User Guide para ajuste de gradiente de abundância

André Felipe de Siqueira Cardoso, PhD

Contents

1 Introdução

Este guia serve como um apoio para a utilização do script do ajuste do perfil de gradientes de abundância de oxigênio, utilizando métodos indiretos. A ideia é que o usuário forneça as informações sobre as galáxias e o script fará, automaticamente, sem a supervisão humana, o ajuste do gradiente de abundância. O melhor modelo será definido com base no critério de Akaike (Akaike, 1973), avaliando se o perfil é linear, quebrado ou duplamente quebrado. Para uma melhor visualização do resultado esperado, confira os ajustes realizados em Cardoso et al. (2025).

2 Como utilizar o script

Este script foi pensado para automatizar os cálculos de abundâncias de oxigênio, bem como ajustar o melhor modelo do gradiente de abundância. A ideia é que o usuário insira informações necessárias da galáxia para esse tipo de análise, como os fluxos, ra e dec, por exemplo. O usuário deve fazer o download dos seis arquivos .py disponíveis no GitHub e salvá-los todos no mesmo diretório onde será utilizado esse script. O arquivo fit_OH.py é o arquivo que compila todos os outros cinco arquivos e realiza a tarefa. Dentro deste arquivo fit_OH.py é definido a função fit_final(...) que deve ser utilizada para rodar o script. Abaixo são apresentadas todas as informações necessárias para utilizar esse script.

```
fit_OH.fit_final(
name = nome da galáxia (sem espaços),
HIIREGID = array com o ID de cada região HII,
ra = array da ascensão reta de cada região HII em graus,
ra0 = ascensão reta do centro da galáxia em graus,
dec = array da declinação de cada região HII em graus,
```

```
dec0 = declinação do centro da galáxia em graus,
pa = ângulo de posição em graus,
ba = razão entre os semi-eixos menor e maior da galáxia,
d = distância da galáxia em Mpc,
re = raio efetivo da galáxia em kpc,
EWHa = array contendo a largura equivalente de cada região HII em Angstroms (\AA),
Hb4861 = array com o fluxo de H$\beta$,
eHb4861 = array com o erro do fluxo de H$\beta$,
Ha6562 = array com o fluxo de H$\alpha$,
eHa6562 = array com o erro do fluxo de H$\alpha$,
OIII5006 = array com o fluxo de OIII5006,
eOIII5006 = array com o erro do fluxo de OIII5006,
NII6583 = array com o fluxo de NII6583,
eNII6583 = array com o erro do fluxo de NII6583,
SII6716 = array com o fluxo de SII6716,
eSII6716 = array com o erro do fluxo de SII6716,
SII6730 = array com o fluxo de SII6730,
eSII6730 = array com o erro do fluxo de SII6730,
calibrator = número que identifica qual índice e calibrador deve ser utilizado,
criterion = identificação do critério utilizado para selecionar as regiões HII,
save_table = True para salvar a tabela, False caso não deseje,
save_graph = True para salvar o gráfico, False caso não deseje,
show_graph = True para exibir o gráfico, False caso não deseje.
)
```

O parâmetro *name* é o nome da galáxia e deve ser do tipo string. Os parâmetros *ra0, dec0, pa, ba, d* e *re* são valores constantes e devem ser do tipo float. Os parâmetros *HIIREGID, ra, dec, EWHa, Hb4861, eHb4861, Ha6562, eHa6562, OIII5006, eOIII5006, NII6583, eNII6583, SII6716, eSII6716, SII6730* e *eSII6730* devem ser do tipo array. O parâmetro *calibrator* deve assumir os valores 1, 2, 3, 4 ou 5, de acordo com o índice e calibrador a ser utilizado no cálculo da abundância (ver seção 3.2). O parâmetro *criterion* deve ser uma string, identificada como *KE01, KE6A, KA03, ST06* ou *CF11* de acordo com o critério desejado para selecionar as regiões HII (ver seção 3.3). Os parâmetros *save_table, save_graph* e *show_graph* são booleanos, onde assume True ou False, a critério do usuário. Para uma explicação mais detalhada de todos esses parâmetros de entrada veja a seção 3 abaixo.

É importante ressaltar que caso o usuário não possua o array HIIREGID, por exemplo, é possível criar um array do tipo HIIREGID = np.zeros(len(amostra)). No entanto, é necessário atenção para usar isso com os outros parâmetros, pois pode calcular de maneira errada as abundâncias.

Ao final da tarefa, o script retorna um dicionário contendo os parâmetros do ajuste do melhor modelo. Contudo, para que isso ocorra, é necessário atribuir o resultado a uma variável, conforme o exemplo abaixo:

```
result = fit_OH.fit_final(name, HIIREGID, ra, ra0, dec, dec0, pa, ba, d, re, EWHa,
Hb4861, eHb4861, Ha6562, eHa6562, OIII5006, eOIII5006, NII6583, eNII6583, calibrator,
criterion, save_table, save_graph, show_graph)
```

E por fim os parâmetros do ajuste podem ser acessados como:

```
galaxy = result['galaxy']
b0 = result['b0']
eb0 = result['eb0']
a1 = result['a1']
ea1 = result['ea1']
```

```
h1 = result['h1']
eh1 = result['eh1']
a2 = result['a2']
ea2 = result['ea2']
h2 = result['h2']
eh2 = result['eh2']
a3 = result['a3']
ea3 = result['ea3']
```

Na seção 4 é apresentado um exemplo de como utilizar esse script. Para isso, será necessário realizar o download dos arquivos "HII.NGC0309.flux_elines.diff_corr.csv", "data_NGC0309.csv" e "example.py", disponíveis no GitHub.

3 Explicação passo a passo das tarefas

O funcionamento por trás desse script está dividido em seis etapas. Cada etapa corresponde a um arquivo que realiza uma função específica. Abaixo é apresentado uma explicação das informações necessárias que cada função necessita e o que ela retorna.

3.1 distance.py

O arquivo distance.py retorna as distâncias deprojetadas de cada região HII da galáxia normalizada pelo raio efetivo da galáxia, de acordo com Scarano et al. (2008). Abaixo são apresentadas as informações necessárias para o cálculo, bem como as unidades específicas de cada grandeza.

```
distance.distances(
ra = array da ascensão reta de cada região HII em graus,
ra0 = ascensão reta do centro da galáxia em graus,
dec = array da declinação de cada região HII em graus,
dec0 = declinação do centro da galáxia em graus,
pa = ângulo de posição em graus,
ba = razão entre os semi-eixos menor e maior da galáxia,
d = distância da galáxia em Mpc,
re = raio efetivo da galáxia em kpc
)
```

Essa função retornará um array com a distância r de cada região HII em relação ao centro da galáxia, normalizada pelo raio efetivo.

3.2 abundance.py

Essa função fará a correção da extinção das linhas de emissão das regiões HII. Internamente, é calculado o fator de extinção para cada linha de emissão (Cavichia et al., 2010). Em seguida, o excesso de cor e a correção de cada linha são realizadas (Cavichia, 2008). Neste caso, as correções das linhas são feitas em termos de Hβ, de tal maneira que os índices O3N2 e N2 são calculados da seguinte maneira:

$$\mathrm{O3N2} = \log\left(\frac{\mathrm{OIII}}{\mathrm{H}\beta}\frac{\mathrm{H}\alpha}{\mathrm{H}\beta}\frac{\mathrm{H}\beta}{\mathrm{NII}}\right) \quad \text{e} \quad \mathrm{N2} = \log\left(\frac{\mathrm{NII}}{\mathrm{H}\beta}\frac{\mathrm{H}\beta}{\mathrm{H}\alpha}\right) \tag{1}$$

Com base nesses índices, para cada calibrador é estimado um intervalo de valores válido, ou seja, o calibrador PP04 (Pettini & Pagel, 2004) com os índices N2 e O3N2 é válido somente dentro do intervalo

$-2.5 < \mathrm{N2} < -0.3$ e $-1.0 < \mathrm{O3N2} < 1.9$, respectivamente. O calibrador M13 (Marino et al., 2013) com os índices N2 e O3N2 é válido somente dentro do intervalo $-1.6 < \mathrm{N2} < -0.2$ e $-1.1 < \mathrm{O3N2} < 1.7$, respectivamente. Para maiores detalhes consulte Marino et al. (2013). No caso do calibrador D16 (Dopita et al., 2016), a advertência é que a calibração é descrita no paper como quase linear até uma abundância de $12 + \log(\mathrm{O/H}) = 9.05$.

Além disso, os fluxos são limitados com base em uma distribuição gaussiana, onde desconsideramos regiões que apresentaram um erro no fluxo maior que 3σ da largura total à meia altura da gaussiana ajustada.

Por fim, as abundâncias são estimadas conforme o interesse do usuário. Abaixo estão as informações necessárias para que este script realize as funções:

```
abundance.abundance(
name = nome da galáxia (sem espaços),
r = distância de cada região HII calculada em distance.py,
HIIREGID = array com o ID de cada região HII,
Hb4861 = array com o fluxo de H$\beta$,
eHb4861 = array com o erro do fluxo de H$\beta$,
Ha6562 = array com o fluxo de H$\alpha$,
eHa6562 = array com o erro do fluxo de H$\alpha$,
OIII5006 = array com o fluxo de OIII5006,
eOIII5006 = array com o erro do fluxo de OIII5006,
NII6583 = array com o fluxo de NII6583,
eNII6583 = array com o erro do fluxo de NII6583,
SII6716 = array com o fluxo de SII6716,
eSII6716 = array com o erro do fluxo de SII6716,
SII6730 = array com o fluxo de SII6730,
eSII6730 = array com o erro do fluxo de SII6730,
calibrator = número que identifica qual índice e calibrador deve ser utilizado.
)
```

Até o momento, existem cinco opções de calibradores, apresentados abaixo:

1 - PP04 com O3N2

2 - PP04 com N2

3 - M13 com O3N2

4 - M13 com N2

5 - D16

Por fim, o script retornará arrays da abundância com o respectivo erro, além das linhas Hα, OIII5006 e NII6583 corrigidas. Um arquivo ”nome da galáxia”.csv é salvo no diretório ”tabelas” contendo a ID, a distância *r* normalizada pelo raio efetivo e a largura equivalente de Hα de cada região HII. Também são salvas as linhas Hα, OIII5006, NII6583, SII6716 e SII6730 corrigidas, com os respectivos erros, e as abundâncias calculadas para todos os calibradores. Essa tabela será salva no diretório “tables” (criado automaticamente) identificado como “nome da galáxia.csv”.

3.3 criteria.py

O arquivo criteria.py seleciona, dentre todas as possíveis regiões HII, aquelas que atendem aos critérios definidos para regiões HII de acordo com os diferentes critérios. Até o momento, cinco diferentes critérios, propostos por diferentes autores, estão implementados, sendo eles Kewley et al. (2001), Kewley et al.

(2001) com EWH$\alpha > 6$ Å, Kauffmann et al. (2003), Stasińska et al. (2006) e Cid Fernandes et al. (2011). Esses critérios dependem, basicamente, das linhas Hα, OIII e NII corrigidas e de EWHα. Sendo assim, as informações necessárias para rodar esse arquivo serão:

```
criteria.points(
name = nome da galáxia (sem espaços),
criterion = identificador do critério usado para selecionar as regiões HII,
r = distância de cada região HII calculada em distance.py,
OH = abundância de cada região calculada em abundance.py,
OH_err = erro da abundância de cada região calculado em abundance.py,
EWHa = array da largura equivalente da região HII em Ångströms (\AA),
Ha6562_cor = array da linha H$\alpha$ corrigida proveniente de abundance.py,
OIII5006_cor = array da linha OIII5006 corrigida proveniente de abundance.py,
NII6583_cor = array da linha NII6583 corrigida proveniente de abundance.py,
calibrator = número que identifica qual índice e calibrador devem ser usados,
save_table = opção para salvar uma tabela contendo r, OH e eOH para o calibrador e
critério específicos.
)
```

Os critérios devem ser escolhidos com base nos critérios disponíveis abaixo.

KE01 - Kewley et al. (2001)

KE6A - Kewley et al. (2001) com EWH$\alpha > 6$ Å

KA03 - Kauffmann et al. (2003)

ST06 - Stasińska et al. (2006)

CF11 - Cid Fernandes et al. (2011)

O arquivo criteria.py retornará arrays com a distância *r*, a abundância *OH* e o erro da abundância *eOH*, de acordo com o critério escolhido. Neste caso, para cada critério escolhido, o número de regiões HII pode ser maior ou menor, o qual pode impactar diretamente no ajuste do gradiente de abundância. Para uma melhor explicação sobre isso, consultar as seções 3.2, 4.1 e 5 de Cardoso et al. (2025).

Caso seja de interesse do usuário, é possível salvar uma tabela que contenha *r*, *OH* e *eOH* para o calibrador e critério escolhido pelo usuário. Para isso, marque save_table = True e essa tabela será salva no diretório "tabelas_criterions" (criado automaticamente).

3.4 models.py

Neste arquivo são realizados os ajustes dos gradientes de abundância de oxigênio. Para o caso do ajuste do perfil linear único é utilizado a biblioteca *statsmodels.api*. Para os perfis com uma ou duas quebras é utilizado a biblioteca *piecewise_regression* (Pilgrim, 2021).

A biblioteca *piecewise_regression* é capaz de realizar ajustes para *n* quebras no gradiente. Aqui, como nosso interesse é no máximo duas quebras, já deixamos fixado no script que será realizado o ajuste de apenas uma e duas quebras. Além disso, definições como *min_distance_to_edge* = 0.05, *start_values* = [0.5, 1.5] e *min_distance_between_breakpoints* = 0.20 foram fixadas para evitar ajustes não realistas. Vale ressaltar que esses valores podem ser alterados dentro do arquivo *models.py*, caso seja de interesse do usuário. Conforme Cardoso et al. (2025), para cada ajuste foi realizado um *bootstrap* de 2000, para escapar do mínimo global. Neste script é realizado um *bootstrap* de apenas 200 e, caso seja de interesse do usuário, esse valor também pode ser alterado dentro do arquivo *models.py*. Para maiores informações sobre a biblioteca *piecewise_regression*, consultar Pilgrim (2021).

Nesta etapa, cada um dos três perfis serão ajustados, seguindo os modelos dados pelas expressões 4, 5 e 6 em Cardoso et al. (2025). A escolha do melhor modelo é obtida pelo critério de Akaike (AIC, Akaike, 1973). Para os casos em que a razão entre os pontos observados e o número de parâmetros livres é menor que 40, é aplicada uma correção ao AIC, conforme Narisetty (2020). Para maiores detalhes, consultar a seção 3.2 em Cardoso et al. (2025).

Para rodar este script são necessárias as seguintes informações:

```
models.fit_models(
x_array = array de distâncias r obtidas em criteria.py,
y_array = array de abundâncias OH obtidas em criteria.py,
ey_array = array de erros de abundância eOH obtidos em criteria.py.
)
```

Por fim, esse script retornará um dicionário contendo informações da distância *r*, da abundância *OH* e do erro da abundância *eOH*. Além disso, retornará a informação do melhor modelo ajustado, ou seja, aquele que apresentou o menor valor de AIC, bem como os valores de AIC de cada modelo ajustado. Também são retornados os coeficientes dos ajustes de acordo com as equações 4, 5 e 6 em Cardoso et al. (2025) e seus respectivos erros. Nesse ponto, é definido o coeficiente a_2 como o coeficiente angular do gradiente negativo principal. Os coeficientes a_1 e a_3 são referentes aos coeficientes angulares da região interna e externa, respectivamente. h_1 e h_2 são as posições em que ocorrem as quebras no gradiente e b_0 é o coeficiente linear. Para o caso linear único, por exemplo, os valores de $a_1 = h_1 = h_2 = a_3 = 0.0$.

3.5 plot.py

Este arquivo fará o gráfico do gradiente de abundância de oxigênio com o fit do melhor ajuste para o calibrador e critério escolhido. Este script recebe as informações do dicionário obtido em 3.4 e usa essas informações para gerar o gráfico. Abaixo são apresentadas as informações para gerar o gráfico.

```
plot.plotar_modelo(
results_dict = dicionário retornado em models.py,
name = nome da galáxia (sem espaços),
criterion = identificador do critério usado para selecionar as regiões HII,
calibrator = número que identifica qual índice e calibrador usar,
save_graph = opção para salvar o gráfico,
show_graph = opção para exibir o gráfico.
)
```

Por fim, o gráfico é gerado e cabe ao usuário escolher se deseja visualizá-lo e/ou salvá-lo. Para isso, marque “save_graph = True” e “show_graph = True”. Além disso, será retornado um dicionário contendo o nome da galáxia e os coeficientes dos ajustes do melhor modelo. Cabe ao usuário decidir se quer salvar esses coeficientes em alguma tabela e como fazê-lo.

4 Exemplo

Para rodar esse script modelo é necessário fazer o download dos sete arquivos: distance.py, abundance.py, criteria.py, models.py, plot.py, fit_OH.py e example.py. Além disso, é necessário fazer o download dos arquivos HII.NGC0309.flux_elines.diff_corr.csv e data_NGC0309.csv. Após o download, salve todos os arquivos em um único diretório. Em seguida, abra o arquivo example.py no spyder (caso prefira o jupyter notebook, pode ser necessário realizar algumas modificações no script) e rode esse script. O arquivo example.py é mostrado abaixo:

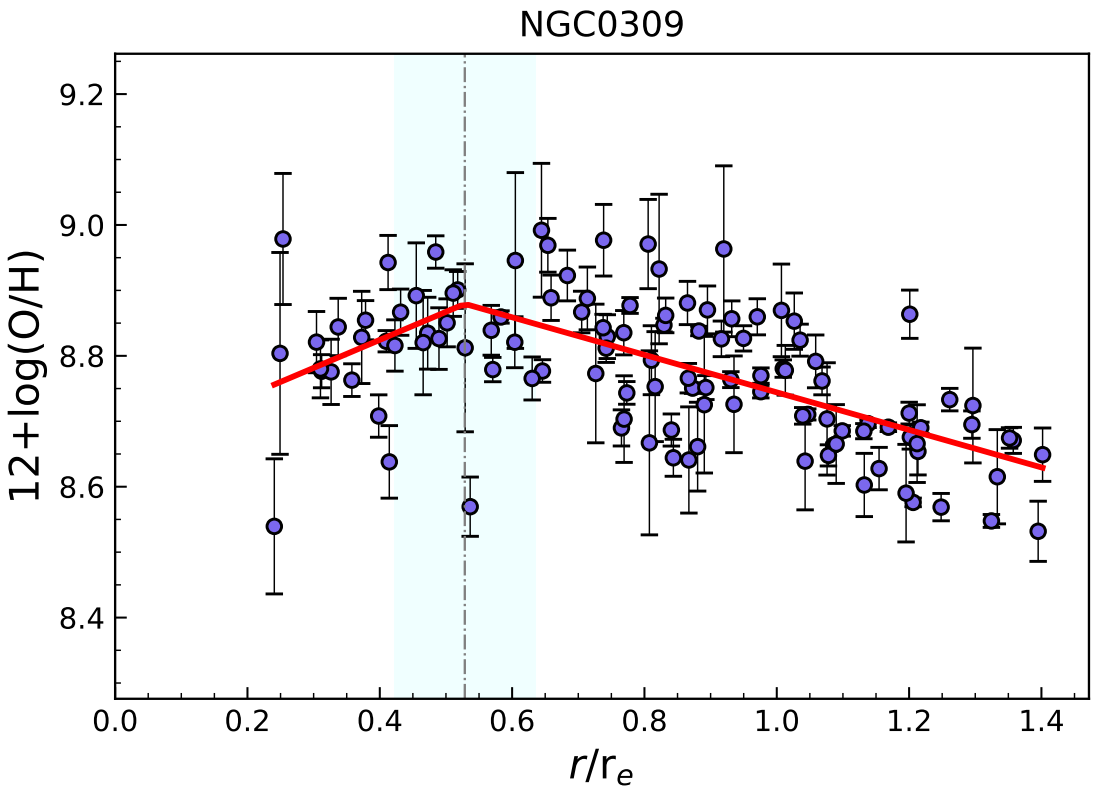
```
1 import fit_OH
```

```
2
3  ##############################################################################
4
5  galaxy_data = pd.read_csv('data_NGC0309.csv')
6
7  name = galaxy_data['galaxy'].iloc[0]
8  ra0 = galaxy_data['ra0'].iloc[0]
9  dec0 = galaxy_data['dec0'].iloc[0]
10 pa = galaxy_data['pa'].iloc[0]
11 ba = galaxy_data['ba'].iloc[0]
12 re = galaxy_data['re'].iloc[0]
13 d = galaxy_data['dist'].iloc[0]
14
15 ##############################################################################
16
17 flux = pd.read_csv('HII.NGC0309.flux_elines.csv')
18
19 ra = flux['RA']
20 dec = flux['DEC']
21 EWHa = flux['EWHa6562']
22 HIIREGID = flux['HIIREGID']
23 Hb4861 = flux['fluxHb4861']
24 eHb4861 = flux['e_fluxHb4861']
25 OIII5006 = flux['fluxOIII5006']
26 eOIII5006 = flux['e_fluxOIII5006']
27 Ha6562 = flux['fluxHa6562']
28 eHa6562 = flux['e_fluxHa6562']
29 NII6583 = flux['fluxNII6583']
30 eNII6583 = flux['e_fluxNII6583']
31 SII6716 = flux['fluxSII6716']
32 eSII6716 = flux['e_fluxSII6716']
33 SII6730 = flux['fluxSII6730']
34 eSII6730 = flux['e_fluxSII6730']
35
36 ##############################################################################
37
38 calibrator = 1
39 criterion = 'KA03'
40 save_table = True
41 save_graph = True
42 show_graph = True
43
44 ##############################################################################
45
46 results = fit_OH.fit_final(name, HIIREGID, ra, ra0, dec, dec0, pa, ba, d, re, EWHa
       , Hb4861, eHb4861, Ha6562, eHa6562, OIII5006, eOIII5006, NII6583, eNII6583,
       SII6716, eSII6716, SII6730, eSII6730, calibrator, criterion, save_table,
       save_graph, show_graph)
47
48 print(results)
```

Esse script fará o ajuste da galáxia NGC 0309, considerando o índice O3N2 com o calibrador PP04. O critério de seleção de regiões HII adotado é KA03 (Kauffmann et al., 2003). O script salva automaticamente a tabela NGC0309.csv no diretório “tables”, que contém as abundâncias para todos os calibradores disponíveis. Como save_table = True e save_graph = True, nos diretórios “tables_criterions” e “graphs” serão

salvos a tabela NGC0309_KA03_PP04_O3N2.csv e o gráfico NGC0309_KA03_PP04_O3N2.png, respectivamente (os diretórios “tables_criterions” e “graphs” são criados automaticamente). Além disso, como show_graph = True, o gráfico deve ser exibido no spyder. Por fim, o comando print(results) mostrará o dicionário que contém os dados dos ajustes do melhor modelo.

Ao final, o usuário deve obter o seguinte gráfico:



References

Akaike, H. 1973, Information Theory and an Extension of the Maximum Likelihood Principle (New York, NY: Springer New York), 199–213

Cardoso, A. F. S., Cavichia, O., Mollá, M., & Sánchez-Menguiano, L. 2025, , 980, 45, doi: 10.3847/1538-4357/ad9eab

Cavichia, O. 2008, Master’s thesis, Universidade de São Paulo, São Paulo

Cavichia, O., Costa, R. D. D., & Maciel, W. J. 2010, , 46, 159, doi: 10.48550/arXiv.1003.0416

Cid Fernandes, R., Stasińska, G., Mateus, A., & Vale Asari, N. 2011, , 413, 1687, doi: 10.1111/j.1365-2966.2011.18244.x

Dopita, M. A., Kewley, L. J., Sutherland, R. S., & Nicholls, D. C. 2016, , 361, 61, doi: 10.1007/s10509-016-2657-8

Kauffmann, G., Heckman, T. M., Tremonti, C., & et al. 2003, , 346, 1055, doi: 10.1111/j.1365-2966.2003.07154.x

Kewley, L. J., Dopita, M. A., Sutherland, R. S., Heisler, C. A., & Trevena, J. 2001, , 556, 121, doi: 10.1086/321545

Marino, R. A., Rosales-Ortega, F. F., Sánchez, S. F., & et al. 2013, , 559, doi: 10.1051/0004-6361/201321956

Narisetty, N. N. 2020, in Handbook of Statistics, Vol. 43, Principles and Methods for Data Science, ed. A. S. R. Srinivasa Rao & C. R. Rao (Elsevier), 207–248, doi: https://doi.org/10.1016/bs.host.2019.08.001

Pettini, M., & Pagel, B. E. J. 2004, , 348, L59–L63, doi: 10.1111/j.1365-2966.2004.07591.x

Pilgrim, C. 2021, Journal of Open Source Software, 6, 3859, doi: 10.21105/joss.03859

Scarano, S., Madsen, F. R. H., Roy, N., & Lépine, J. R. D. 2008, , 386, 963–972, doi: 10.1111/j.1365-2966.2008.13079.x

Stasińska, G., Cid Fernandes, R., Mateus, A., Sodré, L., & Asari, N. V. 2006, , 371, 972, doi: 10.1111/j.1365-2966.2006.10732.x