- Getting Started with TBmon
    - TBmon is a small analysis framework used to study data taken with silicon pixel sensors in a test beam (TB) environment. From raw TB data, output ROOT files are produced during track reconstruction and then taken as input to TBmon. For a given set of runs and devices under test (DUTs), analyses loop over the data and produce a set of plots for further study.
    - Setting up TBmon
        - Get the data
            - `$ mkdir ~/tbAnalysis/data/tb2010-11-eudet`
            - `$ cd ~/tbAnalysis/data/tb2010-11-eudet`
            - from the CERN Mac (`mpnoatlas01`)
                - `$ wget -r -nd -A "tbtrack*.root" http://mpnoatlas01.cern.ch/~atlas3dsi/tb2010-11-eudet/tbtrack`
                - or you can specify a particular run range:
                    - `$ for i in $(jot - 21185 21221); do wget -r -nd http://mpnoatlas01.cern.ch/~atlas3dsi/tb2010-11-eudet/tbtrack/tbtrack0$i.root;done`
            - the more recent DESY testbeams have data saved at CERN on Castor
                - to see the full list of available tbtrack files:
                    - `$ rfdir /castor/cern.ch/atlas/atlascerngroupdisk/det-ibl/TestBeam/Feb2011_DESY/Output`
                - to actually download the files, use the xrootd cp command:
                    - `$ xrdcp -R root://castoratlas///castor/cern.ch/atlas/atlascerngroupdisk/det-ibl/TestBeam/Feb2011_DESY/Output/ .`
            - you may also be able to try the Dortmund server, e.g.:
                - `$ scp -r <USERNAME>@login.e4.physik.uni-dortmund.de:/hdfs/group/atlas-pixel/testbeam/desy_2011_feb_ibl/results .`
        - Get the code from the ATLAS IBL testbeam software CERN SVN repository:
            - `$ cd ~/tbAnalysis`
            - `# replace <USERNAME> with your CERN AFS login`
            - `$ svn checkout svn+ssh://<USERNAME>@svn.cern.ch/reps/atlasibltbsw/tbmon tbmon`
        - In order to run an analysis, TBmon needs to know where to find the data on your machine and where to save any output plots. It looks for these machine-specific paths in `siteconfig.h`. This file is not under

version control, but a template called `siteconfig.h.example` is. Rename and modify this file to suit your needs:

- `$ cd ~/tbAnalysis/tbmon/trunk`
- `$ cp siteconfig.h.example siteconfig.h`
- Change all paths and default configuration preferences to match your own, and note that all paths must actually exist. For example:

```
void siteConfig(TbConfig &config){
  trySet(config.outPath, (char*) "/Users/bdewilde/Desktop/
tbAnalysis/output");
  trySet(config.plotExtension, (char*) ".png");
  trySet(config.tbslot, (char*) "eudetfeb2011");
}
```

- If you have data from different test beams in more than one place, be sure to specify each path in the individual configurations' methods and not in `siteConfig()`. For example:

```
void siteEudetFeb2011(TbConfig &config){
  trySet(config.dataPath, (char*) "/Users/bdewilde/Desktop/
tbAnalysis/data/tb2011-02-eudet/tbtrack");
}
```

- Make sure ROOT is installed on your machine and can be found in your $PATH. If this is so, go ahead and compile the framework:
  - `$ make`

- Running analyses with TBmon
  - Analyses are executed from the command line. For an up-to-date description of the format and arguments:
    - $ cd ~/tbAnalysis/tbmon/trunk
    - $ ./tbmon

      Needs ONE of the following arguments to determine which files to loop over:

-l <runList>	Loop over all runs in <runList>
-r <firstrun> <lastrun>	Loop over all runs from <firstrun> to <lastrun>
-s <run>	Loop over a single run

      Optional arguments:

-a <analysisname> Only run analysis named <analysisname>. If not supplied, run over all available analyses.
-i <iden> Only run over iden <iden>. If not supplied, loop over all available idens.
-c <configname> Configures modules for configuration <configname>. If not supplied, default to that set in siteconfig.h
-v <verbosity-level> Should be one of (in order of increasing verbosity) error, info, debug, debug2 or debug3. Default is info.
-e <extension> Plot extension. Should be a file format recognized by ROOT.
-d <data-path> Path to input tbtrackX.root files.
-o <out-path> Path to output directory.
-b <base-name> Base name for all output. Defaults to name based on run list or range of runs.
-f Redirect output(stdout) to log file named after the base name.
-P:<key> <value> Pass an extra parameter, which can be used by analyses or builders.

Simple Example:
./tbmon -l runlists/may2009-bon-a15
Applies all available analyses to all the runs in runlist for all idens.

Complete Example:
./tbmon -l runlists/may2009-bon-a15 -i 160 164 -a sumtot residuals -c may2009 -b test1 -e png -f -o out -P:param1 1.3 -P:param2 urk
Applies analysis named "sumtot" and analysis named "residuals" to all runs in runlist for idens 160 and 164. Png plots are made and saved to relative path out. A log file(out/test1.log) is made, no output to screen. Extra-parameters "param1" and "param2" are stored internally with values "1.3" and "urk", and may be accessed from any builder and/or analysis.

- List of analyses and what they do
  - “angledist” : intended for BAT data, not covered here
  - “batcorrelation” : intended for BAT data, not covered here
  - “batunbiased” : intended for BAT data, not covered here
  - “beamprofile” : shows where and how the beam hit the sensors; track maps, track chi2, beam angle distributions

- “checkalign” : plots average residual vs hit position (x vs y, and all combinations) to assess whether the sensors were well-aligned during track reconstruction; also quickly spot runs that don’t appear to be well aligned by checking their mean x and y residual values; prints out suspicious runs
- “checkdutsync” : intended for BAT data, not covered here
- “checktrack” : look at chi2 and bad regions in sensors for tracks; of limited use, possibly delete-worthy
- “clusterchecker” : basic cluster analysis; plot track-matched and un-matched cluster size (number of cells), multiplicity, number of pixel hits per event
- “correlations” : show correlation bands (diagonal lines) to visualize orientation of the sensors
- “edgeefficiency” : assess the efficiency of long pixels on the left edge of the sensor; this is only for FE-I3 sensors, as the FE-I4 sensors used in IBL do not have active edges
- “edgeefficiencyshift” : TBD
- “efficiency” : assess sensor efficiency, fairly self-explanatory
- “efficiency2” and “efficiencysimp” : TBD
- “etawidth” : TBD
- “getetacorr” : plot charge-weighted and eta corrections
- “hotpixelfinder” : evaluate dead and noisy (“hot”) pixels; plot raw hit map, lv1, noise occupancy, and masked pixels, where 1=dead and 2=noisy
- “lvl1cut” : plot matched and un-matched hit lvl1 distributions; seems like this could get merged into another analysis somewhere…
- “maxcellres” : basic data quality checks; plot cut and un-cut X and Y residual, lvl1, track chi2, sensor and max ToT, etc. distributions; get approximate overall sensor efficiency numbers; this is a grab-bag, courtesy of Håvard
- “qEfficiency” : TBD
- “qshare1d” : plots ToT against spatial position of various hit pixels in a track-matched cluster
- “qshare2d” : much like qshare1d, but with a second dimension! aspect ratio pixel plots are useful for talks
- “readout” : apparent waste of space; deletion likely
- “residuals” : compare X and Y residuals for four different methods (maxToT, geometric mean, charge-weighted, eta-corrected); cluster-size dependent plots; values printed out for each case
- “simDutEdep” : intended for sim data, not covered here
- “simResiduals” : intended for sim data, not covered here
- “sumtot” : plot the cluster ToT at various positions in a pixel with respect to the electrodes