MOLECULAR ECOLOGY

Supplemental Information for:

Early transcriptional responses pathways in *Daphnia magna* are coordinated in networks of lineage specific genes

Luisa Orsini[1], James B. Brown[2,3,4], Omid Shams Solari[4], Dong Li[5], Shan He[5], Ram Podicheti[6,7], Marcus H. Stoiber[3], Katina I. Spanier[8], Donald Gilbert[9], Mieke Jansen[8], Douglas Rusch[10], Michael E. Pfrender[11], John K. Colbourne[1], Mikko J. Frilander[12], Jouni Kvist[12], Ellen Decaestecker[13], Karel A. C. De Schamphelaere[14] and Luc De Meester[8]

Table of Contents:

Table S1	See excel file Orsini_etal_Table S1
Table S2	See excel file Orsini_etal_Table S2
Table S3	See excel file Orsini etal_Table S3
Table S4	See cvs file Orsini etal_Table S4
Table S5	See excel file Orsini_etal_TableS5
Table S6	See excel file Orsini_etal_Table S6
Table S7	See cvs file Orsini_etal_TableS7
Table S8	See excel file Orsini_etal_TableS8
Table S9	See excel file Orsini_etal_TableS9
Table S10	See excel file Orsini_etal_TableS9
Appendix 1	Page 5
Appendix 2	Page 18

Table S1. Significant differentially expressed genes

List of significant differentially expressed genes under specific treatments and separated per genotype. The gene annotation is based on orthology to other species, including *Daphnia pulex* (dpul), *Danio rerio* (drer), *Caenorhabditis elegans* (cele), *Drosophila melanogaster* (dmel), *Mus musculus* (mmus), and *Homo sapiens* (hsap). Scaffold position on the *D. magna* v2.4 reference genome (LRGB00000000.1); Ka/Ks with associated P-value to *Daphnia pulex*; gene expression fold change in log2 and associated adjusted P-value; and gene functions are listed. Abbreviations for environmental perturbations are as in Figure 1. For environmental perturbations not listed in this table differentially expressed genes were not observed.

See excel file Orsini_etal_Table S1

Table S2. Co-expressed gene networks among genotypes

List of networks of co-expressed genes identified in the genotype analysis. The module number corresponds to the value reported on the y-axis in Figure 2. The gene names used are deposited at NCBI Sequence Read Archive SRP059260 (2015): http://www.ncbi.nlm.nih.gov/bioproject/?term=PRJNA284518 and in OrthoDB (Zdobnov *et al.* 2017).

See excel file Orsini_etal_Table S2

Table S3. Enrichment analysis on gene networks identified in the genotype analysis

GO enrichment analysis on the modules identified in the co-expression network analysis conducted at genotype level. For each module multiple enriched categories and their associated enrichment score are shown. Modules for which no GO enrichment was identified are not included in the file. Each module GO analysis is summarized in a separate spreadsheet numbered as the module in Table S2 and Figure 2. The following parameters as identified in DAVID (Dennis *et al.* 2003; Fresno & Fernandez 2013) are shown in each module and enrichment cluster: **Category**: gene ontology classes with MF being molecular function, CC being cellular component and BP being biological process; **GO term**: gene ontology as identified by homology with *Drosophila melanogaster* (source: Flybase); **count**: number of genes identified in the GO analysis; **%**: percentage of genes identified in the GO analysis; P-value: Modified Fisher Exact P-Value, EASE score; **Genes**: genes identified in the enrichment analysis by homology to *Drosophila melanogaster* (source: Flybase); **list total**: number of genes identified in the enrichment analysis; **pop hits**: number of genes in the total group of genes assayed that belong to the specific Gene Category; **pop total**: number of genes in the total group of genes assayed that belong to any Gene Category within the System; **fold enrichment**: fold change for enrichment score; **Bonferroni**: P-value after Bonferroni correction. This value is a conservative adjustment to the EASE score obtained by multiplying the P-value by the number of Gene Categories for which over-representation was obtained in order to control for multiple comparisons; **Benjamini**: P-value after Benjamini correction; **FDR**: false discovery rate.

See excel file Orsini etal_Table S3

Table S4. Annotation of *Daphnia magna* genes

Daphnia magna genes present in the co-expression modules are shown with orthology to known genes and a description of the orthologous gene function. Only a proportion of the *Daphnia* genes has known function.

See cvs file Orsini etal_Table S4

Table S5. Differentially expressed genes shared among environmental perturbations.

List of differentially expressed genes shared either among abiotic or biotic treatments. The treatment, the *Daphnia magna* gene ID, the orthology to other species (as in Supplementary Table 1), scaffold position on the *D. magna* v2.4 reference genome (LRGB00000000.1), gene expression fold change in log2 and associated adjusted P-value, gene function and direction of change (1 or -1) are listed.

See excel file Orsini_etal_TableS5

Table S6. Responsive orthologs in *Daphnia magna* and *Drosophila melanogaster*

Responsive genes in *Daphnia magna* (present paper) and *Drosophila melanogaster* (Brown *et al.* 2014; Stoiber *et al.* 2016) to environmental stressors. Gene ID, treatment, (abbreviated as in Figure 1), direction of response, P-value and fold change are shown for *D. magna* and *D. melanogaster* for shared treatments.

See excel file Orsini_etal_Table S6

Table S7. Co-expression networks under environmental perturbations

List of networks of co-expressed genes within biotic and abiotic perturbations as well as shared across all environmental perturbations (spreadsheet bio_abio). The module number corresponds to the value reported on the x-axis in Figure 4.

See cvs file Orsini_etal_TableS7

Table S8. Enrichment analysis of gene networks obtained from the analysis of modules shared across environmental perturbations.

GO enrichment analysis on the modules shared across environmental perturbations (modules #54 and #154). Module #54 did not show significant enrichment with known gene categories. The parameters shown are as in Table S3.

See excel file Orsini_etal_TableS8

Table S9. **Enrichment analysis of gene networks obtained from the analysis of abiotic perturbations**

GO enrichment analysis on the modules linked abiotic perturbations. The parameters shown are as in Table S3. Three modules did not show significant enrichment with known gene categories (21, 30, 115). These modules are not listed in the file.

See excel file Orsini_etal_TableS9

Table S10. Enrichment analysis of gene networks obtained from the analysis of biotic perturbations

GO enrichment analysis on the modules linked to biotic perturbations. The parameters shown are as in Table S3. Seven modules did not show significant enrichment with known gene categories (25, 45, 90, 91,098,127,146). These modules are not listed in the file.

See excel file Orsini_etal_TableS10

Appendix 1. Compiled scripts in phyton used to combine methods for ortholog detection. At the beginning of each block of commands a text between hash tags (##) describes the command. An horizontal line separates each block of commands.

```
5. one2Many.py, one2ManyContd.py and orthologList.py: The resulting
dictionary from the previous step was parsed into different formats
based on the need.

#!/usr/bin/env python

## This script queries the OMA browser and TreeFam curated database to
identify the closest dPulex ortholog to a given dMagna gene identified
using orthoMCL and Inparanoid. This step is necessary as OMA and
TreeFam do not include dMagna orthologs.##

import numpy as np

inparanoid = open("../ortholog_data/dmagna_dpulex/inparanoid.dmag-
dpul", "rb")
inparanoid.readline()
inparanoid_output =
open("../ortholog_data/dmagna_dpulex/inparanoid.dmag-
dpul.one2many","wb")
for i in inparanoid.readlines():
    tmp = i.split("\t")[2:]
    dmag_prots = [i for i in tmp[0].split(" ") if "dmag" in i]
    dpul_prots = [i for i in tmp[1].split(" ") if "dpul" in i]
    for j in dmag_prots:
        inparanoid_output.write(j + "\t" + " ".join(dpul_prots) +
"\n")
inparanoid_output.close()
inparanoid.close()


tmp = []
for j in s.split("\t"):
    if ("dmag" in j) or ("dpul" in j):
        print j.split(" ")[0]
        tmp.append(j.split(" ")[0])


orthomcl = open("../ortholog_data/dmagna_dpulex/orthomcl.dmag-dpul",
"rb")
orthomcl_output = open("../ortholog_data/dmagna_dpulex/orthomcl.dmag-
dpul.one2many","wb")
for i in orthomcl.readlines():
    tmp_list = i[:-1].split(" ")[1:]
    dmag_prots = [j for j in tmp_list if "dmag" in j]
    dpul_prots = [j for j in tmp_list if "dpul" in j]
    for k in dmag_prots:
        orthomcl_output.write(k + "\t" + " ".join(dpul_prots) + "\n")
orthomcl_output.close()
orthomcl.close()
```

```
#!/usr/bin/env python

##This script queries Daphnia pulex data base when Daphnia magna
orthologs to generate a list of orthologs using orthoMCL and
Inparanoid##

import numpy as np
import os

efx2dappu = open("../ortholog_data/genome_related/pulex_EFX2Dappu",
"rb")
dappu2efx_dict = dict()
for i in efx2dappu.readlines():
    dappu = i[:-1].split("\t")[1]
    efx = i[:-1].split("\t")[0]
    if dappu2efx_dict.has_key(dappu):
        dappu2efx_dict[dappu].append(efx)
    else:
        dappu2efx_dict[dappu] = [efx]
efx2dappu.close()

dmag_to_dpul_orthomcl =
open("../ortholog_data/dmagna_dpulex/orthomcl.dmag-
dpul.one2many","rb")
dmag_to_dpul_inparanoid =
open("../ortholog_data/dmagna_dpulex/inparanoid.dmag-
dpul.one2many","rb")
ortho_dict = dict()
for i in dmag_to_dpul_orthomcl.readlines():
    tmp = i[:-1].split("\t")
    dmag = tmp[0]
    if tmp[1] !="":
        dpul_list = tmp[1].split(" ")
        ortho_dict[dmag] = dpul_list
for i in dmag_to_dpul_inparanoid.readlines():
    tmp = i[:-1].split("\t")
    dmag = tmp[0]
    dpul_list = tmp[1].split(" ")
    if ortho_dict.has_key(dmag):
        ortho_dict[dmag] = list(set(ortho_dict[dmag] + dpul_list))
    else:
        ortho_dict[dmag] = dpul_list
dmag_to_dpul_orthomcl.close()
dmag_to_dpul_inparanoid.close()

dpul_dict = dict()
for k,v in ortho_dict.iteritems():
    if not isinstance(v, list):
        print 1
    for i in v:
        if dpul_dict.has_key(i):
            dpul_dict[i].append(k)
        else:
            dpul_dict[i] = [k]


oma_files = os.listdir("../ortholog_data/oma_pairwise/")

oma_out = open("../ortholog_data/all_species/oma.all.one2one.tab",
"wb")
y = 0
n = 0
for i in oma_files:
    f_tmp = open("../ortholog_data/oma_pairwise/" + i, "rb")
    print i
    for j in f_tmp.readlines():
        if j != "\n":
            tmp = j.split("\t")
            #print tmp
            if "DAPPU" in tmp[0]:
                dappu = tmp[0].replace("DAPPU","DappuP")
                other = i + "|" + tmp[1]
            else:
                dappu = tmp[1].replace("DAPPU","DappuP")
                other = i + "|" + tmp[0]
            if dappu2efx_dict.has_key(dappu):
                y = y +1
            else:
                n = n +1
            oma_out.write(dappu + "\t" + other + "\t\n")
    f_tmp.close()
oma_out.close()
```

```
#!/usr/bin/env python
```

```
##This script unifies genes nomenclature across databases. This step
is necessary because the naming convention differ among databases. For
example unified entrez PubMed with flybase names##

import numpy as np

prots_file =
open("../ortholog_data/genome_related/ref_pep/Caenorhabditis_elegans.W
Bcel235.pep.all.fa"
                   , "rb")
wb2worm = dict()
for i in prots_file.readlines():
    if i[0] == ">":
        cele_id = i[1:i.find(" ")]
        wb_id = i[i.find("gene:") + len("gene:"):i.find("gene:") +
i[i.find("gene:"):].find(" ")]
        if wb2worm.has_key(wb_id):
            wb2worm[wb_id].append(cele_id)
        else:
            wb2worm[wb_id] = [cele_id]
prots_file.close()


oma_worm = open("../ortholog_data/convert_ids/worm_conv.csv","rb")
oma_worm_converted = open("../ortholog_data/oma_pairwise/cele","wb")
for i in oma_worm.readlines():
    tmp = i.split(",")
    if wb2worm.has_key(tmp[3]):
        for i in wb2worm[tmp[3]]:
            oma_worm_converted.write(tmp[0] + "\t" + i + "\n")
oma_worm.close()
oma_worm_converted.close()

fb2entrez_file = open("../ortholog_data/convert_ids/fly_conv.csv",
"rb")
entrez2fb = dict()
for i in fb2entrez_file.readlines():
    tmp = i[:-1].split(",")
    fb = tmp[2]
    entrez = tmp[3]
    if not(fb == "NA" or entrez == "NA"):
        entrez2fb[entrez] = fb
fb2entrez_file.close()

oma_fly = open("../ortholog_data/convert_ids/omid_entrez.fly.txt",
"rb")
oma_fly_converted = open("../ortholog_data/oma_pairwise/dmel", "wb")
for i in oma_fly.readlines():
    tmp = i[:-1].split(" ")
    if i != "\n":
        tmp = i[:-1].split(" ")
        if entrez2fb.has_key(tmp[1]):
            oma_fly_converted.write(tmp[0] + "\t" + entrez2fb[tmp[1]]
+ "\n")
oma_fly.close
oma_fly_converted.close()

prots_file =
open("../ortholog_data/genome_related/ref_pep/Danio_rerio.Zv9.pep.all.
fa"
                  , "rb")
g2p = dict()
for i in prots_file.readlines():
    if i[0] == ">":
        p = i[1:i.find(" ")]
        g = i[i.find("gene:") + len("gene:"):i.find("gene:") +
i[i.find("gene:"):].find(" ")]
        if g2p.has_key(g):
            g2p[g].append(p)
        else:
            g2p[g] = [p]
prots_file.close()

oma_danio = open("../ortholog_data/oma_pairwise/drer.old","rb")
oma_danio_converted = open("../ortholog_data/oma_pairwise/drer","wb")
for i in oma_danio.readlines():
    tmp = i[:-1].split("\t")
    if len(tmp) > 1:
        dappu = tmp[0]
        drer = tmp[1]
        if g2p.has_key(drer):
            for j in g2p[drer]:
                oma_danio_converted.write(dappu + "\t" + j + "\n")
        else:
            print drer
oma_danio.close()
oma_danio_converted.close()

prots_file =
open("../ortholog_data/genome_related/ref_pep/Mus_musculus.GRCm38.pep.
all.fa"
                  , "rb")
g2p = dict()
for i in prots_file.readlines():
    if i[0] == ">":
        p = i[1:i.find(" ")]
        g = i[i.find("gene:") + len("gene:"):i.find("gene:") + 
i[i.find("gene:"):].find(" ")]
        if g2p.has_key(g):
            g2p[g].append(p)
        else:
            g2p[g] = [p]
prots_file.close()

oma_mmus = open("../ortholog_data/oma_pairwise/mmus.old","rb")
oma_mmus_converted = open("../ortholog_data/oma_pairwise/mmus","wb")
for i in oma_mmus.readlines():
    tmp = i[:-1].split("\t")
    if len(tmp) > 1:
        dappu = tmp[0]
        mmus = tmp[1]
        if g2p.has_key(mmus):
            for j in g2p[mmus]:
                oma_mmus_converted.write(dappu + "\t" + j + "\n")
        else:
            print mmus
oma_mmus.close()
oma_mmus_converted.close()

prots_file = 
open("../ortholog_data/genome_related/ref_pep/Homo_sapiens.GRCh38.pep.
all.fa"
                  , "rb")
g2p = dict()
for i in prots_file.readlines():
    if i[0] == ">":
        p = i[1:i.find(" ")]
        g = i[i.find("gene:") + len("gene:"):i.find("gene:") + 
i[i.find("gene:"):].find(" ")]
        if g2p.has_key(g):
            g2p[g].append(p)
        else:
            g2p[g] = [p]
prots_file.close()

oma_hsap = open("../ortholog_data/oma_pairwise/hsap.old","rb")
oma_hsap_converted = open("../ortholog_data/oma_pairwise/hsap","wb")
for i in oma_hsap.readlines():
    tmp = i[:-1].split("\t")
    if len(tmp) > 1:
        dappu = tmp[0]
        hsap = tmp[1]
        if g2p.has_key(hsap):
            for j in g2p[hsap]:
                oma_hsap_converted.write(dappu + "\t" + j + "\n")
        else:
            print hsap
oma_hsap.close()
oma_hsap_converted.close()
```

```
#!/usr/bin/env python

##This script parses and compiles orthologs into a unified dictionary
by unifying lists of orthologs obtained from the search engines##

import numpy as np
import shelve

def dodo(dodo_one2many_address):
    #dodo_one2many_address =
"../ortholog_data/all_species/DODO.all.one2may"
    dodo_file = open(dodo_one2many_address, "rb")
    dodo_dict = dict()
    for i in dodo_file.readlines():
        tmp = i.split("\t")
        dmag = tmp[0]
        for j in tmp[1:-1]:
            if j != "":
                for k in j.split(" "):
                    dodo_dict[dmag + "--" + k] = "DODO"
    dodo_file.close()
    return dodo_dict

def inparanoid(inparanoid_one2many_address):
    inparanoid_file = open(inparanoid_one2many_address,"rb")
    inparanoid_dict = dict()
    for i in inparanoid_file.readlines():
        tmp = i.split("\t")
        dmag = tmp[0]
        for j in tmp[1:-1]:
            if j != "":
                for k in j.split(" "):
                    inparanoid_dict[dmag + "--" + k] = "inparanoid"
    return inparanoid_dict


def orthomcl(orthomcl_one2many_address):
    orthomcl_file = open(orthomcl_one2many_address,"rb")
    orthomcl_dict = dict()
    for i in orthomcl_file.readlines():
        tmp = i.split("\t")
        dmag = tmp[0]
        for j in tmp[1:-1]:
            if j != "":
                for k in j.split(" "):
                    orthomcl_dict[dmag + "--" + k] = "orthomcl"
    return orthomcl_dict

def proteinortho(proteinortho_one2many_address):
    proteinortho_file = open(proteinortho_one2many_address,"rb")
    proteinortho_dict = dict()
    for i in proteinortho_file.readlines():
        tmp = i.split("\t")
        dmag = tmp[0]
        for j in tmp[1:-1]:
            if j != "":
                for k in j.split(" "):
                    proteinortho_dict[dmag + "--" + k] = 
"proteinortho"
    return proteinortho_dict


def treefam(treefam_one2many_address):
    treefam_file = open(treefam_one2many_address,"rb")
    treefam_dict = dict()
    for i in treefam_file.readlines():
        tmp = i.split("\t")
        dmag = tmp[0]
        for j in tmp[1:-1]:
            if j != "":
                for k in j.split(" "):
                    treefam_dict[dmag + "--" + k] = "treefam"
    return treefam_dict

def merge_dictionaries(list_of_dictionaries):
    Dict = dict()
    for i in list_of_dictionaries:
        for k,v in i.iteritems():
            if Dict.has_key(k):
                Dict[k].append(v)
            else:
                Dict[k] = [v]
    return Dict
```

```
#!/usr/bin/env python
```

```
##This scripts parses the dictionary obtained from the previous step
into different formats. The specific function of each block of
commands is reported at the beginning of each script##

##Generate an ortholog list in the object orthologs.shelve##

import numpy as np
import shelve

ortho =
shelve.open("/srv/scratch/omid/workspace/data/orthology/orthologs.shel
ve")

pairs = ortho.keys()
orthologs = dict()
for i in pairs:
    tmp = i.split("--")
    dmag = tmp[0]
    other = tmp[1]
    if orthologs.has_key(dmag):
        orthologs[dmag].append(other)
    else:
        orthologs[dmag] = [other]


ortho_list = open("../ortholog_data/ortholog_list.tab","rb")
ortho = dict()
for i in ortho_list.readlines():
    tmp = i.split("\t")[0].split("--")
    if ortho.has_key(tmp[0][5:]):
        ortho[tmp[0][5:]].append(tmp[1])
    else:
        ortho[tmp[0][5:]] = [tmp[1]]
ortho_list.close()


aa =
open("../ortholog_data/genome_related/ref_pep/dmagset7mceqm.finloc9b.p
uban.aa", "rb")
aa_list = list()
for i in aa.readlines():
    if i[0] == ">":
        aa_list.append(i[1:i.find(" ")])
aa.close()


c = 0
for i in aa_list:
    tmp_out = 
open("/srv/scratch/omid/workspace/data/orthology/ortholog_deseq_files/
" + i, "wb")
    if ortho.has_key(i):
        tmp_out.write(",".join(ortho[i]))
    else:
        tmp_out.write(i)
    c = c + 1
    tmp_out.close()


ortholog_file = 
open("../ortholog_data/one2many_complete_orthologs.csv","wb")
for k in aa_list:
    if ortho.has_key(k):
        ortholog_file.write(k + "\t" + " ".join(ortho[k]) + "\n")
    else:
        ortholog_file.write(k + "\t\n")
ortholog_file.close()


#!/usr/bin/env python

##the script parses the list of gene ortholog obtained from 
proteinOrtho and DODO into a one2many hash table where the values are 
sorted according to the species in a specific order:

dmag|geneNo  dPul_genes cEle_genes dRer_genes ##


import numpy as np

protein_ortho = 
open("/users/omid/ortholog_data/all_species/proteinortho.all", "rb")
protein_ortho_output = 
open("/users/omid/ortholog_data/all_species/proteinortho.all.one2many"
, "wb")
dmagL = []
protein_ortho.readline()
for i in protein_ortho.readlines():
    prots = i[:-1].split("\t")[3:]
    dmag_prots = [prots[0].split(",") if prots[0] != "*" else ""]
    if prots[0] != "*":
        dmag_prots = 
    print dmag_prots
    dpul_prots = [" ".join(prots[1].split(",")) if prots[1] != "*" 
else ""]
```

```
    cele_prots = [" ".join(prots[2].split(",")) if prots[2] != "*"
else ""]
    drer_prots = [" ".join(prots[3].split(",")) if prots[3] != "*"
else ""]
    mmus_prots = [" ".join(prots[4].split(",")) if prots[4] != "*"
else ""]
    dmel_prots = [" ".join(prots[5].split(",")) if prots[5] != "*"
else ""]
    hsap_prots = [" ".join(prots[6].split(",")) if prots[6] != "*"
else ""]
    for k in dmag_prots:
        dmagL.append(k)
        print k
        protein_ortho_output.write(k + "\t" + "\t".join(dmel_prots +
dpul_prots + drer_prots + cele_prots + hsap_prots + mmus_prots) +
"\n")
protein_ortho.close()
protein_ortho_output.close()

tmp = s
prots = tmp[:-1].split("\t")[3:]
mmus_prots = [" ".join(prots[0].split(",")) if prots[0] != "*" else
""]
hsap_prots = [" ".join(prots[1].split(",")) if prots[1] != "*" else "
"]
cele_prots = [" ".join(prots[2].split(",")) if prots[2] != "*" else
""]
drer_prots = [" ".join(prots[3].split(",")) if prots[3] != "*" else "
"]
dmag_prots = [" ".join(prots[4].split(",")) if prots[4] != "*" else "
"]
dpul_prots = [" ".join(prots[5].split(",")) if prots[5] != "*" else "
"]
dmel_prots = [" ".join(prots[6].split(",")) if prots[6] != "*" else "
"]

protein_ortho =
open("/users/omid/ortholog_data/all_species/proteinortho.all", "rb")
s = protein_ortho.readline()
s = protein_ortho.readline()

protein_ortho.close()

seen = set()
uniq = []
for x in dmagL[:20]:
    if x not in seen:
        uniq.append(x)
        seen.add(x)

dodo = open("/users/omid/ortholog_data/all_species/DODO.all", "rb")
dodo_output =
open("/users/omid/ortholog_data/all_species/DODO.all.one2may", "wb")
dodo.readline()
old_id = '1'
group = []
for i in dodo.readlines():
    tmp_list = i[:-1].split("\t")
    new_id = tmp_list[0][tmp_list[0].find("_")+1:]
    print new_id
    if new_id == old_id :
        #print "yes"
        group.append(tmp_list[2])
    else:
        print "no"
        dmag_prots = [j for j in group if "dmag" in j]
        cele_prots = " ".join([j for j in group if "cele" in j])
        dmel_prots = " ".join([j for j in group if "dmel" in j])
        dpul_prots = " ".join([j for j in group if "dpul" in j])
        drer_prots = " ".join([j for j in group if "drer" in j])
        hsap_prots = " ".join([j for j in group if "hsap" in j])
        mmus_prots = " ".join([j for j in group if "mmus" in j])
        for k in dmag_prots:
            dodo_output.write(k + "\t" + dmel_prots + "\t" +
dpul_prots + "\t" + drer_prots + "\t" + cele_prots + "\t" + hsap_prots
+ "\t" + mmus_prots + "\t\n")
        old_id = new_id
        group = [tmp_list[2]]
dodo.close()
dodo_output.close()

#!/usr/bin/env python

## This script orders the ortholog list by species name ##

import numpy as np

File = open("../ortholog_data/ortholog_list.tab", "rb")
ortholog_dict = dict()
for i in File.readlines():
    tmp = i.split("\t")[0].split("--")
    if ortholog_dict.has_key(tmp[0]):
        ortholog_dict[tmp[0]].append(tmp[1])
    else:
        ortholog_dict[tmp[0]] = [tmp[1]]
```

```
dmagna_prot = 
open("../ortholog_data/genome_related/ref_pep/dmagset7mceqm.finloc9b.p
uban.aa","rb")
for i in dmagna_prot.readlines():
    if i[0] == ">":
        dmagna = i[1:i.find(" ")]
        #print dmagna
        if not ortholog_dict.has_key("dmag|" + dmagna):
            ortholog_dict["dmag|" + dmagna] = [dmagna]



output = 
open("../ortholog_data/deseq_orthologs/ortholog_list.tab","wb")
for k,v in ortholog_dict.iteritems():
    dmagna = k[5:]
    for i in v:
        if "dmel|" in i:
            val = i
        elif "hsap|" in i:
            val = i
        elif "mmus|" in i:
            val = i
        elif "cele|" in i:
            val = i
        elif "drer|" in i:
            val = i
        elif "dpul|" in i:
            val = i
        else:
            val = i
    output.write(dmagna + "\t" + val + "\t\n")
output.close()
```

Appendix 2. NCBI accession numbers to RNASeq data across the 14 environmental conditions and genotypes.

Treatment	Genotype	Accession number (NCBI)
UV	XI	SRX1057978; SRX1057977; SRX1057975:
Pb	XI	SRX1057974; SRX1057971; SRX1057970
pH	XI	SRX1057944; SRX1057906; SRX1057905
NaCl	XI	SRX1057851; SRX1057821; SRX1057818
Control	XI	SRX1057817; SRX1057816; SRX1057815
Cd	XI	SRX1057814; SRX1057813; SRX1057812
Invertebrate predation (triops)	X, I	SRX1057811; SRX1057810; SRX1057808; SRX1057783; SRX1057782; SRX1057781
Parasite Pasteuria ramosa	X, I	SRX1057780; SRX1057772; SRX1057700; SRX1057698; SRX1057697; SRX1057696
Vertebrate predator (fish)	X, I	SRX1057695; SRX1057693; SRX1057650; SRX1057648; SRX1057647; SRX1057646;
crowding	X, I	SRX1057645; SRX1057644; SRX1057361; SRX1057358; SRX1057356; SRX1057353
control	X, I	SRX1057347; SRX1057344; SRX1057343; SRX1057342; SRX1056744; SRX1056743; SRX1056742
carbaryl	X, I	SRX1056741; SRX1056740; SRX1056739; SRX1056738; SRX1056725; SRX1056711
Cyanobacteria toxic	X, I	SRX1056709; SRX1056706; SRX1056705; SRX1056667; SRX1056646; SRX1056639
Cyanobacteria non-toxic	X, I	SRX1056638; SRX1056637; SRX1056636; SRX1053860; SRX1053859; SRX1053858