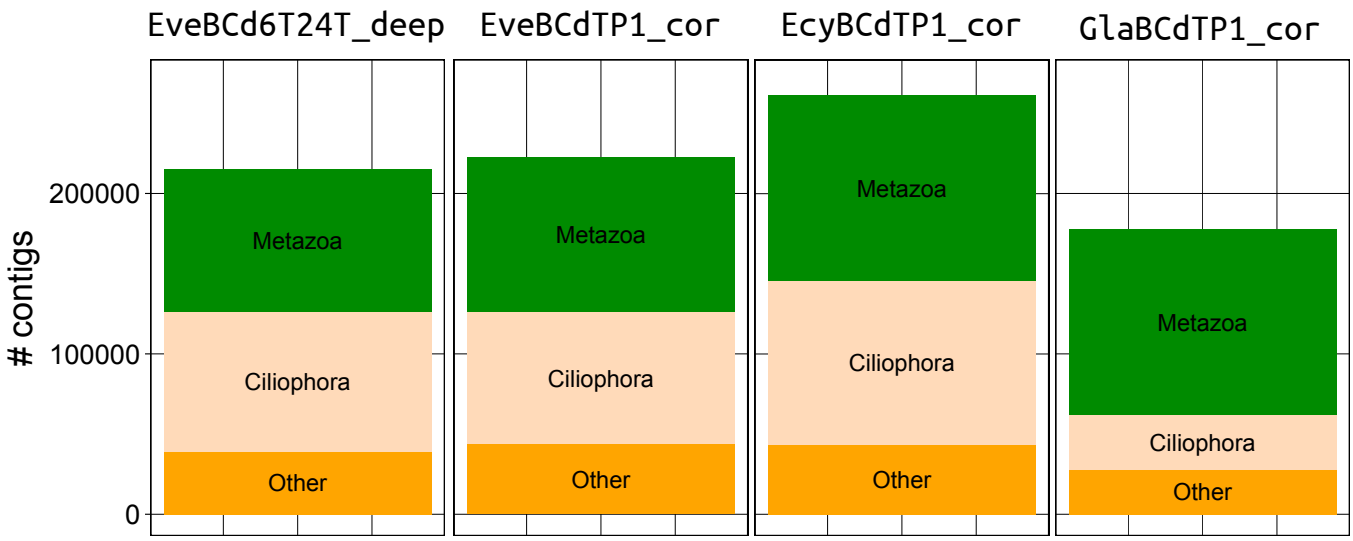

Figure 9: Taxonomic assignation of contigs. Groups with <5% representation were combined to 'Other'. The leftmost column is the same as the rightmost column at Fig. 8

So far, I haven't done anything with the contigs originated from ciliates or with those not identified, but let us remember this information and return to it when speaking about differential expression.

7.2 Trinotate

I also tried the full Trinotate pipeline, which is suggested by the Trinity authors. It includes blasting to uniprot and adds GO terms for some contigs. Diamond results were also included here (but unfortunately, Trinotate includes only best hit and not the full title).

```
export TRINOTATE_HOME='$apps/Trinotate-3.0.2/'

blastx -query EveBCd6T24T_deep_FR.fasta -db $TRINOTATE_HOME/admin/uniprot_sprot.pep \
-num_threads 8 -max_target_seqs 1 -outfmt 6 > blastx.outfmt6

#Transdecoder: find long ORFs
$apps/TransDecoder-5.0.1/TransDecoder.LongOrfs -t EveBCd6T24T_deep_FR.fasta
$apps/TransDecoder-5.0.1/TransDecoder.Predict -t EveBCd6T24T_deep_FR.fasta

#blast to common databases
blastp -query EveBCd6T24T_deep_FR.fasta.transdecoder_dir/longest_orfs.pep \
 -db $TRINOTATE_HOME/admin/uniprot_sprot.pep -num_threads 10 -max_target_seqs 1 \
 -outfmt 6 > blastp.outfmt6
hmmscan --cpu 8 --domtblout
$apps/signalp-4.1/signalp -f short -n signalp.out \
 EveBCd6T24T_deep_FR.fasta.transdecoder.pep >EveBCd6T24T_deep_FR.fasta.signalp.out
$apps/tmhmm-2.0c/bin/tmhmm --short < EveBCd6T24T_deep_FR.fasta.transdecoder.pep \
 > EveBCd6T24T_deep_FR.fasta.tmhmm.out
$apps/trinityrnaseq-Trinity-v2.4.0/util/support_scripts/get_Trinity_gene_to_trans_map.pl \
EveBCd6T24T_deep_FR.fasta >  EveBCd6T24T_deep_FR.fasta.gene_trans_map
```

```
#initiated an SQL database
$TRINOTATE_HOME/Trinotate $TRINOTATE_HOME/admin/Trinotate.sqlite init \
 --gene_trans_map EveBCdT_deep_FR.fasta.gene_trans_map \
 --transcript_fasta EveBCd6T24T_deep_FR.fasta \
 --transdecoder_pep EveBCd6T24T_deep_FR.fasta.transdecoder.pep
#loaded it with data
$TRINOTATE_HOME/Trinotate $TRINOTATE_HOME/admin/Trinotate.sqlite \
 LOAD_swissprot_blastp  blastp.outfmt6
$TRINOTATE_HOME/Trinotate $TRINOTATE_HOME/admin/Trinotate.sqlite \
 LOAD_tmhmm EveBCd6T24T_deep_FR.fasta.tmhmm.out
$TRINOTATE_HOME/Trinotate $TRINOTATE_HOME/admin/Trinotate.sqlite \
 LOAD_signalp EveBCd6T24T_deep_FR.fasta.signalp.out
$TRINOTATE_HOME/Trinotate $TRINOTATE_HOME/admin/Trinotate.sqlite \
 LOAD_swissprot_blastx blastx.outfmt6
$TRINOTATE_HOME/Trinotate $TRINOTATE_HOME/admin/Trinotate.sqlite \
 LOAD_custom_blast --outfmt EveBCd6T24T_deep_FR.diamond.tsv --prog blastp --dbtype nr

#And finally, we have a report
$TRINOTATE_HOME/Trinotate $TRINOTATE_HOME/admin/Trinotate.sqlite report \
 >EveBCd6T24T_deep_FR.gff
```

So, Trinotate provided us with a gff file for the assembly (I have not used it so far, but it might still be useful).

8 Transcript quantification

8.1 Common reference assembly

The first approach I used was to use the available *E. verrucosus* deep read assembly as the common reference.

Trinity has several wrapper scripts for transcript quantification and differential expression, but I decided to use one of the tools, Salmon, directly, as the latest version of this tool is newer than that provided by Trinity and also provides finer tuning.

Salmon is fast (quantification took 30 minutes per sample compared with 8h per sample for bowtie2-based code).

First, the assembly was indexed:

```
$apps/Salmon-latest_linux_x86_64/bin/salmon index
-t $assemblies/EveBCd6T24T_deep_FR.fasta -i EveBCd6T24T_deep.index
```

Then, all raw reads were mapped to this index with `salmon quant`:

```
for sample in ‘ls $transcriptome_raw/Sample*/*R1.fastq.gz‘;
 do base=$(basename  $sample _R1.fastq.gz); dir=$(dirname $sample); \
$apps/Salmon-latest_linux_x86_64/bin/salmon quant -i EveBCd6T24T_deep.index/ -l ISF \
-1 $dir/${base}_R1.fastq.gz -2 $dir/${base}_R2.fastq.gz -p 9 -o gc/$base.quant.s --gcBias; \
done
```

The `ISF` option defines library type, and using `--gcBias` is recommended by the authors of the program.

Even though Salmon uses quasi-mapping instead of real alignment, it also outputs statistics on the proportion of reads of each particular sample "aligned" (mapped) to the reference assembly. These parameters are summarized at Fig. 10. The results (about 50% of reads are aligned in each case, with the best alignment seen for *E. verrucosus*, lower alignment rate for *E. cyaneus* and the lowest alignment rate for *G. lacustris*), nicely corresponds to the 40-50% alignment rate to the rRNA database seen on the cleanup stage and phylogenetic relationship of the species of interest. I believe that these results mean that we can use quantification by this assembly to interpret biological responses to stimuli in both *Eulimnogammarus* species with almost equal confidence, while in the case of *G. lacustris* we should be more cautious, even though these data are still useful for comparison.

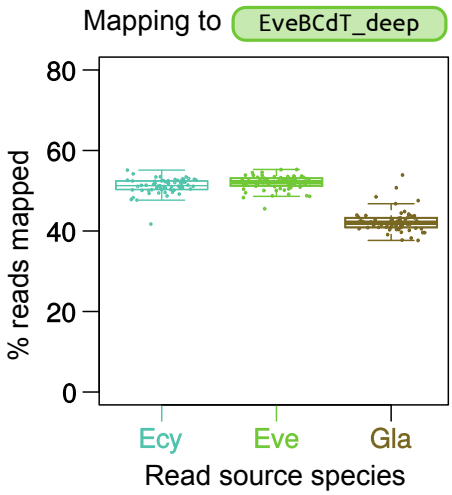

Figure 10: Alignment rate for each sample, grouped by species. Please note that the vertical axis is truncated. All pairwise differences are statistically significant at 0.05 level (pairwise Wilcoxon rank sum test with Holm correction).

8.2 Species-specific reference assemblies

The same results for the other assemblies used as a reference. Mapping rates are quite predictable: in each case the highest alignment rate corresponds to mapping reads to the assembly of the same species (Fig. 11).

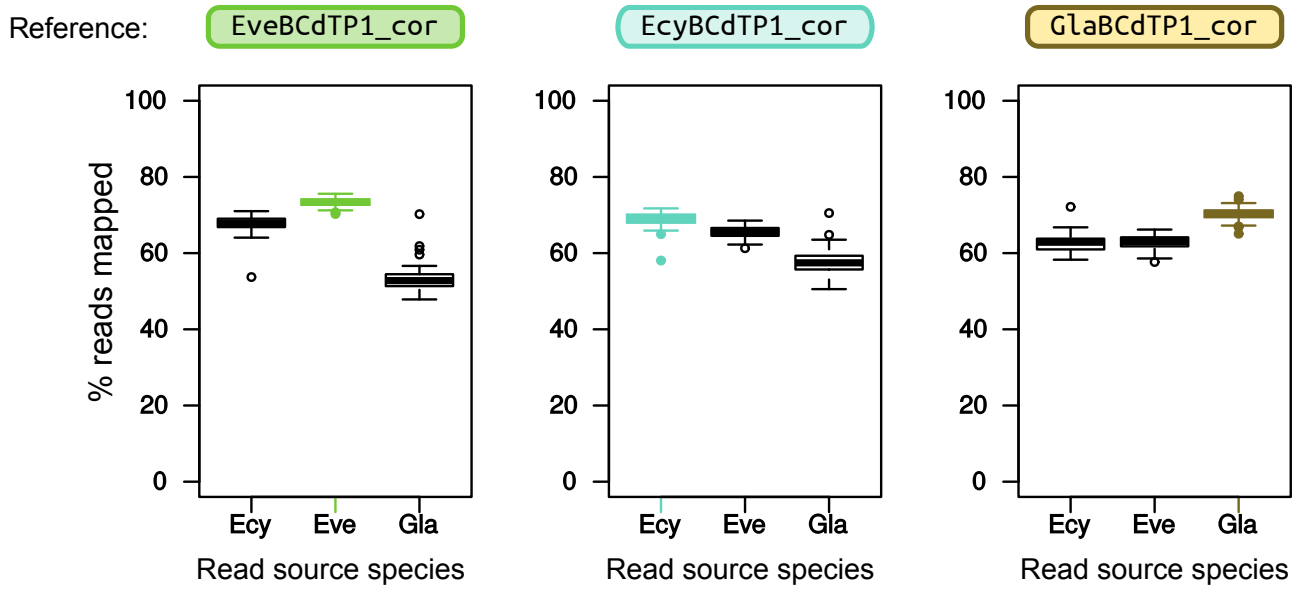

Figure 11: Alignment rate for each sample, grouped by species. All pairwise differences are statistically significant at 0.05 level (pairwise Wilcoxon rank sum test with Holm correction).

9 Differential expression

9.1 Methodology

Now we know expression values for each of hundreds thousands transcripts. However, now we need to know which are differentially expressed under particular conditions. This was done with the DESeq2 R package.

Below is some example code. The functions are too wordy to list them here, so please visit this github page if you are interested. In a nutshell, the functions:

- read data (salmon output);
- perform differential expression analysis with DESeq2, write tables and draw volcano plots;
- add annotation data and then:
- filter out non-metazoan hits or
- filter only known Metazoa hits.

```
#packages
library(DESeq2); library(tximport); library(readr); library(ggplot2)

#Important variables that should be located outside the presumable function
#dir <- "$DE/salmon/eve_deep"; setwd(dir) #change in to ur path
source("DESeq_decontaminate.R") #source functions (github url above)
thisassembly <- "EveBCdT_deep"
#get taxonomic assignation for each contig for further filtering
getTaxonomy(paste0(thiassembly,".taxid"))
qq <- matchTaxonomy(paste0(thiassembly,"_taxreport.txt"), thiassembly)
#get back to the target working directory
setwd(dir)

#Analysis
theseconditions <- c("B12", "T12")
thispecies <- "Eve"; perform.de(dir, thispecies, theseconditions)
thispecies <- "Ecy"; perform.de(dir, thispecies, theseconditions)
thispecies <- "Gla"; perform.de(dir, thispecies, theseconditions)
#other conditions left out for clarity
```

Now we are ready to explore differential expression. These results are summarized in the figures below. They mostly feature volcano plots, a standard way to visualize differential expression. Volcano plots are a type of scatter plots showing fold change (along the horizontal axis) *vs.* significance (along the vertical axis). Naturally, we are interested in the contigs that correspond in the upper left / upper right corners. The former are mRNA that are downregulated upon the respective condition, while the latter are upregulated. However, we are mostly interested in the response of the amphipod itself (not in its symbionts) and in the transcripts with known function. These filtered are labeled as 'Metazoa' at the figures.

But before we uncover these transcripts, we need a few more sanity checks.

9.2 Comparing different reference assemblies

First, below I am showing only those comparisons that were done using the `EveBCdT_deep` alignment as a common reference, as the results are mostly similar regardless of the assembly used as the reference. An example is shown on Fig. 12.

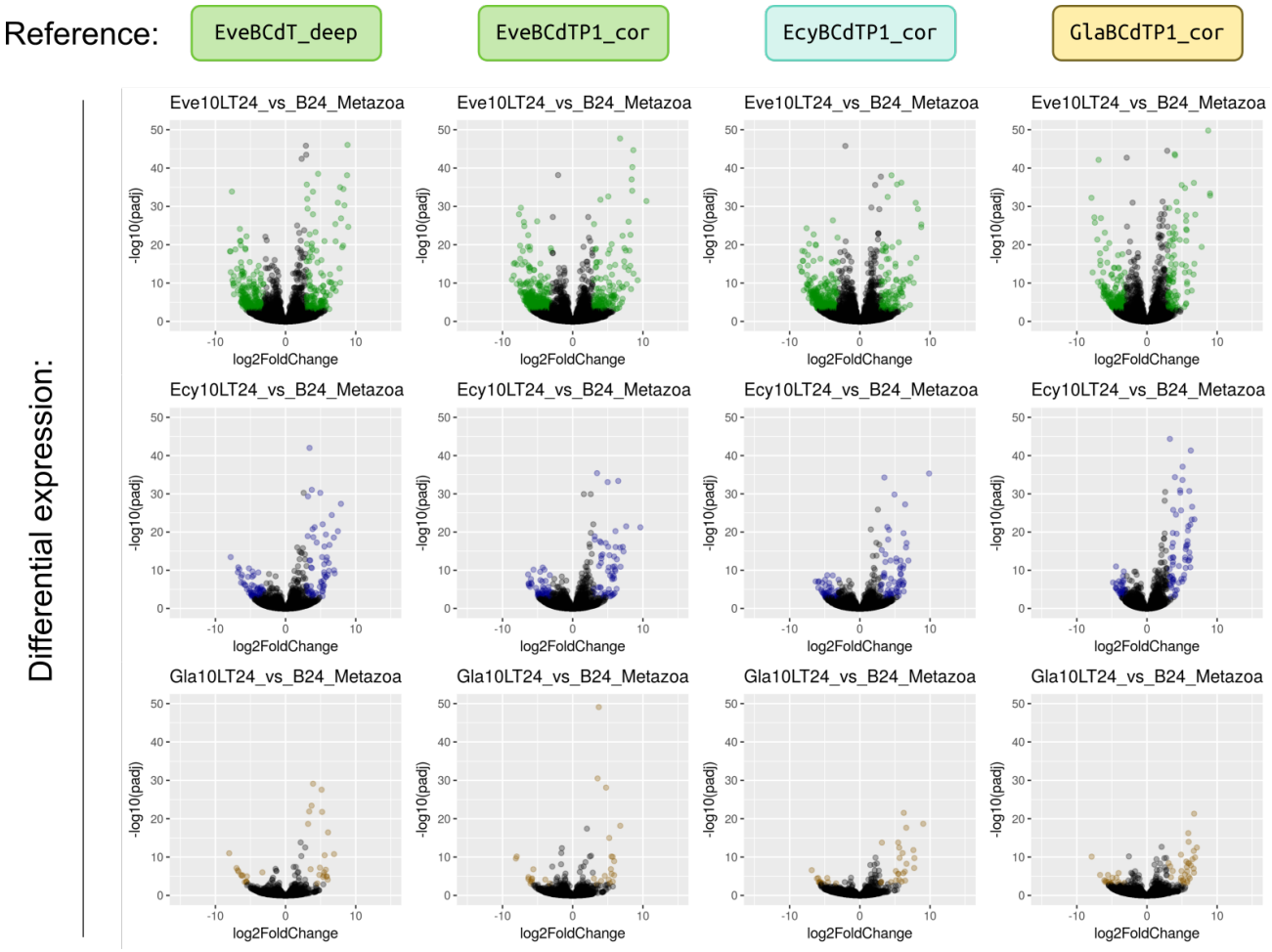

Figure 12: Differential expression analysis results for elevated temperature (10LT24) vs. control.

9.3 Comparing controls

Second, we can compare samples under the control conditions (6 °C) at different times and experiments. Here are some of the possible pairwise comparisons (Fig. 13).

These volcano plots look much flatter then the real comparisons, but still some of them (both within and between experiments) show a number of differentially expressed contigs. I don't think we should adjust the threshold p-value or fold change, as these changes probably reflect some kind of biological variation. However, these differences are definitely something to bear in mind when making conclusions.

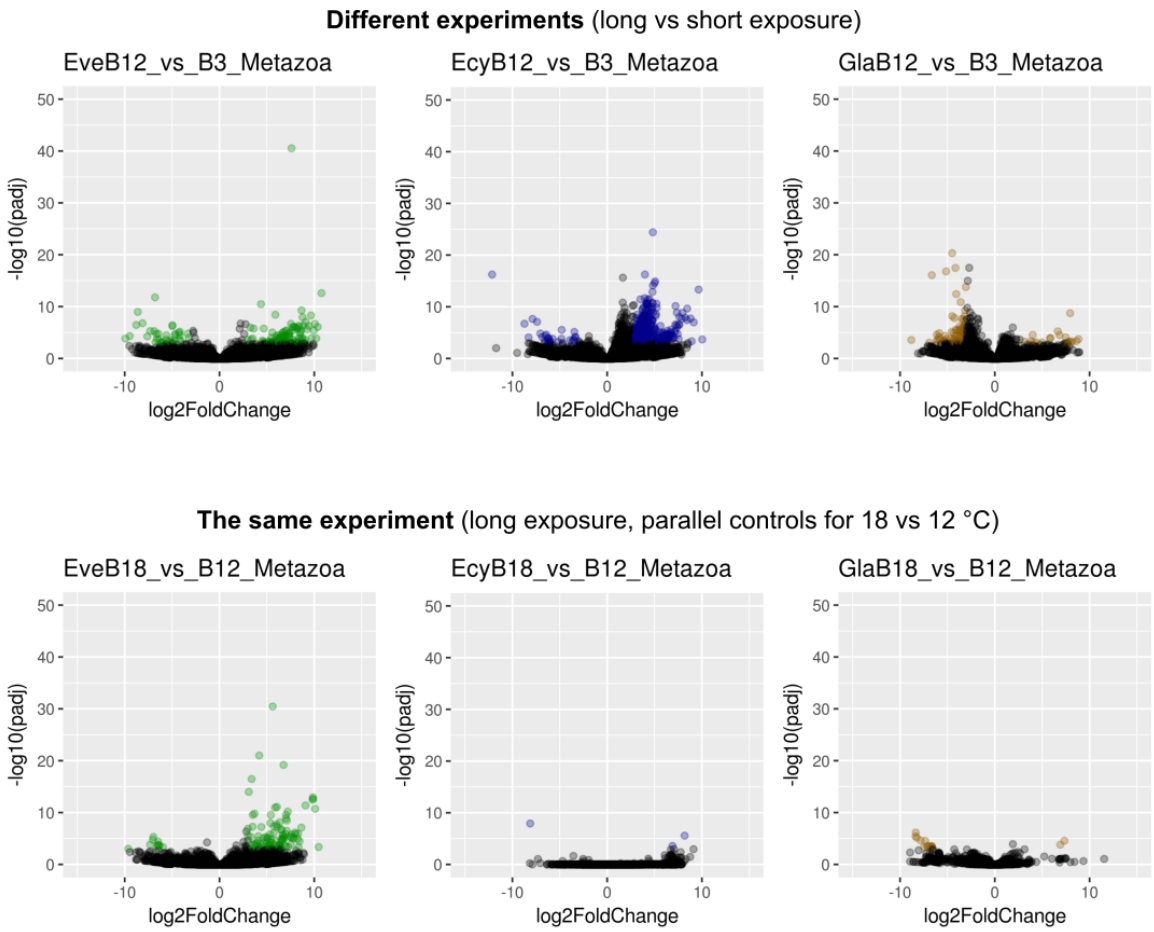

Figure 13: Differential expression in different controls (6 °C).

9.4 The main results

Finally, here are the results for the all comparisons for all species (Figs 14-22), using the `EveBCdT_deep` assembly as a common reference.

All figures are organized in the same way.

- The upper row features DE results for *all* contigs.
- The second row shows the principal component analysis for the samples. This method is actually used to find the most important variable(s) explaining the structure of the data, but here we can use it to illustrate how many samples of each group we have and whether these samples form clear clusters.
- The third row the same volcano plot, but now with the contamination (Ciliata contigs *etc.*) removed.
- The last row shows only the dots corresponding to the known Metazoa transcripts.

Please note that these figures can be explored interactively with, for example, plot.ly web interface (copy `log2FoldChange`, `-log10padj` and `V14` to the table above the graph area and set them as X, Y and hover text, respectively). The same information is also provided as tables.

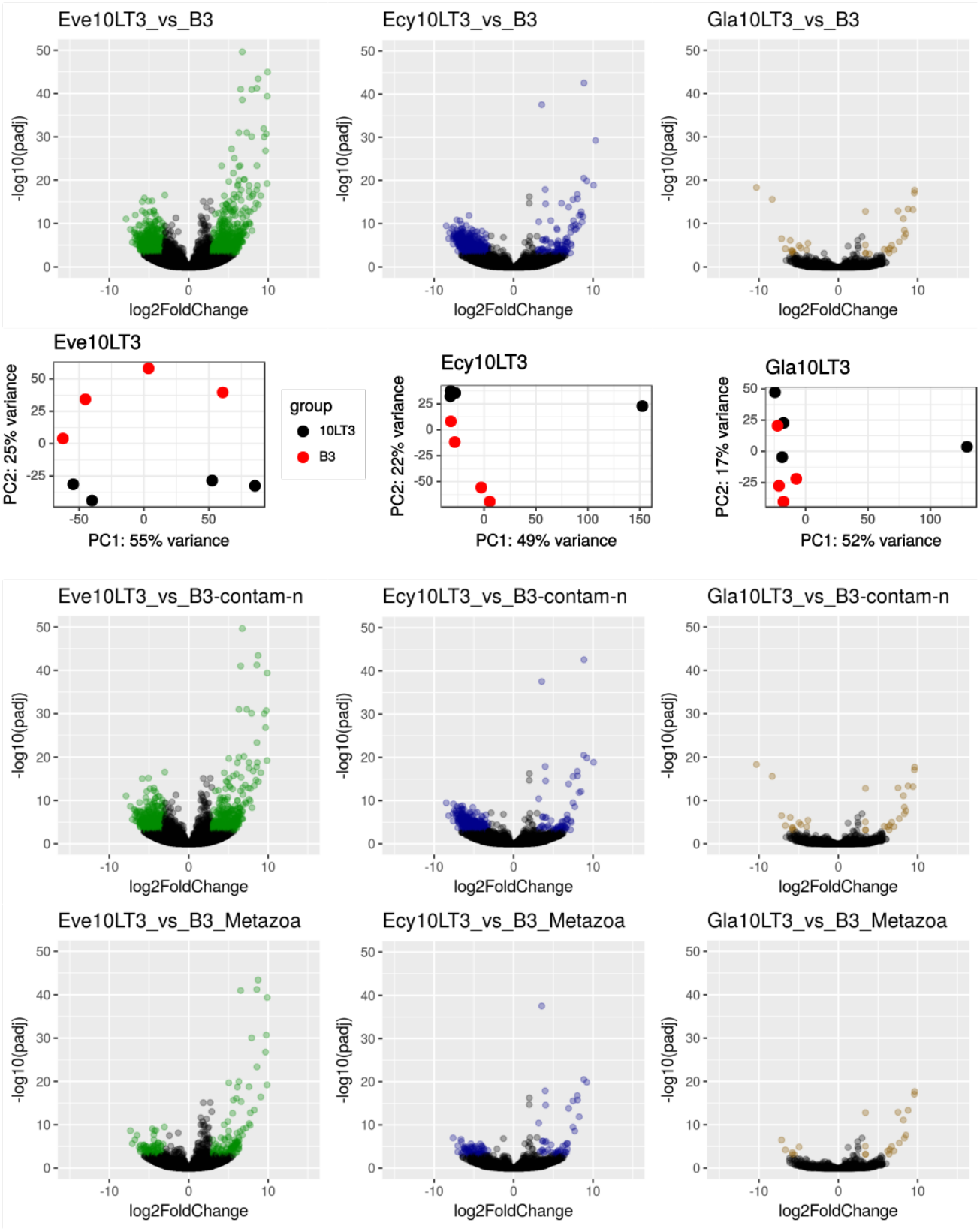

Figure 14: Differential expression analysis results for elevated temperature (10LT3) vs. control. Rows: volcano plot for all genes, PCA plot for all genes. Here should be a large amount of text...

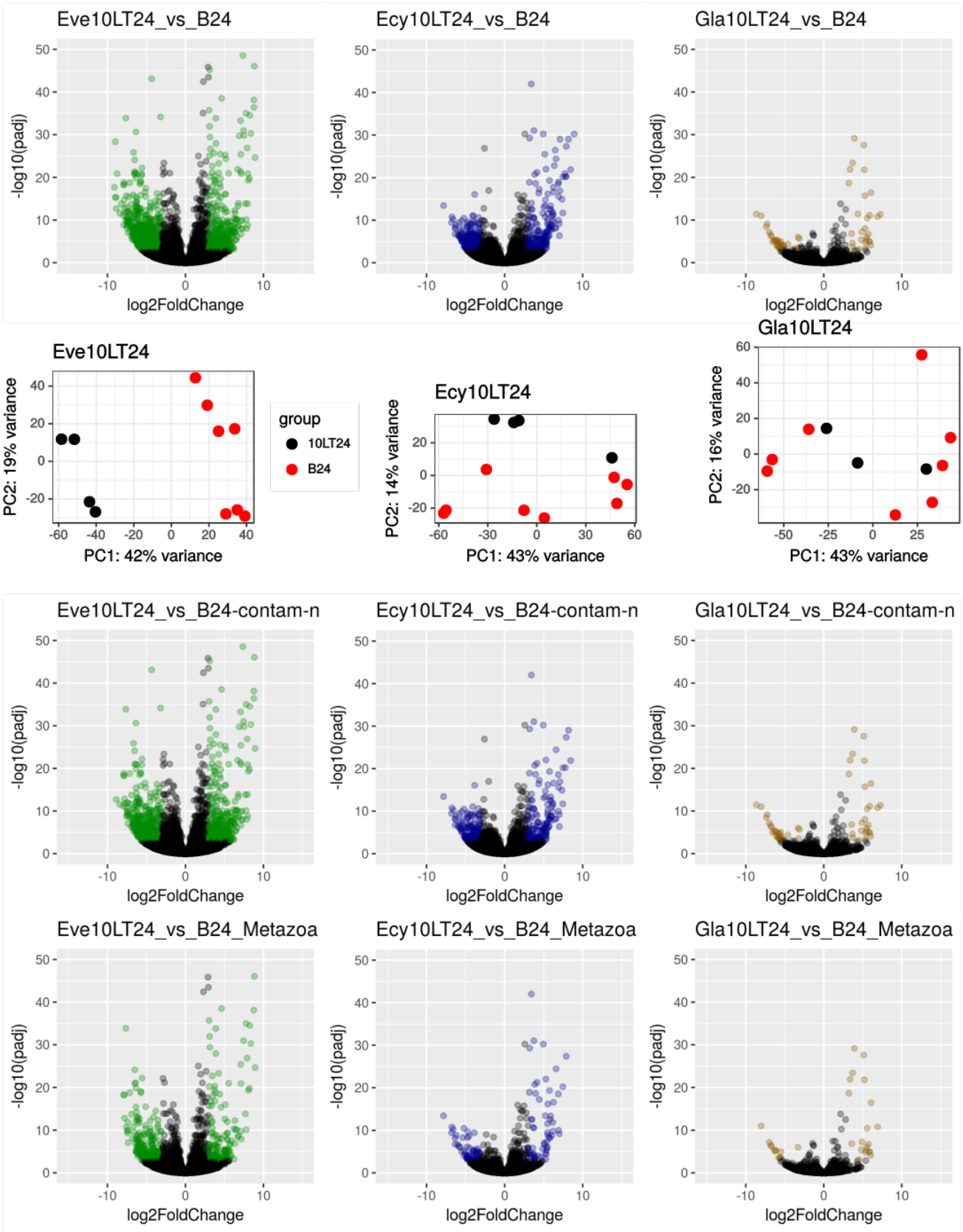

Figure 15: Differential expression analysis results for elevated temperature (10LT3) vs. control. Everything else same as above.

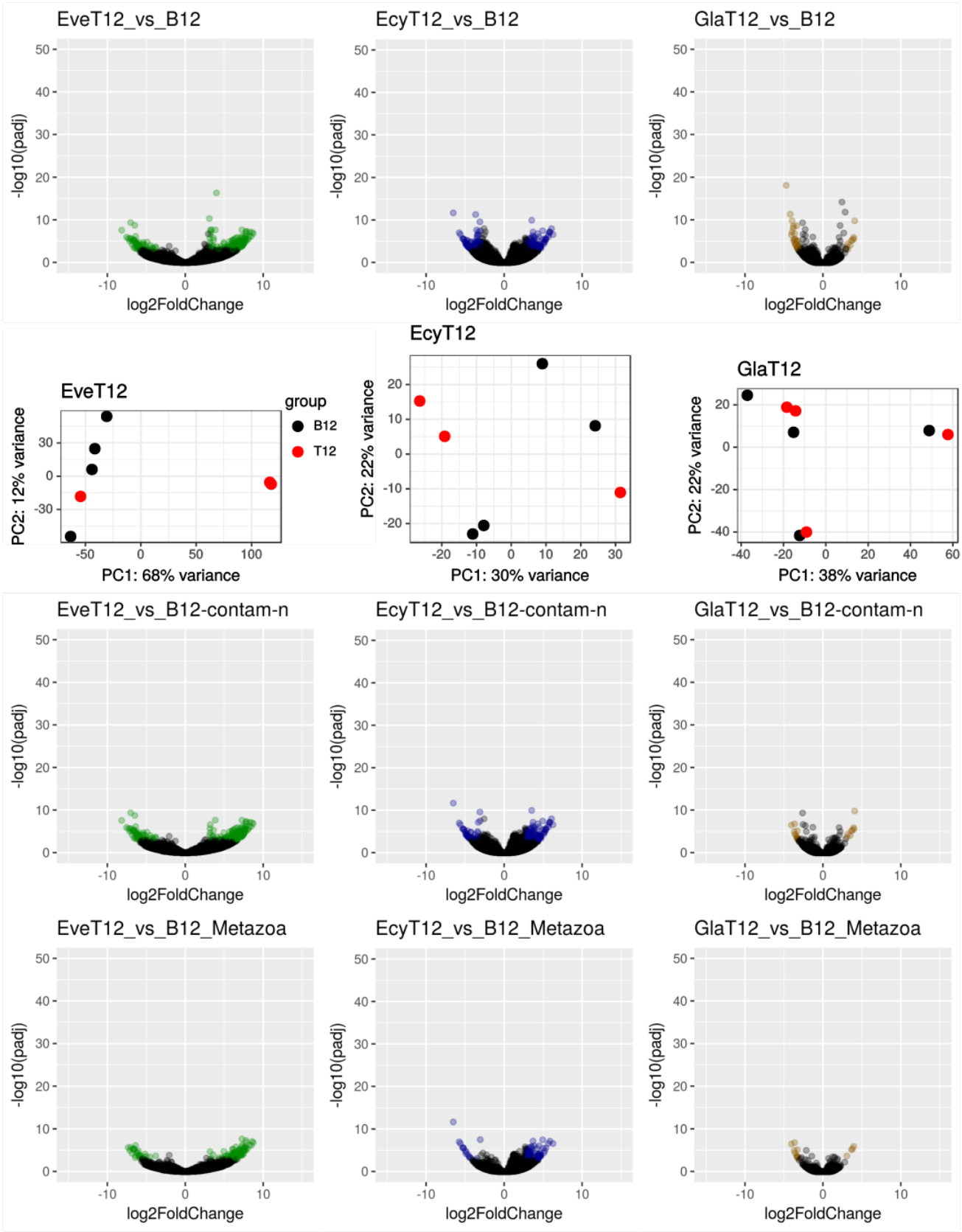

Figure 16: Differential expression analysis results for elevated temperature (12 °C) vs. control. Everything else same as above.

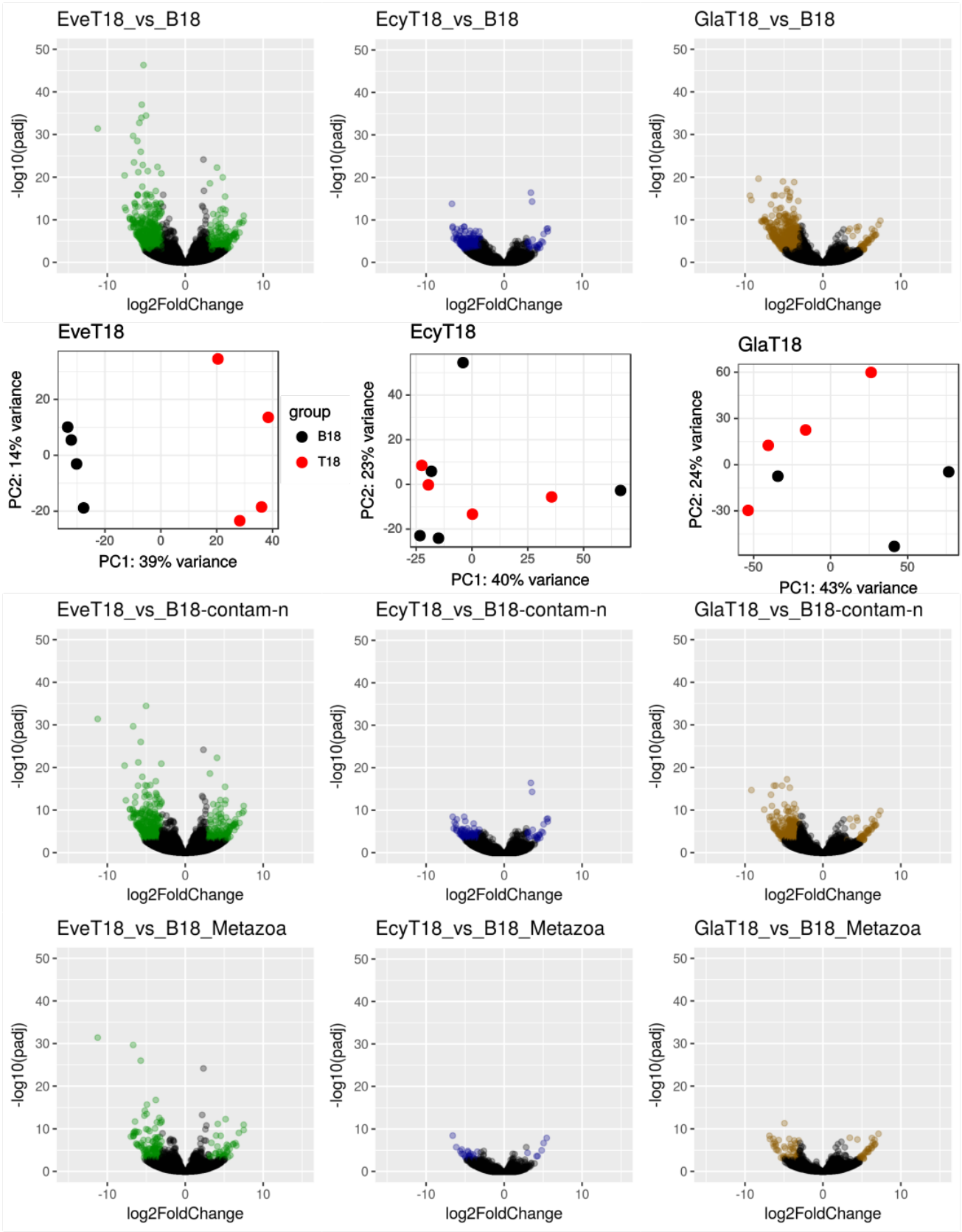

Figure 17: Differential expression analysis results for elevated temperature (18 °C) vs. control. Everything else same as above.

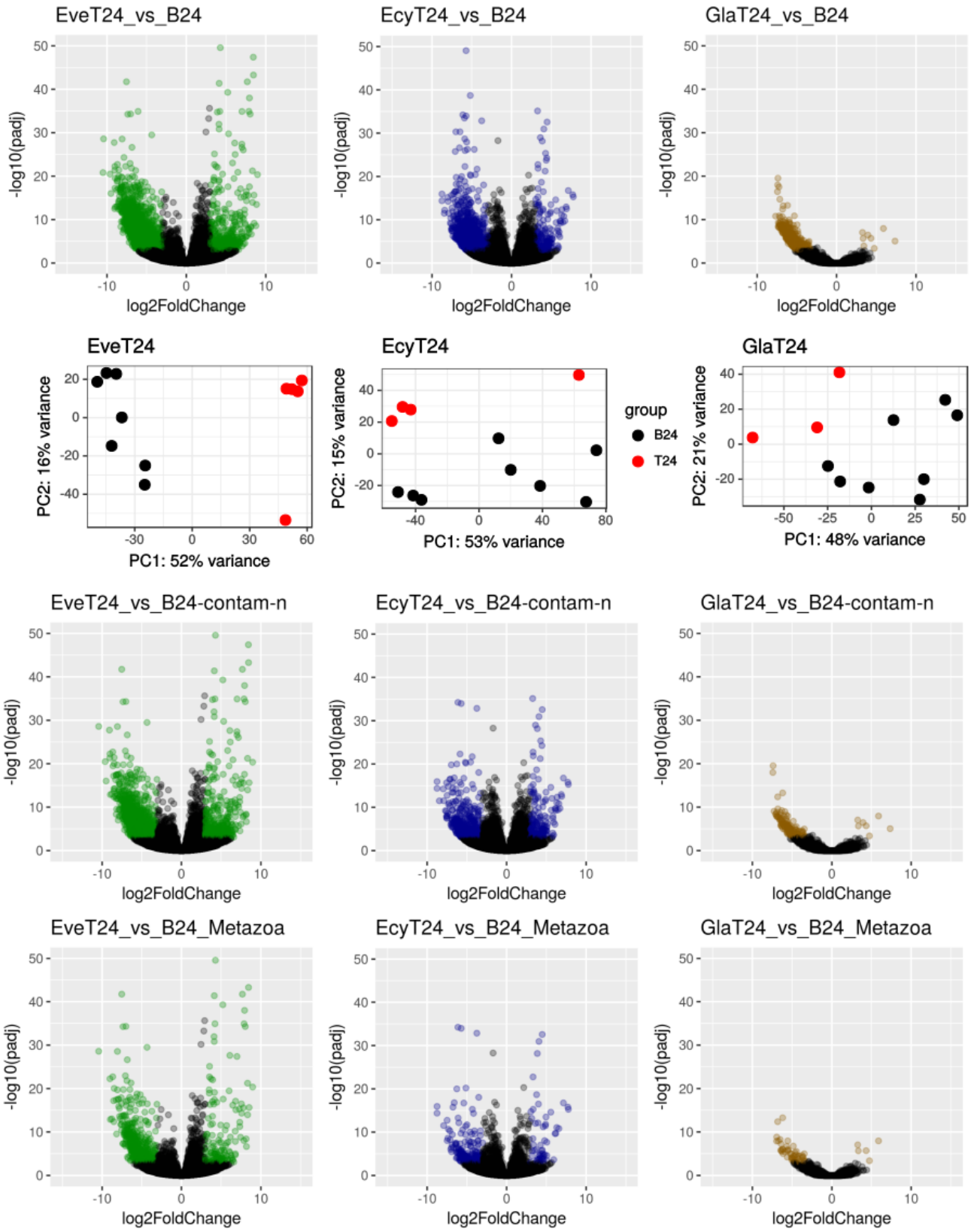

Figure 18: Differential expression analysis results for elevated temperature (24 °C) vs. control. Everything else same as above.

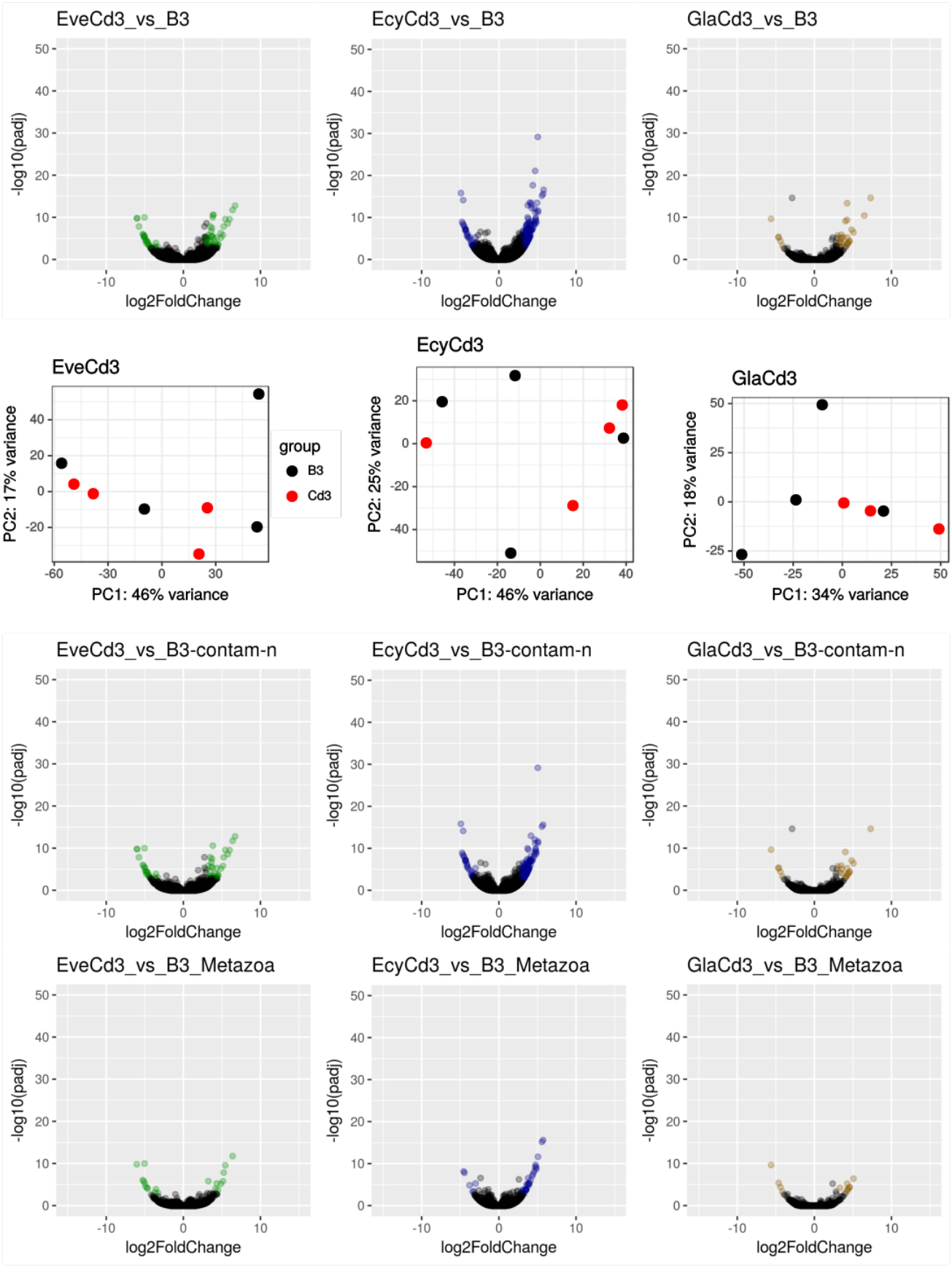

Figure 19: Differential expression analysis results for cadmium exposure (3 hours) vs. control. Everything else same as above.

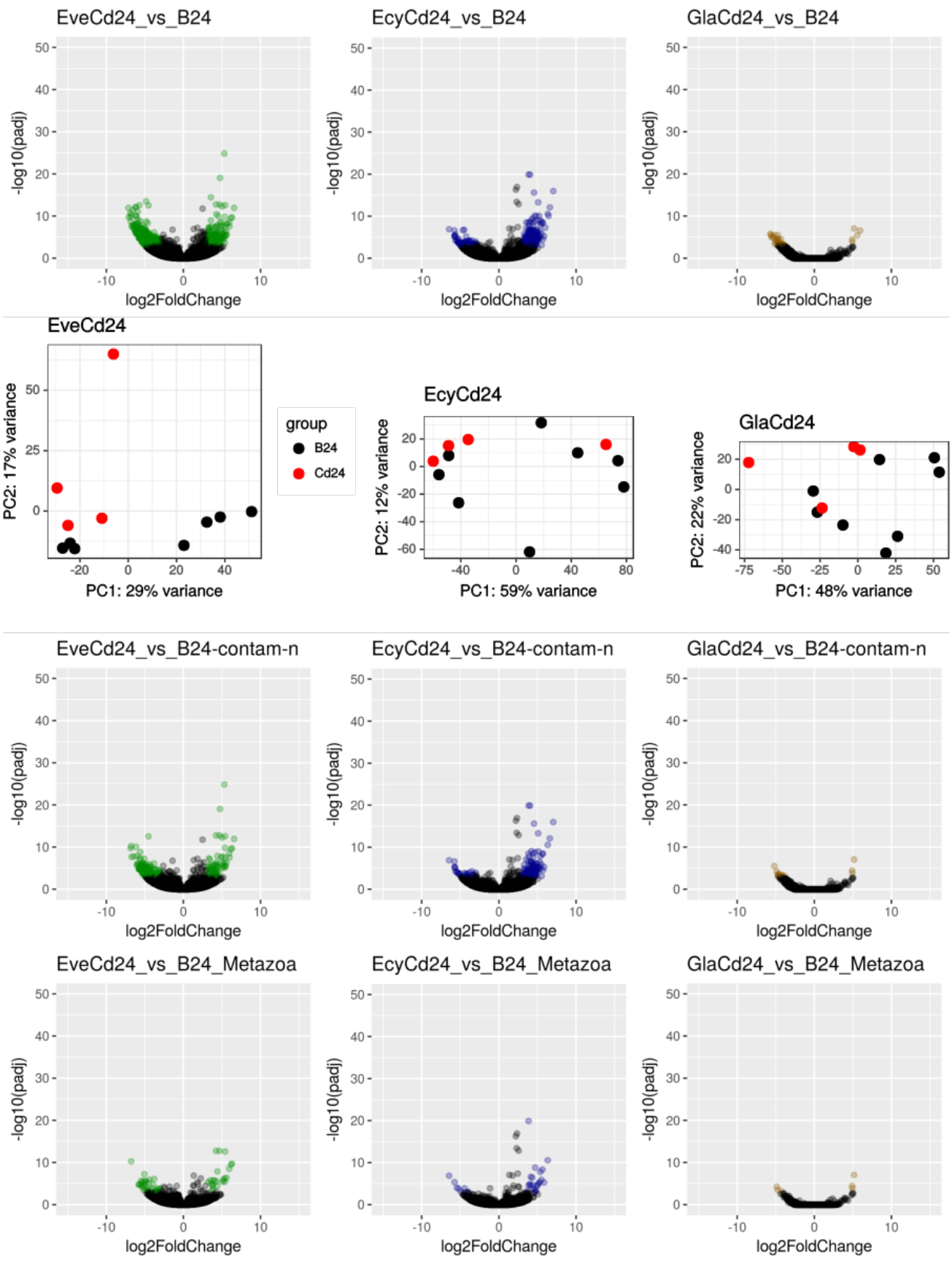

Figure 20: Differential expression analysis results for cadmium exposure (24 hours) vs. control. Everything else same as above.

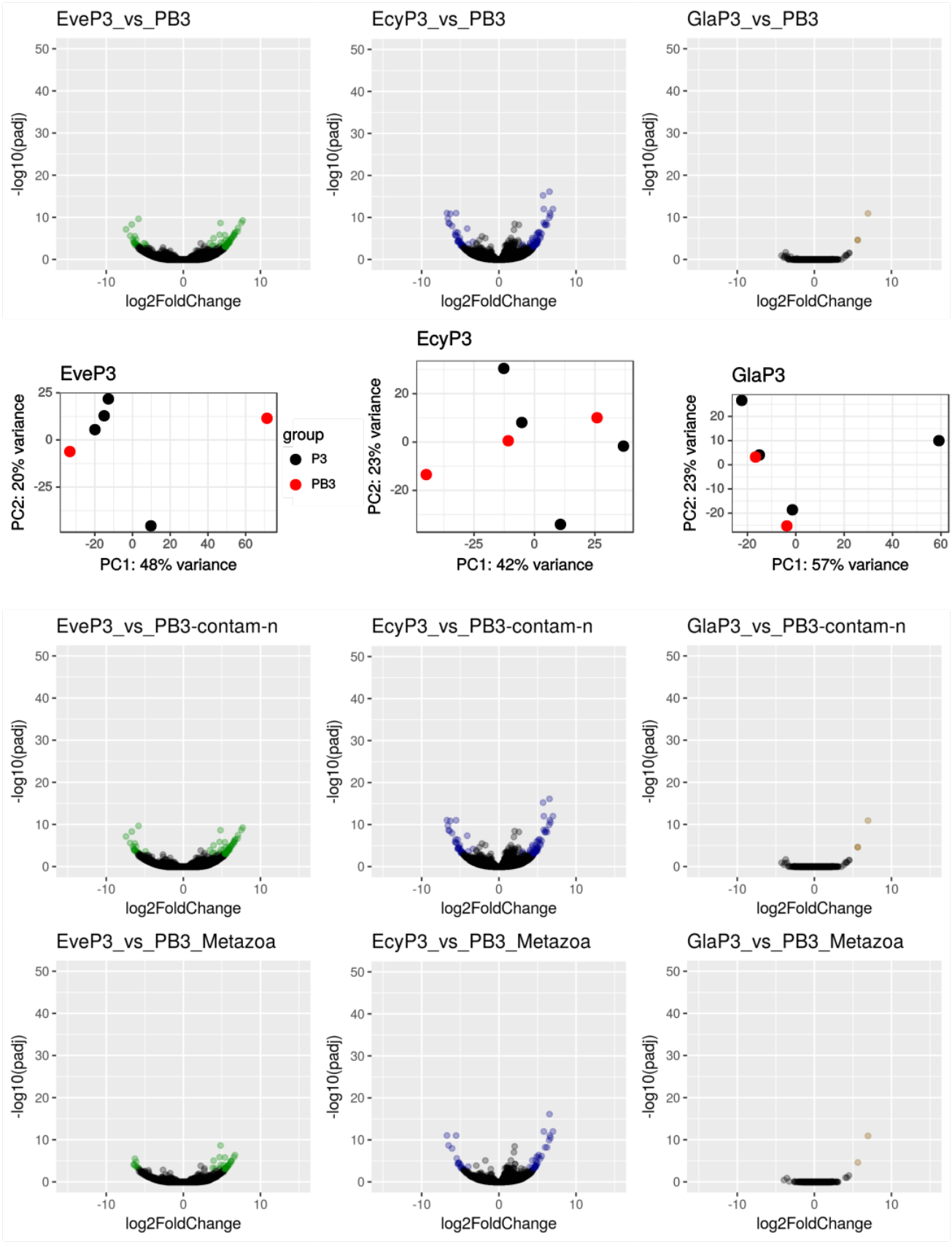

Figure 21: Differential expression analysis results for phenantrene exposure (3 hours) vs. control. Everything else same as above.

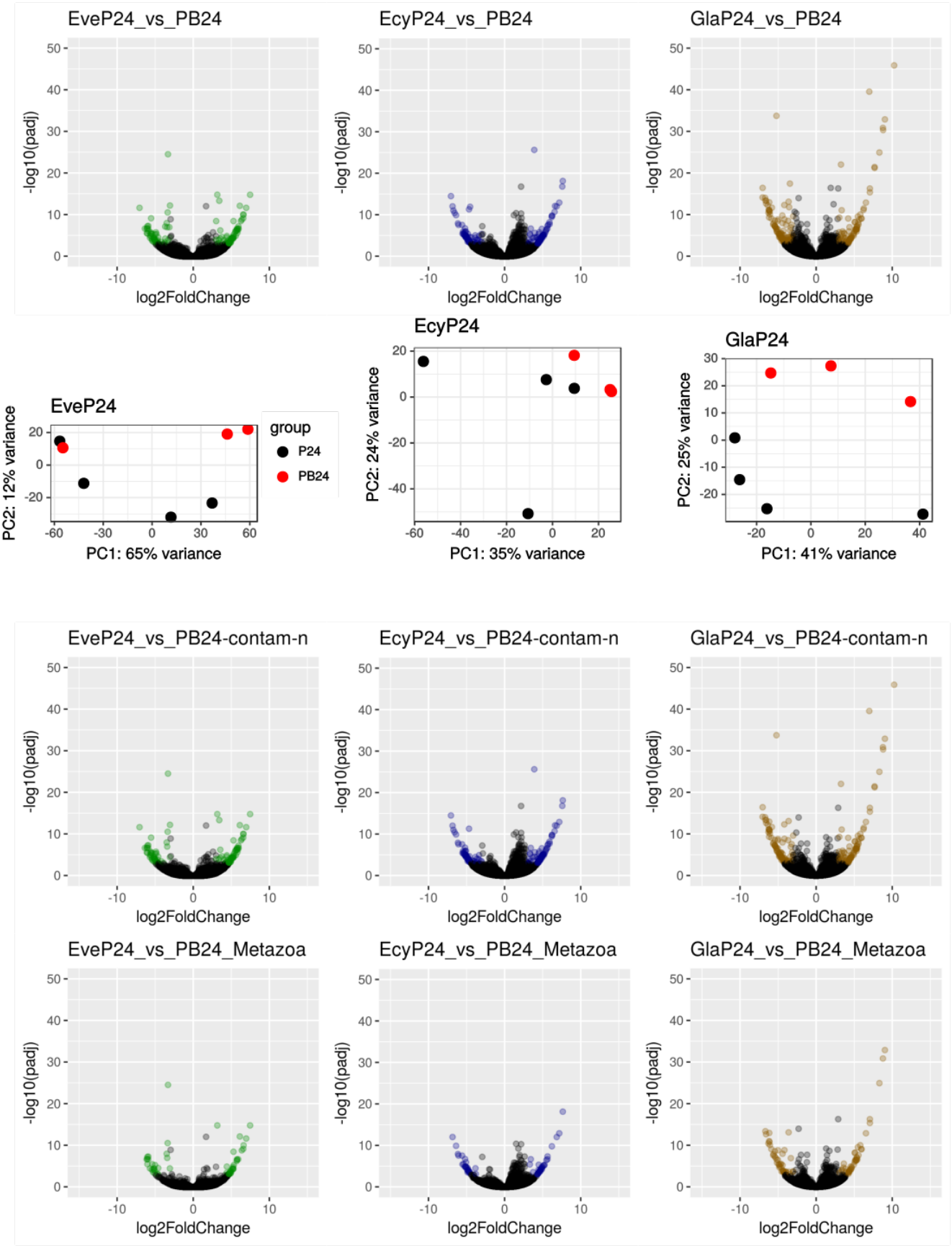

Figure 22: Differential expression analysis results for phenantrene exposure (24 hours) vs. control. Everything else same as above.

10 Possible future directions

1. Time series to find genes with specific dynamic expression patterns.
2. Gene coexpression networks to find gene hubs.
3. Integration with genome and/or proteome data.
4. Make better GO term or other functional summary for differentially expressed genes.
5. Ask more specific biological questions and answer them using these data.