cummeRbund: Visualization and Exploration of Cufflinks High-throughput Sequencing Data

Loyal A. Goff, Cole Trapnell

1 April, 2011

Contents

1 Introduction **1**

2 CummeRbund Classes **2**
2.1 CuffSet Class 2
2.2 CuffData Class 2
2.3 CuffFeatureSet Class 2
2.4 CuffFeature Class 3

3 Reading cuffdiff output **4**
3.1 Adding additional feature annotation 4

4 Global statistics **4**

5 Accessing Data **9**
5.1 Writing your own SQL accessors 10

6 Creating Gene Sets **11**
6.1 Geneset level plots 11

7 Individual Genes **16**
7.1 Gene-level plots 16

8 Miscellaneous **17**

9 Session info **20**

1 Introduction

cummeRbund is a visualization package for Cufflinks high-throughput sequencing data. The base class, *cuffSet* is a 'pointer' to cufflinks data that are stored out-of-memory in a sqlite database.

2 CummeRbund Classes

2.1 CuffSet Class

A pointer class to control access to the sqlite tables holding the Cufflinks data. The primary slot is @DB which contains the RSQLite connection object. The additional slots (genes, isoforms, TSS, and CDS) are each instances of the *CuffData* class and are pointers to sets of tables for each data subtype. This is the default class created by *readCufflinks*. By default, *CuffData* accessor methods applied to a *CuffSet* class will operate on the 'genes' slot.

2.2 CuffData Class

The *CuffData* class is also a pointer class to the SQL backend, but each instance is specific for a data subtype (genes, isoforms, TSS, CDS). Again, there is an @DB slot that contains the RSQLite connection object. There are several accessor, setter, and plotting methods that allow for global analysis of all features within a *CuffData* class.Subsetting is currently being re-written, however, it is primarily done through the 'gene_id' field. Available slots for the CuffData class are:

- DB: RSQLite connection object
- tables: A *list* of tables in the SQLite DB that contain the cufflinks data.
- filters: A *list* of filters for subsetting (not implemented yet).
- type: A *character* field describing the data (ie. 'genes','isoforms','TSS','CDS','other')
- idField: The name of the identifying index field for this object (eg. 'gene_id' for type='gene', or 'isoform_id' for type='isoform')

Making the best use of either the CuffSet or CuffData classes will enable you to keep the entire dataset out of memory and significantly improve performance for large cufflinks datasets.

2.3 CuffFeatureSet Class

The *CuffFeatureSet* class is a data-storage container that holds all available data for a pre-determined list of features. Slots for FPKM data, differential regulation data, and feature-level annotation are all available. Unlike the previous classes, this class contains no connection information to the SQL database, but rather contains several slots with *data.frame* objects storing multiple-features worth of information. There are available accessors, and plotting methods that are designed to present multiple-features worth of information (eg. heatmaps, scatterplots, etc) Available slots for a *CuffFeatureSet* object include:

- annotation: Holds all feature-level annotation information for all features in object.

- fpkm: A data frame of FPKM data across all samples, for all features in object.
- diff: A data frame of differential expression/regulation data for all features in object.

A specialized sub-class of *CuffFeatureSet* is the *CuffGeneSet* class. This subclass adds additional slots to contain all isoforms, TSS, and CDS information for a given set of gene_ids. The *CuffGeneSet* class is designed to aggregate all relevant information for a set of genes into one object for easy analysis and/or manipulation. The *CuffGeneSet* object adds the following slots:

- ids: A 'character' list of all gene_ids used in object.
- isoforms: A *CuffFeatureSet* object for all isoforms of genes in object.
- TSS: A *CuffFeatureSet* object for all TSS of genes in object.
- CDS: A *CuffFeatureSet* object for all CDS of genes in object.

2.4 CuffFeature Class

The *CuffFeature* class is designed for single-feature-level data analysis and plotting. The methods available for this object are designed to analyze or visualize information about a specific feature. This is a 'data' object, as opposed to a 'pointer' object to the database backend. There is a validity requirement that a *CuffFeature* object only point to data from a single feature. Available slots for a *CuffFeature* object include:

- annotation: Holds feature-level annotation information for a given feature.
- fpkm: A data frame of FPKM data across all samples for a given feature.
- diff: A data frame of differential expression/regulation data for a given feature.

A specialized sub-class of *CuffFeature* is the *CuffGene* class. This subclass adds additional slots to contain all isoform, TSS, and CDS information for a given gene. The *CuffGene* object adds the following slots:

- id: The common 'gene_id' for all data in object
- isoforms: A *CuffFeature* object for all isoforms of a given gene.
- TSS: A *CuffFeature* object for all TSS of a given gene.
- CDS: A *CuffFeature* object for all CDS of a given gene.

Note: Future versions of cummeRbund may try to collapse the redundant functionality of the CuffFeature and CuffFeatureSet classes.

3 Reading cuffdiff output

One of the principle benefits of using cummeRbund is that data are stored in a SQLite database. This allows for out-of-memory analysis of data, quick retrieval, and only a one-time cost to setup the tables. By default, cummeRbund assumes that all output files from cuffdiff are in the current working directory. To read these files, populate the 'cuffData.db' database backend, and return the *CuffSet* pointer object, you can do the following.

```
> cuff <- readCufflinks(dir = fileDir)
> cuff

CuffSet instance with:
         3 samples
         35695 genes
         102603 isoforms
         54339 TSS
         20584 CDS
```

Again, by default *dir* is assumed to be the current working directory. Should you need to rebuild the SQLite backend for any reason, you can add the option *rebuild=T* to *readCufflinks*. Once the database is created, *readCufflinks* will default to using the SQL backend and should not need to rebuild this database. Each R session should begin with a call to *readCufflinks* so as to initialize the database connection and create an object with the appropriate RSQLite connection information.

3.1 Adding additional feature annotation

Gene- or feature-level annotation can be permanently added to the database tables for future querying. If you have a data.frame where the first column contains the 'tracking_id' (eg. 'gene_id' for genes, 'isoform_id' for isoforms, etc). You can easily add feature level annotation using the `addFeatures()` function:

By default, features added to a *CuffSet* object are assumed to be gene-level annotations, but the level can selected using the argument *level*. Features added to a *CuffData* object are assumed to be of the same type as the object@type value (e.g. gene-level features for 'genes', isoform-level features for isoforms, etc.)

4 Global statistics

Several plotting methods are available that allow for quality-control or global analysis of cufflinks data. For example, to assess the distributions of FPKM scores across samples, you can use the *csDensity* plot (Figure 1).

```
> dens <- csDensity(cuff@genes)
```

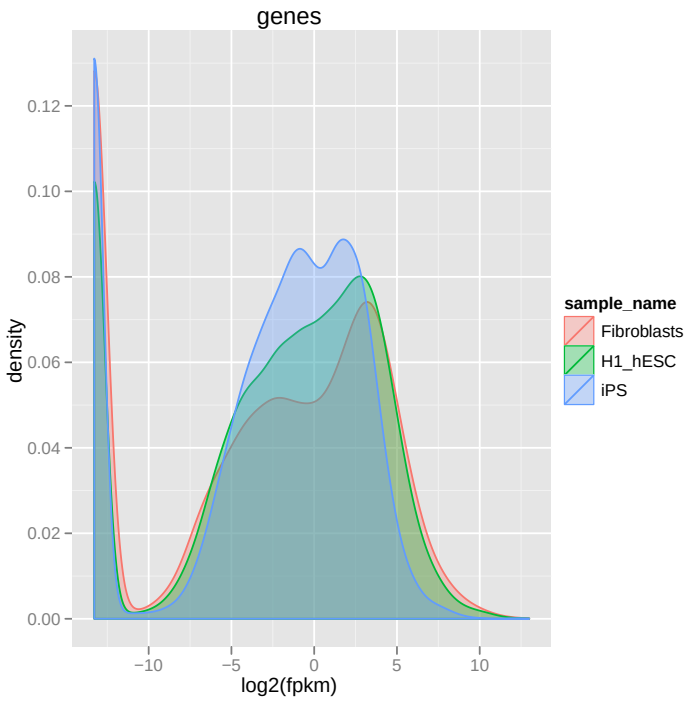

Figure 1: Density plot per sample of cufflinks output FPKM values.

Boxplots can be visualized using the *csBoxplot* method (Figure 2).

```
> b <- csBoxplot(cuff@genes)
```

Pairwise comparisons can be made by using *csScatter*. You must specify the sample names to use for the x and y axes:

```
> s <- csScatter(cuff@genes, "H1_hESC", "Fibroblasts",
+     smooth = T)
```

Volcano plots are also available for the *CuffData* objects. Again, you must specify the comparisons by sample name.

```
> v <- csVolcano(cuff@genes, "H1_hESC", "Fibroblasts")
```

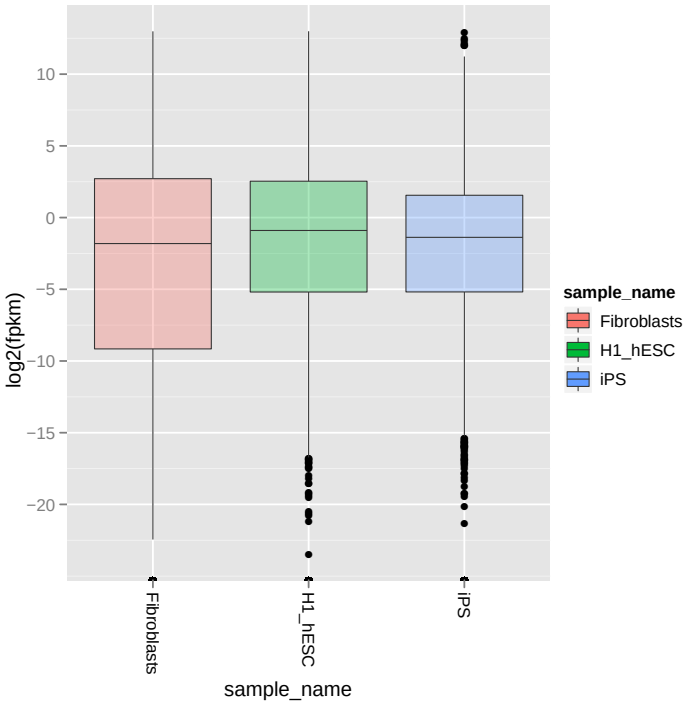

Figure 2: Box plot of FPKM values from cufflinks output.

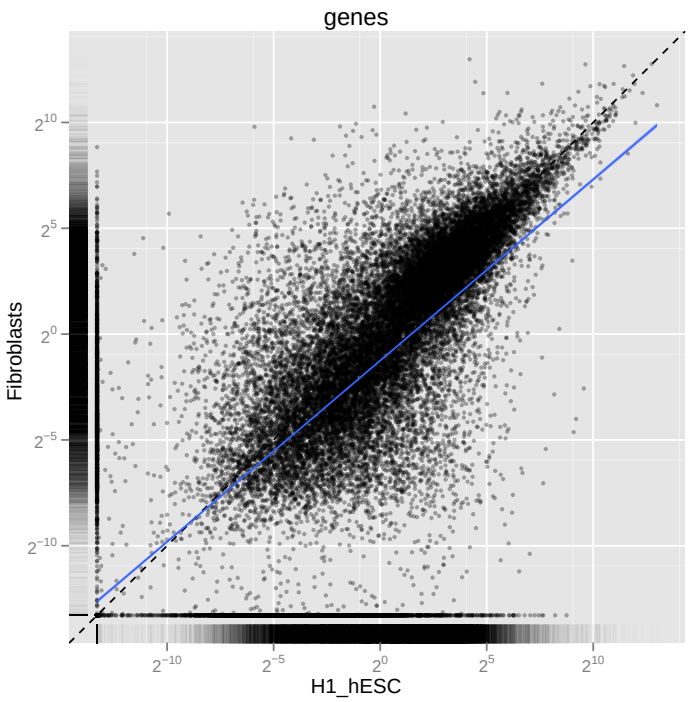

Figure 3: Scatter plot comparing the FPKM values of two samples from cufflinks output.

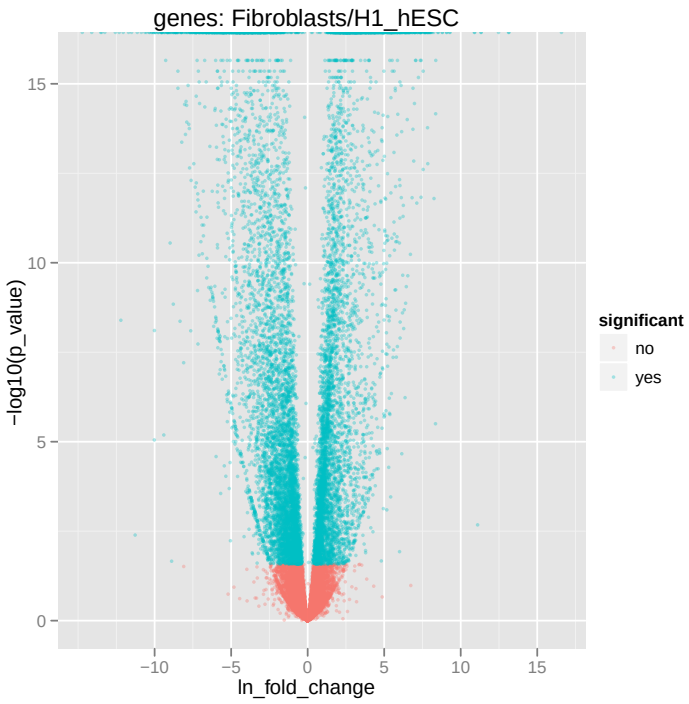

Figure 4: Volcano plot of ln fold change vs significance.

5 Accessing Data

Feature-level information can be accessed directly from a *CuffData* object using the *fpkm*, *diffData*, or *features* methods:

```
> gene.features <- features(cuff@genes)
> head(gene.features)

      gene_id class_code nearest_ref_id gene_short_name
1 XLOC_000001       <NA>           <NA>   linc-OR4F16-4
2 XLOC_000002       <NA>           <NA>   linc-OR4F16-3
3 XLOC_000003       <NA>           <NA>   linc-OR4F16-2
4 XLOC_000004       <NA>           <NA>   linc-OR4F16-1
5 XLOC_000005       <NA>           <NA>   linc-SAMD11-9
6 XLOC_000006       <NA>           <NA>   linc-SAMD11-8
               locus length coverage status gene_id
1  chr1:77369-328580     NA       NA   FAIL    <NA>
2  chr1:77369-328580     NA       NA   FAIL    <NA>
3  chr1:77369-328580     NA       NA   FAIL    <NA>
4 chr1:329783-565234     NA       NA     OK    <NA>
5 chr1:329783-565234     NA       NA     OK    <NA>
6 chr1:329783-565234     NA       NA     OK    <NA>

> gene.fpkm <- fpkm(cuff@genes)
> head(gene.fpkm)

                gene_id sample_name       fpkm   conf_hi conf_lo
1 ucscCodingXLOC_000001 Fibroblasts 0.04082990 0.0879090       0
2 ucscCodingXLOC_000001     H1_hESC 0.00375359 0.0197377       0
3 ucscCodingXLOC_000001         iPS 0.09010400 0.1934730       0
4 ucscCodingXLOC_000002 Fibroblasts 0.00000000 0.0000000       0
5 ucscCodingXLOC_000002     H1_hESC 0.00000000 0.0000000       0
6 ucscCodingXLOC_000002         iPS 0.00000000 0.0000000       0

> isoform.fpkm <- fpkm(cuff@isoforms)
> head(isoform.fpkm)

      isoform_id sample_name      fpkm  conf_hi   conf_lo
1 TCONS_00000001 Fibroblasts 0.0316231 0.387281 0.0000000
2 TCONS_00000001     H1_hESC 0.3640070 0.511629 0.2163860
3 TCONS_00000001         iPS 0.1144880 0.199104 0.0298726
4 TCONS_00000002 Fibroblasts 0.0000000 0.000000 0.0000000
5 TCONS_00000002     H1_hESC 0.1092120 0.179686 0.0387377
6 TCONS_00000002         iPS 0.1314640 0.236988 0.0259410

> gene.diff <- diffData(cuff@genes)
> head(gene.diff)
```

```
      gene_id sample_1    sample_2 status    value_1    value_2
1 XLOC_000001  H1_hESC Fibroblasts   FAIL 0.00000000 0.00000000
2 XLOC_000002  H1_hESC Fibroblasts   FAIL 0.00000000 0.00000000
3 XLOC_000003  H1_hESC Fibroblasts   FAIL 0.00000000 0.00000000
4 XLOC_000004  H1_hESC Fibroblasts     OK 0.06015940 0.04760850
5 XLOC_000005  H1_hESC Fibroblasts     OK 0.00343381 0.06554600
6 XLOC_000006  H1_hESC Fibroblasts     OK 0.00253166 0.00619907
  ln_fold_change test_stat     p_value     q_value significant
1       0.000000   0.00000 1.00000e+00 1.00000e+00          no
2       0.000000   0.00000 1.00000e+00 1.00000e+00          no
3       0.000000   0.00000 1.00000e+00 1.00000e+00          no
4      -0.233987   1.79137 7.32335e-02 1.19646e-01          no
5       2.949080  -3.14964 1.63470e-03 4.33945e-03         yes
6       0.895526  -7.33553 2.20934e-13 2.59665e-12         yes
```

Vectors of sample names and feature names are available by using the *samples* and *featureNames* methods:

```
> sample.names <- samples(cuff@genes)
> head(sample.names)

[1] "H1_hESC"     "Fibroblasts" "iPS"

> gene.featurenames <- featureNames(cuff@genes)
> head(gene.featurenames)

[1] "XLOC_000001" "XLOC_000002" "XLOC_000003" "XLOC_000004"
[5] "XLOC_000005" "XLOC_000006"
```

To facilitate Bioconductor-like operations, an 'FPKM-matrix' can be returned easily using the *fpkmMatrix* method:

```
> gene.matrix <- fpkmMatrix(cuff@genes)
> head(gene.matrix)

               H1_hESC Fibroblasts       iPS
XLOC_000001 6.68605000  0.09024540 1.8593600
XLOC_000002 0.03580660  0.00000000 0.5792420
XLOC_000003 5.14548000  0.05086520 1.0566700
XLOC_000004 0.06015940  0.04760850 0.1490590
XLOC_000005 0.00343381  0.06554600 0.0411692
XLOC_000006 0.00253166  0.00619907 0.0000000
```

5.1 Writing your own SQL accessors

Since the cufflinks is a SQLite database backend, if you are familiar with SQL and/or RSQLite query construction, you can simply design your own SQL queries to access the data that you are after. PUT DATABASE SCHEMA HERE...

6 Creating Gene Sets

Gene Sets (stored in a *CuffGeneSet* object) can be created using the *getGenes* method on a CuffSet object. You must first create a vector of 'gene_ids' to identify the genes you wish to select:

```
> myGeneIds <- sample(features(cuff@genes)$gene_id,
+     20)
> myGeneIds
```

```
 [1] "ucscCodingXLOC_020680" "ucscCodingXLOC_003532"
 [3] "ucscCodingXLOC_017360" "ucscCodingXLOC_004486"
 [5] "ucscCodingXLOC_024974" "ucscCodingXLOC_006835"
 [7] "XLOC_009866"           "XLOC_015046"
 [9] "XLOC_011832"           "XLOC_007758"
[11] "XLOC_013711"           "XLOC_003103"
[13] "XLOC_008598"           "ucscCodingXLOC_015439"
[15] "XLOC_012971"           "ucscCodingXLOC_022751"
[17] "XLOC_000047"           "XLOC_005338"
[19] "XLOC_002657"           "ucscCodingXLOC_005465"
```

```
> myGenes <- getGenes(cuff, myGeneIds)
> myGenes
```

```
CuffGeneSet instance for genes ucscCodingXLOC_020680 ucscCodingXLOC_003532 ucscCodingXLOC_01
Short name:         linc-RPL22-AS linc-SERPINH1-AS linc-RPLP0-AS linc-LOC653550-14 linc-ORC2
Slots:
         annotation
         fpkm
         diff
         isoforms         CuffFeatureSet instance of size 1
         TSS                 CuffFeatureSet instance of size 1
         CDS                 CuffFeatureSet instance of size 1
```

The same *fpkm*, *fpkmMatrix*, *features*, *diffData*, *samples*, and *featureNames* are available for instances of the *CuffGeneSet* class.

6.1 Geneset level plots

There are several plotting functions available for gene-set-level visualization:

```
> h <- csHeatmap(myGenes, cluster = "both")
```



```
> s <- csScatter(myGenes, "Fibroblasts", "H1_hESC",
+     smooth = T)
```



```
> v <- csVolcano(myGenes, cluster = "both")
```

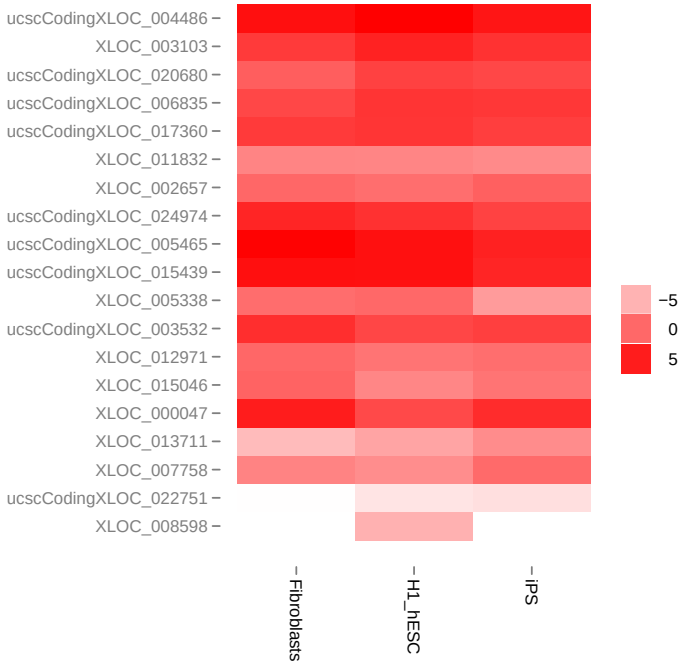

Figure 5: Heatmap of FPKM values for a random sample of 20 genes.

Similar plots can be made for all sub-level features of a *CuffGeneSet* class by specifiying which slot you would like to plot (eg. *@isoforms*, *@TSS*, *@CDS*).

```
> ih <- csHeatmap(myGenes@isoforms, cluster = "both")
```

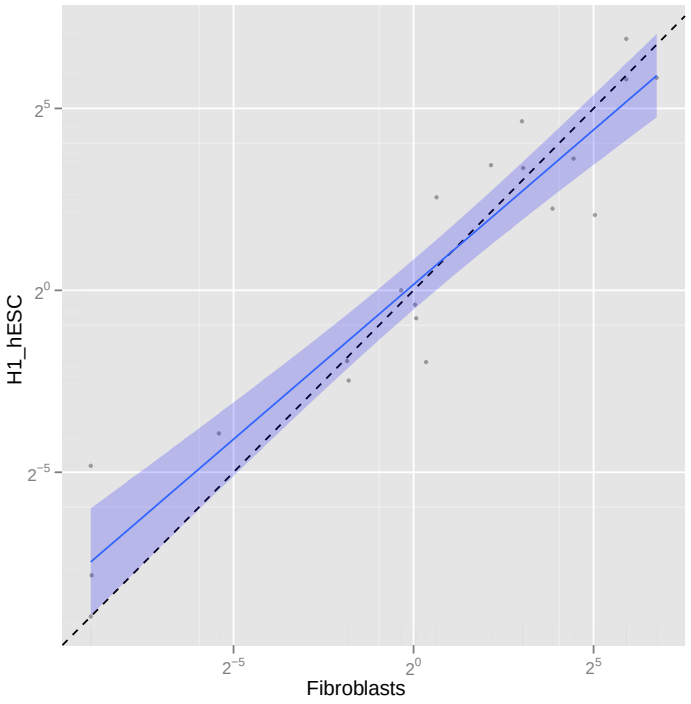

Figure 6: Scatterplot of FPKM values for a random sample of 20 genes between Fibroblasts and H1_hESC.

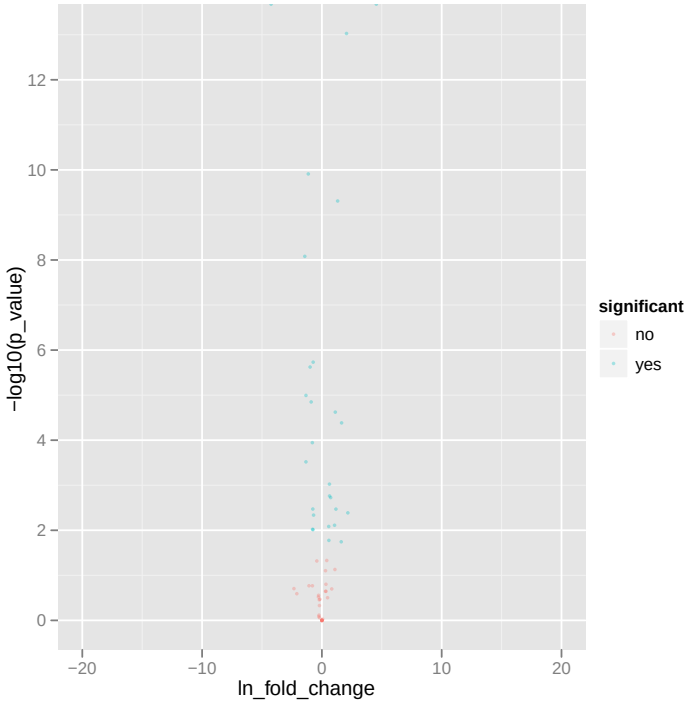

Figure 7: Volcano plot of FPKM vs significance values for a random sample of 20 genes between 2 conditions.

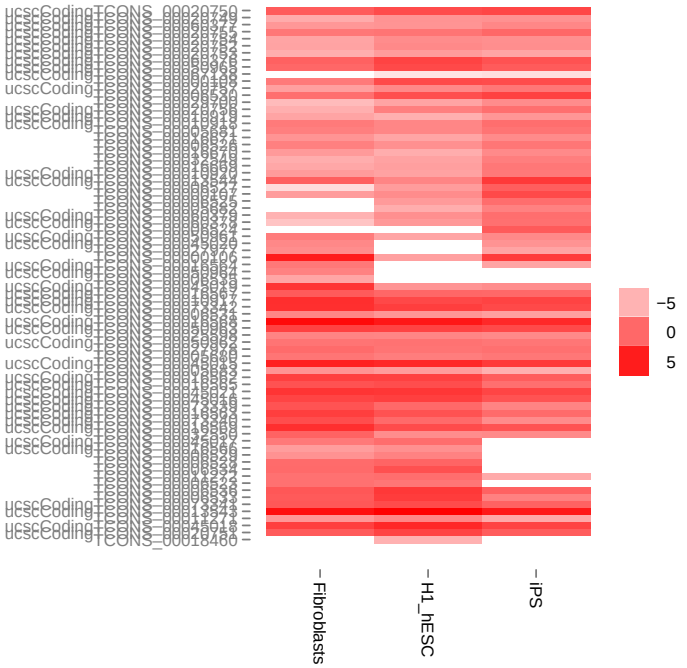

Figure 8: Heatmap of FPKM values of isoforms for a random sample of 20 genes.

7 Individual Genes

An individual CuffGene object can be created by using the `getGene` function for a given 'gene_id'.

```
> myGeneId <- "ucscCodingXLOC_009243"
> myGene <- getGene(cuff, myGeneId)
> myGene

CuffGene instance for gene ucscCodingXLOC_009243
Short name:         SPG7
Slots:
         annotation
         fpkm
         diff
         isoforms         CuffFeature instance of size 4
         TSS                 CuffFeature instance of size 3
         CDS                 CuffFeature instance of size 0

> head(fpkm(myGene))

                gene_id sample_name     fpkm  conf_hi  conf_lo
1 ucscCodingXLOC_009243 Fibroblasts 15.99820 22.11980  9.87668
2 ucscCodingXLOC_009243     H1_hESC 28.60250 34.69960 22.50540
3 ucscCodingXLOC_009243         iPS  7.03099  9.59873  4.46324

> head(fpkm(myGene@isoforms))

                isoform_id sample_name     fpkm  conf_hi
1 ucscCodingTCONS_00026844 Fibroblasts  2.59315  4.24207
2 ucscCodingTCONS_00026844     H1_hESC  1.55258  2.13436
3 ucscCodingTCONS_00026844         iPS  0.85820  1.43080
4 ucscCodingTCONS_00026845 Fibroblasts 12.63330 18.50670
5 ucscCodingTCONS_00026845     H1_hESC 25.18650 31.22720
6 ucscCodingTCONS_00026845         iPS  4.85961  7.27292
    conf_lo
1  0.944239
2  0.970801
3  0.285599
4  6.759920
5 19.145700
6  2.446310
```

7.1 Gene-level plots

```
> gl <- expressionPlot(myGene)
```

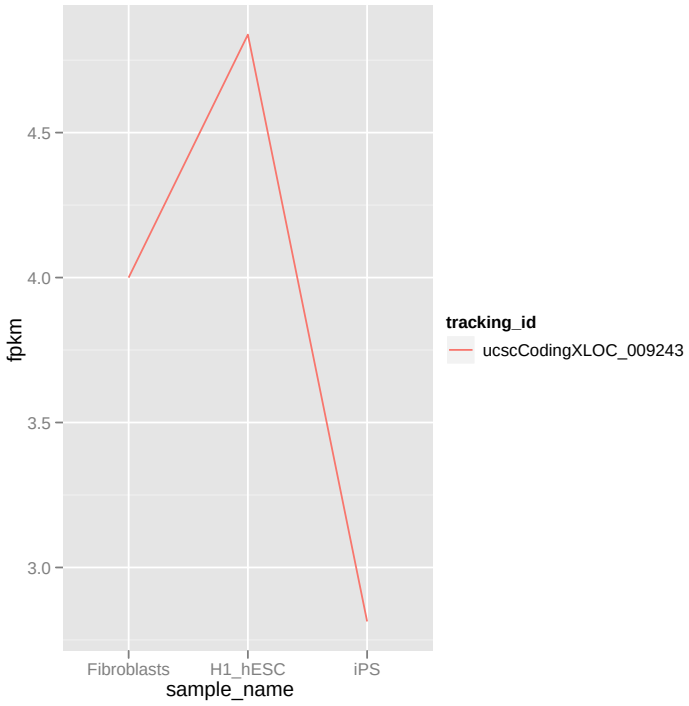

Figure 9: Line plot of FPKM expression values for a given gene

```
> gb <- expressionBarplot(myGene)
```

```
> igb <- expressionBarplot(myGene@isoforms)
```

8 Miscellaneous

- All plotting functions return ggplot objects and the resulting objects can be manipulated/faceted/altered using standard ggplot2 methods.
- There are occasional DB connectivity issues that arise. Not entirely sure why yet. If necessary, just `readCufflinks` again and this should solve connectivity issues with a new RSQLite connection object. If connectivity continues to be a problem, try `cuff<-readCufflinks(rebuild=T)`
- I am still working on fully documenting each of the methods. There are a good number of arguments that exist, but might be hard to find without looking at the source.

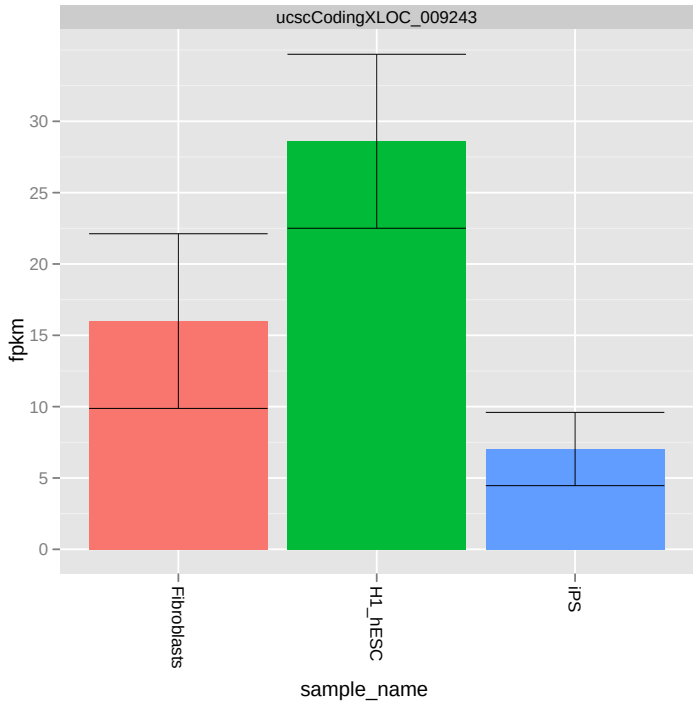

Figure 10: Bar plot of FPKM expression values for a given gene

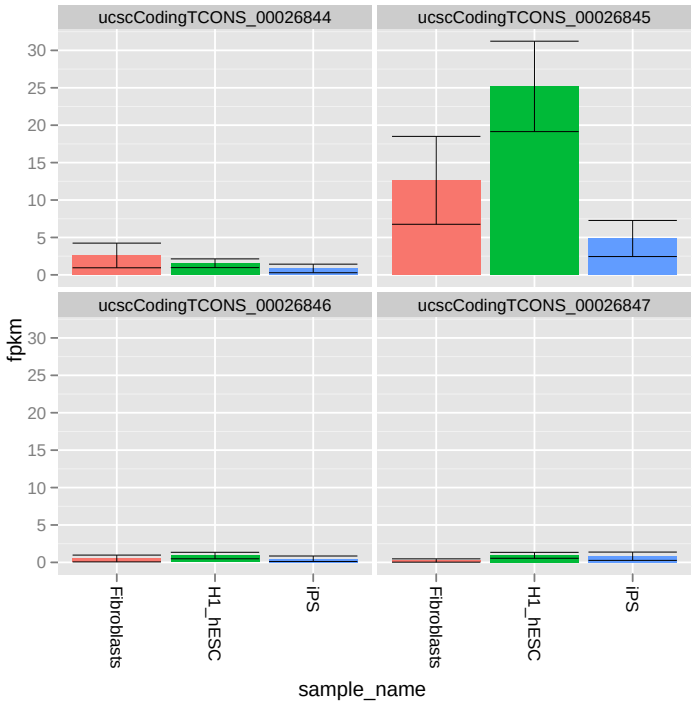

Figure 11: Bar plot of FPKM expression values for all isoforms of a given gene

9 Session info

```
> sessionInfo()

R version 2.12.2 (2011-02-25)
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

locale:
[1] C/en_US.UTF-8/C/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets
[7] methods   base

other attached packages:
[1] cummeRbund_0.1.2 ggplot2_0.8.9    proto_0.3-8
[4] reshape_0.8.4    plyr_1.4         RSQLite_0.9-4
[7] DBI_0.2-5

loaded via a namespace (and not attached):
[1] digest_0.4.2 tools_2.12.2
```