1 Felice ChipSeq Results

Using the ChipSeq Results provided by Adam I was able to use the MACs v1.4 to find the peaks resulting from ChIP of AbcamA, AbcamB, OpbioA, OpbioB. For all comparisons I used the union of the RNAPol-IIA and RNAPol-IIB as the control. I used the option for MACs which controls for the different dataset sizes. All other arguements were the defaults. I'm assuming the Lanes listed as InputB and InputC are controls of some sort.

```
import os
import os.path
from subprocess import check_call
import shlex
import tempfile
import glob
from pandas import *

os.chdir('/home/will/Tip60Analysis/Data/DerivedData/')
```

1.1 Mapping Summary

The MACS algorithm was able to run properly on all samples and did not produce any warnings.

```
macs_result_files = glob.glob('*/NA_peaks.bed')
macs_results = []
col_names = ['Chrom', 'Start', 'End', 'PeakName', 'Score']
for res in macs_result_files:
    anal = res.split('/', 1)[0]
    tdata = read_csv(res, sep='\t', names=col_names)
    tdata['Analysis'] = anal
    macs_results.append(tdata.copy())

macs_res = concat(macs_results, axis = 0, ignore_index=True)
```

Data Extraction

1.1.1 Results

```
print macs_res['Analysis'].value_counts()
```

```
OpbioB    9816
OpbioA    6372
AbcamB    4945
AbcamA    3981
InputC      46
InputB      37
```

Number of Peaks This pretty consistent with my previous experience of 2000-10000 peaks.

```
chrom_dist = crosstab(cols = macs_res['Analysis'],
                      rows = macs_res['Chrom'])
print chrom_dist.drop('dmel_mitochondrion_genome')
```

```
Analysis  AbcamA  AbcamB  InputB  InputC  OpbioA  OpbioB
2L            67      78      15      18      90     125
2LHet         12      20       0       0      22      32
2R           101     115       3       4     145     189
2RHet         72      93       1       1     137     206
3L            87     107       8       5     129     202
3LHet         72      81       0       1     121     182
3R           113     117       9       8     141     181
3RHet         53      63       0       0     103     152
4              4       5       0       0       6      10
U            555     699       0       3     960    1592
Uextra      2767    3471       0       5    4395    6760
X             61      76       0       0     104     164
XHet           6       8       0       0       7       9
YHet           4       5       0       0       5       6
```

Peak Distribution on Chromosomes

```
prom_files = glob.glob('*/promoters.bed')
prom_results = []
col_names = ['Chrom', 'Start', 'End', 'Genbank',
             'JunkA', 'Strand', 'JunkB', 'JunkC',
             'JunkD','JunkE','JunkF','JunkG']
drop_cols = [col for col in col_names if col.startswith('Junk')]
for res in prom_files:
    anal = res.split('/', 1)[0]
    tdata = read_csv(res, sep='\t', names=col_names)
    tdata['Analysis'] = anal
    prom_results.append(tdata.drop(drop_cols, axis = 1))

prom_res = concat(prom_results, axis = 0, ignore_index=True)
```

Data Extraction

```
num_genes = prom_res.pivot_table(rows = 'Analysis',
                                 values = 'Genbank',
                                 aggfunc = lambda x: len(x.unique()))
print num_genes
```

```
Analysis
AbcamA     1017
AbcamB     1182
InputB      214
InputC      164
OpbioA     1338
OpbioB     1579
```

```
Name: Genbank
```

Number of 'Controlled' Genes Found Again, pretty consistent with my previous results. In this case I used the 10Kb upstream of a gene as the 'promoter region'.

```
from itertools import product
overlaps = DataFrame(index = num_genes.index,
                     columns = num_genes.index)
for a, b in product(num_genes.index, repeat = 2):
    maskA = prom_res['Analysis'] == a
    maskB = prom_res['Analysis'] == b
    genesA = prom_res['Genbank'][maskA]
    genesB = prom_res['Genbank'][maskB]

    overlaps[a][b] = len(set(genesA) & set(genesB))

print overlaps
```

```
Analysis AbcamA AbcamB InputB InputC OpbioA OpbioB
Analysis
AbcamA     1017    980      0      5   1014    992
AbcamB      980   1182      0      5   1160   1127
InputB        0      0    214     34      0      0
InputC        5      5     34    164      5      0
OpbioA     1014   1160      0      5   1338   1281
OpbioB      992   1127      0      0   1281   1579
```

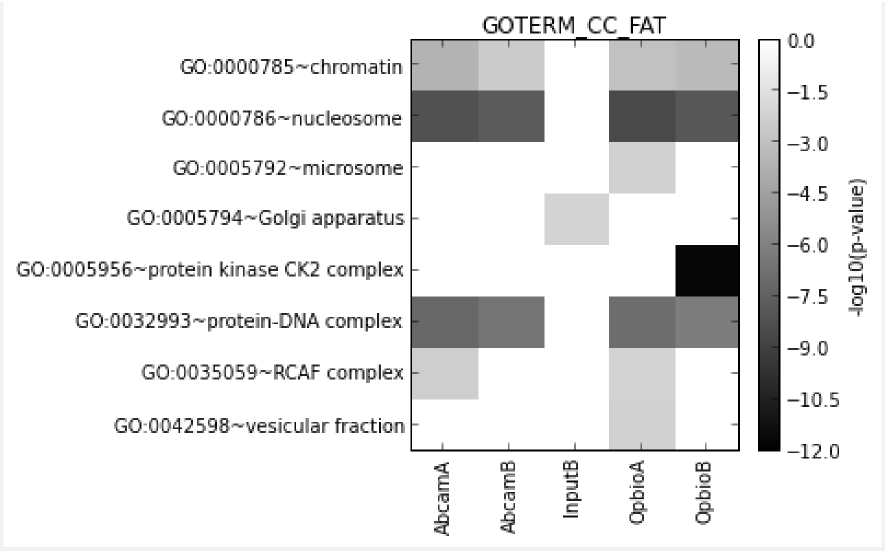
Overlapping Genes You can see that there is quite a bit of overlap amongst all of the proteins.

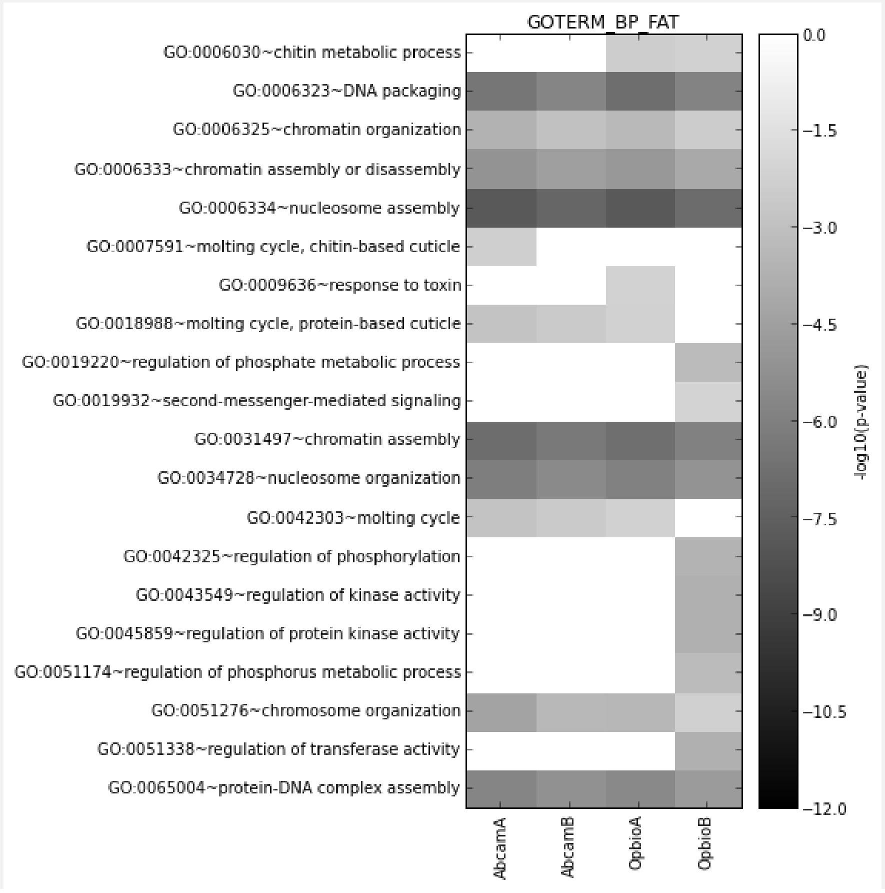
```
charts = glob.glob('*/chart_*.txt')
group_data = []
for chart in charts:
    anal = chart.split('/')[0]
    tdata = read_csv(chart, sep = '\t')
    tdata['Analysis'] = anal
    group_data.append(tdata.copy())

all_data = concat(group_data, axis = 0, ignore_index=True)
```

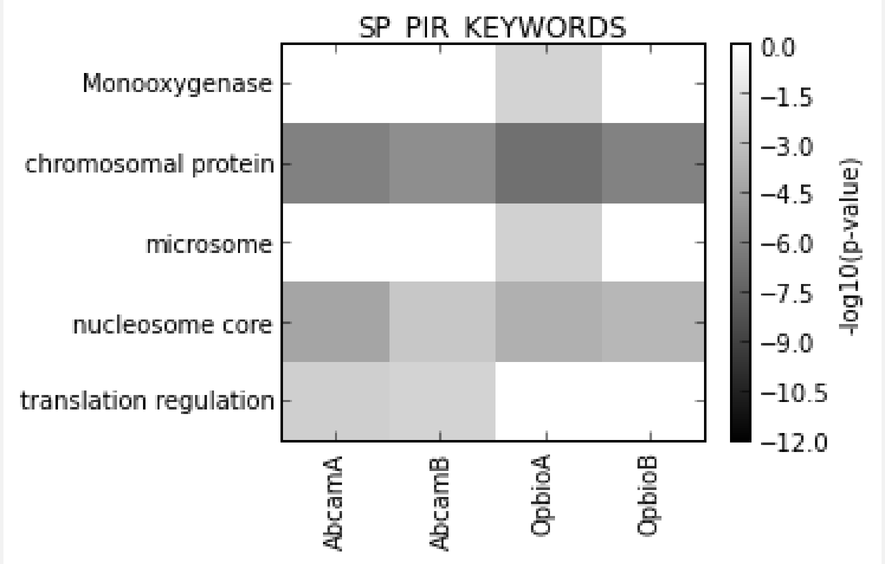
Data Extraction

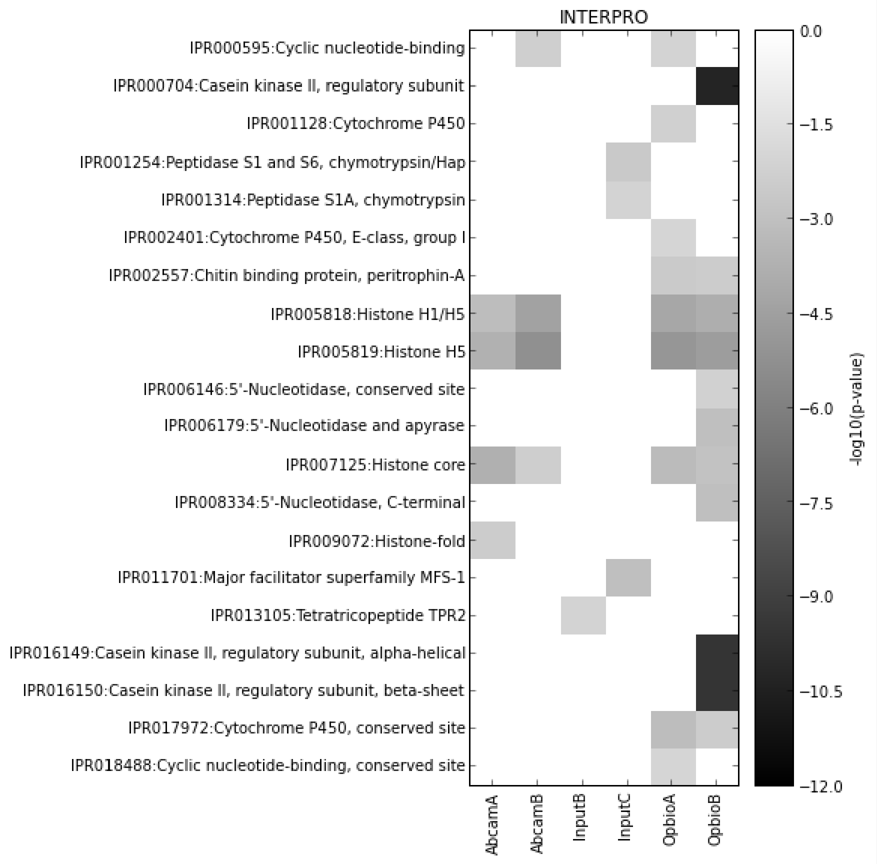
```
wanted_cols = ['Analysis', 'Category', 'Term',
               'Count', '%', 'List Total',
               'Pop Hits', 'Pop Total',
               'PValue', 'Benjamini']
all_data[wanted_cols].to_csv('../Results/enrichment_results.tsv',
                             sep = '\t', index = False)
```

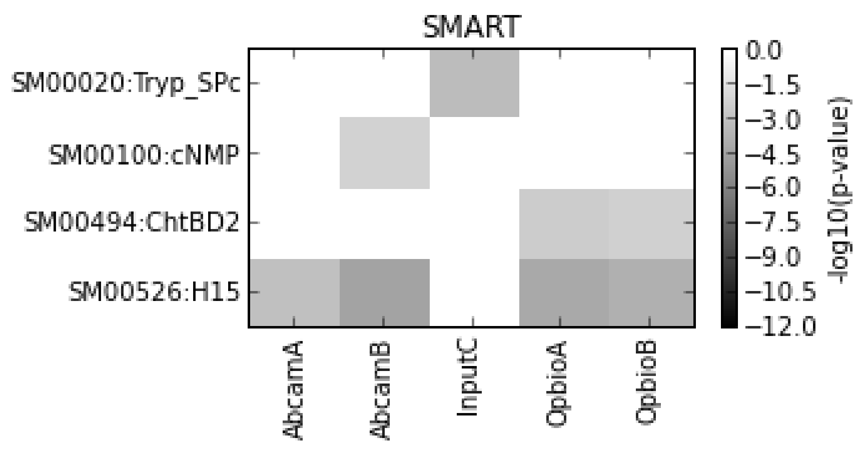
```
from matplotlib import pyplot as plt
import numpy as np
from pylab import get_cmap

sig_results = all_data['PValue'] < 0.01
sig_data = all_data[sig_results]
row_frac = 8.0/20.0

for cat in all_data['Category'].unique():
    wcat = sig_data['Category'] == cat
    if wcat.sum() == 0:
        continue
    res = crosstab(rows = sig_data['Term'][wcat],
                   cols = sig_data['Analysis'][wcat],
                   values = sig_data['PValue'][wcat],
                   aggfunc = min)
    nrows = len(res.index)
    plt.figure(figsize = (4, int(row_frac*nrows)+1))
    plt.imshow(np.log10(res.values), aspect = 'auto',
               interpolation = 'nearest', cmap = get_cmap('gray'))
    cbar = plt.colorbar()
    cbar.set_label('-log10(p-value)')
    plt.clim([-12,0])
    plt.xticks(range(len(res.columns)),
               res.columns, rotation = 90);
    plt.yticks(range(len(res.index)),
               res.index);
    plt.title(cat)

bigres = crosstab(rows = sig_data['Term'],
                   cols = sig_data['Analysis'],
                   values = sig_data['PValue'],
                   aggfunc = min)
bigres.to_csv('../Results/enrichment_table.tsv')
```

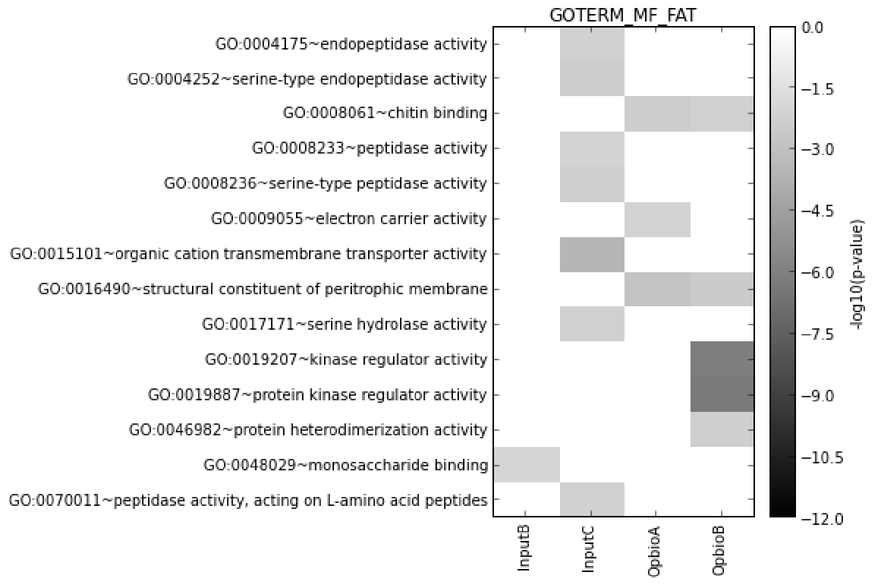

GOTERM_CC_FAT
GO:0000785~chromatin
GO:0000786~nucleosome
GO:0005792~microsome
GO:0005794~Golgi apparatus
GO:0005956~protein kinase CK2 complex
GO:0032993~protein-DNA complex
GO:0035059~RCAF complex
GO:0042598~vesicular fraction
AbcamA
AbcamB
InputB
OpbioA
OpbioB
0.0
−1.5
−3.0
−4.5
−6.0
−7.5
−9.0
−10.5
−12.0
-log10(p-value)

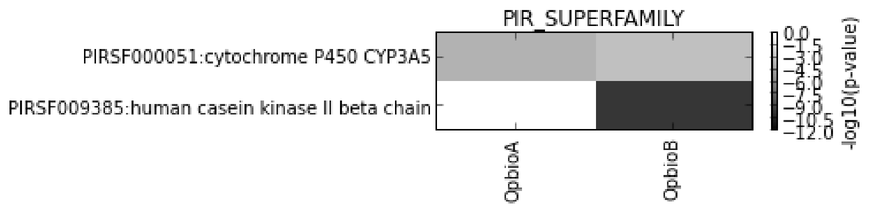

GOTERM_BP_FAT
GO:0006030~chitin metabolic process
GO:0006323~DNA packaging
GO:0006325~chromatin organization
GO:0006333~chromatin assembly or disassembly
GO:0006334~nucleosome assembly
GO:0007591~molting cycle, chitin-based cuticle
GO:0009636~response to toxin
GO:0018988~molting cycle, protein-based cuticle
GO:0019220~regulation of phosphate metabolic process
GO:0019932~second-messenger-mediated signaling
GO:0031497~chromatin assembly
GO:0034728~nucleosome organization
GO:0042303~molting cycle
GO:0042325~regulation of phosphorylation
GO:0043549~regulation of kinase activity
GO:0045859~regulation of protein kinase activity
GO:0051174~regulation of phosphorus metabolic process
GO:0051276~chromosome organization
GO:0051338~regulation of transferase activity
GO:0065004~protein-DNA complex assembly
AbcamA
AbcamB
OpbioA
OpbioB
0.0
−1.5
−3.0
−4.5
−6.0
−7.5
−9.0
−10.5
−12.0
-log10(p-value)

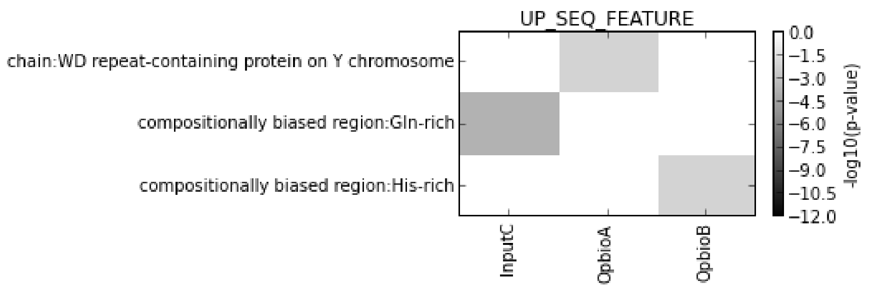

SP_PIR_KEYWORDS
Monooxygenase
chromosomal protein
microsome
nucleosome core
translation regulation
AbcamA
AbcamB
OpbioA
OpbioB
0.0
−1.5
−3.0
−4.5
−6.0
−7.5
−9.0
−10.5
−12.0
-log10(p-value)

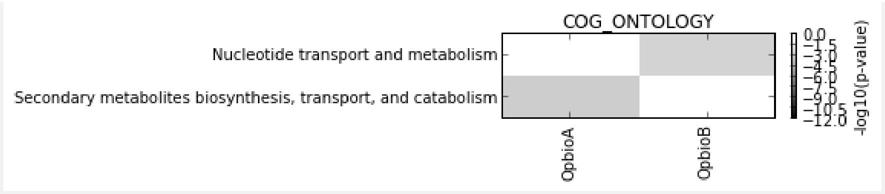

INTERPRO
IPR000595:Cyclic nucleotide-binding
IPR000704:Casein kinase II, regulatory subunit
IPR001128:Cytochrome P450
IPR001254:Peptidase S1 and S6, chymotrypsin/Hap
IPR001314:Peptidase S1A, chymotrypsin
IPR002401:Cytochrome P450, E-class, group I
IPR002557:Chitin binding protein, peritrophin-A
IPR005818:Histone H1/H5
IPR005819:Histone H5
IPR006146:5'-Nucleotidase, conserved site
IPR006179:5'-Nucleotidase and apyrase
IPR007125:Histone core
IPR008334:5'-Nucleotidase, C-terminal
IPR009072:Histone-fold
IPR011701:Major facilitator superfamily MFS-1
IPR013105:Tetratricopeptide TPR2
IPR016149:Casein kinase II, regulatory subunit, alpha-helical
IPR016150:Casein kinase II, regulatory subunit, beta-sheet
IPR017972:Cytochrome P450, conserved site
IPR018488:Cyclic nucleotide-binding, conserved site
AbcamA
AbcamB
InputB
InputC
OpbioA
OpbioB
0.0
−1.5
−3.0
−4.5
−6.0
−7.5
−9.0
−10.5
−12.0
-log10(p-value)


SMART
SM00020:Tryp_SPc
SM00100:cNMP
SM00494:ChtBD2
SM00526:H15
AbcamA
AbcamB
InputC
OpbioA
OpbioB
0.0
−1.5
−3.0
−4.5
−6.0
−7.5
−9.0
−10.5
−12.0
-log10(p-value)


GOTERM_MF_FAT
GO:0004175~endopeptidase activity
GO:0004252~serine-type endopeptidase activity
GO:0008061~chitin binding
GO:0008233~peptidase activity
GO:0008236~serine-type peptidase activity
GO:0009055~electron carrier activity
GO:0015101~organic cation transmembrane transporter activity
GO:0016490~structural constituent of peritrophic membrane
GO:0017171~serine hydrolase activity
GO:0019207~kinase regulator activity
GO:0019887~protein kinase regulator activity
GO:0046982~protein heterodimerization activity
GO:0048029~monosaccharide binding
GO:0070011~peptidase activity, acting on L-amino acid peptides
InputB
InputC
OpbioA
OpbioB
0.0
−1.5
−3.0
−4.5
−6.0
−7.5
−9.0
−10.5
−12.0
-log10(p-value)


PIR_SUPERFAMILY
PIRSF000051:cytochrome P450 CYP3A5
PIRSF009385:human casein kinase II beta chain
OpbioA
OpbioB
0.0
−1.5
−3.0
−4.5
−6.0
−7.5
−9.0
−10.5
−12.0
-log10(p-value)


UP_SEQ_FEATURE
chain:WD repeat-containing protein on Y chromosome
compositionally biased region:Gln-rich
compositionally biased region:His-rich
InputC
OpbioA
OpbioB
0.0
−1.5
−3.0
−4.5
−6.0
−7.5
−9.0
−10.5
−12.0
-log10(p-value)



Significant Terms and Pathways These results show that there is a good deal of overlap in the 'functional space' between the different TFs. For example, in the INTERPRO group you can see a strong signal in the 'Histone Binding'. There is also quiet a bit of chromosomal organization and chromatin binding/assembly/etc in the GO BP group. The full results are in the enrichment_table.tsv results.

```
gene_names = read_csv('../gene_names.txt', sep = '\t')
ndata = merge(gene_names, prom_res,
              left_on = 'GENBANK_ACCESSION', right_on = 'Genbank')
wanted_cols = ['Analysis', 'Name', 'Chrom', 'Start', 'End', 'Strand']
ndata[wanted_cols].to_csv('../Results/FoundPromoters.tsv', sep = '\t')
```

Annotation Information