06_calib.steffl

November 2, 2021

[1]:
```
# default_exp calib.steffl
```

1 Recreating Steffl's Calib

Based on his thesis appendix

[2]:
```
# export
import numpy as np

import holoviews as hv
import hvplot.xarray
import pandas as pd
from nbverbose.showdoc import show_doc
from planetarypy.utils import iso_to_nasa_date
from pyuvis.calib.greg import filter_spica_for_date
from pyuvis.io import UVPDS, UVISObs
from pyuvis.pds import CatalogFilter

hv.extension("bokeh")
```

[3]:
```
missing = []
there = []
for id in obsids:
    try:
        data = UVPDS(id, skip_download=True)
    except FileNotFoundError:
        print(id, "not there.")
        missing.append(id)
    else:
        print("Got", id)
        there.append(id)
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)
```

```
/tmp/ipykernel_143809/2093082731.py in <module>
  1 missing = []
  2 there = []
----> 3 for id in obsids:
  4     try:
  5         data = UVPDS(id, skip_download=True)


NameError: name 'obsids' is not defined
```

[3]:
```
# export
steffl_spica_dates = ["2001-04-3", "2002-07-17", "2003-05-19"]
steffl_spica_nasa_dates = [iso_to_nasa_date(i) for i in steffl_spica_dates]
steffl_spica_nasa_dates
```

[3]:
```
['2001-093', '2002-198', '2003-139']
```

[4]:
```
cat = CatalogFilter(steffl_spica_dates[2])
```

```
Stored index is up-to-date.
```

[5]:
```
cat.date = "2003-05-19"
```

[6]:
```
pids = list(cat.get_euv_date().query("OBSERVATION_TYPE=='CALIB'").index)
```

[7]:
```
pids
```

[7]:
```
['EUV2003_139_18_11',
 'EUV2003_139_19_07',
 'EUV2003_139_20_03',
 'EUV2003_139_21_00',
 'EUV2003_139_21_56',
 'EUV2003_139_22_52',
 'EUV2003_139_23_48']
```

[8]:
```
cat.set_next_day()
```

[9]:
```
pids.extend(list(cat.get_euv_date().query("OBSERVATION_TYPE=='CALIB'").index))
```

[10]:
```
pids
```

[10]:
```
['EUV2003_139_18_11',
 'EUV2003_139_19_07',
 'EUV2003_139_20_03',
 'EUV2003_139_21_00',
 'EUV2003_139_21_56',
 'EUV2003_139_22_52',
```

```
 'EUV2003_139_23_48',
 'EUV2003_140_00_44',
 'EUV2003_140_01_41',
 'EUV2003_140_02_37',
 'EUV2003_140_03_33',
 'EUV2003_140_04_29',
 'EUV2003_140_05_25',
 'EUV2003_140_06_22']
```

[11]:
```
kwargs = {"x": "nx", "y": "ny", "cmap": "viridis", "clim": (0, 50)}
```

1.1 Column-averaging

[12]:
```
class FlatFielder:
    def __init__(self, pid):
        self.pid = pid
        self.data = UVPDS(pid).xarray.astype("int16")

    @property
    def plot_set(self):
        return self.data.hvplot(
            x="spectral", y="spatial", cmap="viridis", title=self.pid
        )

    @property
    def integrated(self):
        return self.data.sum(dim="samples")

    @property
    def plot_integrated(self):
        return self.integrated.hvplot(
            x="spectral", y="spatial", cmap="viridis", title=self.pid
        )

    @property
    def averaged(self):
        return self.integrated.sel(spatial=slice(3, 61)).mean(dim="spatial")

    @property
    def plot_averaged(self):
        return self.averaged.hvplot(x="spectral", title=self.pid)

    @property
    def column_std(self):
        return self.integrated.sel(spatial=slice(2, 60)).std(dim="spatial")

    @property
```

```
    def plot_column_std(self):
        return self.column_std.hvplot(x="spectral", title=f"{self.pid}, Column␣
↪STD")

    @property
    def ff(self):
        return self.integrated / self.averaged

    @property
    def plot_ff(self):
        return self.ff.hvplot(x="spectral", y="spatial", cmap="viridis",␣
↪title=self.pid)
```

[13]:
```
len(pids)
```

[13]:
```
14
```

[14]:
```
pids
```

[14]:
```
['EUV2003_139_18_11',
 'EUV2003_139_19_07',
 'EUV2003_139_20_03',
 'EUV2003_139_21_00',
 'EUV2003_139_21_56',
 'EUV2003_139_22_52',
 'EUV2003_139_23_48',
 'EUV2003_140_00_44',
 'EUV2003_140_01_41',
 'EUV2003_140_02_37',
 'EUV2003_140_03_33',
 'EUV2003_140_04_29',
 'EUV2003_140_05_25',
 'EUV2003_140_06_22']
```

[15]:
```
flatter = FlatFielder(pids[0])
```

[16]:
```
flatter.plot_set
```

[16]:
```
:DynamicMap   [samples]
   :Image   [spectral,spatial]   (EUV2003_139_18_11)
```

[17]:
```
flatter.plot_integrated
```

[17]:
```
:Image   [spectral,spatial]   (EUV2003_139_18_11)
```

[18]:
```
flatter.plot_column_std
```

[18]:
```
:Curve   [spectral]   (EUV2003_139_18_11)
```

[19]:
```
flatter.integrated.hvplot(x="spatial")
```

[19]:
```
:DynamicMap   [spectral]
   :Curve   [spatial]   (EUV2003_139_18_11)
```

[20]:
```
flatter.plot_ff
```

[20]:
```
:Image   [spectral,spatial]   (EUV2003_139_18_11)
```

2 Steffl Calib class

Putting above together into a class

[31]:
```
# export
class SteffiCalib:
    def __init__(
        self,
        pids,  # group of product ids for a raster run
        i=15,  # Minimum column value for evaluation (15:997)
        m=0,  # Default start scan
    ):
        self.pids = pids
        self.i = i
        self.m = m
        scan_df = pd.DataFrame({"pids": pids})
        scan_df.index.name = "m"

        stacked = []
        for m, pid in scan_df.iterrows():
            flatter = FlatFielder(pid.get(0))
            stacked.append(flatter.integrated)

        self.stacked = np.dstack(stacked)

        arr = xr.DataArray(
            self.stacked,
            dims=["spectral", "spatial", "scan"],
            coords={
                "scan": scan_df.index.values,
                "spectral": flatter.integrated.spectral,
                "spatial": flatter.integrated.spatial,
            },
        )
        arr.name = "scan_stack"
        self.arr = arr
        self.corrections = np.ones((64, 1024, 5))
```

```
    @property
    def i(self):
        return self._i

    @i.setter
    def i(self, value):
        if value < 15 or value > 997:
            raise ValueError("Column i should be within 15:997 per Steffl␣
↪Calib")
        self._i = value

    @property
    def m(self):
        return self._m

    @m.setter
    def m(self, value):
        self._m = value

    @property
    def current_column_set(self):
        return [self.i, self.i + 4, self.i + 8]

    @property
    def current_scan_set(self):
        return [self.m, self.m + 5, self.m + 10]

    def plot(self):
        return self.arr.hvplot(
            x="spectral",
            y="spatial",
            cmap="viridis",
            clim=(1, 6000),
            widget_type="scrubber",
            widget_location="bottom",
        )

    def get_triplet_data(self):
        cols = []
        for col, scan in zip(self.current_column_set, self.current_scan_set):
            cols.append(self.arr.isel(spectral=col, scan=scan))
        return cols

    def get_averaged_triplet(self):
        cols = self.get_triplet_data()
        colstacked = np.stack([col.data for col in cols])
        return colstacked.mean(axis=0)
```

```
    def plot_averaged_triplet(self):
        plt.plot(self.get_averaged_triplet(), label=f"{self.i=}, {self.m=}")
        plt.xlabel("Spatial axis")
        plt.title("Corrections for column i")
        plt.legend()

    def plot_triplet(self):
        cols = self.get_triplet_data()
        plots = []
        for col, col_number in zip(cols, self.current_column_set):
            plots.append(col.hvplot(label=f"Columns {col_number}"))
        return hv.Overlay(plots)

    def plot_corrections(self, m=0):
        return hv.Raster(self.corrections[:, :, m]).opts(
            colorbar=True, tools=["hover"], width=500, clim=(None, 2)
        )
```

[32]:
```
steffl = StefflCalib(pids)
```

[33]:
```
pids
```

[33]:
```
['EUV2003_139_18_11',
 'EUV2003_139_19_07',
 'EUV2003_139_20_03',
 'EUV2003_139_21_00',
 'EUV2003_139_21_56',
 'EUV2003_139_22_52',
 'EUV2003_139_23_48',
 'EUV2003_140_00_44',
 'EUV2003_140_01_41',
 'EUV2003_140_02_37',
 'EUV2003_140_03_33',
 'EUV2003_140_04_29',
 'EUV2003_140_05_25',
 'EUV2003_140_06_22']
```

[34]:
```
for m in range(4):
    steffl.m = m
    for i in range(15, 998):
        steffl.i = i
        steffl.corrections[:, i, m] = steffl.get_averaged_triplet() / steffl.
↪get_triplet_data()[0]
```

```
/home/maye/miniconda3/envs/py38/lib/python3.8/site-
packages/xarray/core/computation.py:742: RuntimeWarning: divide by zero
encountered in true_divide
```

```
  result_data = func(*input_data)
/home/maye/miniconda3/envs/py38/lib/python3.8/site-
packages/xarray/core/computation.py:742: RuntimeWarning: invalid value
encountered in true_divide
  result_data = func(*input_data)
```

[35]:
```
steffl.plot_corrections(0)
```

[35]:
```
:Raster   [x,y]   (z)
```

[36]:
```
steffl.plot_corrections(1)
```

[36]:
```
:Raster   [x,y]   (z)
```

[37]:
```
steffl.corrections.shape
```

[37]:
```
(64, 1024, 5)
```

[38]:
```
steffl.plot_triplet()
```

[38]:
```
:Overlay
   .Curve.Columns_997  :Curve   [spatial]   (scan_stack)
   .Curve.Columns_1001 :Curve   [spatial]   (scan_stack)
   .Curve.Columns_1005 :Curve   [spatial]   (scan_stack)
```

[40]:
```
steffl.get_averaged_triplet()
```

[40]:
```
array([0.00000000e+00, 0.00000000e+00, 1.03666667e+03, 2.44133333e+03,
       2.29500000e+03, 1.98833333e+03, 1.78433333e+03, 2.38500000e+03,
       2.24466667e+03, 1.91800000e+03, 2.22333333e+03, 2.17300000e+03,
       1.81233333e+03, 1.92266667e+03, 2.09233333e+03, 2.54666667e+03,
       2.03866667e+03, 2.56266667e+03, 2.21300000e+03, 1.91233333e+03,
       2.12633333e+03, 1.71133333e+03, 2.68400000e+03, 1.84933333e+03,
       2.63100000e+03, 2.06766667e+03, 2.64400000e+03, 1.92766667e+03,
       2.36400000e+03, 1.99966667e+03, 1.75333333e+03, 2.37766667e+03,
       2.27800000e+03, 1.64400000e+03, 2.56766667e+03, 2.60333333e+03,
       1.71633333e+03, 2.01800000e+03, 2.49433333e+03, 1.96033333e+03,
       2.62033333e+03, 2.24366667e+03, 2.15033333e+03, 2.63866667e+03,
       2.43900000e+03, 2.57533333e+03, 2.66033333e+03, 1.85633333e+03,
       1.98533333e+03, 2.56100000e+03, 2.50233333e+03, 2.69100000e+03,
       2.25366667e+03, 2.29566667e+03, 2.81633333e+03, 2.42533333e+03,
       1.98266667e+03, 2.72366667e+03, 2.34466667e+03, 2.18833333e+03,
       2.51800000e+03, 2.73833333e+03, 2.07333333e+02, 1.33333333e+00])
```

[56]:
```

```

[56]:
```
<xarray.DataArray 'scan_stack' (spectral: 1024, spatial: 64, scan: 14)>
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 … 1 4 3 2 7 0 0 0 0 1 0 0 1 1 0 0 0 0 1
```

```
Coordinates:
  * scan      (scan) int64 0 1 2 3 4 5 6 7 8 9 10 11 12 13
  * spectral  (spectral) float64 111.5 111.6 111.7 111.7 … 189.8 189.9 190.0
  * spatial   (spatial) int64 0 1 2 3 4 5 6 7 8 9 … 55 56 57 58 59 60 61 62 63
```

[35]:
```
steffl.plot()
```

[35]:
```
Column
    [0] HoloViews(DynamicMap, widget_location='bottom', widget_type='scrubber')
    [1] Row
        [0] HSpacer()
        [1] WidgetBox
            [0] Player(end=13, width=550)
        [2] HSpacer()
```

[43]:
```
class Col2Col:
    def __init__(self, pids, i=15, m=0):  # set of product_ids
        self.pids = pids
        self.i = i
        self.m = 0
        scan_df = pd.DataFrame({"pids": pids})
        scan_df.index.name = "m"

        stacked = []
        for m, pid in scan_df.iterrows():
            flatter = FlatFielder(pid.get(0))
            stacked.append(flatter.integrated)
        stack = np.dstack(stacked)
        arr = xr.DataArray(
            stack,
            dims=["spectral", "spatial", "scan"],
            coords={
                "scan": scan_df.index.values,
                "spectral": flatter.integrated.spectral,
                "spatial": flatter.integrated.spatial,
            },
        )
        arr.name = "scan_stack"
        self.arr = arr

    @property
    def triple_columns(self):
        return [self.i, self.i + 4, self.i + 8]

    @property
    def get_triplet(self, i=15, m=0):
        cols = []
```

```
        for col, scan in zip([0, 4, 8], [0, 5, 10]):
            cols.append(self.arr.isel(spectral=i + col, scan=m + scan))
        plots = []
        for col, col_number in zip(cols, [i, i + 4, i + 8]):
            plots.append(col.hvplot(label=f"Columns {col_number}, Scan {m}"))
        return hv.Overlay(plots)
```

[44]:
```
col2col = Col2Col(pids)
```

[45]:
```
col2col.triple_columns
```

[45]:
```
[15, 19, 23]
```

[205]:
```
get_triplet()
```

[205]:
```
:Overlay
   .Curve.Columns_15_comma_Scan_0 :Curve   [spatial]   (scan_stack)
   .Curve.Columns_19_comma_Scan_0 :Curve   [spatial]   (scan_stack)
   .Curve.Columns_23_comma_Scan_0 :Curve   [spatial]   (scan_stack)
```

[198]:
```
import panel as pn

pn.extension()

pn.interact(get_triplet)
```

[198]:
```
Column
    [0] Column
        [0] IntSlider(end=45, name='i', start=-15, value=15)
        [1] IntSlider(name='m')
    [1] Row
        [0] HoloViews(Overlay, name='interactive67141')
```

[209]:
```
def get_all_triplets(arr, i=0):
    triplets = []
    for m in range(4):
        plots = get_triplet(arr, i, m)
        triplets.append(hv.Overlay(plots))
    return triplets
```

[210]:
```
triplets = get_all_triplets(arr, 15)
```

[211]:
```
hv.Layout(triplets).cols(2)
```

[211]:
```
:Layout
   .Overlay.I   :Overlay
      .Curve.Columns_15_comma_Scan_0 :Curve   [spatial]   (scan_stack)
      .Curve.Columns_19_comma_Scan_0 :Curve   [spatial]   (scan_stack)
```

```
   .Curve.Columns_23_comma_Scan_0 :Curve   [spatial]   (scan_stack)
.Overlay.II  :Overlay
   .Curve.Columns_15_comma_Scan_1 :Curve   [spatial]   (scan_stack)
   .Curve.Columns_19_comma_Scan_1 :Curve   [spatial]   (scan_stack)
   .Curve.Columns_23_comma_Scan_1 :Curve   [spatial]   (scan_stack)
.Overlay.III :Overlay
   .Curve.Columns_15_comma_Scan_2 :Curve   [spatial]   (scan_stack)
   .Curve.Columns_19_comma_Scan_2 :Curve   [spatial]   (scan_stack)
   .Curve.Columns_23_comma_Scan_2 :Curve   [spatial]   (scan_stack)
.Overlay.IV  :Overlay
   .Curve.Columns_15_comma_Scan_3 :Curve   [spatial]   (scan_stack)
   .Curve.Columns_19_comma_Scan_3 :Curve   [spatial]   (scan_stack)
   .Curve.Columns_23_comma_Scan_3 :Curve   [spatial]   (scan_stack)
```

[166]:
```
col1 = arr.isel(spectral=i + 4, scan=m + 5)
col2 = arr.isel(spectral=i + 8, scan=m + 10)
```

[167]:
```
(
    col0.hvplot(label=f"Scan {col0.scan.data}")
    * col1.hvplot(label=f"Scan {col1.scan.data}")
    * col2.hvplot(label=f"Scan {col2.scan.data}")
).opts(title=f"Column {i}")
```

[167]:
```
:Overlay
   .Curve.Scan_0  :Curve   [spatial]   (scan_stack)
   .Curve.Scan_5  :Curve   [spatial]   (scan_stack)
   .Curve.Scan_10 :Curve   [spatial]   (scan_stack)
```

[ ]:

[ ]:

[128]:
```
((col0 + col1 + col2) / 3).hvplot()
```

[128]:
```
:Curve   [spatial]   (scan_stack)
```

[ ]:

[120]:
```
col_set.mean(dim=["scan", "spectral"]).hvplot()
```

[120]:
```
:Curve   [spatial]   (scan_stack)
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[165]:
```
archive_df.loc["EUV2001_093_08_35_28"]
```

[165]:
```
path    /home/maye/uvis_archive/observations/EUV2001_0…
det                                                  EUV
Name: EUV2001_093_08_35_28, dtype: object
```

[ ]:

[6]:
```
import hvplot.xarray
import xarray as xr
```

[7]:
```
ds = xr.open_dataset(fname)
ds
```

[7]:
```
<xarray.Dataset>
Dimensions:   (integrations: 54, spatial_dim_0: 1, spectral_dim_0: 1024)
Dimensions without coordinates: integrations, spatial_dim_0, spectral_dim_0
Data variables:
    window_0  (integrations, spatial_dim_0, spectral_dim_0) int16 …
Attributes: (12/16)
    windows:                1.0
    compression:            0
    odc_id:                 7
    integration:            32
    channel:                EUV
    hvps_level:             0
    …                       …
    stop_time:
    Version:                1
    start_time_str:         1999-007 17:05:02.000 (1999-Jan-07) SCClock=(12944…
    SCTime:                 1294420183
    SCTimeFine:             0
    NetCDFWriter Version:   1.0
```

[8]:
```
np.percentile(ds.window_0, (5, 95))
```

[8]:
```
array([  0., 201.])
```

[9]:
```
ds.window_0.hvplot.image(
    x="spectral_dim_0", y="integrations", cmap="viridis", clim=(0, 201)
```

```
)
```

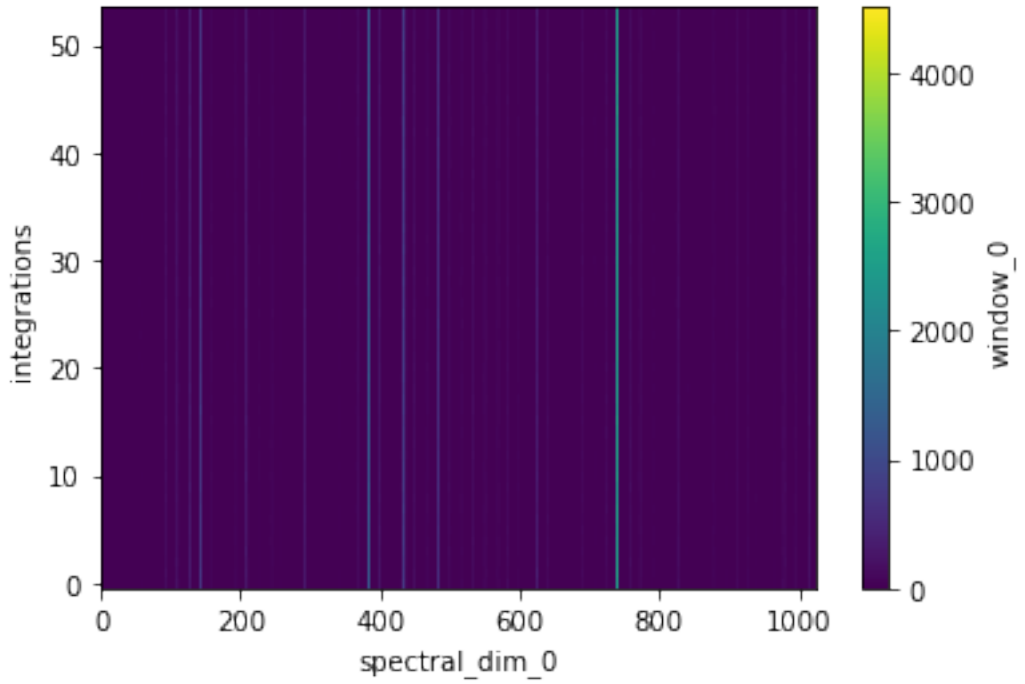
[9]: :Image [spectral_dim_0,integrations] (window_0)

[10]:
```
ds.window_0.mean("integrations").hvplot(x="spectral_dim_0")
```

[10]: :Curve [spectral_dim_0] (window_0)

[11]:
```
ds.window_0.plot()
```

[11]: <matplotlib.collections.QuadMesh at 0x7f01a1bb1790>



[115]:
```
p = obsdir / "index_repaired.tab"
```

[121]:
```
df = pd.read_csv(p, quotechar='"', skipinitialspace=True)
```

```
/home/maye/miniconda3/envs/py38/lib/python3.8/site-
packages/IPython/core/interactiveshell.py:3441: DtypeWarning: Columns
(0,9,10,11,12,13) have mixed types.Specify dtype option on import or set
low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

[122]:
```
from planetarypy.pds.indexes import find_mixed_type_cols
```

[123]:
```
find_mixed_type_cols(df, fix=False)
```

```
9
1999-01-07 10:05:02.093
1999-01-07 10:08:34.0
N/A
EUV1999-01-07 17:05:02.000
Unnamed: 6
USTARE
to evaluate EUV and FUV functions.
7
32
0
0.1
0.2
```

[123]:
```
['9',
 '1999-01-07 10:05:02.093',
 '1999-01-07 10:08:34.0',
 'N/A',
 'EUV1999-01-07 17:05:02.000',
 'Unnamed: 6',
 'USTARE',
 'to evaluate EUV and FUV functions.',
 '7',
 '32',
 '0',
 '0.1',
 '0.2']
```

[125]:
```
df.columns
```

[125]:
```
Index(['9', '1999-01-07 10:05:02.093', '1999-01-07 10:08:34.0', 'EUV', 'N/A',
       'EUV1999-01-07 17:05:02.000', 'Unnamed: 6', 'USTARE',
       'to evaluate EUV and FUV functions.', '7', '32', '0', '0.1', '0.2',
       '0.3', '1', '0.4'],
      dtype='object')
```

[126]:
```
index.columns
```

[126]:
```
Index(['FILE_NAME', 'OBSERVATION_TYPE', 'START_TIME', 'STOP_TIME',
       'TARGET_NAME', 'DATA_SET_ID', 'SPACECRAFT_CLOCK_START_COUNT',
       'SPACECRAFT_CLOCK_STOP_COUNT', 'INTEGRATION_DURATION',
       'COMPRESSION_TYPE', 'HI_VOLTAGE_POWER_SUPPLY_STATE',
       'OCCULTATION_PORT_STATE', 'SLIT_STATE', 'TEST_PULSE_STATE', 'ODC_ID',
       'RIGHT_ASCENSION', 'DECLINATION', 'SUB_SOLAR_LATITUDE',
       'SUB_SOLAR_LONGITUDE', 'SUB_SPACECRAFT_LATITUDE',
       'SUB_SPACECRAFT_LONGITUDE', 'PHASE_ANGLE', 'EMISSION_ANGLE',
       'SOLAR_INCIDENCE_ANGLE', 'CENTRAL_BODY_DISTANCE', 'DWELL_TIME',
       'H_LEVEL', 'D_LEVEL', 'filename'],
```

```
      dtype='object')
```

[133]:
```
index[index.filename.str.startswith("EUV")].iloc[0]
```

[133]:
```
FILE_NAME
/COUVIS_0001/DATA/D1999_007/EUV1999_007_17_05.LBL
OBSERVATION_TYPE
USTARE
START_TIME                                                1999-01-07
17:05:01.949000
STOP_TIME                                                 1999-01-07
17:08:37.949000
TARGET_NAME
NaN
DATA_SET_ID                                                    CO-J-
UVIS-2-SPEC-V1.2
SPACECRAFT_CLOCK_START_COUNT
1/1294420183.000
SPACECRAFT_CLOCK_STOP_COUNT
UNK
INTEGRATION_DURATION
4.0
COMPRESSION_TYPE
NONE
HI_VOLTAGE_POWER_SUPPLY_STATE
OFF
OCCULTATION_PORT_STATE
CLOSED
SLIT_STATE
HIGH_RESOLUTION
TEST_PULSE_STATE
ON
ODC_ID
7
RIGHT_ASCENSION
-999.0
DECLINATION
-999.0
SUB_SOLAR_LATITUDE
-999.0
SUB_SOLAR_LONGITUDE
-999.0
SUB_SPACECRAFT_LATITUDE
-999.0
SUB_SPACECRAFT_LONGITUDE
-999.0
PHASE_ANGLE
```

```
-999.0
EMISSION_ANGLE
-999.0
SOLAR_INCIDENCE_ANGLE
-999.0
CENTRAL_BODY_DISTANCE
-999.0
DWELL_TIME                                          1969-12-31
23:59:59.999999001
H_LEVEL
NaN
D_LEVEL
NaN
filename
EUV1999_007_17_05
Name: 9, dtype: object
```

[127]:
```
obs.head()
```

[127]:

	filename	slit	nx	ny	nz	int	odcid	x1	x2	y1	y2	\
0	EUV1999_016_19_47_15	2	1024	64	2	60	12	0	1023	0	63	
1	EUV1999_016_19_49_06	2	1024	1	123	1	13	0	1023	0	63	
2	EUV1999_016_20_08_10	2	1	64	399	1	11	0	1023	0	63	
3	EUV1999_016_20_16_14	2	1024	64	2	60	12	0	1023	0	63	
4	EUV1999_016_20_18_06	2	1024	1	123	1	13	0	1023	0	63	

	name
0	alp vir
1	alp vir
2	alp vir
3	alp vir
4	alp vir

[ ]:
```
cols = ["index start_time stop_time detector target obsid_time unknown type␣
↪comment "]
```