もし天における天体観測と解析のすすめ

東北大天文学部生 SLA 一同

2022 年 10 月 3 日

目次

図目次

表目次

第 1 章

はじめに

1.1　作成経緯と目標

もし天とは、東北大学天文学教室、宮城教育大学、加速キッチン合同会社、仙台市天文台が主催している高校生向けの研究体験事業で、全国の宇宙、科学に興味のある高校生たちが天文学者としての研究を体験するものです。2022 年で 12 年目となり、多くのもし天卒業生、通称「もしチル」が多方面で活躍されています。また、SLA（Student Learning Assistant）の参加者の中にも、もしチルの方々が多く在籍しています。

そんな「もし天」ですが、もし天に限らず、天文学における観測は**単に写真を撮る**ことにとどまりません。観測の後には必ずその**データ**が存在し、データがどんな意味を持つのかを読み取るために、その解析を行う必要があります。

このテキストは、解析を行う SLA の方のために、光・赤外における観測とは何か、撮像観測と分光観測、また観測画像の処理等についてを解説し、その負担を軽減する目的で作られたものです。

解析の方法にはデータ毎に様々な方法があります。「もし天」で行う観測としては主に「**撮像観測**」と「**分光観測**」があります。これらの観測データの解析には歴史的に「**IRAF**」[1] という、アメリカ国立光学天文台（NOAO）が開発した画像解析ソフトが使われてきました。もし天でもこれまで、このソフトを使って画像解析を行ってきました。

しかし先日、IRAF の NOAO の公式のサポートが今後行われない、つまりディスコンになるという発表がありました。今後の天文学の研究では、IRAF ではなく Python の「Astropy」モジュールを使用した解析が行われていくと考えられます。この Astropy は、現状撮像観測には対応していますが、分光観測には対応していません。そのため、分光観測には IRAF を使い続けることになりますが、撮像観測にはもし天での解析についても Python への移行が必要となります。

また、もし天に参加されたことのある方ならご存知かもしれませんが、もし天の実習期間

中の観測可能な時間は非常に限られたものです。その中で、参加高校生の研究テーマに見合うだけの膨大な量のデータの解析を行わなければならず、毎年、SLA が毎夜のように徹夜で作業している光景が当たり前となっていました。これはもし天の存続性という観点や、SLA の健康という観点からも大変問題であり、解決する必要があります。

以上の理由から、

- 「撮像観測での Python を使った解析」と「分光観測での IRAF を使った解析」のわかりやすい解説
- SLA の負担軽減

を目指した解説資料が必要との判断から、東北大天文の SLA が解説資料を作成することとなりました。

1.2　テキストの構成

このテキストでは主に以下のことを解説しています。

- 天文学の基本 → 2 章
- 天文学における可視光観測の基本 → 3 章
  - 撮像観測
  - 分光観測
- もし天における観測 → 4 章
- 天文学における解析の基本 → 5 章
  - 撮像データ解析
  - 分光データ解析
- 天文学におけるデータ考察の基本 → 6 章
- 付録 → 7 章
  - プログラミング環境の構築
  - Python 実行環境の構築
  - IRAF 実行環境の構築
  - バグ取りの翁
  - Makali'i 実習
  - SAOImage DS9 実習

1.3　掲載内容についての諸注意

このテキストで解説されている内容は、東北大学天文学教室の学部 3 年向け通年授業「天体観測」の講義内容や、2011 年度のもし天観測班の方の資料を参考に、書籍等を参照して肉付けしたものになっています。また、添付の Python コードや、別資料の解析コードについては、「天体観測」の講義内容や、その担当教員である板先生の作成したものや、板研究室の資料を参考にしたものとなっています。

従って、文書、コードについては板さんをはじめとした作成者に、著作権を要求しないような広く一般に普及した部分を除いて権利が帰属しますので、無断で外部への配布を行ったり、公的な文書への転載を行ったりしないように注意してください。これは、万が一このテキストの内容に間違いがあったりした場合に責任を取りきれなくなってしまうためです。

また、この文書には文書内の相互リンク機能がついています。具体的には「2 章」の数字部分をクリックすると、2 章の初めのページにリンクされるようになっています。その他にも節のラベルや、図、数式のラベルにも同様のリンクがついています。適宜活用して頂けると幸いです。

第 2 章

天文学の基本

2.1　天体の種類

2.1.1　惑星

2.1.2　恒星

2.1.3　分子雲

2.1.4　コンパクト天体

2.1.5　銀河/銀河団

2.2　天文学の観測的方法

2.3　電磁波とは

2.4　天体からの放射

2.4.1　連続光

2.4.2　輝線/吸収線

第 3 章

天文学における可視光観測の基本

第 4 章

もし天における観測

第 5 章

天文学における解析の基本

観測によって得られたデータをもとに科学的な議論を行うためには、適切な調理を行うことで、必要な情報が得られるようにしなければなりません。ここでは実際のデータを例にしてデータ解析の基本をおさえていきたいと思います。

解析の流れは図 5.1 のようになります。3 章の流れに則って得られた観測データのことを、ここでは**生データ**と呼ぶことにします。生データに自分の欲しい天体からの光だけが入っていれば苦労することはないのですが、**検出機由来の雑音**や**夜光**などの影響を受けています。図 5.2 がそれを模式的に表したものです。実際に式的にその影響を表すと

$$(\text{生データ}) = (\text{感度の違い}) \times \big\{(\text{真の天体信号}) + (\text{暗電流}) + (\text{夜光などの雑音})\big\} \quad (5.1)$$

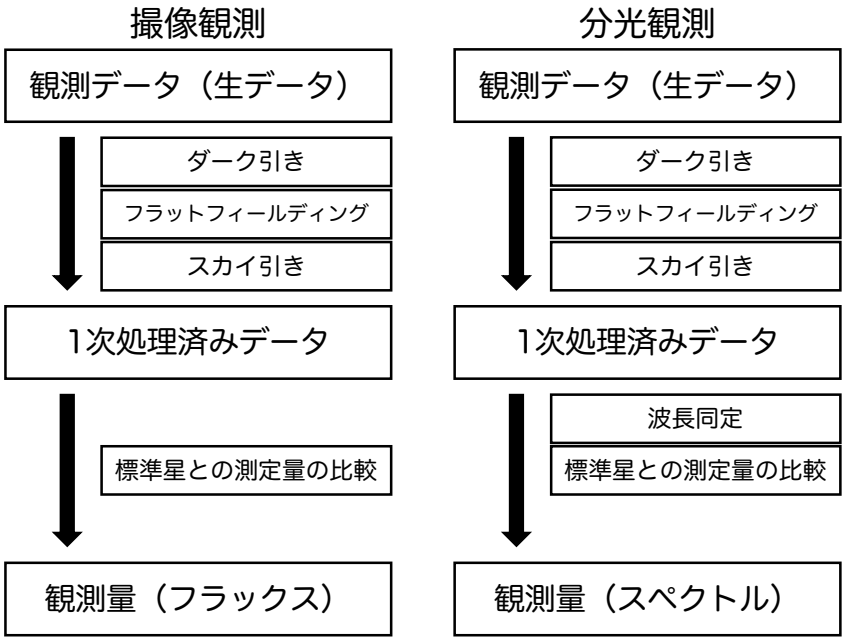

図 5.1　撮像/分光観測データの解析の流れ

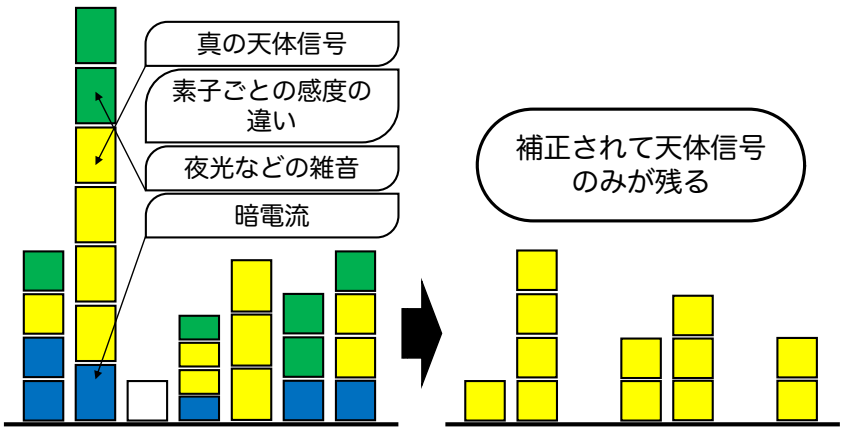

図 5.2　1 次処理の仕組み。黄色が天体からの光、青が暗電流（ダーク）、緑がスカイ成分を表す。また、大きさの違いは検出機の素子の感度が場所によって異なることを表し、色がついていないのはその素子が死んでいることを表す。このような、観測したい天体のみの情報を使って議論をしたいにも関わらず、混ざってしまう雑音などを取り除く作業が 1 次処理である。

のようになります。

暗電流とは、検出機が温度を持っていることで流れてしまう電子雑音のことです。天体放射の検出の仕組みは大まかに

$$(\text{天体からの光子}) \xrightarrow{\text{検出器}} (\text{光電効果による電子}) \to (\text{光電流の発生})$$

となっています。この二つ目の段階で、天体放射やその他の雑音に加えて、熱雑音による電流が加わります。これを「**暗電流**」と呼びます。この雑音は単純に検出器を冷やせば減らすことができます。実際、赤外線検出器は $-76\,\mathrm{K}$、市販の可視光の検出器は $-10\,\mathrm{K}$ に冷やして使われています。しかし、天体からの放射がわからなくなるほどの雑音の影響を軽減することはできても、完全に 0 にすることは不可能なため、この成分を引く作業が必要となります。この作業を「暗電流」が「**ダーク**」と呼ばれることから、「**ダーク引き**」と呼びます。

CCD カメラなどの検出機は、多数の素子によって構成されていますが、望遠鏡や、付属する検出器を合わせた系の、光子検出に対する感度は必ずしも一様ではありません。とても強く光子に対して反応する素子も存在する一方で、全く反応しない、「死んだ素子」も存在します。このような感度の違いを補正する処理を**フラットフィールディング**と呼びます。

「**夜光などの雑音**」とはその名の通り、夜光や大気による雑音の成分です。今回考える主な成分は

- 主に K バンドより長い波長で、大気や望遠鏡の熱的な放射成分を見てしまう ($\sim 300\,\mathrm{K}$)
- 主に H バンドで、OH 夜光の輝線成分によるフリンジパターン

です。これを以降は「**スカイ**」と呼び、スカイを引く補正のことを、「**スカイ引き**」と呼ぶことにします。

これらの補正のことをまとめて **1 次処理**と呼びます。撮像、分光のどちらでもこの処理が必要になりますが、分光はまとめて IRAF でやってしまった方が楽なので、撮像では Python、分光では IRAF を使った処理を行い、観測量を得るにはどうすればよいかをまとめていきます。

5.1 撮像データ解析

今回サンプルとして扱うデータは以下の URL に置いてあります。

```
https://drive.google.com/drive/folders/1ZGkWFySTgc-MVdViP_U_AfD8XPUrSbWi
```

中身は次のようになっています。

```
dark % ls
ir0029.fits   ir0030.fits   ir0031.fits   ir0032.fits   ir0033.fits
flat % ls
ir0006.fits   ir0007.fits   ir0008.fits   ir0009.fits   ir0010.fits
raw % ls
ir0011.fits   ir0012.fits   ir0013.fits   ir0014.fits   ir0015.fits
ir0016.fits   ir0017.fits   ir0018.fits   ir0019.fits
```

5.2 分光データ解析

第 6 章

天文データ考察の基本

第 7 章

終わりに

参考文献

[1] NOAO. Iraf - image reduction and analysis facility, 2019. http://ast.noao.edu/data/software.

付録

A　プログラミング環境の構築

もし天に限らず、データ解析にはプログラミングが必須となりますが、まず環境がなければ解析を行うことはできません。ひと口にプログラミングと言っても、様々な言語が存在し、それぞれに特色があります。例えば Python の実行環境には Anaconda 社提供の「Anaconda」[*1]や Google 社提供の「Google Colaboratory」[*2]などの有名なものがありますが、ここでは統合開発環境をおすすめしたいと思います。Anaconda や Google Colaboratory は Python に特化した環境になっていますが、統合開発環境は自分で拡張機能を追加することによって、自分好みのスタイルで Python だけでなく、C/C++ や HTML/CSS、LaTeX などの様々な言語を扱うことができるものです。統合開発環境にも同じく様々なものがありますが、その中でも特に有名と言える、Microsoft 社提供の「Visual Studio Code」通称「VSCode」[*3]をおすすめします。VSCode は 2015 年が最初のリリースと、非常に新しいエディタですが、さすが Microsoft というべきか、今では非常に多くの人に使われています。個人的には Github 社提供の「Atom」という統合開発環境が好きでずっと使っていたのですが、Github 社が Microsoft 社に買収された影響もあって、2022 年の 12 月で開発終了にまで追い込まれてしまいました。

A.1　VSCode のインストール

まずはインストールしてみましょう。脚注の URL で検索すると、インストール画面が出てきます。図 A.1 がその画面です。どういう仕組みなのかはわからないのですが、自分の PC 環境にあった VSCode のバージョンが「Code editing. Redefined.」の下に出ているはずです。私の環境は intel チップの Mac なのでこの表示になっていますが、自分の PC 環境に本当に合っているのかどうか確かめてからインストールしましょう。Mac ではない場

[*1] https://www.anaconda.com/products/individual
[*2] https://colab.research.google.com
[*3] https://code.visualstudio.com/

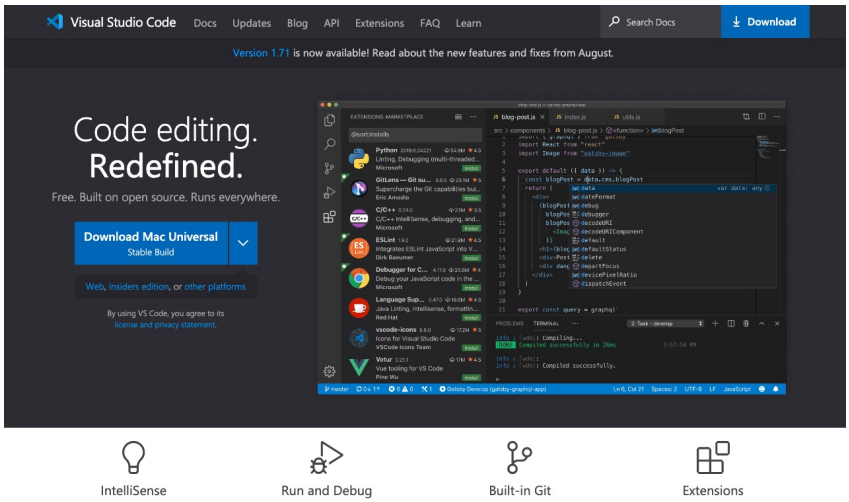

図 A.1　VSCode のインストール画面

合は異なる表示になっていると思います。

ダウンロードが終わるとダウンロードフォルダに `VSCode-darwin-universal.zip` みたいな zip ファイルがあるはずです。これをクリックすると VSCode というアプリケーションが解凍されます。これをアプリケーションフォルダに移動すれば、インストール完了です。

A.2　各種設定

次に、使用に便利な一般の拡張機能を入れてみましょう。ブラウザで「VSCode 拡張機能 おすすめ」とか調べるととても参考になる記事が多いですが、そんな中から筆者が実際に使っているものをピックアップしていくつか紹介します。

vscode-icons

vscode-icons は VSCode で表示されるファイル類のアイコンをおしゃれにしてくれる機能です。アイコンがあると作業の際に視覚的にファイルを区別できて便利だと思います。

indent-rainbow

indent-rainbow はコード内のインデントを色分けして表示してくれる機能です。コーディングの際にはインデントを使って視認性を高くすることが重要です。また、Python ではインデントがあることが `for` 文や `if` 文などで重要となります。そのインデントのレベルをわかりやすくしてくれます。

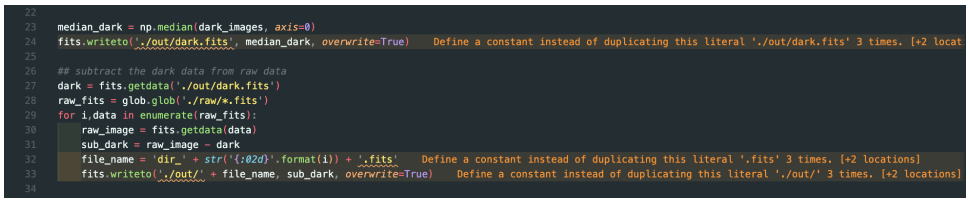

図 A.2　Error Lens によって表示された警告メッセージ

Code Spell Checker

Code Spell Checker は、コードのスペルミスを指摘してくれる機能です。若干曲者で、Python の正しいコマンド名なのにスペルミスを指摘されることがあったり、固有名詞に対してスペルミスを指摘してきたりしますが、ユーザー辞書みたいなのに追加すれば対処できます。コードエラーの原因は些細なスペルミスだったりするので、便利な機能です。

Error Lens

Error Lens は警告などをわかりやすく表示してくれる機能です。例えば図 A.2 のような感じで警告を表示してくれます。

Path Intellisense

Path Intellisense はディレクトリ名の補完をしてくれます。Python に限らず、コード内で相対的にディレクトリ指定を行うことは非常に多いので

Prettier

Prettier

SonarLint

SonarLint

B　Python 実行環境の構築

C　IRAF 実行環境の構築

D　バグ取りの翁

E　Makali'i 実習

F　SAOImage DS9 実習