Praktikumsaufgaben

Release 0.1

Nov 30, 2019

Aufgaben

CHAPTER 1

Intro

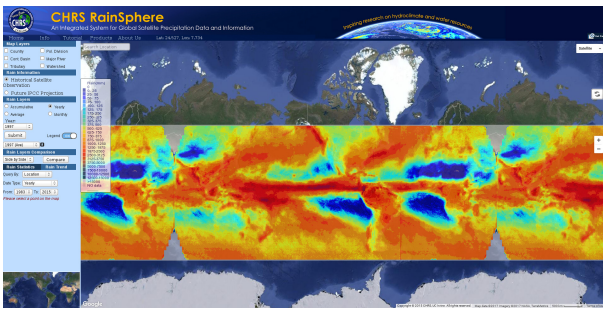
Hier befindet sich eine kleine Sammlung von Aufgaben und Ideen, mit denen sich Praktikanten bei mir am TROPOS beschäftigen könnten. Für Schülerpraktikanten sollte man vielleicht nur den leichteren Teil der Aufgaben auswählen.

CHAPTER 2

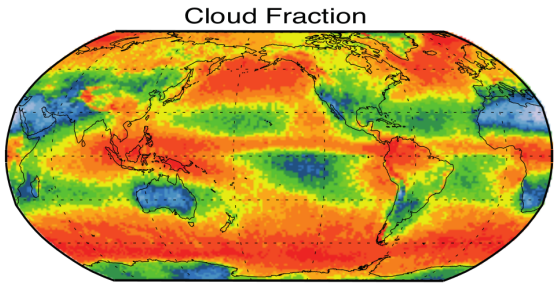
Übersicht

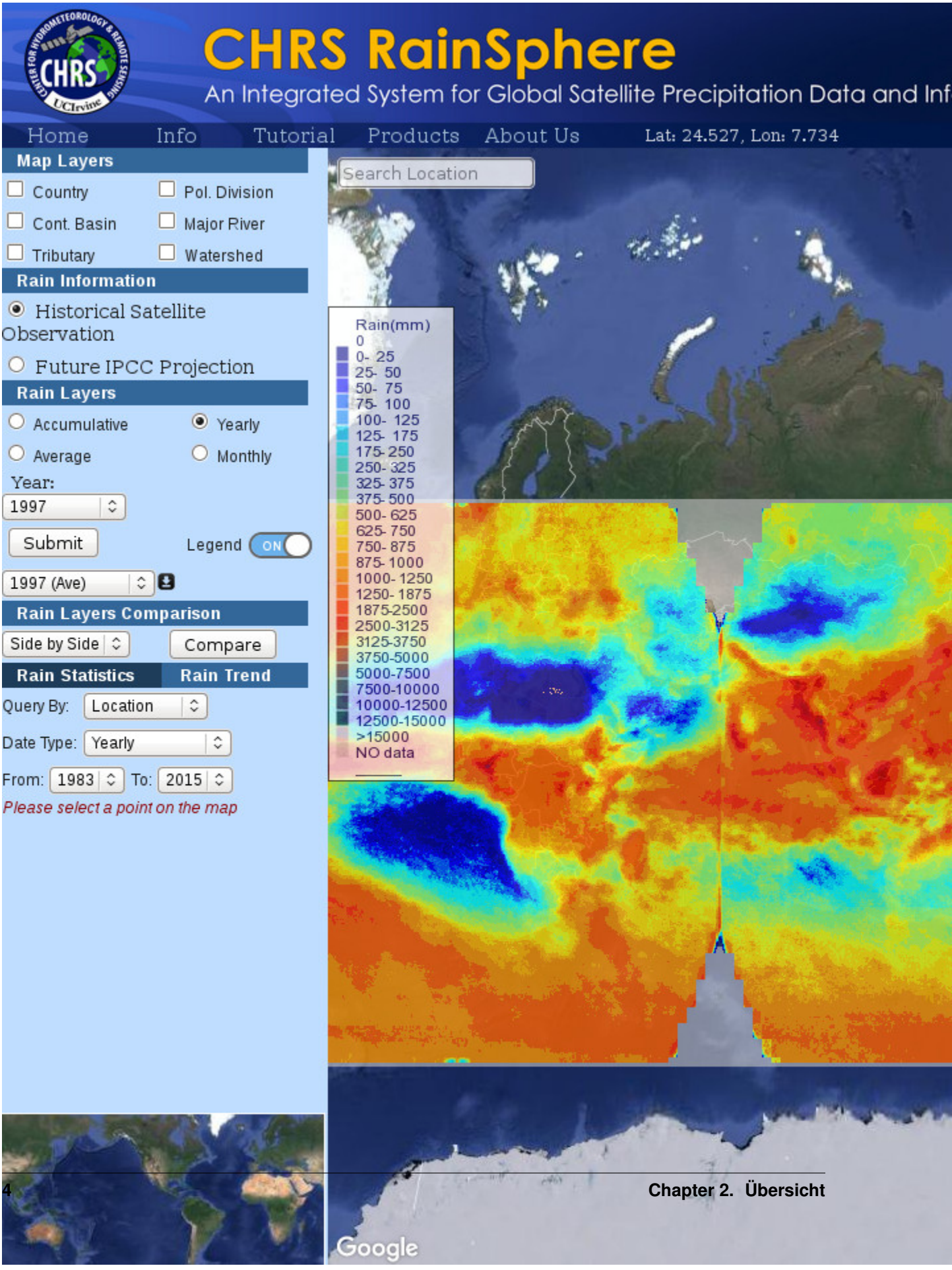
Folgende Aufgaben sind abgelegt

- *Satelliten-basierte Niederschlagsmessung*



- *Wolken und ihre Klimawirkung* wird durch das Jupyter Notebook *Analyse von globale ICON Simulationen* vertieft




CHRS RainSphere
An Integrated System for Global Satellite Precipitation Data and Inf
Home
Info
Tutorial
Products
About Us
Lat: 24.527, Lon: 7.734
Map Layers
Country
Pol. Division
Cont. Basin
Major River
Tributary
Watershed
Rain Information
Historical Satellite Observation
Future IPCC Projection
Rain Layers
Accumulative
Yearly
Average
Monthly
Year:
1997
Submit
Legend
ON
1997 (Ave)
Rain Layers Comparison
Side by Side
Compare
Rain Statistics
Rain Trend
Query By:
Location
Date Type:
Yearly
From:
1983
To:
2015
Please select a point on the map
Search Location
Rain(mm)
0
0- 25
25- 50
50- 75
75- 100
100- 125
125- 175
175-250
250- 325
325- 375
375- 500
500- 625
625- 750
750- 875
875- 1000
1000- 1250
1250- 1875
1875-2500
2500-3125
3125-3750
3750-5000
5000-7500
7500-10000
10000-12500
12500-15000
>15000
NO data
Google

2.1 Satellitenbasierte Niederschlagsmessung

- Autor: Fabian Senf
- Datum: 22.Mai 2017

In einem aktuellen Artikel im Fachjournal *Bulletin of the American Meteorological Society* hat ein amerikanisches Forscherteam das Projekt *RainSphere* vorgestellt, dass sich mit der einfachen Darstellung und Analyse von globalem Niederschlag beschäftigt. (Nguyen et al., 2017).

Die Praktikumsaufgabe beschäftigt sich mit diesem Projekt, welches unter diesem Link beschrieben wird.

Nguyen, P., Sorooshian, S., Thorstensen, A., Tran, H., Huynh, P., Pham, T., … & Ashouri, H. (2016). Exploring trends through "RainSphere": Research data transformed into public knowledge. Bulletin of the American Meteorological Society

2.1.1 Einführung

Beschäftige Dich einführend mit dem Thema:

- Lade das Paper herunter und schaue es Dir an / überfliege es.
- Finde auf Basis des Papers das Portal, in dem die Niederschlagsdaten vorgestellt werden
- Spiele ein bißchen mit den verschiedenen Visualisierungsfunktionen
- Bereite ggfs. ein Dokument vor, in dem Du recherchierte Informationen und Abbildungen sammelst

2.1.2 Globale Verteilung von Niederschlag

Die erste Aufgabe beschäftigt sich mit der globalen Verteilung von Niederschlag. Versuche anhand der visualisierten Daten folgende Fragen zu beantworten:

aus Beobachtungen

- Wie ist Niederschlag im Mittel global verteilt bzgl. Klimazonen und Niederschlagsregimen?
- Wie verändert sich die Verteilung des Niederschlags mit den Jahreszeiten?
- Gibt es Hinweise auf zeitliche Trends in den Niederschlagdaten?

Vergleich mit den Klimasimulationen

- Können Klimasimulationen die prinzipielle Verteilung des Niederschlags widerspiegeln?
- Welche Veränderungen der Niederschlagsverteilung können aus den Klimaprojektionen gewonnen werden?

2.1.3 Niederschlagsverteilung über Deutschland

Als nächstes wird die Verteilung des Niederschlags über Deutschland betrachtet. Folgende Fragen:

- Wie ist Niederschlag über Deutschland im Mittel verteilt? Warum könnte das so sein?
- Welche Veränderung gab es in der Vergangenheit und welche werden für die Zukunft erwartet?
- Wie ist die saisonale und Jahr-zu-Jahr Variabilität der Niederschlagsverteilung?

Der Deutsche Wetterdienst betreibt ein dichtes Netzwerk an bodengebundenen Niederschlagsmessungen in Kombination mit einem Radarnetzwerk für die Niederschlagsfernerkundung. Versuche über das Internet (auch auf der Seite des DWD, https://www.dwd.de, vielleicht Stichwort Radarklimatologie) Informationen über die Niederschlagsverteilung aus den bodengebundenen Messungen zu erhalten.

- Wie vergleichen sich boden- und satelliten-gestützte Niederschlagsverteilungen?

2.1.4 Niederschlagsverteilung im Urlaubsland

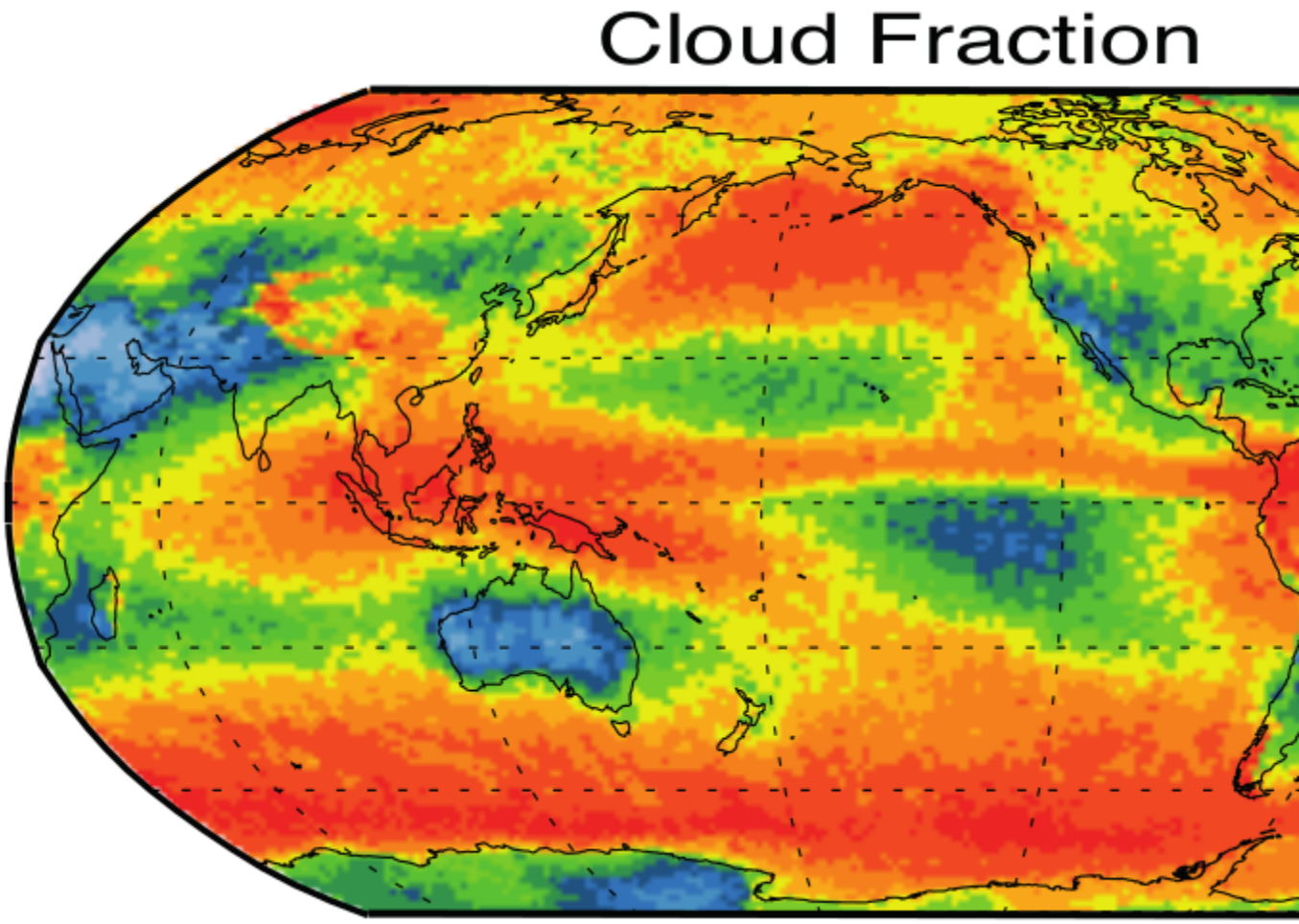
Suche Dir jetzt ein Land aus, in dem Du mal im Urlaub warst, oder in dem Du gerne Urlaub machen möchtest.

- Schaue Dir die räumliche und zeitliche Verteilung von Niederschlag an (wie oben) und vergleiche sie mit der in Deutschland.
- Was bringt die Zukunft für Dein Urlaubsland?

2.1.5 Fortgeschrittene Recherche

Versuche folgende Informationen zu recherchieren:

- Welche Meßinstrumente werden für die Analyse des Niederschlags in *RainSphere* genutzt?
- Wie ist das zugrunde liegende Meßprinzip?
- Was haben die verschiedenen Klimaszenarien RCP2.6, RCP4.5 und RCP8.5 für eine Bedeutung?



2.2 Wolken und ihre Klimawirkung

- Autor: Fabian Senf
- Datum: 19. November 2019

Wolken begegnen uns im Alltag, wenn immer uns nach draußen bewegen und nicht gerade strahlend blauer Himmel ist. Wolken spenden uns Schatten im Sommer und machen einen trüben Novembertag erst sorichtig dunkel und trist. Wenn Wolken auf uns herunterfallen, dann holen wir Schirm oder Schlitten heraus.

Doch nicht nur für uns persönlich omnipräsent und wichtig, auch das Klima unserer Erde wird durch Wolken reguliert. Die Praktikumsaufgabe beschäftigt sich mit der Frage, welche Wolken es gibt, wie sie global verteilt sind und wie sie das Klima der Erde beeinflussen.

2.2.1 Einführung

Beschäftige Dich einführend mit dem Thema:

- Verschaffe Dir einen Überblick über Wolkenarten
- Wie entstehen Wolken?

- Woraus bestehen Wolken?
- Was bewirken Wolken?

Recherchiere im Internet. Starte mit folgenden Links:

- https://wiki.bildungsserver.de/klimawandel/index.php/Wolken
- https://www.planet-wissen.de/natur/klima/wetterphaenomene/pwiewolkenalswetterzeichen100.html
- http://www.wolkenatlas.de/
- https://www.dwd.de/DE/service/lexikon/begriffe/W/Wolkenatlas_pdf.html
- https://cloudatlas.wmo.int/home.html

2.2.2 Aktuelle Bewölkungssituation

Wie ist die aktuelle Bewölkungssituation an dem Ort, wo Du Dich befindest?

- Schaue aus den Fenster! Welche Wolken siehst Du und was verändert sich im Laufe des Tages?
- Schaue Dir Satellitenbilder oder Filme an, um die Bewölkung über Gesamtdeutschland oder Europa zu sehen!
  - https://www.dwd.de/DE/leistungen/satellit_betrachter/sat-viewer/sat-viewer_node.html
  - https://kachelmannwetter.com/de/sat/satellit-hd-5min.html

Erweitert:

Bei EUMETSAT findest Du verschiedene Karten von Meteosat

- https://eumetview.eumetsat.int/mapviewer/
- Schaue Dir die verschiedenen Produkte / RGBs an und versuche zu verstehen, was Du siehst.

EUMeTrain hat ganz viel Material, um besser zu verstehen, was man mit einem Satelliten sehen kann. * Gehe durch die Tutorials bei http://www.eumetrain.org/courses/satellite_image_interpretation.html

2.2.3 Wolken im Urlaubsland

Suche Dir jetzt ein Land aus, in dem Du mal im Urlaub warst, oder in dem Du gerne Urlaub machen möchtest.

- Schaue Dir die räumliche und zeitliche Verteilung von Wolken und vergleiche sie mit der in Deutschland.

2.2.4 Globale Verteilung von Wolkenarten

Wie sind Wolken global über die Erde verteilt?

Benutze Beobachtungen von Satellitendaten um die Frage zu beantworten

- Die Amerikanische Weltraumbehörde NASA stellt eine schöne Übersicht aus Daten polarumlaufender Satelliten zur Verfügung, siehe https://worldview.earthdata.nasa.gov/.
- Hier kann man sich die Bilder des amerikanischen Satelliten GOES anschauen https://www.star.nesdis.noaa.gov/goes/.

2.2.5 Wolken im Klimasystem

Überlege Dir, wie Wolken das Klima der Erde beeinflussen können. Informationen findest Du hier: http://wiki.bildungsserver.de/klimawandel/index.php/Wolken_im_Klimasystem

Langzeit-Beobachtungen

- Das ISCCP-Projekt sammelt seit Jahrzehnten Informationen über Wolken.
  - Gehe auf die Website https://isccp.giss.nasa.gov/products/browsed2.html
  - Stelle eine Variable & Periode ein. Drücke um ein Bild zu bekommen. Was siehst Du?
  - Verändere die Eingaben und spiele damit.

Wie könnte sich unser Klima entwickeln?

- Schaue Dir den Vergleiche von Klimaprojektionen an. Hier kann man sich Vergleichskarten erzeugen. http://climexp.knmi.nl/plot_atlas_form.py. (Kaum Wolken-Variablen...)

2.2.6 Globale Simulation von Wolken

1. Hole Dir Simulationsdaten bei mir
2. Starte Jupyter Notebook
3. Führe das bereitgestellte Jupyter Notebook zum Testen aus
4. Erstelle Dir Karten
   - der mittleren Bedeckungsgrade von flachen, mittelhohen und hohen Wolken aus ICON-Simulationen.
   - der mittleren emitierten thermischen Strahlung (Infrarot)
   - der mittleren reflektieren Strahlung (Sonnenlicht)

2.3 Analyse von hoch-augelösten, globalen ICON Simulationen

2.3.1 Inhalt

Dieses Jupyter Notebook zeigt wie man globale ICON Simulationsdaten lädt und plottet.

- **Jupyter Notebook** ist eine Browser-basierte Programmierlösung, um wissenschaftliche Daten mit der Programmiersprache `Python` zu analysieren.
- **Python** ist eine Programmiersprache, die man leicht erlernen kann
- **ICON** ist ein Computermodell für die Atmosphäre (Wolken & Wetter).

2.3.2 Importieren von Modulen

Python arbeitet mit Modulen, die importiert werden, um bestimmte Aufgaben einfacher zu machen

[22]:
```
%matplotlib inline
# das führt zu Bildern, die direkt im Notebook stehen

import numpy as np  # Standard Bibliothek für mathematische Operationen
import pylab as plt # Standard Bibliothek für Plots
```

(continues on next page)

(continued from previous page)

```
import cartopy.crs as ccrs # Bibliothek für Karten / Kartographie
import xarray as xr  # Einlesen von Daten (speziell gut für wissenschaftliche Daten␣
↪im netCDF-Format )
```

2.3.3 Daten einlesen

Plan

Folgendes muss gemacht werden:

- Finde raus, wo die Daten liegen (oder kopiere die Daten in dein Dateisystem)!
- Setze den Pfad zu den Daten!
- Öffne die Daten mit `Xarray`!

Pfad setzen

[2]:
```
# Das ist mein Pfad - der muss geändert werden!
datei_pfad = '/vols/fs1/store/senf/data/dyamond/ICON-5km-fix-sst-conv-new'
```

Daten öffnen

[3]:
```
dateien_liste = '%s/2D_dyamond_accu_DOM01_ML_0[2-3]*_1x1deg.nc' % datei_pfad
```

[4]:
```
icon_daten = xr.open_mfdataset( dateien_liste, combine = 'by_coords')
```

2.3.4 Daten anschauen

Allgemein

[5]:
```
print( icon_daten )
```

```
<xarray.Dataset>
Dimensions:       (lat: 180, lon: 360, time: 200)
Coordinates:
  * lon           (lon) float64 0.0 1.0 2.0 3.0 4.0 ... 356.0 357.0 358.0 359.0
  * lat           (lat) float64 -89.5 -88.5 -87.5 -86.5 ... 86.5 87.5 88.5 89.5
  * time          (time) datetime64[ns] 2016-08-03T01:45:00 ... 2016-08-05T03:30:00
Data variables:
    ACCTHB_T      (time, lat, lon) float32 dask.array<chunksize=(1, 180, 360),␣
↪meta=np.ndarray>
    ACCSOB_T      (time, lat, lon) float32 dask.array<chunksize=(1, 180, 360),␣
↪meta=np.ndarray>
    SODT_RAD      (time, lat, lon) float32 dask.array<chunksize=(1, 180, 360),␣
↪meta=np.ndarray>
    ACCTHB_S      (time, lat, lon) float32 dask.array<chunksize=(1, 180, 360),␣
↪meta=np.ndarray>
    THUS_RAD      (time, lat, lon) float32 dask.array<chunksize=(1, 180, 360),␣
↪meta=np.ndarray>
```

(continues on next page)

(continued from previous page)

```
    ACCSOB_S      (time, lat, lon) float32 dask.array<chunksize=(1, 180, 360),␣
↪meta=np.ndarray>
    param198.4.0  (time, lat, lon) float32 dask.array<chunksize=(1, 180, 360),␣
↪meta=np.ndarray>
    param199.4.0  (time, lat, lon) float32 dask.array<chunksize=(1, 180, 360),␣
↪meta=np.ndarray>
    ACCLHFL_S     (time, lat, lon) float32 dask.array<chunksize=(1, 180, 360),␣
↪meta=np.ndarray>
    ACCSHFL_S     (time, lat, lon) float32 dask.array<chunksize=(1, 180, 360),␣
↪meta=np.ndarray>
    TOT_PREC      (time, lat, lon) float32 dask.array<chunksize=(1, 180, 360),␣
↪meta=np.ndarray>
Attributes:
    CDI:                       Climate Data Interface version 1.9.7.1 (http:/...
    history:                   Sat Nov 30 01:26:57 2019: cdo -P 4 -z zip_4 -r...
    institution:               European Centre for Medium-Range Weather Forec...
    Conventions:               CF-1.6
    CDO:                       Climate Data Operators version 1.9.7.1 (http:/...
    cdo_openmp_thread_number:  4
```

Da stehen viele Variablen drin, z.B. `ACCTHB_T`. Aber auch Koordinaten wie `time`oder `lon` sind enthalten.

Koordinaten

Erst schauen wir auf die Zeit. Sie steht in der `icon_daten.time` Variable.

[6]:
```
icon_daten.time
```

[6]:
```
<xarray.DataArray 'time' (time: 200)>
array(['2016-08-03T01:45:00.000000000', '2016-08-03T02:00:00.000000000',
       '2016-08-03T02:15:00.000000000', '2016-08-03T02:30:00.000000000',
       '2016-08-03T02:45:00.000000000', '2016-08-03T03:00:00.000000000',
       '2016-08-03T03:15:00.000000000', '2016-08-03T03:30:00.000000000',
       '2016-08-03T03:45:00.000000000', '2016-08-03T04:00:00.000000000',
       '2016-08-03T04:15:00.000000000', '2016-08-03T04:30:00.000000000',
       '2016-08-03T04:45:00.000000000', '2016-08-03T05:00:00.000000000',
       '2016-08-03T05:15:00.000000000', '2016-08-03T05:30:00.000000000',
       '2016-08-03T05:45:00.000000000', '2016-08-03T06:00:00.000000000',
       '2016-08-03T06:15:00.000000000', '2016-08-03T06:30:00.000000000',
       '2016-08-03T06:45:00.000000000', '2016-08-03T07:00:00.000000000',
       '2016-08-03T07:15:00.000000000', '2016-08-03T07:30:00.000000000',
       '2016-08-03T07:45:00.000000000', '2016-08-03T08:00:00.000000000',
       '2016-08-03T08:15:00.000000000', '2016-08-03T08:30:00.000000000',
       '2016-08-03T08:45:00.000000000', '2016-08-03T09:00:00.000000000',
       '2016-08-03T09:15:00.000000000', '2016-08-03T09:30:00.000000000',
       '2016-08-03T09:45:00.000000000', '2016-08-03T10:00:00.000000000',
       '2016-08-03T10:15:00.000000000', '2016-08-03T10:30:00.000000000',
       '2016-08-03T10:45:00.000000000', '2016-08-03T11:00:00.000000000',
       '2016-08-03T11:15:00.000000000', '2016-08-03T11:30:00.000000000',
       '2016-08-03T11:45:00.000000000', '2016-08-03T12:00:00.000000000',
       '2016-08-03T12:15:00.000000000', '2016-08-03T12:30:00.000000000',
       '2016-08-03T12:45:00.000000000', '2016-08-03T13:00:00.000000000',
       '2016-08-03T13:15:00.000000000', '2016-08-03T13:30:00.000000000',
       '2016-08-03T13:45:00.000000000', '2016-08-03T14:00:00.000000000',
       '2016-08-03T14:15:00.000000000', '2016-08-03T14:30:00.000000000',
       '2016-08-03T14:45:00.000000000', '2016-08-03T15:00:00.000000000',
```

(continues on next page)

(continued from previous page)

```
        '2016-08-03T15:15:00.000000000', '2016-08-03T15:30:00.000000000',
        '2016-08-03T15:45:00.000000000', '2016-08-03T16:00:00.000000000',
        '2016-08-03T16:15:00.000000000', '2016-08-03T16:30:00.000000000',
        '2016-08-03T16:45:00.000000000', '2016-08-03T17:00:00.000000000',
        '2016-08-03T17:15:00.000000000', '2016-08-03T17:30:00.000000000',
        '2016-08-03T17:45:00.000000000', '2016-08-03T18:00:00.000000000',
        '2016-08-03T18:15:00.000000000', '2016-08-03T18:30:00.000000000',
        '2016-08-03T18:45:00.000000000', '2016-08-03T19:00:00.000000000',
        '2016-08-03T19:15:00.000000000', '2016-08-03T19:30:00.000000000',
        '2016-08-03T19:45:00.000000000', '2016-08-03T20:00:00.000000000',
        '2016-08-03T20:15:00.000000000', '2016-08-03T20:30:00.000000000',
        '2016-08-03T20:45:00.000000000', '2016-08-03T21:00:00.000000000',
        '2016-08-03T21:15:00.000000000', '2016-08-03T21:30:00.000000000',
        '2016-08-03T21:45:00.000000000', '2016-08-03T22:00:00.000000000',
        '2016-08-03T22:15:00.000000000', '2016-08-03T22:30:00.000000000',
        '2016-08-03T22:45:00.000000000', '2016-08-03T23:00:00.000000000',
        '2016-08-03T23:15:00.000000000', '2016-08-03T23:30:00.000000000',
        '2016-08-03T23:45:00.000000000', '2016-08-04T00:00:00.000000000',
        '2016-08-04T00:15:00.000000000', '2016-08-04T00:30:00.000000000',
        '2016-08-04T00:45:00.000000000', '2016-08-04T01:00:00.000000000',
        '2016-08-04T01:15:00.000000000', '2016-08-04T01:30:00.000000000',
        '2016-08-04T01:45:00.000000000', '2016-08-04T02:00:00.000000000',
        '2016-08-04T02:15:00.000000000', '2016-08-04T02:30:00.000000000',
        '2016-08-04T02:45:00.000000000', '2016-08-04T03:00:00.000000000',
        '2016-08-04T03:15:00.000000000', '2016-08-04T03:30:00.000000000',
        '2016-08-04T03:45:00.000000000', '2016-08-04T04:00:00.000000000',
        '2016-08-04T04:15:00.000000000', '2016-08-04T04:30:00.000000000',
        '2016-08-04T04:45:00.000000000', '2016-08-04T05:00:00.000000000',
        '2016-08-04T05:15:00.000000000', '2016-08-04T05:30:00.000000000',
        '2016-08-04T05:45:00.000000000', '2016-08-04T06:00:00.000000000',
        '2016-08-04T06:15:00.000000000', '2016-08-04T06:30:00.000000000',
        '2016-08-04T06:45:00.000000000', '2016-08-04T07:00:00.000000000',
        '2016-08-04T07:15:00.000000000', '2016-08-04T07:30:00.000000000',
        '2016-08-04T07:45:00.000000000', '2016-08-04T08:00:00.000000000',
        '2016-08-04T08:15:00.000000000', '2016-08-04T08:30:00.000000000',
        '2016-08-04T08:45:00.000000000', '2016-08-04T09:00:00.000000000',
        '2016-08-04T09:15:00.000000000', '2016-08-04T09:30:00.000000000',
        '2016-08-04T09:45:00.000000000', '2016-08-04T10:00:00.000000000',
        '2016-08-04T10:15:00.000000000', '2016-08-04T10:30:00.000000000',
        '2016-08-04T10:45:00.000000000', '2016-08-04T11:00:00.000000000',
        '2016-08-04T11:15:00.000000000', '2016-08-04T11:30:00.000000000',
        '2016-08-04T11:45:00.000000000', '2016-08-04T12:00:00.000000000',
        '2016-08-04T12:15:00.000000000', '2016-08-04T12:30:00.000000000',
        '2016-08-04T12:45:00.000000000', '2016-08-04T13:00:00.000000000',
        '2016-08-04T13:15:00.000000000', '2016-08-04T13:30:00.000000000',
        '2016-08-04T13:45:00.000000000', '2016-08-04T14:00:00.000000000',
        '2016-08-04T14:15:00.000000000', '2016-08-04T14:30:00.000000000',
        '2016-08-04T14:45:00.000000000', '2016-08-04T15:00:00.000000000',
        '2016-08-04T15:15:00.000000000', '2016-08-04T15:30:00.000000000',
        '2016-08-04T15:45:00.000000000', '2016-08-04T16:00:00.000000000',
        '2016-08-04T16:15:00.000000000', '2016-08-04T16:30:00.000000000',
        '2016-08-04T16:45:00.000000000', '2016-08-04T17:00:00.000000000',
        '2016-08-04T17:15:00.000000000', '2016-08-04T17:30:00.000000000',
        '2016-08-04T17:45:00.000000000', '2016-08-04T18:00:00.000000000',
        '2016-08-04T18:15:00.000000000', '2016-08-04T18:30:00.000000000',
        '2016-08-04T18:45:00.000000000', '2016-08-04T19:00:00.000000000',
        '2016-08-04T19:15:00.000000000', '2016-08-04T19:30:00.000000000',
```

(continues on next page)

(continued from previous page)

```
       '2016-08-04T19:45:00.000000000', '2016-08-04T20:00:00.000000000',
       '2016-08-04T20:15:00.000000000', '2016-08-04T20:30:00.000000000',
       '2016-08-04T20:45:00.000000000', '2016-08-04T21:00:00.000000000',
       '2016-08-04T21:15:00.000000000', '2016-08-04T21:30:00.000000000',
       '2016-08-04T21:45:00.000000000', '2016-08-04T22:00:00.000000000',
       '2016-08-04T22:15:00.000000000', '2016-08-04T22:30:00.000000000',
       '2016-08-04T22:45:00.000000000', '2016-08-04T23:00:00.000000000',
       '2016-08-04T23:15:00.000000000', '2016-08-04T23:30:00.000000000',
       '2016-08-04T23:45:00.000000000', '2016-08-05T00:00:00.000000000',
       '2016-08-05T00:15:00.000000000', '2016-08-05T00:30:00.000000000',
       '2016-08-05T00:45:00.000000000', '2016-08-05T01:00:00.000000000',
       '2016-08-05T01:15:00.000000000', '2016-08-05T01:30:00.000000000',
       '2016-08-05T01:45:00.000000000', '2016-08-05T02:00:00.000000000',
       '2016-08-05T02:15:00.000000000', '2016-08-05T02:30:00.000000000',
       '2016-08-05T02:45:00.000000000', '2016-08-05T03:00:00.000000000',
       '2016-08-05T03:15:00.000000000', '2016-08-05T03:30:00.000000000'],
      dtype='datetime64[ns]')
Coordinates:
  * time     (time) datetime64[ns] 2016-08-03T01:45:00 ... 2016-08-05T03:30:00
Attributes:
    standard_name:  time
    axis:           T
```

Okay, die Daten sind aller 15 Minuten gespeichert und gehen über mehrere Tage.

[7]:
```
icon_daten.lon
```

[7]:
```
<xarray.DataArray 'lon' (lon: 360)>
array([  0.,   1.,   2., ..., 357., 358., 359.])
Coordinates:
  * lon      (lon) float64 0.0 1.0 2.0 3.0 4.0 ... 355.0 356.0 357.0 358.0 359.0
Attributes:
    standard_name:  longitude
    long_name:      longitude
    units:          degrees_east
    axis:           X
```

Die geographische Länge in Grad.

[8]:
```
icon_daten.lat
```

[8]:
```
<xarray.DataArray 'lat' (lat: 180)>
array([-89.5, -88.5, -87.5, -86.5, -85.5, -84.5, -83.5, -82.5, -81.5, -80.5,
       -79.5, -78.5, -77.5, -76.5, -75.5, -74.5, -73.5, -72.5, -71.5, -70.5,
       -69.5, -68.5, -67.5, -66.5, -65.5, -64.5, -63.5, -62.5, -61.5, -60.5,
       -59.5, -58.5, -57.5, -56.5, -55.5, -54.5, -53.5, -52.5, -51.5, -50.5,
       -49.5, -48.5, -47.5, -46.5, -45.5, -44.5, -43.5, -42.5, -41.5, -40.5,
       -39.5, -38.5, -37.5, -36.5, -35.5, -34.5, -33.5, -32.5, -31.5, -30.5,
       -29.5, -28.5, -27.5, -26.5, -25.5, -24.5, -23.5, -22.5, -21.5, -20.5,
       -19.5, -18.5, -17.5, -16.5, -15.5, -14.5, -13.5, -12.5, -11.5, -10.5,
        -9.5,  -8.5,  -7.5,  -6.5,  -5.5,  -4.5,  -3.5,  -2.5,  -1.5,  -0.5,
         0.5,   1.5,   2.5,   3.5,   4.5,   5.5,   6.5,   7.5,   8.5,   9.5,
        10.5,  11.5,  12.5,  13.5,  14.5,  15.5,  16.5,  17.5,  18.5,  19.5,
        20.5,  21.5,  22.5,  23.5,  24.5,  25.5,  26.5,  27.5,  28.5,  29.5,
        30.5,  31.5,  32.5,  33.5,  34.5,  35.5,  36.5,  37.5,  38.5,  39.5,
        40.5,  41.5,  42.5,  43.5,  44.5,  45.5,  46.5,  47.5,  48.5,  49.5,
        50.5,  51.5,  52.5,  53.5,  54.5,  55.5,  56.5,  57.5,  58.5,  59.5,
        60.5,  61.5,  62.5,  63.5,  64.5,  65.5,  66.5,  67.5,  68.5,  69.5,
```

(continues on next page)

(continued from previous page)

```
        70.5,  71.5,  72.5,  73.5,  74.5,  75.5,  76.5,  77.5,  78.5,  79.5,
        80.5,  81.5,  82.5,  83.5,  84.5,  85.5,  86.5,  87.5,  88.5,  89.5])
Coordinates:
  * lat      (lat) float64 -89.5 -88.5 -87.5 -86.5 -85.5 ... 86.5 87.5 88.5 89.5
Attributes:
    standard_name:  latitude
    long_name:      latitude
    units:          degrees_north
    axis:           Y
```

Die geographische Breite ebenfalls in Grad.

2.3.5 Zeit auswählen

Wir wählen einfach erst mal einen Tag aus.

[9]:
```
tag = '2016-08-04'
```

Die Schreibweise ist `Jahr-Monat-Tag`, d.h. oben steht der 4. August 2016, klar oder?

[10]:
```
icon_daten_tag = icon_daten.sel( time = tag )
icon_daten_tag = icon_daten_tag.chunk( {'time': 96 } )
print( icon_daten_tag )
```

```
<xarray.Dataset>
Dimensions:       (lat: 180, lon: 360, time: 96)
Coordinates:
  * lon           (lon) float64 0.0 1.0 2.0 3.0 4.0 ... 356.0 357.0 358.0 359.0
  * lat           (lat) float64 -89.5 -88.5 -87.5 -86.5 ... 86.5 87.5 88.5 89.5
  * time          (time) datetime64[ns] 2016-08-04 ... 2016-08-04T23:45:00
Data variables:
    ACCTHB_T      (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360),␣
↪meta=np.ndarray>
    ACCSOB_T      (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360),␣
↪meta=np.ndarray>
    SODT_RAD      (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360),␣
↪meta=np.ndarray>
    ACCTHB_S      (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360),␣
↪meta=np.ndarray>
    THUS_RAD      (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360),␣
↪meta=np.ndarray>
    ACCSOB_S      (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360),␣
↪meta=np.ndarray>
    param198.4.0  (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360),␣
↪meta=np.ndarray>
    param199.4.0  (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360),␣
↪meta=np.ndarray>
    ACCLHFL_S     (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360),␣
↪meta=np.ndarray>
    ACCSHFL_S     (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360),␣
↪meta=np.ndarray>
    TOT_PREC      (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360),␣
↪meta=np.ndarray>
Attributes:
    CDI:                      Climate Data Interface version 1.9.7.1 (http:/...
    history:                  Sat Nov 30 01:26:57 2019: cdo -P 4 -z zip_4 -r...
```

(continues on next page)

(continued from previous page)

```
    institution:                European Centre for Medium-Range Weather Forec...
    Conventions:                CF-1.6
    CDO:                        Climate Data Operators version 1.9.7.1 (http:/...
    cdo_openmp_thread_number:   4
```

Mit `icon_daten.sel` können Datenbereiche ausgewählt werden (siehe Link)

96 Zeitschritte. Das kommt von 4 Datenfeldern pro Stunde und 24 Stunden pro Tag.

[11]:
```
24 * 4
```

[11]:
```
96
```

Auch klar, oder?

2.3.6 Variablen genauer anschauen

Zwei Variablen repräsentieren Strahlungsflüsse am Oberrand der Atmosphäre (wo auch immer der nun wieder liegt ...).

Die Variablen heißen:

- `ACCTHB_T`
- `ACCSOB_T`

Wir schauen mal genauer hin:

[12]:
```
print(icon_daten_tag['ACCTHB_T'])
```

```
<xarray.DataArray 'ACCTHB_T' (time: 96, lat: 180, lon: 360)>
dask.array<rechunk-merge, shape=(96, 180, 360), dtype=float32, chunksize=(96, 180,␣
↪360), chunktype=numpy.ndarray>
Coordinates:
  * lon      (lon) float64 0.0 1.0 2.0 3.0 4.0 ... 355.0 356.0 357.0 358.0 359.0
  * lat      (lat) float64 -89.5 -88.5 -87.5 -86.5 -85.5 ... 86.5 87.5 88.5 89.5
  * time     (time) datetime64[ns] 2016-08-04 ... 2016-08-04T23:45:00
Attributes:
    standard_name:  toa_outgoing_longwave_flux
    long_name:      Net long wave radiation flux  - accumulated _ model top
    units:          W m-2
    param:          5.5.0
    level_type:     toa
```

Das ist der Strahlungsfluss der thermischen (oder infraroten) Ausstrahlung, der nach oben geht. Die Erde strahlt mit ihrer eigenen Temperatur (siehe Wärmestrahlung).

Interpretation: Das Wort `accumulated` (im `long_name` Attribut) bedeutet, dass die Strahlungsleistung pro Fläche (Watt pro Quadradmeter) aufsummiert wird und so die "akkumulierte" Strahlungsenergie pro Fläche (Joule pro Quadradmeter) abgespeichert wird.

Achtung: Das Attribut `units` ist also falsch. Da müsste `J m-2` stehen!

[13]:
```
print(icon_daten_tag['ACCSOB_T'])
```

```
<xarray.DataArray 'ACCSOB_T' (time: 96, lat: 180, lon: 360)>
dask.array<rechunk-merge, shape=(96, 180, 360), dtype=float32, chunksize=(96, 180,␣
↪360), chunktype=numpy.ndarray>
Coordinates:
```

(continues on next page)

(continued from previous page)

```
  * lon      (lon) float64 0.0 1.0 2.0 3.0 4.0 ... 355.0 356.0 357.0 358.0 359.0
  * lat      (lat) float64 -89.5 -88.5 -87.5 -86.5 -85.5 ... 86.5 87.5 88.5 89.5
  * time     (time) datetime64[ns] 2016-08-04 ... 2016-08-04T23:45:00
Attributes:
    long_name:   Net short wave radiation flux - accumulated _ model top
    units:       W m-2
    param:       9.4.0
    level_type:  toa
```

Das ist der akkumulierte Strahlungsfluss im solaren Bereich (also durch sichtbare Sonnenstrahlung bestimmt). Auch hier sollte die Einheit `J m-2` sein.

Wir nehmen nur die beiden Strahlungsvariablen:

[14]:
```
icon_akku_strahlung = icon_daten_tag[['ACCSOB_T', 'ACCTHB_T']]
```

2.3.7 Strahlungsflüsse

Um einen Strahlungsfluss aus unseren Daten zu erhalten, muss also die Strahlungsenergie (pro Fläche) zu zwei Zeiten (jetzt und 15 min früher) betrachtet werden. Berechnet man die Differenz der beiden Strahlungsenergien, so bekommt man die Energie, die innerhalb von 15 min dazu gekommen ist. Dividiert man diese Differenz noch durch das Zeitinterval bekommt man eine momentane Strahlungsleistung pro Fläche.

Wir nehmen die `xarray`-Methode `differentiate`. Mit ihr können wir ganz einfach Differenzen berechnen.

[17]:
```
icon_strahlungsfluss = icon_akku_strahlung.differentiate('time', datetime_unit = 's')
```

Ich glaube, es wäre auch schöner die Strahlungsflüsse umzubenennen. Üblich sind englische Kürzel für "shortwave flux" (swf) und "longwave flux" (lwf). Wir nehmen dafür die Methode `icon_strahlungsfluss.rename_vars` und müssen ein "Dictionary" übergeben, dass alte in neue Namen abbildet.

[18]:
```
icon_strahlungsfluss = icon_strahlungsfluss.rename_vars({'ACCSOB_T' : 'swf', 'ACCTHB_T
↪' : 'lwf'})
```

[19]:
```
print( icon_strahlungsfluss )
```

```
<xarray.Dataset>
Dimensions:  (lat: 180, lon: 360, time: 96)
Coordinates:
  * lon      (lon) float64 0.0 1.0 2.0 3.0 4.0 ... 355.0 356.0 357.0 358.0 359.0
  * time     (time) datetime64[ns] 2016-08-04 ... 2016-08-04T23:45:00
  * lat      (lat) float64 -89.5 -88.5 -87.5 -86.5 -85.5 ... 86.5 87.5 88.5 89.5
Data variables:
    swf      (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360), meta=np.
↪ndarray>
    lwf      (time, lat, lon) float32 dask.array<chunksize=(96, 180, 360), meta=np.
↪ndarray>
Attributes:
    CDI:                       Climate Data Interface version 1.9.7.1 (http:/...
    history:                   Sat Nov 30 01:26:57 2019: cdo -P 4 -z zip_4 -r...
    institution:               European Centre for Medium-Range Weather Forec...
    Conventions:               CF-1.6
    CDO:                       Climate Data Operators version 1.9.7.1 (http:/...
    cdo_openmp_thread_number:  4
```

So, die Daten sehen doch ganz gut aus!

2.3.8 Plots für einen Zeitpunkt

Wir wollen für einen Zeitpunkt, die Strahlungsflüsse in eine Karte plotten.

- Zeitpunkt und Feld wählen
- Karte plotten

Langwellig

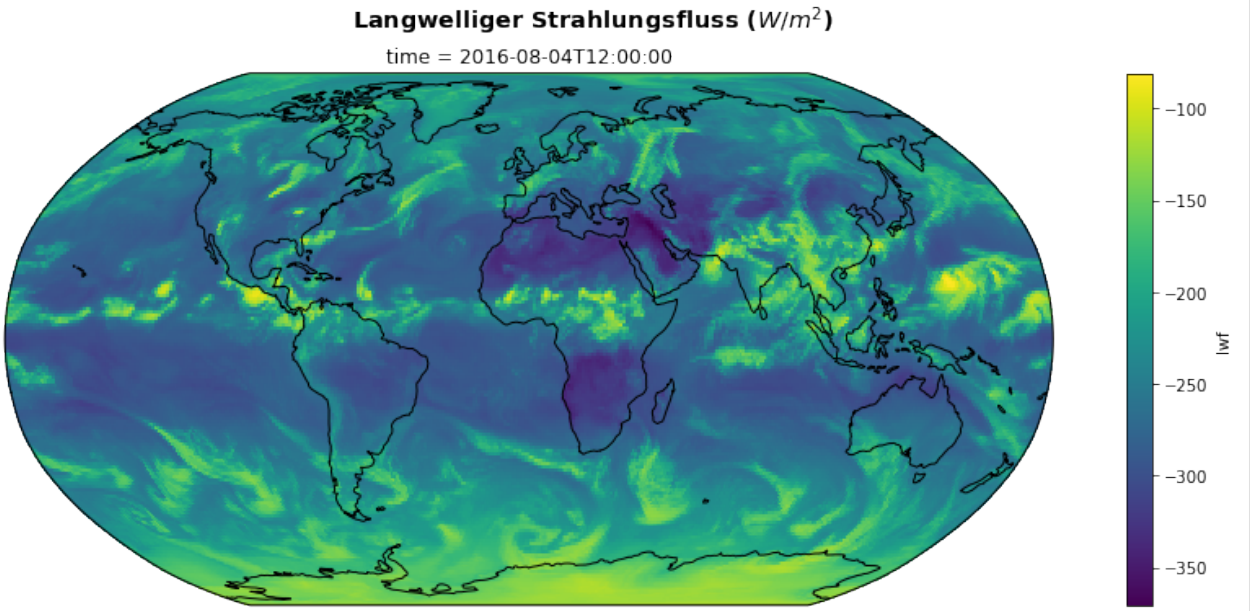
[36]:
```
zeit_punkt = '2016-08-04 12:00'
variable = icon_strahlungsfluss['lwf'].sel(time = zeit_punkt)
```

[37]:
```
fig = plt.figure(figsize=(16, 6))

ax = fig.add_subplot(1, 1, 1, projection=ccrs.Robinson())
ax.coastlines()
ax.set_global()
crs = ccrs.PlateCarree()

variable.plot.pcolormesh( ax = ax, transform = crs )
plt.suptitle('Langwelliger Strahlungsfluss ($W / m^2$)', fontsize = 'x-large',␣
↪fontweight = 'bold' )
```

[37]:
```
Text(0.5, 0.98, 'Langwelliger Strahlungsfluss ($W / m^2$)')
```



Kurzwellig

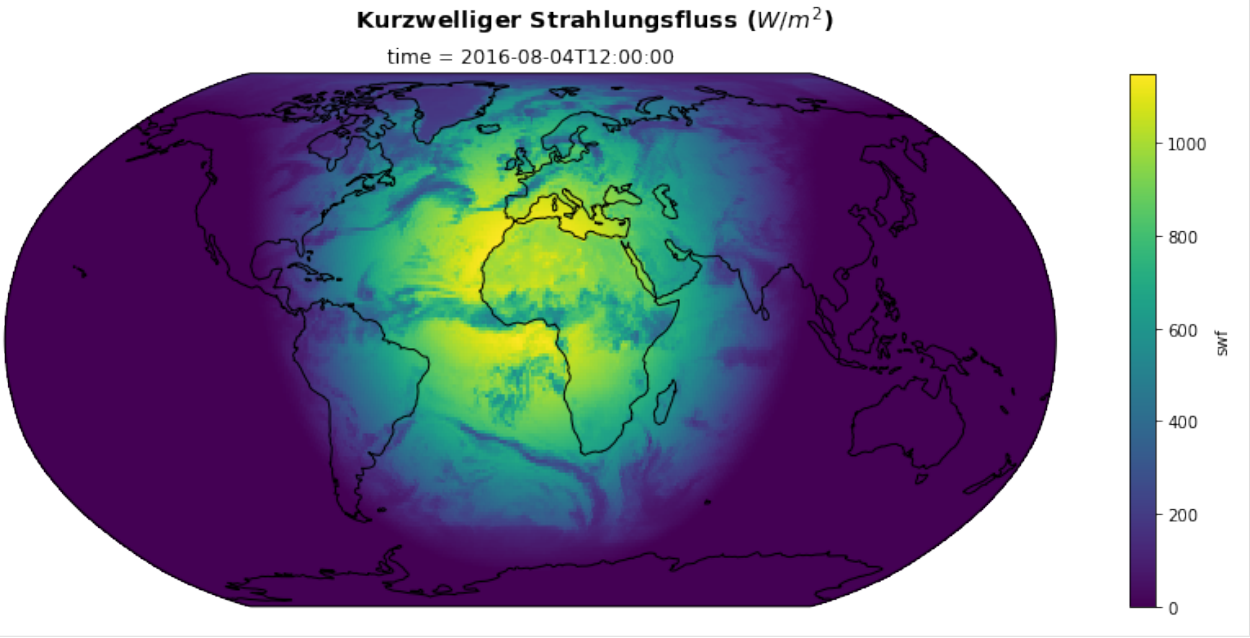
[38]:
```
zeit_punkt = '2016-08-04 12:00'
variable = icon_strahlungsfluss['swf'].sel(time = zeit_punkt)
```

[42]:
```
fig = plt.figure(figsize=(16, 6))

ax = fig.add_subplot(1, 1, 1, projection=ccrs.Robinson())
ax.coastlines()
ax.set_global()
crs = ccrs.PlateCarree()

variable.plot.pcolormesh( ax = ax, transform = crs )
plt.suptitle('Kurzwelliger Strahlungsfluss ($W / m^2$)', fontsize = 'x-large',␣
↪fontweight = 'bold' )
```

[42]:
```
Text(0.5, 0.98, 'Kurzwelliger Strahlungsfluss ($W / m^2$)')
```



Was siehst Du? Verstehst Du die Unterschiede?

2.3.9 Plots für einen ganzen Tag

Wir wollen für einen ganzen Tag, die Strahlungsflüsse in eine Karte plotten.

- Feld wählen und Mittelwert bilden
- Karte plotten

Langwellig

[43]:
```
variable = icon_strahlungsfluss['lwf'].mean('time')
```

[44]:
```
fig = plt.figure(figsize=(16, 6))

ax = fig.add_subplot(1, 1, 1, projection=ccrs.Robinson())
ax.coastlines()
ax.set_global()
```

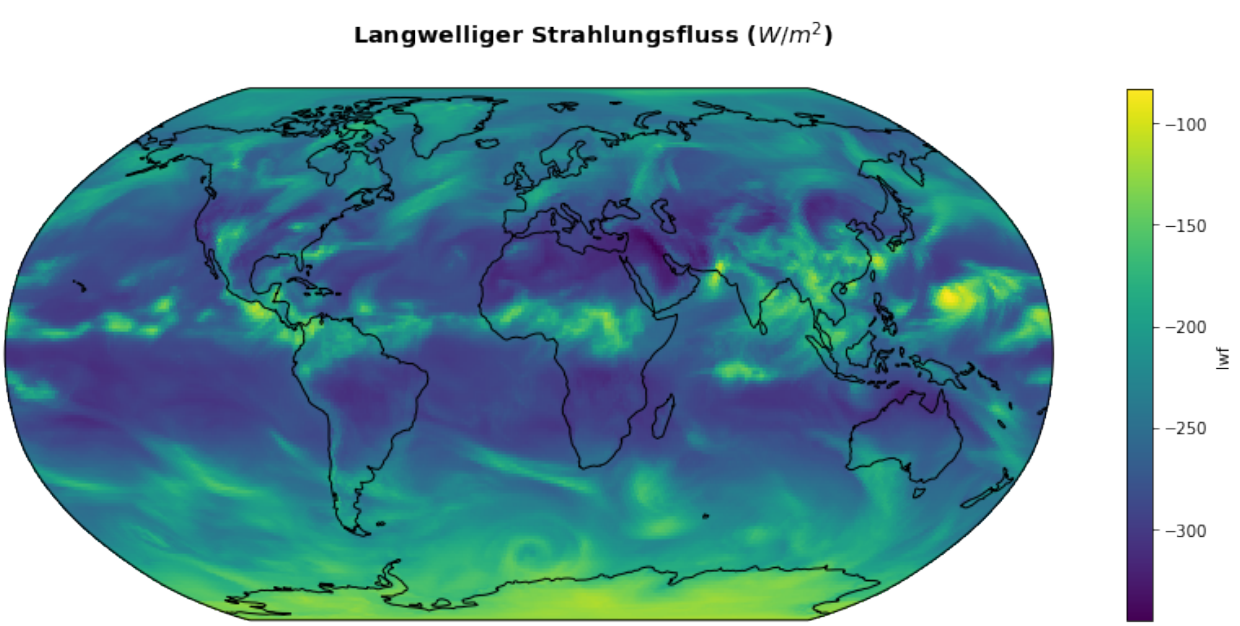
(continues on next page)

(continued from previous page)

```
crs = ccrs.PlateCarree()

variable.plot.pcolormesh( ax = ax, transform = crs )
plt.suptitle('Langwelliger Strahlungsfluss ($W / m^2$)', fontsize = 'x-large',␣
↪fontweight = 'bold' )
```

[44]:
```
Text(0.5, 0.98, 'Langwelliger Strahlungsfluss ($W / m^2$)')
```



Kurzwellig

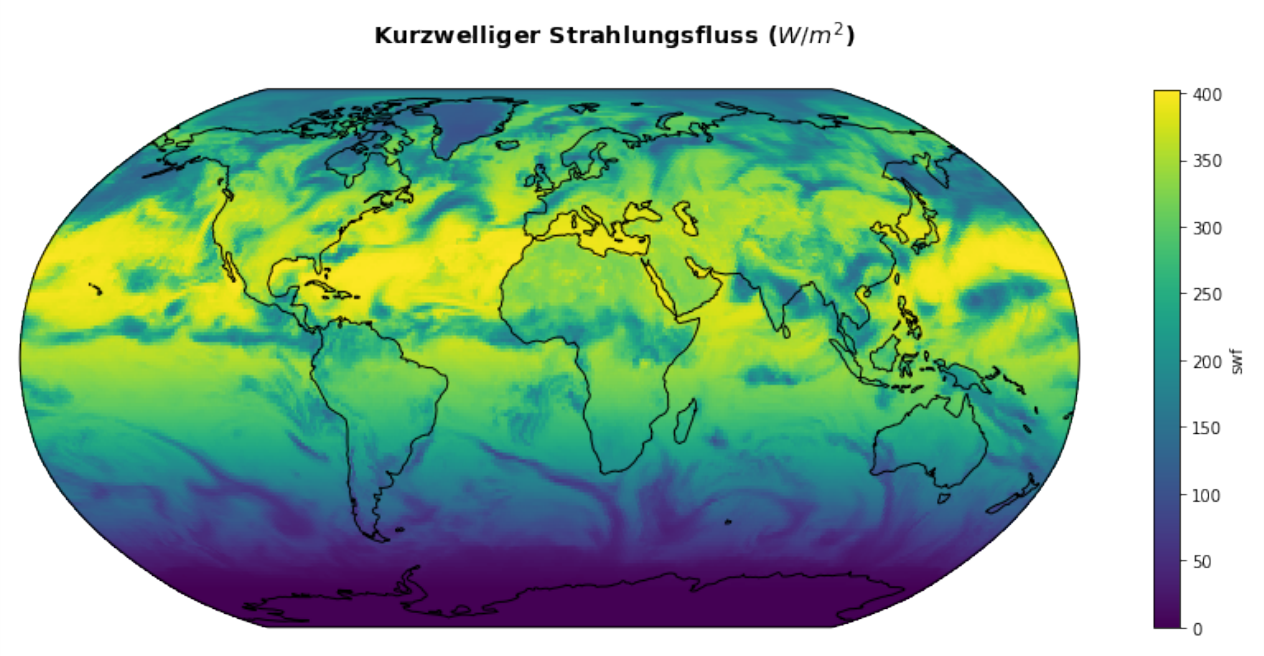
[45]:
```
variable = icon_strahlungsfluss['swf'].mean('time')
```

[46]:
```
fig = plt.figure(figsize=(16, 6))

ax = fig.add_subplot(1, 1, 1, projection=ccrs.Robinson())
ax.coastlines()
ax.set_global()
crs = ccrs.PlateCarree()

variable.plot.pcolormesh( ax = ax, transform = crs )
plt.suptitle('Kurzwelliger Strahlungsfluss ($W / m^2$)', fontsize = 'x-large',␣
↪fontweight = 'bold' )
```

[46]:
```
Text(0.5, 0.98, 'Kurzwelliger Strahlungsfluss ($W / m^2$)')
```



Was siehst Du jetzt?

Vergleiche mit den Abbildungen oben!