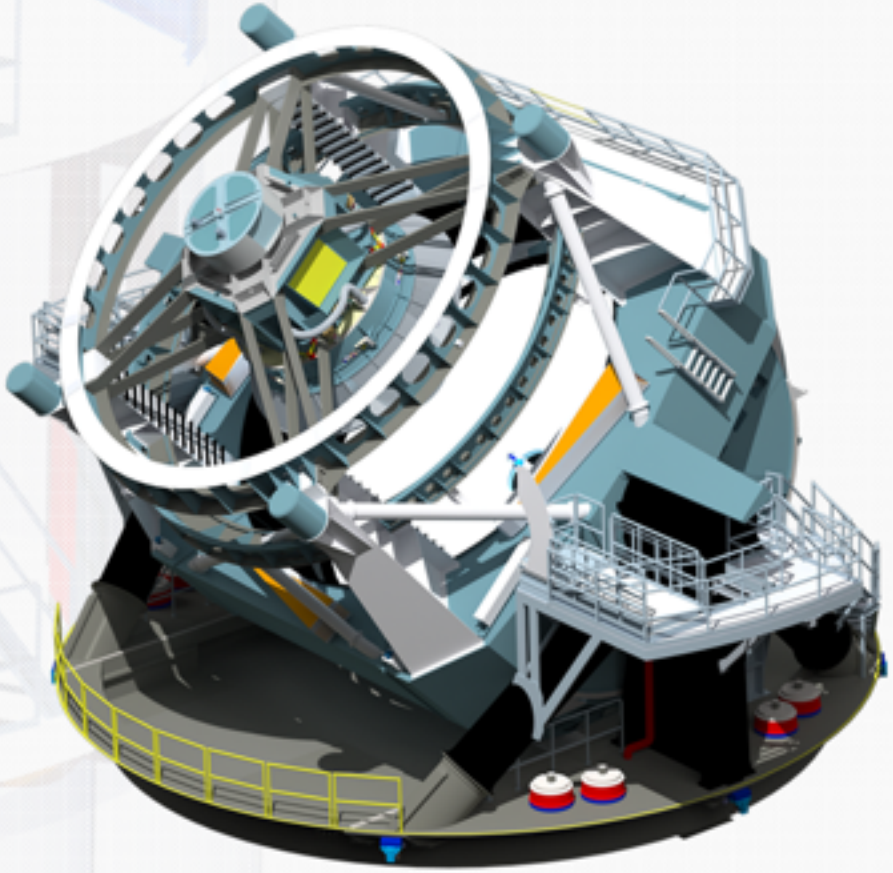
Using Tasks

John Swinbank

Technical Manager, Data Release Production

5 October 2015



DM Boot Camp

5–7 October 2015 | Princeton

What is a Task?

- A coherent "unit of work" which is carried out as part of some data processing pipeline.
- "Unit of work" varies from the trivial ("add two numbers") to the complex ("do everything needed to detect & measure sources, including ISR, calibration, source finding, …").
- Tasks are combined hierarchically.
  - We don't write a "do everything necessary…" task; we compose it from lower level tasks.
- A task may be exposed to the end user through a command line interface (`CmdLineTask`) or accessible only from Python (`Task`).

Getting set up

- We will use the example scripts provided in the `pipe_tasks` package:

```
$ setup pipe_tasks -t v11_0
$ cd ${PIPE_TASKS_DIR}/examples
```

- We'll grab some test data from the `afwdata` repository:

```
$ curl -k -O https://dev.lsstcorp.org/cgit/LSST/DMS/testdata/afwdata.git/plain/data/
small.fits
```

- And now demonstrate that we can run a simple task:

```
$ ./exampleStatsTask.py small.fits
running ExampleSimpleStatsTask
exampleSimpleStats: simple mean=62.12; meanErr=0.22; stdDev=55.41; stdDevErr=inf
result  = Struct(meanErr=0.2164367369890884; stdDevErr=inf; stdDev=55.407804669206634;
mean=62.118194580078125)
running ExampleSigmaClippedStatsTask
exampleSigmaClippedStats: clipped mean=59.08; meanErr=0.03; stdDev=6.87; stdDevErr=nan
result  = Struct(meanErr=0.027137912763189356; stdDevErr=nan; stdDev=6.87371775922687;
mean=59.07955732211054)
```

Running a Task in Python

```
# Edited highlights of ${PIPE_TASKS_DIR}/example/exampleStatsTask.py
import sys
from lsst.afw.image import MaskedImageF
from lsst.pipe.tasks.exampleStatsTasks import ExampleSimpleStatsTask

# Load a MaskedImageF -- an image containing floats
# together with a mask and a per-pixel variance.
maskedImage = MaskedImageF(sys.argv[1])

# We initialize the Task once but can call it many times.
task = ExampleSimpleStatsTask()

# Simply call the .run() method with the MaskedImageF.
result = task.run(maskedImage)

# And print the result.
print(result)
```

Running a Task in Python

```
# Edited highlights of ${PIPE_TASKS_DIR}/example/exampleStatsTask.py
import sys
from lsst.afw.image import MaskedImageF
from lsst.pipe.tasks.exampleStatsTasks import ExampleSimpleStatsTask

# Load a MaskedImageF -- an image containing floats
# together with a mask and a per-pixel variance.
maskedImage = MaskedImageF(sys.argv[1])

# We initialize the Task once but can call it many times.
task = ExampleSimpleStatsTask()

# Simply call the .run() method with the MaskedImageF.
result = task.run(maskedImage)

# And print the result.
print(result)
```

There's nothing special about run(); Tasks can provide other methods as required.

Running a Task in Python

```
# Edited highlights of ${PIPE_TASKS_DIR}/example/exampleStatsTask.py
import sys
from lsst.afw.image import MaskedImageF
from lsst.pipe.tasks.exampleStatsTasks import ExampleSimpleStatsTask

# Load a MaskedImageF -- an image containing floats
# together with a mask and a per-pixel variance.
maskedImage = MaskedImageF(sys.argv[1])

# We initialize the Task once but can call it many times.
task = ExampleSimpleStatsTask()

# Simply call the .run() method with the MaskedImageF.
result = task.run(maskedImage)

# And print the result.
print(result)
```

Note the return type is a `Struct`; access e.g. `result.mean`

Adding configuration

```
# Edited highlights of ${PIPE_TASKS_DIR}/example/exampleStatsTask.py
import sys
from lsst.afw.image import MaskedImageF
from lsst.pipe.tasks.exampleStatsTasks import ExampleSigmaClippedStatsTask

maskedImage = MaskedImageF(sys.argv[1])

config1 = ExampleSigmaClippedStatsTask.ConfigClass(numSigmaClip=1)
config2 = ExampleSigmaClippedStatsTask.ConfigClass()
config2.numSigmaClip = 3

#config3 = ExampleSigmaClippedStatsTask.ConfigClass(numSigmaClip="ten")

task1 = ExampleSigmaClippedStatsTask(config=config1)
task2 = ExampleSigmaClippedStatsTask(config=config2)

print(task1.run(maskedImage).mean)
print(task2.run(maskedImage).mean)
```

Adding configuration

Different task: this one has some configurable options.

```
# Edited highlights of ${PIPE_TASKS_DIR}/example/exampleS tsTask.py
import sys
from lsst.afw.image import MaskedImageF
from lsst.pipe.tasks.exampleStatsTasks import ExampleSigmaClippedStatsTask

maskedImage = MaskedImageF(sys.argv[1])

config1 = ExampleSigmaClippedStatsTask.ConfigClass(numSigmaClip=1)
config2 = ExampleSigmaClippedStatsTask.ConfigClass()
config2.numSigmaClip = 3

#config3 = ExampleSigmaClippedStatsTask.ConfigClass(numSigmaClip="ten")

task1 = ExampleSigmaClippedStatsTask(config=config1)
task2 = ExampleSigmaClippedStatsTask(config=config2)

print(task1.run(maskedImage).mean)
print(task2.run(maskedImage).mean)
```

Adding configuration

```
# Edited highlights of ${PIPE_TASKS_DIR}/example/exampleStatsTask.py
import sys
from lsst.afw.image import MaskedImageF
from lsst.pipe.tasks.exampleStatsTasks import Ex

maskedImage = MaskedImageF(sys.argv[1])

config1 = ExampleSigmaClippedStatsTask.ConfigClass(numSigmaClip=1)
config2 = ExampleSigmaClippedStatsTask.ConfigClass()
config2.numSigmaClip = 3

#config3 = ExampleSigmaClippedStatsTask.ConfigClass(numSigmaClip="ten")

task1 = ExampleSigmaClippedStatsTask(config=config1)
task2 = ExampleSigmaClippedStatsTask(config=config2)

print(task1.run(maskedImage).mean)
print(task2.run(maskedImage).mean)
```

Each task has its own accompanying ConfigClass.

Adding configuration

```
# Edited highlights of ${PIPE_TASKS_DIR}/example/exampleStatsTask.py
import sys
from lsst.afw.image import MaskedImageF
from lsst.pipe.tasks.exampleStatsTasks import ExampleSigmaClippedStatsTask

maskedImage = MaskedImageF(sy

config1 = ExampleSigmaC                                            ip=1)
config2 = ExampleSigmaCl
config2.numSigmaClip = 3

#config3 = ExampleSigmaClippedStatsTask.ConfigClass(numSigmaClip="ten")

task1 = ExampleSigmaClippedStatsTask(config=config1)
task2 = ExampleSigmaClippedStatsTask(config=config2)

print(task1.run(maskedImage).mean)
print(task2.run(maskedImage).mean)
```

Sanity/type checking: this raises an exception.

Adding configuration

```
# Edited highlights of ${PIPE_TASKS_DIR}/example/exampleStatsTask.py
import sys
from lsst.afw.image import MaskedImageF
from lsst.pipe.tasks.exampleStatsTasks import ExampleSigmaClippedStatsTask

maskedImage = MaskedImageF(sys.argv[1])

config1 = ExampleSigmaClippedStatsTask.ConfigClass(numSigmaClip=1)
config2 = ExampleSigmaClippedStatsTask.ConfigClass()
config2.numSigmaClip = 3

#config3 = ExampleSigmaClippedStatsTask.ConfigClass(numSigmaClip="ten")

task1 = ExampleSigmaClippedStatsTask(config=config1)
task2 = ExampleSigmaClippedStatsTask(config=config2)

print(task1.run(maskedImage).mean)
print(task2.run(maskedImage).mean)
```

These tasks do the same thing, but with different configs.

Command line tasks

- You *could* wrap all the tasks in a home-brewed command line interface like the one we just saw.
- …but we *shouldn't*:
  - Using CmdLineTask provides us with a standard interface across all our tasks.
  - Includes interaction with the Butler (talk by K-T).
    - Read and write data, store task configuration and metadata.
  - Set and show configuration, parallelization, data repositories, etc.
  - (Ultimately) integrate with the LSST process execution middleware.

Processing model

- All that integration with the Butler etc adds up to complexity: we have a lot more to think about.
- Rather than specify a filename or similar on the command line, we specify the path to an input repository and the ID of data within that repository.

```
$ myTask.py /path/to/repository --id data_id [options]
```

  - More about what "`data_id`" looks like in a moment.
- The middleware will iterate over everything in the repository that matches "`data_id`" and call the task's `run()` method on it.
  - (The middleware *will* call `run()`: here, the name matters if you're writing a task).
- You cannot easily just say "run on this file" (see `processFile`).

Getting started

```
$ setup obs_test -t v11_0
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --help
: Config override file does not exist: '/Users/jds/Projects/Astronomy/LSST/stack/
DarwinX86/obs_test/11.0+1/config/exampleTask.py'
: Config override file does not exist: '/Users/jds/Projects/Astronomy/LSST/stack/
DarwinX86/obs_test/11.0+1/config/test/exampleTask.py'
usage: exampleCmdLineTask.py input [options]

positional arguments:
  input                 path to input data repository, relative to
                        $PIPE_INPUT_ROOT

optional arguments:
  -h, --help            show this help message and exit
  --calib CALIB         path to input calibration repository, relative to
                        $PIPE_CALIB_ROOT
  --output OUTPUT       path to output data repository (need not exist),
                        relative to $PIPE_OUTPUT_ROOT
  -c [NAME=VALUE [NAME=VALUE ...]], --config [NAME=VALUE [NAME=VALUE ...]]
                        config override(s), e.g. -c foo=newfoo bar.baz=3
  -C [CONFIGFILE [CONFIGFILE ...]], --configfile [CONFIGFILE [CONFIGFILE ...]]
                        config override file(s)
  -L [LEVEL|COMPONENT=LEVEL [LEVEL|COMPONENT=LEVEL ...]], --loglevel [LEVEL|COMPONENT=LEV
                        logging level; supported levels are
```

Getting started

obs_test

contains some convenient data for experimenting

```
$ setup obs_test -t v11_0
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --help
: Config override file does not exist: '/Users/jds/Projects/Astronomy/LSST/stack/
DarwinX86/obs_test/11.0+1/config/exampleTask.py'
: Config override file does not exist: '/Users/jds/Projects/Astronomy/LSST/stack/
DarwinX86/obs_test/11.0+1/config/test/exampleTask.py'
usage: exampleCmdLineTask.py input [options]

positional arguments:
  input                 path to input data repository, relative to
                        $PIPE_INPUT_ROOT

optional arguments:
  -h, --help            show this help message and exit
  --calib CALIB         path to input calibration repository, relative to
                        $PIPE_CALIB_ROOT
  --output OUTPUT       path to output data repository (need not exist),
                        relative to $PIPE_OUTPUT_ROOT
  -c [NAME=VALUE [NAME=VALUE ...]], --config [NAME=VALUE [NAME=VALUE ...]]
                        config override(s), e.g. -c foo=newfoo bar.baz=3
  -C [CONFIGFILE [CONFIGFILE ...]], --configfile [CONFIGFILE [CONFIGFILE ...]]
                        config override file(s)
  -L [LEVEL|COMPONENT=LEVEL [LEVEL|COMPONENT=LEVEL ...]], --loglevel [LEVEL|COMPONENT=LEV
                        logging level; supported levels are
```

Getting started

```
$ setup obs_test -t v11_0
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --help
: Config override file does not exist: '/Users/jds/Projects/Astronomy/LSST/stack/
DarwinX86/obs_test/11.0+1/config/exampleTask.py'
: Config override file does not exist: '/Users/jds/Proj
DarwinX86/obs_test/11.0+1/config/test/exampleTask.py'
usage: exampleCmdLineTask.py input [options]

positional arguments:
  input                 path to input data repository, relative to
                        $PIPE_INPUT_ROOT

optional arguments:
  -h, --help            show this help message and exit
  --calib CALIB         path to input calibration repository, relative to
                        $PIPE_CALIB_ROOT
  --output OUTPUT       path to output data repository (need not exist),
                        relative to $PIPE_OUTPUT_ROOT
  -c [NAME=VALUE [NAME=VALUE ...]], --config [NAME=VALUE [NAME=VALUE ...]]
                        config override(s), e.g. -c foo=newfoo bar.baz=3
  -C [CONFIGFILE [CONFIGFILE ...]], --configfile [CONFIGFILE [CONFIGFILE ...]]
                        config override file(s)
  -L [LEVEL|COMPONENT=LEVEL [LEVEL|COMPONENT=LEVEL ...]], --loglevel [LEVEL|COMPONENT=LEV
                        logging level; supported levels are
```

Most important option!

Getting started

```
$ setup obs_test -t v11_0
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --help
: Config override file does not exist: '/Users/jds/Projects/Astronomy/LSST/stack/
DarwinX86/obs_test/11.0+1/config/exampleTask.py'
: Config override file does not exist: '/Users/jds/Projects/Astronomy/LSST/stack/
DarwinX86/obs_test/11.0+1/config/test/exampleTask.py'
usage: exampleCmdLineTask.py input [options]

positional arguments:
  input                 path to input data repositor
                        $PIPE_INPUT_ROOT

optional arguments:
  -h, --help            show this help message and exit
  --calib CALIB         path to input calibration repository, relative to
                        $PIPE_CALIB_ROOT
  --output OUTPUT       path to output data repository (need not exist),
                        relative to $PIPE_OUTPUT_ROOT
  -c [NAME=VALUE [NAME=VALUE ...]], --config [NAME=VALUE [NAME=VALUE ...]]
                        config override(s), e.g. -c foo=newfoo bar.baz=3
  -C [CONFIGFILE [CONFIGFILE ...]], --configfile [CONFIGFILE [CONFIGFILE ...]]
                        config override file(s)
  -L [LEVEL|COMPONENT=LEVEL [LEVEL|COMPONENT=LEVEL ...]], --loglevel [LEVEL|COMPONENT=LEV
                        logging level; supported levels are
```

Trying to load camera-specific configuration

Getting started

```
$ setup obs_test -t v11_0
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --help
: Config override file does not exist: '/Users/jds/Projects/Astronomy/LSST/stack/
DarwinX86/obs_test/11.0+1/config/exampleTask.py'
: Config override file does not exist: '/Users/jds/Projects/Astronomy/LSST/stack/
DarwinX86/obs_test/11.0+1/config/test/exampleTask.py'
usage: exampleCmdLineTask.py input [options]

positional arguments:
  input                                           y, relative to


optional arguments:
  -h, --help            show this help message and exit
  --calib CALIB         path to input calibration repository, relative to
                        $PIPE_CALIB_ROOT
  --output OUTPUT       path to output data repository (need not exist),
                        relative to $PIPE_OUTPUT_ROOT
  -c [NAME=VALUE [NAME=VALUE ...]], --config [NAME=VALUE [NAME=VALUE ...]]
                        config override(s), e.g. -c foo=newfoo bar.baz=3
  -C [CONFIGFILE [CONFIGFILE ...]], --configfile [CONFIGFILE [CONFIGFILE ...]]
                        config override file(s)
  -L [LEVEL|COMPONENT=LEVEL [LEVEL|COMPONENT=LEVEL ...]], --loglevel [LEVEL|COMPONENT=LEV
                        logging level; supported levels are
```

We'll cover some of these later

Processing some data

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: clipped mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
exampleTask: Processing data ID {'filter': 'g', 'visit': 2}
exampleTask.stats: clipped mean=1228.79; meanErr=0.02; stdDev=34.19; stdDevErr=nan
exampleTask: Processing data ID {'filter': 'r', 'visit': 3}
exampleTask.stats: clipped mean=1433.76; meanErr=0.03; stdDev=37.36; stdDevErr=0.93
```

Processing some data

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: clipped mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
exampleTask: Processing data ID {'filter': 'g', 'visit': 2}
exampleTask.stats: clipped mean=1228.79; meanErr=0.02; stdDev=34.19; stdDevErr=nan
exampleTask: Processing data ID {'filter': 'r', 'visit': 3}
exampleTask.stats: clipped mean=1433.76; meanErr=0.03; stdDev=37.36; stdDevErr=0.93

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'g', 'visit': 2}
id dataRef.dataId = {'filter': 'r', 'visit': 3}
```

Processing some data

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: clipped mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
exampleTask: Processing data ID {'filter': 'g', 'visit': 2}
exampleTask.stats: clipped mean=1228.79; meanErr=0.02; stdDev=34.19; stdDevErr=nan
exampleTask: Processing data ID {'filter': 'r', 'visit': 3}
exampleTask.stats: clipped mean=1433.76; meanErr=0.03; stdDev=37.36; stdDevErr=0.93

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'g', 'visit': 2}
id dataRef.dataId = {'filter': 'r', 'visit': 3}

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id filter=g
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: clipped mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
exampleTask: Processing data ID {'filter': 'g', 'visit': 2}
exampleTask.stats: clipped mean=1228.79; meanErr=0.02; stdDev=34.19; stdDevErr=nan
```

Processing some data

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: clipped mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
exampleTask: Processing data ID {'filter': 'g', 'visit': 2}
exampleTask.stats: clipped mean=1228.79; meanErr=0.02; stdDev=34.19; stdDevErr=nan
exampleTask: Processing data ID {'filter': 'r', 'visit': 3}
exampleTask.stats: clipped mean=1433.76; meanErr=0.03; stdDev=37.36; stdDevErr=0.93

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'g', 'visit': 2}
id dataRef.dataId = {'filter': 'r', 'visit': 3}

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id filter=g
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: clipped mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
exampleTask: Processing data ID {'filter': 'g', 'visit': 2}
exampleTask.stats: clipped mean=1228.79; meanErr=0.02; stdDev=34.19; stdDevErr=nan

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id filter=g visit=1
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: clipped mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
```

“Advanced” data IDs

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1..3 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'g', 'visit': 2}
id dataRef.dataId = {'filter': 'r', 'visit': 3}
```

“Advanced” data IDs

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1..3 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'g', 'visit': 2}
id dataRef.dataId = {'filter': 'r', 'visit': 3}
```

Range is **in**clusive.

“Advanced” data IDs

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1..3 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'g', 'visit': 2}
id dataRef.dataId = {'filter': 'r', 'visit': 3}

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1..3:2 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'r', 'visit': 3}
```

“Advanced” data IDs

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1..3 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'g', 'visit': 2}
id dataRef.dataId = {'filter': 'r', 'visit': 3}

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1..3:2 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'r', 'visit': 3}

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1^3 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'r', 'visit': 3}
```

“Advanced” data IDs

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1..3 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'g', 'visit': 2}
id dataRef.dataId = {'filter': 'r', 'visit': 3}

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1..3:2 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'r', 'visit': 3}

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1^3 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'r', 'visit': 3}
```

– Also different data types depending on task & camera:

```
$ makeDiscreteSkyMap.py ${PATH} --id visit=12345 ccd=30..60:2 …

$ makeCoaddTempExp.py ${PATH} --id filter='HSC-I' tract=0 patch=7,7 …
```

“Advanced” data IDs

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1..3 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'g', 'visit': 2}
id dataRef.dataId = {'filter': 'r', 'visit': 3}

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1..3:2 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'r', 'visit': 3}

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1^3 --show data
id dataRef.dataId = {'filter': 'g', 'visit': 1}
id dataRef.dataId = {'filter': 'r', 'visit': 3}
```

– Also different data types depending on task & camera:

```
$ makeDiscreteSkyMap.py ${PATH} --id visit=12345 ccd=30..60:2 …

$ makeCoaddTempExp.py ${PATH} --id filter='HSC-I' tract=0 patch=7,7 …
```

– One last gotcha:

```
$ assembleCoadd.py ${PATH} --id tract=0 patch=7,7 filter='HSC-I' \
                           --selectID visit=904028^904006 ccd=18^19^25 …
```

Configuration

Another use for --show;
cf --show data.

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 --show config
import lsst.pipe.tasks.exampleCmdLineTask
assert type(config)==lsst.pipe.tasks.exampleCmdLineTask.ExampleCmdLineConfig, 'config is
of type %s.%s instead of lsst.pipe.tasks.exampleCmdLineTask.ExampleCmdLineConfig' %
(type(config).__module__, type(config).__name__)
config.stats.badMaskPlanes=['EDGE']
config.stats.numSigmaClip=3.0
config.stats.numIter=2
config.doFail=False

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 \
                          --config stats.numSigmaClip=10
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: clipped mean=1183.30; meanErr=0.03; stdDev=37.83; stdDevErr=nan

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 \
                          --config stats.badMaskPlanes=['EDGE', 'NO_DATA']
usage: exampleCmdLineTask.py input [options]
exampleCmdLineTask.py: error: --config value NO_DATA] must be in form name=value
```

Configuration

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 --show config
import lsst.pipe.tasks.exampleCmdLineTask
assert type(config)==lsst.pipe.tasks.exampleCmdLineTask.ExampleCmdLineConfig, 'config is
of type %s.%s instead of lsst.pipe.tasks.exampleCmdLineTask.ExampleCmdLineConfig' %
(type(config).__module__, type(config).__name__)
config.stats.badMaskPlanes=['EDGE']
config.stats.numSigmaClip=3.0
config.stats.numIter=2
config.doFail=False

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 \
                          --config stats.numSigmaClip=10
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: clipped mean=1183.30; meanErr=0.03; stdDev=37.83; stdDevErr=nan

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 \
                          --config stats.badMaskPlanes=['EDGE', 'NO_DATA']
usage: exampleCmdLineTask.py input [options]
exampleCmdLineTask.py: error: --config value NO_DATA] must be in form name=value
```

Python code!

Configuration

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 --show config
import lsst.pipe.tasks.exampleCmdLineTask
assert type(config)==lsst.pipe.tasks.exampleCmdLineTask.ExampleCmdLineConfig, 'config is
of type %s.%s instead of lsst.pipe.tasks.exampleCmdLineTask.ExampleCmdLineConfig' %
(type(config).__module__, type(config).__name__)
config.stats.badMaskPlanes=['EDGE']
config.stats.numSigmaClip=3.0
config.stats.numIter=2
config.doFail=False

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 \
                          --config stats.numSigmaClip=10
exampleTask: Pr                         {'filter': 'g', 'visit': 1}
exampleT                             1183.30; meanErr=0.03; stdDev=37.83; stdDevErr=nan

$ ./exam                             EST_DIR}/data/input --id visit=1 \
                                 config stats.badMaskPlanes=['EDGE', 'NO_DATA']
usage: exampleCmdLineTask.py input [options]
exampleCmdLineTask.py: error: --config value NO_DATA] must be in form name=value
```

Dropped the config. prefix.

Configuration

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 --show config
import lsst.pipe.tasks.exampleCmdLineTask
assert type(config)==lsst.pipe.tasks.exampleCmdLineTask.ExampleCmdLineConfig, 'config is
of type %s.%s instead of lsst.pipe.tasks.exampleCmdLineTask.ExampleCmdLineConfig' %
(type(config).__module__, type(config).__name__)
config.stats.badMaskPlanes=['EDGE']
config.stats.numSigmaClip=3.0
config.stats.numIter=2
config.doFail=False

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 \
                          --config stats.numSigmaClip=10
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: clipped mean=1183.30; meanErr=0.03; stdDev=37.83; stdDevErr=nan

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 \
                          --config stats.badMaskPlanes=['EDGE', 'NO_DATA']
usage: exampleCmdLineTask.py input [options]
exampleCmdLineTask.py: error: --config value NO_DATA] m                       lue
```

This is impossible!

Configuration files

```
$ cat example.config
config.stats.badMaskPlanes=['EDGE', 'NO_DATA']

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 -C example.config
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: clipped mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
```

- Be smart: catch the output of `--show config`, edit that, run the task.
- There are some types of configuration which can only be set in the configuration file, not on the command line (lists of values, subtasks, …)
- Recall the standard overrides we saw earlier:
  - `${PACKAGE}/config/${TASK_NAME}.py`;
  - `${PACKAGE}/config/${CAMERA_NAME}/${TASK_NAME}.py`

Storing and persisting configuration

- *Most* tasks (but not our exampleCmdLineTask.py) will store their configuration to the repository when they are run.
  - Task authors have to explicitly disable this.
- They will then *refuse to run again* if the configuration has changed.
  - This is an effort to aid reproducibility.

```
Mosaic FATAL: Comparing configuration: Inequality in doSolveFlux: False != True
Mosaic FATAL: Comparing configuration: Inequality in doColorTerms: False != True
Mosaic FATAL: Comparing configuration: Inequality in doSolveWcs: False != True
Mosaic FATAL: Failed in task initialization: Config does not match existing task config
              'Mosaic_config' on disk; task configurations must be consistent within the
              same output repo
```

- Great in production; can be tiresome when experimenting.
  - Override with `--clobber-config`.

Storing and persisting configuration

- *Most* tasks (but not our exampleCmdLineTask.py) will store their configuration to the repository when they are run.
  - Task authors have to explicitly disable this.
- They will then *refuse to run again* if the configuration has changed.
  - This is an effort to aid reproducibility.

(Different task)

```
Mosaic FATAL: Comparing configuration: Inequality in doSolveFlux: False != True
Mosaic FATAL: Comparing configuration: Inequality in doColorTerms: False != True
Mosaic FATAL: Comparing configuration: Inequality in doSolveWcs: False != True
Mosaic FATAL: Failed in task initialization: Config does not match existing task config
              'Mosaic_config' on disk; task configurations must be consistent within the
              same output repo
```

- Great in production; can be tiresome when experimenting.
  - Override with `--clobber-config`.

Hierarchical tasks

- `exampleCmdLineTask.py` uses the `ExampleSigmaClippedStatsTask` (seen earlier) as a "subtask".

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 --show tasks
Subtasks:
stats: lsst.pipe.tasks.exampleStatsTasks.ExampleSigmaClippedStatsTask
```

- We can swap this with (or "retarget it to") another task which follows the same interface, like `ExampleSimpleStatsTask`.

```
$ cat example.config
from lsst.pipe.tasks.exampleStatsTasks import ExampleSimpleStatsTask
config.stats.retarget(ExampleSimpleStatsTask)

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 --show config \
                                                     -C example.config
config.stats.retarget(target=lsst.pipe.tasks.exampleStatsTasks.ExampleSimpleStatsTask,
                      ConfigClass=lsst.pex.config.config.Config)
config.doFail=False

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 -C example.config
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: simple mean=1216.76; meanErr=0.97; stdDev=1395.78; stdDevErr=34.95
```

Hierarchical tasks

- `exampleCmdLineTask.py` uses the `ExampleSigmaClippedStatsTask` (seen earlier) as a "subtask".

```
$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 --show tasks
Subtasks:
stats: lsst.pipe.tasks.exampleStatsTasks.ExampleSigmaClippedStatsTask
```

- We can swap this with (or "retarget it to") another task which follows the same … `SimpleStatsTask`.

Note updated config; no `numIter`, etc.

```
$ cat example.config
from lsst.pipe.tasks.example                              pleStatsTask
config.stats.retarget(ExampleSim

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 --show config \
                                                     -C example.config
config.stats.retarget(target=lsst.pipe.tasks.exampleStatsTasks.ExampleSimpleStatsTask,
                      ConfigClass=lsst.pex.config.config.Config)
config.doFail=False

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 -C example.config
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: simple mean=1216.76; meanErr=0.97; stdDev=1395.78; stdDevErr=34.95
```

Elaborate hierarchies

Compose high-level processing out of lower-level tasks.

```
$ processCcd.py ${OBS_TEST_DIR}/data/input --show tasks
Subtasks:
calibrate: lsst.pipe.tasks.calibrate.CalibrateTask
calibrate.astrometry: lsst.meas.astrom.astrometry.AstrometryTask
calibrate.astrometry.matcher: lsst.meas.astrom.matchOptimisticB.MatchOptimisticBTask
calibrate.astrometry.refObjLoader:
lsst.meas.astrom.loadAstrometryNetObjects.LoadAstrometryNetObjectsTask
calibrate.astrometry.wcsFitter: lsst.meas.astrom.fitTanSipWcs.FitTanSipWcsTask
calibrate.detection: lsst.meas.algorithms.detection.SourceDetectionTask
calibrate.initialMeasurement: lsst.meas.base.sfm.SingleFrameMeasurementTask
calibrate.initialMeasurement.applyApCorr: lsst.meas.base.applyApCorr.ApplyApCorrTask
calibrate.measureApCorr: lsst.meas.base.measureApCorr.MeasureApCorrTask
calibrate.measurePsf: lsst.pipe.tasks.measurePsf.MeasurePsfTask
calibrate.measurement: lsst.meas.base.sfm.SingleFrameMeasurementTask
calibrate.measurement.applyApCorr: lsst.meas.base.applyApCorr.ApplyApCorrTask
calibrate.photocal: lsst.pipe.tasks.photoCal.PhotoCalTask
calibrate.repair: lsst.pipe.tasks.repair.RepairTask
deblend: lsst.meas.deblender.deblend.SourceDeblendTask
detection: lsst.meas.algorithms.detection.SourceDetectionTask
isr: lsst.ip.isr.isrTask.IsrTask
isr.assembleCcd: lsst.ip.isr.assembleCcdTask.AssembleCcdTask
isr.fringe: lsst.ip.isr.fringe.FringeTask
measurement: lsst.meas.base.sfm.SingleFrameMeasurementTask
measurement.applyApCorr: lsst.meas.base.applyApCorr.ApplyApCorrTask
```

Additional repositories

- *Most* tasks (but not our exampleCmdLineTask.py) produce data products (calibrated exposures, coadds, source lists, …).
- These are normally written to the repository provided as the first command line argument.
- We can use the `--output` option to specify another location for these:

```
$ taskName.py ${INPUT} --output=${OUTPUT}
```

- The task configuration is written to the output repository; use multiple outputs to experiment with different configs rather than continuously clobbering.
- Also note a similar mechanism for pointing to repositories of calibration products (bias, flats, darks, …) required by some tasks: `--calib=[…]`.

Debugging

```
$ cat debug.py
import lsstDebug

def DebugInfo(name):
    di = lsstDebug.getInfo(name)
    di.display = True
    return di

lsstDebug.Info = DebugInfo

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 --debug
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: simple mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
```

Debugging

Must use this filename

```
$ cat debug.py
import lsstDebug

def DebugInfo(name):
    di = lsstDebug.getInfo(name)
    di.display = True
    return di

lsstDebug.Info = DebugInfo

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 --debug
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: simple mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
```

Debugging

```
$ cat debug.py
import lsstDebug

def DebugInfo(name):
    di = lsstDebug.getInfo(name)
    di.display = True
    return di

lsstDebug.Info = DebugInfo

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id visit=1 --debug
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: simple mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
```

Can filter on name to only debug some tasks

Debugging

```
$ cat debug.py
import lsstDebug

def DebugInfo(name):
    di = lsstDebug.getInfo(name)
    di.display = True
    return di

lsstDebug.Info = Debug[illegible]

$ ./exampleCmdLineTask.py $[illegible]ut --id visit=1 --debug
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: simple mean=1184.70; meanErr=0.02; stdDev=33.64; stdDevErr=1.04
```

See task docs for relevant attributes.

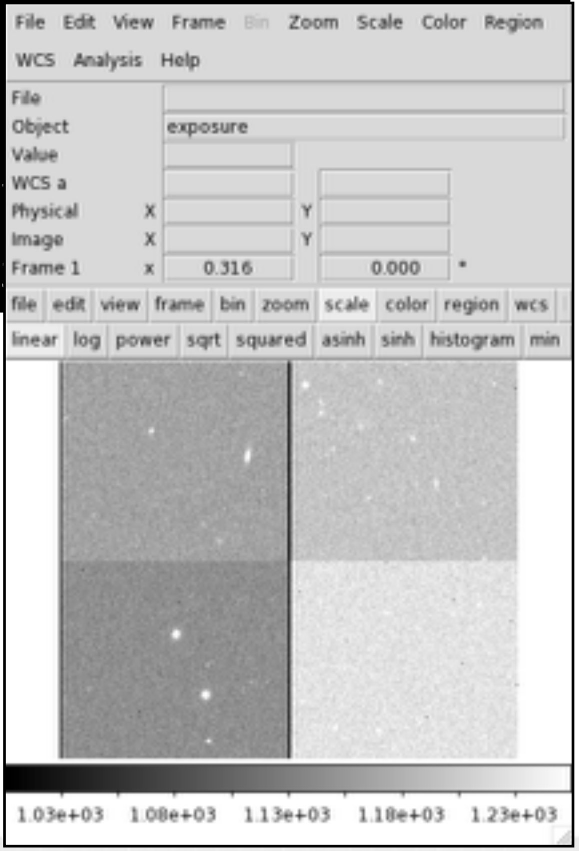
Debugging

```
$ cat debug.py
import lsstDebug

def DebugInfo(name):
    di = lsstDebug.getInfo(name)
    di.display = True
    return di

lsstDebug.Info = DebugInfo

$ ./exampleCmdLineTask.py ${OBS_TEST_DIR}/data/input --id vis
exampleTask: Processing data ID {'filter': 'g', 'visit': 1}
exampleTask.stats: simple mean=1184.70; meanErr=0.02; stdDev=
```


File Edit View Frame Bin Zoom Scale Color Region
WCS Analysis Help
file edit view frame bin zoom scale color region wcs
linear log power sqrt squared asinh sinh histogram min
1.03e+03 1.08e+03 1.13e+03 1.18e+03 1.23e+03

File			
Object	exposure		
Value			
WCS a			
Physical	X	Y	
Image	X	Y	
Frame 1	x	0.316	0.000

If you had ds9 running
or on your path.

Useful tricks when debugging

- Set the log level to something verbose:

  ```
  $ ./exampleCmdLineTask ${PATH} ---id --loglevel DEBUG
  ```

  - You'll see a lot more on-screen output explaining exactly what the task is doing.
  - **Caution:** `--loglevel DEBUG` is not the same as `--debug`, despite both "enabling debug output".

- Abort on error:

  ```
  $ ./exampleCmdLineTask ${PATH} --id --doraise
  ```

  - Throw an exception and exit as soon as an error occurs.
  - The alternative, default, behaviour is to press on to the next item of data.

Parallelism

- Use `-j ${NUMBER}` on the command line to use multiple processes to execute your task.
  - A given data reference will be run in a single process; multiple data references can run simultaneously.
  - Always on a single machine (using Python's built-in `multiprocessing`); you are limited by CPU/RAM/etc available.
  - Requires support from the task: not always possible.
- Larger scale parallelization is a job for the process execution middleware.

References

- Lots of valuable Doxygen documentation. In particular:
  - https://lsst-web.ncsa.illinois.edu/doxygen/x_masterDoxyDoc/pipe_base.html
    - An introduction to running (command line) tasks.
  - https://lsst-web.ncsa.illinois.edu/doxygen/x_masterDoxyDoc/group___l_s_s_t__task__documentation.html
    - List of available tasks and their documentation.