RNA-seq Report

Nathan Hughes (JIC)

May 16, 2019

Contents

1 Data setup

1.1 Helper funcs for pprinting

```
import tabulate
import IPython

class OrgFormatter(IPython.core.formatters.BaseFormatter):
    format_type = IPython.core.formatters.Unicode('text/org')
    print_method = IPython.core.formatters.ObjectName('_repr_org_')

def pd_dataframe_to_org(df):
    return tabulate.tabulate(df, headers='keys', tablefmt='orgtbl', showindex='always')

ip = get_ipython()
ip.display_formatter.formatters['text/org'] = OrgFormatter()

f = ip.display_formatter.formatters['text/org']
f.for_type_by_name('pandas.core.frame', 'DataFrame', pd_dataframe_to_org)

print('Lets go!')
```

```
Lets go!
```

1.2 Load up counts and DE

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

counts = pd.read_csv(
    "/Users/hughesn/Transcripts/RNA-Seq/Analysis/Data/norml_count_data.csv",
    index_col=0)
xl = pd.ExcelFile(
    "/Users/hughesn/Transcripts/RNA-Seq/Analysis/Data/diff_from_col0:False_onlyDiff:False.xlsx")
sheet_names = xl.sheet_names
dfs = []
for s in sheet_names:
    d = xl.parse(s)
    d['sample'] = s.split("|")[0].replace(" ", "")
    dfs.append(d)

DE = pd.concat(dfs)
DE = DE.rename_axis('gene').sort_values(by=['gene', 'log2FoldChange'],
                                        ascending=[False, False])
print("Loaded data")
```

```
Loaded data
```

2 Inspect Samples

2.1 Creating a distance map of samples using normalised counts

2.1.1 Samples separated

```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.spatial.distance import pdist, squareform

distances = pdist(counts.T.values, metric='euclidean')
dist_matrix = squareform(distances)
dist_df = pd.DataFrame(dist_matrix, columns = counts.columns, index=counts.columns)

sns.clustermap(dist_df)
```

```
<Figure size 720x720 with 4 Axes>
```

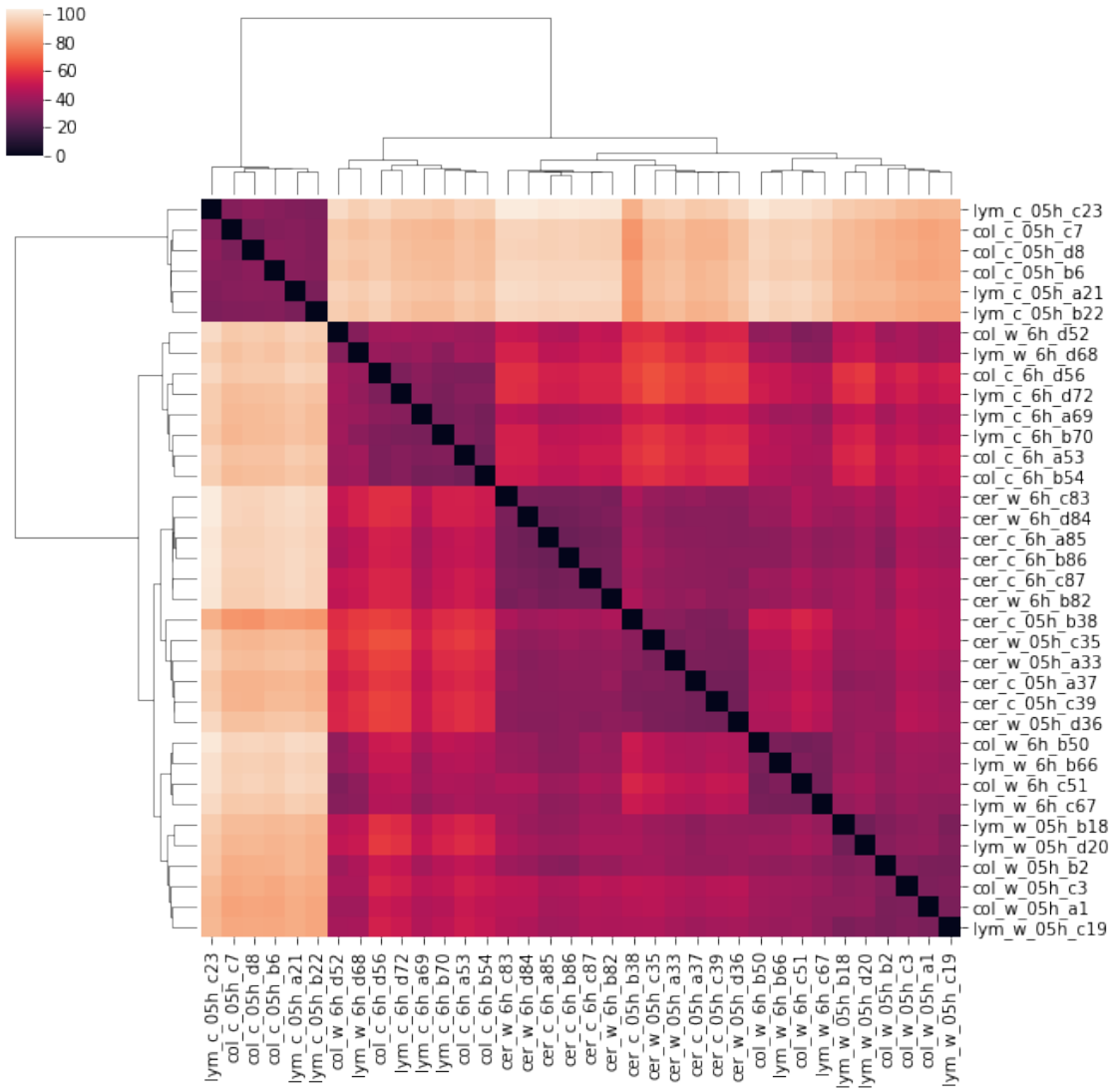

Figure 1: Distance map between samples

2.1.2 Samples together

```
def collapse_counts(counts):
    u_cols = list(set([l.rsplit("_", 1)[0] for l in list(counts.columns)]))
    cols = list(counts.columns)
    ss = []
    for uc in u_cols:
        cs = [c for c in cols if c.startswith(uc)]
        ss.append(counts[cs].sum(axis=1).rename(uc))
    dc = pd.concat(ss, axis=1)
    return dc
collapsed_counts = collapse_counts(counts)
distances = pdist(collapsed_counts.T.values, metric='euclidean')
dist_matrix = squareform(distances)
dist_df = pd.DataFrame(dist_matrix, columns = collapsed_counts.columns, index=collapsed_counts.columns)
sns.clustermap(dist_df)
```

```
<Figure size 720x720 with 4 Axes>
```

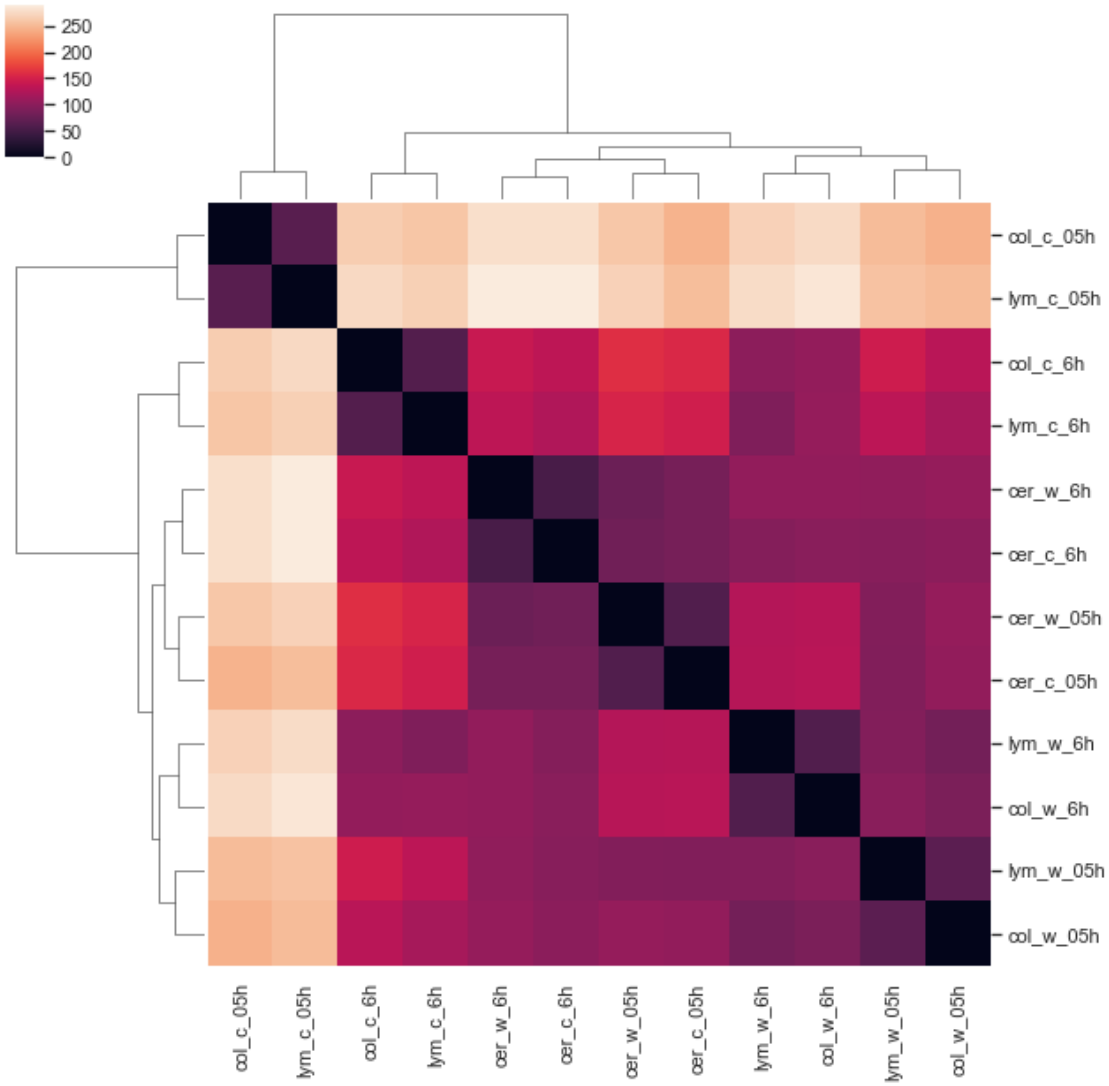

Figure 2: Distance map between samples, pooled together

3 Simple Analysis

3.1 Largest/Lowest expression sum

```
#DE.sum(axis=1).sort_values(by=['log2FoldChange'], ascending=[False]).head(3)

locs = DE[['log2FoldChange']].groupby(['gene']).sum().sort_values(by='log2FoldChange', ascending=False).head(20).index.
top = DE.loc[locs]
top = top.pivot(columns='sample', values='log2FoldChange')

locs = DE[['log2FoldChange']].groupby(['gene']).sum().sort_values(by='log2FoldChange', ascending=True).head(20).index.
bot = DE.loc[locs]
bot = bot.pivot(columns='sample', values='log2FoldChange')

both = pd.concat([top,bot])
both['col_w_05h'] = 0

sns.clustermap(both, cmap='bwr')
```

```
<Figure size 720x720 with 4 Axes>
```

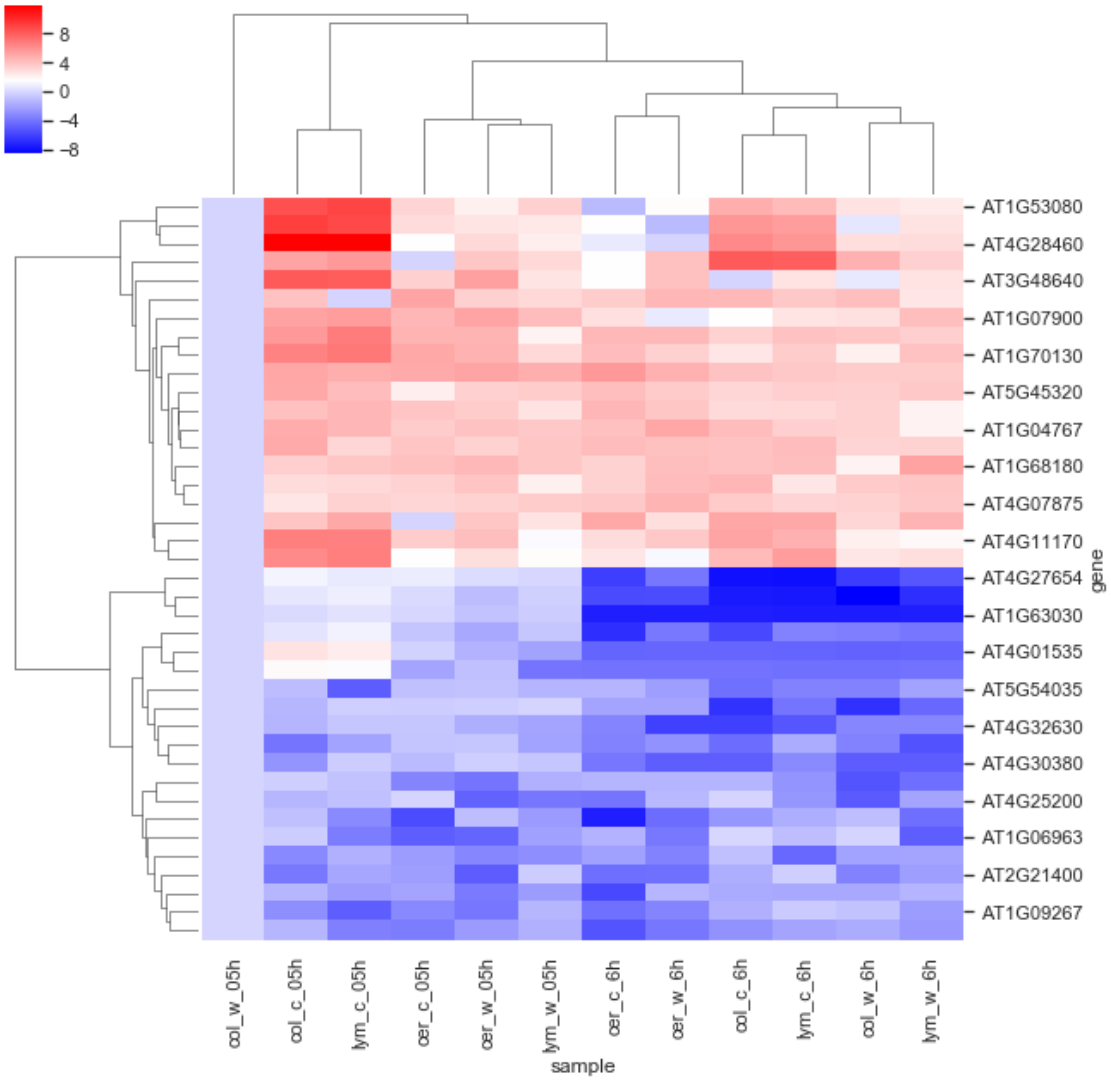

Figure 3: Largest and least DE genes

3.2 PCA on count data

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
sns.set()

cols = list(counts.columns)

counts_geno = [c.split("_")[0] for c in cols]
counts_treat = [c.split("_")[1] for c in cols]
counts_time = [c.split("_")[2] for c in cols]

x = StandardScaler().fit_transform(counts.T.values)

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data=principalComponents, columns=[
                    'principal component 1', 'principal component 2'])

principalDf['genotype'] = counts_geno
principalDf['treatment'] = counts_treat
principalDf['time'] = counts_time

g = sns.FacetGrid(principalDf, col='time', row='genotype', hue='treatment')

g = g.map(plt.scatter, 'principal component 1',
          'principal component 2').add_legend()

print("Explained varience from PC1 & 2 respectively:")
print(pca.explained_variance_ratio_)
```

```
Explained varience from PC1 & 2 respectively: [0.21077632 0.14325373]
```

```
<Figure size 483.925x648 with 6 Axes>
```

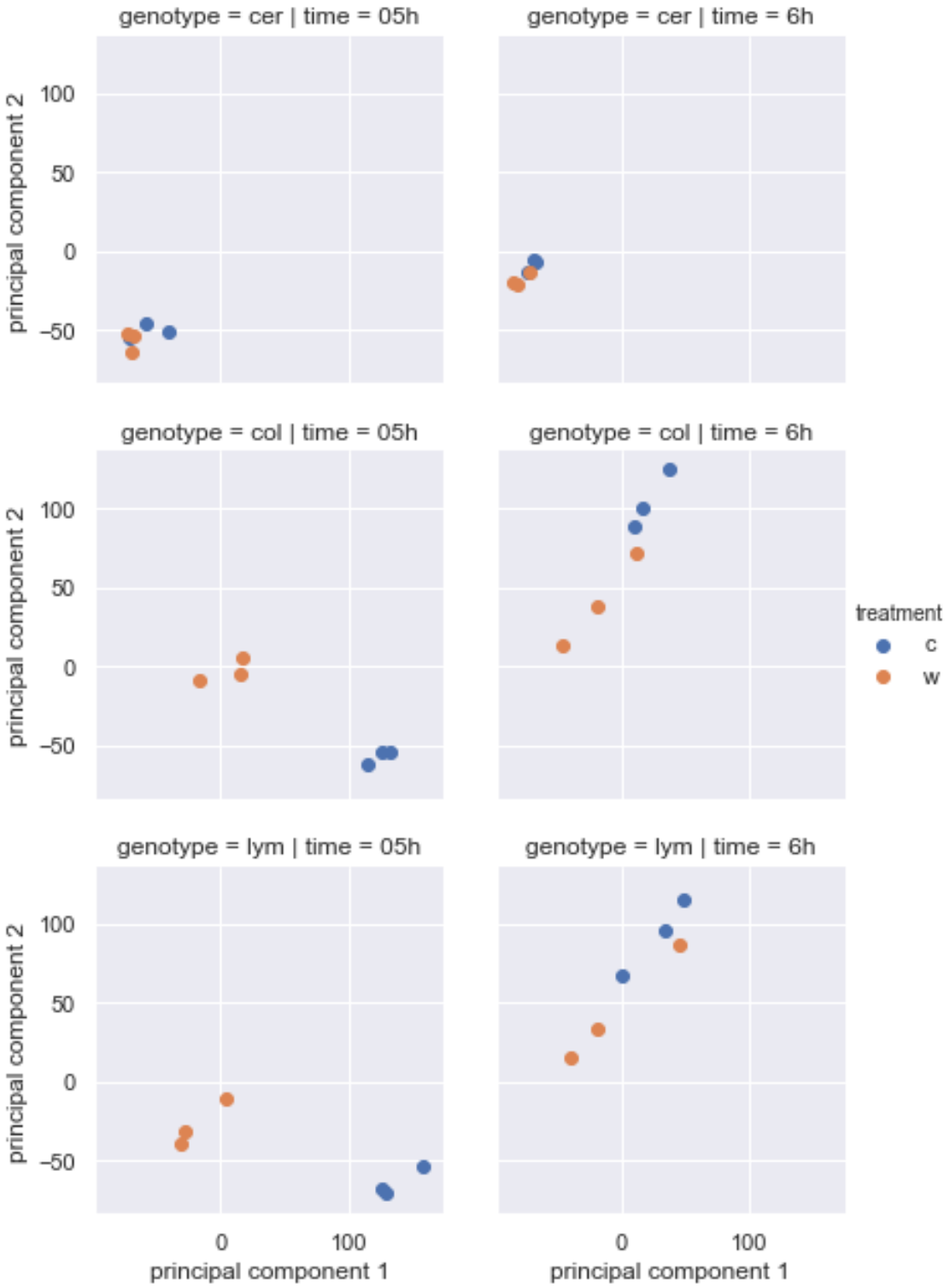

Figure 4: PCA of sample counts