Time optimization

Rule #1: Don't do it!

Rule #2: Don't do it yet!

Rule #3: Don't mess up!

a.k.a.

Rule #1: First make it work, then make it fast

Rule #2: Measure before you optimize

Rule #3: Refactor under passing tests

Profilers (CPU time)

gprof

- Requires recompilation (static linking)
- Runtime overhead (2 – 3 times)
- Provides function call count

oprofile

- No recompile necessary (default build includes debug info)
- Negligible runtime overhead (few percent)
- Requires admin priviledges
- Linux only (Mac: see Shark)
- No function call count

oprofile how-to

Make sure that “sudo opcontrol” and “dot” works from the shell, then

```
> sudo opcontrol --deinit && sudo opcontrol --init && sudo opcontrol --reset && \
  sudo opcontrol --start --callgraph=999 --no-vmlinux --separate=lib --event="default"
> my_benchmark
> sudo opcontrol --stop && sudo opcontrol --dump
> opreport -clf image-exclude:/no-vmlinux my_benchmark |
.../code/xmlcasa/scripts/regressions/admin/gprof2dot.py -e0.1 -n1 -f oprofile | dot -Tpng -o
profile.png
> opannotate --source > annotated_source.cc
```

Or from casapy, after putting your script in .../code/xmlcasa/scripts/regressions/my_script.py,

```
<2> publish_summary.runTest('my_script', profile=True)
```

A faster flagdata

flagdata(mode="quack", …) benchmark using a 0.98 GB MS (15 MB flags), npol = 1, nchan = 256, 470520 rows

TSM mode	cache	buffer	buffer2
Execution time	1:24 min	1:40 min	0:43 min
Throughput rate	0.17 MB/s	0.14 MB/s	0.35 MB/s
RFASelector::iterTime()	9%	8%	20%
VisIter::operator++()	12%	10%	25%
VisIter::flag()	30%	35%	6%
VisIter::setFlag()	31%	32%	6%
TSMDataColumns:: accessSection()	61%	66%	12%