Wrangling data into track hub visualizations with **hubward** github.com/daler/hubward

Ryan Dale | National Institute of Diabetes and Digestive and Kidney Diseases | National Institutes of Health | Bethesda, MD USA

The problem:

Published data are stored in a multitude of formats. This means we need to write custom code to convert into suitable display formats for the UCSC Genome Browser. Doing so for dozens of studies requires careful organization and managment to track data provenance and to allow biologists to easily find data of interest.

The goal:

We would like to quickly and easily view any interesting published data sets in the UCSC Genome Browser, alongside our own results for genomic assays like RNA-seq, ChIP-seq, etc.

The solution:

hubward restricts the scope of custom code while providing a set of tools for managing many data sets. It provides a uniform schema for processing, handles downloading of original data, performs bulk liftover of coordinates to other assemblies, aggregates processed data into "track hubs" (configurable collections of tracks), and automates the upload of many such track hubs to the configured host.

This poster:

This poster shows a complete working example of using **hubward** to view predicted enhancers and long-range chromatin interaction data from two separate studies for a locus of interest. This visualization would be helpful, for example, when choosing distal targets for CRISPR/Cas9 deletion to investigate mechanisms of gene regulation.

1. Identify datasets

For brevity this example only uses two studies. In practice, we would use many more. Data are from human K562 cells.

[A] Distal regulatory modules (DRM) within enhancers (E) and weak enhancers (WE) predicted by ChromHMM and Segway from Yip et al. Genome Biology 2012. PMID:22950945

```
chr1    834244  834337  DRM_E                   chr1    1092417 1092540 DRM_WE
chr1    1091836 1092160 DRM_E                   chr1    4067069 4067200 DRM_WE
chr1    1094000 1094028 DRM_E                   chr1    4067619 4067800 DRM_WE
chr1    1521517 1521600 DRM_E                   chr1    5364809 5365049 DRM_WE
```

- already in BED format - just need to convert to bigBed
- study will be created in `yip-2012` directory

[B] HiC long-range interaction data from Lieberman-Aiden et al. Science 2009. PMID:19815776

```
#K562all.0.maq.ctg11.ctg11.100000bp.hm.newtracks12forBryan.OvsE.matrix.tab
HIC_bin1|hg18|chr11:1-99999          HIC_bin2|hg18|chr11:100000-199999  HIC_bin3|hg18|chr11:200000-299999  HIC_bin4|hg18|chr11:300000-399999 ...
HIC_bin1|hg18|chr11:1-99999          0.43954983653193269                0.05494372956491586                0
HIC_bin2|hg18|chr11:100000-199999    0.05494372956491586                1.4834806982952728                 0.38460610696544112
HIC_bin3|hg18|chr11:200000-299999    0                                  0.38460610696544112                8.2965031645402298
...
```

- 1345 x 1345 matrix of obs:exp values in 10-kb bins needs conversion to UCSC-compatible format
- needs liftover from hg18 to hg19
- study will be created in `lieberman-2009` directory.

2. Write conversion scripts

Must accept 2 positional args, <input> <output>, which will be automatically provided by **hubward** (in step 4).

Any language, any path -- whatever gets the job done.

For track hubs, the output must be a format supported by UCSC: BAM, VCF, bigBed, or bigWig.

[A] `yip-2012/bed2bigbed.sh` (sorts a BED file and converts to bigBed)

```
#!/bin/bash
set -e
source="$1"; target="$2"
gunzip -c $source | LC_COLLATE=C sort -k1,1 -k2,2n > ${target}.tmp.bed
chromsizes="$(dirname $source)/hg19"
[[ ! -e $chromsizes ]] && fetchChromSizes hg19 > $chromsizes
bedToBigBed -type=bed4 ${target}.tmp.bed $chromsizes $target
rm ${target}.tmp.bed
```

[B] `lieberman-2009/src/process.py` (makes a bigWig of signal for one locus)

```
#!/usr/bin/env python
import sys, pandas, os, hubward
source, target = sys.argv[1:3]

coord = os.path.basename(target).replace('.bigwig', '')
chrom, start, stop = coord.split('_')
coord = '{0}:{1}-{2}'.format(chrom, start, stop)

x = pandas.read_table(source, sep='\t', comment="#", index_col=0)
row_ind = x.index.to_series().apply(lambda y: coord in y)
row = x[row_ind].T.dropna()
row.columns = ['score']

def to_bed(f):
    """E.g., 'HIC_bin1|hg18|chr11:1-99999' -> ('chr11', 1, 99999)"""
    chrom, startend = f.split('|')[-1].split(':')
    start, end = startend.split('-')
    return chrom, int(start), int(end)

row['chr'], row['start'], row['end'] = zip(*(row.index.to_series().apply(to_bed)))
tmp = target + '.tmp'
hubward.utils.makedirs(os.path.dirname(tmp))
row[['chr', 'start', 'end', 'score']].sort(['chr', 'start'])\
    .to_csv(tmp, sep='\t', index=False, header=False)
hubward.utils.bigwig(tmp, genome='hg18', output=target)
os.unlink(tmp)
```

- extract coordinates of interest from output filename
- extract relevant row of dataframe
- convert column names into BED features,
- create bigWig of values from extracted row

`lieberman-2009/src/make_anchor.py` (makes a bigBed for a single locus)

```
#!/usr/bin/env python
import sys, pandas, os, hubward
source, target = sys.argv[1:3]
# extract anchor bin coords from output filename:
coord = os.path.basename(target).replace('.bigbed', '')
chrom, start, stop = coord.split('_')

tmp = target + '.tmp'
with open(tmp, 'w') as fout:
    fout.write('{0}\t{1}\t{2}\n'.format(chrom, start, stop))
hubward.utils.bigbed(tmp, genome='hg18', output=target)
os.unlink(tmp)
```

- extract coordinates of interest from the output filename.
- create a bigBed file containing a single feature corresponding to the anchor.

3. Customize config for each study

These YAML-format files completely define source data, target filenames, and how each track should look in the Genome Browser. Must be named `metadata.yaml` and conform to the documented schema.

[A] `yip-2012/metadata.yaml`

```
study:
  label: "yip-2012"
  short_label: "Candidate enhancers"
  long_label: "Candidate enhancers from K562 cells"
  reference: "Yip et al. 2012. Genome Biol. 13(9):R48"
  PMID: "22950945"
  description: "Distal regulatory modules (DRMS) intersecting ..."

tracks:
  -
    short_label: "K562 DRM+E"
    genome: "hg19"
    original: "raw-data/DRM_E_K562_merged.bed.gz"
    processed: "processed-data/drm_k562_enhancers.bigbed"
    script: "bed2bigbed.sh"
    source:
      url: "http://encodenets.gersteinlab.org/metatracks/DRM_E_K562_merged.bed.gz"
      fn: "DRM_E_K562_merged.bed.gz"
    trackinfo:
      tracktype: "bigBed 4"
      color: "255,102,0"
    type: "bigbed"

  -
    short_label: "K562 DRM+WE"
    genome: "hg19"
    original: "raw-data/DRM_WE_K562_merged.bed.gz"
    processed: "processed-data/drm_k562_weak-enhancers.bigbed"
    script: "bed2bigbed.sh"
    source:
      url: "http://encodenets.gersteinlab.org/metatracks/DRM_WE_K562_merged.bed.gz"
      fn: "DRM_WE_K562_merged.bed.gz"
    trackinfo:
        tracktype: "bigBed 4"
        color: "204,153,102"
    type: "bigbed"
```

- bibliographic info will be included in HTML track hub documentation
- each item in the "tracks" section represents a genome browser track to create and upload
- map input file to desired output file, plus script to do conversion
- download and extract data if missing
- configure how tracks look in the UCSC Genome Browser
- this track is processed in the same way, but has different source data and a different color

[B] `lieberman-2009/metadata.yaml`

```
study:
  label: 'lieberman-2009'
  reference: "Lieberman-Aiden et al. Science. 2009 326(5950):289-93"
  PMID: "19815776"
  short_label: "HiC K562"
  description: "Hi-C data in K562 cells. *Note: only selected regions included*"

tracks:
  -
   description: "K562 HiC contacts within beta globin locus"
   short_label: "HiC chr11_5200000-5299999"
   genome: "hg18"
   original: "raw-data/HIC_k562_chr11_chr11_100000_obsexp.txt"
   processed: 'processed-data/chr11_5200000_5299999.bigwig'
   source: &source
     fn: "GSE18199_binned_heatmaps.zip.gz"
     url: "http://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE18199&format=file&file=GSE18199%5Fbinned%5Fheatmaps%2Ezip%2Egz"
   script: "src/process.py"
   trackinfo:
     color: "75,105,131"
     viewLimits: "0:10"
   type: "bigwig"

  -
    short_label: "Anchor bin"
    genome: "hg18"
    original: "raw-data/HIC_k562_chr11_chr11_100000_obsexp.txt"
    processed: "processed-data/chr11_5200000_5299999.bigbed"
    script: "src/make_anchor.py"
    source: *source
    trackinfo:
        tracktype: "bigBed 3"
        color: "180,0,0"
    type: "bigbed"
```

- description will be converted from ReStructuredText to HTML in uploaded documentation
- any UCSC track hub params can be configured here. (here: set y-axis max to 10 rather than default 127)
- same input file, different output and different script

4. Run `hubward` on individual studies

`hubward process study <directory>` reads the metadata.yaml file and applies conversion script if target file needs updating

[A] `$ hubward process yip-2012`

```
yip-2012/                          yip-2012/
├── bed2bigbed.sh                  ├── bed2bigbed.sh
├── metatdata.yaml                 ├── metadata.yaml
└── raw-data                       ├── processed-data
    ├── DRM_E_K562_merged.bed.gz   │   ├── drm_k562_enhancers.bigbed
    ├── DRM_WE_K562_merged.bed.gz  │   └── drm_k562_weak-enhancers.bigbed
    └── hg19                       └── raw-data
                                       ├── DRM_E_K562_merged.bed.gz
                                       ├── DRM_WE_K562_merged.bed.gz
                                       └── hg19
```

[B] `$ hubward process lieberman-2009`

```
lieberman-2009/                    lieberman-2009/
├── metadata.yaml                  ├── metadata.yaml
├── raw-data                       ├── processed-data
│   └── HIC_k562_chr11_chr11_100000_obsexp.txt  │   ├── chr11_5200000_5299999.bigbed
└── src                            │   └── chr11_5200000_5299999.bigwig
    └── process.py                 ├── raw-data
                                   │   └── HIC_k562_chr11_chr11_100000_obsexp.txt
                                   └── src
                                       └── process.py
```

4a: Liftover (optional)

Use CrossMap to handle conversion of bigBed, bigWig, BAM, VCF.

```
$ hubward liftover --from_assembly hg18 --to_assembly hg19 \
      lieberman-2009 lieberman-2009-hg19
```

5. Group studies

Mix-and-match any configured studies into a single aggregated hub.
Shortcut: use `hubward process myhub.yaml` to process all listed studies instead of individually as above.

[A+B] `myhub.yaml`

```
name: "hubward-example"
genome: "hg19"
short_label: "Hubward example"
long_label: "Hubward example tracks"
hub_url: "http://example.com/hubs/example.hub.txt"
email: "dalerr@niddk.nih.gov"

server:
    hub_remote: "/www/hubs/example.hub.txt"
    host: "example.com"
    user: "username"

studies:
    - "yip-2012"
    - "lieberman-2009-hg19"
```

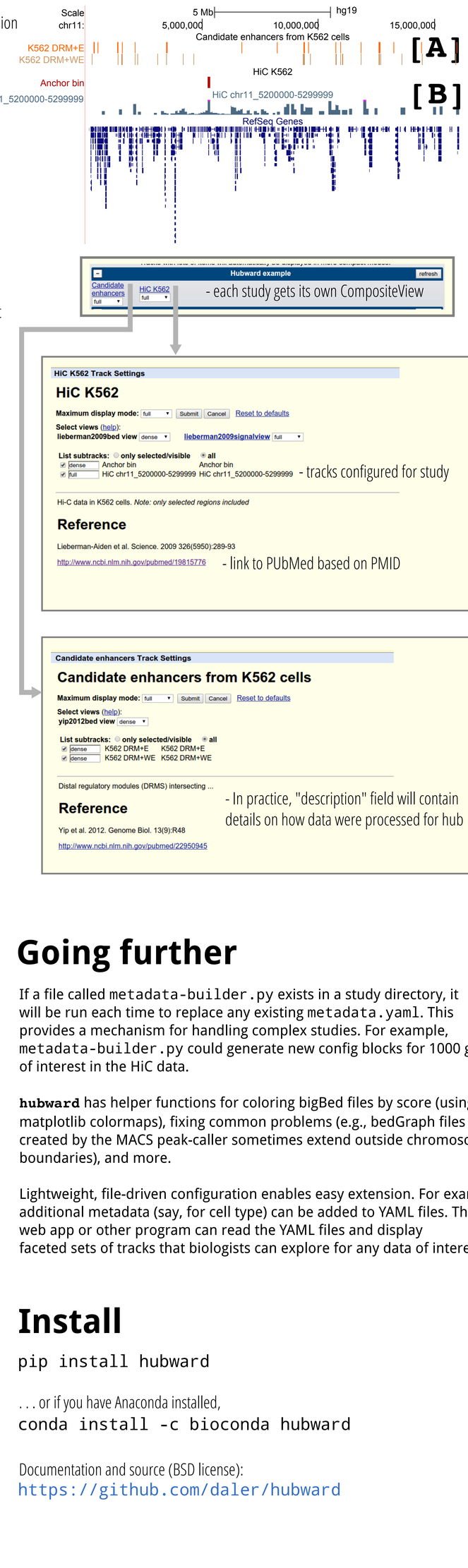
- studies can be re-used across different hubs, so these tracks could be included in another hub perhaps called "enhancers"
- processed files will be rsynced to this server
- specify an arbitrary number of studies to include in the track hub

6: Upload and view

All tracks for all studies are uploaded via rsync to the configured server.

```
$ hubward upload myhub.yaml
```

- server details can also be configured from command line



Going further

If a file called `metadata-builder.py` exists in a study directory, it will be run each time to replace any existing `metadata.yaml`. This provides a mechanism for handling complex studies. For example, `metadata-builder.py` could generate new config blocks for 1000 genes of interest in the HiC data.

hubward has helper functions for coloring bigBed files by score (using matplotlib colormaps), fixing common problems (e.g., bedGraph files created by the MACS peak-caller sometimes extend outside chromosome boundaries), and more.

Lightweight, file-driven configuration enables easy extension. For example, additional metadata (say, for cell type) can be added to YAML files. Then a web app or other program can read the YAML files and display faceted sets of tracks that biologists can explore for any data of interest.

Install

```
pip install hubward
```

. . . or if you have Anaconda installed,

```
conda install -c bioconda hubward
```

Documentation and source (BSD license):
https://github.com/daler/hubward