Uu_RMS_Summary

August 11, 2021

1 Exploring the Uu RMS

1.0.1 Steve Mairs

In this notebook, I will use pandas to explore the csv files produced by the Uu RMS python code written by Mark Rawlings and David Berry which queries the database to find Uu observations, then compares the ITC-calculated RMS to the ORACDR-measured RMS.

1. CSV File 1: Uu_RMS_table_20200721_to_20210729_excluding_20210529_to_20210709.csv:

All "Good" observations in this time range, excluding data obtained between 2021-05-29 and 2021-07-09, when the 3DB PADs were removed and TSYS values were high.

2. CSV File 2: Uu_RMS_table_20200721_to_20210729.csv:

All "Good" observations in this time range.

These CSV files come from Mark Rawlings/David Berry's code (with my own modifications): Uu_RMS_comparison_v5.py

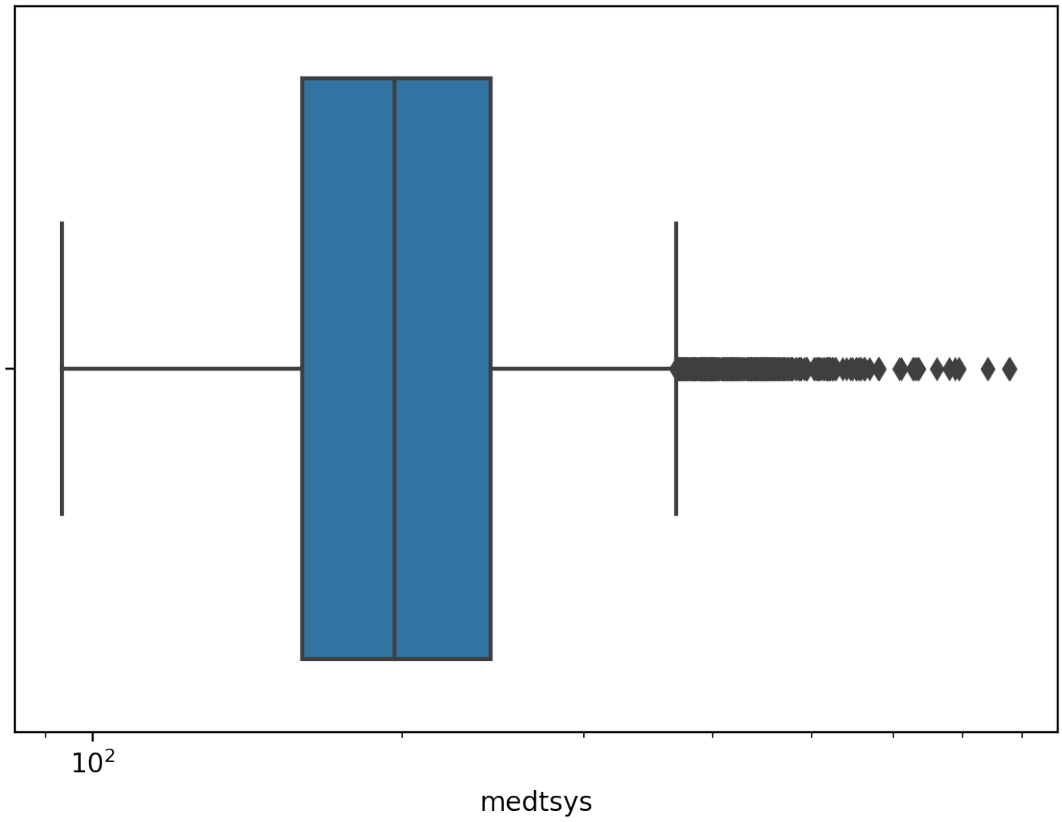
1.1 User Inputs (Code Only - Removed from PDF)

```
/Users/smairs/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

1.2 Remove Bad Tsys

Right off the bat, lets just remove bad Tsys values.

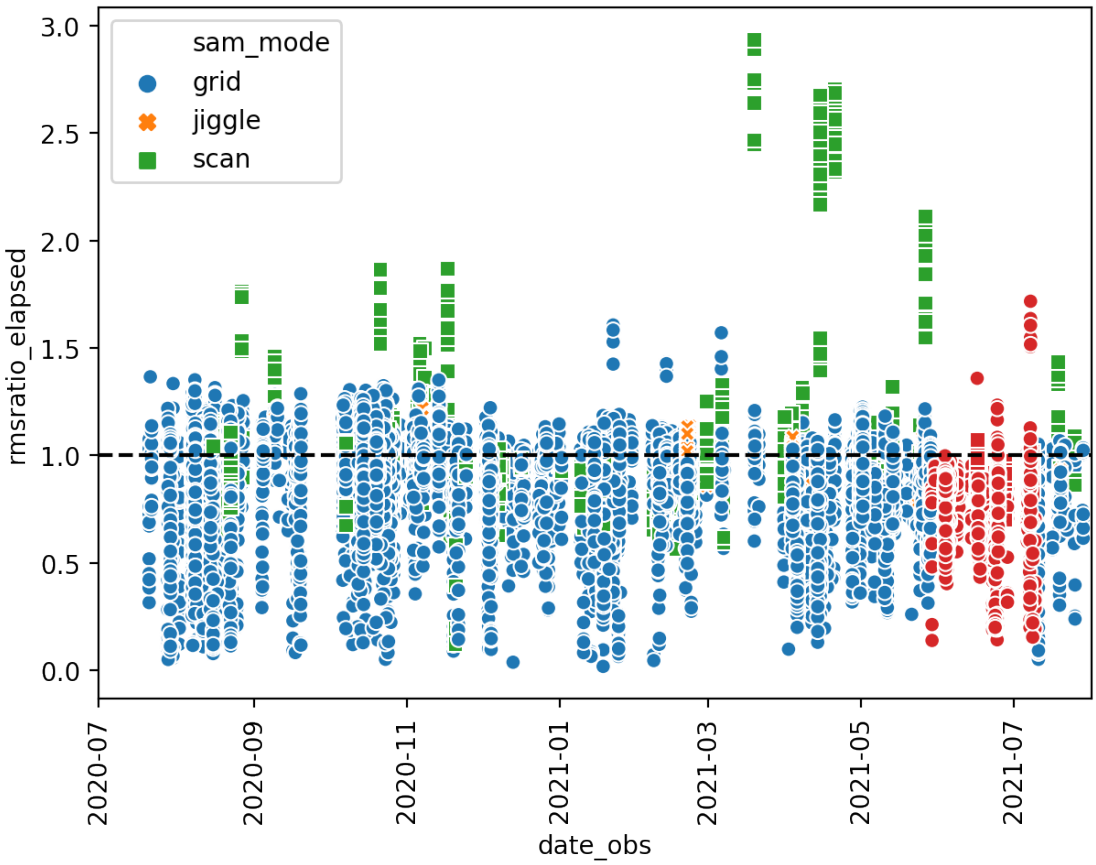



```
<Figure size 1400x1000 with 0 Axes>
```

From the figure, above, it seems like 1.5* the 75th percentile extends to about **medtsys = 400 K**. So we'll draw our cutoff there.

1.3 RMS Ratio Over Time Grouped By Observing Mode

"Bad Dates" are plotted in Red.



1.3.1 Bin the data now by month so trends are easier to see – use the median value when grouping by month

First, we will make a separate dataframe for each sam_mode so we can work with them individually (CODE REMOVED FROM PDF)

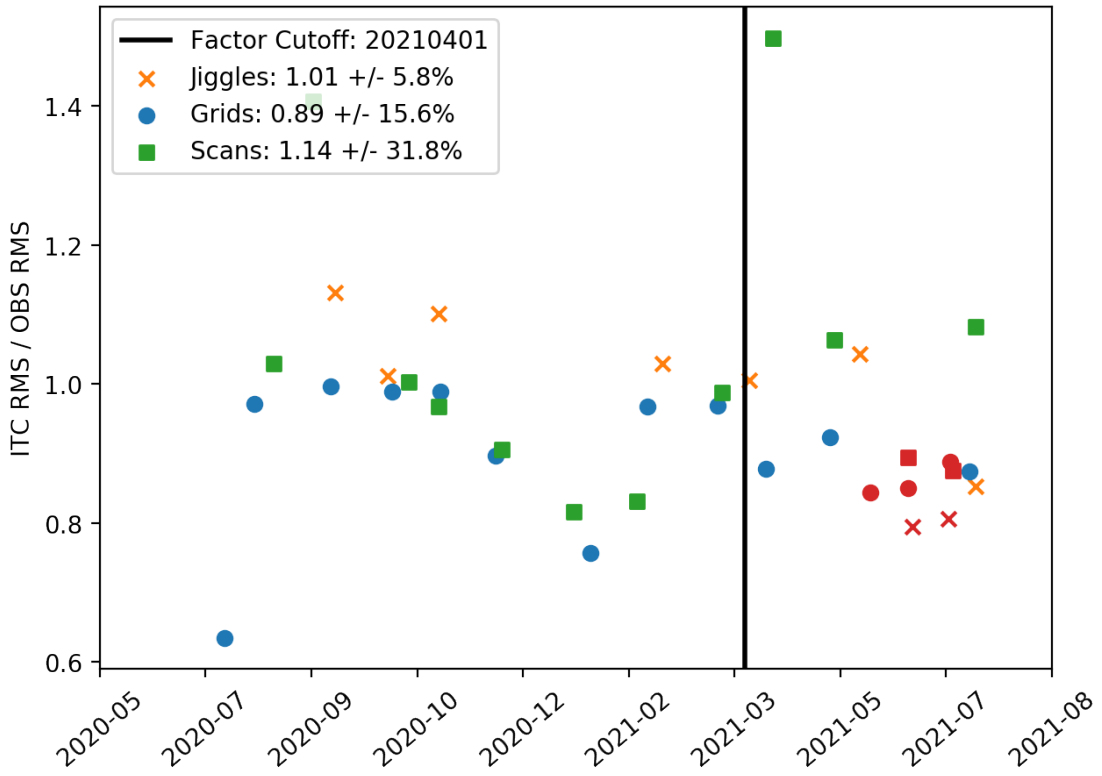
Next, we will group the data by month for each sam_mode (CODE REMOVED FROM PDF)

Of course, now we don't have date_obs in our key - but we still have UT date - so we can plot by the median UT date. To make this look a little better though, we will first convert to MJD so we don't have big gaps in the integers (CODE REMOVED FROM PDF)

Next, calculate the median rms ratios (the *fudge factors*) since the fudge_factor_cutoff, excluding the bad dates, for each sam_mode

(CODE REMOVED FROM PDF)

Next we can make a plot with pyplot! The fudge factors are given in the legend and also below.




```
<Figure size 1400x1000 with 0 Axes>
```

We see a 10% decrease in scan value during the 3DB PAD removal

1.4 Fudge Factors Calculated since 2021-04-01

```
Jiggles: 1.01 +/- 5.8%
Grids  : 0.89 +/- 15.6%
Scans  : 1.14 +/- 31.8%
```

Fudge Factor = ITC RMS / OBS RMS **Jiggles**: 1.01 +/- 5.8%

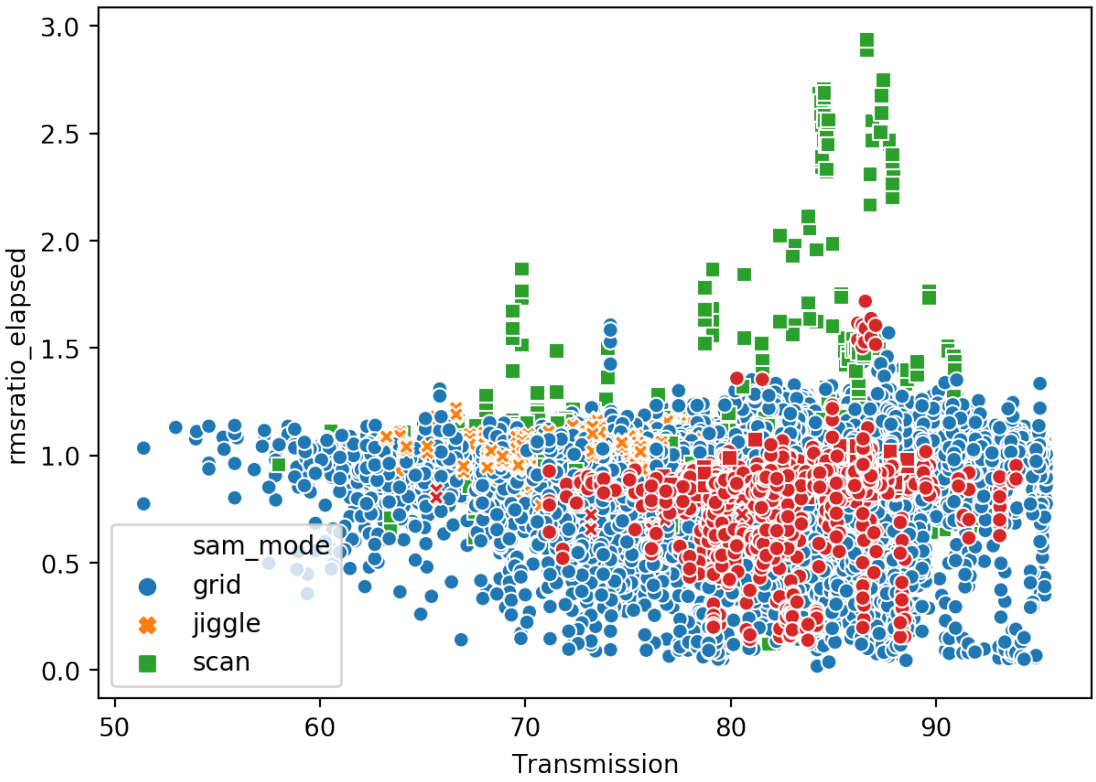
Grids : 0.89 +/- 15.6%

Scans : 1.14 +/- 31.8%

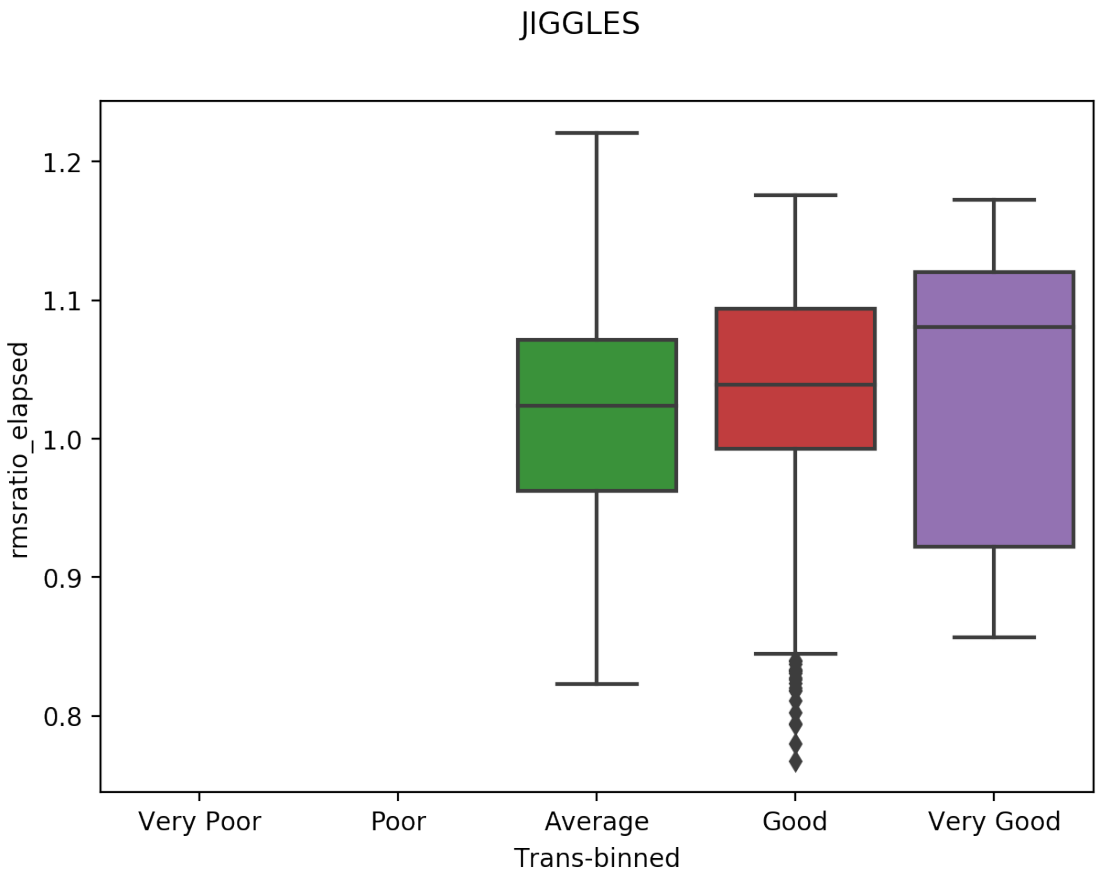
Calculated by the median *rmsratio* value since 2021-04-01 (excluding 2021-05-29 to 2021-07-09). Errors are the median absolute deviation.

1.5 Transmission

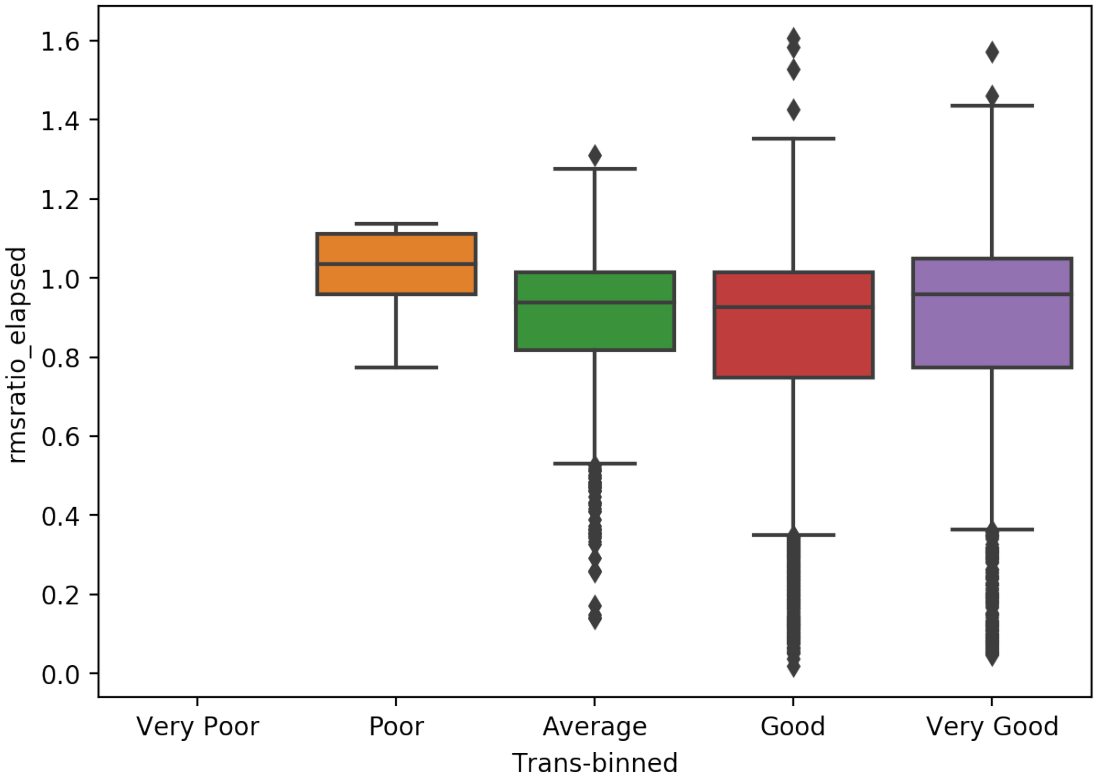
Now we'll take a look at the *rmsratio* as a function of atmospheric transmission.



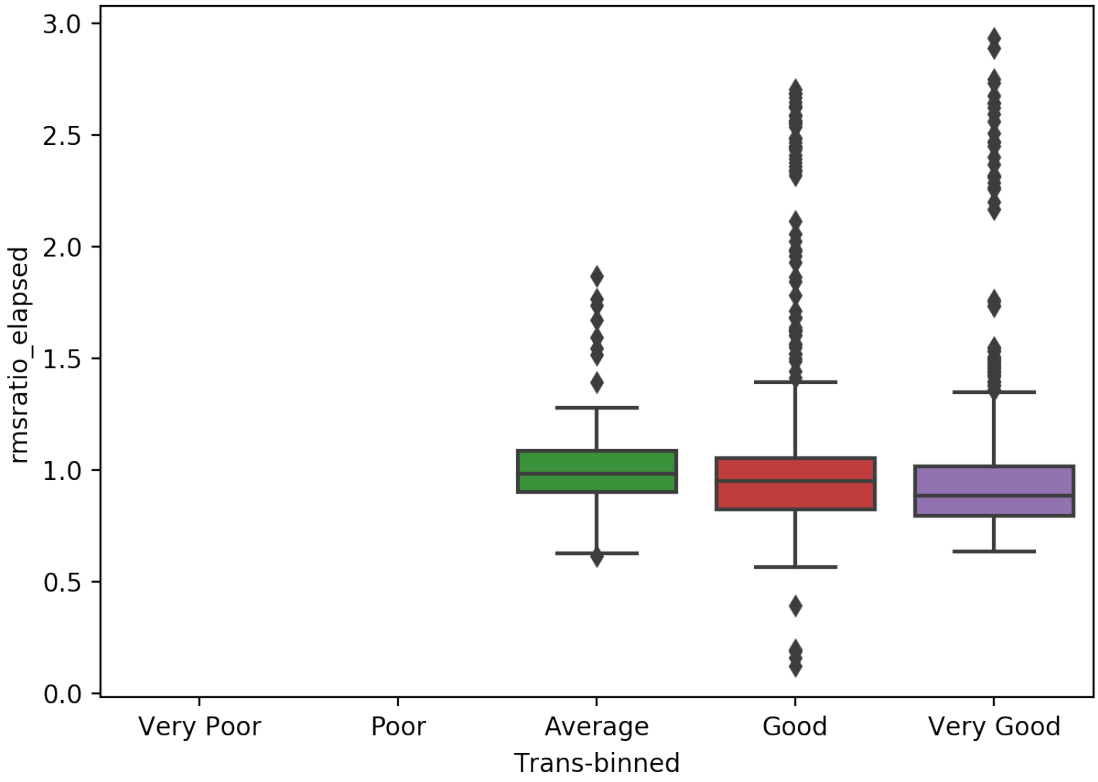
Separate this out by binning the transmission into 5 weather bins and plot by *sam_mode*



<Figure size 1400x1000 with 0 Axes>

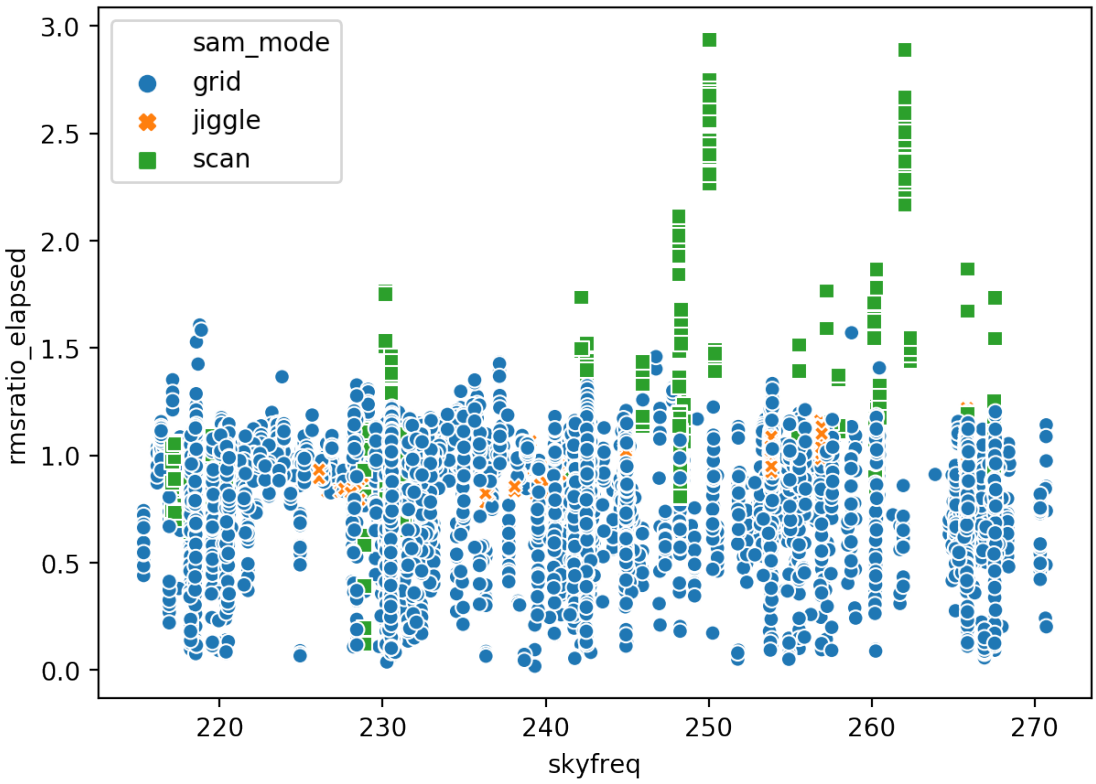



```
<Figure size 1400x1000 with 0 Axes>
```

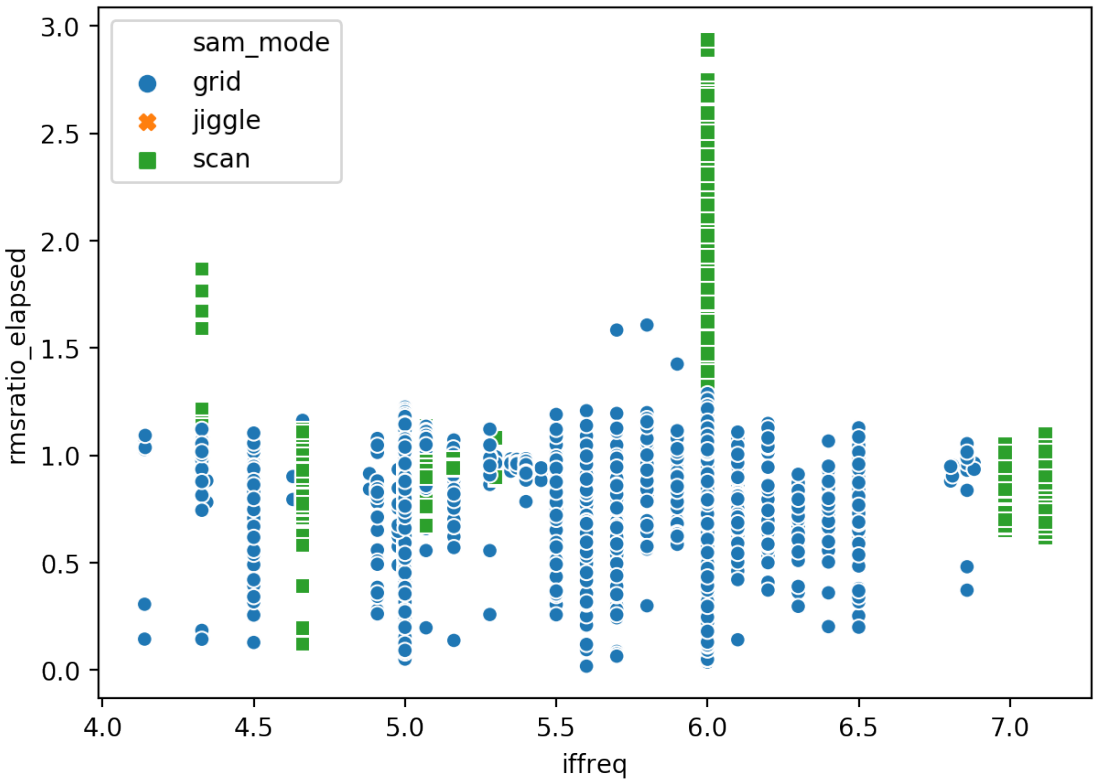



```
<Figure size 1400x1000 with 0 Axes>
```

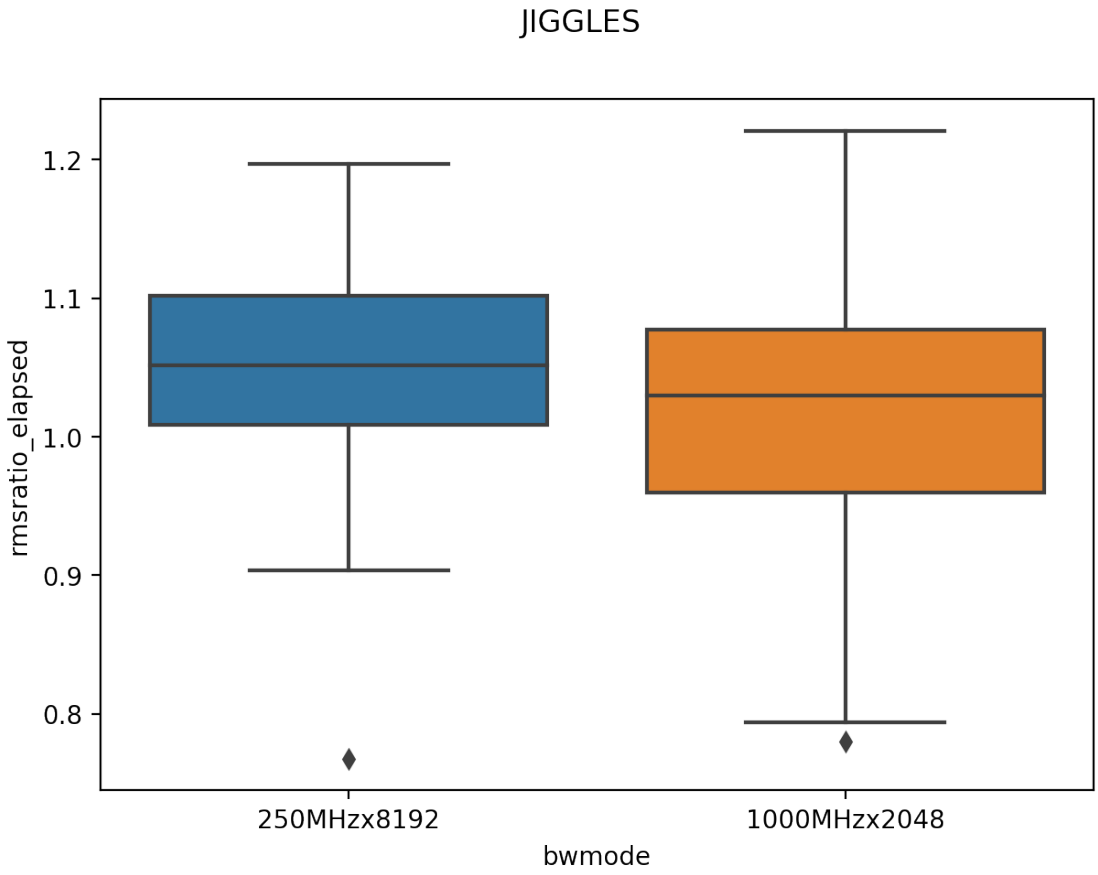
1.6 Plot by Sky Frequency



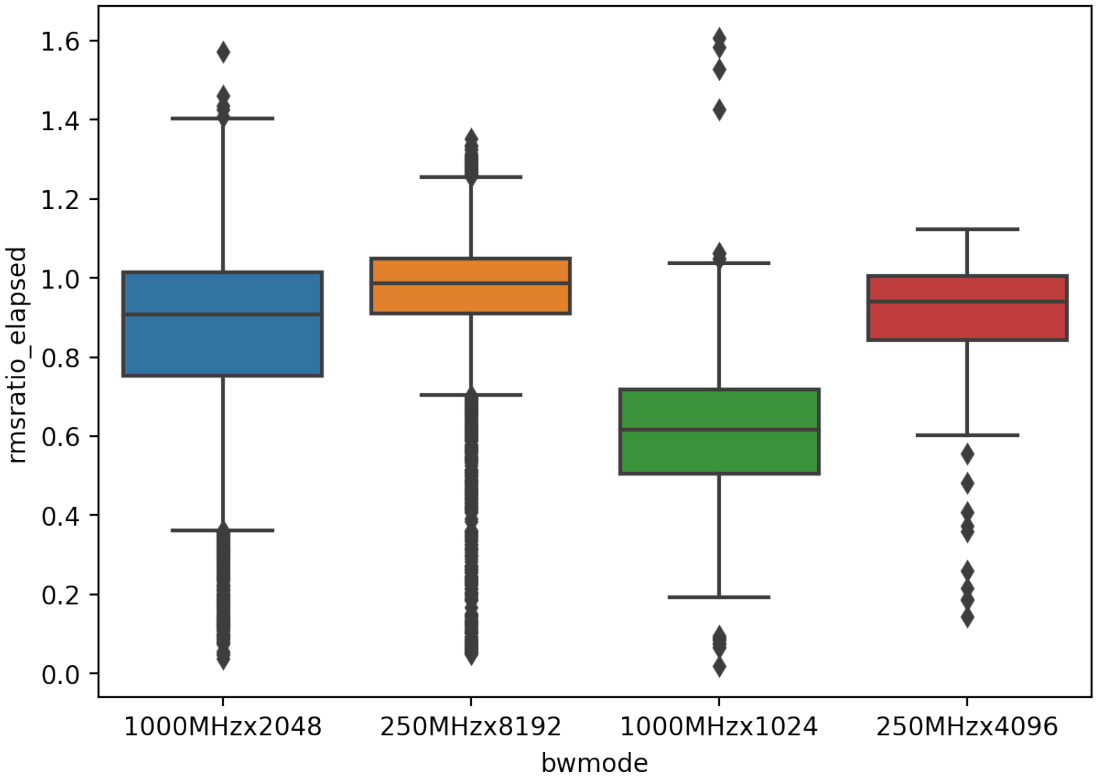
1.7 Plot by IF Freq



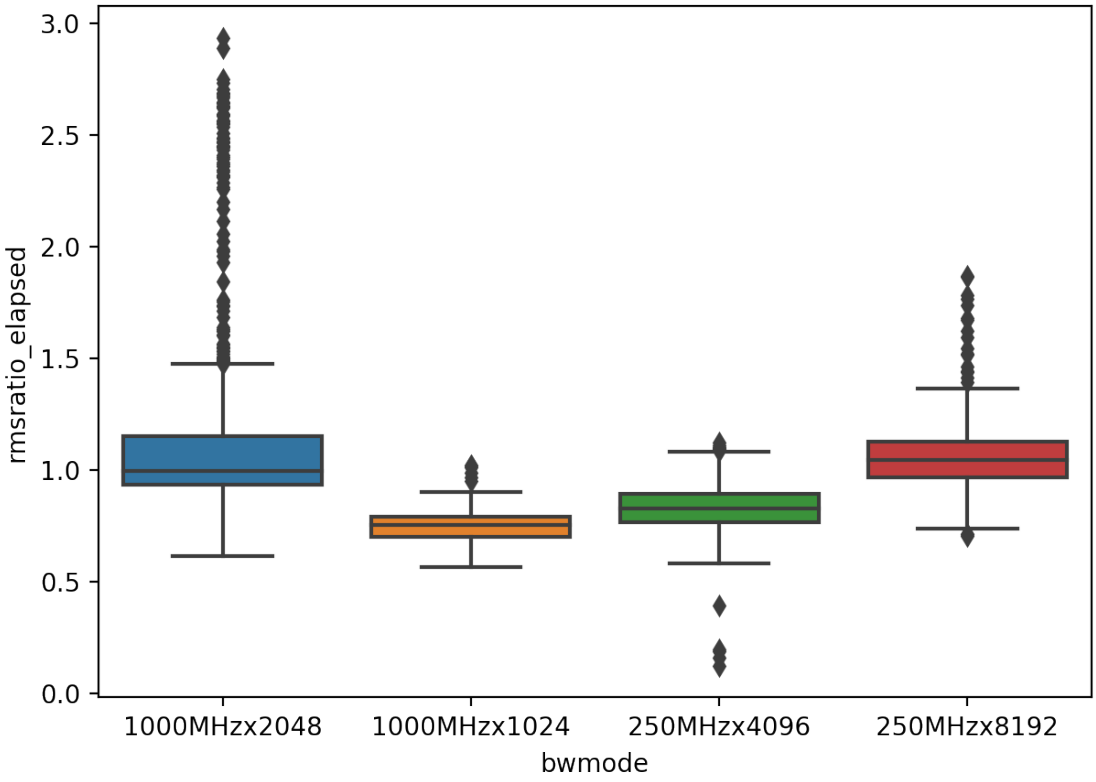
1.8 Plot by BW Mode




```
<Figure size 1400x1000 with 0 Axes>
```



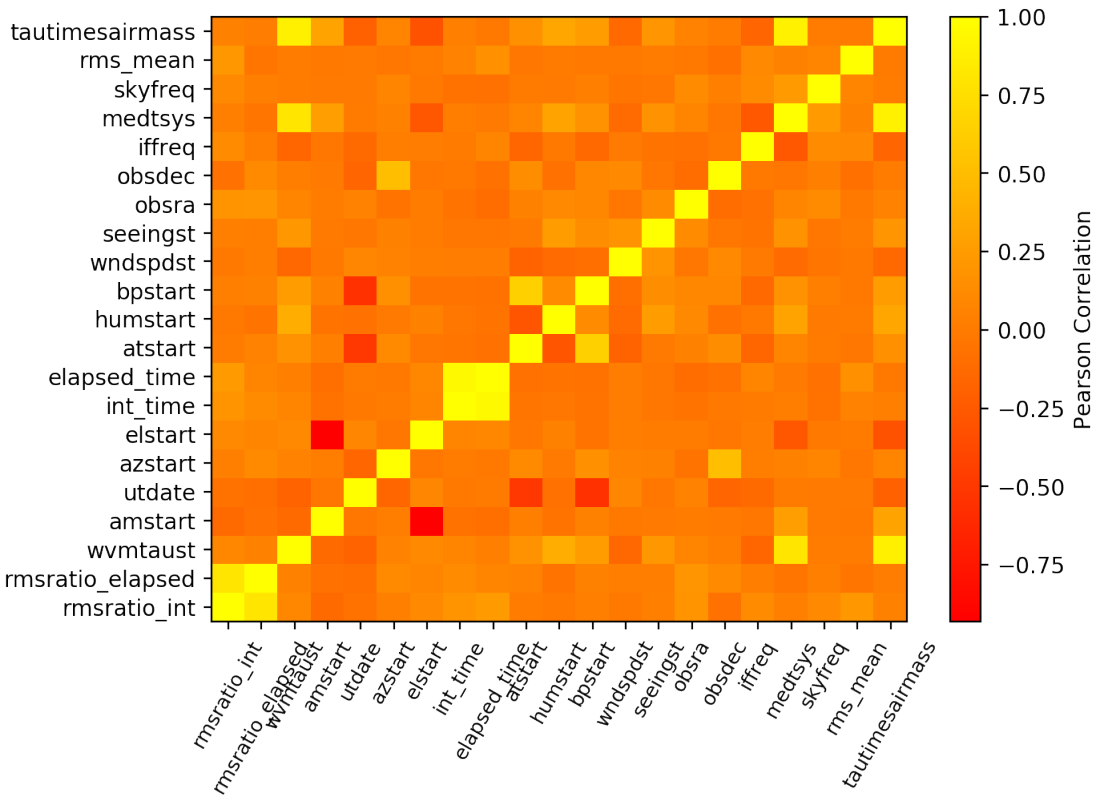

```
<Figure size 1400x1000 with 0 Axes>
```




```
<Figure size 1400x1000 with 0 Axes>
```

1.9 Correlation Heatmap of the most important features for rmsratio

The graph below shows the pearson r correlation between parameters. Along the rmsratio row - nothing in particular sticks out as having a strong trend other than perhaps Airmass (as gauged quickly by a linear fit).



<Figure size 1400x1000 with 0 Axes>