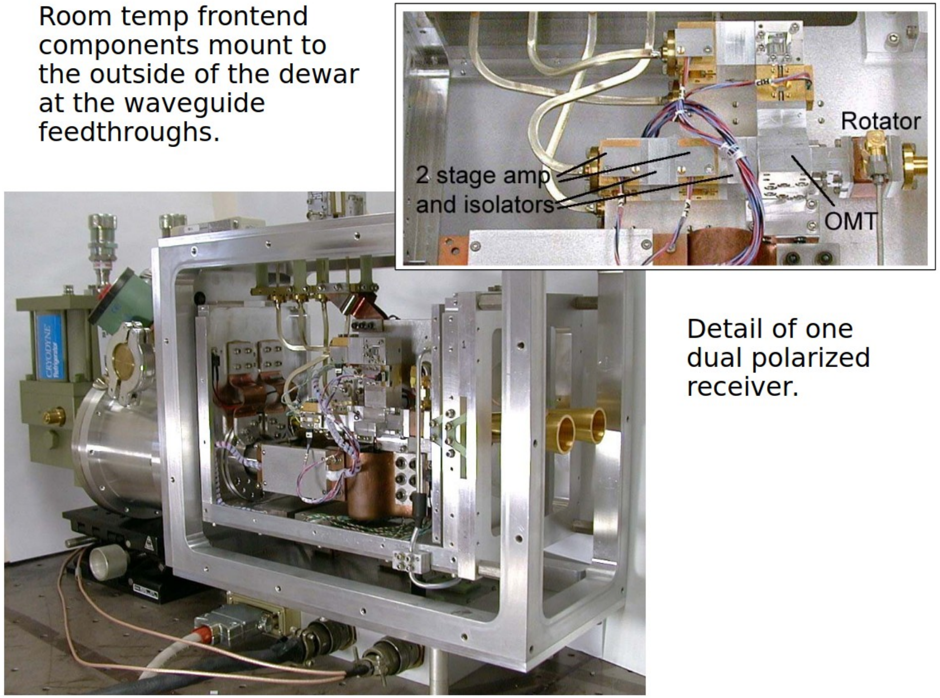
RSR Data Reduction (QA)



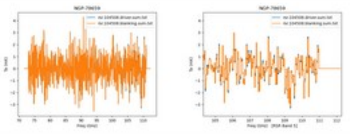
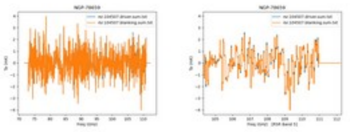
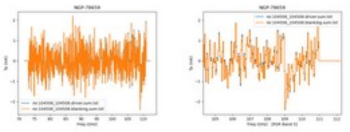
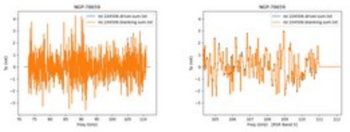
Peter Teuben
teuben@umd.edu

8-feb-2023

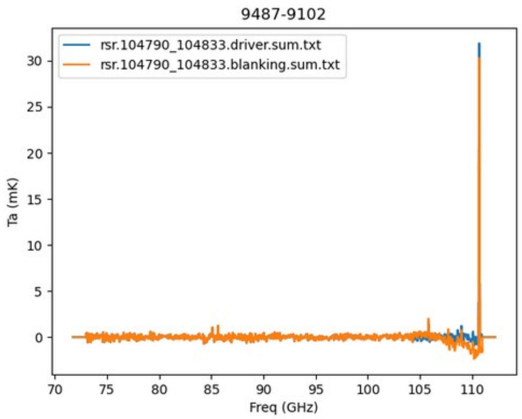
https://www.astro.umd.edu/~teuben/LMT/RSR_pipeline_2023.pdf

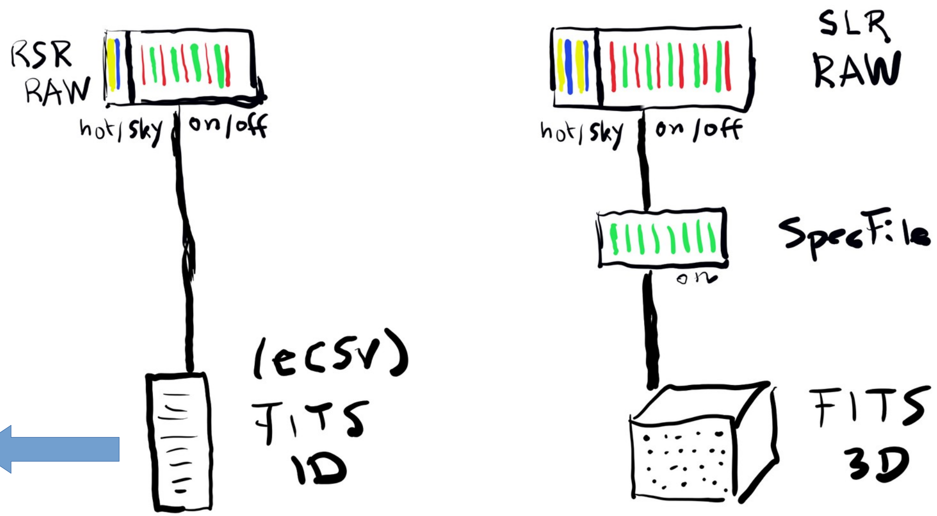
SLpipeline: RSR quality

Summary of all obsnum's and combinations: (click on column name to sort by that column)

#	ObservingDate ▲	ObsNum	SourceName	RestFreq	inttime [s]	tau	RMS [mK]	RMS/RMS0 ratio	aPlot	comments
83.	"2023-02-02T12:29:42"	104508	NGP-78659		318.1	0.041	0.976351	0.97345 /100K		
82.	"2023-02-02T12:23:20"	104507	NGP-78659		319.3	0.041	0.962454	0.961396 /100K		
81.	"2023-02-02T12:17:03"	104506_104508	NGP-78659		951.1	0.041	0.58102	1.00168 /100K		
80.	"2023-02-02T12:17:03"	104506	NGP-78659		313.7	0.041	0.93598	0.926722 /100K		

RSR pipeline (diagram)





RSR vs. SEQ

Data: 256 channels x 6 boards x 4 chassis x 5 integrations

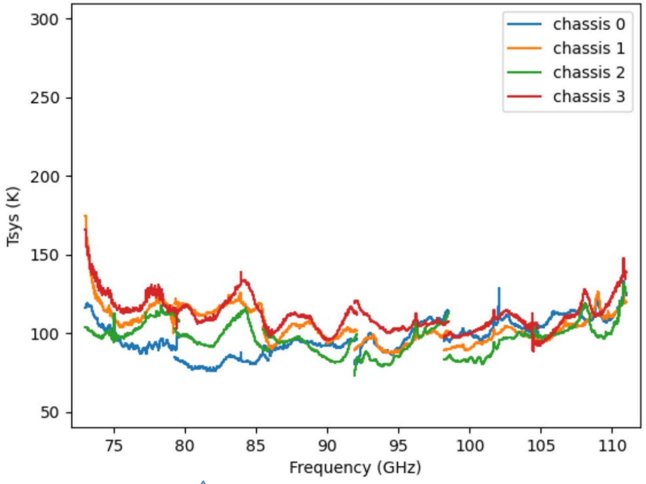
~ 1300 spectral channels x 20 times

30,720 data → ~1300 channels

Or: DATA[5,4,6,256]

RSR pipeline

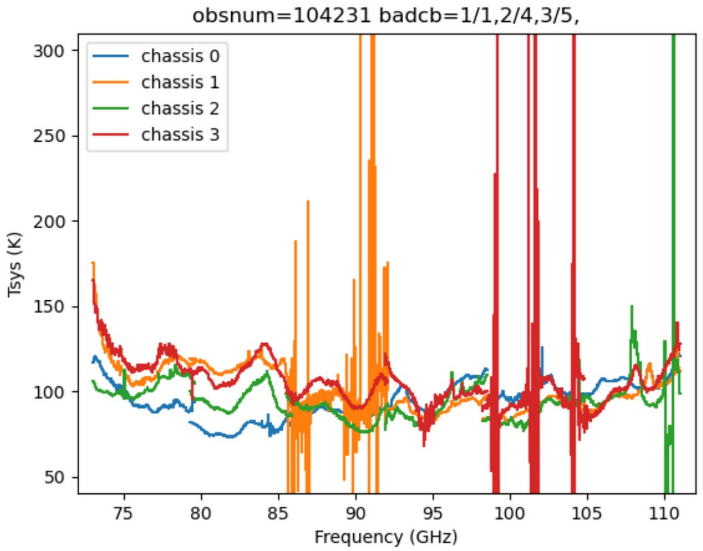
1) **Tsys** look at large channel-to-channel jitter
   currently 1/1, 2/4 and 3/5 appear “bad”. This might give some **badcb**’s
2) **BadLags** plot shows ACF variations and resulting **badlags** and might give some **badcb**’s as well.
3) **Waterfall** plots created using rsr_driver, with all badlags and badcb applied. Are the repeats consistent? Need additional manual **badcb**’s or other tuning?
4) **Spectrum** created using rsr_driver and rsr_sum, compare.
5) **Stats**, radiometer, peak detection (even useful for linecheck=0) etc.

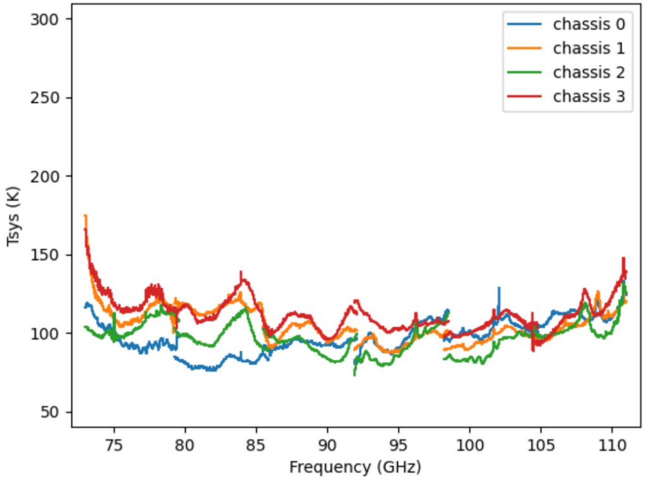


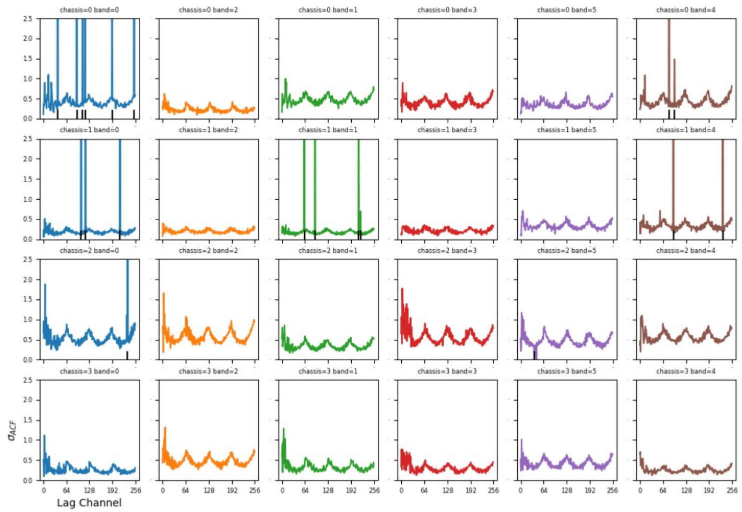
1) Tsys look at large channel-to-channel jitter
currently 1/1, 2/4 and 3/5 appear "bad".
This might give some **badcb**'s

January 2015

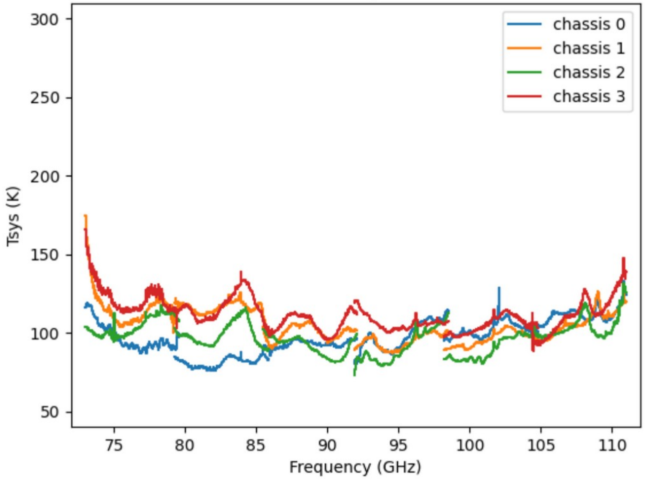
January 2023

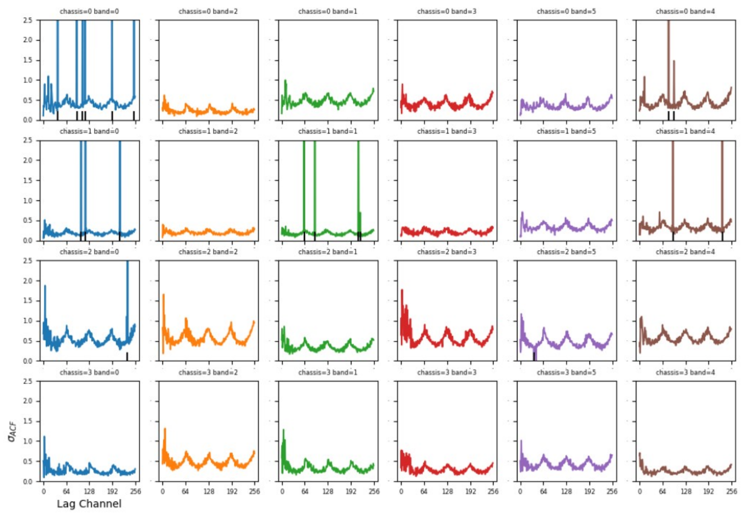


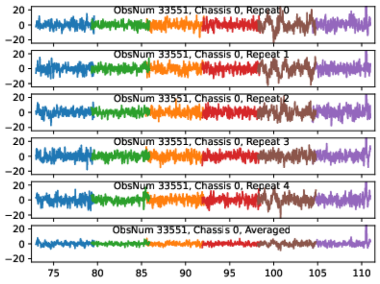


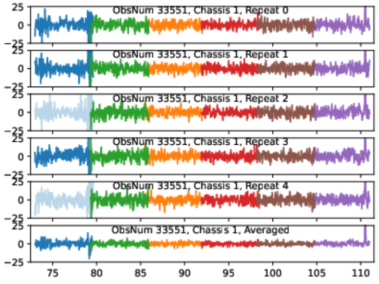


2) BadLags plot shows ACF variations and resulting **badlags** and might give some **badcb**’s as well.

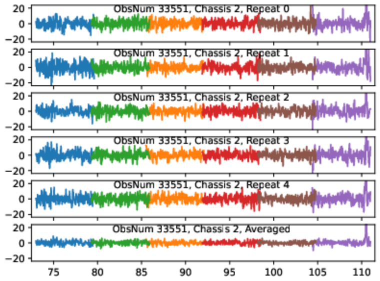


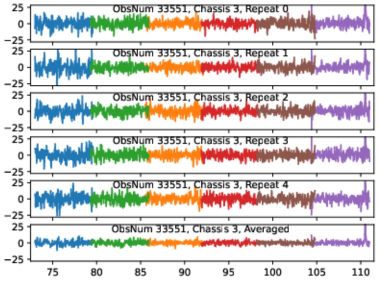


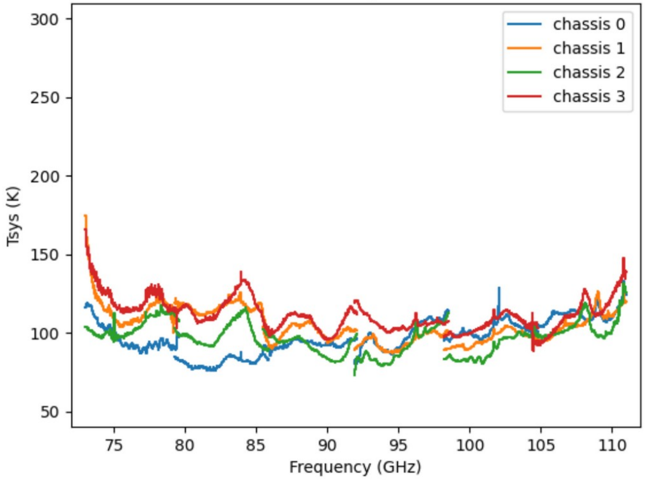


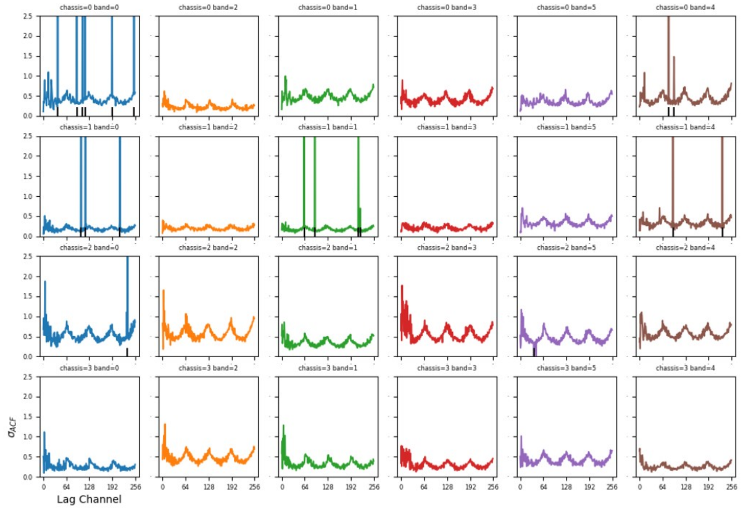


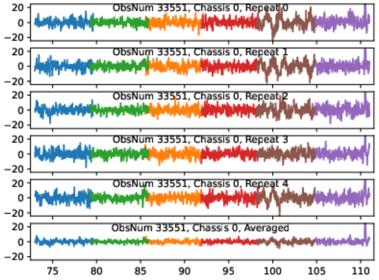
3) Waterfall plots created using rsr_driver, with all badlags and badcb applied. Are the repeats consistent? Need additional manual **badcb**'s or other tuning?

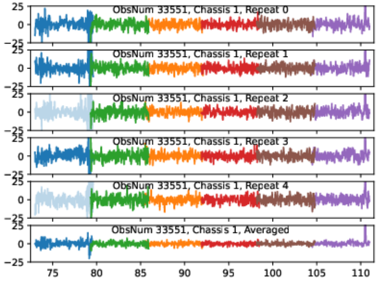


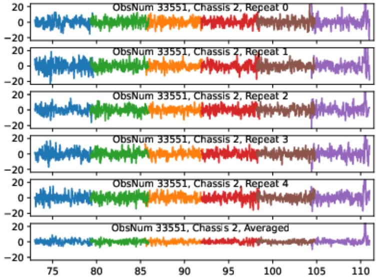


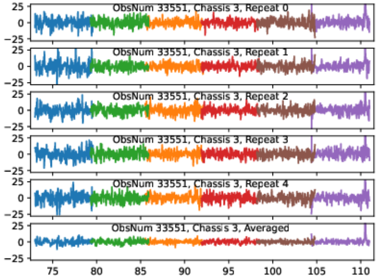


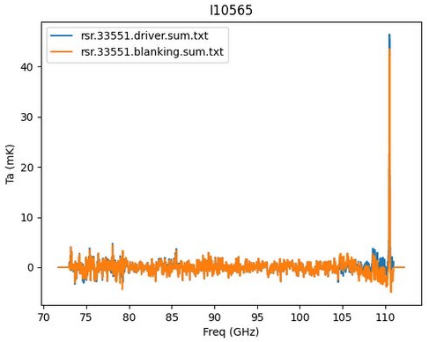


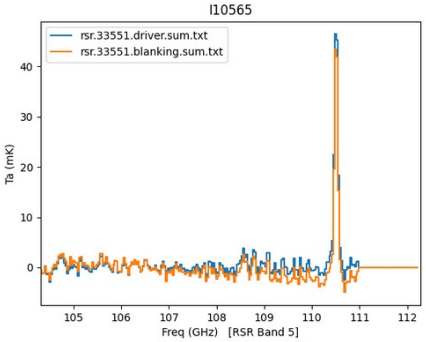




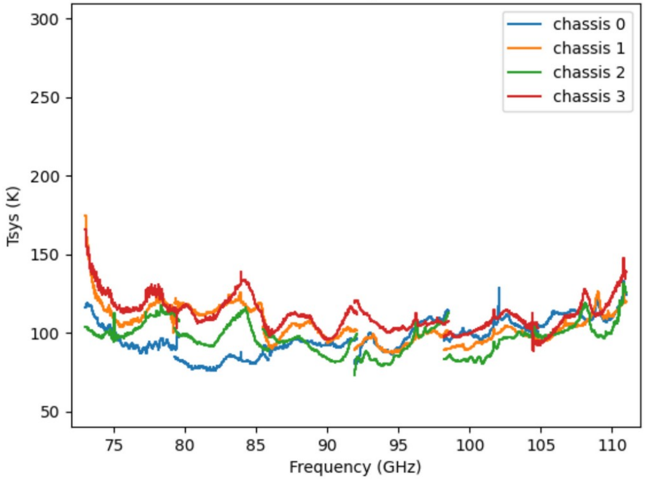


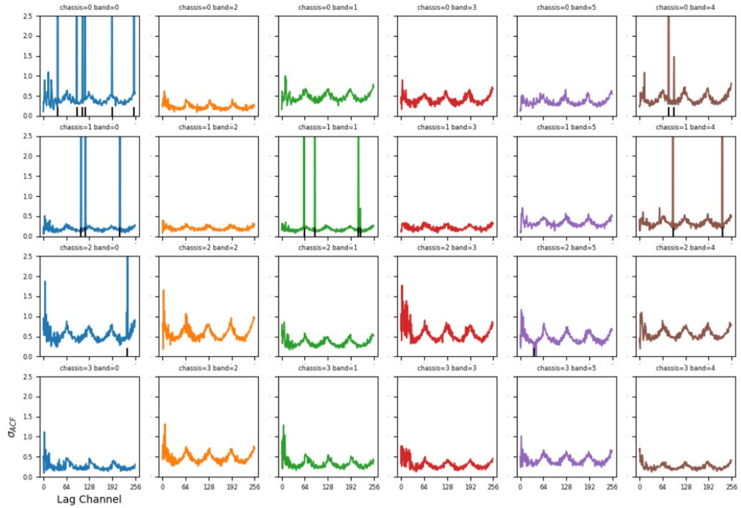


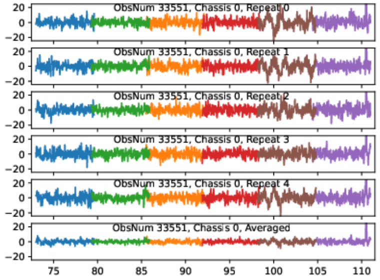


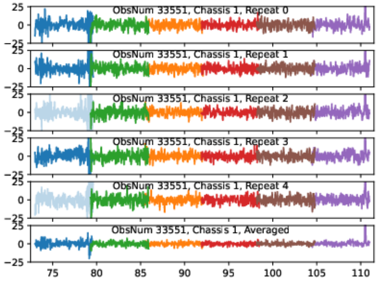


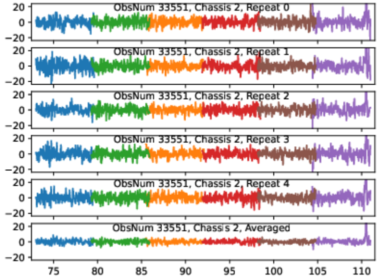
4) **Spectrum** created using rsr_driver and rsr_sum, compare.

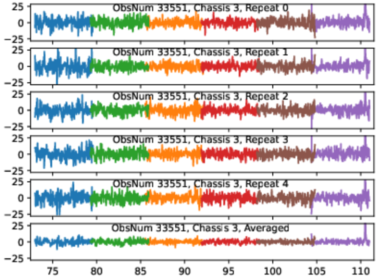


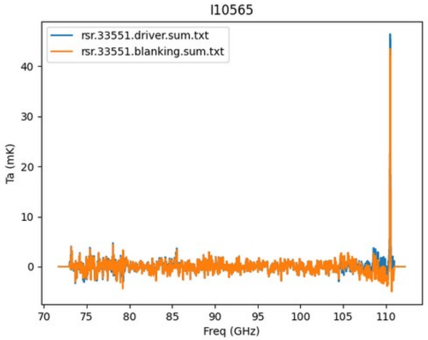


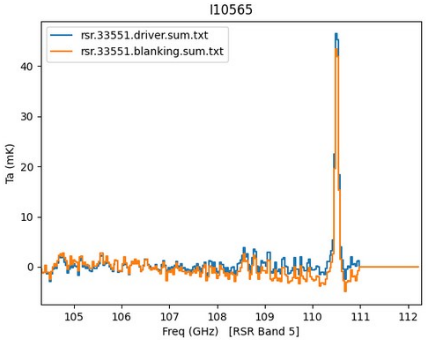










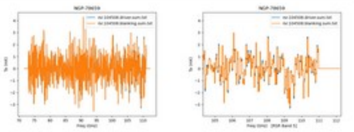
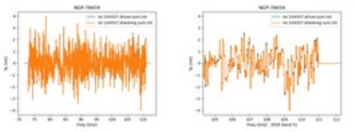
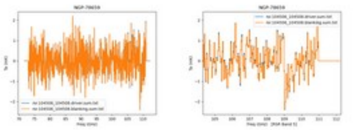
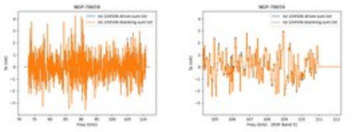


rsr.33551.driver.sum.txt	rms=0.919 mK
rsr.33551.blanking.sum.txt	rms=0.986 mK
Expected noise for Tsys=100K	rms=1.025 mK
LineCheck driver	rms=1.129 mK base=0.135 mK peak=50.373 mK line=110.514 GHz sigma=0.0351429 GHz
LineCheck blanking	rms=1.343 mK base=-1.026 mK peak=48.756 mK line=110.514 GHz sigma=0.0336182 GHz

5) **Stats**, radiometer, peak detection etc.

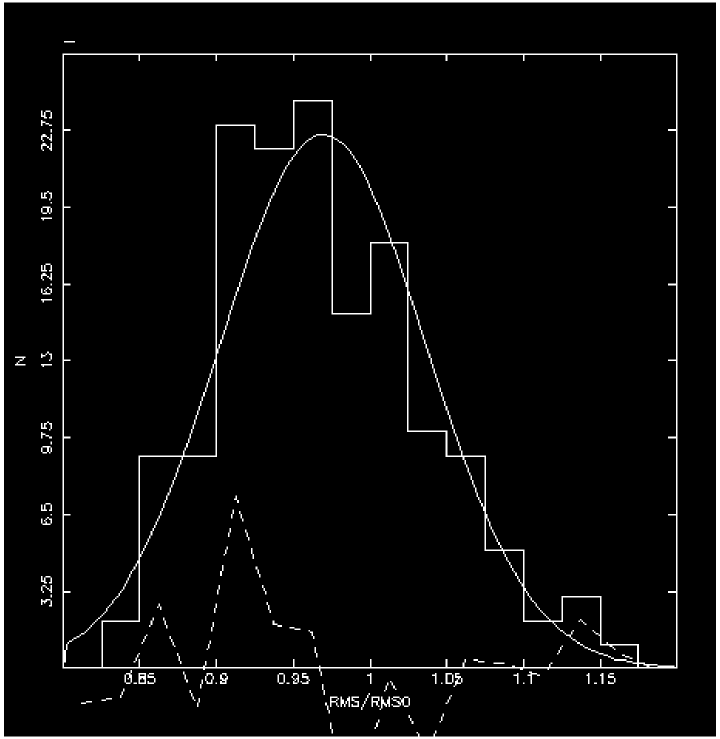
Project Summary

Summary of all obsnum's and combinations: (click on column name to sort by that column)

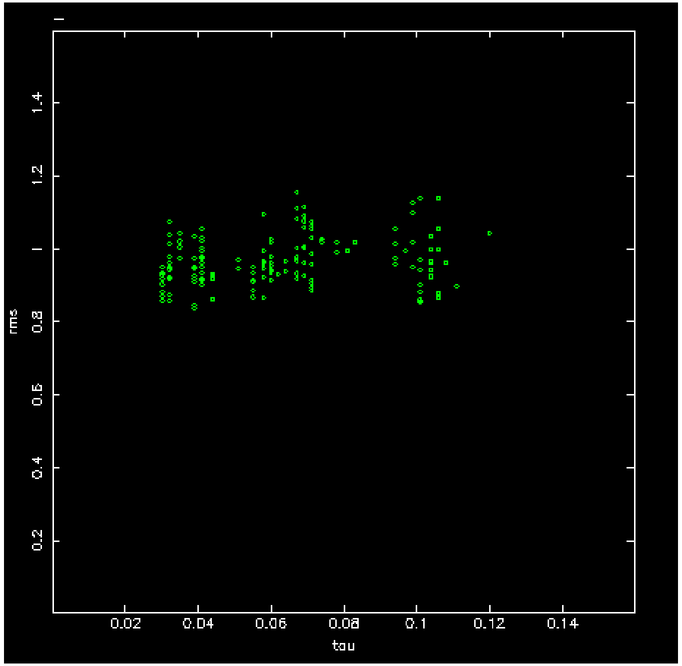
#	ObservingDate ▲	ObsNum	SourceName	RestFreq	inttime [s]	tau	RMS [mK]	RMS/RMS0 ratio	aPlot	comments
83.	"2023-02-02T12:29:42"	104508	NGP-78659		318.1	0.041	0.976351	0.97345 /100K		
82.	"2023-02-02T12:23:20"	104507	NGP-78659		319.3	0.041	0.962454	0.961396 /100K		
81.	"2023-02-02T12:17:03"	104506_104508	NGP-78659		951.1	0.041	0.58102	1.00168 /100K		
80.	"2023-02-02T12:17:03"	104506	NGP-78659		313.7	0.041	0.93598	0.926722 /100K		

summary.csv contains a number of project parameters:

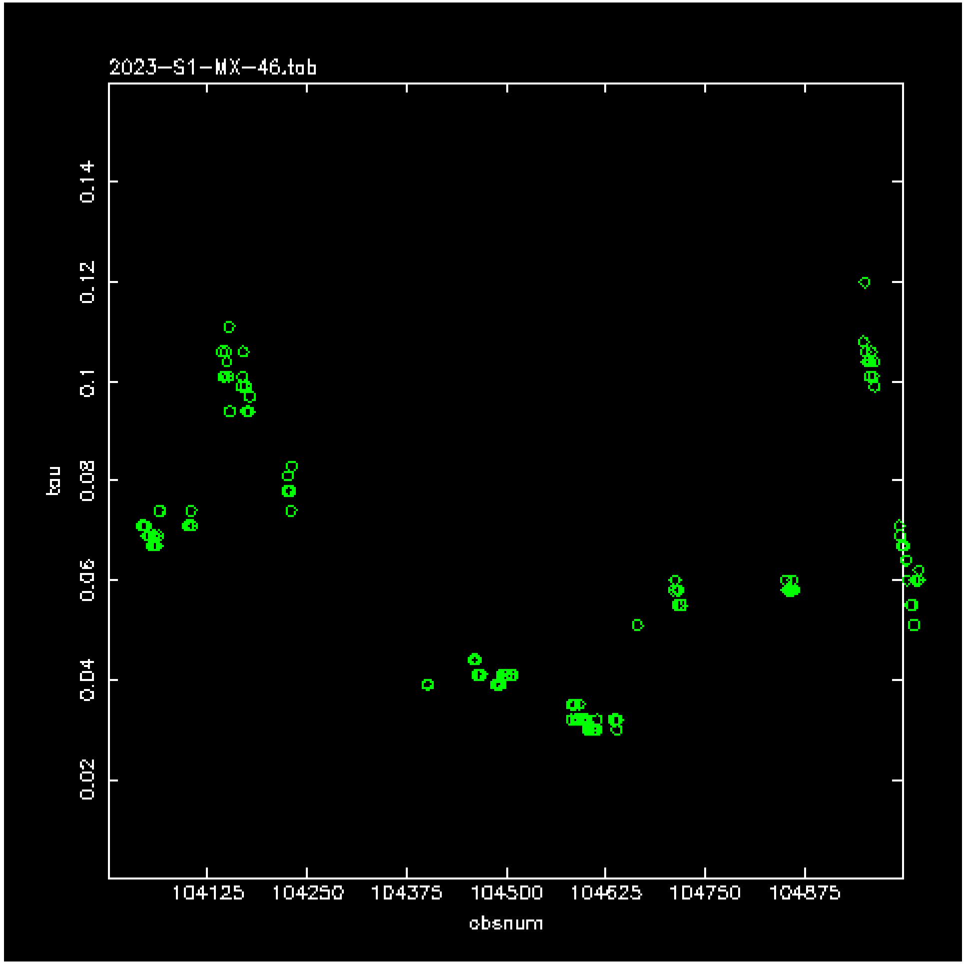
- ObsNum
- ObsDateTime
- Source
- IntTime
- Tau
- RMS
- RMS0
- ...



RMS/RMS0 histogram



Tau vs. RMS



ObsNum vs. Tau

2013-S1-MX-46
(152 obsnum’s through Feb 8)

Let’s go online?

1) TAP from LMT (usually < 5 mins after observing)
   http://taps.lmtgtm.org/lmtslr/2023-S1-MX-46/TAP/

TAP = Timely Analysis Products

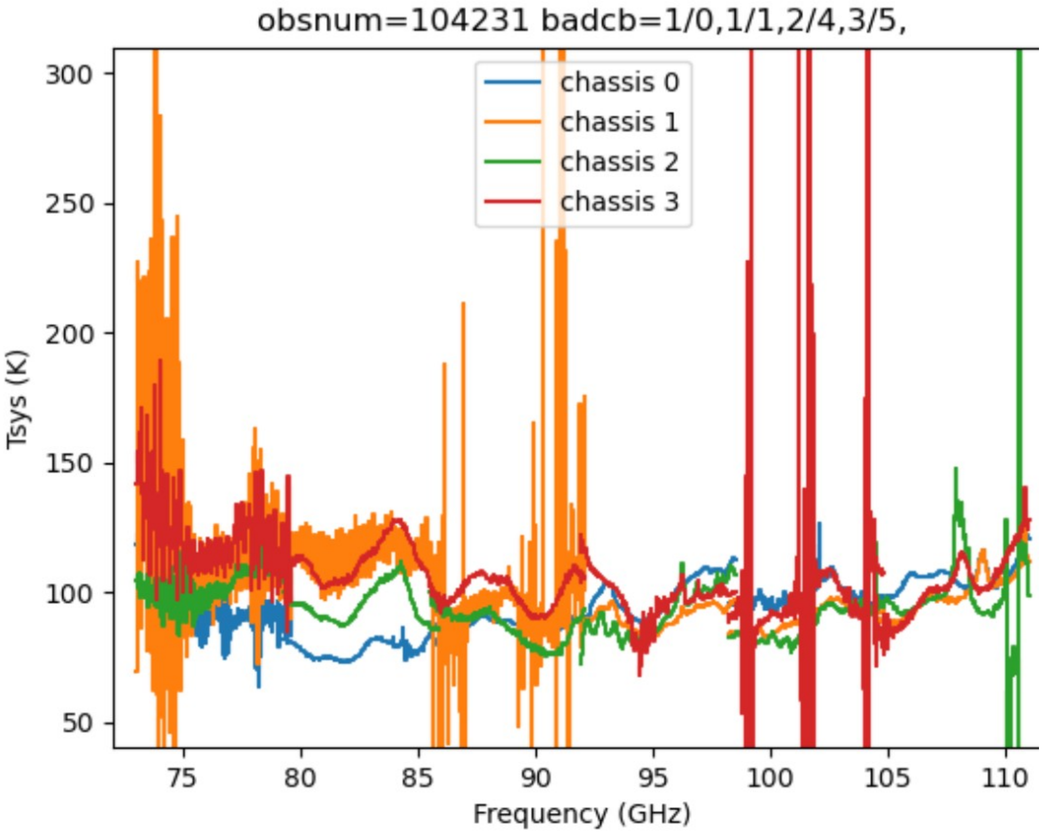
2) Reduced on Unity (usually ~1 day later)
   http://taps.lmtgtm.org/lmtslr/2023-S1-MX-46/

full SRDP

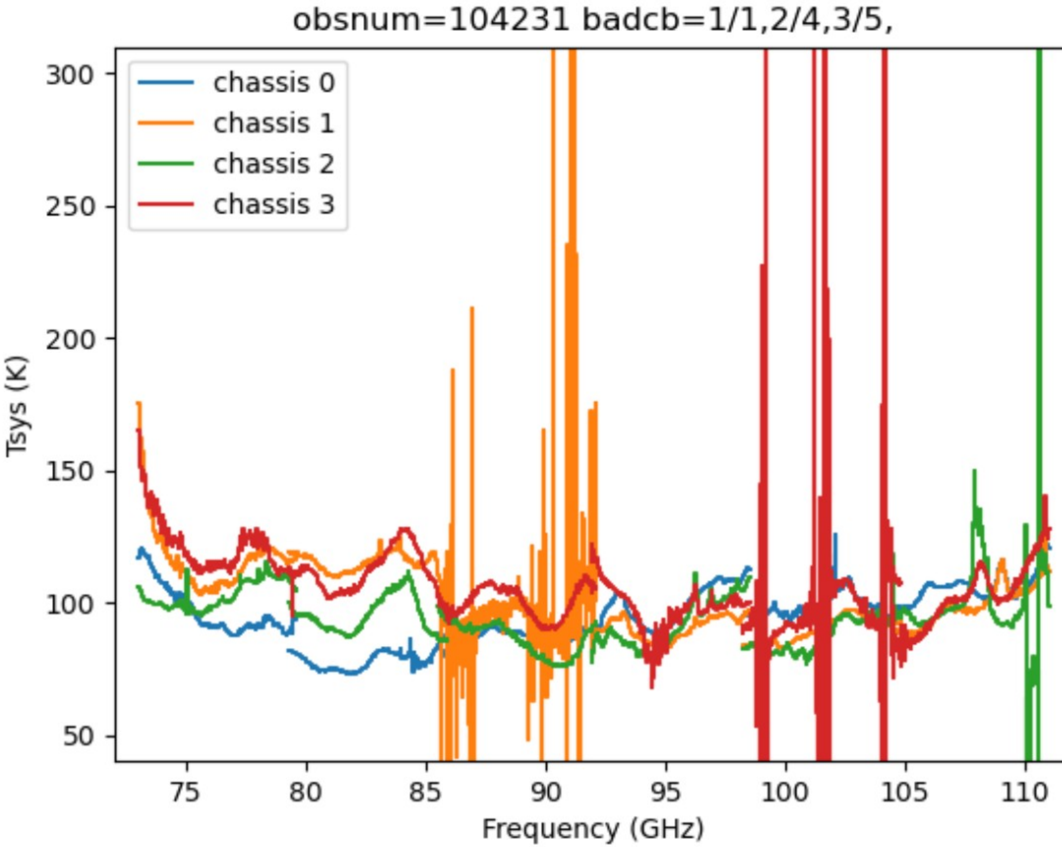
3) Reduced and QA done (TBD)

4) Archive into DataVerse (TBD)

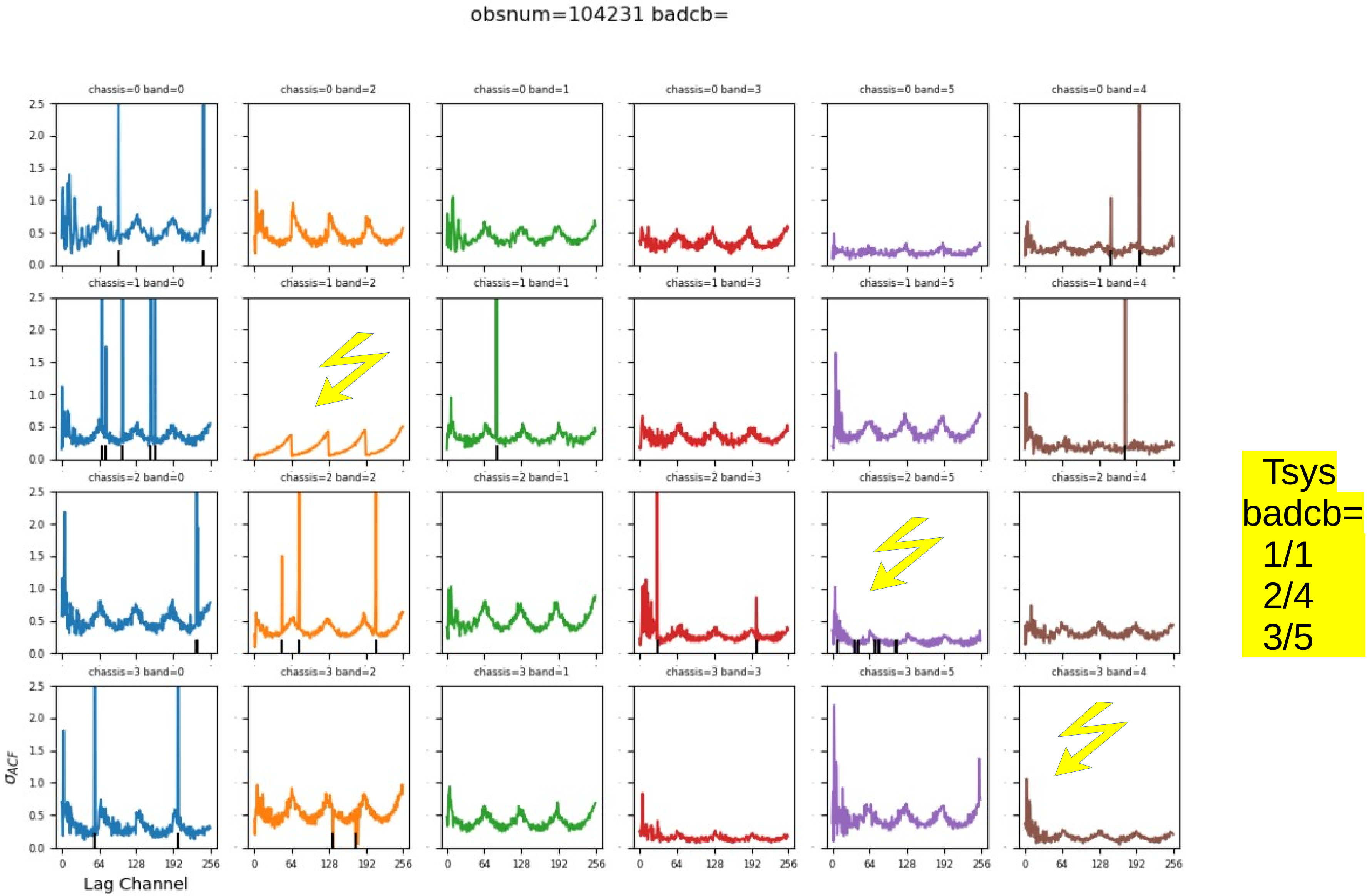
badcb= based on jitter in the Tsys value per Chassis/Board (C/B)

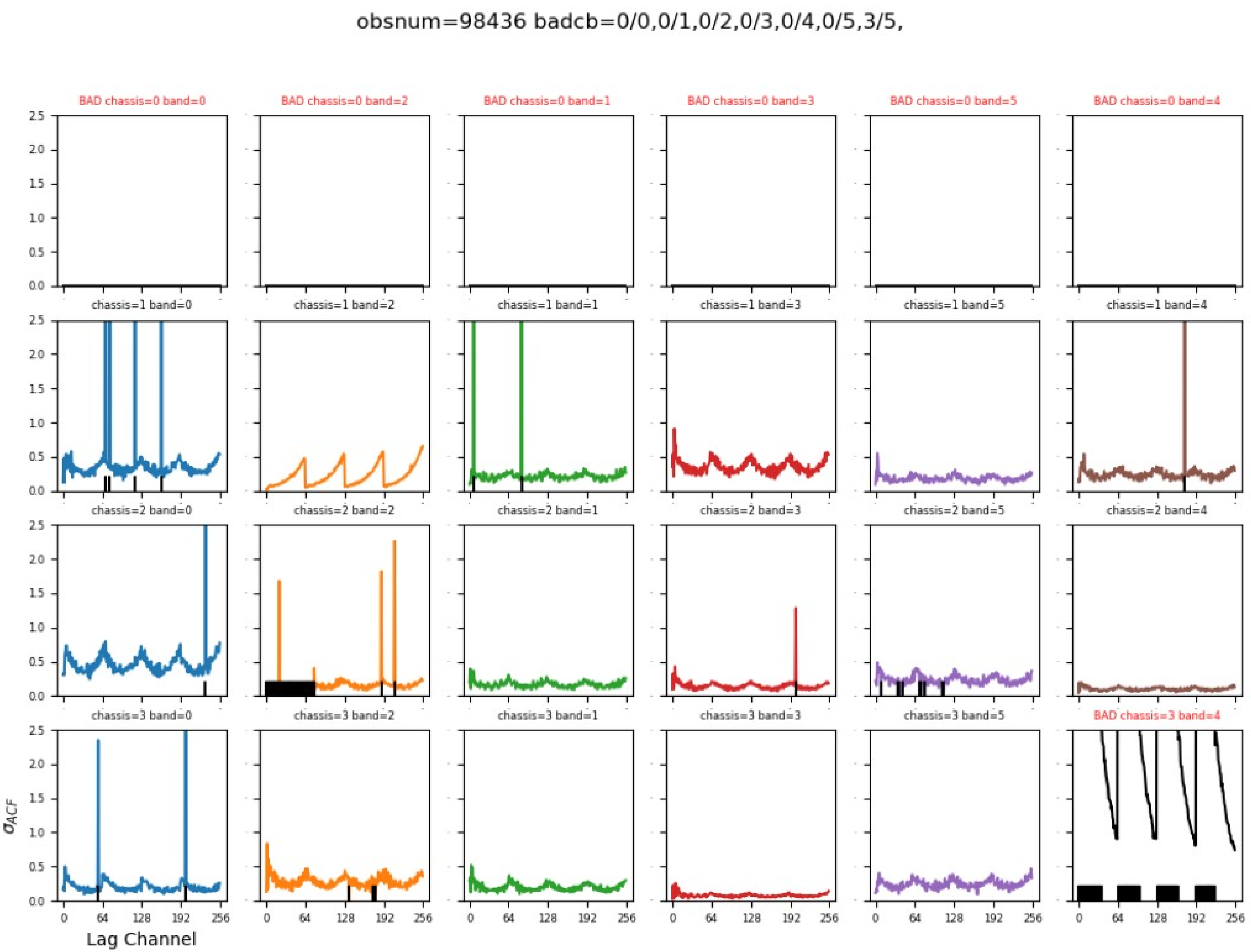


no badlags applied

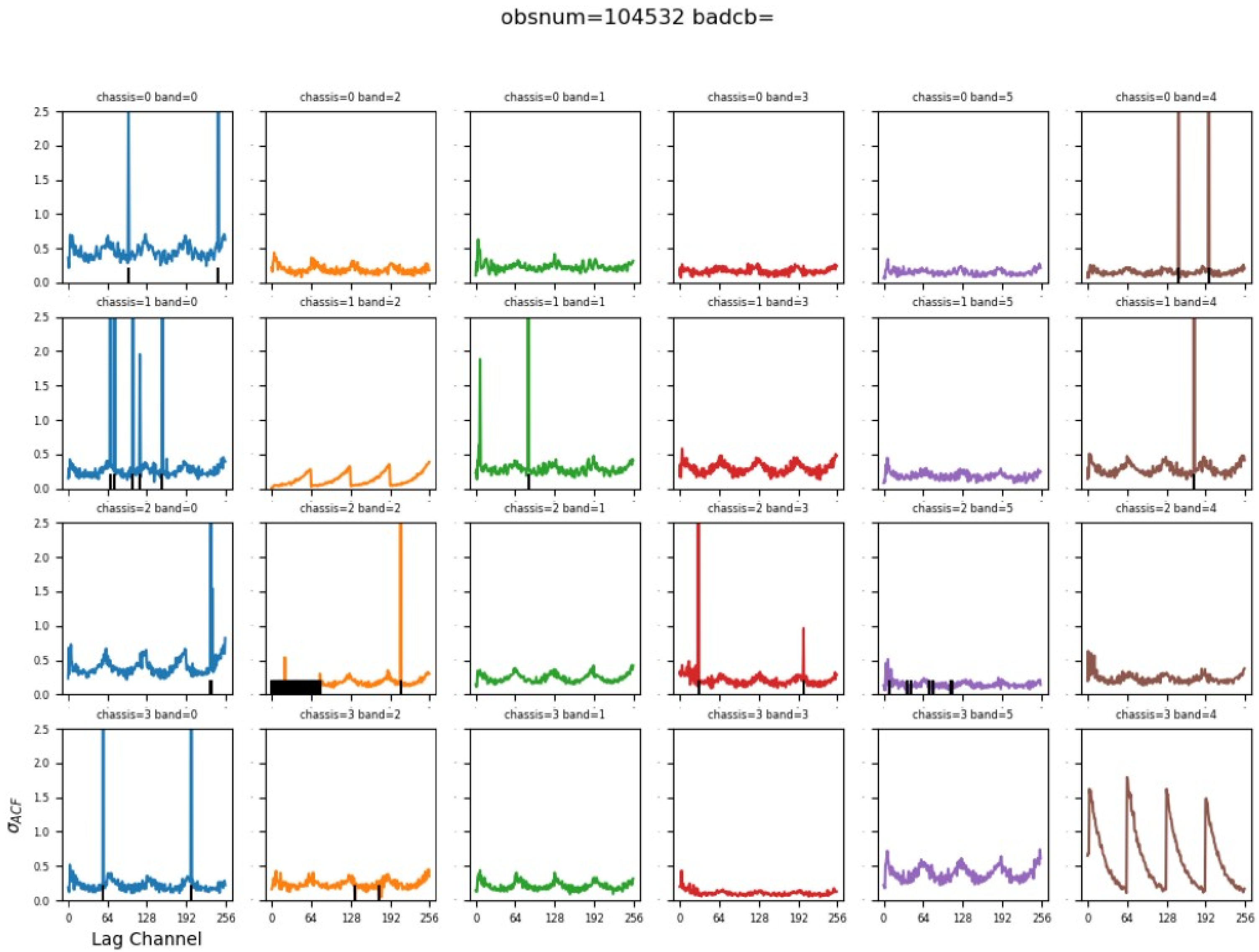


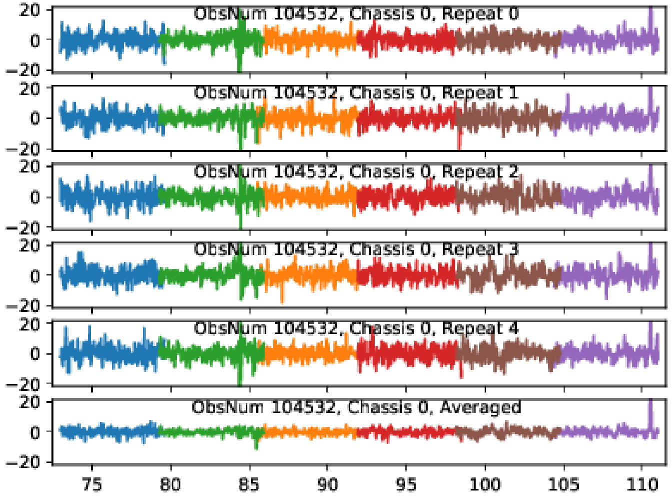
badlags applied

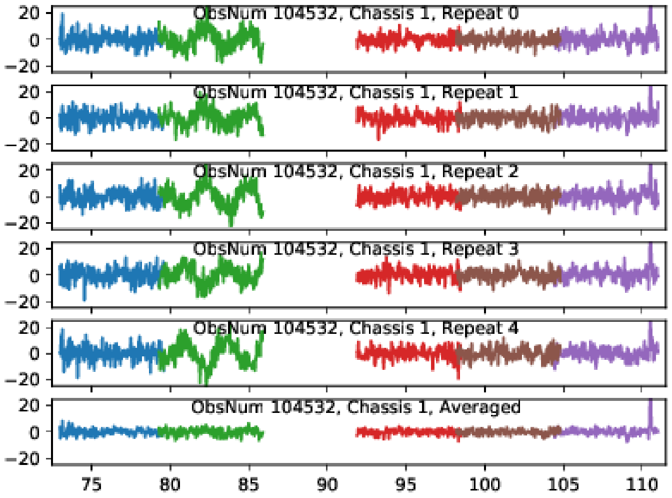

obsnum=104231 badcb=
chassis=0 band=0
chassis=0 band=2
chassis=0 band=1
chassis=0 band=3
chassis=0 band=5
chassis=0 band=4
chassis=1 band=0
chassis=1 band=2
chassis=1 band=1
chassis=1 band=3
chassis=1 band=5
chassis=1 band=4
chassis=2 band=0
chassis=2 band=2
chassis=2 band=1
chassis=2 band=3
chassis=2 band=5
chassis=2 band=4
chassis=3 band=0
chassis=3 band=2
chassis=3 band=1
chassis=3 band=3
chassis=3 band=5
chassis=3 band=4
σ_{ACF}
Lag Channel
Tsys
badcb=
1/1
2/4
3/5

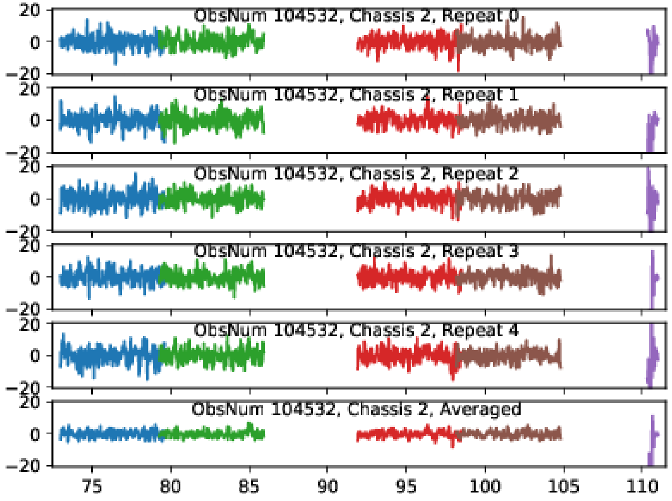


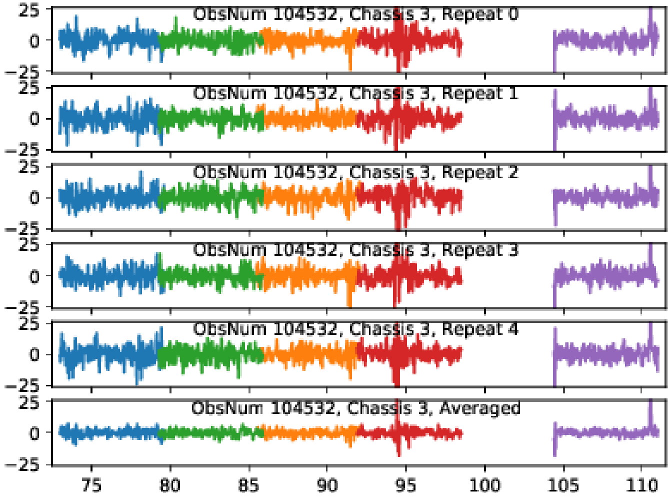
obsnum=98436 badcb=0/0,0/1,0/2,0/3,0/4,0/5,3/5,
BAD chassis=0 band=0
BAD chassis=0 band=2
BAD chassis=0 band=1
BAD chassis=0 band=3
BAD chassis=0 band=5
BAD chassis=0 band=4
chassis=1 band=0
chassis=1 band=2
chassis=1 band=1
chassis=1 band=3
chassis=1 band=5
chassis=1 band=4
chassis=2 band=0
chassis=2 band=2
chassis=2 band=1
chassis=2 band=3
chassis=2 band=5
chassis=2 band=4
chassis=3 band=0
chassis=3 band=2
chassis=3 band=1
chassis=3 band=3
chassis=3 band=5
BAD chassis=3 band=4
σ_{ACF}
Lag Channel


obsnum=104532 badcb=
chassis=0 band=0
chassis=0 band=2
chassis=0 band=1
chassis=0 band=3
chassis=0 band=5
chassis=0 band=4
chassis=1 band=0
chassis=1 band=2
chassis=1 band=1
chassis=1 band=3
chassis=1 band=5
chassis=1 band=4
chassis=2 band=0
chassis=2 band=2
chassis=2 band=1
chassis=2 band=3
chassis=2 band=5
chassis=2 band=4
chassis=3 band=0
chassis=3 band=2
chassis=3 band=1
chassis=3 band=3
chassis=3 band=5
chassis=3 band=4
0.0
0.5
1.0
1.5
2.0
2.5
0
64
128
192
256
σ_{ACF}
Lag Channel





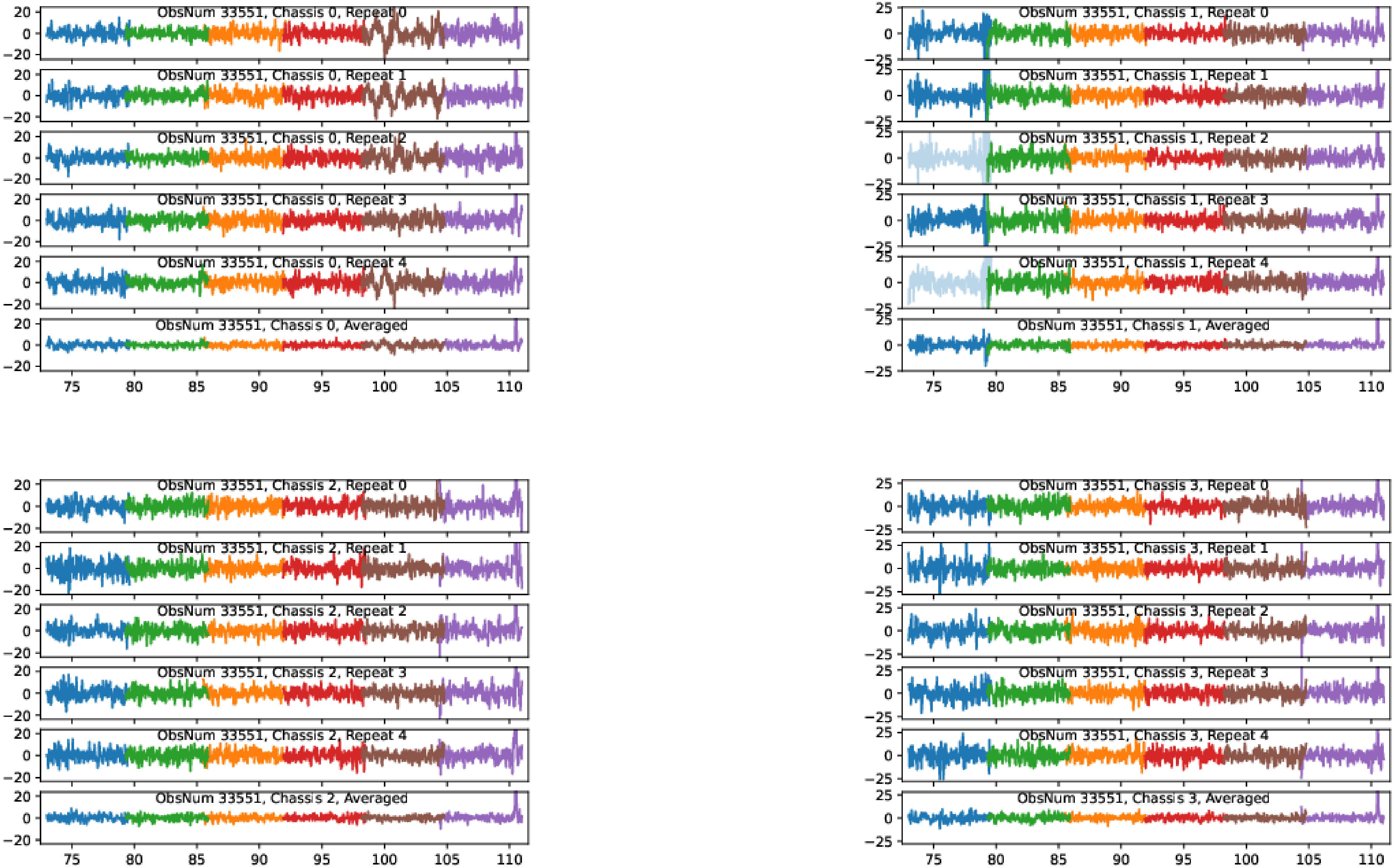




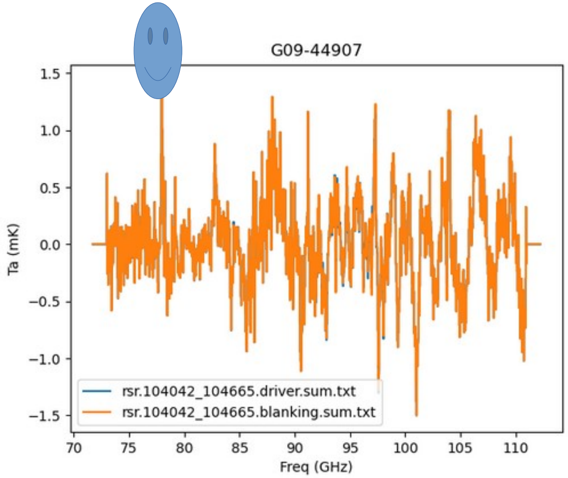
0/2 bad freq
3/3 bad freq

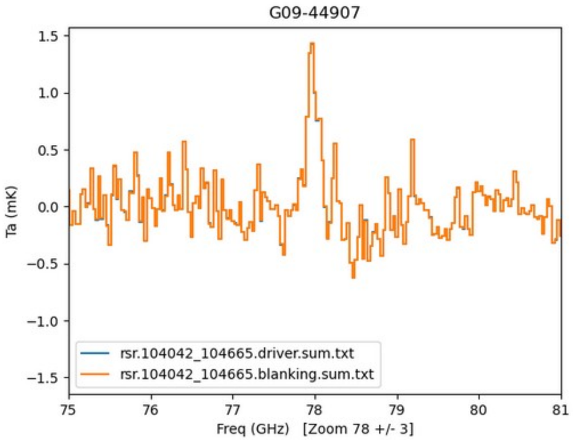
1/2 wavy

2/4 should be blank (bug)

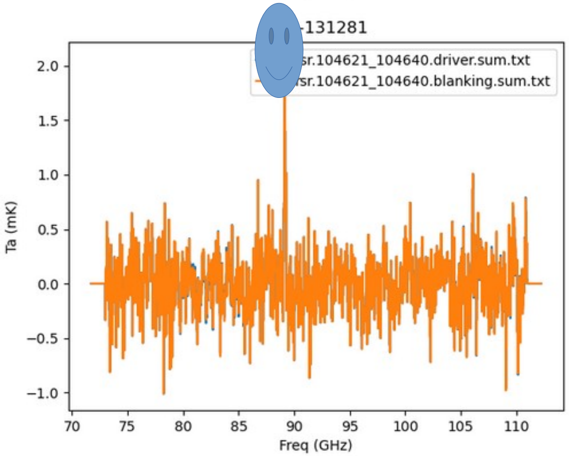

ObsNum 33551, Chassis 0, Repeat 0
ObsNum 33551, Chassis 0, Repeat 1
ObsNum 33551, Chassis 0, Repeat 2
ObsNum 33551, Chassis 0, Repeat 3
ObsNum 33551, Chassis 0, Repeat 4
ObsNum 33551, Chassis 0, Averaged
ObsNum 33551, Chassis 1, Repeat 0
ObsNum 33551, Chassis 1, Repeat 1
ObsNum 33551, Chassis 1, Repeat 2
ObsNum 33551, Chassis 1, Repeat 3
ObsNum 33551, Chassis 1, Repeat 4
ObsNum 33551, Chassis 1, Averaged
ObsNum 33551, Chassis 2, Repeat 0
ObsNum 33551, Chassis 2, Repeat 1
ObsNum 33551, Chassis 2, Repeat 2
ObsNum 33551, Chassis 2, Repeat 3
ObsNum 33551, Chassis 2, Repeat 4
ObsNum 33551, Chassis 2, Averaged
ObsNum 33551, Chassis 3, Repeat 0
ObsNum 33551, Chassis 3, Repeat 1
ObsNum 33551, Chassis 3, Repeat 2
ObsNum 33551, Chassis 3, Repeat 3
ObsNum 33551, Chassis 3, Repeat 4
ObsNum 33551, Chassis 3, Averaged
20
0
−20
25
0
−25
75
80
85
90
95
100
105
110

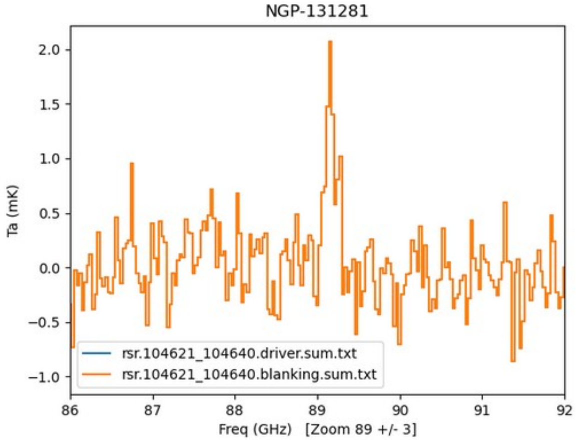
Detections?



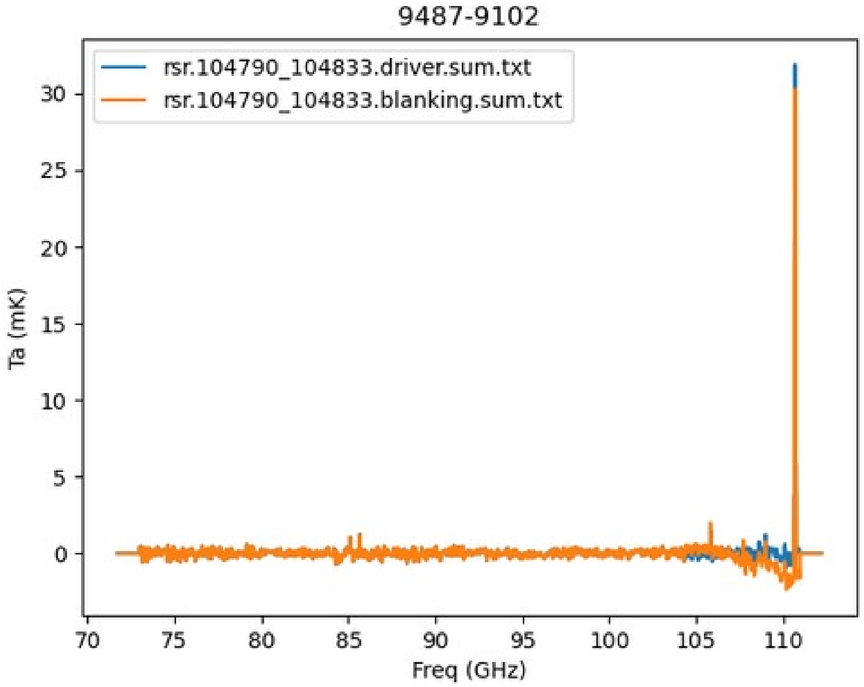


RMS = 0.34 mK
RMS0 = 0.17
Tint = 3.24 hrs
37 obsnums

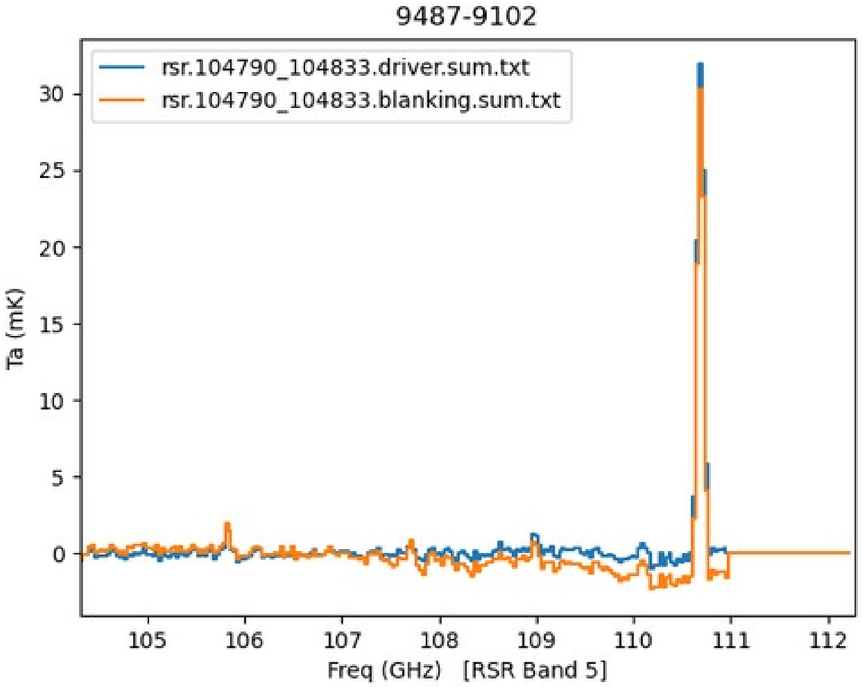




RMS = 0.25 mK
RMS0 = 0.26
Tint = 1.32 hrs
15 obsnums



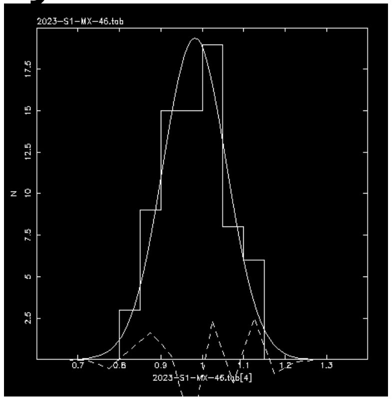
9487-9102
rsr.104790_104833.driver.sum.txt
rsr.104790_104833.blanking.sum.txt
Ta (mK)
Freq (GHz)



9487-9102
rsr.104790_104833.driver.sum.txt
rsr.104790_104833.blanking.sum.txt
Ta (mK)
Freq (GHz) [RSR Band 5]

QA checklist?   (M.Yun)

- Integration finished?   [yes/no]   < 100% always fails
- How many chassis?   [0..4]
- How many C/B's?   [0..24]
- Weather ok?   [yes/no]
- RMS noise ok?   [yes/no]   RMS/RMS0 < 1.4
- Additional flagging needed?   Add comments



Reducing RSR data

1. Use the scripts (badlags, rsr_driver etc.) - for black belt users (mostly **dreampy3**)
2. Use the pipeline (SLpipeline) – **LMTOY** needs to be installed   [PI parameters can be used from here on]
3. Use the script generator – useful to process many obsnums as well as stacking them (**lmtoy_run**)
4. Use the web interface [future]

PI-parameters for RSR
(and their defaults)

- **badcb**=”” # e.g. “1/1,2/4,3/5”
- **xlines**=”” # e.g. “109.1,0.3,105.2,0.2” – or use linecheck=1
- **linecheck**=0 # 0 or 1 (1 assumes source has the xlines)
- **bandzoom**=5 # pick a band (not board) 0..5
- **speczoom**=”” # e.g. “89,2” (overrides bandzoom)
- **rthr**=0.01 # threshold sigma for single obsnum repeats
- **cthr**=??? # threshold sigma for co-adding all observations
- **blo**=1 # baseline order
- **sgf**=0 # Savitzky-Golay high pass filter ; odd number > 21
- **notch**=0 # sigma cut for notch filter to eliminate large frecuency oscillations. Needs sgf > 21
- <more to come>

Questions?

Backup Slides

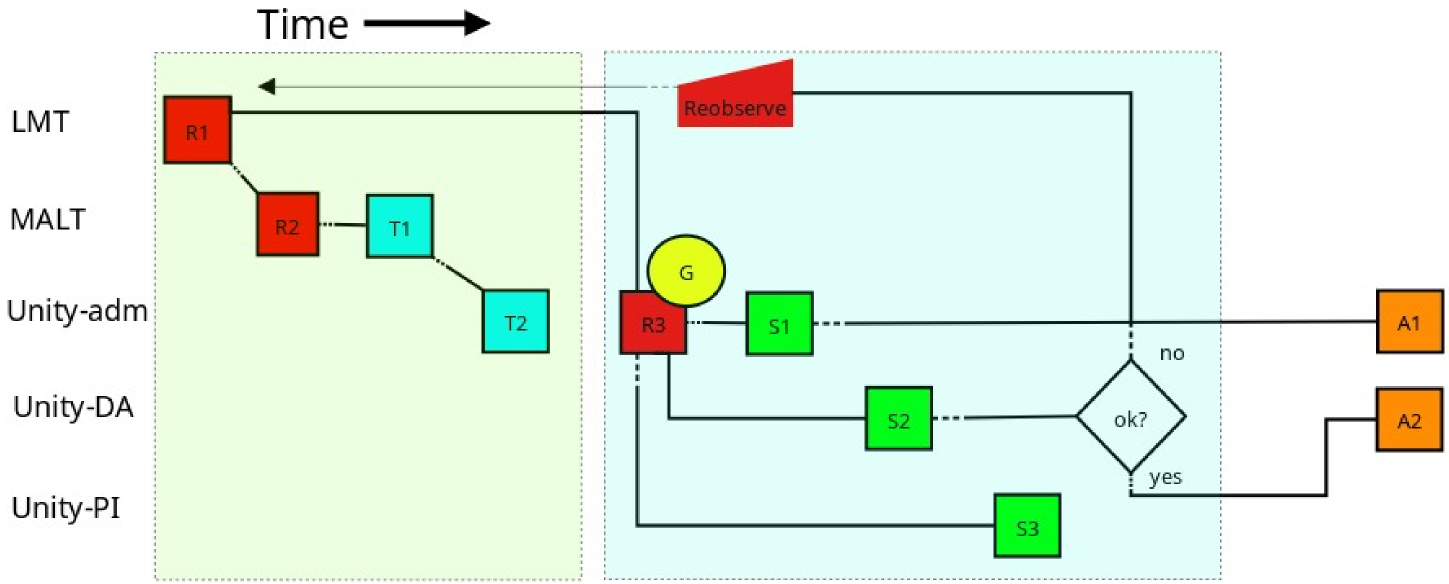
Future ?

- Collect **Pointing** and **LineCheck** with script generator?
- More PI parameters
  - Z=   : if set, it will draw CO ladder in the spectrum
- More columns in **summary.cvs**

RSR 2023 data (jan 16 --)

- Data taken in 2022 suffer from missing chassis 0 and numerous bad boards and lags. LineCheck seem ok, but no science data results yet.
  Early Jan 2023 RSR was revamped and all chassis working OK now, weather has been pretty darn good!
- Projects so far:
  - **2023S1RSRCommisioning**
    Many linechecks
  - **2023-S1-US-18**
    Source: HZ1, HZ4 (PI: Julia)
  - **2023-S1-MX-46**
    Source: G09-44907 etc. (PI: Arturo)
  - **2023-S1-MX-47**
    Source: 9487-9102 (PI: Castalia)
  - **2023-S1-MX-55**
    Source: ??? (PI: Abigail)
  - **2023-S1-UM-10**
    Source: many J*.* sources (PI: Min)

Data Flow LMT → Umass-Archive

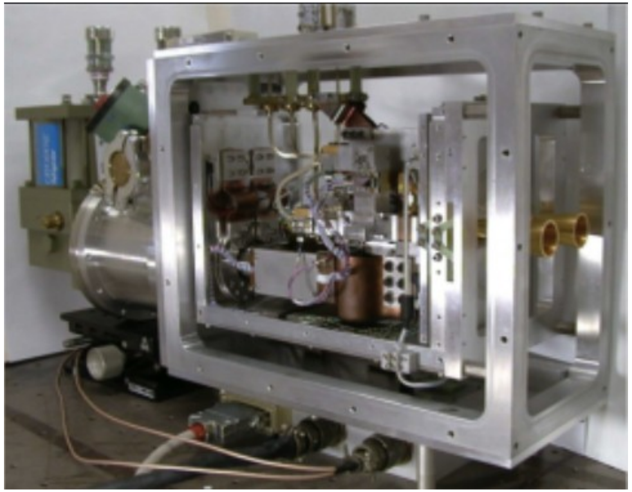


R = RAW
T = TAP
C = CAL (not shown)
S = SRDP (contains CAL)
G = github script generator
A = Archive (DataVerse)

Typical RSR workflow

- **cd** $WORK_LMT/2023-S1-US-18
- **SLpipeline.sh** obsnum=123456
  badcb=""
  xlines=""
  linecheck=0
  bandzoom=3
  …   <other PI parameters>

- **SLpipeline.sh** obsnums=123456,123457,123458
- **mk_summary1.sh** > README.html
- **xdg-open** ...



RSR: script generator

- Files:
  - **mk_runs.py**

    on[‘foo’] = [123456, 123457, 123458]
    pars1[‘foo’] = “”
    pars2[‘foo’] = “”
  - **comments.txt**

    123456   band5 waves   # badcb=3/4

Using the script generator

- *Unity shell*:
  make runs
  sbatch_lmtoy.sh *.run1a
  sbatch_lmtoy.sh *.run2a
  make summary

- *LMA shell*:
  parallel –jobs 32 < *.run1a
  parallel –jobs 32 < *.run2a

RSR: viewing results

- On malt (“secret”, during observing)
- On unity – TAP
  https://taps.lmtslr.org/.../2000-S0-AA-00/TAP
- On unity – stage 1 SRDP
- On unity – stage 2 SRDP

RSR dreampy3 based scripts

- badlags.py -
- rsr_tsys.py -
- rsr_driver.py -
- rsr_sum.py -
- rsr_spectra.py -
-

Radiometer output

- We collect:
  - RMS (mK; scales as 1/sqrt(tint))
  - RMS/RMS0 (at 100K, ideally near 1.0)
  - tau @ 220GHz (but not as function of time)
  -

RMS/RMS0

- Individual (5 min) OBSNUM often close to 1
- Combinations often larger (1.4-1.6)
  - Adding systematics
  - Total integration time < Sum integration times ?