import sys import os from math import isnan from pylatex import Subsection, NoEscape, Figure, LongTabu, Subsubsection, Enumerate, MultiColumn, NewPage, TextColor from pylatex.utils import bold from pylatex.section import Paragraph import datetime as dt from nested_lookup import nested_lookup import pandas as pd

from gfzrnx import gfzrnx_constants as gfzc

from ltx import ltx_gfzrnx_report

author = "amuls"

def rnxobs_script_information(dCli: dict, dHdr: dict, dInfo: dict, script_name: str) -> Subsection: """ rnxobs_script_information creates the section with information about the script obstab_analyze """ info_report = ltx_gfzrnx_report.report_information(dInfo=dInfo)

```
n = 10  # max elements per line in longtabu

ssec = Subsection('Script details')
with ssec.create(Subsubsection(title='Program information', numbering=True)) as
    with sssec.create(LongTabu('rcl', pos='l', col_space='4pt')) as longtabu:
        longtabu.add_row(('Script', ':', script_name))
        longtabu.add_row(('Run at', ':', '{dt!s}'.format(dt=dt.datetime.now().st
        longtabu.add_row(('Run by', ':', '{author:s}'.format(author=', '.join(in
        longtabu.add_row(('', '', '{company:s}'.format(company=', '.join(info_re
        # longtabu.add_empty_row()

with ssec.create(Subsubsection(title='Parameters', numbering=True)) as sssec:
    with sssec.create(LongTabu('rcl', pos='l', col_space='4pt')) as longtabu:
        longtabu.add_row(('RINEX root directory', ':', os.path.expanduser(dCli['
        longtabu.add_row(('RINEX observation file', ':', dCli['obsf']))
        longtabu.add_row(('RINEX version', ':', dHdr['file']['version']))
        longtabu.add_row(('Marker', ':', dInfo['marker']))
        longtabu.add_row(('Year/day-of-year', ':', '{yyyy:04d}/{doy:03d}'.format
        longtabu.add_empty_row()

with ssec.create(Subsubsection(title='Observation header information', numbering
    with sssec.create(LongTabu('rcl', pos='l', col_space='4pt')) as longtabu:
        # add start / end DTG and interval
        epoch_first = nested_lookup(key='first', document=dHdr)[0]
        epoch_last = nested_lookup(key='last', document=dHdr)[0]
        interval = float(nested_lookup(key='interval', document=dHdr)[0])
        longtabu.add_row(('First epoch', ':', dt.datetime.strptime(epoch_first.s
        longtabu.add_row(('Last epoch', ':', dt.datetime.strptime(epoch_last.spl
        longtabu.add_row(('Interval', ':', '{intv:.1f}'.format(intv=interval)))
        longtabu.add_empty_row()

        for i, gnss in enumerate(dCli['GNSSs']):
            if i == 0:
                longtabu.add_row(('GNSS', ':', '{gnss:s} ({name:s}) '.format(gns
            else:
                longtabu.add_row(('', ':', '{gnss:s} ({name:s}) '.format(gnss=gn

        # add the frequencies
        for gnss, sysfreq in dHdr['file']['sysfrq'].items():
            if gnss in dCli['GNSSs']:
                longtabu.add_row(('Frequencies {syst:s}'.format(syst=gnss), ':',
        longtabu.add_empty_row()

        # add observable types available
        for i, obst in enumerate(gfzc.lst_obstypes):
            if i == 0:
                longtabu.add_row(('Observable types', ':', '{obst:s} ({name:s})
            else:
                longtabu.add_row(('', ':', '{obst:s} ({name:s}) '.format(obst=ob
        longtabu.add_empty_row()

with ssec.create(Subsubsection(title='Logged observables', numbering=True)) as s
    # add info about observable types logged
    with sssec.create(LongTabu('rcl', pos='l', col_space='4pt')) as longtabu:
        for gnss, obstypes in dHdr['file']['sysobs'].items():
            if gnss in dCli['GNSSs']:
                if len(obstypes) > n:
                    subobstypes = [obstypes[i * n:(i + 1) * n] for i in range((l
1) // n)]
                    for i, subsubobstypes in enumerate(subobstypes):
                        if i == 0:
                            longtabu.add_row(('Observable types {syst:s}'.format
                        else:
                            longtabu.add_row(('', '', '{obst:s}'.format(obst=',
                else:
                    longtabu.add_row(('Observable types {syst:s}'.format(syst=gn
        longtabu.add_empty_row()

return ssec
```

def ltx_obsstat_analyse(obsstatf: str, dfObsTle: pd.DataFrame, plots: dict, script_name: str) -> Subsection: """ ltx_obsstat_analyse summarises the observations compared to TLE information obtained from file obsstatf """ ssec = Subsection("Analysis of observation statistics")

```
# select the columns used for plotting
col_names = dfObsTle.columns.tolist()
GNSS = col_names[col_names.index('PRN') - 1]
obstypes = [x for x in col_names[col_names.index('PRN') + 1:]]
print('obstypes = {}'.format(obstypes))
# we only look at the SNR values since the same value for C/L/D, thus remove sta
navsigs = ['{gnss:s}{navsig:s}'.format(gnss=GNSS, navsig=x[1:]) for x in obstype
1]]

with ssec.create(LongTabu('rcl', col_space='4pt')) as longtabu:
    longtabu.add_row(['statistics observation file', ':', '{statf:s}'.format(sta
    longtabu.add_row(['navigation services for {gnss:s}'.format(gnss=gfzc.dict_G

# add TLE_count to navsigs
# navsigs.append(obstypes[-1])

with ssec.create(Subsubsection(title='Observables count per navigation signal',
    # add timing and observation count info
    sssec.append('The following table represents the number of observations made

    # determine align formats for langtabu
    fmt_tabu = 'l|' + 'rr|' * len(navsigs) + 'r'

    # with sssec.create(Table(position='H')) as table:
    with sssec.create(LongTabu(fmt_tabu, pos='l', col_space='4pt')) as longtabu:
        print(['PRN'] + navsigs + [obstypes[-1]])
        col_row = ['PRN']
        for navsig in navsigs:
            # col_row += [MultiColumn(size=2, align='|c|', data=navsig)]
            col_row += [navsig, '']
        col_row += [obstypes[-1]]
        print(col_row)

        longtabu.add_row(col_row, mapper=[bold])  # header row , mapper=[bold]
        longtabu.add_hline()
        longtabu.end_table_header()

        for index, row in dfObsTle.iterrows():

            # add percentage in the following row if TLE_count differs 0
            # tle_obs = row[obstypes[-1]] / 100
            # if tle_obs > 0:
            #     obstle_perc = ['{:.1f}%'.format(float(x / tle_obs)) for x in r
1]].tolist()]
            #     # longtabu.add_row([''] + ['{:.1f}% '.format(float(x / tle_obs
1]].tolist()] + [''])
            # else:
            #     obstle_perc = ['---'] * len(obstypes[:-1])
            #     # longtabu.add_row([''] + ['---'] * len(obstypes[:-
1]) + [''])  # add emty row

            print('index = {}'.format(index))
            print('row = {}'.format(row))

            prn_row = [row.PRN]
            tle_obs = row[obstypes[-1]] / 100
            for obstype in obstypes[:-1]:
                print("type(row[obstype]) = {}".format(type(row[obstype])))
                prn_row += ['{}'.format(row[obstype])]

                if tle_obs > 0:
                    print("row[obstype]/tle = {:.1f}%".format(row[obstype] / tle
                    prn_row += ['{:.1f}%'.format(row[obstype] / tle_obs)]
                else:
                    prn_row += ['---']

            prn_row += ['{}'.format(row[obstypes[-1]])]
            print('prn_row = {}'.format(prn_row))

            # print([row.PRN, '{}'.format(row[obstypes[:-1]].tolist()), '{}'.for
1])])
            longtabu.add_row(prn_row)

    # add figures representing the observations
    ssec.append(NoEscape(r'Figure \ref{fig:obst_gnss_' + '{gnss:s}'.format(gnss=

    with sssec.create(Figure(position='htbp')) as plot:
        plot.add_image(plots['obs_count'],
                       width=NoEscape(r'0.95\textwidth'),
                       placement=NoEscape(r'\centering'))
        # plot.add_caption('Observation count per navigation signal')
        plot.add_caption(NoEscape(r'\label{fig:obst_gnss_' + '{gnss:s}'.format(g

    with sssec.create(Figure(position='htbp')) as plot:
        plot.add_image(plots['obs_perc'],
                       width=NoEscape(r'0.95\textwidth'),
                       placement=NoEscape(r'\centering'))
        plot.add_caption(NoEscape(r'\label{fig:rel_obst_gnss_' + '{gnss:s}'.form

    with sssec.create(Figure(position='htbp')) as plot:
        plot.add_image(plots['relative'],
                       width=NoEscape(r'0.95\textwidth'),
                       placement=NoEscape(r'\centering'))
        plot.add_caption(NoEscape(r'\label{fig:prec_obst_gnss_' + '{gnss:s}'.for

    ssec.append(NoEscape(r'\clearpage'))

return ssec
```

def obstab_tleobs_ssec(obstabf: str, lst_PRNs: list, lst_NavSignals: list, lst_ObsFreqs: list, dfTle: pd.DataFrame) -> Subsection: """ obstab_tleobs_ssec creates a subsection used for adding info about analysis of the observations """ n = 10 # max elements per line in longtabu

```
ssec = Subsection('Detailed analysis of observation types per navigation signal'

with ssec.create(LongTabu('rcl', pos='c', col_space='4pt')) as longtabu:
    longtabu.add_row(['Observation tabular file', ':', '{tabf:s}'.format(tabf=ob
    if len(lst_PRNs) > n:
        sublst_PRNs = [lst_PRNs[i * n:(i + 1) *n] for i in range((len(lst_PRNs)
1) // n)]
        for i, subsublst_PRNs in enumerate(sublst_PRNs):
            if i == 0:
                longtabu.add_row(('Examined satellites', ':', '{prns:s}'.format(
            else:
                longtabu.add_row(('', '', '{prns:s}'.format(prns=', '.join(subsu
    else:
        longtabu.add_row(('Examined satellites', ':', '{prns:s}'.format(prns=',

    # longtabu.add_empty_row()

    if len(lst_NavSignals) > n:
        sublst_NavSignals = [lst_NavSignals[i * n:(i + 1) * n] for i in range((l
1) // n)]
        for i, subsublst_NavSignals in enumerate(sublst_NavSignals):
            if i == 0:
                longtabu.add_row(('Examined navigation signals', ':', '{obst:s}'
            else:
                longtabu.add_row(('', '', '{obst:s}'.format(obst=', '.join(subsu
    else:
        longtabu.add_row(('Examined navigation signals', ':', '{obst:s}'.format(

    # longtabu.add_empty_row()

    if len(lst_ObsFreqs) > n:
        sublst_ObsFreqs = [lst_ObsFreqs[i * n:(i + 1) * n] for i in range((len(l
1) // n)]
        for i, subsublst_ObsFreqs in enumerate(sublst_ObsFreqs):
            if i == 0:
                longtabu.add_row(('Examined observables', ':', '{obst:s}'.format
            else:
                longtabu.add_row(('', '', '{obst:s}'.format(obst=', '.join(subsu
    else:
        longtabu.add_row(('Examined observables', ':', '{obst:s}'.format(obst=',

# add the TLE rise / set / cul info
with ssec.create(Subsubsection(r'TLE time spans')) as sssec:
    sssec.append(NoEscape(r'The table below represents the calculated rise and s

    with sssec.create(LongTabu('l|l|l|l|l', pos='c', col_space='4pt')) as longta
        longtabu.add_row(['PRN'] + dfTle.columns.tolist(), mapper=[bold])  # hea
        longtabu.add_hline()
        longtabu.end_table_header()
        # longtabu.add_row(['PRN'] + dfTle.columns.tolist(), mapper=[bold])  # h
        # longtabu.add_hline()
        # longtabu.end_header()

        for prn, row in dfTle.iterrows():
            tabu_row = [prn]
            for col in dfTle.columns.tolist():
                tle_toadd = [tle_conversion(tle_val) for tle_val in row[col]]
                tabu_row.append(', '.join(tle_toadd))

            longtabu.add_row(tabu_row)

        longtabu.add_hline()

return ssec
```

def tle_conversion(tle_value): """ converts to HMS string if needed """ if isinstance(tle_value, dt.time): return tle_value.strftime(“%H:%M:%S”) else: if isnan(tle_value): return ” else: return str(tle_value)

def obstab_tleobs_overview(gnss: str, navsigs: list, navsig_plts: dict, navsig_obst_lst: dict, dPNT: dict) -> Subsubsection: """ obstab_tleobs_overview adds the info about the TLE rise/set/cul times and the general overview plot """ sssec = Subsubsection(r’Navigation signals analysis’)

```
# go over the available navigation signals
print('navsigs = {}'.format(navsigs))
for navsig in navsigs:
    print('navsig = {}'.format(navsig))
    print('navsig_plts = {}'.format(navsig_plts))
    sssec.append(NewPage())

    with sssec.create(Paragraph(r'Analysis of navigation signal {gnss:s}{navs:s}

        with paragraph.create(Enumerate()) as enum:

            # add figures representing the observations per navigation signal
            enum.add_item(NoEscape(r'Figure \ref{fig:tle_navsig_' + '{navs:s}'.f

            with enum.create(Figure(position='htbp')) as plot:
                plot.add_image(navsig_plts[navsig]['tle-obs'],
                               width=NoEscape(r'0.95\textwidth'),
                               placement=NoEscape(r'\centering'))

                # print(r'\label{fig:tle_navsig_' + r'{gnss:s}'.format(gnss=gnss

                plot.add_caption(NoEscape(r'\label{fig:tle_navsig_' + '{navs:s}'

            print('navsig_obst_lst[{}] = {}'.format(navsig, navsig_obst_lst[navs

            for navsig_obst in navsig_obst_lst[navsig]:
                print('navsig_obst = {}'.format(navsig_obst))
                enum.add_item(NoEscape(r'Figure \ref{fig:tle_navsig_' + '{gnss:s
                with enum.create(Figure(position='htbp')) as plot:
                    plot.add_image(navsig_plts[navsig]['obst'][navsig_obst],
                                   width=NoEscape(r'0.95\textwidth'),
                                   placement=NoEscape(r'\centering'))

                    # print(r'\label{fig:tle_navsig_' + r'{gnss:s}'.format(gnss=

                    plot.add_caption(NoEscape(r'\label{fig:tle_navsig_' + '{gnss

            # add evolution of the PRNcnt over time (more or less than 4) and re
            if len(dPNT[navsig]['loss']) > 0:
                enum.append('The table below summarises the PRN count statistics

                with enum.create(LongTabu('r|r|r', pos='l', col_space='4pt')) as
```

```
                    longtabu.add_row((MultiColumn(3, align='c', data=TextColor('
                    longtabu.add_row(['Loss of PNT', 'PNT Reacquisition', 'Durat
                    longtabu.add_hline()
                    longtabu.end_table_header()
                    # longtabu.add_row(['PRN'] + dfTle.columns.tolist(), mapper=
                    # longtabu.add_hline()
                    # longtabu.end_header()

                    print('len loss / reacq = {} {}'.format(len(dPNT[navsig]['lo
                    for loss, reacq, gap in zip(dPNT[navsig]['loss'], dPNT[navsi
                        print('{} -> {}: {}'.format(loss.strftime('%H:%M:%S'), r
                        longtabu.add_row([loss.strftime('%H:%M:%S'), reacq.strft

return sssec
```