

Modul Praktikum Pemodelan Iklim

Akhmad Faqih, Perdinan, Agung Baruna Setiawan Noor

2023

Daftar Isi

	1
1	3
1. Dynamical Downscaling: Model Weather Research Forecasting (WRF)	3
1.1 Pendahuluan	3
1.2 Pemrograman Bash	7
1.2.1 Pemrograman Bash pada Sistem Operasi Windows	7
1.2.2 Pemrograman Bash pada MacOS	9
1.2.3 Dasar-dasar Pemrograman Bash	11
1.3 Instalasi Software Pengolahan Data dan Model WRF	14
1.3.1 Instalasi Software Pengolahan Data	14
1.3.2 Persiapan	17
1.3.3 Data dan Software Pendukung	19
1.3.4 Instalasi Software Compiler	20
1.3.5 Instalasi Package	23
1.3.6 Instalasi WRF	28
1.3.7 Instalasi WRF Pre-Processing (WPS)	31
1.4 Menjalankan Simulasi WRF-ARW	33
1.4.1 Program WPS	35
1.4.2 Program WRF	40
1.5 Visualisasi Luaran WRF	48
1.5.1 Python	49
1.5.2 Python (tanpa wrf-python)	55
1.5.3 R	57
1.5.4 NCL	59
1.5.5 QGIS	61
1.5.6 Julia	65
2	69
2. Model Pendugaan Radiasi Matahari	69
2.1 Pendahuluan	69
2.2 Pendugaan Radiasi Matahari Insolasi Harian	69
Perhitungan	70
2.3 Model Pendugaan Ball et al. (2004)	72
Pengolahan Data	73
2.4 Model Pendugaan Hunt et al. (1998)	80
Pengolahan Data	82
3	91
3. Koreksi Bias Statistik	91
3.1 Pendahuluan	91
3.2 Metode Delta	92
3.3 Metode Distribusi Statistik	93

Daftar Isi

3.4	Sebelum Mulai	94
3.5	Pengolahan Data	95
1.	R	95
2.	Python	99

1 *Dynamical Downscaling: Model Weather Research Forecasting (WRF)*

1.1 Pendahuluan

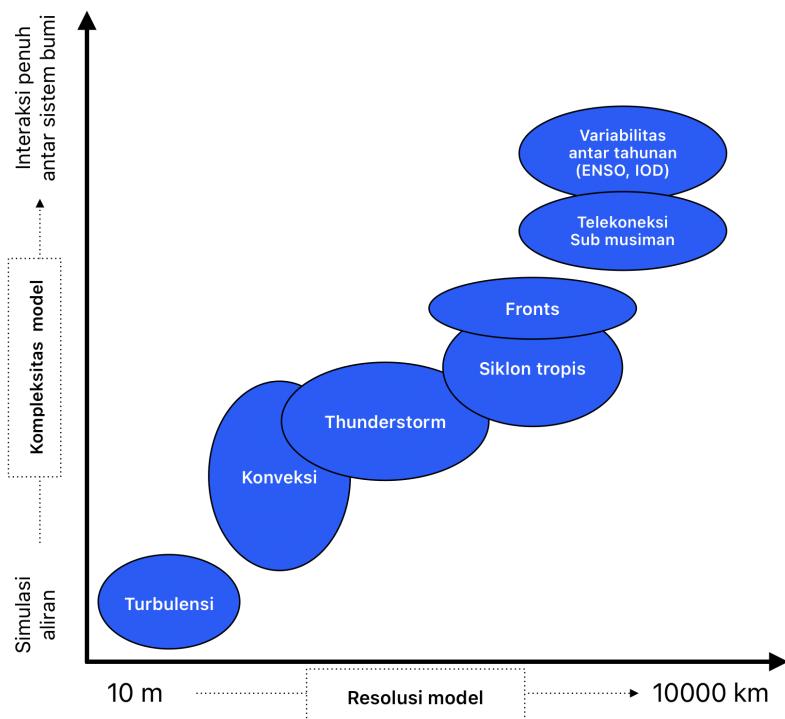
Global Climate Model (GCM) adalah alat yang digunakan dalam mensimulasikan keadaan iklim pada masa lalu, masa sekarang, maupun masa depan. GCM memiliki kemampuan dalam melakukan simulasi variabilitas iklim, sifat-sifat fisis, serta kimia di bumi dengan perhitungan secara matematis yang menggambarkan proses interaksi dan timbal balik pada komponen atmosfer, lautan, dan biotik. Kelemahan dalam GCM adalah ketidakmampuan menangkap kejadian-kejadian iklim pada skala regional maupun lokal karena memiliki resolusi spasial yang kecil, yaitu sekitar >100 km. Ketidakmampuan GCM dalam menjelaskan keadaan iklim secara regional maupun lokal disebabkan oleh keterbatasan sumber daya komputasi. Seiring dengan berkembangnya teknologi komputasi, beberapa instansi seperti Met Office Hadley Center, National Center for Atmospheric Research (NCAR), dan European Centre for Medium-Range Weather Forecasts (ECMWF) telah mengembangkan GCM yang mampu menjelaskan fenomena-fenomena cuaca pada skala regional dengan sumber daya komputer yang sangat tinggi. Teknologi komputasi saat ini sangat memudahkan bagi para pengembang GCM, terutama dalam kecepatan eksekusi algoritme prakiraan cuaca numerik. Contohnya, model prakiraan cuaca dari ECMWF, yaitu Integrated Forecasts System (IFS) memiliki resolusi spasial sekitar 9 km. Apakah dengan berkembangnya GCM yang sudah bisa menjelaskan fenomena regional telah mengantikan Regional Climate Model (RCM)? Tentu saja tidak. Model iklim regional tentu masih dapat digunakan untuk mensimulasikan fenomena cuaca pada skala lokal maupun mikro, sebagai contoh turbulensi dan proses konveksi awan kumulus yang tidak dapat dijelaskan oleh GCM (Gambar 1.1).

Metode *downscaling* merupakan suatu cara dalam mendapatkan informasi spesifik pada suatu wilayah tertentu dengan resolusi tinggi, baik spasial maupun temporal. Metode *downscaling* di dalam ilmu iklim umum digunakan pada aplikasi dalam bidang hidrologi [17], pertanian [5], dan iklim perkotaan [22]. Misalnya dalam bidang hidrologi, teknik *downscaling* digunakan untuk pemodelan debit sungai dan banjir [16]. Wilby dan Wigley [30] mengelompokkan teknik *downscaling* menjadi 4 kategori, yaitu

1. Regresi

Metode regresi merupakan metode *downscaling* paling awal yang telah digunakan pada kajian perubahan iklim. Hal ini dapat dibuktikan dari penelitian oleh Kim pada tahun 1984 [11]. Pendekatan ini secara umum membangun hubungan linier atau non-linier antara parameter titik lokasi dengan prediktor variabel dari resolusi kasar. Contoh dari metode ini adalah regresi linier sederhana, regresi linier berganda, Artificial Neural Network (ANN), regresi komponen utama (Principle Component Regression/PCR), dan lain sebagainya. Sudah banyak penelitian yang menerapkan metode ini untuk kajian perubahan iklim [15, 10, 6, 20].

2. Pola cuaca



Gambar 1.1: Resolusi model iklim dalam mensimulasikan berbagai fenomena cuaca

Metode ini dibangun dari hubungan statistik dari variabel cuaca di stasiun observasi atau rata-rata area dengan klasifikasi cuaca tertentu yang dapat diturunkan secara obyektif maupun subyektif. Metode ini secara obyektif dapat meliputi komponen utama, Canonical Correlation Analyses (CCA), aturan Fuzzy, dan Neural Networks. Contoh prosedur pengelompokan pola cuaca, yaitu European Grosswetterlagen dan British Isles Lamb Weather Types.

3. Stokastik

Model WGEN [19] merupakan contoh dari pendekatan ini. Model ini memiliki kemampuan dalam membangkitkan data curah hujan harian berdasarkan peluang kejadian hujan (hujan dan tidak hujan) dengan rantai Markov order satu. Model ini telah digunakan dalam kajian perubahan iklim dan analisis dampak. Model stokastik yang telah diperoleh dari data observasi deret waktu dapat divalidasi dengan GCM dan perlu dikalibrasi terlebih dahulu.

4. *Limited-area climate models (LAM)*

Metode terakhir untuk melakukan *downscaling* terhadap GCM adalah dengan menyematkan model iklim dengan area terbatas. Sebenarnya, LAM pada saat ini dapat diistilahkan sebagai *Regional Climate Model* (RCM). RCM memiliki resolusi spasial kurang dari 100 km. RCM memerlukan sumber daya komputer yang hampir sama dengan menjalankan GCM. RCM memiliki kemampuan dalam mensimulasikan proses-proses atmosfer pada skala menengah, seperti curah hujan orografis dan konveksi awan atau skala tinggi, seperti turbulensi. Contoh dari RCM adalah Weather Research Forecasting (WRF) [21].

Ada dua cara dalam melakukan metode *downscaling*, yaitu statistik dan dinamik.

Teknik *dynamical downscaling* dapat menjalankan simulasi berbagai proses fisika (termodinamika, kekekalan energi, dan gerak) untuk setiap skala piksel. Teknik ini membutuhkan kondisi batas menggunakan data GCM dan perlu menentukan pemilihan lokasi spesifik. Teknik ini merujuk pada penggunaan RCM untuk meningkatkan skala spasial dan temporal. Model iklim regional meliputi komponen dinamik, fisik, maupun kimia. Komponen dinamik atmosfer memperhitungkan komponen fisika atmosfer yang mencakup proses-proses fisik, seperti radiasi gelombang panjang dan pendek, presipitasi, dan proses pertukaran energi di permukaan bumi atau laut. Proses pada skala sub-piksel dimodelkan oleh berbagai skema parameterisasi yang tentunya dapat berasal dari perhitungan matematika (integral atau diferensial) serta statistik. Model *dynamical downscaling* telah tersedia banyak saat ini, seperti [Weather Research Forecasting \(WRF\)](#), [ICTP Regional Climate Model \(RegCM\)](#), Hadley Centre Regional Climate Model (HadRM), [The Regional Climate Model \(REMO\)](#), dan [The Model for Prediction Across Scale \(MPAS\)](#). Model WRF akan dijelaskan pada modul ini, mulai dari instalasi hingga menampilkan hasil.

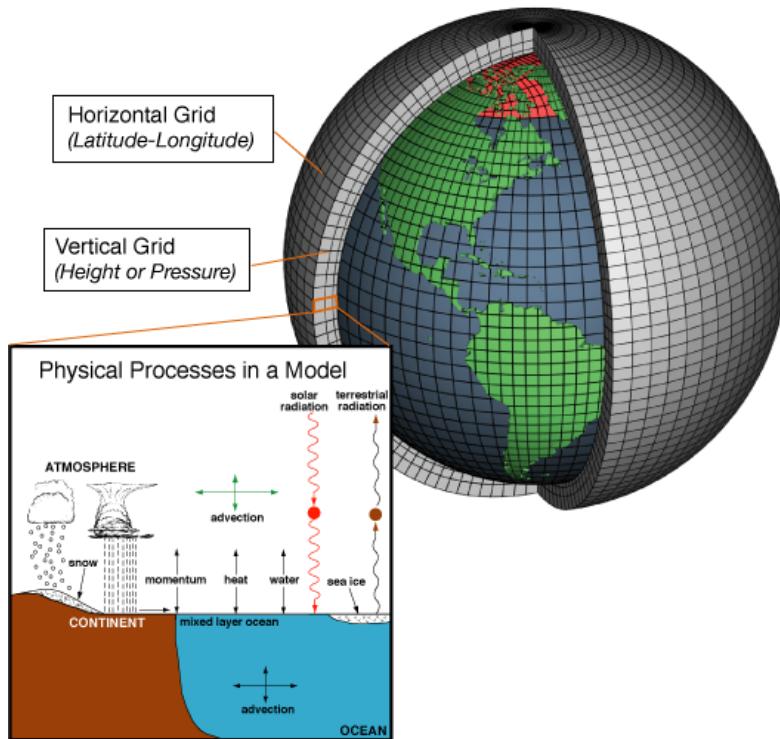
WRF merupakan salah satu model *dynamical downscaling* yang saat ini telah berkembang pesat dengan tujuan riset dan operasional. WRF memiliki spesifikasi dalam berbagai aplikasi prediksi di dalam sistem bumi, seperti kimia atmosfer, hidrologi, kebakaran hutan, siklon, dan iklim regional. Selain simulasi dalam skala regional, WRF telah mampu menjalankan simulasi fenomena cuaca mikro cukup akurat, seperti turbulensi [27] [31] [3]. Sebanyak lebih dari 36.000 pengguna WRF tersebar di 162 negara, termasuk Indonesia yang telah menggunakan WRF untuk kebutuhan operasional [18]. Ada dua jenis model WRF berdasarkan penyelesaian persamaan aliran fluida atmosfer, yaitu *Advanced Research WRF* (ARW) dan *Nonhydrostatic Mesoscale Model* (NMM). WRF-ARW dikembangkan oleh National Centre of Atmosphere Research (NCAR), sedangkan WRF-NMM dikembangkan oleh National Centre of Environmental Prediction (NCEP) [21]. Secara umum, WRF mampu melakukan simulasi beberapa komponen *Numerical Weather Prediction* (NWP) Gambar 1.2. Berdasarkan aplikasi tertentu, WRF memiliki beberapa turunan model, seperti WRF-Chem (kimia atmosfer), WRF-Hydro (hidrologi), dan WRF-Fire (kebakaran hutan dan lahan).

WRF diimplementasikan dalam bahasa pemrograman komputer. Jika Anda melihat repositori github [WRF](#), jenis bahasa pemrograman yang paling banyak digunakan adalah Fortran yang berisikan algoritme berbagai perhitungan fisika dan kimia. Model WRF dapat dijalankan pada personal komputer hingga super komputer. Sebenarnya, Anda dapat menjalankan WRF dengan hanya 1 prosesor, tetapi mungkin membutuhkan waktu lama akibat dari resolusi spasial dan temporal yang tinggi, serta cakupan pemilihan wilayah yang cukup luas. Proses perhitungan model dinamik maupun fisik akan lebih cepat apabila memakai banyak prosesor atau yang bisa disebut dengan **komputasi paralel**. Kerangka kerja perangkat lunak WRF mendukung komputasi paralel yang efisien pada berbagai platform komputasi. Model atmosfer membawa kumpulan komputasi yang sama di setiap piksel vertikal maupun horizontal Gambar 1.3. Kita mungkin telah mengenal jenis prosesor yang tersedia saat ini, seperti Intel dan AMD. Jenis prosesor yang dibuat dapat mempengaruhi kecepatan proses perhitungan. Untuk perhitungan paralel, Anda dapat menggunakan tipe prosesor desktop, mobile (prosesor di laptop), maupun server, tetapi juga perlu memperhatikan banyaknya *core* maupun frekuensi yang tertera pada setiap tipe prosesor.

Terkait dengan bahasa pemrograman yang digunakan di dalam WRF, yaitu bahasa Fortran dan C, kode skrip terlebih dahulu harus dikompilasi dengan program Compiler. Compiler ini bertujuan mengubah bahasa dari kode yang diketik menjadi bahasa mesin. Ada berbagai macam program Compiler yang tersedia secara gratis, salah satunya adalah [GNU Compiler Collection \(gcc\)](#). Selain GNU, perusahaan seperti [Intel](#), [AMD](#), maupun [NVIDIA](#) juga memiliki program Compiler yang dapat digunakan secara gratis serta mendukung komputasi paralel. Di dalam modul praktikum ini, Anda dapat memilih salah satu dari dua Compiler yang akan dijelaskan, yaitu **GNU** atau **Intel**. Kami mencoba melakukan simulasi



Gambar 1.2: Proses secara umum perlu diparameterisasi di dalam model iklim [23]



Gambar 1.3: Ilustrasi Model Iklim

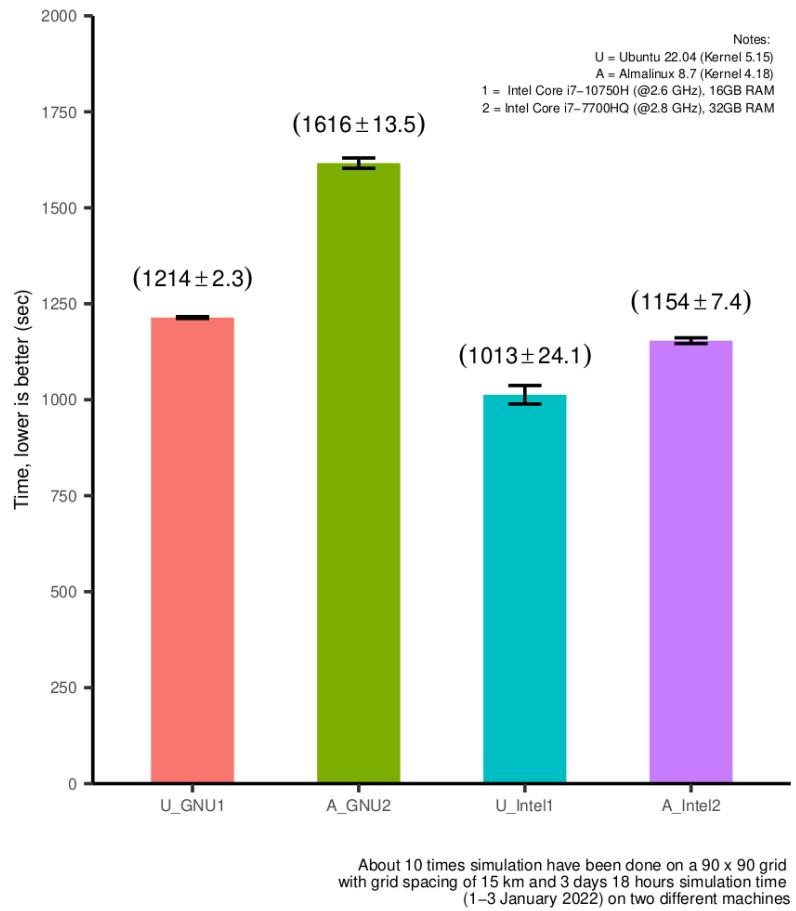
WRF dengan 10 kali ulangan pada dua tipe Compiler dan Prosesor. Perbedaan penggunaan tipe Compiler serta Prosesor dapat mempengaruhi waktu simulasi Gambar 1.4.

1.2 Pemrograman Bash

Sebagai salah satu *tool scripting* yang umum digunakan pada Linux, Bash (**Bourne Again Shell**) sangat bermanfaat bagi para penggunanya. Sebagian besar dalam menjalankan model iklim, Bash digunakan untuk menjalankan instalasi, simulasi, sampai pada analisis data. Pengguna model iklim disarankan mempelajari dasar-dasar pemrograman ini agar memahami berbagai perintah dari cara kerja instalasi dan simulasi dari model iklim tertentu. Para pengembang model iklim biasanya menyediakan dokumen *User Guide*. File ini berisi mengenai cara instalasi, simulasi, informasi file, dan berbagai aplikasi tambahan dalam mengoperasikan/menjalankan model iklim tersebut. Bagi para pengguna Windows atau MacOS, perhatikan subbab di bawah ini.

1.2.1 Pemrograman Bash pada Sistem Operasi Windows

Bagi Anda yang memiliki sistem operasi Windows, Anda diharuskan memasang Windows Subsystem Linux (WSL) untuk dapat menjalankan Bash. Untuk menjalankan WRF, Anda memerlukan sistem operasi berbasis Linux dengan distribusi yang tersedia saat ini. Pada modul ini, distro Ubuntu dipilih untuk simulasi WRF. WSL dapat digunakan bagi pengguna Windows 10/11. WSL dapat dijalankan pada Windows 10/11 dan Windows Server 2019 dengan versi minimum 1803. Saat ini, WSL versi 2 (WSL-2)



Gambar 1.4: Perbedaan waktu simulasi WRF pada tipe Compiler, Distro, dan Prosesor yang digunakan

telah tersedia yang dapat berjalan pada Windows 10 versi 1903 ke atas. Kami merekomendasikan Anda untuk memasang WSL-2 daripada WSL-1 karena prosesnya lebih cepat. Untuk mengaktifkan WSL, Anda dapat mengikuti beberapa langkah berikut ini.

1. Unduh WSL-2 pada [halaman ini](#) dan Install.
2. Buka menu Control Panel dan cari Program and Features
3. Klik Turn Windows Features On or Off
4. Scroll ke paling bawah. Kemudian, aktifkan tanda centang pada **Windows Subsystem for Linux**, **Windows Hypervisor Platform**, dan **Virtual Machine Platform**. Sebagai catatan, Anda perlu memeriksa apakah laptop telah teraktivasi fitur *Virtualization Technology*. Untuk memeriksanya, Anda perlu masuk ke menu BIOS (setiap merk laptop memiliki cara yang berbeda, seperti menekan tombol F2 / Fn + F2 pada merk ASUS).
5. Tunggu proses *update* sampai selesai, kemudian klik **Restart Now** (pastikan simpan data-data pekerjaan dan tutup semua aplikasi)
6. Setelah proses *restart* selesai, cari aplikasi dan unduh Ubuntu pada Microsoft Store
7. Buka aplikasi Ubuntu yang telah terunduh dan tunggu konfigurasi selesai
8. Masukkan Username dan Password (**Catatan:** disarankan sama dengan user dan password Windows 10/11, tetapi boleh beda. Hasil ketikan password **tidak** muncul di terminal)
9. Ketikkan pada terminal

```
sudo apt update
sudo apt upgrade -y
```

10. Anda juga dapat membuka terminal Ubuntu pada terminal Windows PowerShell atau Command Prompt dengan mengetik **wsl** atau **wsl.exe**

Selain WSL, Anda juga dapat menggunakan [Cygwin](#) atau [MinGW](#), hanya saja Anda perlu memilih beberapa paket yang akan digunakan. Tentu cara ini sangat rumit dan membutuhkan waktu cukup lama saat instalasi.

1.2.2 Pemrograman Bash pada MacOS

Aplikasi terminal pada MacOS sudah mendukung Bash, untuk arsitektur arm64 (MacBook M1/M2/M3) maupun Intel. Anda dapat mengikuti atau melewati langkah ini dalam memasang aplikasi [Docker](#) yang digunakan dalam pemasangan Model WRF.

1. Unduh dan pasang Xcode dengan perintah berikut.

```
xcode-select --install
```

Setelah proses instalasi selesai, Anda dapat mengecek versi Xcode dengan perintah berikut.

1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)

```
xcode-select -p
```

Jika versi Xcode sudah muncul, maka proses instalasi sudah selesai.

2. Unduh dan pasang Docker untuk MacOS berdasarkan prosesor pada link di bawah ini.

- [MacOS Intel](#)
- [MacOS M1 atau M2](#)

3. Buka aplikasi Docker melalui menu atau melalui terminal dengan perintah berikut ini.

```
open -a Docker
```

Jendela Docker Desktop akan muncul.

4. Kemudian, tutup aplikasi Docker Desktop. Perintah ini tidak memberhentikan proses Docker, hanya menutup mode GUI-nya saja. Aplikasi Docker masih terus berjalan apabila ada *icon* Docker pada notifikasi.



Gambar 1.5: Icon Docker

5. Buka terminal
6. Ketikkan perintah berikut untuk mengunduh *image* Ubuntu versi 20.04

```
docker pull ubuntu:20.04
```

7. Untuk menjalankan *image*, ketikkan perintah berikut.

```
docker run -it --name wrf-ubuntu ubuntu:20.04
```

Perintah ini menjalankan *image* Ubuntu versi 20.04 dan masuk ke *container* dengan nama *wrf-ubuntu*. Anda dapat mengetikkan perintah-perintah Bash di dalam *container* ini.

8. Terdapat beberapa perintah yang belum tersedia seperti pada Ubuntu versi Desktop maupun WSL. Misalnya perintah `nano`, `wget`, `bash-completion`, dan `git`. Untuk mengunduhnya, ketikkan perintah berikut.

```
apt update  
apt -y install nano wget git bash-completion
```

i Catatan

Untuk selanjutnya, Anda tidak perlu mengetik perintah `sudo` (seperti pada WSL) di dalam terminal Docker karena Anda sudah masuk sebagai *root*.

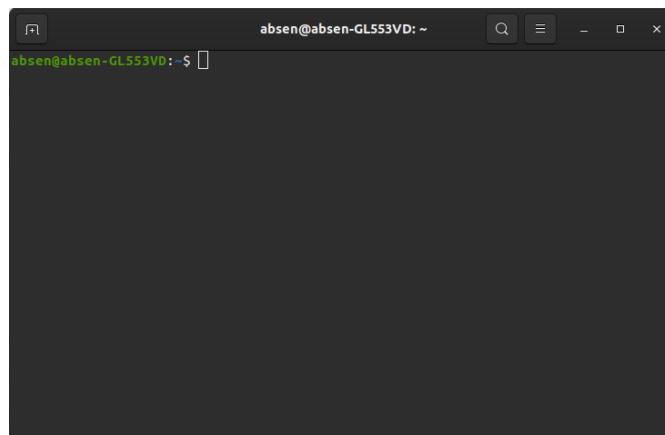
9. Untuk keluar dari *container* Ubuntu, ketikkan perintah `exit` atau dengan shortcut `Ctrl + D`.
10. Jika Anda ingin menjalankan *container* Ubuntu yang sudah dibuat sebelumnya, ketikkan perintah berikut.

```
docker start -i wrf-ubuntu
```

Penggunaan Docker pada MacOS **HANYA** untuk menjalankan WRF saja. Selebihnya pada tahapan analisis/visualisasi data, Anda dapat menggunakan software lain yang masih mendukung MacOS.

1.2.3 Dasar-dasar Pemrograman Bash

Bash merupakan terminal shell umum pada Linux. Anda bisa menjalankan shell lain, seperti ksh (Korn Shell), zsh, dan csh (C shell). Untuk menjalankan Bash, Anda bisa menekan `Ctrl+Alt+T` atau carilah aplikasi Terminal. Khusus Windows 10, Anda dapat mencari aplikasi “Ubuntu” atau “wsl.exe”. Tampilan Bash seperti pada Gambar 1.6. Tulisan `absen@absen-GL553VD` menunjukkan nama user dan tanda `~` berarti menunjukkan lokasi folder saat ini.



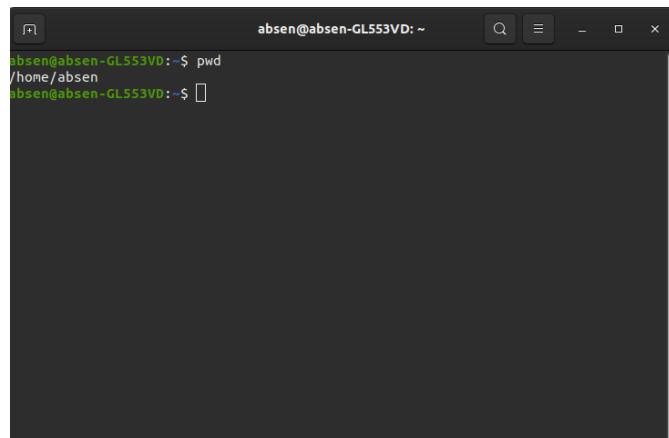
Gambar 1.6: Bash pada Ubuntu

Jika Anda mengetik perintah `pwd` pada Bash, hasilnya adalah seperti di bawah ini atau bisa dilihat pada Gambar 1.7.

Perintah dasar lainnya yang sering digunakan oleh pengguna Linux adalah `ls` atau `ll` (Gambar 1.8). Perintah ini berfungsi untuk melihat nama folder dan file yang berada di direktori pada lokasi saat ini. Untuk berpindah lokasi folder, Anda dapat mengetik perintah `cd` dan pilih folder yang ingin dibuka, contohnya `cd Documents/` atau `cd Documents` (Gambar 1.9). Anda dapat kembali ke folder sebelumnya dengan mengetik perintah `cd ..`

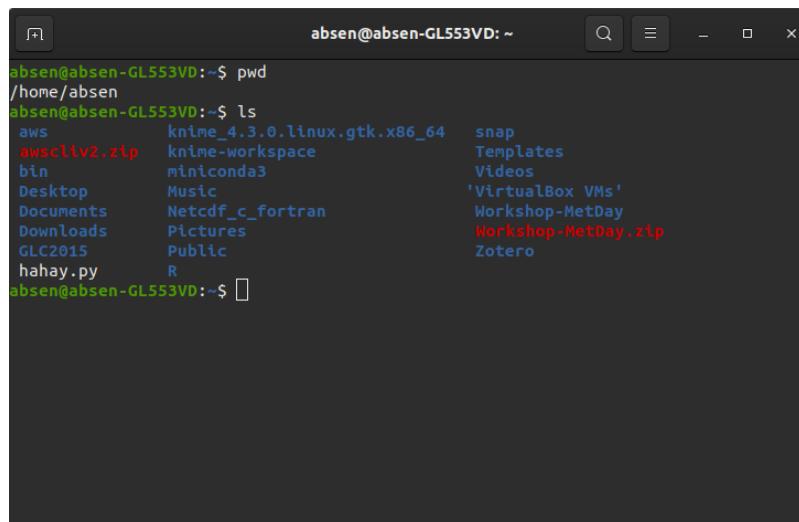
Anda dapat membuat folder baru dengan perintah `mkdir`, memindahkan folder atau file dengan `mv`, menghapus file dengan `rm`, menghapus folder dengan `rmdir`, membuat file baru dengan `touch`, dan

1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)



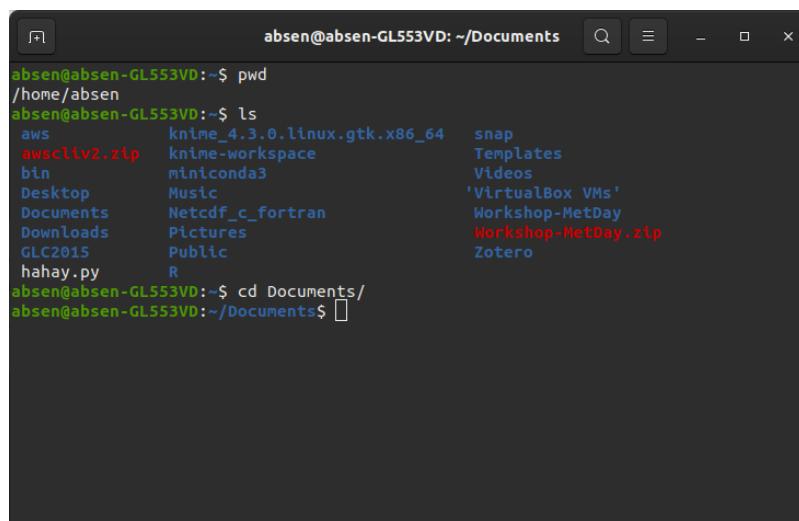
```
absen@absen-GL553VD:~$ pwd
/home/absen
absen@absen-GL553VD:~$ 
```

Gambar 1.7: Perintah `pwd` pada terminal Bash



```
absen@absen-GL553VD:~$ pwd
/home/absen
absen@absen-GL553VD:~$ ls
aws           knime_4.3.0.linux.gtk.x86_64    snap
awscliv2.zip   knime-workspace                Templates
bin           miniconda3                     Videos
Desktop       Music                         'VirtualBox VMs'
Documents     Netcdf_c_fortran              Workshop-MetDay
Downloads     Pictures                      Workshop-MetDay.zip
GLC2015       Public                        Zotero
hahay.py      R
absen@absen-GL553VD:~$ 
```

Gambar 1.8: Perintah `ls` pada terminal Bash



```
absen@absen-GL553VD:~$ pwd
/home/absen
absen@absen-GL553VD:~$ ls
aws           knime_4.3.0.linux.gtk.x86_64    snap
awscliv2.zip   knime-workspace                Templates
bin           miniconda3                     Videos
Desktop       Music                         'VirtualBox VMs'
Documents     Netcdf_c_fortran              Workshop-MetDay
Downloads     Pictures                      Workshop-MetDay.zip
GLC2015       Public                        Zotero
hahay.py      R
absen@absen-GL553VD:~$ cd Documents/
absen@absen-GL553VD:~/Documents$ 
```

Gambar 1.9: Perintah `cd` pada terminal Bash

menyalin folder atau file dengan `cp`. Contoh penggunaan beberapa perintah tersebut dapat dilihat sebagai berikut.

- Misalkan kita ingin membuat folder `Tes-folder`

```
mkdir Tes-folder
```

Anda juga dapat membuat lebih dari 1 folder, misalnya `Tes-folder-1` dan `Tes-folder-2`

```
mkdir Tes-folder-1 Tes-folder-2
```

Untuk memastikan ketiga folder tersebut telah dibuat, ketik perintah `ls`

```
ls
```

- Misalkan kita ingin membuat file `tes-file` di dalam folder `Tes-folder`. Perintah `touch` bertujuan membuat file kosong.

```
cd Tes-folder
touch tes-file
ls
```

Kemudian, file `tes-file` kita pindahkan ke direktori sebelumnya dan diganti namanya dengan `tes-file-pindah`

```
mv tes-file ../tes-file-pindah
```

Kembali ke folder sebelumnya

```
cd ..
```

Ketik perintah `ls` untuk memastikan file `tes-file` telah dipindahkan ke direktori sebelumnya dengan berubah nama menjadi `tes-file-pindah`

```
ls
```

- File `tes-file-pindah` dihapus menggunakan perintah `rm`

```
rm tes-file-pindah
ls
```

- Untuk menghapus folder, misalkan `Tes-folder-1`, Anda dapat menggunakan perintah `rm` dengan ditambahkan parameter `-rf`.

```
rm -rf Tes-folder-1
```

🔥 Perhatian

Hati-hati dalam menggunakan parameter `-rf` karena folder yang telah dihapus beserta file-file di dalamnya **tidak** berada di dalam *Recycle Bin*

1.3 Instalasi Software Pengolahan Data dan Model WRF

Pada tahapan ini, Kami menjelaskan mengenai tahapan instalasi aplikasi pengolahan data pendukung untuk memproses data luaran WRF serta instalasi model WRF di Ubuntu versi 20.04 (Ubuntu versi Desktop dan WSL) maupun MacOS Ventura. Saat kami menguji WRF di Ubuntu maupun MacOS Ventura, proses instalasi serta simulasi tidak mengalami masalah.

1.3.1 Instalasi Software Pengolahan Data

NCAR Command Language (NCL) dan Python

NCL merupakan bahasa pemrograman interpreter yang dikembangkan oleh National Centre of Atmospheric Research (NCAR) dan memiliki kegunaan dalam proses analisis dan visualisasi data-data geosains. Untuk aplikasi pada model WRF, NCL dibutuhkan dalam memvisualisasikan lokasi kajian sebelum disimulasikan. Pengembang WRF telah menyediakan skrip NCL untuk memudahkan dalam ketepatan pemilihan lokasi sesuai dengan keinginan pengguna. Python merupakan bahasa pemrograman general yang memiliki banyak kegunaan, khususnya dalam analisis dan visualisasi data-data dengan tambahan modul (*packages*). Dalam menuliskan kode Python, Anda dapat menggunakan teks editor yang umum digunakan seperti **Jupyter Notebook**. Modul Python yang dipakai di dalam praktikum ini adalah `wrf-python` yang telah dikembangkan oleh NCAR juga. Langkah-langkah pemasangan NCL dan Python sebagai berikut.

- Untuk memasang NCL, Anda harus mengunduh aplikasi Miniconda untuk Linux pada website https://repo.anaconda.com/miniconda/Miniconda3-py39_22.11.1-1-Linux-x86_64.sh dengan perintah pada terminal Bash sebagai berikut.

```
wget
↪ https://repo.anaconda.com/miniconda/Miniconda3-py39_22.11.1-1-Linux-x86_64.sh
```

Anda tidak dapat menggunakan NCL pada sistem operasi Windows, kecuali Anda memiliki Windows Subsystem Linux (WSL).

- Kemudian, lakukan pemasangan Miniconda dengan perintah di bawah ini.

```
bash Miniconda3-py39_22.11.1-1-Linux-x86_64.sh
```

- Selanjutnya tekan ENTER. Terminal akan menampilkan *End-User License Agreement* (EULA), tekan ENTER atau SPACE sampai muncul perintah seperti di bawah ini.

```
Do you accept the license terms? [yes|no]
[no] >>>
```

Kemudian, ketikkan `yes` untuk melanjutkan proses instalasi dan tekan ENTER. Secara otomatis, Python sebenarnya telah terpasang di dalam Miniconda.

4. Tutup terminal Anda dan buka kembali. Perhatikan pada tulisan (`base`) di paling kiri nama user. Jika tulisan tersebut sudah muncul, proses pemasangan Miniconda telah berhasil.
5. Lakukan proses pembuatan *environment* dengan nama `ncl` dan pemasangan NCL beserta package lainnya dengan perintah.

```
conda create -n ncl
conda activate ncl
conda install -c conda-forge jupyter notebook xarray netcdf4 scipy pyngl
    ↳ pynio matplotlib cartopy wrf-python ncl
```

6. Saat Anda ingin memulai menggunakan program NCL, aktifkan terlebih dahulu *environment* `ncl` dengan perintah di bawah ini. Kemudian, tulisan (`base`) menjadi (`ncl`) yang menandakan bahwa *environment* telah berhasil diaktifkan.

```
conda activate ncl
```

7. Untuk membuka **Jupyter Notebook**, gunakan perintah ini. Aplikasi akan muncul pada browser bawaan Anda (Google Chrome, Microsoft Edge, atau Safari).

```
jupyter notebook
```

Untuk menutup Jupyter Notebook, Anda dapat menekan tombol **CTRL+C** pada terminal.

8. Untuk keluar dari *environment* `ncl`, ketikkan perintah `conda deactivate`.

Bagi para pengguna MacOS M1/M2/M3, pemasangan NCL dan `wrf-python` tidak dapat melalui Miniconda. NCL dipasang menggunakan MacPorts, sedangkan `wrf-python` dipasang menggunakan pip. Langkah-langkahnya sebagai berikut.

1. Unduh dan pasang MacPorts pada <https://www.macports.org/install.php>
2. Buka terminal dan ketikkan perintah berikut.

```
sudo port install ncl
```

3. Pada file `~/.bash_profile` atau `~/.zshrc`, tambahkan perintah berikut ini dan letakkan pada baris paling bawah.

```
export NCARG_ROOT=/opt/local
```

4. Selanjutnya, sebelum memasang `wrf-python` dengan pip, Anda perlu membuat *environment* `ncl` dengan conda serta pasang juga python versi 3.10. Secara otomatis, modul pip akan terpasang.
`'bash conda create -n ncl -c conda-forge python=3.10 conda activate ncl'`

5. Kemudian, pasang `wrf-python` serta modul lainnya (`netcdf4` dan `cartopy`) menggunakan `pip`.

```
pip install wrf-python netcdf4 cartopy
```

i Catatan

Environment ncl akan terus dipakai, mulai dari pemasangan WRF hingga analisis dan visualisasi luaran WRF

R dan RStudio

Sama seperti Python dan NCL, R merupakan bahasa pemrograman interpreter, namun dibuat secara khusus untuk analisis dan visualisasi data-data statistik. Beberapa package R telah dikembangkan untuk memudahkan proses analisis data-data cuaca dan iklim, seperti `ncdf4`, `raster`, dan `metR`. `RStudio` umum digunakan dalam mengetik bahasa pemrograman R secara interaktif. Untuk menuliskan skrip R, dibutuhkan aplikasi teks editor, salah satunya adalah RStudio. RStudio mendukung bukan hanya menuliskan skrip R, melainkan bahasa pemrograman lainnya, seperti Markdown, C++, Javascript, dan Python. Sebenarnya, Anda dapat menuliskan skrip Python pada RStudio dengan bantuan package `reticulate`. Hanya saja di dalam modul ini, Kami menggunakan *Jupyter Notebook* untuk menuliskan skrip Python.

Untuk mengunduh R dan RStudio Desktop, Anda dapat menggunakan halaman website di bawah ini sesuai dengan sistem operasi yang Anda gunakan. Bagi pengguna **Windows**, Anda diharuskan mengunduh aplikasi `Rtools` karena beberapa paket tertentu membutuhkan kompilasi dengan Compiler GNU pada saat proses instalasi, seperti `raster`, `tidyverse`, dan `ncdf4`.

- R
 - Windows 10/11: <https://cran.r-project.org/bin/windows/base/R-4.2.2-win.exe>
 - MacOS (Intel): <https://cran.r-project.org/bin/macosx/base/R-4.2.2.pkg>
 - MacOS (ARM, M1/M2): <https://cran.r-project.org/bin/macosx/big-sur-arm64/base/R-4.2.2-arm64.pkg>
 - Ubuntu: ikuti perintah dan langkah-langkah di <https://cran.r-project.org/bin/linux/ubuntu>
- RStudio
 - Windows 10/11: <https://download1.rstudio.org/electron/windows/RStudio-2022.12.0-353.exe>
 - MacOS (Intel/ARM): <https://download1.rstudio.org/electron/macos/RStudio-2022.12.0-353.dmg>
 - Ubuntu 22.04 LTS: <https://download1.rstudio.org/electron/jammy/amd64/rstudio-2022.12.0-353-amd64.deb>

Julia

Bahasa pemrograman ini relatif baru dibandingkan dengan Python dan R. Sama seperti keduanya, Julia merupakan bahasa pemrograman interpreter dan tersedia gratis. Pengembang Julia mengatakan bahwa bahasa ini memiliki kecepatan eksekusi secepat bahasa C/C++. Artinya, kecepatan eksekusi Julia berpotensi lebih cepat daripada Python maupun R. Anda dapat mengunduh Julia di julialang.org sesuai

dengan sistem operasi yang Anda miliki. Untuk membaca data netcdf dari WRF, Anda dapat menggunakan package `NetCDF.jl`. Anda perlu memasang package tersebut pada terminal interaktif Julia. Untuk menuliskan skrip Julia, Anda dapat memanfaatkan **Jupyter Notebook** dan tentunya memerlukan package `IJulia.jl`. Berikut ini adalah langkah-langkahnya.

1. Buka terminal interaktif Julia (julia) dan ketikkan perintah berikut untuk memasang package `NetCDF.jl` dan `IJulia.jl`.

```
using Pkg
Pkg.add("NetCDF")
Pkg.add("IJulia")
```

2. Setelah berhasil memasang package, Anda dapat menuliskan skrip Julia pada Jupyter Notebook. Untuk membuka Jupyter Notebook, ketikkan perintah berikut pada terminal interaktif Julia.

```
using IJulia
notebook()
```

Skrip tersebut akan mengeksekusi instalasi miniconda di dalam Julia dengan disertai **Jupyter Notebook**. Jika instalasi berhasil, **Jupyter Notebook** akan terbuka di browser Anda (Google Chrome, Microsoft Edge, atau Safari). Untuk menutup Jupyter Notebook, Anda dapat menekan tombol **CTRL+C** pada terminal interaktif Julia.

1.3.2 Persiapan

Sebelum menuju ke tahapan instalasi WRF, dasar-dasar perintah Bash mutlak dikuasai. Ini karena model WRF dibagun dengan bahasa C dan Fortran, di mana Anda perlu mengkompilasi *source code* WRF dengan Compiler (GNU/Intel) dan kemudian dapat dijalankan dengan perintah Bash. Langkah-langkah pada subbab selanjutnya, Anda akan disajikan cara mengubah isi di dalam file konfigurasi untuk menjalankan WRF dengan teks editor, yaitu `nano`. `nano` adalah salah satu teks editor yang bekerja pada terminal Bash, sama seperti Notepad pada Windows, hanya saja berbasis pada terminal. Anda juga dapat menggunakan teks editor lainnya, seperti `nvim`, tetapi jika sudah terbiasa. Untuk mulai menggunakan `nano`, Anda hanya cukup mengetikkan perintah di bawah ini. `<nama_file>` adalah nama dari file yang akan diubah atau dibuat jika belum tersedia pada direktori saat ini. Jika ingin keluar, tekan tombol **CTRL + X** dan pilih **Y** untuk menyimpan perubahan atau **N** untuk membatalkan perubahan, kemudian tekan **ENTER**.

```
nano nama_file
```

Ikutilah langkah-langkah di bawah ini agar memudahkan Anda mengerjakan praktikum ini.

1. Buka terminal Bash
 - Untuk Windows 10/11, buka aplikasi Windows Power Shell. Kemudian, ketikkan perintah seperti di bawah ini. Setelahnya, terminal Bash akan muncul.

```
wsl
```

- Untuk Ubuntu atau distro Linux lainnya dan MacOS tanpa Docker, carilah program **Terminal**. Untuk MacOS, Anda dapat menggunakan aplikasi terminal lain seperti **iTerm2**.
- Untuk MacOS M1/M2 yang menggunakan Docker, jalankan *image* Ubuntu versi 20.04 pada **Terminal** dengan perintah di bawah ini.

```
docker run -it --name wrf-ubuntu ubuntu:20.04
```

2. Saat Anda membuka terminal, pastikan lokasi folder saat ini adalah `/home/<user_name>`, di mana `user_name` adalah nama pengguna pada laptop/komputer Anda masing-masing. Perhatikan kembali Gambar 1.7 bahwa untuk melihat lokasi folder Anda sekarang bisa mengetik perintah `pwd` pada terminal atau dengan melihat simbol ~ yang terletak di sebelah kiri \$. Khusus Docker, lokasi direktori saat ini adalah /. Lokasi `$HOME` berada di folder `/root`.

```
cd $HOME
```

3. Buat folder kerja dengan nama `WRF-Model` untuk menampung data-data, package, serta *source code* WRF. Perintah pada terminal Bash adalah sebagai berikut.

```
mkdir WRF-Model
```

4. Kemudian, bukalah folder `WRF-Model` (atau dengan nama lainnya) dengan perintah

```
cd WRF-Model
```

Sekarang, Anda berada di folder `WRF-Model` (Perhatikan ~ berubah menjadi ~/`WRF-Model`)

5. Di dalam `WRF-Model`, buatlah folder `data` guna untuk meletakkan data masukan WRF (data statik dan data cuaca).
6. Unduh *source code* WPS (WRF Pre-Processing) dan WRF pada halaman [Github NCAR](#). *Source code* WRF berisikan algoritma perhitungan fisik dan dinamik atmosfer, sedangkan WPS berisikan algoritma persiapan data masukan atmosfer dari GCM, pemilihan lokasi, dan penentuan periode simulasi. Gunakan perintah di bawah ini untuk mengunduh WPS dan WRF.

```
# 1. Unduh WPS  
git clone https://www.github.com/wrf-model/WPS  
# 2. Unduh WRF  
git clone https://www.github.com/wrf-model/WRF
```

Langkah-langkah di atas masih berlanjut pada subbab selanjutnya.

1.3.3 Data dan Software Pendukung

Software Pendukung

Untuk memasang WRF, Anda perlu menyiapkan perangkat lunak pendukung serta data contoh untuk mensimulasikan WRF. Perangkat lunak tersebut dapat diunduh melalui halaman [Google Drive ini](#) di folder **LIBRARIES**. Setelah terunduh, pindahkan folder tersebut ke dalam folder **WRF-Model**. Hasil unduhan folder dari Google Drive berupa file `*.zip` dengan nama `drive-download-.....zip`. Oleh karena itu, ekstrak file tersebut setelah dipindahkan ke dalam folder **WRF-Model**. Anda dapat menggunakan perintah pada terminal berikut ini.

```
mv $HOME/Downloads/drive-download-* $HOME/WRF-Model # Pada Ubuntu atau MacOS
# mv /mnt/c/Users/<user_name>/Downloads/drive-download-*
↳ /mnt/c/Users/<user_name>/WRF-Model # --> Pada Windows 10/11, dengan
↳ <user_name> adalah nama pengguna
cd $HOME/WRF-Model
unzip drive-download-* # Ganti simbol * menyesuaikan hasil unduhan
mv drive-download* LIBRARIES
```

Data

Data-data masukan untuk WRF telah tersedia dan unduh pada halaman web <https://s.id/wrf-data>. Kami sarankan untuk menggunakan Wi-Fi karena ukuran data cukup besar. Data yang telah diunduh Anda pindahkan ke folder **data** yang berada di dalam **WRF-Model**.

1. Global Forecast System (GFS)

Untuk data masukan yang digunakan berasal dari NOAA yang bernama Global Forecasts System (GFS). GFS memiliki resolusi spasial 0.25° (~ 25 km), 0.50° (~ 50 km), dan 1.00° (~ 100 km). GFS memiliki produk data prakiraan maupun analisis/historis cuaca secara global dengan resolusi temporal per 1 dan 3 jam. Anda dapat mengunduh data ini pada salah satu halaman web berikut.

- **AWS S3:** <https://noaa-gfs-bdp-pds.s3.amazonaws.com>
- **Resarch Data Archive (RDA) NCAR:** <https://rda.ucar.edu/datasets/ds084.1>
- **NCEP Central Operations:** <https://nomads.ncep.noaa.gov/>

Data GFS yang digunakan dalam praktikum ini memiliki waktu 1-3 Januari 2022 pada pukul 00:00, 06:00, 12:00, dan 18:00 UTC dengan resolusi spasial 1.00° yang telah diunduh pada halaman web AWS S3 Bucket.

2. ERA5

Selain GFS, Anda dapat menggunakan data masukan dari institusi lain, seperti ECMWF pada produk ERA5. ERA5 merupakan data reanalisis sehingga hanya memiliki produk historis. Anda dapat mengunduhnya melalui Climate Data Store (CDS) pada halaman <https://cds.climate.copernicus.eu>. ERA5

1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)

memiliki resolusi spasial sebesar 0.25° dengan temporal per 1 jam. Data ERA5 memiliki dua tipe, yaitu data permukaan tanah (*Single Levels*) dan atmosfer untuk setiap ketinggian (*Pressure Levels*). Anda diharuskan mengunduh dua tipe data ini dengan variabel yang dipilih adalah sebagai berikut.

- ERA5 hourly data on Pressure Levels:

geopotential temperature	relative humidity u-component wind	specific humidity v-component wind
--------------------------	------------------------------------	------------------------------------

- ERA5 hourly data on Single Levels:

10m u-component of wind	10m v-component of wind	2m dewpoint temperature
2m temperature	land sea mask	mean sea level pressure
sea ice cover	sea surface temperature	skin temperature
snow depth	soil temperature level 1	soil temperature level 2
soil temperature level 3	soil temperature level 4	surface pressure
volumetric soil water layer 1	volumetric soil water layer 2	volumetric soil water layer 3
volumetric soil water layer 4		

Anda tidak diharuskan mengunduh semua waktu karena CDS membatasi banyaknya permintaan data dari pengguna sehingga Anda tidak dapat mengunduh semua data sekaligus. Data ERA5 yang akan digunakan pada praktikum ini memiliki waktu yang sama dengan GFS.

3. Data Statik WPS

Data statik ini wajib diunduh untuk menjalankan WRF. Isi data ini seperti tipe permukaan lahan, nilai Leaf Area Index (LAI), tipe tanah, dan elevasi permukaan. Anda dapat mengunduhnya di https://www2.mmm.ucar.edu/wrf/src/wps_files/geog_high_res_mandatory.tar.gz. File tersebut berukuran 2 GB dan setelah diekstrak dapat mencapai 30 GB. Data ini sudah tersedia di dalam link <https://s.id/wrf-data>. Untuk mengekstraknya, gunakan perintah berikut.

```
cd data
gunzip geog_high_res_mandatory.tar.gz
tar -xf geog_high_res_mandatory.tar
cd .. # Kembali ke folder WRF-Model
```

Selanjutnya, folder WPS_GEOG akan muncul di dalam folder data/

1.3.4 Instalasi Software Compiler

Instalasi ini meliputi cara melakukan kompilasi dengan dua Compiler berbeda, yaitu GNU dan Intel. Anda dipersilahkan memilih **salah satu cara**. Proses kompilasi untuk semua package yang telah terunduh tidak dapat dilakukan pada Compiler yang berbeda.

GNU

1.3.4.0.1 * WSL, Linux, dan Docker

Bagi pengguna WSL, Ubuntu, serta MacOS yang menggunakan Docker, pasang terlebih dahulu package dependencies sebelum Anda memasuki tahapan instalasi package `zlib`, `libpng`, `jasper`, `openmpi`, `netcdf`, dan `hdf5` dengan perintah di bawah ini.

```
sudo apt -y update && sudo apt -y upgrade
sudo apt -y install gfortran gcc make m4 csh g++
```

1.3.4.0.2 * MacOS

Jika MacOS Anda tidak menggunakan Docker, ikuti langkah-langkah berikut untuk memasang package dependencies.

- Pasang aplikasi `Xcode` terlebih dahulu dan gunakan perintah berikut ini.

```
xcode-select --install
```

Perintah tersebut memunculkan jendela perintah untuk proses pemasangan `Xcode`. Kemudian, klik **Install**. Selanjutnya, muncul jendela **License Agreement** dan klik **Agree**

- Setelah pemasangan `Xcode`, pasanglah aplikasi `homebrew` dengan perintah sebagai berikut (**Perhatikan tipe MacOS Anda!**).

```
# Untuk High Sierra, Sierra, El Capitan
/usr/bin/ruby -e "$(curl -fsSL
  https://raw.githubusercontent.com/Homebrew/install/master/install)"
# Untuk Ventura, Catalina, Mojave, Big Sur
/bin/bash -c "$(curl -fsSL
  https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Setelah menjalankan perintah tersebut, ketikkan Password dan kemudian tekan ENTER.

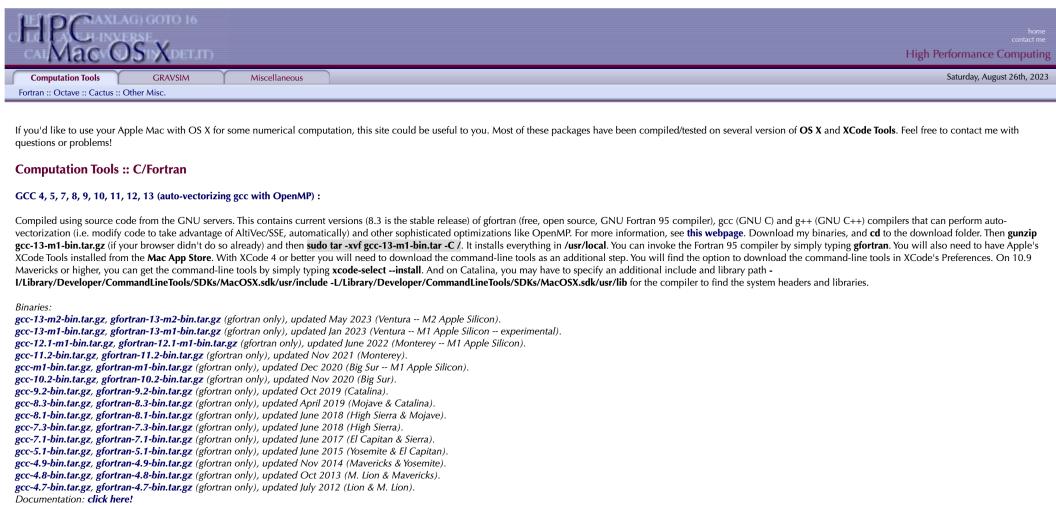
- Setelah `homebrew` berhasil terpasang, perbarui aplikasi tersebut dengan perintah.

```
brew update && brew upgrade
```

- Untuk memasang GNU Compiler Collection (GCC) yang berisikan `gfortran`, `gcc`, dan `g++`, lihatlah website <https://hpc.sourceforge.net/>. Unduh file dengan nama yang berawalan `gcc` dengan menyesuaikan versi MacOS dan arsitektur Macbook Anda. Misalkan pada Mac Anda terpasang MacOS Ventura dengan arsitektur M1 (**gcc-13-m1-bin.tar.gz**).

- Setelah selesai pengunduhan `gcc*`, ekstrak file dengan perintah berikut ini

1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)



Gambar 1.10: Daftar GCC pada MacOS

```
cd $HOME/Downloads  
gunzip gcc-13-m1-bin.tar.gz  
sudo tar -xvf $HOME/Downloads/gcc-13-m1-bin.tar -C /
```

Hasil ekstrak file tersebut disimpan di dalam folder `/usr/local`.

6. Anda dapat mengecek ketiga aplikasi tersebut dengan perintah ini.

```
which gcc gfortran g++
```

Jika terdapat respons seperti di bawah ini, Anda telah berhasil memasangnya.

```
/usr/local/bin/gfortran  
/usr/local/bin/gcc  
/usr/local/bin/g++
```

Intel

Berbeda dengan GNU, pemasangan tipe Compiler Intel agak cukup rumit. Namun, Compiler ini sebenarnya lebih efisien dalam hal kecepatan saat mensimulasikan WRF. Kami sudah membandingkan waktu simulasi dengan Compiler tipe ini lebih cepat dibandingkan dengan GNU pada arsitektur amd64 Gambar 1.4. Anda dapat memperoleh software ini di halaman resmi [Intel](#). Kami sudah menyediakan software tersebut pada tautan [s.id/wrf-intel-compiler](#), baik yang menggunakan Linux (file berawalan `1_`) maupun Mac Intel (file berawalan `m_`). Terdapat 2 file instalasi Compiler. Letakkan 2 file ini di dalam folder `WRF-Model`. Perhatikan langkah-langkah berikut ini.

```
# 0. Package pendukung  
sudo apt -y install make csh
```

```
# 1. oneAPI Base Toolkit Linux
bash l_BaseKit_p_2023.1.0.46401_offline.sh -a -s --eula accept
# 2. oneAPI HPC Toolkit Linux
bash l_HPCKit_p_2023.1.0.46346_offline.sh -a -s --eula accept
```

Untuk instalasi pada Mac Intel, Anda hanya perlu membuka file .dmg dengan melakukan proses instalasi aplikasi pada umumnya. Pemasangan compiler Intel harus sesuai urutan dengan yang pertama adalah `m_BaseKit_p_2023.2.0.49398_offline.dmg` dan selanjutnya adalah `m_HPCKit_p_2023.2.0.49443_offline.dmg`.

Lokasi folder hasil instalasi terdapat di `$HOME/intel`. Perintah untuk memanggil program Compiler Intel belum menjadi **ENVIRONMENT VARIABLE**. Untungnya, Intel menyediakan skrip untuk memanggil semua program Compiler secara default, yaitu terdapat di dalam `$HOME/intel/oneapi/setvars.sh`. Ketika memanggil skrip tersebut, **ENVINRONMENT VARIABLE** milik Miniconda secara otomatis dinonaktifkan (tulisan `<base>` hilang). Perintah untuk memanggil skrip `setvars.sh` adalah sebagai berikut.

```
source $HOME/intel/oneapi/setvars.sh
```

1.3.5 Instalasi Package

Instalasi beberapa package meliputi `zlib`, `libpng`, `jasper`, `hdf5`, `netcdf-c`, dan `netcdf-fortran`. Perlu diperhatikan dan lihat pada terminal bahwa Anda berada di folder `~/WRF-Model`. Masukkan beberapa perintah **ENVIRONMEN VARIABLE** di bawah ini dengan perintah `export`. Anda dapat memasukkan perintah di bawah ini pada file teks dengan format `.sh`, misalnya `environment.sh`. Untuk menjalankannya, gunakan perintah `source environment.sh`. Pendefinisian variabel ini dapat pula dimasukkan ke file `.bashrc` atau `.bash_profile` dengan perintah `nano ~/.bashrc` atau `nano ~/.bash_profile`. Masukkan perintah di atas pada baris paling bawah dan simpan dengan menekan tombol `CTRL + X` dan kemudian `Y` dan `ENTER`.

```
export ODIR=$HOME/WRF-Model
export PATH=$ODIR/bin:$PATH
export LD_LIBRARY_PATH=$ODIR/lib:$LD_LIBRARY_PATH
export LDFLAGS=-L$ODIR/lib
export CPPFLAGS=-I$ODIR/include
export NETCDF=$ODIR
export HDF5=$ODIR
export JASPERLIB=$ODIR/lib
export JASPERINC=$ODIR/include
```

Langkah ini wajib dilakukan pada saat melakukan instalasi package, WRF, dan WPS. Pendefinisian **ENVIRONMENT VARIABLE** ini berlaku untuk GNU maupun Intel.

GNU

Berikut ini langkah-langkah memasang package pada Compiler **GNU**. Buka terlebih dahulu folder **\$ODIR/LIBRARIES** dengan mengetik perintah `cd $ODIR/LIBRARIES`. Kemudian, ikuti langkah-langkah berikut ini.

1. zlib

```
# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf zlib-1.3.tar.gz
# 2. Buka folder hasil ekstrak
cd zlib-1.3
# 3. Konfigurasi
CC=gcc ./configure --prefix=$ODIR
# 4. Instalasi
make check install
```

2. libpng

```
# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf libpng-1.6.40.tar.gz
# 2. Buka folder hasil ekstrak
cd libpng-1.6.40
# 3. Konfigurasi
CC=gcc ./configure --prefix=$ODIR
# 4. Instalasi
make check install
```

3. jasper

```
# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf jasper-1.900.1.tar.gz
# 2. Buka folder hasil ekstrak
cd jasper-1.900.1
# 3. Konfigurasi
CC=gcc ./configure --prefix=$ODIR
# 4. Instalasi
make check install
```

Bagi pengguna MacOS M1/M2 dengan atau tanpa terminal Docker, gunakan perintah berikut ini untuk instalasi Jasper.

```

# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf jasper-1.900.1.tar.gz
# 2. Buka folder hasil ekstrak
cd jasper-1.900.1
# 3. Mengunduh config.guess
wget -N -O acaux/config.guess
↳ "http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD"
wget -N -O acaux/config.sub
↳ "http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD"
# 4. Konfigurasi
CC=gcc ./configure --prefix=$ODIR
# 5. Instalasi
make check install

```

4. openMPI

Jika Anda berencana menjalankan WRF hanya dengan 1 prosesor, Anda dapat melewati langkah ini. Akan tetapi, kami menyarankan untuk memasang program ini untuk mempersingkat waktu simulasi WRF.

```

# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf openmpi-4.1.5.tar.gz
# 2. Buka folder hasil ekstrak
cd openmpi-4.1.5
# 3. Konfigurasi
CC=gcc FC=gfortran ./configure --prefix=$ODIR
# 4. Instalasi
make && make install

```

5. hdf5

```

# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf hdf5-1.14.2.tar.gz
# 2. Buka folder hasil ekstrak
cd hdf5-1.14.2
# 3. Konfigurasi
CC=gcc FC=gfortran ./configure --prefix=$ODIR --enable-fortran
# 4. Instalasi
make && make install

```

6. netcdf-c

```
# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf netcdf-c-4.9.2.tar.gz
# 2. Buka folder hasil ekstrak
cd netcdf-c-4.9.2
# 3. Konfigurasi
CC=gcc ./configure --prefix=$ODIR --disable-dap
# 4. Instalasi
make && make install
```

7. netcdf-fortran

```
# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf netcdf-fortran-4.6.1.tar.gz
# 2. Buka folder hasil ekstrak
cd netcdf-fortran-4.6.1
# 3. Konfigurasi
CC=gcc FC=gfortran ./configure --prefix=$ODIR
# 4. Instalasi
make && make install
```

Intel

Berikut ini langkah-langkah memasang package pada Compiler **Intel**. Buka terlebih dahulu folder `pyWRF-install/libraries` dengan mengetik perintah `cd pyWRF-install/libraries`. Untuk proses instalasi menggunakan Intel, sama saja dengan GNU. Hanya saja, perbedaannya adalah definisi dari variabel CC maupun FC. Untuk Intel, variabel `CC=icc` dan `FC=ifort`. Jangan lupa untuk mengaktifkan **ENVIRONMENT VARIABLE** dari Intel dengan mengetik perintah ini!

```
source ~/intel/oneapi/setvars.sh
```

1. zlib

```
# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf zlib-1.3.tar.gz
# 2. Buka folder hasil ekstrak
cd zlib-1.3
# 3. Konfigurasi
CC=icc CFLAGS='-diag-disable=10441' ./configure --prefix=$ODIR
```

```
# 4. Instalasi
make check install
```

2. libpng

```
# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf libpng-1.6.40.tar.gz
# 2. Buka folder hasil ekstrak
cd libpng-1.6.40
# 3. Konfigurasi
CC=icc CFLAGS='-diag-disable=10441' ./configure --prefix=$ODIR
# 4. Instalasi
make check install
```

3. jasper

```
# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf jasper-1.900.1.tar.gz
# 2. Buka folder hasil ekstrak
cd jasper-1.900.1
# 3. Konfigurasi
CC=icc CFLAGS='-diag-disable=10441' ./configure --prefix=$ODIR
# 4. Instalasi
make check install
```

4. hdf5

```
# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf hdf5-1.14.2.tar.gz
# 2. Buka folder hasil ekstrak
cd hdf5-1.14.2
# 3. Konfigurasi
CC=icc FC=ifort CFLAGS='-diag-disable=10441' ./configure --prefix=$ODIR
    --enable-fortran
# 4. Instalasi
make && make install
```

5. netcdf-c

```
# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf netcdf-c-4.9.2.tar.gz
# 2. Buka folder hasil ekstrak
cd netcdf-c-4.9.2
# 3. Konfigurasi
CC=icc CFLAGS='-diag-disable=10441' ./configure --prefix=$ODIR --disable-dap
# 4. Instalasi
make && make install
```

6. netcdf-fortran

```
# 0. Buka folder libraries
cd $ODIR/LIBRARIES
# 1. Extract
tar -xf netcdf-fortran-4.6.1.tar.gz
# 2. Buka folder hasil ekstrak
cd netcdf-fortran-4.6.1
# 3. Konfigurasi
CC=icc FC=ifort CFLAGS='-diag-disable=10441' ./configure --prefix=$ODIR
# 4. Instalasi
make && make install
```

1.3.6 Instalasi WRF

Proses instalasi WRF membutuhkan waktu agak lama, yaitu sekitar 20-60 menit. Tentunya, ini bergantung pada spesifikasi prosesor yang Anda gunakan, serta tipe Compiler. Ikuti langkah-langkah berikut.

1. Buka direktori WRF yang berada di dalam \$ODIR/WRF dengan perintah.

```
cd $ODIR/WRF
```

2. Jalankan file `configure` dengan perintah.

```
./configure
```

Anda akan disajikan beberapa teks di dalamnya. Anda diperintahkan untuk memilih opsi Compiler. Tipe Compiler tersebut selain GNU dan Intel, ada pula IBM, PGI, Fujitsu, Pathscale, dan CRAY. Jika Anda menggunakan Compiler tipe GNU, ketikkan angka **35**. Untuk Intel, ketik angka **16**. Setelah itu, tekan Enter.

```

└─ ./configure
checking for perl... no
checking for perl... found /usr/bin/perl (perl)
Will use NETCDF in dir: /home/absen/WRF-Model
Will use HDF5 in dir: /home/absen/WRF-Model
HDF5 not set in environment. Will configure WRF for use without.
Will use 'time' to report timing information

If you REALLY want Grid2 output from WRF, modify the arch/Config.pl script.
Right now you are not getting the Jasper lib, from the environment, compiled into WRF.

Please select from among the following Linux x86_64 options:

 1. (serial) 2. (mpar) 3. (dmpar) 4. (dm+smp) PGI (pgf90/gcc)
 5. (serial) 6. (mpar) 7. (dmpar) 8. (dm+smp) PGI (pgf90/pgcc): SGI MPT
 9. (serial) 10. (mpar) 11. (dmpar) 12. (dm+smp) PGI (pgf90/gcc): PGK accelerator
13. (serial) 14. (mpar) 15. (dmpar) 16. (dm+smp) INTEL (ifort/icc)
17. (serial) 18. (mpar) 19. (dmpar) 20. (dm+smp) INTEL (ifort/icc): Xeon Phi (MIC architecture)
21. (serial) 22. (mpar) 23. (dmpar) 24. (dm+smp) INTEL (ifort/icc): Xeon SNB with AVX mods
25. (serial) 26. (mpar) 27. (dmpar) 28. (dm+smp) INTEL (ifort/icc): IBM POE
29. (serial) 30. (mpar) 31. (dmpar) PATHSCALE (pathf90/pathcc)
32. (serial) 33. (mpar) 34. (dmpar) 35. (dm+smp) GNU (gfortran/gcc)
36. (serial) 37. (mpar) 38. (dmpar) 39. (dm+smp) IBM (xlf90_r/c_r)
40. (serial) 41. (mpar) 42. (dmpar) 43. (dm+smp) PGI (ftn/gcc): Cray XC CLE
44. (serial) 45. (mpar) 46. (dmpar) 47. (dm+smp) CRAY CCE (ftn $NODMP$icc): Cray XE and XC
48. (serial) 49. (mpar) 50. (dmpar) 51. (dm+smp) INTEL (iftn/icc): Cray XC
52. (serial) 53. (mpar) 54. (dmpar) 55. (dm+smp) PGI (pgf90/gcc)
56. (serial) 57. (mpar) 58. (dmpar) 59. (dm+smp) PGI (pgf90/gcc): -f90=pgf90
60. (serial) 61. (mpar) 62. (dmpar) 63. (dm+smp) PGI (pgf90/pgcc): -f90=pgf90
64. (serial) 65. (mpar) 66. (dmpar) 67. (dm+smp) INTEL (ifort/icc): HSW/BDN
68. (serial) 69. (mpar) 70. (dmpar) 71. (dm+smp) INTEL (ifort/icc): KNL MIC
72. (serial) 73. (mpar) 74. (dmpar) 75. (dm+smp) FUJITSU (frtpx/fccpx): FX10/FX100 SPARC64 IXfx/Xlfx

Enter selection [1-75] : 35

```



```

└─ ./configure
checking for perl... no
checking for perl... found /usr/bin/perl (perl)
Will use NETCDF in dir: /home/absen/WRF-Model
Will use HDF5 in dir: /home/absen/WRF-Model
HDF5 not set in environment. Will configure WRF for use without.
Will use 'time' to report timing information

If you REALLY want Grid2 output from WRF, modify the arch/Config.pl script.
Right now you are not getting the Jasper lib, from the environment, compiled into WRF.

Please select from among the following Linux x86_64 options:

 1. (serial) 2. (mpar) 3. (dmpar) 4. (dm+smp) PGI (pgf90/gcc)
 5. (serial) 6. (mpar) 7. (dmpar) 8. (dm+smp) PGI (pgf90/pgcc): SGI MPT
 9. (serial) 10. (mpar) 11. (dmpar) 12. (dm+smp) PGI (pgf90/gcc): PGK accelerator
13. (serial) 14. (mpar) 15. (dmpar) 16. (dm+smp) INTEL (ifort/icc)
17. (serial) 18. (mpar) 19. (dmpar) 20. (dm+smp) INTEL (ifort/icc): Xeon Phi (MIC architecture)
21. (serial) 22. (mpar) 23. (dmpar) 24. (dm+smp) INTEL (ifort/icc): Xeon SNB with AVX mods
25. (serial) 26. (mpar) 27. (dmpar) 28. (dm+smp) INTEL (ifort/icc): IBM POE
29. (serial) 30. (mpar) 31. (dmpar) PATHSCALE (pathf90/pathcc)
32. (serial) 33. (mpar) 34. (dmpar) 35. (dm+smp) GNU (gfortran/gcc)
36. (serial) 37. (mpar) 38. (dmpar) 39. (dm+smp) IBM (xlf90_r/c_r)
40. (serial) 41. (mpar) 42. (dmpar) 43. (dm+smp) PGI (ftn/gcc): Cray XC CLE
44. (serial) 45. (mpar) 46. (dmpar) 47. (dm+smp) CRAY CCE (ftn $NODMP$icc): Cray XE and XC
48. (serial) 49. (mpar) 50. (dmpar) 51. (dm+smp) INTEL (iftn/icc): Cray XC
52. (serial) 53. (mpar) 54. (dmpar) 55. (dm+smp) PGI (pgf90/gcc)
56. (serial) 57. (mpar) 58. (dmpar) 59. (dm+smp) PGI (pgf90/gcc): -f90=pgf90
60. (serial) 61. (mpar) 62. (dmpar) 63. (dm+smp) PGI (pgf90/pgcc): -f90=pgf90
64. (serial) 65. (mpar) 66. (dmpar) 67. (dm+smp) INTEL (ifort/icc): HSW/BDN
68. (serial) 69. (mpar) 70. (dmpar) 71. (dm+smp) INTEL (ifort/icc): KNL MIC
72. (serial) 73. (mpar) 74. (dmpar) 75. (dm+smp) FUJITSU (frtpx/fccpx): FX10/FX100 SPARC64 IXfx/Xlfx

Enter selection [1-75] : 16

```

Untuk pengguna Mac M1/M2 tanpa Docker, Anda dapat menggunakan opsi **36**.

- Untuk melakukan instalasi, ketik perintah ini.

```
./compile em_real -j jumlah_prosesor
```

dimana **jumlah_prosesor** adalah jumlah dari prosesor pada laptop/komputer Anda yang akan digunakan untuk proses instalasi dan kompilasi kode-kode WRF. Proses kompilasi akan memakan waktu yang sangat lama apabila Anda hanya menggunakan 1 prosesor. Pastikan berbagai program pada komputer/laptop Anda yang saat ini sedang dibuka, seperti Google Chrome atau Spotify harap ditutup terlebih dahulu karena ini membantu proses instalasi lebih stabil.

- Untuk Compiler Intel**, setelah langkah ke-2 dijalankan, buka file **configure.wrf** dengan perintah.

```
nano configure.wrf
```

Kemudian, scroll ke bawah dengan menekan tombol ↓ pada keyboard dan ubahlah isinya sesuai aturan pada tabel di bawah ini.

Tabel 1.3: Pengubahan variabel DM_FC dan DM_CC

Sebelum	Sesudah
DM_FC = mpif90 -f90=\$(SFC)	DM_FC = mpiifort -f90=\$(SFC)
DM_CC = mpicc -cc=\$(SCC)	DM_CC = mpicc -cc=\$(SCC)

Setelah selesai diubah, keluar dari editor nano dengan menekan tombol **Ctrl + X**

Proses instalasi WRF berhasil dilakukan apabila terdapat file yang berekstensi .exe: `ndown.exe`, `tc.exe`, `real.exe`, dan `wrf.exe` di dalam folder `main`. Anda bisa melihatnya dengan perintah

```
ls main/*.exe
```

```
main/ndown.exe  main/real.exe  main/tc.exe  main/wrf.exe
```

Untuk pengguna **MacOS M1/M2** dengan **Docker**, ikuti langkah-langkah berikut.

1. Buka direktori WRF yang berada di dalam `$ODIR/WRF`.
2. Ubah file `arch/configure.defaults` menggunakan teks editor `nano` dengan menambahkan teks ini.

```
nano arch/configure.defaults
```

```
#####
#ARCH Linux aarch64, Arm compiler OpenMPI # serial smpar dmpar dm+sm#
DESCRIPTION      =      GCC ($SFC/$SCC): Aarch64
DMPARALLEL      =
OMPCPP          =      -fopenmp
OMP              =      -fopenmp
OMPCC           =      -fopenmp
SFC              =      gfortran
SCC              =      gcc
CCOMP            =      gcc
DM_FC            =      mpif90
DM_CC            =      mpicc -DMPI2_SUPPORT
FC               =      CONFIGURE_FC
CC               =      CONFIGURE_CC
LD               =      $(FC)
RWORDSIZE        =      CONFIGURE_RWORDSIZE
PROMOTION        =
ARCH_LOCAL       =
CFLAGS_LOCAL     =      -w -O3 -c
LDFLAGS_LOCAL    =      -fopenmp
FCOPTIM          =      -Ofast -march=armv8.2-a+fp16+rcpc+dotprod+crypto -fopenmp -frecusive
FCREDUCEDOPT    =      $(FCOPTIM)
```

```

FCNOOPT      =      -O0 -fopenmp -frecursive
FCDEBUG      =      -g $(FCNOOPT)
FORMAT_FIXED =      -ffixed-form -ffixed-line-length-0 -Wno-invalid-pch -Wno-argument-mis-
FORMAT_FREE   =      -ffree-form -ffree-line-length-0 -Wno-argument-mismatch -Wno-invalid-p
FCSUFFIX     =
BYTESWAPIO   =      -fconvert=big-endian -frecord-marker=4
FCBASEOPTS   =      -w $(FORMAT_FREE) $(BYTESWAPIO)
MODULE_SRCH_FLAG= -I$(WRF_SRC_ROOT_DIR)/main
TRADFLAG     =      -traditional-cpp
CPP          =      /lib/cpp CONFIGURE_CPPFLAGS
AR           =      ar
ARFLAGS      =      ru
M4           =      m4 -B 14000
RANLIB       =      ranlib
RLFLAGS      =
CC_TOOLS    =      $(SCC)

```

- Jalankan program `configure` dengan perintah ini.

```
./configure
```

Kemudian, pilih nomor 4 untuk memilih opsi **Linux aarch64, Arm compiler OpenMPI # serial smpar dmpar dm+sm**. Setelah itu, tekan Enter. Abaikan saja terhadap respons seperti pada gambar ini.

```
#####
-----
Settings listed above are written to configure.wrf.
If you wish to change settings, please edit that file.
If you wish to change the default options, edit the file:
  arch/configure.defaults
NetCDF users note:
This installation of NetCDF supports large file support. To DISABLE large file
support in NetCDF, set the environment variable WRFCIO_NCD_NO_LARGE_FILE_SUPPORT
to 1 and run configure again. Set to any other value to avoid this message.

Testing for NetCDF, C and Fortran compiler

./configure: 1: file: not found
./configure: 1: file: not found
./configure: 1: file: not found
  One of compilers testing failed!
    Please check your compiler
```

- Selanjutnya, jalankan program `compile` dengan perintah ini.

```
./compile em_real -j jumlah_prosesor
```

1.3.7 Instalasi WRF Pre-Processing (WPS)

Program WPS digunakan untuk menyesuaikan data masukan dari berbagai sumber (ERA5, GFS, NAM, ...) sebelum ke simulasi WRF. Terdapat 3 program utama: `geogrid.exe`, `ungrib.exe`, dan `metgrid.exe`.

1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)

- **geogrid.exe**: memilih lokasi yang akan dilakukan simulasi. Luaran program ini berupa file geo_em* yang berisi nilai-nilai dari variabel di dalam file hasil ekstrak geog_high_res_mandatory.tar
- **ungrib.exe**: mengubah file berformat grib menjadi nc, serta memungkinkan dapat melakukan interpolasi (waktu dan lokasi)
- **metgrid.exe**: menggabungkan luaran dari **geogrid.exe** dan **ungrib.exe** sehingga menghasilkan file met_em* sesuai dengan ketentuan lokasi dan waktu di dalam **namelist.wps**

Proses instalasi WPS tidak membutuhkan waktu yang lama, sekitar 2-5 menit. Untuk melakukan instalasi WPS, ikuti langkah-langkah berikut ini.

1. Saat ini, Anda berada di folder WRF. Buka folder WPS dengan perintah ini.

```
cd $ODIR/WPS
```

2. Jalankan file **configure** dengan perintah ini.

```
./configure
```

Ketik angka **3** untuk GNU, angka **19** untuk Intel, atau angka **19** untuk Mac M1/M2 tanpa Docker.

3. Setelah selesai, di folder WPS akan muncul file **configure.wps**. Beberapa baris dari isi file tersebut ada yang perlu ditambahkan dan diganti dengan ketentuan ini.

- Untuk **Intel**, tambahkan *flags -liomp5* setelah *-lnetcdf* pada bagian variabel **WRF_LIB =** Kemudian, ubah pula **DM_FC** dan **DM_CC** seperti pada Tabel **1.3**.
- Untuk **GNU** serta **Mac M1/M2 tanpa Docker**, tambahkan *flags -lgomp* setelah *-lnetcdf* pada bagian variabel **WRF_LIB =**

4. Setelah diganti, lakukan kompilasi dengan mengetik perintah

```
./compile
```

Proses instalasi berhasil apabila terdapat 3 file .exe: **geogrid.exe**, **ungrib.exe**, dan **metgrid.exe** di folder WPS yang berupa shortcut. Anda dapat menggunakan perintah

```
ls *.exe
```

geogrid.exe metgrid.exe ungrib.exe

Untuk pengguna **MacOS M1/M2/M3** dengan **Docker**, ikuti langkah-langkah berikut.

1. Buka direktori WPS yang berada di dalam **\$ODIR/WPS**.
2. Ubah file **arch/configure.defaults** menggunakan teks editor **nano** dengan menambahkan teks ini.

```
nano arch/configure.defaults
```

```
#####
#ARCH Linux aarch64, Arm compiler OpenMPI # serial smpar dmpar dm+sm#
COMPRESSION_LIBS      = CONFIGURE_COMP_L
COMPRESSION_INC       = CONFIGURE_COMP_I
FDEFS                 = CONFIGURE_FDEFS
SFC                   = gfortran
SCC                   = gcc
DM_FC                 = mpif90
DM_CC                 = mpicc
FC                    = CONFIGURE_FC
CC                    = CONFIGURE_CC
LD                    = $(FC)
FFLAGS                = -ffree-form -O -fconvert=big-endian -frecord-marker=4 -ffixed-line-len
F77FLAGS              = -ffixed-form -O -fconvert=big-endian -frecord-marker=4 -ffree-line-len
FCSUFFIX              =
FNGFLAGS              = $(FFLAGS)
LDFLAGS               =
CFLAGS                =
CPP                   = /usr/bin/cpp -P -traditional
CPPFLAGS              = -D_UNDERSCORE -DBYTE_SWAP -DLINUX -DIO_NETCDF -DBIT32 -DNO_SIGNAL CONF
RANLIB                = ranlib
```

- Jalankan program `configure` dengan perintah ini. Kemudian, pilih nomor 2. Tetap abaikan respons pada tulisan `./configure: 1: file: not found`

```
./configure
```

- Kemudian, ubah isi file `configure.wps` dengan menambahkan `flags -lgomp` setelah `-lnetcdf` pada variabel `WRF_LIB =`
- Selanjutnya, jalankan program `compile` dengan perintah ini.

```
./compile
```

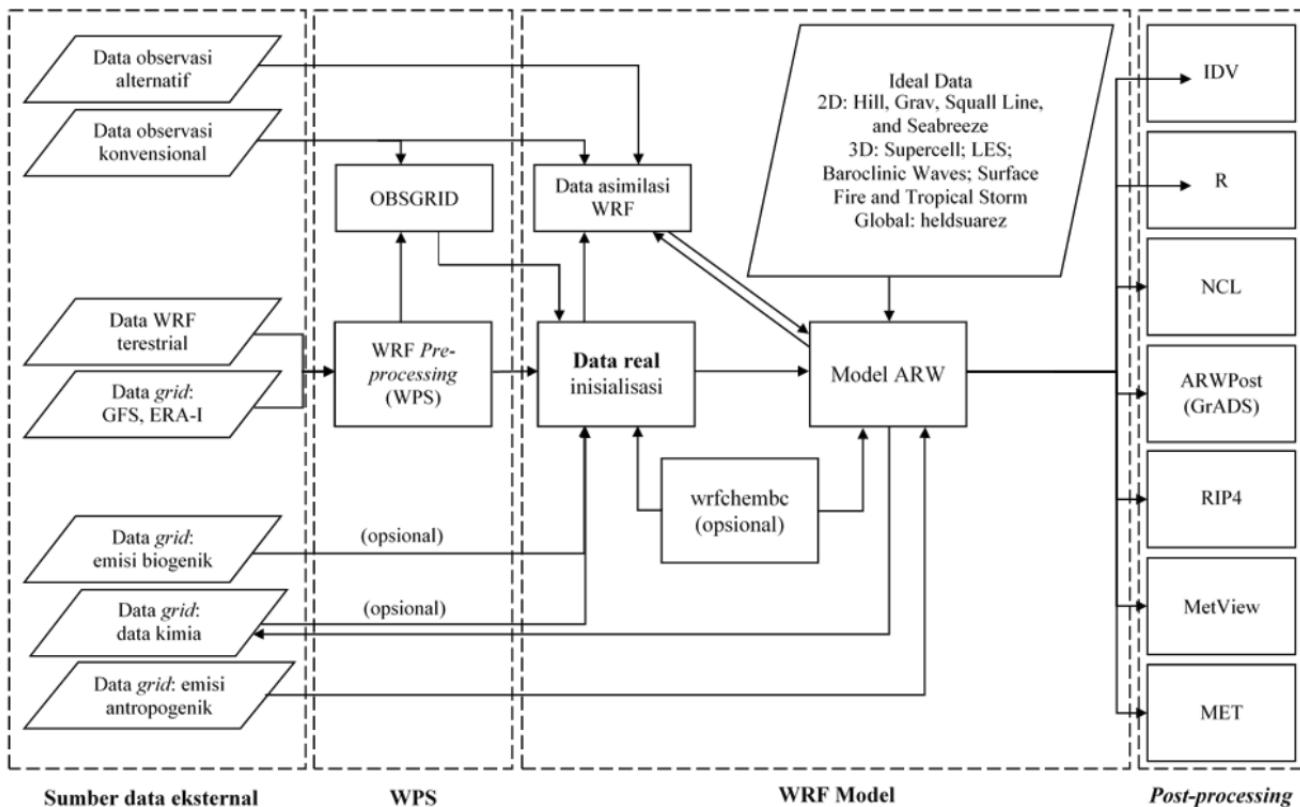
1.4 Menjalankan Simulasi WRF-ARW

Anda dapat melanjutkan ke tahapan ini apabila seluruh program telah berhasil terpasang. Bagi Anda yang belum berhasil, sabar :D dan ulangi kembali langkah-langkah di atas. Secara umum, diagram pada Gambar 1.11 menunjukkan proses menjalankan WRF dari tahapan memasukkan data hingga plot dan analisis akhir. Seperti yang telah dijelaskan pada subbab sebelumnya bahwa langkah awal dalam menjalankan WRF adalah menjalankan WPS terlebih dahulu. Anda perlu menyiapkan data masukan atmosfer maupun permukaan (data statik). Program `geogrid.exe` dijalankan pertama, kemudian diikuti dengan `ungrib.exe` dan terakhir `metgrid.exe`. Selanjutnya, Anda dapat melangkah ke program WRF, yaitu `real.exe` dan `wrf.exe`. Program `real.exe` digunakan sebagai pendefinisian kondisi awal dan kondisi batas berdasarkan informasi dari `namelist.input` yang berada dalam folder `test/em_real/`. Kemudian, Anda dapat menggunakan perangkat lunak apapun (mis. NCAR Command Language (NCL),

1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)

GrADS, R, Python, Julia, Matlab, ArcMAP, atau QGIS) untuk menganalisis serta visualisasi luaran WRF.

Untuk data masukan, Anda juga dapat menggunakan data observasi, tetapi harus berupa grid. Namun, ini merupakan program yang berbeda dari WRF-ARW, yaitu WRFDA (WRF Data Assimilation). Selain itu, terdapat pula data masukan dari emisi kimia, seperti emisi biogenik dan atropogenik. Akan tetapi, program ini merupakan turunan WRF-ARW, yaitu WRF-Chem. Penjelasan mengenai WRFDA dan WRF-Chem tidak disampaikan di dalam praktikum ini.



Gambar 1.11: Diagram WRF

Jika Anda menutup terminal bash atau mematikan laptop/komputer, definisikan kembali *ENVIRONMENT VARIABLE* seperti pada saat proses memasang WRF dan WPS. Variabel yang perlu didefinisikan hanya 3 saja seperti pada perintah di bawah ini. Anda dapat memasukkan variabel ini pada file *.bashrc* atau *.bash_profile* di direktori *\$HOME*.

```
export ODIR=$HOME/WRF-Model
export PATH=$ODIR/bin:$PATH
export LD_LIBRARY_PATH=$ODIR/lib:$LD_LIBRARY_PATH
```

Bagi pengguna Compiler **Intel**, Anda perlu menambahkan perintah di bawah ini. Perintah ini diperlukan pada saat menjalankan WRF.

```
ulimit -s unlimited
```

1.4.1 Program WPS

Untuk lebih mudah dalam memahami alur proses simulasi WRF pada modul ini, lokasi/domain yang dipilih untuk adalah Kota Surabaya dengan periode 1-3 Januari 2022 dengan data GFS dan ERA5. Interval waktu dari kedua data tersebut adalah 6 jam, yaitu pukul 00:00, 06:00, 12:00, dan 18:00 waktu Zulu (UTC+0). Langkah awal sebelum menjalankan WRF adalah menentukan lokasi dan waktu terlebih dahulu di dalam program WPS. Seperti yang telah dijelaskan sebelumnya, WPS memiliki 3 program utama: `geogrid.exe`, `ungrib.exe`, dan `metgrid.exe`.

Sesuai dengan kelanjutan dari subbab sebelumnya mengenai instalasi WRF, Anda saat ini berada di dalam folder `WPS/`. Jika lupa, Anda bisa mengetikkan kembali perintah ini.

```
cd $ODIR/WPS
```

Langkah-langkah menjalankan program WPS adalah sebagai berikut.

1.4.1.1 Penentuan Lokasi dan Waktu Simulasi

Isi dari file `namelist.wps` pada saat awal instalasi seperti ini.

```
&share
wrf_core = 'ARW',
max_dom = 2,
start_date = '2019-09-04_12:00:00','2019-09-04_12:00:00',
end_date   = '2019-09-06_00:00:00','2019-09-04_12:00:00',
interval_seconds = 10800
/

&geogrid
parent_id      = 1,    1,
parent_grid_ratio = 1,    3,
i_parent_start  = 1,    53,
j_parent_start  = 1,    25,
e_we            = 150,  220,
e_sn            = 130,  214,
geog_data_res = 'default','default',
dx = 15000,
dy = 15000,
map_proj = 'lambert',
ref_lat  = 33.00,
ref_lon   = -79.00,
truelat1 = 30.0,
truelat2 = 60.0,
stand_lon = -79.0,
geog_data_path = '/glade/work/wrfhelp/WPS_GEOG/'
/
```

```
&ungrib
  out_format = 'WPS',
  prefix = 'FILE',
/

&metgrid
  fg_name = 'FILE'
/
```

Terdapat berbagai macam parameter di dalam `&share`, `&geogrid`, `&ungrib`, dan `&metgrid`. Anda perlu mengubah beberapa parameter tersebut yang dapat disesuaikan dengan simulasi. Tapi, Anda harus memperhatikan aturan atau template yang telah diberikan di dalam panduan pengguna. Untuk lebih rincinya, Anda bisa lihat di [Panduan Pengguna WRF-ARW Bab 3](#). Di dalam tahapan ini, Anda perlu mengubah parameter-parameter yang ada di dalam `&geogrid`. Pada contoh yang telah kami berikan, Anda perlu mengganti parameter pada bagian `&share` dan `&geogrid` di dalam file `namelist.wps`. Anda dapat menggunakan teks editor `nano`.

```
nano namelist.wps
```

File yang telah berubah menjadi seperti di bawah ini.

```
&share
  wrf_core = 'ARW',
  max_dom = 3,
  start_year = 2022, 2022, 2022,
  start_month = 01, 01, 01,
  start_day = 01, 01, 01,
  start_hour = 00, 00, 00,
  end_year = 2022, 2022, 2022,
  end_month = 01, 01, 01,
  end_day = 03, 03, 03,
  end_hour = 18, 18, 18,
  interval_seconds = 21600,
  io_form_geogrid = 2,
/

&geogrid
  parent_id = 1, 1, 1,
  parent_grid_ratio = 1, 3, 9,
  i_parent_start = 1, 11, 15,
  j_parent_start = 1, 11, 15,
  e_we = 33, 40, 46,
  e_sn = 33, 40, 46,
  geog_data_res = 'default','default','default',
  dx = 18000,
  dy = 18000,
  map_proj = 'mercator',
```

```

ref_lat    = -7.328,
ref_lon    = 112.741,
truelat1   = -7.328,
geog_data_path = '/home/absen/WRF-Model/data/WPS_GEOG/'
/

&ungrib
out_format = 'WPS',
prefix = 'FILE',
/

&metgrid
fg_name = 'FILE'
io_form_metgrid = 2,
/

```

Penjelasan setiap variabel dapat dilihat pada Tabel 1.4.

Tabel 1.4: Informasi beberapa variabel di dalam `namelist.wps`

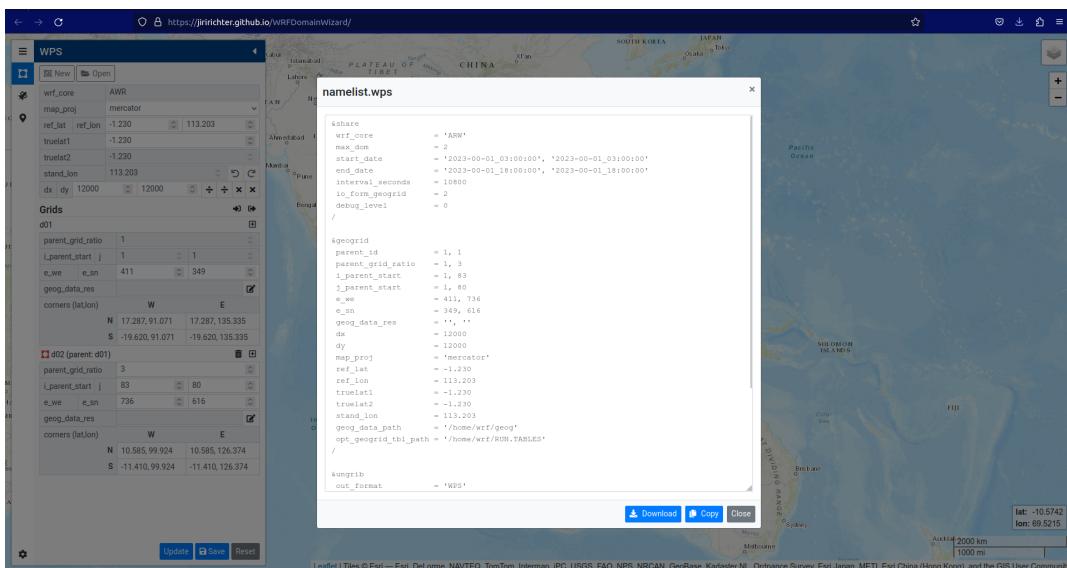
Variabel	Keterangan
wrf_core	Tipe penyelesaian WRF, ARW atau NMM
max_dom	Jumlah domain, semakin banyak domain maka semakin tinggi resolusi spasialnya
start_date	Waktu mulai simulasi sesuai dengan data masukan. Format: YYYY-MM-DD_HH:mm:ss
end_date	Waktu akhir simulasi sesuai dengan data masukan. Format: YYYY-MM-DD_HH:mm:ss
interval_seconds	Interval waktu dari data masukan (dalam detik)
io_from_geog	Tipe format file luaran <code>geogrid.exe</code> (1 = binary, 2 = netcdf, 3 = GRIB1)
parent_id	Untuk domain paling kasar, nilainya 1. Domain selanjutnya juga bernilai 1 yang menandakan bahwa subdomain merupakan bagian dari domain utamanya
parent_grid_ratio	Rasio piksel dari dx dan dy pada domain ke-1. Domain paling kasar adalah 1 dan selanjutnya mengikuti rasio yang diinginkan.
i_parent_start	Nomor indeks acuan untuk koordinat X. Domain paling kasar bernilai 1, ditentukan dari kiri-bawah
j_parent_start	Nomor indeks acuan untuk koordinat Y. Domain paling kasar bernilai 1, ditentukan dari kiri-bawah
e_we	Jumlah grid/piksel dari barat ke timur mengikuti rumus <code>parent_grid_ratio * N + 1</code> , dengan $N > 0$
e_sn	Jumlah grid/piksel dari selatan ke utara mengikuti rumus <code>parent_grid_ratio * N + 1</code> , dengan $N > 0$
geog_data_resolution	Pemilihan resolusi spasial dari data statik
dx	Resolusi spasial pada koordinat X (dalam meter)
dy	Resolusi spasial pada koordinat Y (dalam meter)
map_proj	Sistem proyeksi peta (<code>mercator</code> , <code>lambert</code> , <code>polar</code> , <code>lat-lon</code>). Untuk simulasi di sekitar khatulistiwa, direkomendasikan menggunakan <code>mercator</code>
ref_lat	Koordinat lintang titik tengah acuan dari domain paling kasar
ref_lon	Koordinat bujur titik tengah acuan dari domain paling kasar

Variabel	Keterangan
truclat1	Koordinat lintang sebenarnya. Dibutuhkan untuk sistem proyeksi mercator, polar, dan lambert
geog_data_path	lokasi folder WPS_GEOG

Pada parameter `start_date` dan `end_date`, Anda bisa menggunakan parameter lain dengan membagi masing-masing format tahun (`start_year; end_year`), bulan (`start_month; end_month`), tanggal (`start_day; end_day`), dan jam (`start_hour; end_hour`).

Berkaitan dengan skala resolusi spasial pada simulasi yang akan dijalankan dengan pemilihan lokasi di Kota Surabaya, skala tertingginya adalah 2 km. Anda perhatikan parameter `parent_grid_ratio`. Nilai 1, 3, dan 9 secara berturut-turut merupakan rasio terhadap dx atau dy untuk setiap domain. Nilai 1 berarti untuk domain terluar dengan skala $1/1 * 18000 = 18000$ meter, sedangkan nilai 3 untuk domain ke-2 dengan skala $1/3 * 18000 = 6000$ meter.

Pada bagian `&geogrid`, penentuan lokasi kajian ini cukup rumit. Anda bisa menggunakan halaman Github <https://jirichter.github.io/WRFDomainWizard> untuk membuat file `namelist.wps` sesuai dengan lokasi yang Anda inginkan, setidaknya Anda dapat menyalin teks pada bagian `&geogrid` Gambar 1.12.

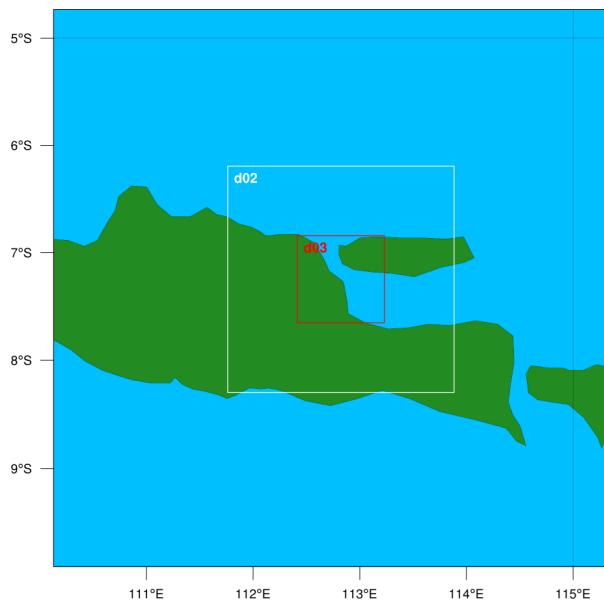


Gambar 1.12: WRF Domain Wizard oleh [Jirichter](#)

Untuk memverifikasi kesesuaian pemilihan lokasi yang telah dibuat, Anda dapat menggunakan skrip `plotgrids_new.ncl` di dalam folder `util/`. Buka file tersebut. Cari dan ubahlah variabel `type = x11` menjadi `type = png` untuk mengatur luaran file dalam bentuk gambar (format `.png`). Untuk menjalankan skrip ini, aktifkan terlebih dahulu `environment ncl` dengan perintah `source activate ncl` atau `conda activate ncl`. Kemudian, jalankan perintah berikut ini. File `.png` akan muncul di dalam folder `WPS/` dengan nama `wps_show_dom.png`. Hasilnya seperti Gambar 1.13

```
cd $HOME/WRF-Model/WPS
ncl util/plotgrids_new.ncl
```

WPS Domain Configuration



Gambar 1.13: Konfigurasi domain WRF

1.4.1.2 Menjalankan Program geogrid.exe

Selanjutnya, Anda dapat menjalankan program `geogrid.exe` dengan perintah di bawah ini. Hasilnya, terdapat 3 file dengan nama `geo_em.d0x.nc` (dengan x = nomor domain; 1, 2, 3) karena pengaturan `max_dom = 3`.

```
./geogrid.exe
```

```
ls geo_em*
```

`geo_em.d01.nc geo_em.d02.nc geo_em.d03.nc`

1.4.1.3 Menyambungkan File Data Masukan dan Tabel Variabel

WPS menyediakan program `link_grib.csh` untuk menyambungkan file data masukan ke dalam folder WPS/ dengan membuat shortcut yang bernama `GRIBFILE.*` (`GRIBFILE.AAA`, `GRIBFILE.AAB`, ...). Perintahnya sebagai berikut.

```
./link_grib.csh $ODIR/data/GFS/*
```

Kemudian, buatlah shortcut dengan nama Vtable di dalam folder WPS/ dari file yang berada di dalam folder ungrib/Variable_Tables. Untuk data GFS, nama file tersebut adalah Vtable.GFS, sedangkan ERA5 adalah Vtable.ERA-Interim.pl. Perintahnya sebagai berikut.

```
ln -sf ungrib/Variable_Tables/Vtable.GFS Vtable # GFS  
ln -sf ungrib/Variable_Tables/Vtable.ERA-Interim.pl Vtable # ERA5
```

1.4.1.4 Menjalankan Program ungrib.exe

Setelah menyambungkan data masukan serta tabel variabel, jalankan program **ungrib.exe** dengan perintah berikut.

```
./ungrib.exe
```

Luaran dari program ini adalah file dengan nama FILE:* yang memiliki format .nc (netcdf).

1.4.1.5 Menjalankan Program metgrid.exe

Jalankan program **metgrid.exe** dengan perintah berikut.

```
./metgrid.exe
```

Luaran dari program ini adalah file dengan nama **met_em*** yang memiliki format .nc. File-file ini nanti yang akan dipindahkan menuju folder WRF/test/em_real atau Anda dapat juga membuat shortcut. Perintahnya sebagai berikut.

```
mv met_em* $ODIR/WRF/test/em_real # memindahkan semua file met_em* ke folder  
    ↳ WRF/test/em_real  
# atau  
ln -sf met_em* $ODIR/WRF/test/em_real # membuat shortcut di folder  
    ↳ WRF/test/em_real
```

1.4.2 Program WRF

Pada bagian ini, program WRF yang digunakan hanya 2: **real.exe** dan **wrf.exe**. Sebelum itu, Anda diharuskan mengubah beberapa parameter pada file **namelist.input** seperti di dalam file **namelist.wps**. Parameter di dalam **namelist.input** sangat banyak karena terdapat bagian pemilihan lokasi dan waktu (disesuaikan dengan **namelist.wps**), pemilihan skema parameter fisik, dan parameter dinamik. Kami hanya memberikan beberapa parameter yang diperlukan untuk contoh simulasi. Anda dapat membaca lebih lanjut di [Panduan Pengguna WRF Bab 5](#).

1.4.2.1 Mengubah isi namelist.input

Saat ini Anda masih berada di folder WPS/. Buka terlebih dahulu folder `em_real` dengan perintah

```
cd $ODIR/WRF/test/em_real
```

Di dalam folder `em_real`, terdapat 4 program, yaitu `real.exe`, `ndown.exe`, `tc.exe`, dan `wrf.exe`. Nilai `max_dom` pada `namelist.wps` adalah 3. Jika pemilihan lokasi Anda memiliki >1 domain, WRF akan memproses simulasi sampai pada domain tertinggi atau disebut dengan metode *nesting*. Proses ini juga dibagi lagi menjadi 2, yaitu *nesting* dua arah dan satu arah. Untuk *nesting* satu arah, program yang digunakan bisa atau tanpa dengan program `ndown.exe`. Penggunaan *nesting* satu arah biasanya pengguna hanya tertarik pada analisis domain tertinggi. Penggunaan `ndown.exe` juga sering digunakan pada turunan WRF, yaitu WRF-Chem. Untuk *nesting* dua arah, proses-proses perhitungan fisis pada domain tertinggi dikembalikan (*feedback*) ke domain terendah sehingga proses analisis dapat dilakukan pada semua domain. Anda dapat membaca lebih lanjut mengenai *nesting* di https://ruc.noaa.gov/wrf/wrf-chem/wrf_tutorial_2012_brazil/WRF_nesting.pdf.

Bukalah file `namelist.input` dengan cara sama seperti Anda membuka `namelist.wps`. Ketika pertama kali dibuka, file `namelist.input` seperti di bawah ini. Anda juga dapat melihat beberapa contoh/template lain (mis. `namelist.input.4km`, `namelist.input.chem`, `namelist.input.volc`, ...) sesuai dengan kebutuhan tertentu.

```
&time_control
run_days = 0,
run_hours = 36,
run_minutes = 0,
run_seconds = 0,
start_year = 2019, 2019,
start_month = 09, 09,
start_day = 04, 04,
start_hour = 12, 12,
end_year = 2019, 2019,
end_month = 09, 09,
end_day = 06, 06,
end_hour = 00, 00,
interval_seconds = 10800
input_from_file = .true.,.true.,
history_interval = 60, 60,
frames_per_outfile = 1, 1,
restart = .false.,
restart_interval = 7200,
io_form_history = 2
io_form_restart = 2
io_form_input = 2
io_form_boundary = 2
/
&domains
```

```

time_step = 90,
time_step_fract_num = 0,
time_step_fract_den = 1,
max_dom = 2,
e_we = 150, 220,
e_sn = 130, 214,
e_vert = 45, 45,
dzstretch_s = 1.1
p_top_requested = 5000,
num_metgrid_levels = 34,
num_metgrid_soil_levels = 4,
dx = 15000,
dy = 15000,
grid_id = 1, 2,
parent_id = 0, 1,
i_parent_start = 1, 53,
j_parent_start = 1, 25,
parent_grid_ratio = 1, 3,
parent_time_step_ratio = 1, 3,
feedback = 1,
smooth_option = 0
/

&physics
physics_suite = 'CONUS'
mp_physics = -1, -1,
cu_physics = -1, -1,
ra_lw_physics = -1, -1,
ra_sw_physics = -1, -1,
bl_pbl_physics = -1, -1,
sf_sfclay_physics = -1, -1,
sf_surface_physics = -1, -1,
radt = 15, 15,
bldt = 0, 0,
cudt = 0, 0,
icloud = 1,
num_land_cat = 21,
sf_urban_physics = 0, 0,
fractional_seaice = 1,
/

```

```

&fdda
/

```

```

&dynamics
hybrid_opt = 2,
w_damping = 0,
diff_opt = 2, 2,

```

```

km_opt = 4,      4,
diff_6th_opt = 0,      0,
diff_6th_factor = 0.12, 0.12,
base_temp = 290.
damp_opt = 3,
zdamp = 5000., 5000.,
dampcoef = 0.2, 0.2,
khdif = 0,      0,
kvdif = 0,      0,
non_hydrostatic = .true., .true.,
moist_adv_opt = 1,      1,
scalar_adv_opt = 1,      1,
gwd_opt = 1,      0,
/
&bdy_control
spec_bdy_width = 5,
specified = .true.
/
&grib2
/
&namelist_quilt
nio_tasks_per_group = 0,
nio_groups = 1,
/

```

Pada file ini, samakan beberapa parameter seperti di file `namelist.wps`. Perhatikan Tabel 1.5. Anda cukup mencari parameter yang sama antara `namelist.wps` dengan `namelist.input`, tetapi tidak semuanya ada di dalam `namelist.input`. Untuk parameter lainnya, seperti `parent_time_step_ratio`, `time_step`, `history_interval`, `frame_per_outfile`, dan seterusnya, Anda dapat membacanya lebih banyak di [Panduan Pengguna WRF Bab 5](#) atau bisa dilihat pada file `README.namelist` di dalam folder `test/em_real` untuk setiap penjelasan singkat berbagai parameter.

Bagian `&time_control` berfungsi sebagai pengaturan waktu simulasi serta luaran yang akan dihasilkan. Sebagai informasi, file luaran WRF berformat NetCDF (.nc) dengan nama `wrfout_<domain>_<yyyy>-<mm>-<dd>_<HH>` dimana

- domain: identitas domain (d01, d02, ...)
- yyyy: tahun, dengan format 4 digit
- mm: bulan, dengan format 2 digit
- dd: tanggal, dengan format 2 digit
- HH: jam, dengan format 2 digit
- MM: menit, dengan format 2 digit
- SS: detik, dengan format 2 digit

Pada parameter `run_days`, `run_hours`, `run_minutes`, dan `run_seconds`, ini dapat dihitung dari selisih waktu akhir simulasi dengan awal simulasi. Pada simulasi yang akan dicoba dalam modul ini, yaitu 1

Januari 2022 pukul 00:00 UTC hingga 3 Januari 2022 18:00 UTC, nilai dari `run_days` dan `run_hours` secara berturut-turut adalah 2 dan 18. Anda juga dapat mengatur `run_days` ini menjadi 0 setelah dikonversi menjadi jam (2 hari = 48 jam) dan tambahkan ke `run_hours`, yaitu menjadi 66. Parameter `history_interval` digunakan untuk meletakkan nilai pada file luaran WRF dalam format .nc dengan waktu tertentu (*dalam menit*). Misalkan diatur ke 60, berarti hasil perhitungan dari berbagai algoritma WRF dimasukkan ke file setiap 60 menit sekali. Anda bebas mengatur angka pada parameter ini. Dampaknya, ukuran file akan semakin besar jika Anda mengatur nilainya menjadi kecil. Tentu ini tidak akan menjadi masalah apabila ruang kosong penyimpanan internal/eksternal Anda masih tersedia. Banyaknya file luaran dapat pula diatur jumlahnya, yaitu di dalam parameter `frames_per_outfile`. Jika parameter diatur pada `frames_per_outfile = 1, history_interval = 60, dan run_hours = 66`, berarti file luaran yang akan dibuat dan disimpan ke dalam penyimpanan adalah sebanyak 66 file. Agar lebih efektif saat akan melakukan analisis, kami menyarankan untuk mengatur `frames_per_outfile = 1000`.

Kemudian untuk bagian `&domains`, digunakan untuk mengatur kondisi dari domain agar dapat sesuai dengan data masukan (banyak grid horizontal dan vertikal, posisi, rasio grid, tipe interpolasi). Pada parameter `feedback`, Anda dapat mengatur tipe *nesting* dua arah (1) atau satu arah (0). Parameter `num_metgrid_levels` dan `num_metgrid_soil_levels` harus diatur sesuai dengan yang ada di dalam salah satu file `met_em*`. Untuk melihatnya, gunakan perintah di bawah ini. Nilai kedua parameter `num_metgrid_levels` dapat berbeda sesuai dengan data masukan yang digunakan. Sebagai contoh, GFS memiliki `num_metgrid_levels = 34`, sedangkan ERA5 `num_metgrid_levels = 38`.

```
ncdump -h met_em.d01.2022-01-01_00:00:00.nc | grep num_metgrid_levels      # = 34
ncdump -h met_em.d01.2022-01-01_00:00:00.nc | grep NUM_METGRID_SOIL_LEVELS # = 4
```

Di dalam bagian `&physics`, terdapat berbagai skema parameterisasi dalam penyelesaian perhitungan pembentukan awan, skemar radiasi, lapisan perbatas, serta proses-proses di permukaan tanah. Pada parameter `physics_suite`, Anda bisa mengurnya ke TROPICAL karena wilayah yang ingin disimulasikan berada di daerah tropis. Hanya saja, opsi default untuk skema parameterisasi belum tentu menghasilkan luaran yang bagus sesuai dengan kondisi sebenarnya di daerah tropis. Anda perlu mengkaji dari beberapa publikasi nasional maupun internasional dalam memilih skema parameterisasi, khususnya di wilayah Indonesia. Ketika Anda mendefinisikan `physics_suite = 'TROPICAL'`, Anda tidak perlu lagi menambahkan angka pada parameter di bawah ini.

- `mp_physics` = 6: WSM6
- `cu_physics` = 16: New-Tiedke
- `ra_lw_physics` = 4: Rapid Radiative Model Transformation for GCM (RRTMG)
- `ra_sw_physics` = 4: Rapid Radiative Model Transformation for GCM (RRTMG)
- `bl_pbl_physics` = 1: Yonsei University
- `sf_sfclay_physics` = 91: MM5 Monin-Obukhov
- `sf_surface_physics` = 2: Noah Land Surface Model

Anda hanya perlu menambahkan nilai -1 pada parameter-parameter tersebut, tentu saja sesuai dengan banyaknya domain. Anda juga dapat menghilangkan parameter `physics_suite` dan mengganti parameter `mp_physics`, `cu_physics`, `sf_surface_physics`, `sf_sfclay_physics`, `ra_lw_physics`, `ra_sw_physics`, dan `bl_pbl_physics` ke opsi lain yang ada di dalam [Panduan Pengguna WRF Bab 5](#). Anda harus berhati-hati dan memperhatikan pemilihan skema parameterisasi karena terdapat parameter yang harus ditambahkan di dalam bagian `&physics`, menambahkan bagian lain, atau kombinasi antar

skema. Sebagai contoh untuk `sf_surface_physics = 4` (*Noah-MP Land Surface Model*), perlu menambahkan bagian `&noah_mp`; Ketika `cu_physics = 14` (*Scale-aware SAS*), perlu menambahkan parameter `shcu_physics = 4`; Parameter `b1_pbl_physics = 2` (*Mellor-Yamada-Janjic, MYJ*) nilai dari parameter `sf_sfclay_physics = 2` (*Eta Model*). Untuk pengaturan nilai-nilai di dalam `&physics`, Anda harus menambahkannya sebanyak jumlah domain. Anda juga dapat menonaktifkan skema parameterisasi tertentu dengan mengubahnya ke 0, misalnya pada `cu_physics` untuk domain dengan resolusi <10 km, yaitu pada domain 2 dan 3.

Tabel 1.5: Informasi sebagian variabel di dalam `namelist.input` yang perlu disesuaikan dengan `namelist.wps`

Parameter di <code>namelist.wps</code>	Parameter di <code>namelist.input</code>
<pre>start_year = 2022, 2022, 2022, start_month = 01, 01, 01, start_day = 01, 01, 01, start_hour = 00, 00, 00, end_year = 2022, 2022, 2022, end_month = 01, 01, 01, end_day = 03, 03, 03, end_hour = 18, 18, 18, - - max_dom = 3, interval_seconds = 21600, parent_id = 1, 1, 1, parent_grid_ratio = 1, 3, 9, dx = 15000, dy = 15000, i_parent_start = 1, 11, 15, j_parent_start = 1, 11, 15, e_we = 33, 40, 46, e_sn = 33, 40, 46, - -</pre>	<pre>start_year = 2022, 2022, 2022, start_month = 01, 01, 01, start_day = 01, 01, 01, start_hour = 00, 00, 00, end_year = 2022, 2022, 2022, end_month = 01, 01, 01, end_day = 03, 03, 03, end_hour = 18, 18, 18, run_days = 2, run_hour = 18, max_dom = 3, interval_seconds = 21600, parent_id = 1, 1, 1, parent_grid_ratio = 1, 3, 9, dx = 15000, dy = 15000, i_parent_start = 1, 11, 15, j_parent_start = 1, 11, 15, e_we = 33, 40, 46, e_sn = 33, 40, 46, parent_time_step_ratio = 1, 3, 9, (sama dengan parent_grid_ratio) time_step = 90, dengan rumus 6 * dx (dalam km)</pre>

Isi file `namelist.input` yang telah diubah sepenuhnya menjadi seperti ini.

```
&time_control
run_days          = 2,
run_hours         = 18,
run_minutes       = 0,
```

```

run_seconds = 0,
start_year = 2022, 2022, 2022,
start_month = 01, 01, 01,
start_day = 01, 01, 01,
start_hour = 00, 00, 00,
end_year = 2022, 2022, 2022,
end_month = 01, 01, 01,
end_day = 03, 03, 03,
end_hour = 18, 18, 18,
interval_seconds = 21600,
input_from_file = .true.,.true.,.true.,
history_interval = 60, 60, 60,
frames_per_outfile = 1000, 1000, 1000,
restart = .false.,
restart_interval = 7200,
io_form_history = 2
io_form_restart = 2
io_form_input = 2
io_form_boundary = 2
/

&domains
time_step = 90,
time_step_fract_num = 0,
time_step_fract_den = 1,
max_dom = 3,
e_we = 33, 40, 46,
e_sn = 33, 40, 46,
e_vert = 44, 44, 44,
dzstretch_s = 1.1
p_top_requested = 5000,
use_surface = .false.,
sfcp_to_sfcp = .true.,
num_metgrid_levels = 34, ! 34 = GFS, 38 = ERA5
num_metgrid_soil_levels = 4,
dx = 18000,
dy = 18000,
grid_id = 1, 2, 3,
parent_id = 1, 1, 1,
i_parent_start = 1, 11, 15,
j_parent_start = 1, 11, 15,
parent_grid_ratio = 1, 3, 9,
parent_time_step_ratio = 1, 3, 9,
feedback = 1,
smooth_option = 0
/

&physics

```

```

physics_suite          = 'TROPICAL'
mp_physics            = -1,      -1,      -1,
cu_physics            = -1,      0,       0,
ra_lw_physics         = -1,      -1,      -1,
ra_sw_physics         = -1,      -1,      -1,
bl_pbl_physics        = -1,      -1,      -1,
sf_sfclay_physics    = -1,      -1,      -1,
sf_surface_physics   = -1,      -1,      -1,
radt                  = 18,     18,     18,
bldt                  = 0,       0,       0,
cudt                  = 0,       0,       0,
icloud                = 1,
num_land_cat          = 21,
sf_urban_physics      = 0,       0,       0,
fractional_seaice    = 1,
/

&fdda
/


&dynamics
hybrid_opt            = 2,
w_damping              = 0,
diff_opt               = 2,      2,      2,
km_opt                 = 4,      4,      4,
diff_6th_opt           = 0,      0,      0,
diff_6th_factor        = 0.12,   0.12,   0.12,
base_temp              = 290.
damp_opt               = 3,
zdamp                  = 5000.,  5000.,  5000.,
dampcoef               = 0.2,    0.2,    0.2,
khdif                  = 0,       0,       0,
kvdif                  = 0,       0,       0,
non_hydrostatic         = .true., .true., .true.,
moist_adv_opt          = 1,       1,       1,
scalar_adv_opt          = 1,       1,       1,
gwd_opt                = 1,       0,       0,
/


&bdy_control
spec_bdy_width         = 5,
specified              = .true.
/


&grib2
/


&namelist_quilt

```

```
nio_tasks_per_group = 0,  
nio_groups = 1,  
/
```

1.4.2.2 Menjalankan program `real.exe` dan `wrf.exe`

Program `real.exe` mengeluarkan file-file dalam format `.nc`, yaitu `wrfbdy_d01` dan `wrfinput_<domain>`. Perintah menjalankan program ini adalah sebagai berikut.

```
./real.exe
```

Untuk melihat respon dari program ini, Anda dapat melihat file `rsl.error.0000` dengan perintah

```
tail rsl.error.0000 -n 1
```

Jika respon yang diberikan terdapat kalimat `real_em: SUCCESS COMPLETE REAL_EM INIT`, Anda bisa melanjutkan ke tahapan `wrf.exe`.

```
./wrf.exe
```

Simulasi WRF dari program `wrf.exe` telah berhasil selesai apabila terdapat kalimat `wrf: SUCCESS COMPLETE WRF` pada baris terakhir `rsl.error.0000` dan terdapat file `wrfout_d0*`. Untuk simulasi ini, terdapat 3 file `wrfout_d0*`: `wrfout_d01_2022-01-01_00:00:00`, `wrfout_d02_2022-01-01_00:00:00`, dan `wrfout_d03_2022-01-01_00:00:00`. Sejatinya, file-file tersebut berformat `*.nc` walaupun tidak tertera pada nama file. File `wrfout_d0*` memiliki interval waktu per 1 jam, dari pukul 00:00 UTC 1 Januari - 18:00 UTC 3 Januari 2022. Langkah selanjutnya, Anda dapat menganalisis luaran WRF dengan aplikasi apapun yang Anda bisa, selama mendukung format `*.nc`. Hanya saja, Anda perlu mengubah bentuk dan nama variabel WRF untuk disesuaikan dengan *CF-Convention* dengan program NCL yang telah disediakan di https://sundowner.colorado.edu/wrfout_to_cf/wrfout_to_cf.ncl.

Pengguna Docker di MacOS M1/M2 dapat menyalin file `wrfout_*` ke direktori lokal dengan perintah sebagai berikut.

```
docker cp <nama-container>:$HOME/WRF-Model/WRF/test/em_real/wrfout_* .
```

1.5 Visualisasi Luaran WRF

Terdapat 3 file luaran WRF yang telah Anda jalankan dengan masing-masing memiliki resolusi spasial yang berbeda, yaitu 18 km, 6 km, dan 2 km dengan pemilihan lokasi titik tengah longitude dan latitude di Kota Surabaya. Selanjutnya, Anda dapat melihat hasil dari simulasi tersebut dengan berbagai macam perangkat lunak. Pada modul ini, kami menampilkan hasil simulasi dengan menggunakan aplikasi NCL, QGIS, Python, R, dan Julia.

1.5.1 Python

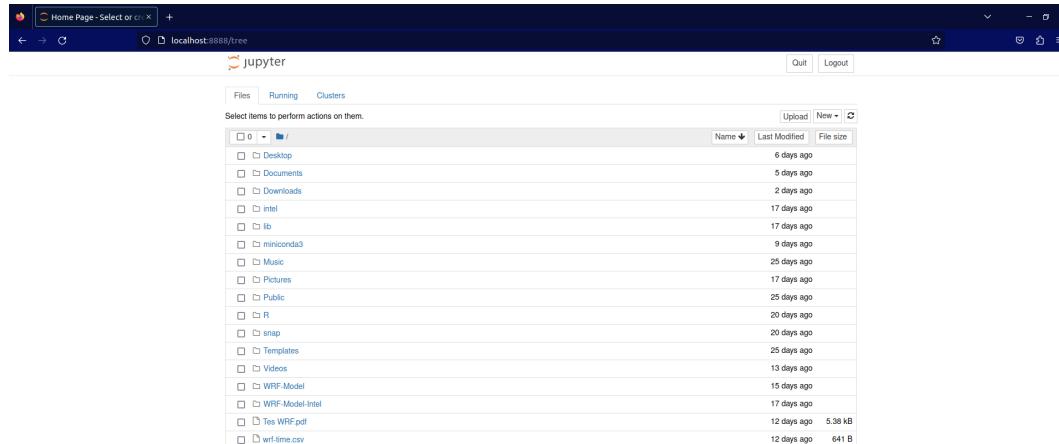
Untuk mengetikkan kode Python. Anda dapat menggunakan kode editor **Jupyter Notebook** atau aplikasi lainnya, seperti Visual Studio Code, Notepad++, atau Atom. Untuk langkah-langkah di bawah ini, kami lebih menjelaskan cara penulisan kode Python pada **Jupyter Notebook**. Kode editor ini dapat digunakan pada aplikasi Browser default Anda, misalnya Google Chrome, Safari, atau Microsoft Edge. Kami menyarankan Anda untuk menggunakan **Jupyter Notebook** agar hasil kode langsung bisa ditampilkan sehingga respons dari setiap sel dapat diketahui/dicetak secara langsung. Ikuti langkah-langkah berikut ini untuk membuka **Jupyter Notebook**.

1. Buka terminal Bash Anda. Saat ini, Anda berada di direktori `$HOME` atau ~.
2. Buka direktori `WRF-Model` dengan mengetik perintah

```
cd $HOME/WRF-Model
```

3. Aktifkan terlebih dahulu *Environment ncl* dengan perintah `conda activate ncl`.
4. Ketikkan perintah berikut ini untuk memulai pengetikan kode Python. Browser default Anda akan terbuka dengan menampilkan kode editor **Jupyter Notebook** Gambar 1.14

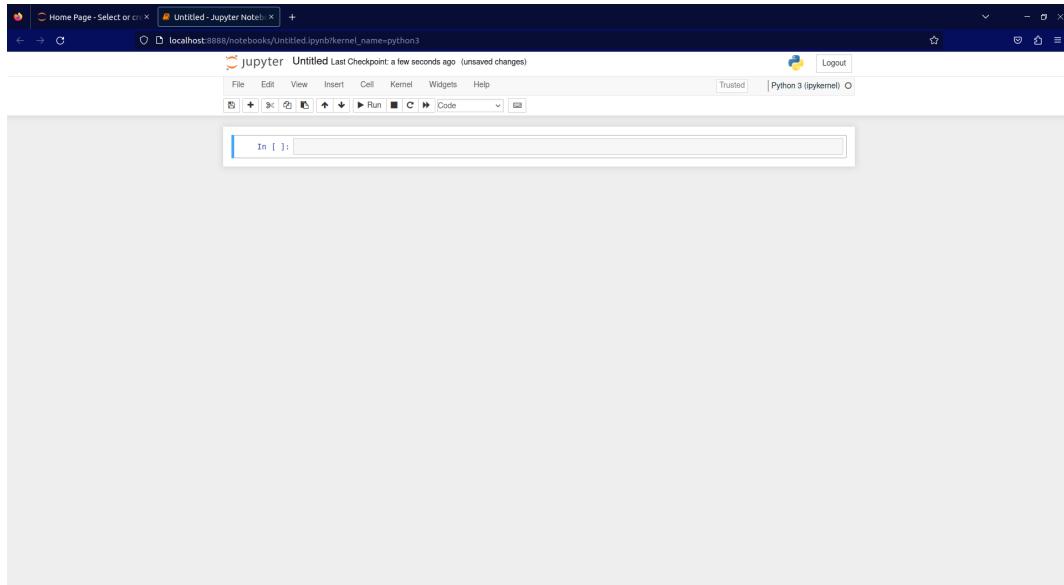
```
jupyter notebook
```



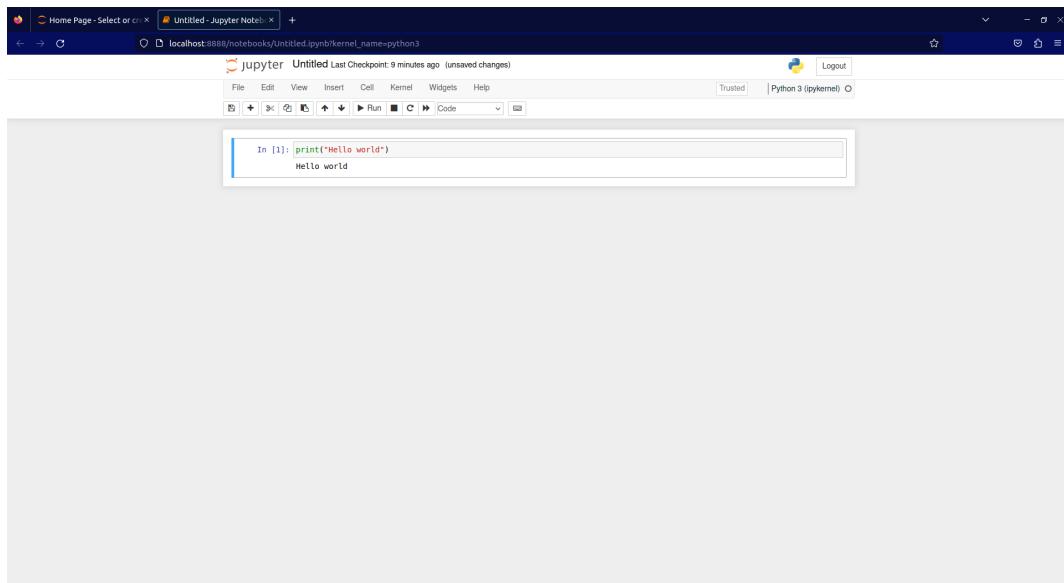
Gambar 1.14: Tampilan Jupyter Notebook

5. Klik **New** dan pilih **Python 3 (ipykernel)**. Tampilan awal **Jupyter Notebook** seperti pada Gambar 1.15
6. Anda dapat langsung mendapatkan hasil dari kode yang ditulis pada sel (lihat In [1]) (Gambar 1.16)

1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)



Gambar 1.15: Tampilan awal Jupyter Notebook



Gambar 1.16: Tampilan interaktif Jupyter Notebook

Untuk pengolahan data WRF di Python, kami menggunakan `wrf-python` [12]. Paket ini dikembangkan oleh NCAR yang dikhkususkan untuk *Post-Processing* luaran WRF, mendukung pembacaan file, perhitungan interpolasi, serta visualisasi WRF. Perhatikan langkah-langkah berikut cara penggunaannya mulai dari pembacaan file hingga pembuatan grafik.

1. Impor paket

```
from netCDF4 import Dataset
import wrf
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as crs
```

2. Buka salah satu file `wrfout_d0*`, misalnya `wrfout_d03_2022-01-01_00:00:00` dan cetak variabel tersebut, misalkan variabel T2 (*Air Temperature at 2m*: suhu udara 2 meter dari permukaan tanah). Isinya adalah beberapa metadata. Anda dapat mengetahui variabel-variabel di dalamnya dengan menambahkan metode `.variables`. Anda dapat melihat penjelasan variabel-variabel pada panduan pengguna WRF-ARW.

```
# Lokasi folder luaran WRF
wrf_path = '/home/absen/WRF-Model/WRF/test/em_real'
# Membuka file wrfout
wrf_d03 = Dataset(f"{wrf_path}/wrfout_d03_2022-01-01_00:00:00")
# Melihat variabel
wrf_d03.variables
```

3. Anda dapat mengambil variabel dengan fungsi `wrf.getvar()` dengan menyertakan argumen dari nama variabel WRF. Untuk mengambil T2, gunakan perintah di bawah ini. Perhatikan hasil yang diperoleh merupakan tipe `xarray.DataArray` dan metode `wrf.ALL_TIMES` pada argumen `timeidx` berguna untuk mengambil seluruh waktu, mulai dari awal hingga akhir simulasi. Anda dapat mengambil salah satu waktu dengan mencantumkan angka bulat (mis. 0, 1, 2, ...).

```
wrf_t2 = wrf.getvar(wrf_d03, "T2", timeidx=wrf.ALL_TIMES)
```

4. Untuk mendapatkan nilai koordinat latitude dan longitude dari variabel T2, Anda dapat memakai fungsi `wrf.latlon_coords()`. Pastikan dalam satu baris terdapat 2 variabel.

```
lats, lons = wrf.latlon_coords(wrf_t2)
```

5. Untuk mendapatkan waktu, Anda dapat memanggil *Coordinates Time* yang telah tersedia di dalam `wrf_t2`.

```
wrf_time = wrf_t2.Time
```

6. Anda juga dapat mengekstrak beberapa variabel yang tidak tersedia di dalam WRF, tentunya terbatas, seperti resultan kecepatan angin (`wspd`), arah angin (`wdir`), *Convective Available Potential*

Energy (CAPE; cape3d_only/mcape), atau kelembapan relatif (rh2). Anda dapat membaca lebih lanjut di <https://wrf-python.readthedocs.io/en/latest/diagnostics.html>.

7. Variabel curah hujan tidak ada di dalam WRF. Anda harus mengekstrak dan menjumlahkan variabel RAINC (*Accumulated Total Cumulus Precipitation*) dan RAINNC (*Accumulated Total Grid Scale Cumulus Precipitation*), maka diperoleh akumulasi curah hujan dari awal sampai akhir waktu simulasi. Anda perlu mengurangi curah hujan dari waktu ke t dengan $t - 1$ dengan memanfaatkan program perulangan (*looping*).

```
# Ekstrak RAINC dan RAINNC
rainc = wrf.getvar(wrf_d03, "RAINC", timeidx = wrf.ALL_TIMES)
rainnc = wrf.getvar(wrf_d03, "RAINNC", timeidx = wrf.ALL_TIMES)

# Menghitung Curah hujan akumulasi
rain = rainc + rainnc
rain_diff = rain.copy() # Metode .copy() agar var rain tidak ikut terubah

# Lakukan perulangan
length = len(wrf_time.values)
for i in range(1, length):
    rain_diff[i, :, :] = rain[i, :, :] - rain[i-1, :, :]

# Copy Attribute (dari RAINC atau RAINNC)
rain_diff.attrs = rainc.attrs
# Tambahkan deskripsi
rain_diff.attrs["description"] = "Total Rainfall"
# Menghapus variabel rain untuk mengoptimalkan memori
del rain
```

Setelah berhasil mengekstrak variabel dari langkah sebelumnya, langkah-langkah berikut ini adalah pembuatan grafik spasial dari variabel T2. Untuk membuatnya, Anda membutuhkan tambahan package `matplotlib` dan `cartopy`.

1. Anda telah mendefinisikan lokasi (variabel `lats` dan `lons`) pada langkah sebelumnya. Kedua variabel tersebut digunakan dalam membuat grafik spasial.
2. Variabel suhu udara yang telah diekstrak pada langkah sebelumnya memiliki waktu dari awal hingga akhir simulasi. Anda hanya dapat memilih salah satu waktu dalam 1 grafik. Anda juga dapat membuat grafik untuk masing-masing waktu dalam satu grafik dalam bentuk *panel plot*. Untuk membuat grafik pada waktu tertentu, Anda perlu memilih waktu dan mengetahui letak indeksnya.

```
time      = "2022-01-02 13:00:00"
time      = np.array([time], dtype='datetime64[ns]')
time_idx = np.where(wrf_time.values == time)[0]
time_idx = int(time_idx)
```

3. Sebagai contoh skrip di bawah ini untuk membuat grafik spasial hanya satu waktu.

```

# Mengambil informasi sistem proyeksi peta (dalam data ini adalah mercator)
cart_proj = wrf.get_cartopy(wrf_t2, timeidx=time_idx)

# Membuat dan mengatur ukuran grafik
fig = plt.figure(figsize=(12, 10))
# Mengatur sistem proyeksi sesuai metadata WRF
ax = plt.axes(projection=cart_proj)

# Menambahkan garis pantai. Sumber data: www.naturalearthdata.com
ax.coastlines(linewidth=0.8)

# Menambahkan garis lintang dan bujur
gl = ax.gridlines(draw_labels=True, linewidth=1, color='gray', alpha=0.5,
    linestyle='--')
gl.top_labels = False # Menghilangkan label bujur di atas
gl.right_labels = False # Menghilangkan label lintang di kanan

# Mengatur nilai untuk skala legenda
lvl = np.arange(290, 304, 2)

# Menambahkan garis kontur terisi
plt.contourf(lons, lats, wrf_t2[time_idx, :, :], levels=lvl,
    transform=crs.PlateCarree(),
    cmap='viridis')

# Menambahkan legenda warna
plt.colorbar(ax=ax, shrink=0.7)

# Menampilkan grafik
plt.show()

```

Selain spasial, Anda dapat pula menampilkan grafik seri waktu untuk titik koordinat atau rata-rata grid tertentu.

1. Tentukan titik koordinat yang akan dibuat grafik seri waktu

```

lats_sel = -7.271372797667375
lons_sel = 112.73417496409039

```

2. Untuk menentukan indeks dari titik koordinat yang telah didefinisikan tersebut, gunakan fungsi `wrf.ll_to_xy()`. Nilai yang dikeluarkan adalah indeks dari lokasi terdekat.

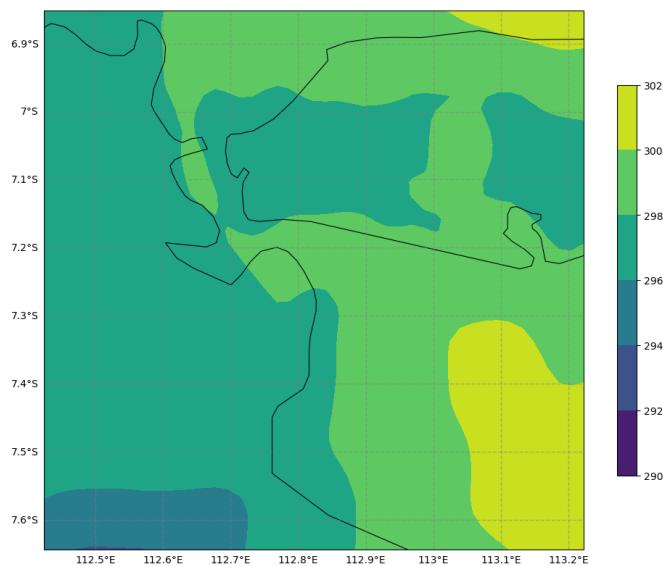
```

latlon_idx = wrf.ll_to_xy(wrf_d03, lats_sel, lons_sel)
wrf_t2_sel = wrf_t2[:, latlon_idx[1], latlon_idx[0]]

```

3. Lakukan plot seri waktu

1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)



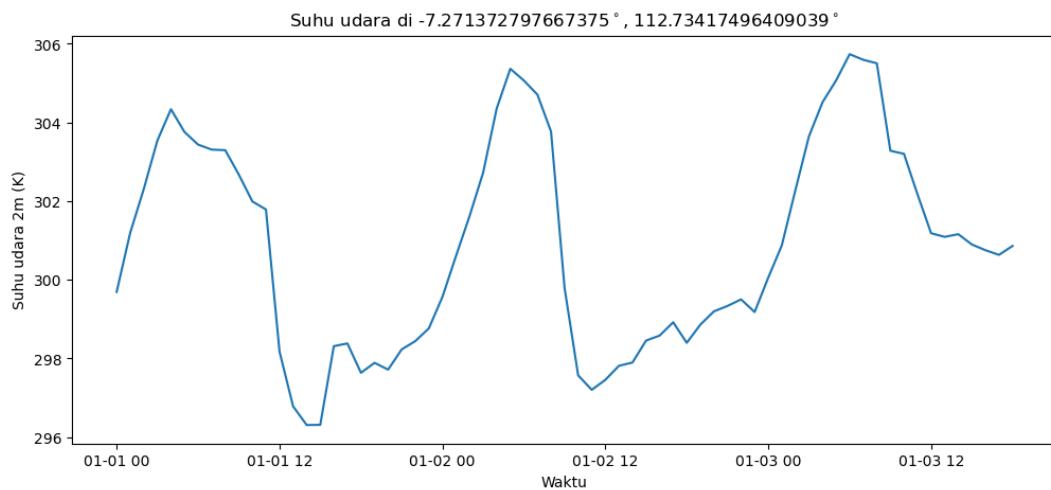
Gambar 1.17: Grafik suhu udara 2-meter di atas permukaan tanah pada 2022-01-02 13:00:00 UTC

```
# Mengatur ukuran grafik
fig = plt.figure(figsize=(12, 5))
ax = plt.axes()

# Plot -> x: waktu, y: suhu udara (K)
ax.plot(wrf_time, wrf_t2_sel.values)

# Mengatur label dan judul
ax.set_xlabel('Waktu')
ax.set_ylabel('Suhu udara 2m (K)')
ax.set_title(f'Suhu udara di {lats_sel}°\circ, {lons_sel}°\circ')

# Tampilkan grafik
plt.show()
```



1.5.2 Python (tanpa wrf-python)

Anda juga dapat mengolah data luaran WRF tanpa memakai `wrf-python`. Anda bisa menggunakan modul `xarray` untuk membaca dan memanipulasi data. Perlu Anda ketahui bahwa data koordinat (latitude, longitude, dan waktu) dari luaran WRF akan terlihat berbeda karena belum menyesuaikan **CF-Conventions**. Oleh sebab itu, Anda bisa mengkonversi data WRF menjadi format **CF-Conventions** dengan NCL lalu dibaca dengan mengikuti langkah-langkah di bawah ini. Beberapa langkah di bawah ini, kami menjelaskan cara-cara teknik pembacaan data WRF tanpa mengikuti **CF-Conventions**.

1. Impor paket

```
import xarray as xr
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as crs
```

2. Buka salah satu file `wrfout_d0*`, misalnya `wrfout_d03_2022-01-01_00:00:00` dan cetak variabel tersebut, misalkan variabel T2 (*Air Temperature at 2m*: suhu udara 2 meter dari permukaan tanah). Untuk melihat semua variabel, ketik saja nama variabel Python yang telah didefinisikan.

```
# Lokasi folder luaran WRF
wrf_path = '/home/absen/WRF-Model/WRF/test/em_real'
# Membuka file wrfout
wrf_d03 = xr.open_dataset(f"{wrf_path}/wrfout_d03_2022-01-01_00:00:00")
# Mencetak isi semua variabel
wrf_d03
# Mengambil variabel
wrf_t2m = wrf_d03['T2']
```

3. Pada langkah sebelumnya, Anda bisa langsung melakukan pembuatan grafik spasial

```

time      = "2022-01-02 13:00:00"
time      = np.array([time], dtype='datetime64[ns]')
time_idx = np.where(wrf_tme == time)[0]
time_idx = int(time_idx)
wrf_t2m[time_idx, :, :].plot.contourf()

```

Akan tetapi, informasi koordinat spasial seperti latitude dan longitude tidak sesuai, yang ditampilkan hanyalah indeks lokasi (0, 1, 2, ...). Anda perlu memodifikasi `wrf_t2m` dengan cara memodifikasi informasi koordinat dengan mengambil variabel lokasi dan waktu di dalam file `wrfout_d0*`. Untuk latitude dan longitude, Anda bisa mengambil variabel `XLAT` dan `XLONG`. Variabel waktu (`XTIME`) di dalam informasi `coordinates` tidak perlu dimodifikasi. Berikut ini adalah contoh kode untuk memodifikasi informasi koordinat.

```

# Mengambil variabel XLAT dan XLONG
wrf_lat = wrf_d03['XLAT'][0, :, 0].values
wrf_lon = wrf_d03['XLONG'][0, 0, :].values
# Mengambil variabel XTIME
wrf_tme = wrf_d03['XTIME'].values
# Membuat DataArray baru dengan informasi koordinat yang telah dimodifikasi
wrf_t2m_new = xr.DataArray(wrf_t2m.values,
                            coords = {'time': wrf_tme,
                                       'lat': wrf_lat,
                                       'lon': wrf_lon},
                            dims = ['time', 'lat', 'lon'])

```

4. Terakhir, Anda bisa melakukan pembuatan grafik spasial dengan memanggil variabel `wrf_t2m_new` yang telah dimodifikasi informasi koordinatnya. Tampilan grafik sama seperti Gambar 1.17.

```

# Membuat dan mengatur ukuran grafik
fig = plt.figure(figsize=(10, 7))
# Mengatur sistem proyeksi peta
ax = plt.axes(projection=crs.PlateCarree())

# Menambahkan garis pantai. Sumber data: www.naturalearthdata.com
ax.coastlines(linewidth=0.8)

# Menambahkan garis lintang dan bujur
gl = ax.gridlines(draw_labels=True, linewidth=1, color='gray', alpha=0.5,
                   linestyle='--')
gl.top_labels = False # Menghilangkan label bujur di atas
gl.right_labels = False # Menghilangkan label lintang di kanan

# Mengatur nilai untuk skala legenda
lvl = np.arange(290, 304, 2)

# Menambahkan garis kontur terisi

```

```

plt.contourf(wrf_lon, wrf_lat, wrf_t2m_new[time_idx, :, :],
             levels=lvl,
             cmap='viridis')

# Menambahkan legenda warna
plt.colorbar(ax=ax, shrink=0.7)

# Menampilkan grafik
plt.show()

```

1.5.3 R

Anda dapat menggunakan RStudio untuk menuliskan skrip R. Untuk mengolah data WRF di R, Anda perlu memasang package `ncdf4` dan `raster` terlebih dahulu. Untuk kebutuhan plot seri waktu, kami menggunakan package `tidyverse`.

1. Sebelum memulai pengetikan kode R, buatlah file skrip R dengan memilih menu **File > New File > R Script**.
2. Di console R pada RStudio, gunakan perintah berikut untuk memasang `ncdf4` dan `raster`.

```
install.packages(c('ncdf4', 'raster'))
```

Anda juga dapat menggunakan menu pada RStudio: **Tools > Install Packaages**. Kemudian, ketik “`ncdf4, raster`” (tanpa tanda petik) pada bagian *Packages (separate multiple with space or comma)*. Setelah itu, klik *Install*.

3. Kami telah menyediakan fungsi skrip R untuk membuka file luaran WRF. Pada teks editor di RStudio, masukkan perintah berikut untuk mengimpor package `ncdf4` dan `raster` beserta skrip `wrf-raster.R`. Fungsi `wrf-raster.R` dapat diunduh di <https://github.com/agungbaruna/modul-pemodelan-iklim/blob/main/scripts/wrf-raster.R>. Letakkan fungsi ini di dalam folder **Project RStudio** Anda.

```

library(ncdf4)
library(raster)
library(tidyverse)
source('wrf-raster.R')

```

Anda dapat menjalankan ketiga baris kode tersebut dengan memblok seluruh baris atau arahkan kursor pada akhir kode kemudian tekan tombol **CTRL + ENTER**.

4. Saat Anda menjalankan `source('wrf-raster.R')`, muncul Functions dengan nama `wrf.raster` pada jendela **Environment** di RStudio (letak jendela di sebelah kanan atas).
5. Untuk menggunakan fungsi `wrf.raster()`, Anda hanya membutuhkan argumen `wrf.file` (nama folder `wrfout_*`) dan `var.name` (nama variabel di dalam `wrfout_*`). Nilai kembalian setelah menjalankan fungsi ini berbentuk `RasterBrick`. Argumen `nlev` dapat dicantumkan dengan angka

1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)

bilangan bulat (1, 2, 3, ...) khusus untuk variabel yang memiliki variasi terhadap ketinggian dan kedalaman, seperti suhu udara atau suhu tanah. Variabel lain yang tidak tercantum seperti curah hujan, sudah tersedia di dalam fungsi ini (`var.name = rain`).

6. Sebagai contoh mengambil variabel curah hujan.

```
rain <- wrf.raster(wrf.file =
  ↵ '/home/absen/WRF-Model/WRF/test/em_real/wrfout_d01_2022-01-01_00:00:00',
  ↵ var.name = 'rain')
rain
```

```
class      : RasterBrick
dimensions : 32, 32, 1024, 67 (nrow, ncol, ncell, nlayers)
resolution : 0.1581326, 0.1567945 (x, y)
extent     : 110.2099, 115.2701, -9.821663, -4.804237 (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs
source     : memory
names      : layer.1,   layer.2,   layer.3,   layer.4,   layer.5,   layer.6, ...
min values :       0,       0,       0,       0,       0,       0, ...
max values : 0.000000, 2.944275, 2.439148, 6.013021, 14.848666, 47.531946, ...
time       : 2022-01-01 00:00:00, 2022-01-03 18:00:00 (min, max)
```

Pada respons melalui console R, terdapat berbagai informasi seperti `dimensions`, `resolution`, `crs`, serta `time`. Anda dapat mudah memahami isi dari file `wrfout_d01*` tersebut.

7. Anda bisa langsung membuat grafik spasial dengan perintah 1 baris ini dan ditampilkan pada Gambar 1.18. Cukup sederhana.

```
plot(rain)
```

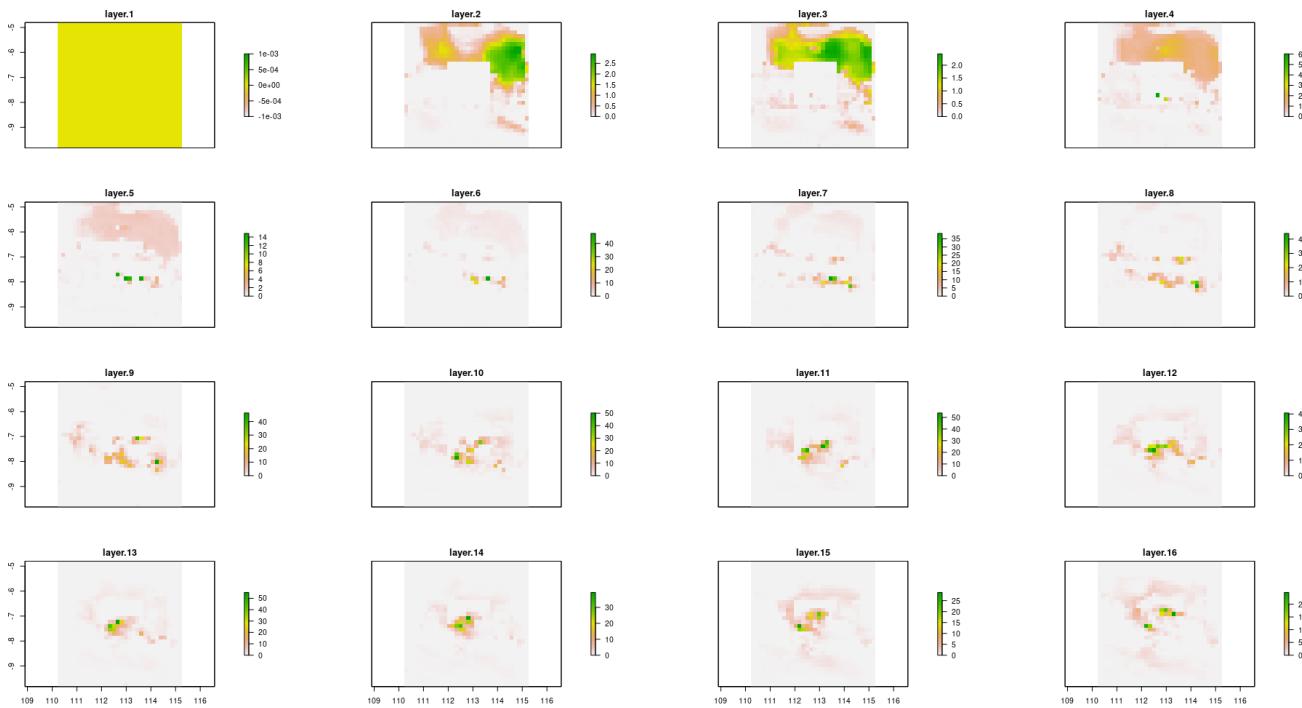
8. Untuk membuat grafik seri waktu pada lokasi tertentu, Anda dapat menggunakan fungsi `extract()` dan `plot()`

```
# Waktu
time_sel <- getZ(rain)

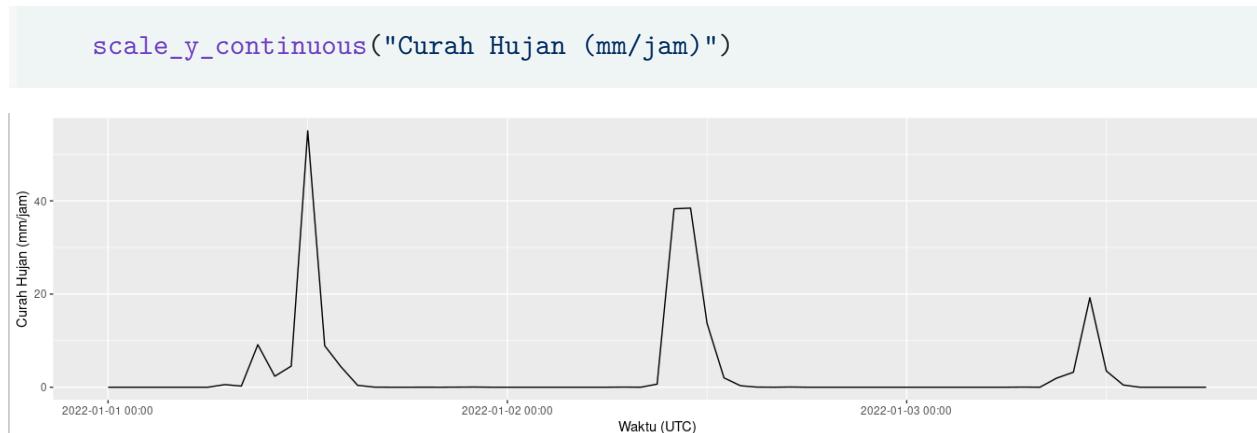
# Lokasi
lats_sel <- -7.271372797667375
lons_sel <- 112.73417496409039

# Ekstrak nilai curah hujan berdasarkan lokasi
rain_sel <- extract(rain, data.frame(x = lons_sel, y = lats_sel))

# Plot grafik seri waktu
ggplot() +
  geom_line(aes(x = time_sel, y = rain_sel[1,])) +
  scale_x_datetime("Waktu (UTC)", date_labels = "%Y-%m-%d %H:%M") +
```



Gambar 1.18: Plot curah hujan per 1 jam dari wrfout_d01*



1.5.4 NCL

NCAR telah menyediakan contoh skrip pengolahan data WRF dengan NCL. Anda dapat mengakses lebih banyak di <https://www.ncl.ucar.edu/Applications/wrf.shtml>. File netcdf luaran WRF pada dasarnya berbeda dengan file netcdf pada umumnya karena tidak mengikuti pedoman *Climate and Forecast Convention*. Alhasil, mengolah data WRF menggunakan aplikasi lain seperti ArcMap atau Matlab cukup rumit. Jika Anda ingin cukup mudah mengolahnya dengan aplikasi lainnya, skrip NCL untuk mengubah WRF menjadi CF-Convention dibuat oleh Mark Seefeldt dan telah tersedia di https://sundowner.colorado.edu/wrfout_to_cf/wrfout_to_cf.ncl. Berikut ini adalah cara penggunaan skrip `wrfout_to_cf.ncl`.

1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)

1. Aktifkan terlebih dahulu *environment ncl* pada terminal. Tulisan (`base`) menjadi (`ncl`) setelah perintah berhasil dipanggil.

```
conda activate ncl
```

2. Anda hanya perlu memasukkan variabel `dir_in` (lokasi folder wrfout), `dir_out` (lokasi folder wrfout setelah dikonversi), `file_in` (nama file wrfout), dan `file_out` (nama file wrfout setelah dikonversi).
3. Masukkan perintah berikut.

```
ncl 'dir_in="/home/absen/WRF-Model/WRF/test/em_real/"'  
    ↵ 'file_in="wrfout_d01_2022-01-01_00:00:00"'  
    ↵ 'file_out="wrfout_cf_d01_2022-01-01_00:00:00.nc"' wrfout_to_cf.ncl
```

4. Anda dapat mengolah data luaran WRF setelah dikonversi menjadi CF-Convention pada aplikasi apapun dengan mudah. Hanya saja, Anda perlu memperhatikan perubahan nama variabel dan sesuaikan dengan data WRF asli.

Selain mengubah menjadi CF-Convention, Anda dapat langsung menerapkan skrip NCL untuk analisis maupun visualisasi. Sebagai contoh skrip berikut ini untuk menampilkan suhu udara dekat permukaan dari luaran `wrfout_d01*`. Perlu diperhatikan bahwa simbol ; adalah komentar di dalam NCL. Anda dapat mengganti format file dari luaran skrip ini (mis. `png`, `pdf`, atau `x11`).

```
;---Read file  
fpath      = "/home/absen/WRF-Model/WRF/test/em_real/"  
filename   = "wrfout_d01_2022-01-01_00:00:00"  
a          = addfile(fpath+filename, "r")  
  
;---Get CEN_LAT and CEN_LON from WRF metadata (in global attributes)  
cen_lat   = a@CEN_LAT  
cen_lon   = a@CEN_LON  
  
;---Select time index  
nt = 30  
  
;---Read temperature at first time step  
tc = wrf_user_getvar(a, "tc", nt)  
  
;---Open worksheet  
wks = gsn_open_wks("png", "wrf_nogsn") ; Format file luaran grafik, contoh png  
  
;---Set up resource list  
res           = True  
res@gsnDraw   = False  
res@gsnFrame  = False  
res@tfDoNDCOverlay = True
```

```

;---Plotting options for air temperature
opts_r             = res

opts_r@cnFillOn      = True
opts_r@cnLevelSelectionMode = "ExplicitLevels"
opts_r@cnLevels       = (/ 20, 22, 24, 26, 28, 30, 32 /)
opts_r@cnSmoothingOn = True
opts_r@cnSmoothingDistanceF = .005

;---Option for plt_res
plt_res = res

;---Option for plt_res
map_res = res
map_res@mpDataBaseVersion      = "HighRes"           ; Jika memilih
  ↵ "HighRes", Anda harus mengunduh semua file di
map_res@mpDataResolution       = "FinestResolution" ;
  ↵ https://www.io-warnemuende.de/rangs-en.html
map_res@mpCenterLatF           = cen_lat           ; dan letakkan di
  ↵ folder $NCARG_ROOT/lib/ncarg/database/rangs
map_res@mpCenterLonF           = cen_lon
map_res@mpGeophysicalLineColor = "black"
map_res@mpGeophysicalLineThicknessF = 2.0
map_res@mpGridSpacingF          = 1.0
map_res@mpGridLineColor         = "black"

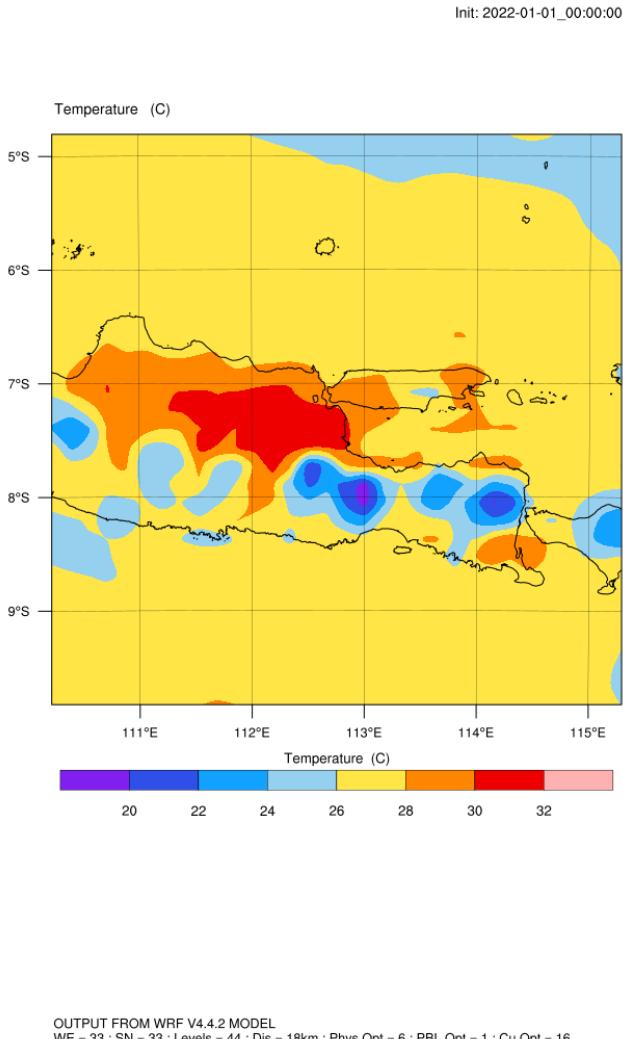
;---Plot
contour_tot = wrf_contour(a, wks, tc(0,:,:), opts_r)
plot = wrf_map_overlays(a, wks, (/contour_tot/), plt_res, map_res)

```

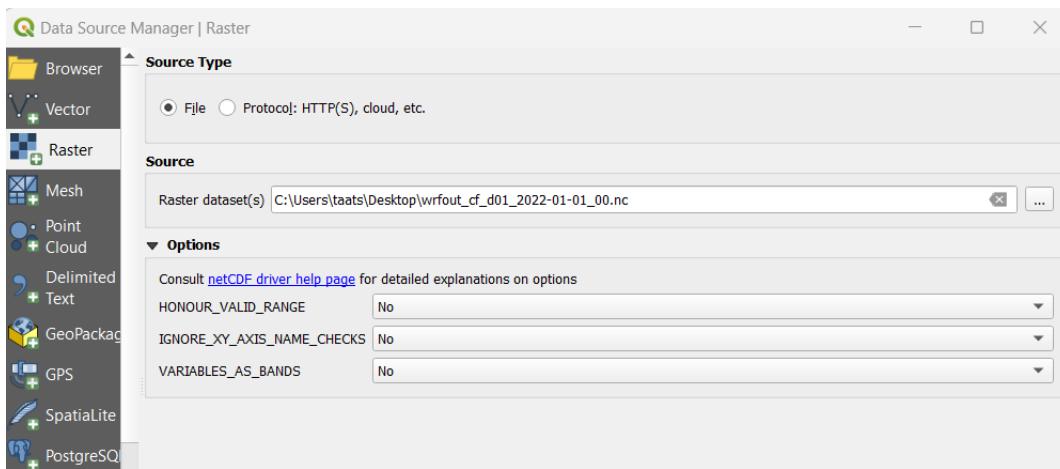
1.5.5 QGIS

Aplikasi QGIS dapat juga digunakan untuk mengolah data WRF. Untuk memudahkan pengolahan data WRF, Anda perlu mengubah data WRF menjadi CF-Convention dengan skrip NCL sebelumnya. Setelah itu, Anda dapat mengimpor data WRF yang telah terkonversi dalam format NetCDF Classic ke QGIS. Berikut ini adalah langkah-langkahnya.

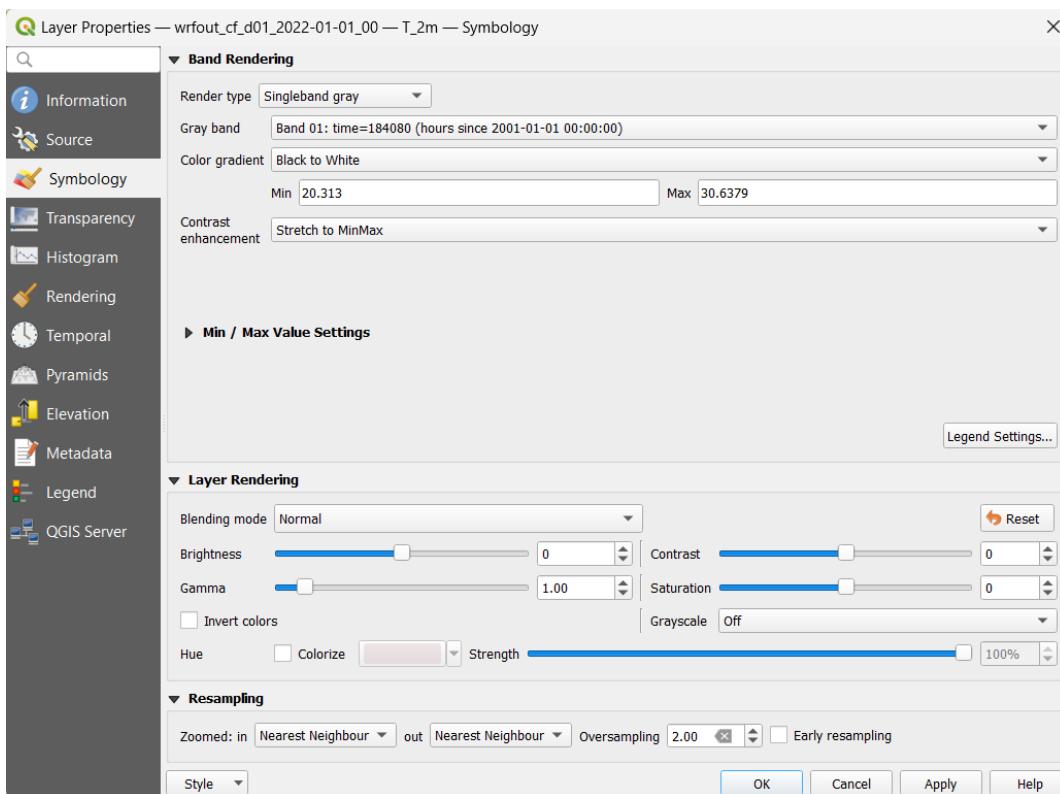
1. Pada menu QGIS, pilih **Layer > Add Layer > Add Raster Layer** (CTRL + SHIFT + R).
2. Pada bagian **Source**, klik ikon ... dan pilih file `wrfout_*` yang telah dikonversi menjadi CF-Convention.



Gambar 1.19: Plot suhu udara dekat permukaan pada 2022-01-02 06:00:00 UTC



3. Pilih variabel yang ingin Anda tampilkan. Misalnya, Anda dapat memilih **T_2m** untuk menampilkan suhu udara pada ketinggian 2 meter. Klik **Add Layers**, kemudian klik **Add**.
4. Klik kanan pada raster yang telah diimpor dan pilih **Properties**. Pada **Band Rendering** di bagian *Render type*, pilih Singleband Gray. Pada bagian *Gray band*, terdapat pilihan nomor *Band* yang berisikan waktu.

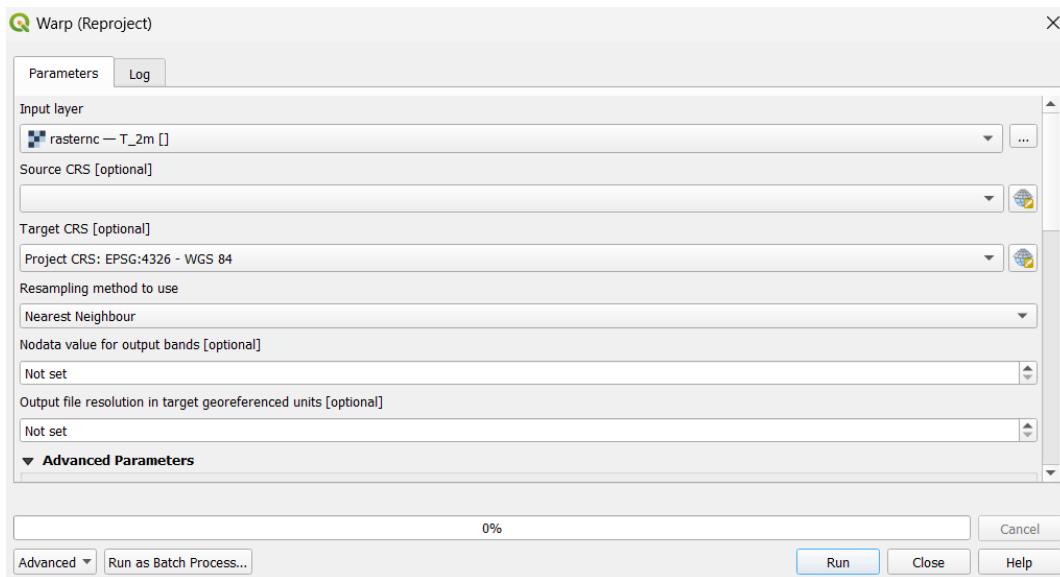


Nilai maksimum dan minimum suhu udara dapat terlihat di bawah layer.

5. Setelah berhasil mengimpor, data raster dari WRF belum terproyeksi ke koordinat. Default sistem proyeksi pada QGIS kami adalah **EPSG:4326 - WGS 84**. Untuk mengubah sistem proyeksi, klik menu **Raster > Projections > Warp (Reproject)**. Bagian **Input layer** diisi dengan raster yang

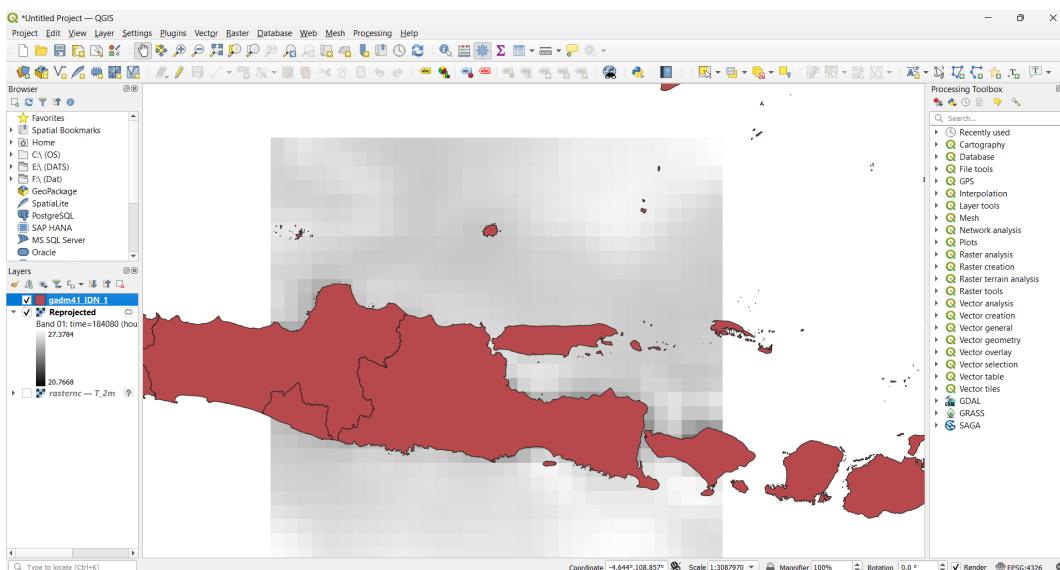
1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)

akan diubah sistem proyeksinya dan Target CRS diisi dengan **EPSG:4326 - WGS 84**. Kemudian, klik **Run**.



Layer raster baru akan muncul dengan nama **Reprojected**. Klik kanan pada raster tersebut dan pilih **Zoom to Layer(s)**

6. Agar lebih meyakinkan, Anda dapat mengimpor file vector dari batas wilayah Indonesia. Untuk data ini, bisa Anda unduh di gadm.org. Pada contoh ini, kami menggunakan file *.json. Untuk mengimpornya, klik menu **Layer > Add Layer > Add Vector Layer** (CTRL + SHIFT + V). Pada bagian **Source**, klik ikon ... dan pilih file `gadm41_IDN_1.json`. Klik **Add Layers**, kemudian klik **Add**.



1.5.6 Julia

Selain R dan Python, bahasa pemrograman Julia juga dapat digunakan untuk mengolah data WRF. Hanya saja, package khusus untuk WRF masih belum tersedia. Julia dapat digunakan di sistem operasi apa saja. Untuk pengguna Windows 10/11, Anda bisa menggunakan WSL atau terminal Windows langsung. Untuk membaca data netcdf dari WRF, algoritma yang kami berikan cukup memerlukan logika sehingga tidak semudah mengolahnya seperti di R. Anda dapat menggunakan package `NCDatasets.jl`, sedangkan untuk membuat grafik menggunakan `Plots.jl`. Berikut ini adalah langkah-langkahnya.

1. Buka terminal interaktif Julia dengan mengetik `julia.exe` pada Command Prompt/PowerShell atau carilah program Julia pada Start Menu.
2. Buka **Jupyter Notebook** pada terminal interaktif Julia dengan perintah berikut ini.

```
using IJulia
notebook()
```

3. Aktifkan package `NCDatasets.jl` dan `Plots.jl`.

```
using NCDatasets
using Plots
```

4. Anda dapat menggunakan fungsi `ncread()` untuk membaca data WRF sekaligus menentukan variabel yang ingin diimpor. Berikut ini adalah contoh untuk mengimpor data suhu udara pada ketinggian 2 meter.

```
wrf_path = "C:\\\\Users\\\\taats\\\\Desktop\\\\";
wrf_file = joinpath(wrf_path, "wrfout_d01_2022-01-01_00");
ds = NCDataset(wrf_file, "r");
t2 = ds["T2"]
```

```
T2 (32 × 32 × 67)
Datatype:    Float32
Dimensions:  west_east × south_north × Time
Attributes:
FieldType          = 104
MemoryOrder        = XY
description        = TEMP at 2 M
units              = K
stagger            =
coordinates        = XLONG XLAT XTIME
```

i Catatan

Perhatikan tanda ; pada akhir baris kode. Tanda ini digunakan untuk menghilangkan respons dari kode tersebut. Cara ini sama seperti sintaks pada Matlab.

1 Dynamical Downscaling: Model Weather Research Forecasting (WRF)

Dimensi pada variabel `t2` bervariasi terhadap longitude (west_east), latitude (south_north), dan waktu (Time) secara berturut-turut. Cara membaca dimensi ini sama seperti pada package `ncdf4` di R. Hanya saja, ini sudah didefinisikan di dalam skrip `wrf-raster.R`.

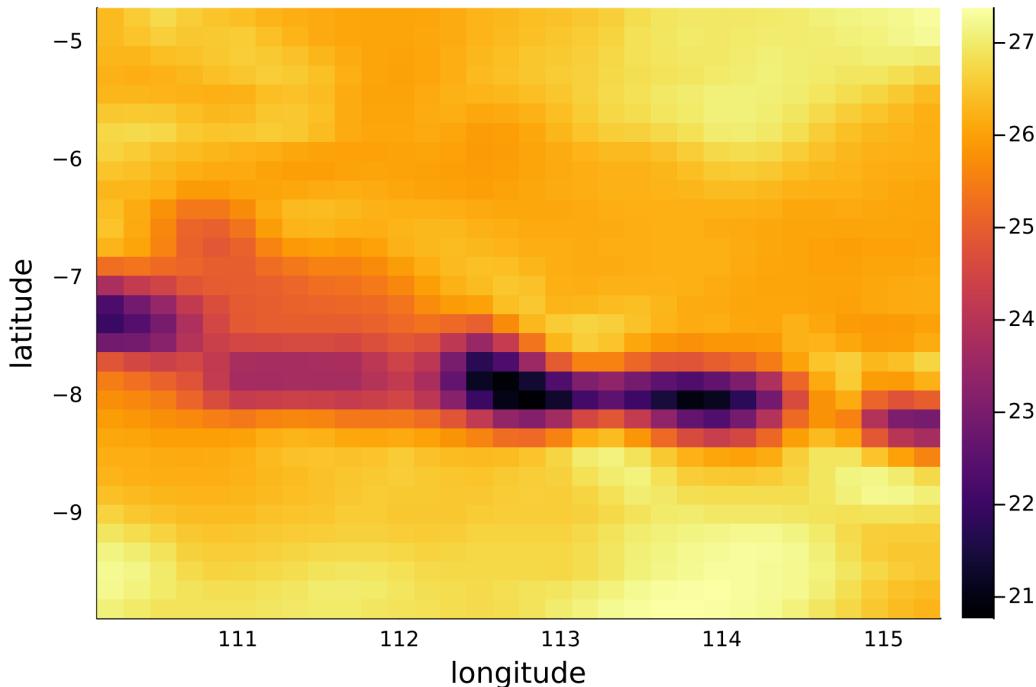
3. Sebelum menampilkan data, Anda perlu mengambil lokasi (latitude dan longitude).

```
lat = ds["XLAT"] [1, :, 1];
lon = ds["XLONG"] [:, 1, 1];
```

4. Anda dapat menampilkan grafik spasial untuk 1 waktu dengan package `Plots.jl` dengan fungsi `heatmap()`.

```
nt = 1 # Indeks waktu = "2022-01-01 00:00:00"
heatmap(lon, lat, transpose(t2[:, :, nt]))
ylabel!("latitude") # Menambahkan label sumbu y
xlabel!("longitude") # Menambahkan label sumbu x
```

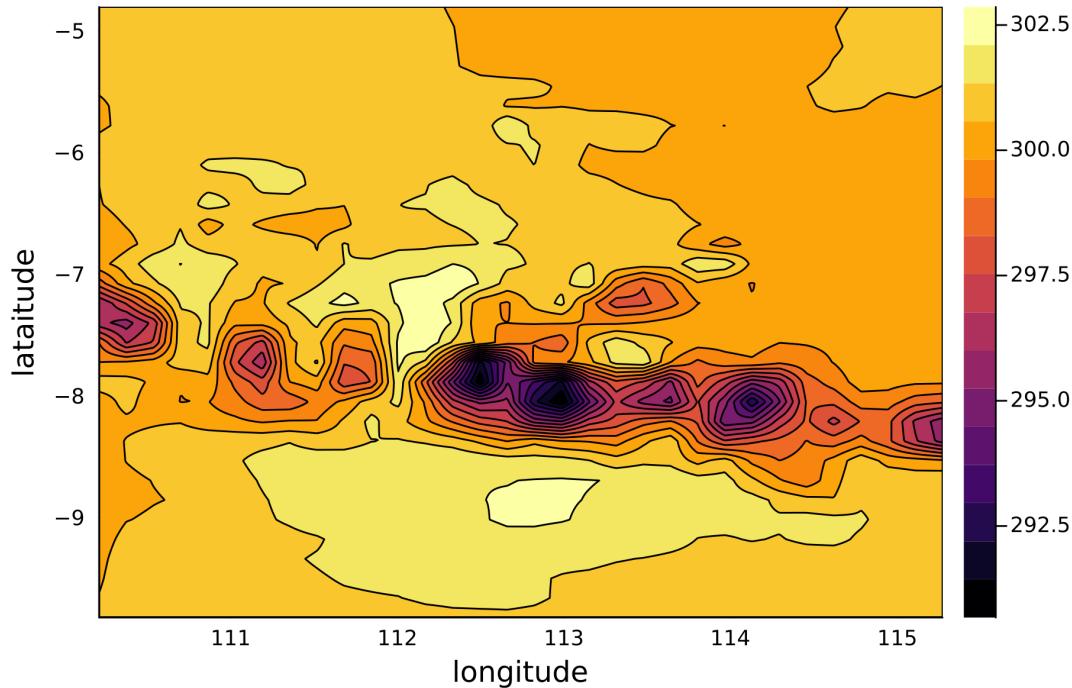
Nilai pada setiap baris dari `t2` bervariasi terhadap longitude, sedangkan pada kolom bervariasi terhadap latitude. Untuk menampilkan grafik dengan koordinat yang benar, Anda perlu membalikkan (baris ke kolom, dan sebaliknya) pada matriks `t2` dengan fungsi `transpose()`.



5. Selain divisualisasikan dalam bentuk grid/raster, Anda juga dapat menampilkan dalam bentuk kontur terisi dengan fungsi `contourf()` dari package `Plots.jl`.

```
nt = 10 # Indeks waktu = "2022-01-01 09:00:00"
contourf(lon, lat, transpose(t2[:, :, nt]))
```

```
ylabel!("latitude") # Menambahkan label sumbu y
xlabel!("longitude") # Menambahkan label sumbu x
```



6. Untuk grafik seri waktu, Anda dapat menggunakan fungsi `plot()` dari package `Plots.jl`. Berikut ini adalah contoh untuk menampilkan grafik seri waktu suhu udara pada ketinggian 2 meter.

```
using Dates

# Mengambil variabel waktu
time = ds["XTIME"];
ticks = Dates.format.(time, "yyyy-mm-dd HH:MM");

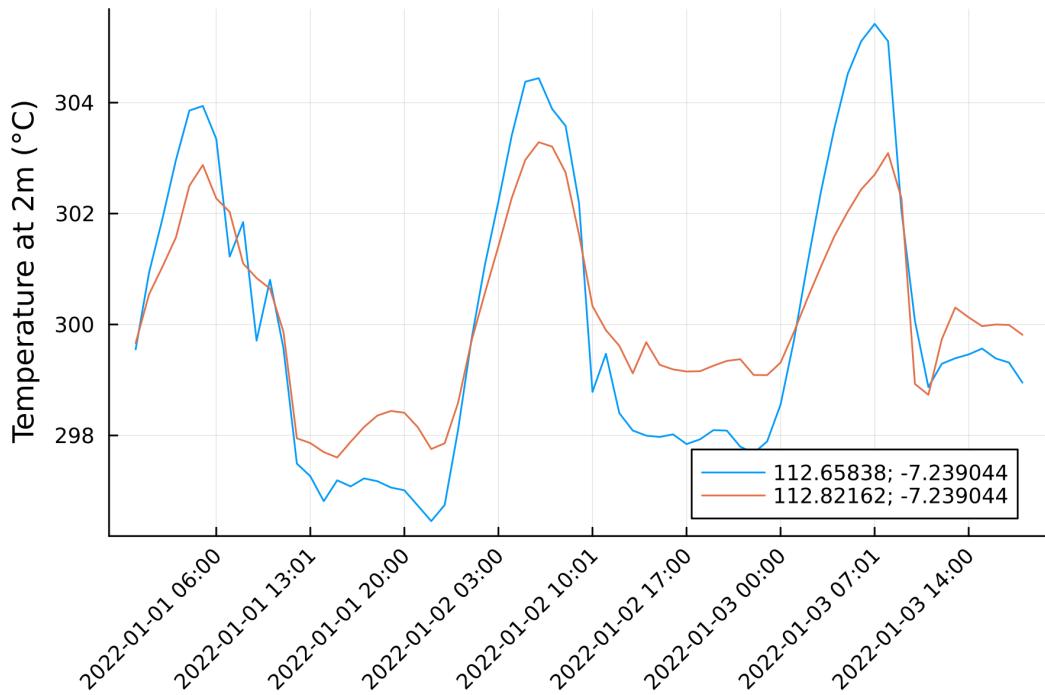
# Pemilihan lokasi
lats_sel = -7.27;
lons_sel = 112.73;

lat_idx = findall(y -> (y > lats_sel - 0.1) && (y < lats_sel + 0.1), lat);
lon_idx = findall(x -> (x > lons_sel - 0.1) && (x < lons_sel + 0.1), lon);

lons = lon[lon_idx];
lats = lat[lat_idx];

# Membuat grafik seri waktu
plot(ticks, t2[lon_idx[1], lat_idx[1], 1:length(time)],
      labels="$(lons[1]); $(lats[1])", xrotation=45)
```

```
plot!(ticks, t2[lon_idx[2], lat_idx[1], 1:length(time)],
      labels=$(lons[2]); $(lats[1]))
ylabel!("Suhu udara 2m (°C)") # Menambahkan label sumbu y
xlabel!("Waktu (per 1 jam)") # Menambahkan label sumbu x
```



2 Model Pendugaan Radiasi Matahari

2.1 Pendahuluan

Radiasi matahari sebagai sumber energi utama memiliki peran dalam pertukaran energi di atmosfer, lautan, dan daratan melalui siklus, yaitu hidrologi, karbon, dan energi. Berbagai fenomena cuaca bumi, seperti pembentukan awan hujan, siklon, dan turbulensi dipengaruhi oleh intensitas serta sebaran radiasi matahari yang diterima oleh bumi. Intensitas radiasi matahari yang diterima sampai atmosfer terluar bumi adalah $1366 \pm 7 \text{ W/m}^2$. Ini disebut juga sebagai konstanta penyinaran matahari dengan tetapan 1367.6 W/m^2 [26]. Sampai pada permukaan bumi, intensitas radiasi matahari (insolasi) menjadi berkurang menjadi seperempatnya, yaitu sekitar $341.4 \pm 1.2 \text{ W/m}^2$, dengan energi yang dipantulkan dan diserap secara berturut-turut sebesar $106.1 \pm 6.5 \text{ W/m}^2$ dan $233.8 \pm 3.8 \text{ W/m}^2$ [28].

Pengukuran radiasi matahari secara langsung masih terbilang sedikit dengan periode perekaman data historis yang pendek. Padahal, pengamatan radiasi matahari di permukaan bumi sangat penting dalam menjelaskan variabilitas dan perubahan iklim. Pada pemodelan iklim yang terus berkembang untuk menjelaskan kondisi iklim di masa lalu, saat ini dan masa depan tak lepas dari masukan data radiasi matahari dan konsentrasi karbon di atmosfer. Untuk mengetahui besaran nilai penduga radiasi matahari di suatu wilayah, model mekanistik maupun empirik dapat digunakan dengan masukan beberapa data historis variabel iklim lainnya seperti curah hujan dan suhu udara.

Pada bagian pertama (Section 2.2), Anda akan mempelajari cara menghitung radiasi matahari ekstraterestrial (insolasi) menggunakan model mekanistik serta menampilkan sebaran nilainya berdasarkan letak lintang dan Julian Days. Bagian kedua (Section 2.3) dan ketiga (Section 2.4), Anda akan menghitung radiasi matahari yang diterima di permukaan bumi menggunakan model empirik. Data pendukung bagian kedua dan ketiga menggunakan data cuaca di Bandara Laguardia, New York, Amerika Serikat (*LaguardiaAirport-NYC.xlsx*). *Worksheet* ke-3 (*RawData*) adalah data yang akan diolah dengan berisikan 7 kolom: DOY (*Day of Year*, 1 sampai 365 atau 366), YEAR (tahun), PRCP (curah hujan, mm), TAVG (suhu udara rata-rata, $^{\circ}\text{C}$), TMAX (suhu udara maksimum, $^{\circ}\text{C}$), TMIN (suhu udara minimum, $^{\circ}\text{C}$), dan SRAD (radiasi matahari langsung, W/m^2). Data pada *worksheet* tersebut sudah dirapikan sehingga Anda dapat langsung mengolahnya. Periode data yang digunakan dibagi menjadi 2, yaitu untuk pembuatan dan validasi model dengan pemilihan tahun 1998-2018 dan 2019-2020, secara berturut-turut.

2.2 Pendugaan Radiasi Matahari Insolasi Harian

Nilai insolasi matahari sebesar 341.4 W/m^2 merupakan rata-rata secara global. Terdapat variasi nilai tersebut berdasarkan letak geografis dan waktu. Untuk menduga nilai insolasi matahari di suatu wilayah, data yang diperlukan hanyalah letak lintang. Persamaan rata-rata harian insolasi matahari di suatu wilayah dapat dihitung dengan persamaan berikut.

2 Model Pendugaan Radiasi Matahari

$$\bar{E} = \frac{S_0}{\pi} \cdot \left(\frac{a}{R}\right)^2 \cdot [h'_0 \cdot \sin(\phi) \cdot \sin(\delta_s) + \cos(\phi) \cdot \cos(\delta_s) \cdot \sin(h_0)]$$

dengan simbol-simbol sebagai berikut:

- \bar{E} : rata-rata harian insolasi matahari (W/m^2)
- S_0 : konstanta penyinaran matahari ($= 1366.7 \text{ W/m}^2$)
- a : rata-rata jarak bumi ke matahari ($= 149Gm$)
- R : jarak bumi ke matahari setiap hari (Gm)
- h'_0 : sudut saat matahari terbit dan terbenam (radian)
- ϕ : letak lintang
- δ_s : sudut deklinasi matahari

Sudut deklinasi matahari (δ_s) dapat dihitung dengan persamaan berikut:

$$\delta_s = 23.44 \cdot \cos\left(\frac{360}{365} \cdot (d - 172)\right)$$

dengan d adalah julian day. Sudut saat matahari terbit dan terbenam (h_0) dapat dihitung dengan persamaan berikut:

$$h_0 = \arccos(\min[1, \max[-1, -\tan(\phi) \cdot \tan(\delta_s)]])$$

Perhitungan

Untuk mengetahui sebaran nilai insolasi matahari, letak lintang yang digunakan dari -90° sampai 90° dengan jarak 1° . Julian day yang digunakan dari 1 sampai 365.

R

1. Impor paket `ggplot2`, `metR`, dan `dplyr`

```
library(ggplot2)
library(dplyr)
library(metR)
```

2. Membuat array letak lintang dan Julian day

```
lat <- seq(-90, 90, 1)
doy <- seq(1, 365, 1)
```

3. Membuat fungsi untuk menghitung insolasi harian matahari.

```
insolation <- function(lat, doy) {
  # Konstanta
```

```

S0 <- 1366.7
a <- 149

# Konversi latitude ke radian
lat_radian <- pi / 180 * lat

# Menghitung jarak bumi ke matahari untuk setiap DOY
v <- 2 * pi * (doy - 4) / 365.256363 # radian
R <- a * (1 - 0.0167 ^ 2) / (1 + 0.0167 * cos(v)) # radian

# Menghitung insolasi harian
delta_s <- 23.44 * pi / 180 * cos(2 * pi * (doy - 172) / 365)
h0 <- acos(pmin(1, pmax(-1, -tan(lat_radian) * tan(delta_s))))
E <- S0 / pi * (a / R)^2 * (h0 * sin(lat * pi / 180) * sin(delta_s) +
  cos(lat * pi / 180) * cos(delta_s) * sin(h0))
  return(E)
}

```

4. Membuat *dataframe* untuk menyimpan hasil perhitungan

```

df <- as_tibble(expand_grid(lat = lat, doy = doy))
df <- df %>% mutate(E_bar = insolation(lat, doy))

```

5. Membuat *plot* sebaran nilai insolasi matahari

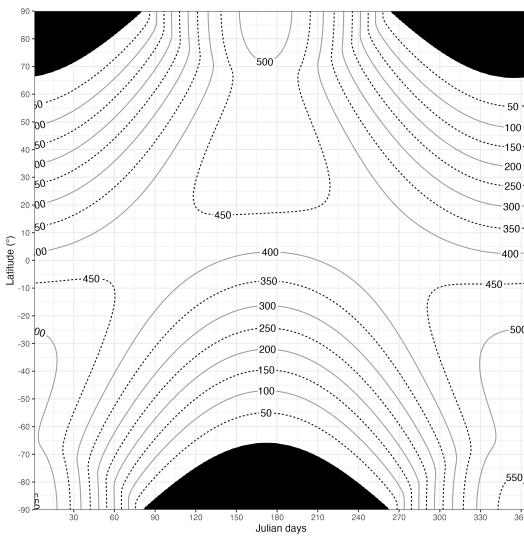
```

kontur_50 <- seq(50, 550, 100)
kontur_100 <- seq(100, 600, 100)

ggplot(df, aes(x = doy, y = lat)) +
  # Menambahkan garis kontur untuk per skala 100
  geom_contour(aes(z = E_bar, linetype = 'dashed'), show.legend = F, color =
    'grey60', breaks = kontur_100) +
  # Menambahkan garis kontur untuk per skala 50
  geom_contour(aes(z = E_bar, linetype = 'dotted'), show.legend = F, color =
    'black', breaks = kontur_50) +
  # Menambahkan kontur terisi hanya untuk nilai E_bar = 0
  geom_contour_filled(aes(z = E_bar), breaks = c(0, 1), fill = 'black') +
  # Menambahkan teks pada kontur
  geom_text_contour(aes(z = E_bar), stroke = 0.3, skip = 0) +
  # Mengatur tema
  scale_x_continuous('Julian days', expand = c(0, 0), breaks = seq(0, 365,
    30)) +
  scale_y_continuous('Latitude (°)', expand = c(0, 0), breaks = seq(-90, 90,
    10)) +
  theme_bw()

```

2 Model Pendugaan Radiasi Matahari



Gambar 2.1: Variasi rata-rata harian insolasi matahari terhadap posisi lintang

Python

2.3 Model Pendugaan Ball et al. (2004)

Ball et al. [2] membangun model empiris untuk menduga radiasi matahari di permukaan bumi dengan masukan suhu udara maksimum, suhu udara minimum, curah hujan, dan *julian days*. Persamaan yang digunakan berupa regresi linier berganda, yaitu

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \dots + \beta_{12} X_{12}$$

di mana keterangan setiap variabel bebas dapat dilihat pada tabel berikut.

Tabel 2.1: Keterangan variabel bebas dari persamaan empirik [2]

Prediktor	Keterangan
X_1	Curah hujan (mm)
X_2	Suhu udara maksimum (°C)
X_3	Suhu udara minimum (°C)
X_4	Day of Year
X_5	(Curah hujan) ²
X_6	(Suhu udara maksimum) ²
X_7	(Suhu udara minimum) ²
X_8	(Day of Year) ²
X_9	Curah hujan * Suhu udara minimum
X_{10}	Suhu udara maksimum * Suhu udara minimum
X_{11}	Curah hujan * Suhu udara maksimum
X_{12}	Suhu udara maksimum * Day of Year

Pengolahan Data

R

1. Package yang digunakan dalam pengolahan data di R adalah `tidyverse` dan `readxl`. Jika Anda belum memasang package ini, gunakan perintah berikut.

```
install.packages(c("tidyverse", "readxl"))
```

2. Impor data excel (`LaguardiaAirport-NYC.xlsx`) pada sheet `RawData` dengan perintah berikut.

```
dpath <- "data/" # Lokasi folder data
data <- read_excel(paste0(dpath, "LaguardiaAirport-NYC.xlsx"), sheet =
  ↪ "RawData")
data
```

```
# A tibble: 8,401 × 7
  YEAR DOY PRCP TAVG TMAX TMIN SRAD
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 1998     1     0    -5   -1.1  -8.9 107.
2 1998     2     0     4.7    10   -0.6  82.8
3 1998     3     0     12    15.6   8.3  74.7
4 1998     4     0    11.7   17.2   6.1  78.8
5 1998     5     0     5.9    6.7    5    75.6
6 1998     6     1    10.3    15    5.6  34.3
7 1998     7    28.7    9.5   13.9    5    26.8
8 1998     8     0.8   10.3   16.1   4.4  37.5
9 1998     9     0     12    15    8.9  46.8
10 1998    10     0     7.8    10    5.6  97.6
# ... with 8,391 more rows
# Use `print(n = ...)` to see more rows
```

3. Pilih periode tahun untuk pembuatan dan validasi model dengan perintah berikut.

```
# Pembuatan model
data_train <- data %>% filter(YEAR <= 2018)

# Validasi model
data_test <- data %>% filter(YEAR >= 2019)
```

4. Lakukan perhitungan prediktor ke-5 sampai ke-12 sesuai dengan Tabel 2.1

```
data_train <- data_train %>%
  mutate(
    PRCP_sq = PRCP^2, TMAX_sq = TMAX^2, TMIN_sq = TMIN^2, DOY_sq = DOY^2,
    PRCP_TMAX = PRCP * TMAX, PRCP_TMIN = PRCP * TMIN,
```

2 Model Pendugaan Radiasi Matahari

```

    TMAX_TMIN = TMAX * TMIN, TMAX_DOY = TMAX * DOY
  )

data_test <- data_test %>%
  mutate(
    PRCP_sq = PRCP^2, TMAX_sq = TMAX^2, TMIN_sq = TMIN^2, DOY_sq = DOY^2,
    PRCP_TMAX = PRCP * TMAX, PRCP_TMIN = PRCP * TMIN,
    TMAX_TMIN = TMAX * TMIN, TMAX_DOY = TMAX * DOY
  )

```

5. Lakukan pembuatan model regresi linier berganda pada data `data_train` sesuai dengan model Ball (Tabel 2.1).

```

model <- lm(SRAD ~ PRCP + TMAX + TMIN + DOY +
             PRCP_sq + TMAX_sq + TMIN_sq + DOY_sq +
             PRCP_TMAX + PRCP_TMIN + TMAX_TMIN + TMAX_DOY,
             data = data_train)
summary(model)

```

Call:

```
lm(formula = SRAD ~ PRCP + TMAX + TMIN + DOY + PRCP_sq + TMAX_sq +
   TMIN_sq + DOY_sq + PRCP_TMAX + PRCP_TMIN + TMAX_TMIN + TMAX_DOY,
   data = data_train)
```

Residuals:

Min	1Q	Median	3Q	Max
-230.598	-30.858	2.829	32.497	189.113

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-4.120e+01	4.068e+00	-10.126	< 2e-16 ***
PRCP	-3.315e+00	1.933e-01	-17.151	< 2e-16 ***
TMAX	2.259e+01	8.621e-01	26.203	< 2e-16 ***
TMIN	-2.705e+01	8.212e-01	-32.945	< 2e-16 ***
DOY	1.988e+00	4.015e-02	49.508	< 2e-16 ***
PRCP_sq	2.469e-02	1.324e-03	18.649	< 2e-16 ***
TMAX_sq	-9.876e-01	4.577e-02	-21.577	< 2e-16 ***
TMIN_sq	-1.481e+00	5.217e-02	-28.386	< 2e-16 ***
DOY_sq	-5.281e-03	1.029e-04	-51.317	< 2e-16 ***
PRCP_TMAX	-1.217e-01	1.966e-02	-6.189	6.35e-10 ***
PRCP_TMIN	1.448e-01	2.069e-02	6.998	2.82e-12 ***
TMAX_TMIN	2.602e+00	9.450e-02	27.534	< 2e-16 ***
TMAX_DOY	-1.184e-02	7.625e-04	-15.534	< 2e-16 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1				

```
Residual standard error: 47.29 on 7657 degrees of freedom
Multiple R-squared:  0.7423,    Adjusted R-squared:  0.7419
F-statistic: 1838 on 12 and 7657 DF,  p-value: < 2.2e-16
```

Bisa Anda lihat pada bagian **Multiple R-squared** untuk mengetahui nilai dari koefisien determinasi (R^2). Semua prediktor yang digunakan untuk mengestimasi radiasi matahari signifikan (p-value < 0.05) dengan nilai $R^2=74\%$.

6. Anda dapat melakukan estimasi nilai radiasi matahari pada model yang sudah dibangun dengan menggunakan data validasi (`data_test`) dengan perintah berikut.

```
data_test <- data_test %>%
  mutate(SRAD_pred = predict(model, data_test))
```

7. Kemudian, lakukan perhitungan koefisien determinasi (R^2) dan korelasi Pearson (r) pada `data_test` dengan perintah berikut.

```
# korelasi Pearson (r)
r <- cor(data_test$SRAD, data_test$SRAD_pred)

# koefisien determinasi (R²)
R2 <- r^2

# Print
print(r); print(R2)
```

```
[1] 0.8508384
[1] 0.7239259
```

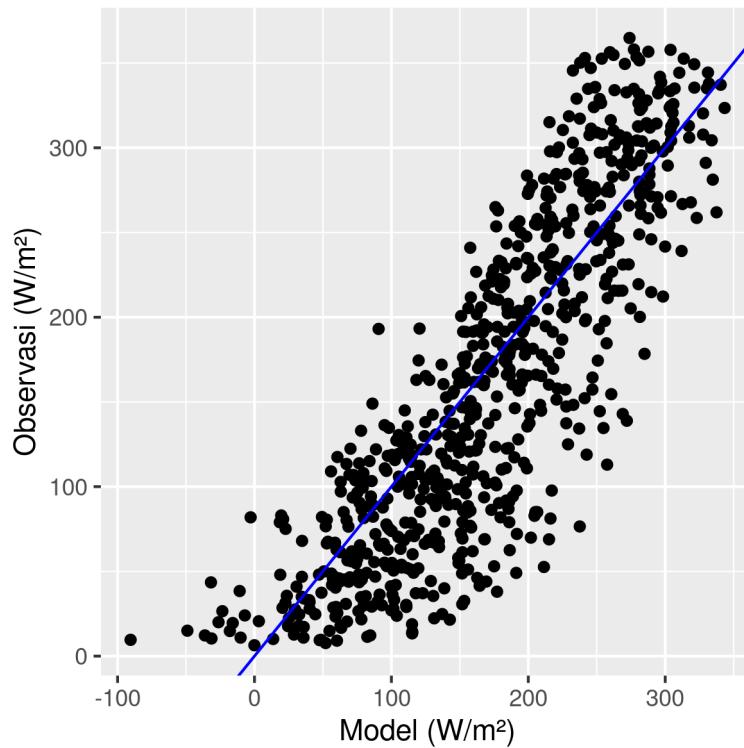
Atau bisa juga dengan melakukan plot antara nilai aktual dan prediksi radiasi matahari dengan menampilkan garis regresi beserta persamaan regresinya ditambah dengan nilai R^2 dan r (Gambar 2.2).

```
ggplot(data_test, aes(x = SRAD_pred, y = SRAD)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "blue") +
  labs(y = "Observasi (W/m²)", x = "Model (W/m²)")
```

8. Jika Anda perhatikan pada Gambar 2.2, nilai radiasi matahari tidak mungkin bernilai negatif dan ini umum terjadi saat menggunakan model regresi. Oleh karena itu, nilai radiasi matahari yang bernilai negatif diubah menjadi 0 (Gambar 2.3).

```
data_test <- data_test %>%
  mutate(SRAD_pred = ifelse(SRAD_pred < 0, 0, SRAD_pred))

ggplot(data_test, aes(x = SRAD_pred, y = SRAD)) +
```



Gambar 2.2: Perbandingan radiasi matahari aktual dengan model (Ball)

```
geom_point() +  
  geom_abline(intercept = 0, slope = 1, color = "blue") +  
  scale_x_continuous(limits = c(0, 400), expand = c(0, 0)) +  
  scale_y_continuous(limits = c(0, 400), expand = c(0, 0)) +  
  labs(y = "Observasi ( $\text{W}/\text{m}^2$ )", x = "Model ( $\text{W}/\text{m}^2$ )")
```

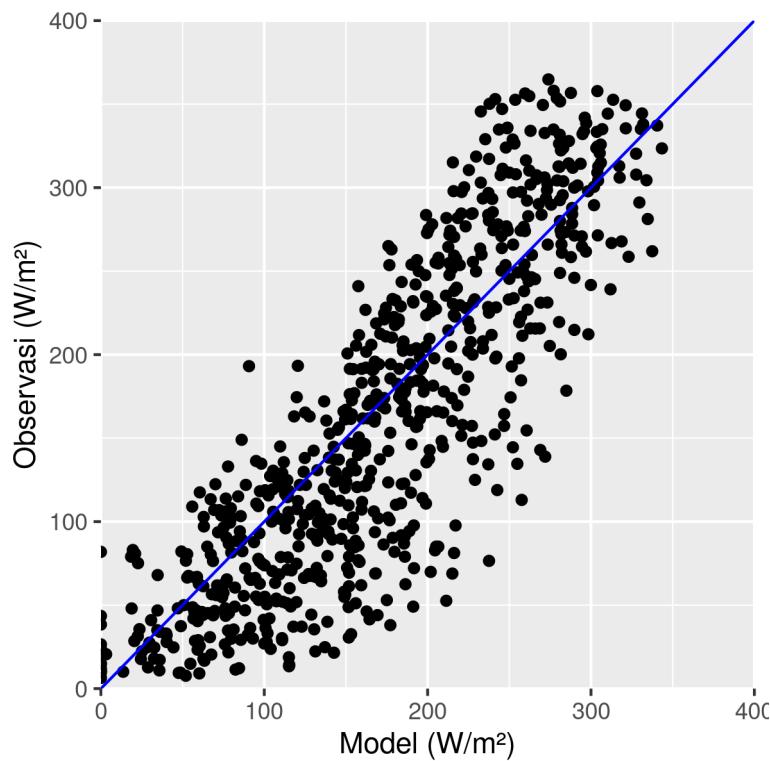
Python

1. Impor package `pandas`, `sklearn`, dan `matplotlib`. Anda perlu memasang `openpyxl` jika belum memasangnya.

```
pip install openpyxl
```

```
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.linear_model import LinearRegression  
from sklearn import metrics
```

2. Impor data dengan perintah berikut.



Gambar 2.3: Perbandingan radiasi matahari aktual dengan model (Ball)

```
path = 'data'
data = pd.read_excel(f'{path}/LaguardiaAirport-NYC.xlsx',
                     sheet_name='RawData')
data
```

	YEAR	DOY	PRCP	TAVG	TMAX	TMIN	SRAD
0	1998	1	0.0	-5.0	-1.1	-8.9	106.8
1	1998	2	0.0	4.7	10.0	-0.6	82.8
2	1998	3	0.0	12.0	15.6	8.3	74.7
3	1998	4	0.0	11.7	17.2	6.1	78.8
4	1998	5	0.0	5.9	6.7	5.0	75.6
...
8396	2020	362	0.0	-0.2	4.4	-2.7	99.4
8397	2020	363	0.0	6.1	11.1	2.2	61.3
8398	2020	364	0.0	5.6	7.2	0.0	97.1
8399	2020	365	0.0	1.8	7.2	-1.0	56.5
8400	2020	366	13.5	7.7	10.0	3.3	24.0

[8401 rows x 7 columns]

3. Pilih periode tahun untuk pembuatan dan validasi model dengan perintah berikut.

2 Model Pendugaan Radiasi Matahari

```
# Pembuatan model
data_train = data[data['YEAR'] < 2019]
# Validasi model
data_test = data[data['YEAR'] >= 2019]
```

4. Lakukan perhitungan prediktor ke-5 sampai ke-12 sesuai dengan Tabel 2.1

```
data_train.loc[:, 'PRCP_sq'] = data_train.loc[:, 'PRCP'] ** 2
data_train.loc[:, 'TMAX_sq'] = data_train.loc[:, 'TMAX'] ** 2
data_train.loc[:, 'TMIN_sq'] = data_train.loc[:, 'TMIN'] ** 2
data_train.loc[:, 'DOY_sq'] = data_train.loc[:, 'DOY'] ** 2
data_train.loc[:, 'PRCP_TMAX'] = data_train.loc[:, 'PRCP'] *
    ↳ data_train.loc[:, 'TMAX']
data_train.loc[:, 'PRCP_TMIN'] = data_train.loc[:, 'PRCP'] *
    ↳ data_train.loc[:, 'TMIN']
data_train.loc[:, 'TMAX_TMIN'] = data_train.loc[:, 'TMAX'] *
    ↳ data_train.loc[:, 'TMIN']
data_train.loc[:, 'TMAX_DOY'] = data_train.loc[:, 'TMAX'] *
    ↳ data_train.loc[:, 'DOY']
```

5. Lakukan pembuatan model regresi linier berganda pada data `data_train` sesuai dengan model Ball (Tabel 2.1).

```
# Model regresi linier
model = LinearRegression()

model.fit(
    # Prediktor
    data_train.loc[:, ['PRCP', 'TAVG', 'TMAX', 'TMIN',
                      'PRCP_sq', 'TMAX_sq', 'TMIN_sq',
                      'DOY_sq', 'PRCP_TMAX', 'PRCP_TMIN',
                      'TMAX_TMIN', 'TMAX_DOY']],
    # Predikton
    data_train.loc[:, 'SRAD']
)
```

6. Oleh karena respon dari package `sklearn` tidak bisa menghasilkan *summary* seperti pada R, Anda perlu membuat fungsi tersendiri. Pada fungsi `metrics`, telah tersedia beberapa metrik yang dapat digunakan untuk mengevaluasi model.

```
def summary(model, x, y):
    # Estimasi
    y_pred = model.predict(x)

    # Jika terdapat nilai negatif, maka nilai tersebut akan diubah menjadi 0
```

```

y_pred[y_pred < 0] = 0

# Metrik
r2 = metrics.r2_score(y, y_pred)
rmse = metrics.mean_squared_error(y, y_pred, squared=False)
mae = metrics.mean_absolute_error(y, y_pred)

print(f'R² = {round(r2, 2)}')
print(f'RMSE = {round(rmse, 2)}')
print(f'MAE = {round(mae, 2)}')

summary(model,
        data_train.loc[:, ['PRCP', 'TAVG', 'TMAX', 'TMIN',
                           'PRCP_sq', 'TMAX_sq', 'TMIN_sq',
                           'DOY_sq', 'PRCP_TMAX', 'PRCP_TMIN',
                           'TMAX_TMIN', 'TMAX_DOY']],
        data_train.loc[:, 'SRAD']
)

```

$R^2 = 0.74$
 RMSE = 47.25
 MAE = 37.6

7. Untuk memvalidasi model, Anda perlu menghitung prediktor ke-5 sampai ke-12 pada data `data_test` sesuai dengan Tabel 2.1. Kemudian, gunakan `model.predict()` untuk menghitung prediksi pada data `data_test`.

```

data_test.loc[:, 'PRCP_sq'] = data_test.loc[:, 'PRCP'] ** 2
data_test.loc[:, 'TMAX_sq'] = data_test.loc[:, 'TMAX'] ** 2
data_test.loc[:, 'TMIN_sq'] = data_test.loc[:, 'TMIN'] ** 2
data_test.loc[:, 'DOY_sq'] = data_test.loc[:, 'DOY'] ** 2
data_test.loc[:, 'PRCP_TMAX'] = data_test.loc[:, 'PRCP'] * data_test.loc[:, 
  ↵ 'TMAX']
data_test.loc[:, 'PRCP_TMIN'] = data_test.loc[:, 'PRCP'] * data_test.loc[:, 
  ↵ 'TMIN']
data_test.loc[:, 'TMAX_TMIN'] = data_test.loc[:, 'TMAX'] * data_test.loc[:, 
  ↵ 'TMIN']
data_test.loc[:, 'TMAX_DOY'] = data_test.loc[:, 'TMAX'] * data_test.loc[:, 
  ↵ 'DOY']

summary(model,
        data_test.loc[:, ['PRCP', 'TAVG', 'TMAX', 'TMIN',
                           'PRCP_sq', 'TMAX_sq', 'TMIN_sq',
                           'DOY_sq', 'PRCP_TMAX', 'PRCP_TMIN',
                           'TMAX_TMIN', 'TMAX_DOY']],
        data_test.loc[:, 'SRAD']
)

```

)

 $R^2 = 0.72$

RMSE = 51.69

MAE = 40.72

8. Untuk membuat grafik dari `data_test` maupun `data_train`, Anda dapat menggunakan `matplotlib` (Gambar 2.4)

```
y_pred = model.predict(data_train.loc[:, ['PRCP', 'TMAX', 'TMIN', 'DOY',
                                             'PRCP_sq', 'TMAX_sq', 'TMIN_sq',
                                             'DOY_sq', 'PRCP_TMAX', 'PRCP_TMIN',
                                             ↵
                                             'TMAX_TMIN', 'TMAX_DOY']])

# Grafik data train
plt.figure(figsize=(10, 10))
plt.scatter(data_train.loc[:, 'SRAD'], y_pred)

plt.xlim(0, 380)
plt.ylim(0, 380)

# Menambahkan label pada sumbu x dan y
plt.xlabel('Model (W/m²)')
plt.ylabel('Observasi (W/m²)')

plt.show()
```

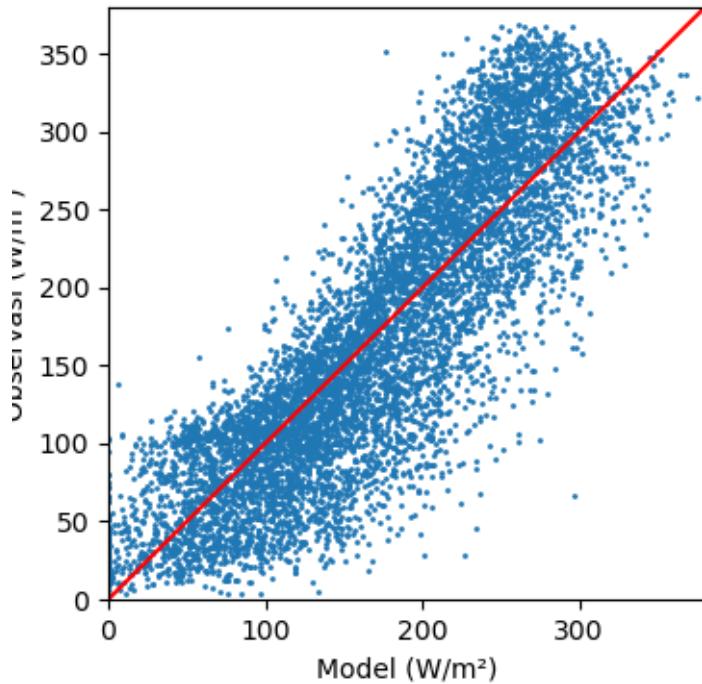
2.4 Model Pendugaan Hunt et al. (1998)

Model pendugaan radiasi matahari lain adalah oleh Hunt [9], yaitu gabungan model mekanistik dan empirik dengan masukan suhu udara maksimum, suhu udara minimum, dan curah hujan. Model mekanistik digunakan untuk menduga radiasi matahari yang berada di permukaan atmosfer (radiasi ekstraterestrial, S_0). Model pendugaan Hunt (1998) adalah sebagai berikut.

$$R_s = a_0 S_0 (T_{max} - T_{min})^{0.5} + a_1 T_{max} + a_2 P + a_3 P^2 + a_4$$

dimana R_s adalah radiasi matahari harian ($MJ\ m^{-2}\ hari^{-1}$), S_0 adalah radiasi matahari di puncak atmosfer ($MJ\ m^{-2}\ hari^{-1}$), P adalah curah hujan (mm), T_{max} adalah suhu udara maksimum ($^{\circ}C$), dan T_{min} adalah suhu udara minimum ($^{\circ}C$). Untuk mengestimasi nilai S_0 , Hunt menggunakan persamaan mekanistik dalam Spitters [25], yaitu:

$$S_0 = S_{sc} \left[1 + 0.033 \cos \left(\frac{360 t_d}{365} \right) \right] \sin(\beta)$$



Gambar 2.4: Perbandingan radiasi matahari aktual terhadap model (Ball)

dimana, S_0 adalah irradiasi ekstra terestrial (W/m^2), S_{sc} konstanta matahari ($1370 \text{ W}/\text{m}^2$), suku \cos adalah jarak tahunan antara bumi dan matahari yang dinyatakan dalam derajat, t_d adalah julian day, dan $\sin(\beta)$ adalah sinus sudut elevasi matahari (satuan detik) yang didefinisikan pada persamaan:

$$\sin(\beta) = 3600 \left[D \sin(\lambda) \sin(\delta) + \frac{24}{\pi} \cos(\lambda) \cos(\delta) \sqrt{(1 - \tan^2(\lambda) \tan^2(\delta))} \right]$$

dimana, λ adalah letak lintang dari lokasi stasiun dan δ adalah sudut deklinasi matahari pada saat julian day dan dinyatakan dalam derajat dengan estimasi pada persamaan:

$$\sin(\delta) = -\sin(23.45) \cos \left(\frac{360 (t_d + 10)}{365} \right)$$

dan D adalah panjang hari (jam) dengan persamaan:

$$D = 12 + \frac{24}{180} \arcsin(\tan(\lambda) \tan(\delta))$$

Pengolahan Data

R

Langkah-langkah pembuatan model radiasi matahari dengan menggunakan model Hunt et al. (1998) adalah sebagai berikut.

1. Impor data Excel. Caranya sama seperti pada subbab sebelumnya.

```
dpath <- "data/" # Lokasi folder data
data <- read_excel(paste0(dpath, "LaguardiaAirport-NYC.xlsx"), sheet =
  ↪ "RawData")
data

# A tibble: 8,401 × 7
  YEAR   DOY   PRCP   TAVG   TMAX   TMIN   SRAD
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 1998     1     0    -5   -1.1   -8.9  107.
2 1998     2     0    4.7    10   -0.6  82.8
3 1998     3     0    12   15.6    8.3  74.7
4 1998     4     0   11.7   17.2    6.1  78.8
5 1998     5     0    5.9    6.7     5  75.6
6 1998     6     1   10.3    15    5.6  34.3
7 1998     7  28.7    9.5   13.9     5  26.8
8 1998     8     0.8  10.3   16.1    4.4  37.5
9 1998     9     0    12    15    8.9  46.8
10 1998    10     0    7.8    10    5.6  97.6
# ... with 8,391 more rows
#   Use `print(n = ...)` to see more rows
```

2. Sebelum Anda melakukan pembuatan model regresi linier berganda, Anda harus melakukan perhitungan S_0 terlebih dahulu. Kami menyediakan fungsi untuk menghitung nilai S_0 seperti pada Section 2.4. Anda hanya memasukkan nilai latitude serta *day of year* pada fungsi ini.

```
S0 <- function(lat, doy){
  # Fungsi untuk menghitung sin(delta)
  sin_delta <- -sinpi(23.45 / 180) * cospi((360 * (doy + 10) / 365) / 180)
  asin_delt <- asin(sin_delta) * 180 / pi

  # sin(lat) * sin(delta)
  s_lat_delt <- sinpi(lat / 180) * sinpi(asin_delt / 180)

  # cos(lat) * cos(delta)
  c_lat_delt <- cospi(lat / 180) * cospi(asin_delt / 180)

  # (sin(lat) * sin(delta)) / (cos(lat) * cos(delta))
  t_lat_delt <- s_lat_delt / c_lat_delt
```

```

# Fungi perhitungan panjang hari (D)
D <- 12 + 24/180 * asin(t_lat_delt) * 180 / pi

# Fungsi perhitungan sudut elevasi matahari
sin_beta <- 3600 * (D * s_lat_delt + 24/pi * c_lat_delt * sqrt(1 -
← t_lat_delt^2))

# Fungsi radiasi matahari ekstra terestrial
S_0 <- 1370 * (1 + 0.033 * cospi(360 / 180 * doy / 365)) * sin_beta

# Konversi J/m2 ke MJ/m2
return(S_0 / 1000000)
}

```

3. Kemudian, tambahkan kolom baru pada `data` yang berisi nilai S_0 dengan perintah berikut.

```

data <- data %>%
  mutate(
    S0 = S0(40.77945, DOY) * 0.0864 # 0.0864 adalah konversi dari MJ/m2 ke
    ← W/m2
  )
data

# A tibble: 8,401 × 8
  YEAR   DOY   PRCP   TAVG   TMAX   TMIN   SRAD     S0
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 1998     1     0     -5    -1.1   -8.9  107.  1.15
2 1998     2     0      4.7    10    -0.6   82.8  1.16
3 1998     3     0     12    15.6    8.3   74.7  1.16
4 1998     4     0    11.7   17.2    6.1   78.8  1.17
5 1998     5     0      5.9    6.7     5    75.6  1.18
6 1998     6     1     10.3    15     5.6   34.3  1.18
7 1998     7    28.7    9.5   13.9     5    26.8  1.19
8 1998     8     0.8   10.3   16.1    4.4   37.5  1.20
9 1998     9     0     12     15     8.9   46.8  1.20
10 1998    10     0     7.8    10     5.6   97.6  1.21
# ... with 8,391 more rows
# Use `print(n = ...)` to see more rows

```

4. Untuk membuat model, lakukan pembagian data menjadi `data_test` dan `data_train` dengan perintah berikut.

```

data_train <- data %>% filter(YEAR <= 2018)
data_test <- data %>% filter(YEAR >= 2019)

```

2 Model Pendugaan Radiasi Matahari

5. Hitunglah $S_0 * (T_{\max} - T_{\min})^{0.5}$ dan $PRCP^2$ dengan menambahkan dua kolom baru pada `data_train` dan `data_test` dengan perintah berikut.

```
data_train <- data_train %>%
  mutate(
    TMAX_TMIN = S0 * (TMAX - TMIN)^0.5,
    PRCP_sq = PRCP^2
  )

data_test <- data_test %>%
  mutate(
    TMAX_TMIN = S0 * (TMAX - TMIN)^0.5,
    PRCP_sq = PRCP^2
  )
```

6. Buat model regresi linier berganda

```
model <- lm(SRAD ~ TMAX_TMIN + TMAX + PRCP + PRCP_sq, data = data_train)
summary(model)
```

Call:
`lm(formula = SRAD ~ TMAX_TMIN + TMAX + PRCP + PRCP_sq, data = data_train)`

Residuals:

Min	1Q	Median	3Q	Max
-303.18	-30.23	5.38	33.75	172.69

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	17.315218	1.349717	12.829	<2e-16 ***
TMAX_TMIN	24.430366	0.284492	85.874	<2e-16 ***
TMAX	0.144476	0.084287	1.714	0.0866 .
PRCP	-4.237145	0.098195	-43.150	<2e-16 ***
PRCP_sq	0.028795	0.001301	22.129	<2e-16 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 47.51 on 7665 degrees of freedom
Multiple R-squared: 0.7396, Adjusted R-squared: 0.7395
F-statistic: 5443 on 4 and 7665 DF, p-value: < 2.2e-16

7. Kemudian, lakukan prediksi pada `data_test` dan evaluasi hasil estimasi model menggunakan metrik R^2 dan korelasi Pearson dengan perintah berikut .

```
data_test <- data_test %>%
  mutate(SRAD_pred = predict(model, data_test))
```

```
# korelasi Pearson (r)
r <- cor(data_test$SRAD, data_test$SRAD_pred)

# koefisien determinasi (R2)
R2 <- r^2

print(r); print(R2)
```

```
[1] 0.8474758
[1] 0.7182153
```

8. Sama seperti Ball, model Hunt juga dapat dipastikan estimasi radiasi matahari bernilai negatif. Untuk mengubahnya menjadi 0, lakukan hal yang sama seperti pada model Ball pada langkah ke-7 dan buat grafiknya (Gambar 2.5)

```
data_test <- data_test %>%
  mutate(SRAD_pred = ifelse(SRAD_pred < 0, 0, SRAD_pred))

ggplot(data_test, aes(x = SRAD_pred, y = SRAD)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "blue") +
  scale_x_continuous(limits = c(0, 400), expand = c(0, 0)) +
  scale_y_continuous(limits = c(0, 400), expand = c(0, 0)) +
  labs(y = "Observasi (W/m2)", x = "Model (W/m2)")
```

Python

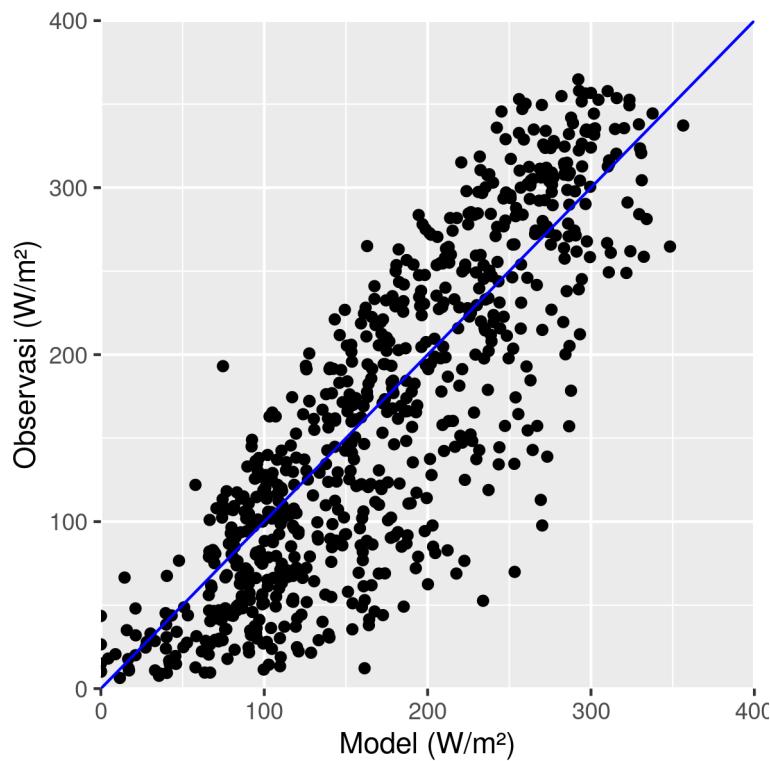
1. Impor package

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

2. Impor data Excel. Caranya sama seperti pada subbab sebelumnya.

```
path = 'data'
data = pd.read_excel(f'{path}/LaguardiaAirport-NYC.xlsx',
                     sheet_name='RawData')
data
```

	YEAR	DOY	PRCP	TAVG	TMAX	TMIN	SRAD
0	1998	1	0.0	-5.0	-1.1	-8.9	106.8
1	1998	2	0.0	4.7	10.0	-0.6	82.8



Gambar 2.5: Perbandingan radiasi matahari aktual terhadap model (Hunt)

2	1998	3	0.0	12.0	15.6	8.3	74.7
3	1998	4	0.0	11.7	17.2	6.1	78.8
4	1998	5	0.0	5.9	6.7	5.0	75.6
...
8396	2020	362	0.0	-0.2	4.4	-2.7	99.4
8397	2020	363	0.0	6.1	11.1	2.2	61.3
8398	2020	364	0.0	5.6	7.2	0.0	97.1
8399	2020	365	0.0	1.8	7.2	-1.0	56.5
8400	2020	366	13.5	7.7	10.0	3.3	24.0

[8401 rows x 7 columns]

- Sebelum Anda melakukan pembuatan model regresi linier berganda, Anda harus melakukan perhitungan S_0 terlebih dahulu. Kami menyediakan fungsi untuk menghitung nilai S_0 seperti pada Section 2.4. Anda hanya memasukkan nilai latitude serta *day of year* pada fungsi ini. Perlu modul tambahan `numpy` untuk menghitung nilai trigonometri.

```
from numpy import sin, cos, pi, arcsin, sqrt

def S0(lat, doy):
    # Fungsi untuk menghitung sin(delta)
```

```

sin_delta = -sin(pi * 23.45 / 180) * cos(pi * (360 * (doy + 10) / 365) /
↪ 180)
asin_delt = arcsin(sin_delta) * 180 / pi

# sin(lat) * sin(delta)
s_lat_delt = sin(pi * lat / 180) * sin(pi * asin_delt / 180)

# cos(lat) * cos(delta)
c_lat_delt = cos(pi * lat / 180) * cos(pi * asin_delt / 180)

# (sin(lat) * sin(delta)) / (cos(lat) * cos(delta))
t_lat_delt = s_lat_delt / c_lat_delt

# Fungi perhitungan panjang hari (D)
D = 12 + 24/180 * arcsin(t_lat_delt) * 180 / pi

# Fungsi perhitungan sudut elevasi matahari
sin_beta = 3600 * (D * s_lat_delt + 24/pi * c_lat_delt * sqrt(1 -
↪ t_lat_delt**2))

# Fungsi perhitungan radiasi matahari
S_0 = 1370 * (1 + 0.033 * cos(pi * 360 / 180 * doy / 365)) * sin_beta

# Konversi J/m2 ke MJ/m2
return S_0 / 1000000

```

4. Kemudian, tambahkan kolom baru pada `data` yang berisi nilai S_0 dengan perintah berikut.

```
data.loc[:, 'S0'] = S0(40.77945, data.loc[:, 'DOY'])
```

5. Untuk membuat model, lakukan pembagian data menjadi `data_test` dan `data_train` dengan perintah berikut.

```

# Pembuatan model
data_train = data[data['YEAR'] < 2019]
# Validasi model
data_test = data[data['YEAR'] >= 2019]

```

6. Hitunglah $S_0 * (T_{max} - T_{min})^{0.5}$ dan $PRCP^2$ dengan menambahkan dua kolom baru pada `data_train` dan `data_test` dengan perintah berikut.

```

# Data train
data_train.loc[:, 'S0_TMAX_TMIN'] = data_train.loc[:, 'S0'] *
↪ sqrt(data_train.loc[:, 'TMAX'] - data_train.loc[:, 'TMIN'])
data_train.loc[:, 'PRCP_sq'] = data_train.loc[:, 'PRCP'] ** 2

```

2 Model Pendugaan Radiasi Matahari

```
# Data test
data_test.loc[:, 'SO_TMAX_TMIN'] = data_test.loc[:, 'SO'] *
    sqrt(data_test.loc[:, 'TMAX'] - data_test.loc[:, 'TMIN'])
data_test.loc[:, 'PRCP_sq'] = data_test.loc[:, 'PRCP'] ** 2
```

7. Buat model regresi linier berganda

```
model = LinearRegression()
model.fit(
    # Prediktor
    data_train.loc[:, ['SO_TMAX_TMIN', 'PRCP', 'PRCP_sq']],
    # Prediktan
    data_train.loc[:, 'SRAD']
)
```

8. Kemudian, lakukan validasi model dengan menggunakan `data_test`.

```
summary(model,
        data_test.loc[:, ['SO_TMAX_TMIN', 'PRCP', 'PRCP_sq']],
        data_test.loc[:, 'SRAD']
)
```

```
R2 = 0.71
RMSE = 52.53
MAE = 41.64
```

9. Terakhir, lakukan pembuatan grafik dari `data_test` (Gambar 2.6)

```
y_pred = model.predict(data_test.loc[:, ['SO_TMAX_TMIN', 'PRCP', 'PRCP_sq']])
y_pred[y_pred < 0] = 0

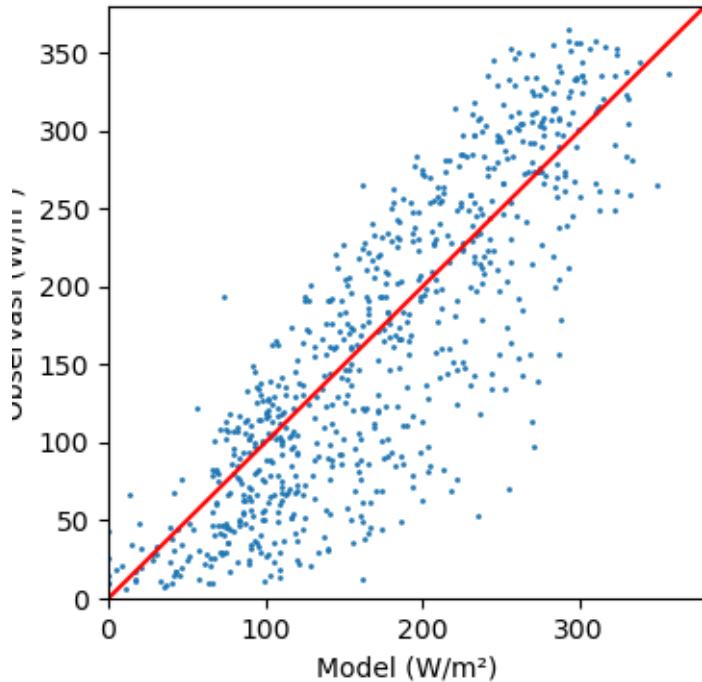
# Visualize
fig = plt.figure(figsize=(4, 4))

# Make plot scatter with small size
plt.scatter(y_pred, data_test.loc[:, 'SRAD'], s=1)

# Make line plot 1:1
plt.plot([0, 400], [0, 400], color='red')

# expand plot area
plt.xlim(0, 380)
plt.ylim(0, 380)
```

```
# Set axis labels
plt.xlabel('Model (W/m2)')
plt.ylabel('Observasi (W/m2)')
plt.show()
```



Gambar 2.6: Perbandingan radiasi matahari aktual terhadap model (Hunt)

3 Koreksi Bias Statistik

3.1 Pendahuluan

Metode koreksi bias biasanya menggunakan fungsi transfer/koreksi yang mengubah data model menjadi data terkoreksi secara statistik. Selain Delta Method [24, 7], metode koreksi bias lain telah dikembangkan, seperti linear scaling correction, power transformation, empirical quantile mapping, dan parametric quantile mapping [13]. Beberapa keterbatasan dalam melakukan koreksi bias, yaitu umumnya tidak memiliki dasar fisika yang kuat, menghasilkan perubahan konsistensi secara spasial, perubahan hubungan antar variabel, melawan aturan hukum konservasi energi, tidak memperhitungkan ketidakpastian dari kumpulan data observasi dan variabilitas internal, dan galat model dan fungsi koreksi umumnya dianggap stasioner terhadap waktu [4, 1, 14]. Ringkasan pengklasifikasian metode koreksi bias dapat dilihat pada Tabel 3.1 [29].

Tabel 3.1: Klasifikasi metode koreksi bias statistik

	Tipe konstan	Tipe variabel	Tipe parametrik	Tipe non-parametrik
Definisi	Nilai statistik pada periode masa depan tidak dimasukkan pada persamaan koreksi bias	Nilai statistik pada periode masa depan dimasukkan pada persamaan koreksi bias	Distribusi parametrik diasumsikan masuk ke simulasi dan/atau data observasi	Distribusi parametrik tidak digunakan dalam koreksi bias

	Tipe konstan	Tipe variabel	Tipe parametrik	Tipe non-parametrik
Karakteristik	<ul style="list-style-type: none"> • Statistik pada periode masa depan tidak mempengaruhi hasil dari koreksi bias • Panjang data koreksi bias dapat diperoleh berapapun panjang periodenya 	<ul style="list-style-type: none"> • Statistik pada periode masa depan mempengaruhi hasil dari koreksi bias • Mudah untuk melakukan penyesuaian perubahan secara statistik dari periode dasar ke masa depan antara data belum terkoreksi dan data terkoreksi • Panjang data terkoreksi bias harus sama dengan data periode dasar 	<ul style="list-style-type: none"> • Kuat secara statistik • Nilai yang tidak realistik dapat dihasilkan jika distribusi tidak diperkirakan dengan baik 	<ul style="list-style-type: none"> • Diperlukan ekstrapolasi untuk nilai-nilai yang lebih/kurang dari periode dasar • Karena derajat bebas sama dengan jumlah parameter, ada permasalahan dalam <i>functional robustness</i> jika asumsinya distribusi non parametrik

Modul ini menjelaskan tata cara melakukan koreksi bias statistik dengan metode Delta dan Distribusi Statistik.

3.2 Metode Delta

Cara paling sederhana untuk melakukan koreksi bias adalah dengan menambahkan/mengalikan data model ke data observasi pada periode dasar. Persamaan 1 umum digunakan pada koreksi data suhu dan Persamaan 2 untuk curah hujan.

$$x_{cor,i} = x_{o,i} + \mu_p - \mu_b \quad (1)$$

$$x_{cor,i} = x_{o,i} \cdot \frac{\mu_p}{\mu_b} \quad (2)$$

dimana $x_{cor,i}$ dan $x_{o,i}$ ($i = 1, 2, \dots$) secara berturut-turut dinotasikan sebagai data terkoreksi bias dan observasi dalam periode dasar tertentu, μ adalah rata-rata. Subskrip b , o , dan p secara berturut-turut adalah data dasar, observasi, dan proyeksi

3.3 Metode Distribusi Statistik

Koreksi bias dengan metode ini berdasarkan distribusi statistik dari data model yang ditransformasikan untuk menyesuaikan distribusi data dari observasi. Distribusi normal umum diasumsikan untuk data suhu udara [17], sedangkan distribusi gamma untuk data curah hujan [16]. Persamaan *Probability Density Function* (PDF) untuk sebaran normal (Persamaan 3) dan sebaran gamma (Persamaan 4) adalah sebagai berikut.

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-0.5(\frac{x-\mu}{\sigma})^2} \quad (3)$$

$$f(x, \alpha, \beta) = \frac{e^{-\frac{x}{\beta}} x^{\alpha-1}}{\Gamma(\alpha)\beta^\alpha} \quad (4)$$

dengan x adalah data curah hujan atau suhu udara, α adalah parameter skala, β adalah parameter bentuk, σ adalah standar deviasi, dan μ adalah rata-rata.

Di dalam publikasi Piani [16], persamaan yang digunakan untuk menghitung koreksi bias hujan harian dengan persamaan *Cumulative Density Function* (CDF) sebaran gamma (Persamaan 8), di mana $F(0)$ adalah peluang hari tidak hujan.

$$F(x, \alpha, \beta) = \begin{cases} \int_0^x \frac{e^{-\frac{x}{\beta}} x^{\alpha-1}}{\Gamma(\alpha)\beta^\alpha} dx + F(0) & , x > 0 \\ F(0) & , x = 0 \end{cases} \quad (5)$$

$$F(0) = \frac{\text{jumlah hari tidak hujan}}{\text{jumlah data}} \quad (6)$$

CDF ditentukan dari dua tipe data, yaitu observasi dan model. Bentuk *transfer function* untuk koreksi bias hujan harian diperoleh dari invers CDF model dan observasi. Persamaannya adalah sebagai berikut

$$F^{-1}(p, \alpha, \beta) = \begin{cases} \frac{1}{\beta} \left(p - F(0) \right)^{-1} \exp\left(-\frac{1}{\beta(p-F(0))}\right) & , p > F(0) \\ 0 & , p \leq F(0) \end{cases} \quad (7)$$

dengan nilai p adalah peluang ($0 < p \leq 1$). Dengan mengetahui invers CDF dari kedua data, hujan harian dari model terkoreksi dapat dihitung dengan persamaan regresi. Anda dapat menggunakan bentuk regresi seperti linier, kuadratik, atau eksponensial, tergantung dari hubungan antara invers CDF model dan observasi. Untuk menghindari nilai negatif, nilai intersep pada persamaan regresi diatur menjadi 0.

3 Koreksi Bias Statistik

Proses koreksi bias dihitung dari invers CDF. Pada Persamaan 4, nilai α dan β dapat diestimasi dengan metode Momen dengan persamaan sebagai berikut

$$\alpha = \frac{n\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (8)$$

$$\beta = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n\bar{x}} \quad (9)$$

Selain itu, *Maximum Likelihood Estimation* (MLE) dengan metode Greenwood dan Durand [8] juga dapat digunakan. Persamaannya sebagai berikut.

$$\alpha = \begin{cases} \frac{0.5000876+0.1648852y-0.0544276y^2}{y}, & \text{jika } 0 \leq y \leq 0.5772 \\ \frac{8.898919+9.05995y+0.9775373y^2}{17.79728y+11.968477y^2+y^3}, & \text{jika } 0.5772 < y \leq 17 \\ \frac{1}{y}, & \text{jika } y > 17 \end{cases} \quad (10)$$

$$y = \ln(\bar{x}) - \frac{1}{n} \sum_{i=1}^n \ln(x_i) \quad (11)$$

$$\beta = \frac{\bar{x}}{\alpha} \quad (12)$$

Pada modul ini, penentuan parameter pada distribusi gamma menggunakan metode MLE.

3.4 Sebelum Mulai

Variabel yang akan dikoreksi adalah curah hujan harian dari data Model CNRM-CM6-1 untuk *Baseline* dan *Projection* SSP245 terhadap data Climate Hazards group InfraRed Precipitation with Stations (CHIRPS). Model CNRM-CM6-1 dapat Anda unduh pada halaman Climate Data Store (CDS) dan Anda dapat memilih lokasi. Data contoh yang telah kami unduh sudah dipilih berdasarkan wilayah Indonesia. Format data berbentuk netCDF (.nc). Tabel 3.2 adalah informasi penggunaan data pada modul ini.

Tabel 3.2: Informasi data yang digunakan pada modul ini

Data	Tipe	Periode	Resolusi	Sumber
CHIRPS	Historis	1981 - 2020	0.05° x 0.05°	https://data.chc.ucsb.edu
CNRM-CM6-1	Baseline	1950 - 2014	1.41° x 1.41°	https://cds.climate.copernicus.eu
CNRM-CM6-1	Proyeksi SSP 245	2021 - 2100	1.41° x 1.41°	https://cds.climate.copernicus.eu

Satuan curah hujan pada data CHIRPS berbeda dengan CNRM-CM6-1. Satuan curah hujan CHIRPS adalah milimeter per hari, sedangkan CNRM-CM6-1 adalah kg/m²/s. Untuk mengubah satuan pada model CNRM-CM6-1, Anda dapat menggunakan Persamaan 13.

$$CH(mm/hari) = CH(kg/m^2/s) \times 86400 \quad (13)$$

3.5 Pengolahan Data

Koreksi bias curah hujan harian menggunakan data spasial. Teknik koreksi bias dengan Metode Delta dan Distribusi dijelaskan pada subbab ini menggunakan R, Python, dan Julia.

1. R

Sebelum menuju ke metode koreksi bias, data-data yang telah kami sediakan perlu dilakukan penyesuaian/modifikasi terlebih dahulu. Hal ini dikarenakan cakupan area dan resolusi spasial maupun periode dari masing-masing data berbeda. Untuk itu, kami akan melakukan beberapa tahapan berikut ini.

1. Impor terlebih dahulu paket yang digunakan. Jika belum ada, silahkan pasang terlebih dahulu dengan fungsi `install.packages()`.

```
library(dplyr)
library(ggplot2)
library(glue)
library(zoo)
library(ncdf4)
library(raster)
```

2. Bacalah ketiga data dengan fungsi `brick()`.

```
# Direktori/folder data
dpth <- 'data/'
# Membaca data
chp <- brick(glue('{dpth}/chirps-v2.0.1981-2020.days_p05.nc'))
bsl <-
  ↳ brick(glue('{dpth}/pr_day_CNRM-CM6-1_historical_r1i1p1f2_gr_19500101-20141231_v20180911.nc'))
ssp <-
  ↳ brick(glue('{dpth}/pr_day_CNRM-CM6-1_ssp245_r1i1p1f2_gr_20150101-21001231_v20190219.nc'))
```

3. Tentukan cakupan area serta waktu untuk koreksi. Misalnya, lokasi yang dipilih adalah Pulau Kalimantan dengan periode dari tahun 1981-2010 untuk baseline dan tahun 2021-2050 untuk skenario.

```
# Cakupan area
kalimantan <- extent(108, 122, -4, 7)
chp <- crop(chp, kalimantan)
bsl <- crop(bsl, kalimantan)
ssp <- crop(ssp, kalimantan)
# Periode koreksi
```

```

his_date <- seq(as.Date('1981-01-01'), as.Date('2010-12-31'), by = 'day')
prj_date <- seq(as.Date('2021-01-01'), as.Date('2050-12-31'), by = 'day')
# Subset data
chp <- raster::subset(chp, which(getZ(chp) %in% his_date))
bsl <- raster::subset(bsl, which(getZ(bsl) %in% his_date))
ssp <- raster::subset(ssp, which(getZ(ssp) %in% prj_date))

```

4. Samakan resolusi spasial CNRM-CM6-1 dengan CHIRPS dengan metode interpolasi. Secara default, metode interpolasi yang digunakan adalah *bilinear*. Anda dapat mengganti metodenya dengan menambahkan argumen `method` di dalam fungsi `resample()`. Metode yang tersedia adalah *nearest neighbor*.

```

bsl <- resample(bsl, chp)
ssp <- resample(ssp, chp)

```

5. Selanjutnya, konversikan satuan curah hujan CNRM-CM6-1 menjadi mm/hari dengan Persamaan 13. Setelah itu, nilai hasil konversi diubah menjadi 0 mm/hari jika nilainya kurang dari 1 mm/hari.

```

bsl <- bsl * 86400
ssp <- ssp * 86400

bsl[bsl < 1] <- 0
ssp[ssp < 1] <- 0

```

Metode Delta

1. Dengan menggunakan Persamaan 2, carilah rata-rata curah hujan dari data CNRM-CM6-1 untuk proyeksi dan baseline.

```

bsl_mean <- calc(bsl, mean)
ssp_mean <- calc(ssp, mean)

```

2. Carilah rasio antara rata-rata baseline dengan proyeksi.

```

ratio <- ssp_mean / bsl_mean

```

3. Terakhir, kalikan rasio dengan CHIRPS sehingga diperoleh curah hujan proyeksi terkoreksi.

```

ssp_corr <- ratio * chirps

```

Langkah 1-3 merupakan proses perhitungan koreksi bias Metode Delta. Selanjutnya, kita akan memvisualisasikan hasil koreksi bias tersebut. Sebagai contoh, kita akan memvisualisasikan rataan klimatologis

bulanan (2021-2050) hasil koreksi bias secara spasial, serta perbandingan data CNRM-CM6-1 SSP 245 sebelum dan sesudah terkoreksi.

1. Lakukan perhitungan akumulasi curah hujan bulanan

```
# Menambah informasi tanggal
ssp <- setZ(ssp, prj_date)
ssp_corr <- setZ(ssp_corr, prj_date)
# Akumulasi curah hujan bulanan
ssp_monthly <- zApply(ssp, as.yearmon, sum)
ssp_corr_monthly <- zApply(ssp_corr, as.yearmon, sum)
```

2. Lakukan perhitungan rata-rata klimatologi bulanan

```
# Rata-rata klimatologi bulanan
ssp_monthly_mean <- zApply(ssp_monthly, months, mean)
ssp_corr_monthly_mean <- zApply(ssp_corr_monthly, months, mean)
# Masking daratan
ssp_corr_monthly_mean[ssp_corr_monthly_mean < 0.1] <- NA
ssp_monthly_mean <- mask(ssp_monthly_mean, ssp_corr_monthly_mean)
```

3. Oleh karena informasi bulan di dalam variabel `ssp_monthly_mean` dan `ssp_corr_monthly_mean` tidak berurutan, gunakan perintah berikut ini.

```
ssp_monthly_mean <- brick(
  ssp_monthly_mean[[5]], ssp_monthly_mean[[4]],
  ssp_monthly_mean[[8]], ssp_monthly_mean[[1]],
  ssp_monthly_mean[[9]], ssp_monthly_mean[[7]],
  ssp_monthly_mean[[6]], ssp_monthly_mean[[2]],
  ssp_monthly_mean[[12]], ssp_monthly_mean[[11]],
  ssp_monthly_mean[[10]], ssp_monthly_mean[[3]])
)
ssp_corr_monthly_mean <- brick(
  ssp_corr_monthly_mean[[5]], ssp_corr_monthly_mean[[4]],
  ssp_corr_monthly_mean[[8]], ssp_corr_monthly_mean[[1]],
  ssp_corr_monthly_mean[[9]], ssp_corr_monthly_mean[[7]],
  ssp_corr_monthly_mean[[6]], ssp_corr_monthly_mean[[2]],
  ssp_corr_monthly_mean[[12]], ssp_corr_monthly_mean[[11]],
  ssp_corr_monthly_mean[[10]], ssp_corr_monthly_mean[[3]])
```

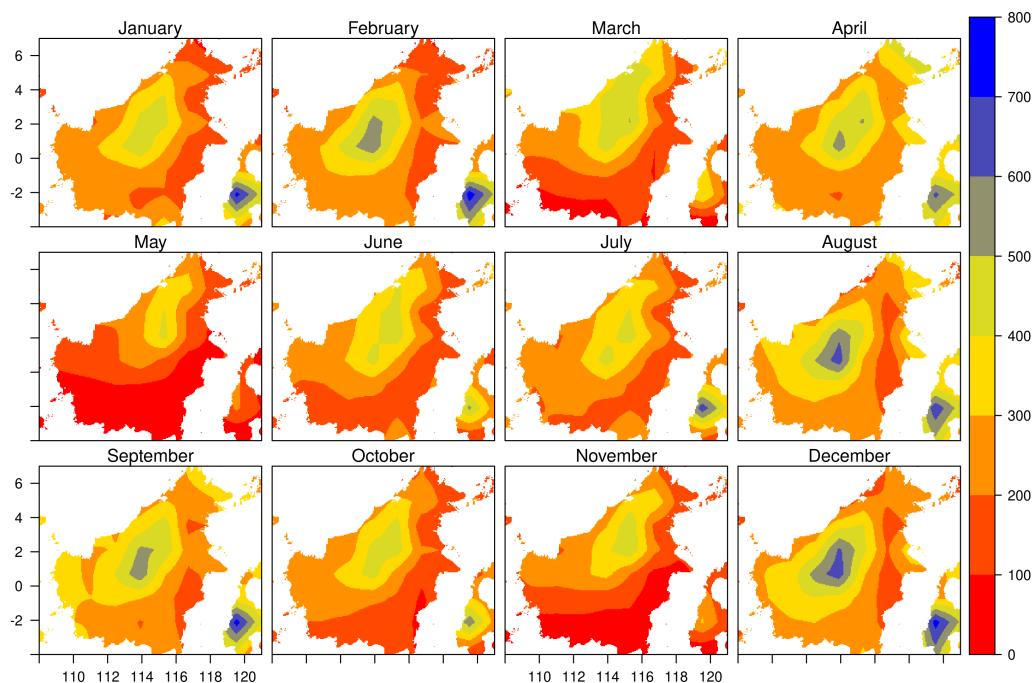
4. Visualisasikan hasil koreksi bias secara spasial. Anda dapat menggunakan perintah berikut ini.

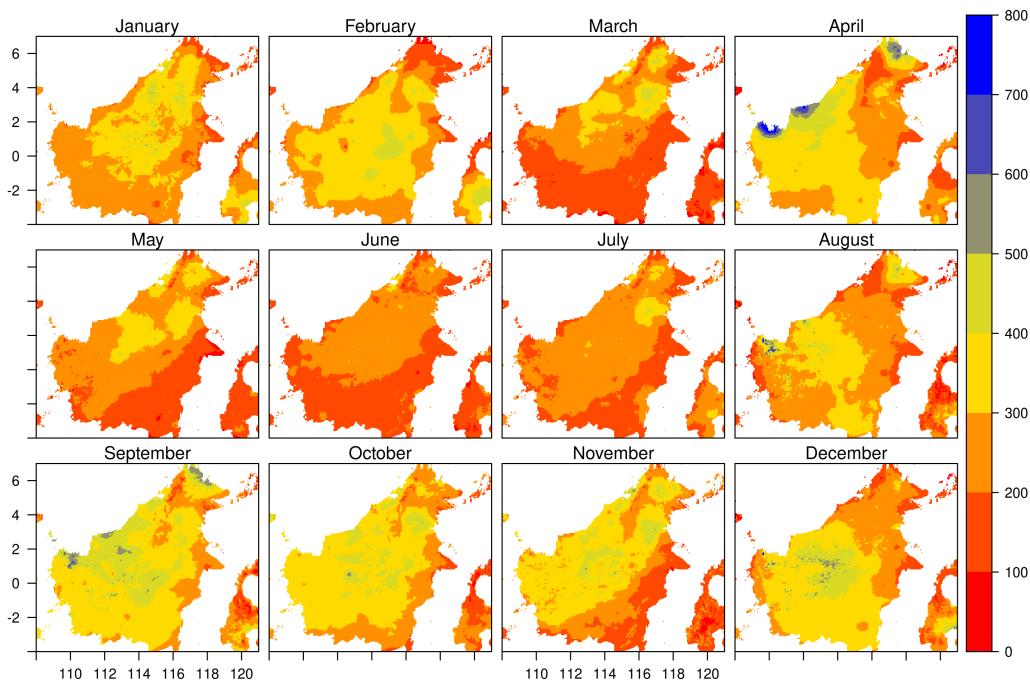
```
# Plot CNRM-CM6-1 setelah terkoreksi delta
levelplot(ssp_corr_monthly_mean,
  # Pengaturan skala
```

3 Koreksi Bias Statistik

```
at = seq(0, 800, 100),
# Pengaturan layout kolom dan baris
layout = c(4, 3),
# Pewarnaan
col.regions = colorRampPalette(c('red', 'yellow', 'blue')))

# Plot CNRM-CM6-1 sebelum terkoreksi delta
levelplot(ssp_monthly_mean,
at = seq(0, 800, 100),
layout = c(4, 3),
col.regions = colorRampPalette(c('red', 'yellow', 'blue')))
```





Metode Distribusi

1. aaaa
2. 2222
3. 2
4. 2
5. 2
6. 2
7. 2
8. 2
9. 2
- 10.

2. Python

Beberapa paket Python yang digunakan adalah `cartopy`, `matplotlib`, dan `xarray`. Jika Anda menggunakan `conda`, pasang terlebih dahulu paket-paket tersebut, ditambah dengan paket `jupyter`, `netcdf4` dan `dask` dengan membuat *environment* baru, misalnya `bias-correction`. Perintahnya diketik di terminal Bash (Linux/Mac) atau Anaconda3 Prompt (Windows) sebagai berikut.

```
conda create -n bias-correction -c conda-forge xarray matplotlib cartopy jupyter
↪ netcdf4 dask
```

3 Koreksi Bias Statistik

Untuk menuliskan skrip Python, ketik `jupyter notebook` pada terminal. Kemudian, muncul tampilan Jupyter Notebook pada Browser default Anda dan Anda dapat langsung membuat skrip dengan menekan tombol *New -> Python 3 (ipykernel)*.

1. Impor paket-paket yang telah terpasang

```
import numpy as np
import pandas as pd
import xarray as xr
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from scipy.stats import gamma, linregress
```

2. Impor ketiga data dengan menggunakan `xarray`.

```
pth = 'data'
chp = xr.open_dataset(f'{pth}/chirps-v2.0.1981-2020.days_p05.nc',
                     chunks={'time': 30})
bsl =
    xr.open_dataset(f'{pth}/pr_day_CNRM-CM6-1_historical_r1i1p1f2_gr_19500101-20141231_v20
ssp =
    xr.open_dataset(f'{pth}/pr_day_CNRM-CM6-1_ssp245_r1i1p1f2_gr_20150101-21001231_v201902
```

Argumen `chunks` digunakan untuk membagi data menjadi beberapa bagian yang lebih kecil. Hal ini dilakukan untuk mengurangi penggunaan memori. `chunks={'time': 30}` berarti data dibagi menjadi 30 bagian. Argumen ini dapat diaktifkan apabila Anda telah memasang `dask`.

3. Ubah nama dari koordinat `longitude` dan `latitude` menjadi `lon` dan `lat` pada data CHIRPS, juga variabel curah hujan `precip` menjadi `pr`. Langkah ini dilakukan agar nama koordinat serta variabel sama dengan data baseline maupun skenario. Selain itu, ini dapat memudahkan Anda dalam menghitung ketika melakukan koreksi bias.

```
chp = chp.rename({'longitude' : 'lon', 'latitude' : 'lat', 'precip' : 'pr'})
```

4. Tentukan cakupan area serta waktu untuk koreksi. Misalnya, lokasi yang dipilih adalah Pulau Kalimantan dengan periode dari tahun 1981-2010 untuk *Baseline* dan tahun 2031-2060 untuk skenario SSP 245.

```
# Cakupan area
lon1, lon2, lat1, lat2 = 108, 123, -4, 8

# Potong data sesuai cakupan area
chp = chp.sel(lon=slice(lon1, lon2), lat=slice(lat1, lat2))
bsl = bsl.sel(lon=slice(lon1, lon2), lat=slice(lat1, lat2))
ssp = ssp.sel(lon=slice(lon1, lon2), lat=slice(lat1, lat2))
```

```
# Periode waktu untuk koreksi
t1, t2 = '1981-01-01', '2010-12-31'
t3, t4 = '2031-01-01', '2060-12-31'

# Potong data sesuai periode waktu
chp = chp.sel(time=slice(t1, t2))
bsl = bsl.sel(time=slice(t1, t2))
ssp = ssp.sel(time=slice(t3, t4))
```

5. Samakan resolusi spasial CNRM-CM6-1 terhadap CHIRPS dengan metode interpolasi linier. Anda dapat memilih metode interpolasi lain, seperti `nearest`, `zero`, `slinear`, `quadratic`, `cubic`, atau `polynomial`. Kegunaan dari mencantumkan `kwargs` adalah untuk mengatasi nilai `Nan` yang muncul akibat interpolasi. Nilai `Nan` tersebut diubah menjadi nilai hasil ekstrapolasi.

```
# Interpolasi CNRM-CM6-1 ke resolusi CHIRPS
resample_bsl = bsl.interp(coords = {
    'lat' : chp['latitude'],
    'lon' : chp['longitude']
}, method = 'linear', kwargs={"fill_value": "extrapolate"})

resample_ssp = ssp.interp(coords = {
    'lat' : chp['latitude'],
    'lon' : chp['longitude']
}, method = 'linear', kwargs={"fill_value": "extrapolate"})
```

6. Selanjutnya, konversikan satuan curah hujan CNRM-CM6-1 menjadi mm/hari dengan Persamaan 13. Setelah itu, nilai hasil konversi diubah menjadi 0 mm/hari jika nilainya <1 mm/hari.

```
# Baseline
resample_bsl['pr'] = resample_bsl['pr'] * 86400
resample_bsl['pr'] = xr.where(resample_bsl['pr'] < 1, 0, resample_bsl['pr'])
# SSP 245
resample_ssp['pr'] = resample_ssp['pr'] * 86400
resample_ssp['pr'] = xr.where(resample_ssp['pr'] < 1, 0, resample_ssp['pr'])
```

Setelah mengerjakan langkah-langkah di atas, Anda dapat melanjutkan langkah-langkah koreksi bias pada subbab berikutnya.

Metode Delta

- Dengan menggunakan Persamaan 2, carilah rata-rata curah hujan dari data CNRM-CM6-1 untuk proyeksi dan baseline untuk semua waktu.

3 Koreksi Bias Statistik

```
# Baseline  
bsl_mean = resample_bsl.mean(dim='time')  
# SSP 245  
ssp_mean = resample_ssp.mean(dim='time')
```

2. Hitung rasio antara rata-rata baseline dengan proyeksi.

```
ratio = ssp_mean / bsl_mean
```

3. Terakhir, kalikan rasio dengan CHIRPS sehingga diperoleh curah hujan proyeksi terkoreksi.

```
# Kalikan rasio dengan data CHIRPS  
ssp_corrected = ratio * chp  
  
# Lakukan transpos dimensi agar dimensi waktu berada di posisi pertama  
ssp_corrected = ssp_corrected.transpose('time', 'lat', 'lon')  
  
# Mengubah waktu historis menjadi skenario SSP  
ssp_corrected = ssp_corrected.assign(time=ssp.time)
```

Langkah 1-3 merupakan proses perhitungan koreksi bias Metode Delta. Selanjutnya, kita akan memvisualisasikan hasil koreksi bias tersebut. Sebagai contoh, kita akan memvisualisasikan rataan klimatologi bulanan (2021-2050) hasil koreksi bias secara spasial, serta perbandingan data CNRM-CM6-1 SSP 245 sebelum dan sesudah terkoreksi.

1. Lakukan perhitungan akumulasi curah hujan bulanan

```
# Akumulasi curah hujan bulanan  
ssp_corrected_monthly = ssp_corrected.resample(time='1M').sum(skipna=False)
```

2. Lakukan perhitungan rata-rata klimatologi bulanan

```
# Rata-rata klimatologi bulanan  
ssp_corrected_monthly_mean =  
    ssp_corrected_monthly.groupby('time.month').mean(dim='time')
```

3. Visualisasikan hasil koreksi bias secara spasial

```
# Create a subplot of spatial plots for each month  
fig, axes = plt.subplots(3, 4, figsize = (15, 10), subplot_kw =  
    {'projection': ccrs.PlateCarree()})  
for month, ax in zip(range(1, 13), axes.ravel()):  
    # Memilih data untuk setiap bulan  
    month_data = ssp_corrected_monthly_mean.sel(month=month)  
    month_data = month_data['pr']
```

```
# Membuat plot
month_data.plot.imshow(ax=ax, cmap='jet', add_colorbar=False,
↪ transform=ccrs.PlateCarree())

# Menambahkan judul grafik dan garis pantai
ax.set_title(f'Month: {month}')
ax.coastlines(linewidth=0.5)

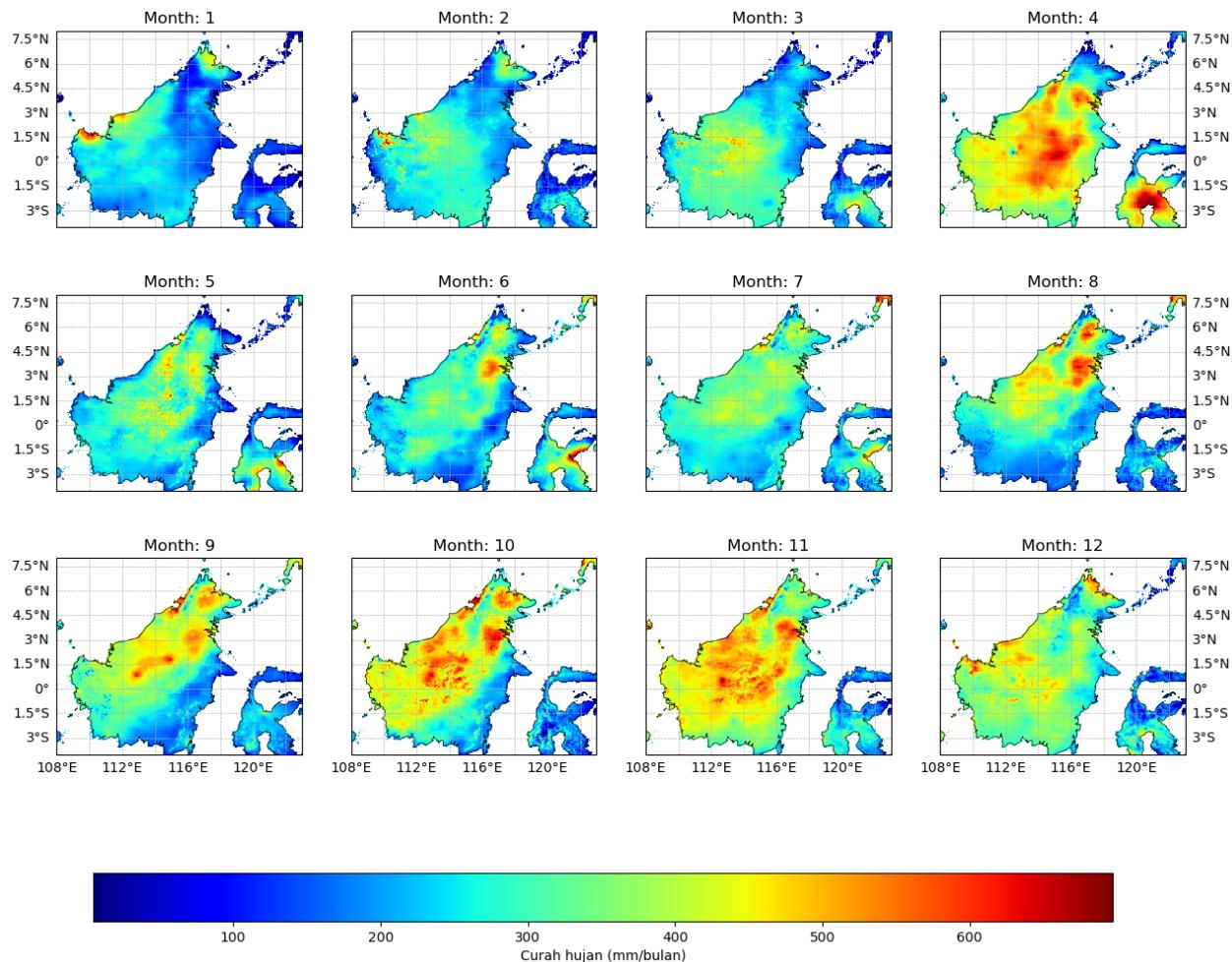
# Menambahkan garis lintang dan bujur
gl = ax.gridlines(draw_labels=True, linestyle='--', linewidth=0.5)

# Menghilangkan label untuk beberapa plot agar tidak terlalu padat
if month not in [1, 5, 9]:
    gl.left_labels = False
if month not in [9, 10, 11, 12]:
    gl.bottom_labels = False
if month not in [4, 8, 12]:
    gl.right_labels = False
gl.top_labels = False

# Menambahkan colorbar
cbar_ax = fig.add_axes([0.15, -0.05, 0.7, 0.05])
fig.colorbar(ax.images[0], cax=cbar_ax, orientation='horizontal',
↪ label='Curah hujan (mm/bulan)')

plt.show()
```

3 Koreksi Bias Statistik



4. Selanjutnya, buat grafik garis untuk membandingkan data CNRM-CM6-1 SSP 245 sebelum dan sesudah terkoreksi. Sebagai contoh, lokasi Kota Banjarmasin digunakan sebagai perbandingan.

```
# Membuat plot
fig, ax = plt.subplots(figsize=(10, 5))

# Memilih data untuk setiap bulan
lat, lon = -3.322423232458966, 114.59597874153853
ssp_corrected.pr.sel(lat=lat, lon=lon, method='nearest').plot(ax=ax,
    label='Corrected')
ssp_mean.pr.sel(lat=lat, lon=lon, method='nearest').plot(ax=ax, label='Raw')

# Menambahkan judul grafik
ax.set_title('Perbandingan data CNRM-CM6-1 SSP 245 sebelum dan sesudah
    terkoreksi')

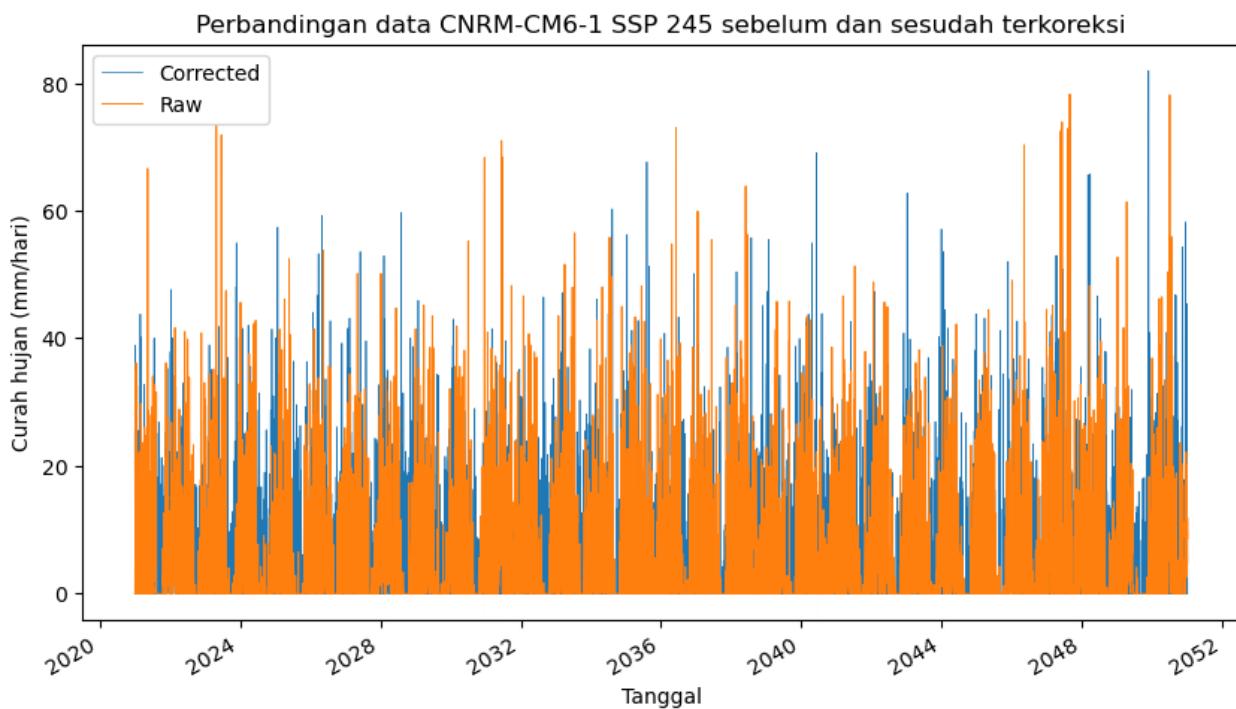
# Menambahkan label sumbu x dan y
ax.set_xlabel('Tanggal')
```

```

ax.set_ylabel('Curah hujan (mm/hari)')

# Menambahkan legenda
ax.legend()
plt.show()

```



Metode Distribusi Statistik

Untuk metode ini, kita akan melakukan koreksi curah hujan *Baseline* pada model CNRM-CM6-1 terhadap CHIRPS. Secara umum, pola distribusi curah hujan harian adalah gamma. Dengan menggunakan Persamaan 4, kita perlu menghitung parameter skala (α) dan bentuk (β) dengan metode MLE (Persamaan 10, Persamaan 11, Persamaan 12).

- Pertama, hitunglah peluang kejadian tidak hujan untuk setiap grid (Persamaan 6) pada kedua data.

```

resample_bsl_prob = (resample_bsl.pr < 1).mean(dim='time')

non_nan_mask = ~np.isnan(chp.pr) # Menentukan grid yang tidak NaN
chp_prob = (chp.pr < 1) \
    .where(non_nan_mask) \
    .mean(dim='time') # Hanya menghitung peluang pada grid yang tidak NaN

```

- Hitunglah nilai y ([eq-11]) untuk menentukan kondisi awal α untuk setiap grid. Untuk menghitung \bar{x} , curah hujan bernilai <1 mm tidak termasuk dalam perhitungan.

3 Koreksi Bias Statistik

```
# Menghitung rataan curah hujan >= 1 mm
resample_bsl_mean = resample_bsl.pr.where(resample_bsl.pr >=
    1).mean(dim='time')
chp_mean = chp.pr.where(chp.pr >= 1).mean(dim='time')
# Menghitung nilai ln(x_bar)
resample_bsl_mean_ln = np.log(resample_bsl_mean)
chp_mean_ln = np.log(chp_mean)
# Mencari nilai ln(x) pada setiap grid dan waktu pada curah hujan >= 1 mm
resample_bsl_ln = np.log(resample_bsl.pr.where(resample_bsl.pr >= 1))
chp_ln = np.log(chp.pr.where(chp.pr >= 1))
# Menghitung nilai y
resample_bsl_y = resample_bsl_mean_ln - resample_bsl_ln.mean(dim='time')
chp_y = chp_mean_ln - chp_ln.mean(dim='time')
```

3. Hitunglah nilai α untuk setiap grid berdasarkan kondisi dari nilai y .

```
def alpha(y):
    if y <= 0.5772:
        return (0.5000876 + 0.1648852*y - 0.0544276*y**2) / y
    elif y <= 17:
        return (8.898919 + 9.05995*y + 0.9775373*y**2) / \
            (17.79728*y + 11.968477*y**2 + y**3)
    else:
        return 1 / y
alpha_ufunc = np.vectorize(alpha)

resample_bsl_alpha = xr.apply_ufunc(alpha_ufunc, resample_bsl_y.compute())
chp_alpha = xr.apply_ufunc(alpha_ufunc, chp_y.compute())
```

4. Hitunglah nilai β untuk setiap grid.

```
resample_bsl_beta = resample_bsl_alpha / resample_bsl_mean
chp_beta = chp_alpha / chp_mean
```

5. Sebagai ilustrasi untuk membandingkan distribusi antar kedua data, buatlah nilai CDF untuk setiap grid berdasarkan parameter skala dan bentuk.

```
# Fungsi untuk menghitung CDF
def calculate_cdf(shape, scale, x, prob):
    if x < 1:
        return prob
    else:
        cdf_values = gamma.cdf(x, a=shape, scale=scale) * (1 - prob) + prob
        return cdf_values
ufunc_calculate_cdf = np.vectorize(calculate_cdf)
```

```

# Membuat array dari 0.1 hingga 50 dengan 100 data
rain_min, rain_max, counts = 0.1, 50, 100
x_values = np.linspace(rain_min, rain_max, counts)
# Informasi lokasi sebagai informasi dari x_values
lat = resample_bsl['lat']
lon = resample_bsl['lon']
# Create a 3D x_values array matching the shape of shape_variable and
# scale_variable
x_values_3d = np.tile(x_values, (len(chp['lat']), len(chp['lon']), 1))
# Create xarray DataArray for x_values_3d
x_values_3d = xr.DataArray(
    x_values_3d,
    dims=['lat', 'lon', 'time'],
    coords={'lat': lat, 'lon': lon, 'time': x_values}
)

# Transpose with time as first dimension
x_values_3d = x_values_3d.transpose('time', 'lat', 'lon')

# Baseline CNRM-CM6-1
resample_bsl_cdf = xr.apply_ufunc(
    ufunc_calculate_cdf,
    resample_bsl_beta,
    resample_bsl_alpha,
    x_values_3d,
    resample_bsl_prob,
    input_core_dims=
        [['lat', 'lon'],           # Dimensi resample_bsl_beta
         ['lat', 'lon'],           # Dimensi resample_bsl_alpha
         ['time', 'lat', 'lon'],   # Dimensi x_values_3d
         ['lat', 'lon']],          # Dimensi resample_bsl_prob
    dask='parallelized',
    output_core_dims=[['time', 'lat', 'lon']],
    output_sizes={'time': len(x_values)},
    output_dtypes=[float],
    vectorize=True
)

# Historis CHIRPS
chp_cdf = xr.apply_ufunc(
    ufunc_calculate_cdf,
    chp_beta,
    chp_alpha,
    x_values_3d,
    chp_prob,

```

```

input_core_dims=
    [['lat', 'lon'],           # Dimensi chp_beta
     ['lat', 'lon'],           # Dimensi chp_alpha
     ['time', 'lat', 'lon'],   # Dimensi x_values_3d
     ['lat', 'lon']],          # Dimensi chp_prob
dask='parallelized',
output_core_dims=[['time', 'lat', 'lon']],
output_sizes={'time': len(x_values)},
output_dtypes=[float],
vectorize=True
)

resample_bsl_cdf = resample_bsl_cdf.compute()
chp_cdf = chp_cdf.compute()

```

6. Kemudian, buatlah grafik CDF antara CHIRPS dan CNRM-CM6-1 Baseline pada grid tertentu. Sebagai contoh lokasi yang diambil adalah Kota Banjarmasin.

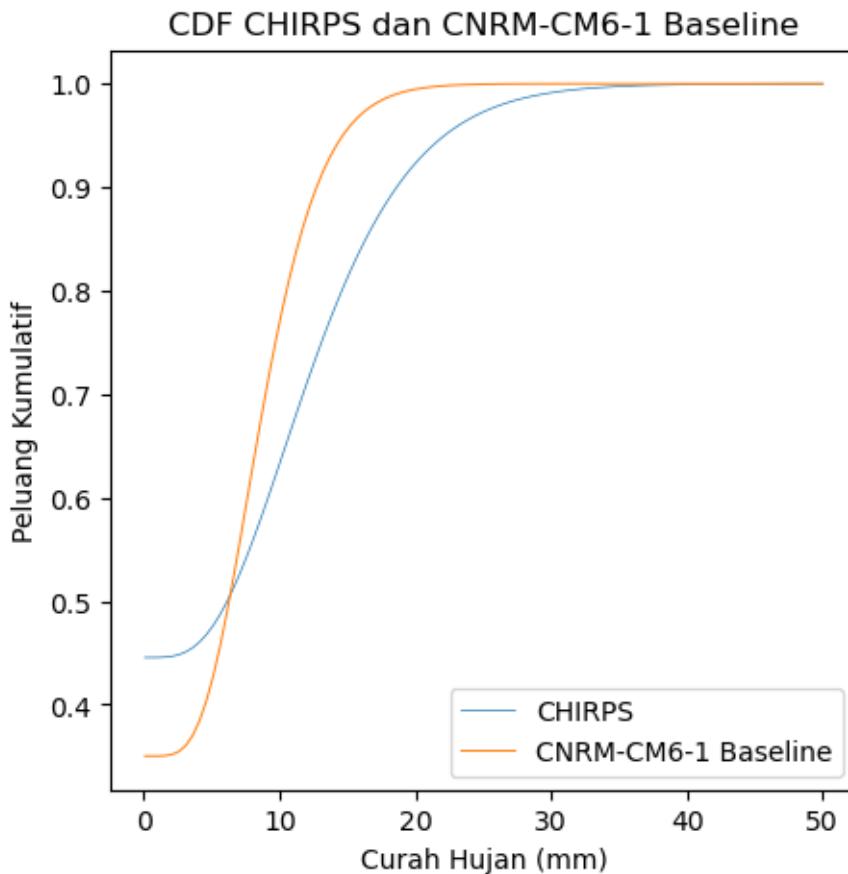
```

# Plot CDF
fig, ax = plt.subplots(figsize=(5, 5))

lat, lon = -3.3224, 114.5960
chp_cdf.sel(lat=lat, lon=lon, method='nearest').plot(ax=ax, label='CHIRPS',
    linewidth=0.7)
resample_bsl_cdf.sel(lat=lat, lon=lon, method='nearest').plot(ax=ax,
    label='CNRM-CM6-1 Baseline', linewidth=0.7)

ax.legend()
ax.set_title('CDF CHIRPS dan CNRM-CM6-1 Baseline')
ax.set_xlabel('Curah Hujan (mm)')
ax.set_ylabel('Peluang Kumulatif')

```



Jika Anda perhatikan, terdapat perbedaan antara CDF CHIRPS dengan CNRM-CM6-1 pada grid contoh.

7. Untuk menyamakan grafik CDF CNRM-CM6-1 terhadap CHIRPS, Bentuk *transfer function* menggunakan persamaan regresi. Langkah selanjutnya adalah mencari invers dari CDF CHIRPS dan CNRM-CM6-1.

```
# Fungsi untuk menghitung invers CDF
def calculate_invcdf(shape, scale, p, prob):
    if p < prob:
        return 0.0
    else:
        pr = (p - prob) / (1 - prob)
        invcdf_values = gamma.ppf(pr, a=shape, scale=scale)
        return invcdf_values
ufunc_calculate_invcdf = np.vectorize(calculate_invcdf)

p_min, p_max, counts = 0.00, 0.99, 100
# Informasi lokasi sebagai informasi untuk p_values
lat = resample_bsl['lat']
```

```

lon = resample_bsl['lon']
# Membuat array peluang dari 0 hingga 0.99 dengan 100 data
p_values = np.linspace(p_min, p_max, counts)
# Create a 3D x_values array matching the shape of shape_variable and
# scale_variable
p_values_3d = np.tile(p_values, (len(lat), len(lon), 1))
# Create xarray DataArray for x_values_3d
p_values_3d = xr.DataArray(
    p_values_3d,
    dims=['lat', 'lon', 'time'],
    coords={'lat': lat, 'lon': lon, 'time': p_values}
)

# Transpose with time as first dimension
p_values_3d = p_values_3d.transpose('time', 'lat', 'lon')

# Baseline CNRM-CM6-1
resample_bsl_invcdf = xr.apply_ufunc(
    ufunc_calculate_invcdf,
    resample_bsl_beta,
    resample_bsl_alpha,
    p_values_3d,
    resample_bsl_prob,
    input_core_dims=
        [['lat', 'lon'],
         ['lat', 'lon'],
         ['time', 'lat', 'lon'],
         ['lat', 'lon']],
    dask='parallelized',
    output_core_dims=[['time', 'lat', 'lon']],
    output_sizes={'time': len(p_values)},
    output_dtypes=[float],
    vectorize=True
)

# Historis CHIRPS
chp_invcdf = xr.apply_ufunc(
    ufunc_calculate_invcdf,
    chp_beta,
    chp_alpha,
    p_values_3d,
    chp_prob,
    input_core_dims=
        [['lat', 'lon'],
         ['lat', 'lon'],
         ['time', 'lat', 'lon'],

```

```

        ['lat', 'lon']],
dask='parallelized',
output_core_dims=[['time', 'lat', 'lon']],
output_sizes={'time': len(p_values)},
output_dtypes=[float],
vectorize=True
)

resample_bsl_invcdf = resample_bsl_invcdf.compute()
chp_invcdf = chp_invcdf.compute()

```

8. Setelah selesai menghitung nilai invers CDF dari distribusi gamma, Anda perlu melakukan regresi invers CDF antara CNRM-CM6-1 dengan CHIRPS. Untuk mengetahui jenis regresi yang ingin dipilih, Anda dapat membuat plot garis pada grid tertentu.

```

# Plot Invers CDF CHIRPS dan CNRM-CM6-1 Baseline, serta cantumkan persamaan
# regresi
# Perform linear regression
fit = linregress(bsl_sel, chp_sel)
bsl_fit = fit.slope * bsl_sel + fit.intercept

# Plot invers CDF
fig, ax = plt.subplots(figsize=(5, 5))

# Memilih grid
lat, lon = -3.3224, 114.5960
chp_sel = chp_invcdf.sel(lat=lat, lon=lon, method='nearest').values
bsl_sel = resample_bsl_invcdf.sel(lat=lat, lon=lon, method='nearest').values

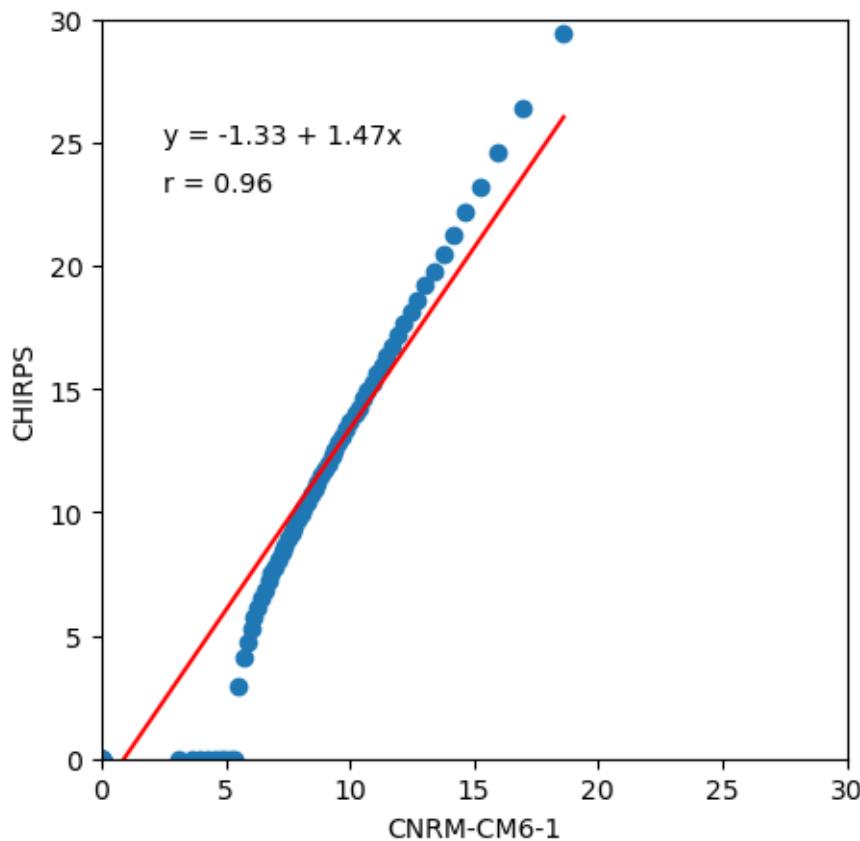
# Plot scatter
ax.scatter(x = bsl_sel, y = chp_sel)

# Plot garis linier
ax.plot(bsl_sel, chp_fit, color = 'red')

# Menambahkan informasi persamaan regresi serta korelasi
regression_equation = f'y = {intercept:.2f} + {slope:.2f}x'
ax.text(2.5, 25, regression_equation)
ax.text(2.5, 23, f'r = {r_value:.2f}')

ax.set_xlim([0, 30]); ax.set_ylim([0, 30])
ax.set_xlabel('CNRM-CM6-1')
ax.set_ylabel('CHIRPS')

```



Berdasarkan grafik di atas, jenis regresi yang dipilih adalah regresi linier sederhana.

9. Langkah ke-8 berlaku untuk satu grid saja. Untuk mencari nilai slope pada regresi linier sederhana untuk setiap grid, gunakan perintah berikut.

```
nlat = chp_invcdf.lat.values
nlon = chp_invcdf.lon.values

# Define a function to calculate the slope using linregress
def calculate_slope_invcdf(x, y):
    # Menyediakan array kosong sebanyak nlat x nlon
    slopes = np.empty((len(nlat), len(nlon)))
    # Meletakkan nilai slope untuk setiap grid ke array kosong
    for lat_idx, lat in enumerate(nlat):
        for lon_idx, lon in enumerate(nlon):
            fit = linregress(x[:, lat_idx, lon_idx], y[:, lat_idx, lon_idx])
            slopes[lat_idx, lon_idx] = fit.slope
    return slopes

# Calculate the slope for each latitude and longitude grid point
slope_invcdf = xr.apply_ufunc(
    calculate_slope_invcdf,
```

```

    resample_bsl_invcdf,
    chp_invcdf,
    input_core_dims=[['time', 'lat', 'lon'], ['time', 'lat', 'lon']],
    dask='parallelized',
    output_core_dims=[['lat', 'lon']],
    output_sizes={'lat': len(nlat), 'lon' : len(nlon)},
    output_dtypes=[float],
    vectorize=True
)

```

10. Setelah mendapatkan nilai `slope` setiap grid, kalikan Baseline CNRM-CM6-1 terhadap nilai `slope` sehingga diperoleh Baseline CNRM-CM6-1 terkoreksi. Ini adalah langkah terakhir. `python resample_bsl_corrected = resample_bsl.pr * slope_invcdf`
11. Anda dapat membandingkan data terkoreksi Baseline CNRM-CM6-1 terhadap data CHIRPS, serta data Baseline CNRM-CM6-1 sebelum terkoreksi.

```

# Membuat fungsi plot klimatologi
def create_clim_plot(data, label):
    # Create a subplot of spatial plots for each month
    fig, axes = plt.subplots(3, 4, figsize=(15, 10), subplot_kw={'projection':
        ccrs.PlateCarree()})
    for month, ax in zip(range(1, 13), axes.ravel()):
        month_data = data.sel(month=month)
        month_data.plot.imshow(ax=ax, cmap='jet', add_colorbar=False,
        transform=ccrs.PlateCarree(), vmin = 0, vmax = 800)

        # Menambahkan judul grafik dan garis pantai
        ax.set_title(f'Month: {month}')
        ax.coastlines(linewidth=0.5)

        # Menambahkan garis lintang dan bujur
        gl = ax.gridlines(draw_labels=True, linestyle='--', linewidth=0.5)

        # Menghilangkan label untuk beberapa plot agar tidak terlalu padat
        if month in [2, 3, 4, 6, 7, 8, 10, 11, 12]:
            gl.left_labels = False
        if month in [1, 2, 3, 4, 5, 6, 7, 8]:
            gl.bottom_labels = False
        if month not in [4, 8, 12]:
            gl.right_labels = False
        gl.top_labels = False

    # Menambahkan colorbar
    cbar_ax = fig.add_axes([0.11, 0.05, 0.8, 0.02]) # Left, bottom, width,
    ↵ height.

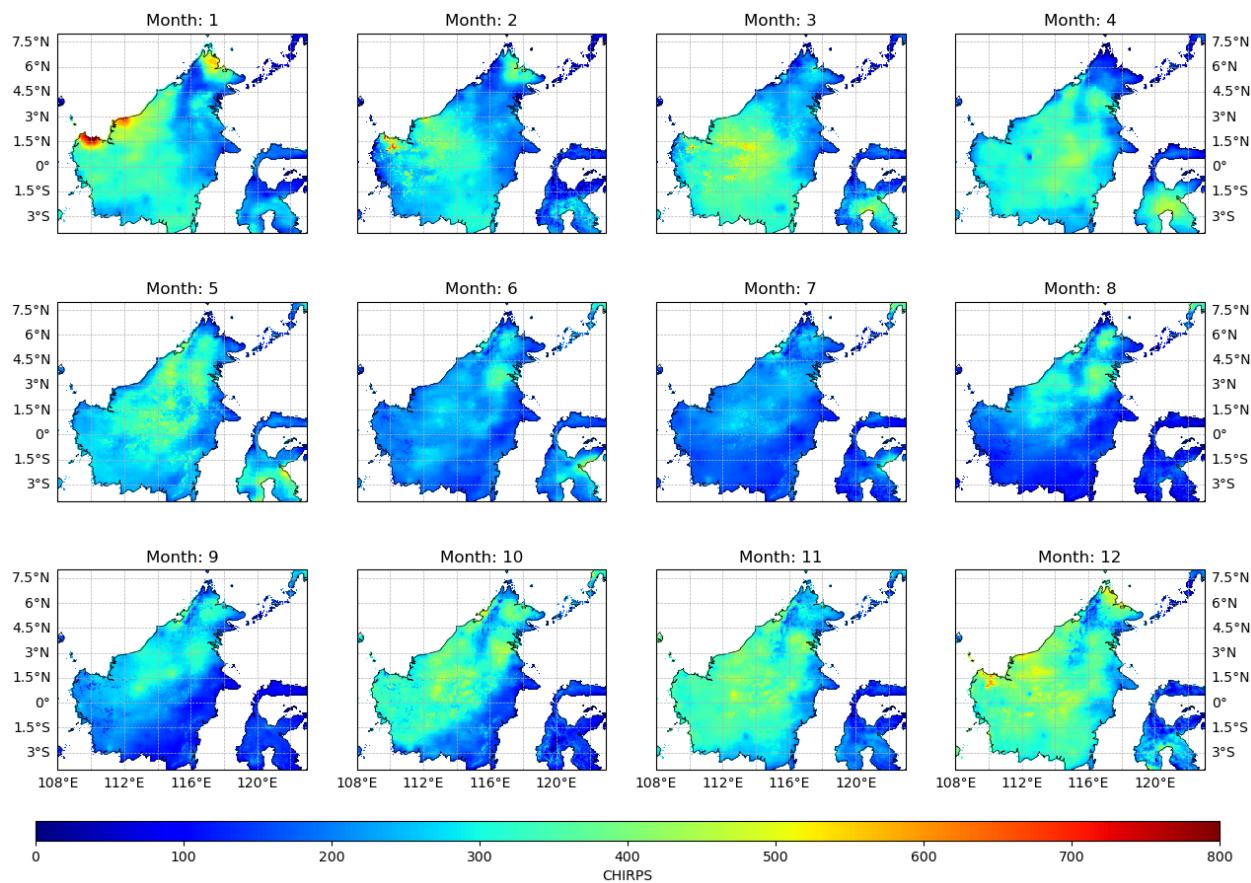
```

3 Koreksi Bias Statistik

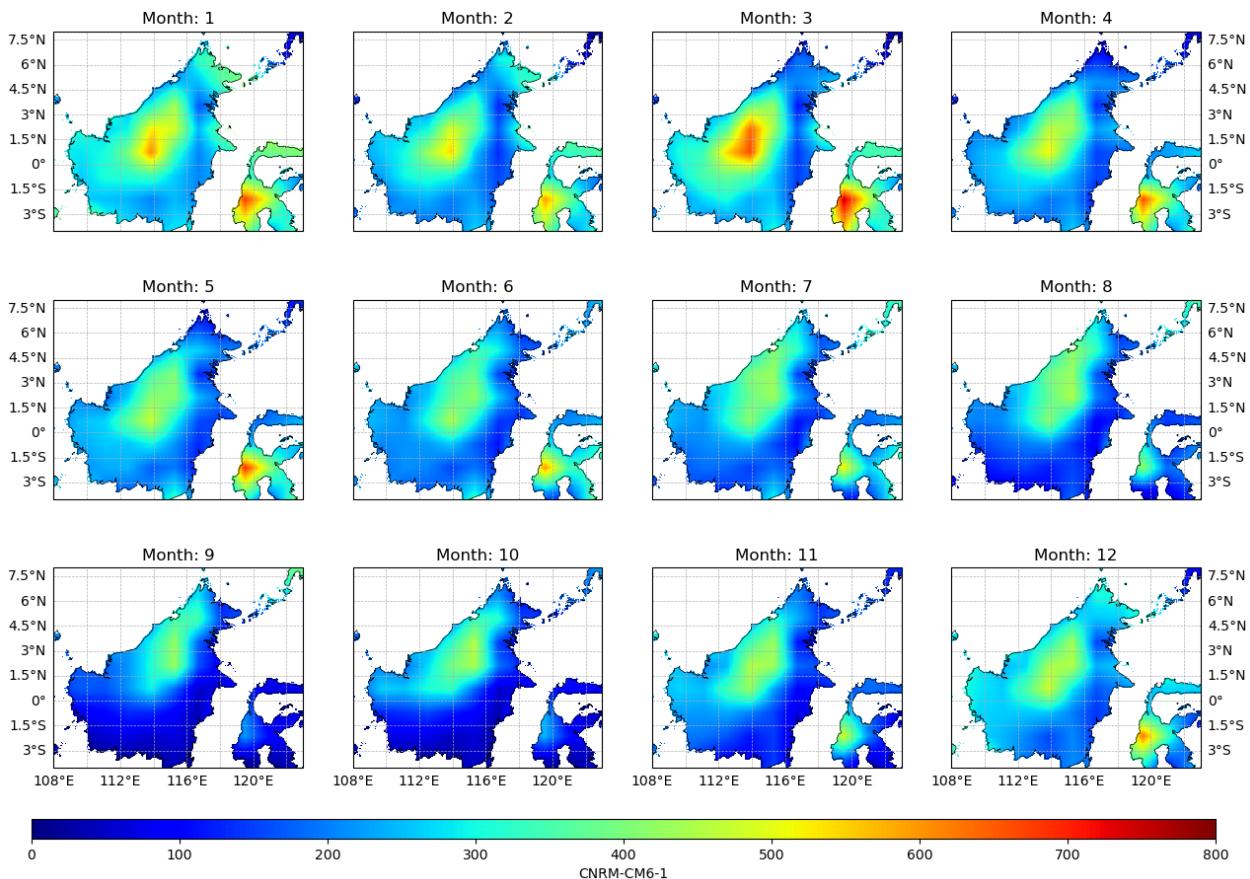
```
fig.colorbar(ax.images[0], cax=cbar_ax, orientation='horizontal',
             label=label)

plt.show()

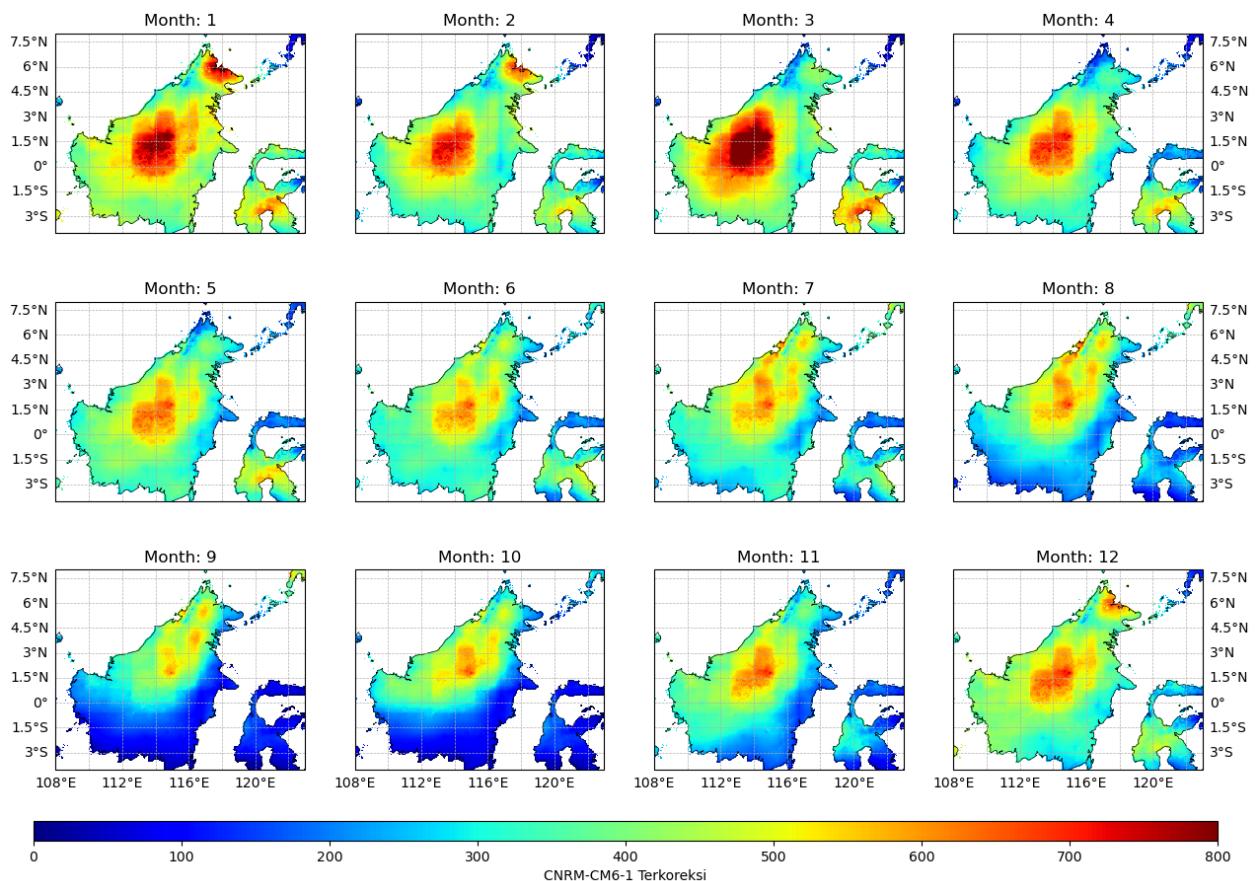
# Plot 3 grafik: CNRM-CM6-1 sebelum dan sesudah terkoreksi dan CHIRPS
create_clim_plot(chp_monthly_mean, 'CHIRPS')
create_clim_plot(resample_bsl_monthly_mean, 'CNRM-CM6-1')
create_clim_plot(resample_bsl_corrected_monthly_mean, 'CNRM-CM6-1
                 Terkoreksi')
```



3.5 Pengolahan Data



3 Koreksi Bias Statistik



Referensi

- [1] Nans Addor and Erich M Fischer. “The influence of natural variability and interpolation errors on bias characterization in RCM simulations”. In: *Journal of Geophysical Research: Atmospheres* 120.19 (2015), pp. 10–180.
- [2] Rosalind A Ball, Larry C Purcell, and Sean K Carey. “Evaluation of solar radiation prediction models in North America”. In: *Agronomy Journal* 96.2 (2004), pp. 391–397.
- [3] Jingyi Bao, Fotini Katopodes Chow, and Katherine A Lundquist. “Large-eddy simulation over complex terrain using an improved immersed boundary method in the Weather Research and Forecasting Model”. In: *Monthly Weather Review* 146.9 (2018), pp. 2781–2797.
- [4] Uwe Ehret et al. “HESS Opinions” Should we apply bias correction to global and regional climate model data?”” In: *Hydrology & Earth System Sciences Discussions* 9.4 (2012).
- [5] Michael Glotter et al. “Evaluating the utility of dynamical downscaling in agricultural impacts projections”. In: *Proceedings of the National Academy of Sciences* 111.24 (2014), pp. 8776–8781.
- [6] Aneesh Goly, Ramesh SV Teegavarapu, and Arpita Mondal. “Development and evaluation of statistical downscaling models for monthly precipitation”. In: *Earth Interactions* 18.18 (2014), pp. 1–28.
- [7] L Phil Graham, Johan Andréasson, and Bengt Carlsson. “Assessing climate change impacts on hydrology from an ensemble of regional climate models, model scales and linking methods—a case study on the Lule River basin”. In: *Climatic Change* 81.1 (2007), pp. 293–307.
- [8] J Arthur Greenwood and David Durand. “Aids for fitting the gamma distribution by maximum likelihood”. In: *Technometrics* 2.1 (1960), pp. 55–65.
- [9] LA Hunt, L Kuchar, and CJ Swanton. “Estimation of solar radiation for use in crop modelling”. In: *Agricultural and Forest Meteorology* 91.3-4 (1998), pp. 293–300.
- [10] R Huth and J Kysely. “Constructing site-specific climate change scenarios on a monthly scale using statistical downscaling”. In: *Theoretical and Applied Climatology* 66.1 (2000), pp. 13–27.
- [11] JW Kim et al. “The statistical problem of climate inversion: Determination of the relationship between local and large-scale climate”. In: *Monthly weather review* 112.10 (1984), pp. 2069–2077.
- [12] W Ladwig. *wrf-python Version 1.3.4*. 2017. DOI: <https://doi.org/10.5065/D6W094P1>.
- [13] Delei Li et al. “Statistical Bias Correction for Simulated Wind Speeds Over CORDEX-East Asia”. In: *Earth and Space Science* 6.2 (2019), pp. 200–211. eprint: <https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2018EA000493>. URL: <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018EA000493>.
- [14] Douglas Maraun. “Bias correcting climate change simulations—a critical review”. In: *Current Climate Change Reports* 2.4 (2016), pp. 211–220.
- [15] HA Pahlavan et al. “Improvement of multiple linear regression method for statistical downscaling of monthly precipitation”. In: *International journal of environmental science and technology* 15.9 (2018), pp. 1897–1912.

- [16] C Piani, JO Haerter, and E Coppola. "Statistical bias correction for daily precipitation in regional climate models over Europe". In: *Theoretical and Applied Climatology* 99.1-2 (2010), pp. 187–192.
- [17] C Piani et al. "Statistical bias correction of global simulated daily precipitation and temperature for the application of hydrological models". In: *Journal of Hydrology* 395.3-4 (2010), pp. 199–215.
- [18] Jordan G. Powers et al. "The Weather Research and Forecasting Model: Overview, System Efforts, and Future Directions". In: *Bulletin of the American Meteorological Society* 98.8 (2017), pp. 1717–1737. DOI: [10.1175/BAMS-D-15-00308.1](https://doi.org/10.1175/BAMS-D-15-00308.1). eprint: <https://doi.org/10.1175/BAMS-D-15-00308.1>. URL: <https://doi.org/10.1175/BAMS-D-15-00308.1>.
- [19] Clarence W Richardson. "Stochastic simulation of daily precipitation, temperature, and solar radiation". In: *Water resources research* 17.1 (1981), pp. 182–190.
- [20] DA Sachindra et al. "Statistical downscaling of general circulation model outputs to precipitation—part 1: calibration and validation". In: *International Journal of Climatology* 34.11 (2014), pp. 3264–3281.
- [21] William C Skamarock et al. *A description of the advanced research WRF version 4*. Tech. rep. National Center For Atmospheric Research Boulder Co Mesoscale and Microscale ..., 2019.
- [22] Marek Smid and Ana Cristina Costa. "Climate projections and downscaling techniques: a discussion for impact studies in urban systems". In: *International Journal of Urban Sciences* 22.3 (2018), pp. 277–307. DOI: [10.1080/12265934.2017.1409132](https://doi.org/10.1080/12265934.2017.1409132). eprint: <https://doi.org/10.1080/12265934.2017.1409132>. URL: <https://doi.org/10.1080/12265934.2017.1409132>.
- [23] T Sonkaew et al. "Finding the Optimum Microphysics and Convective Parameterization Schemes for the WRF Model for LPRU, Thailand". In: () .
- [24] FC Sperna Weiland et al. "The ability of a GCM-forced hydrological model to reproduce global discharge variability". In: *Hydrology and Earth System Sciences* 14 (2010), pp. 1595–1621.
- [25] CJT Spitters, HAJM Toussaint, and J Goudriaan. "Separating the diffuse and direct component of global radiation and its implications for modeling canopy photosynthesis Part I. Components of incoming radiation". In: *Agricultural and Forest Meteorology* 38.1-3 (1986), pp. 217–229.
- [26] Ronald B Stull. *Practical meteorology: an algebra-based survey of atmospheric science*. University of British Columbia, 2015.
- [27] Charles Talbot, Elie Bou-Zeid, and Jim Smith. "Nested mesoscale large-eddy simulations with WRF: Performance in real test cases". In: *Journal of Hydrometeorology* 13.5 (2012), pp. 1421–1441.
- [28] Kevin E Trenberth, John T Fasullo, and Jeffrey Kiehl. "Earth's global energy budget". In: *Bulletin of the american meteorological society* 90.3 (2009), pp. 311–324.
- [29] Satoshi Watanabe et al. "Intercomparison of bias-correction methods for monthly temperature and precipitation simulated by multiple climate models". In: *Journal of Geophysical Research: Atmospheres* 117.D23 (2012).
- [30] Robert L Wilby and Thomas ML Wigley. "Downscaling general circulation model output: a review of methods and limitations". In: *Progress in physical geography* 21.4 (1997), pp. 530–548.
- [31] Hongxiong Xu et al. "Performance of WRF large eddy simulations in modeling the convective boundary layer over the Taklimakan desert, China". In: *Journal of Meteorological Research* 32.6 (2018), pp. 1011–1025.