Analisys_parameters

February 13, 2020

1 Analysis of the parameters

Analise the paremeters found with the iterative approach in the 0hours region of DR2.

[1]:
```
from glob import glob
import pickle
import os
import sys
import numpy as np
from astropy.table import Table
```

[2]:
```
try:
    BASEPATH = os.path.dirname(os.path.realpath(__file__))
    data_path = os.path.join(BASEPATH, "..", "..", "data")
except NameError:
    if os.path.exists("data"):
        BASEPATH = "."
        data_path = os.path.join(BASEPATH, "data")
    else:
        BASEPATH = os.getcwd()
        data_path = os.path.join(BASEPATH, "..", "..", "data")
```

[3]:
```
sys.path.append(os.path.join(BASEPATH, '..', '..', 'src'))
```

[4]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

[5]:
```
idp = os.path.join(BASEPATH, "..", "..", "data", "idata", "params")
```

1.1 Load the data

[6]:
```
#combined_all = Table.read(os.path.join(data_path, "samples", "test_combined.
↪fits"))
```

[7]:
```
param_list = sorted(glob(os.path.join(idp, "lofar_params*.pckl")))
```

[8]:
```
params = [pickle.load(open(p, "rb")) for p in param_list]
```

[9]:
```
# bin_list, centers, Q_0_colour, n_m, q_m
```

[10]:
```
lofar_table_list = sorted(glob(os.path.join(idp, "lofar_m*.fits")))
```

[11]:
```
lofar = Table.read(lofar_table_list[-1])
```

1.2 Analysis

1.2.1 Number of sources changed

[12]:
```
for i in range(1, 7):
    n_changes = np.sum((lofar["lr_index_sel_{}".format(i+1)] !=␣
 ↪lofar["lr_index_{}".format(i)]) &
                       ~np.isnan(lofar["lr_index_sel_{}".format(i+1)]) &
                       ~np.isnan(lofar["lr_index_{}".format(i)]))
    print(f"Changes from {i} to {i+1}: {n_changes}")
```

```
Changes from 1 to 2: 3135
Changes from 2 to 3: 139
Changes from 3 to 4: 20
Changes from 4 to 5: 8
Changes from 5 to 6: 3
Changes from 6 to 7: 0
```

[ ]:
```

```

[13]:
```
from matplotlib import cm
from matplotlib.collections import LineCollection
from matplotlib.lines import Line2D
```

[14]:
```
bin_list, centers, Q_0_colour, n_m, q_m = params[-1]
```

[15]:
```
Q_0_colour
```

[15]:
```
array([0.00772401, 0.12601914, 0.00148692, 0.00587783, 0.04278144,
       0.06475596, 0.05079485, 0.05203229, 0.06675182, 0.0828984 ,
       0.10428413, 0.21186145])
```

[16]:
```
len(n_m)
```

[16]:
```
12
```

[33]:
```
plt.rcParams["figure.figsize"] = (6.64*1.2, 6.64*0.74)
plt.rcParams["figure.dpi"] = 300
plt.rcParams['lines.linewidth'] = 1.75
plt.rcParams['lines.markersize'] = 8.0
plt.rcParams['lines.markeredgewidth'] = 0.75
plt.rcParams['font.size'] = 15.0 ## not 18.0
plt.rcParams['font.family'] = "serif"
plt.rcParams['font.serif'] = "CM"
plt.rcParams['xtick.labelsize'] = 'small'
plt.rcParams['ytick.labelsize'] = 'small'
plt.rcParams['xtick.major.width'] = 1.0
```

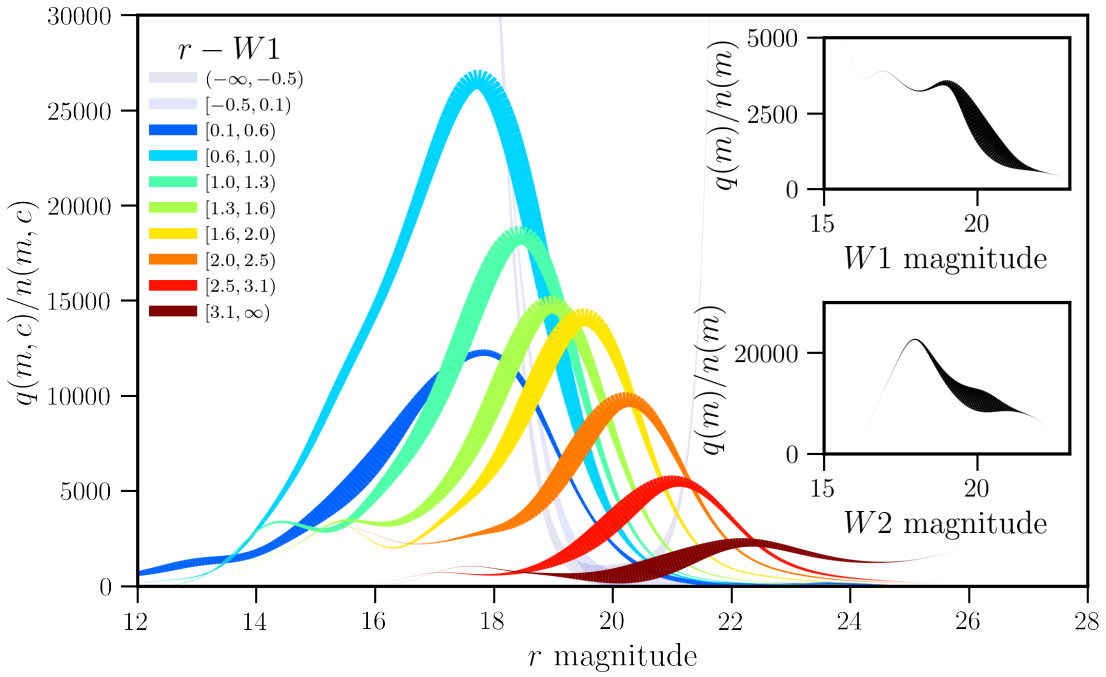
```
plt.rcParams['xtick.major.size'] = 8
plt.rcParams['ytick.major.width'] = 1.0
plt.rcParams['ytick.major.size'] = 8
plt.rcParams['xtick.minor.width'] = 1.0
plt.rcParams['xtick.minor.size'] = 4
plt.rcParams['ytick.minor.width'] = 1.0
plt.rcParams['ytick.minor.size'] = 4
plt.rcParams['axes.linewidth'] = 1.5
plt.rcParams['legend.fontsize'] = 'small'
plt.rcParams['legend.numpoints'] = 1
plt.rcParams['legend.labelspacing'] = 0.4
plt.rcParams['legend.frameon'] = False
plt.rcParams['text.usetex'] = True
plt.rcParams['savefig.dpi'] = 300
```

[44]:
```
low = np.nonzero(centers[1] >= 15)[0][0]
high = np.nonzero(centers[1] >= 22.2)[0][0]

cm_subsection = np.linspace(0., 1., len(q_m)-2)
colors = [ cm.jet(x) for x in cm_subsection ]

alphas = [0.1, 0.1, 1, 1, 1, 1, 1, 1, 1, 1]

fig, a = plt.subplots()
lcs = []
proxies = []

def make_proxy(zvalue, scalar_mappable, **kwargs):
    color = scalar_mappable.cmap(scalar_mappable.norm(zvalue))
    return Line2D([0, 1], [0, 1], color=color, **kwargs)

for i, q_m_k in enumerate(q_m[2:]):
    #plot(centers[i], q_m_old[i]/n_m_old[i])
    q_m_aux = q_m[i]/np.sum(q_m[i])
    lwidths = (q_m_aux/np.max(q_m_aux)*10).astype(float)
    #if save_pdf and (i<6): # Solve problems with the line with in pdfs
    #    lwidths[lwidths < 0.005] = 0
    #print(lwidths)

    y_aux = q_m_k/n_m[i]
    factor = np.max(y_aux[low:high])
    y = y_aux
    #print(y)
    x = centers[i]

    points = np.array([x, y]).T.reshape(-1, 1, 2)
    segments = np.concatenate([points[:-1], points[1:]], axis=1)
```

```
    #lc = LineCollection(segments, linewidths=lwidths, color=colors[i])
    if i not in [-1]:
        color = colors[i]

        lcs.append(LineCollection(segments, linewidths=lwidths, color=color,␣
 ↪alpha=alphas[i]))
        proxies.append(Line2D([0, 1], [0, 1], color=color, lw=5,␣
 ↪alpha=alphas[i]))

        a.add_collection(lcs[-1])
        plt.xlim([12, 28])
        plt.ylim([0, 30000])
        plt.xlabel("$r$ magnitude")
        plt.ylabel("$q(m,c)/n(m,c)$")

a.legend(proxies,
         [
   "$(-\infty, -0.5)$",
   "$[-0.5, 0.1)$",
   "$[0.1, 0.6)$",
   "$[0.6, 1.0)$",
   "$[1.0, 1.3)$",
   "$[1.3, 1.6)$",
   "$[1.6, 2.0)$",
   "$[2.0, 2.5)$",
   "$[2.5, 3.1)$",
   "$[3.1, \infty)$"
    ],
        fontsize="xx-small",
        title="$r-W1$",
        loc=2)

inset = plt.axes([0.685, 0.3, .2, .2])
q_m_aux = q_m[0]/np.sum(q_m[0])
lwidths = (q_m_aux/np.max(q_m_aux)*10).astype(float)
y_aux = q_m[0]/n_m[0]
factor = np.max(y_aux[low:high])
y = y_aux
x = centers[0]
points = np.array([x, y]).T.reshape(-1, 1, 2)
segments = np.concatenate([points[:-1], points[1:]], axis=1)
lc = LineCollection(segments, linewidths=lwidths, color="k")
proxy = Line2D([0, 1], [0, 1], color="k", lw=5)
inset.add_collection(lc)
plt.xlim([15, 23])
plt.ylim([0, 30000])
plt.xlabel("$W2$ magnitude")
```

```
plt.ylabel("$q(m)/n(m)$")

inset2 = plt.axes([0.685, 0.65, .2, .2])
q_m_aux = q_m[1]/np.sum(q_m[1])
lwidths = (q_m_aux/np.max(q_m_aux)*10).astype(float)
y_aux = q_m[1]/n_m[1]
factor = np.max(y_aux[low:high])
y = y_aux
x = centers[1]
points = np.array([x, y]).T.reshape(-1, 1, 2)
segments = np.concatenate([points[:-1], points[1:]], axis=1)
lc = LineCollection(segments, linewidths=lwidths, color="k")
proxy = Line2D([0, 1], [0, 1], color="k", lw=5)
inset2.add_collection(lc)
plt.xlim([15, 23])
plt.ylim([0, 5000])
plt.xlabel("$W1$ magnitude")
plt.ylabel("$q(m)/n(m)$")
```

[44]: `Text(0, 0.5, '$q(m)/n(m)$')`



[ ]: