ILMATIETEEN LAITOS
METEOROLOGISKA INSTITUTET
FINNISH METEOROLOGICAL INSTITUTE

Tools for visualization and analysis of climate related data

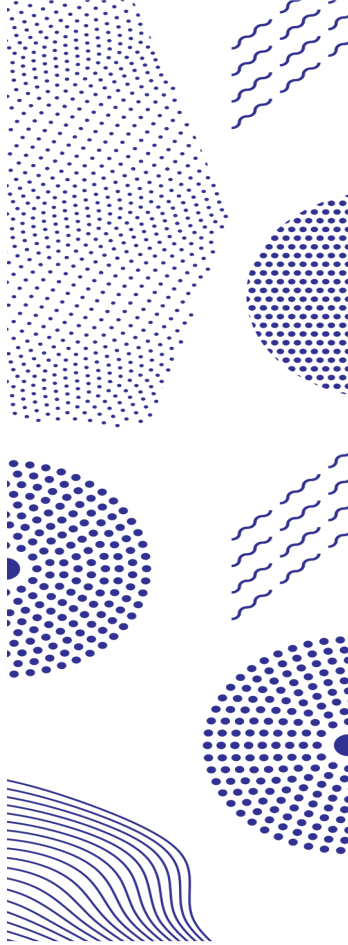
Antti Mäkelä

Finnish Meteorological Institute (FMI)

28.10.2021

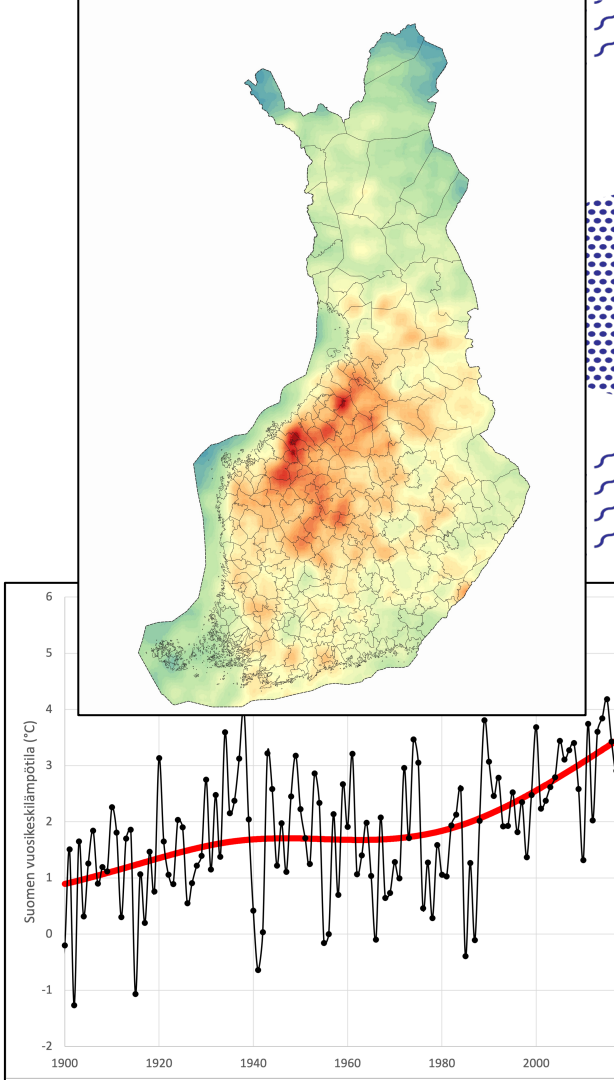
Contents

- Introduction
- C3S CDS Toolbox
- Summary



Introduction

- The method(s) for analysing/visualising climate data depends on the purpose
  - Plots
  - Value tables
  - Maps
  - Interactive maps/plots
  - etc
- Python and R scripting nowadays very popular
- “Traditional” CDO (Climate Data Operators)
  - CDO is a collection of command line Operators to manipulate and analyse Climate and NWP model Data.
    Supported data formats are GRIB, netCDF, SERVICE, EXTRA and IEG. There are more than 600 operators available. https://code.mpimet.mpg.de/projects/cdo/
- GIS-applications → plenty of functions available



C3S CDS Toolbox

Climate Change

Antti Mäkelä

Material from C3S_512 project

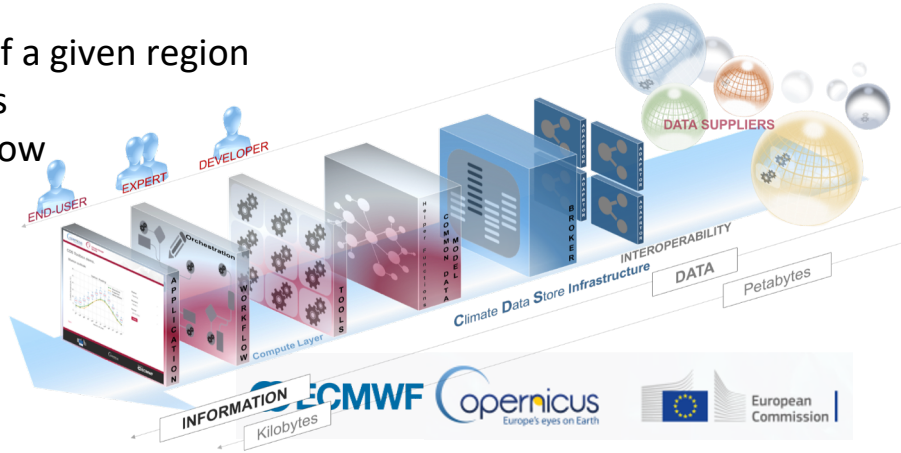
Copernicus Europe's eyes on Earth | European Commission | ECMWF

The CDS Toolbox in a nutshell

- The Toolbox allows users to develop applications that make use of the data content of the CDS
- The Toolbox allows data manipulation (calculation, plotting, selection, downloading...) and can be accessed here: https://cds.climate.copernicus.eu/user/login?destination=/toolbox-user
- The software runs on a server (not on the user's computer), and is accessible through web browsers.
- Each user has a private workspace, where the personal software is stored. Documentation and examples are available to all users. More here: https://cds.climate.copernicus.eu/toolbox/doc/tutorial.html

- Glossary:
  - TOOL = a basic function, e.g. the selection of a given region
  - WORKFLOW = a script using a series of tools
  - APPLICATION = a web interface for a workflow

- The user of the Toolbox can write python software to access, manipulate and export climate data made available through the CDS.



CDS Toolbox – MANIPULATE DATA

Climate Change

Dedicated to Expert Users to build **workflows** and **applications**

Chiara Cagnazzo Logout

Your feedback helps us to improve the service

Home Search Datasets Applications Your requests Toolbox FAQ

Toolbox editor

Applications Data Documentation

Search for app or example

- navigation
  - **Iceberg presence**
  - Route optimiser dynamic map
  - Operational indicators dynamic map
  - data_production_monthly_probability_of_exeeding
  - data_production_tc_active_points
  - Shaft power on routes dynamic map
  - Tropical cyclones by basins
  - Cost of arctic route
  - Extremes probability of exceedance on seasonal forecast
  - Route optimiser
  - Iceberg risk
  - Projected ice class limits
  - Projected ice limits
  - Extremes probability of exceedance
  - Arctic ice limits
  - Seasonal tercile summary
  - Speed over ground on fields and routes
  - Historic arctic route availability
  - Projected arctic route availability
  - Shaft power on fields and routes
- examples
  - 00 Hello World
  - 01 Retrieve data
  - 02 Plot map
  - 03 Extract time series and plot graph
  - 11 Calculate time mean and standard deviation

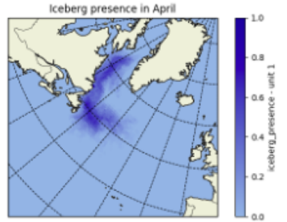
Iceberg presence | Console | Your queue

Layout | Copy | Run

```
import cdstoolbox as ct
import calendar

layout = {
    'input_ncols': 1,
    'output_ncols': 1,
    'output_align': 'bottom',
}
@ct.application(title='Iceberg presence', layout=layout)
@ct.output.carousel()
@ct.output.markdown()
def iceberg_presence():
    from matplotlib import colors
    ua,  va, sst = ct.catalogue.retrieve(
        'reanalysis-era5-single-levels',
        {
            'variable': [
                '10m_u_component_of_wind',
                '10m_v_component_of_wind',
                'sea_surface_temperature'
            ],
            'product_type': 'reanalysis',
            'year': [
                '2009', '2010', '2011', '2012',
                '2013','2014', '2015'
            ],
            'month': [
                '1', '2', '3', '4',
                '5', '6', '7', '8',
                '9', '10', '11', '12'
            ],
            'day': [
                    '01', '02', '03',
                    '04', '05', '06',
                    '07', '08', '09',
                    '10', '11', '12',
```

Iceberg presence



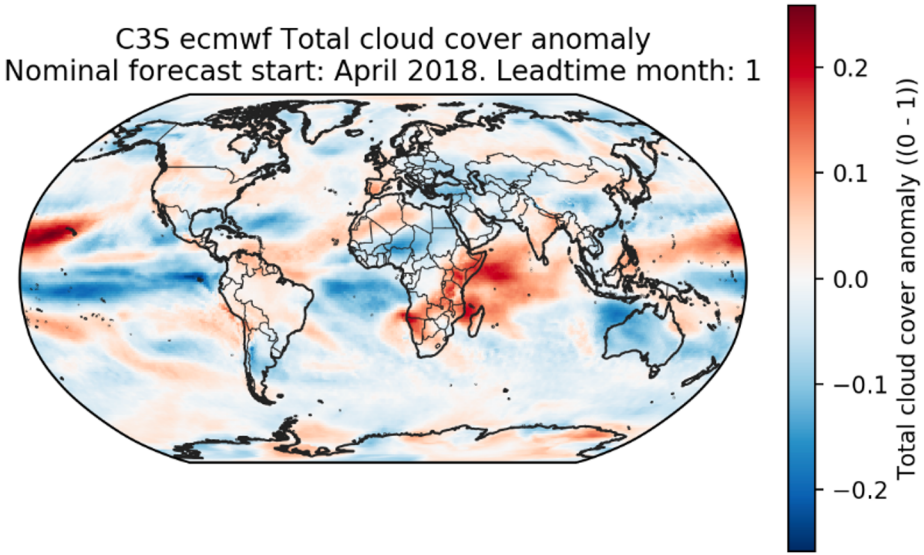
data source: CDS and CMEMS (http://marine.copernicus.eu)

Climate Change

CDS Toolbox – CREATE APPLICATIONS

Dedicated to End-Users. Can be published, described and accessed through the CDS

Seasonal



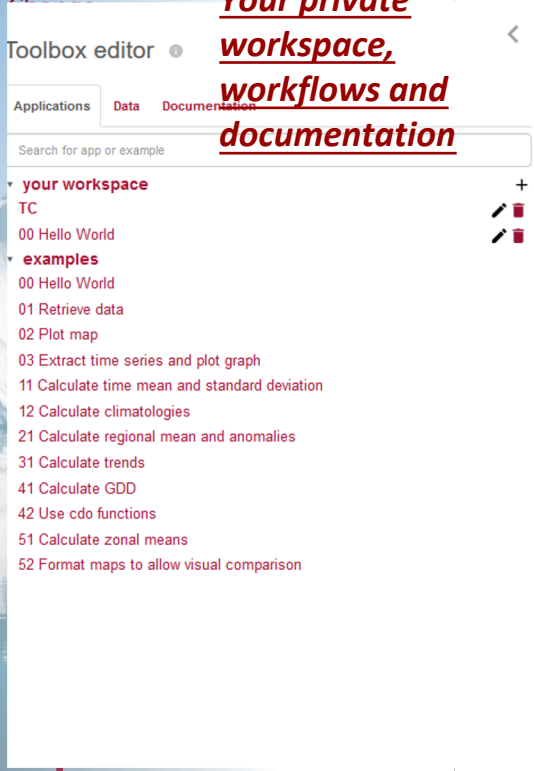
Built for the user to be Interactive

ECMWF Copernicus Europe's eyes on Earth European Commission

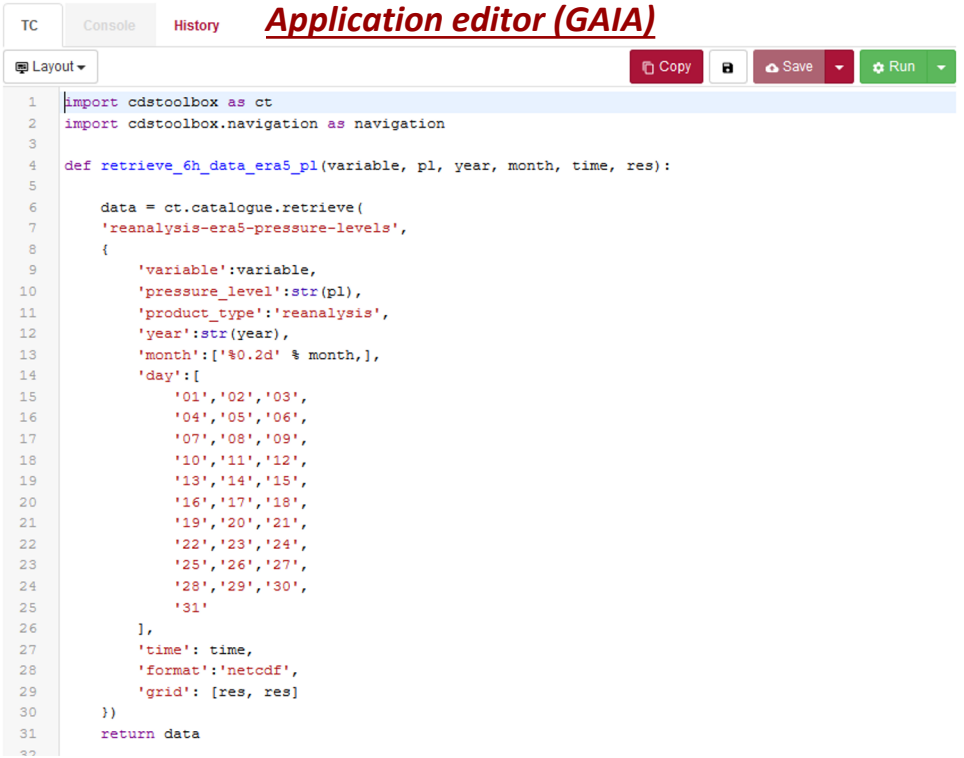
The CDS Toolbox: the elements

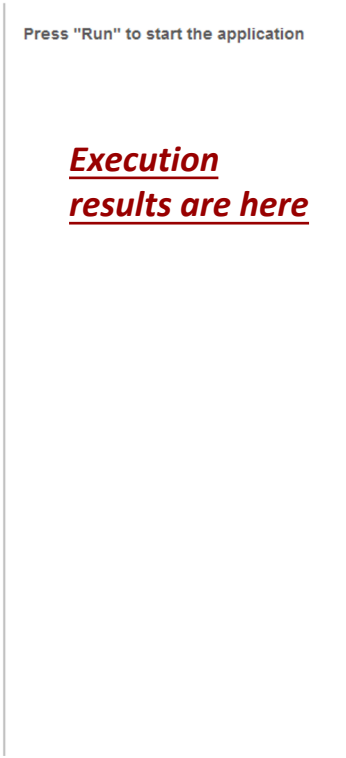
Climate

Your private workspace, workflows and documentation



Application editor (GAIA)





Execution results are here

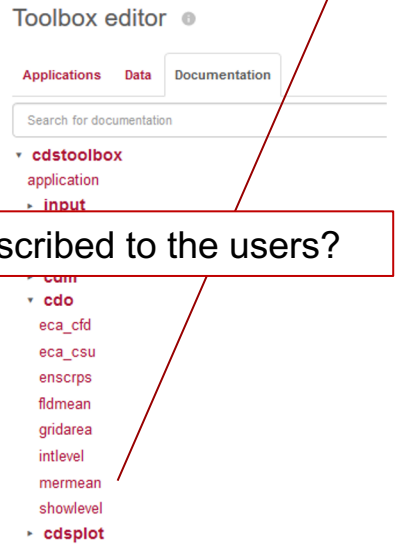
ECMWF Copernicus Europe's eyes on Earth European Commission

The CDS toolbox: tools' documentation

Climate Change

- Each Toolbox tool has a documentation on input(s), output(s), and examples. This is accessed through the documentation tab.
- The documentation is organized by tool category: plotting, climate indices, SIS,...
- Tools make use of the xarray python library to I/O and manipulate data.
- **Note: Not all the CDS datasets can be processed with the toolbox**

How/where should this be described to the users?



cdstoolbox.cdo.mermean(*data: xr.DataArray, extent: Tuple[float, float, float, float] = None, output_file: str = '{name}-{uuid}.nc', remove_atexit: bool = True*) → xr.DataArray [source]

Computes meridional mean for a xarray DataArray (weighted over all latitudes) using `mermean` cdo command. NaN or missing values are discarded.

Parameters:
- **data** – xr.DataArray DataArray to compute mermean on.
- **extent** – tuple (lon_min, lon_max, lat_min, lat_max), tuple of floats for setting data extent. If None whole data extent is considered. For applying the extent on input, cdo sellonlatbox command is used.
- **output_file** – str String for output file path. If None a random local temporary file is generated.
- **remove_atexit** – bool bool to decide whether keeping the output as netcdf file on disk

Returns: xr.DataArray Returns xarray DataArray of area averaged input.

Example:

It computes a simple mean when used on an array that has a grid without coordinates units:

```
>>> data = xr.DataArray(
...     [[1., 3.], [2., 2.]],
...     coords=[('lon', [0., 20.]), ('lat', [0., 20.])],
...     name='data'
... )
>>> mermean(data).values
array([[2.]])
```

Otherwise it computes the area weighted average:

```
>>> data.lon.attrs['units'] = 'degrees_east'
>>> data.lat.attrs['units'] = 'degrees_north'
>>> fldmean(data).values
array([[1.98...]])
```

ECMWF Copernicus Europe's eyes on Earth European Commission

The CDS Toolbox: (example) workflows

Climate Change

```python
import cdstoolbox as ct


@ct.application(title='Plot Map')
@ct.input.dropdown('variable', values=[
    '2m_temperature', '10m_u_component_of_wind',
'10m_v_component_of_wind',
    'mean_sea_level_pressure', 'sea_surface_temperature',
])
@ct.output.figure()
def plot_map(variable):
    """
    Application main steps:

    - retrieve a variable over a defined date
    - show the result on a map

    """

    data = ct.catalogue.retrieve(
        'reanalysis-era5-single-levels',
        {
            'variable': variable,
            'grid': ['3', '3'],
            'product_type': 'reanalysis',
            'year': '2010',
            'month': '08',
            'day': '15',
            'time': '12:00',
        }
    )

    fig = ct.cdsplot.geomap(data)

    return fig


```

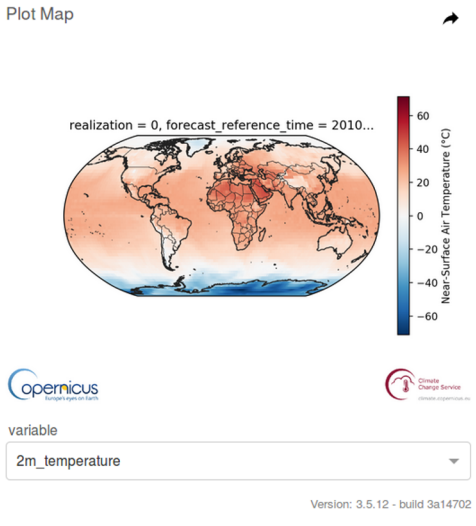
← Title and user-defined parameters

← Definition of the 'main'

PRODUCE THIS PLOT

← Retrieve data -- might be slow!

← Make a map with the results (a built-in tool)



ECMWF Copernicus Europe's eyes on Earth European Commission

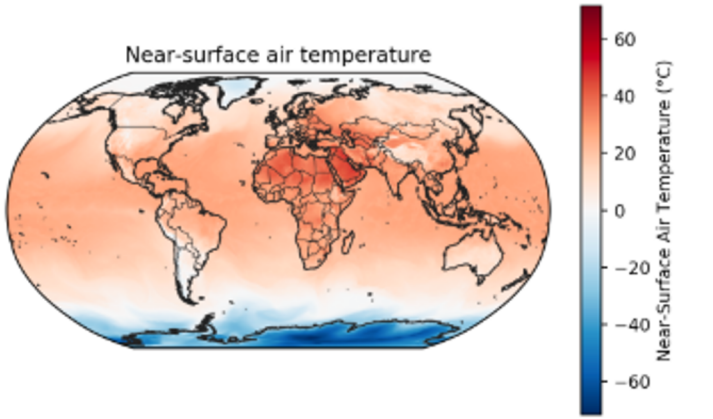
Application Structure

Clima
Chan

Plot Map

Variable

Near-Surface Air Temperature




```python
import cdstoolbox as ct

layout = {
    'output_align': 'bottom'
}

variables = {
    'Near-Surface Air Temperature': '2m_temperature',
    'Eastward Near-Surface Wind': '10m_u_component_of_wind',
    'Westward Near-Surface Wind': '10m_v_component_of_wind',
    'Sea Level Pressure': 'mean_sea_level_pressure',
    'Sea Surface Temperature': 'sea_surface_temperature',
}


@ct.application(title='Plot Map', layout=layout)
@ct.input.dropdown('variable', label='Variable', values=variables.keys())
@ct.output.figure()
def plot_map(variable):
    """
    Application main steps:

    - set the application layout with output at the bottom
    - select a variable name from a list in the dropdown menu
    - retrieve the selected variable
    - compose a title
    - show the result on a map using the chosen title

    """

    data = ct.catalogue.retrieve(
        'reanalysis-era5-single-levels',
        {
            'variable': variables[variable],
            'product_type': 'reanalysis',
            'year': '2010',
            'month': '08',
            'day': '15',
            'time': '12:00',
        }
    )

    title = '{}'.format(' '.join([text.capitalize() for text in variable.split('_')]))
    fig = ct.cdsplot.geomap(data, title=title)

    return fig
```

Climate Change

Application Structure

```python
import cdstoolbox as ct

layout = {
    'output_align': 'bottom'
}

variables = {
    'Near-Surface Air Temperature': '2m_temperature',
    'Eastward Near-Surface Wind': '10m_u_component_of_wind',
    'Westward Near-Surface Wind': '10m_v_component_of_wind',
    'Sea Level Pressure': 'mean_sea_level_pressure',
    'Sea Surface Temperature': 'sea_surface_temperature',
}


@ct.application(title='Plot Map', layout=layout)
@ct.input.dropdown('variable', label='Variable', values=variables.keys())
@ct.output.figure()
def plot_map(variable):
    """
    Application main steps:

    - set the application layout with output at the bottom
    - select a variable name from a list in the dropdown menu
    - retrieve the selected variable
    - compose a title
    - show the result on a map using the chosen title

    """

    data = ct.catalogue.retrieve(
        'reanalysis-era5-single-levels',
        {
            'variable': variables[variable],
            'product_type': 'reanalysis',
            'year': '2010',
            'month': '08',
            'day': '15',
            'time': '12:00',
        }
    )

    title = '{}'.format(' '.join([text.capitalize() for text in variable.split('_')]))
    fig = ct.cdsplot.geomap(data, title=title)

    return fig
```

Application Structure

Climate Change

Import Libraries

```
import cdstoolbox as ct

layout = {
    'output_align': 'bottom'
}

variables = {
    'Near-Surface Air Temperature': '2m_temperature',
    'Eastward Near-Surface Wind': '10m_u_component_of_wind',
    'Westward Near-Surface Wind': '10m_v_component_of_wind',
    'Sea Level Pressure': 'mean_sea_level_pressure',
    'Sea Surface Temperature': 'sea_surface_temperature',
}


@ct.application(title='Plot Map', layout=layout)
@ct.input.dropdown('variable', label='Variable', values=variables.keys())
@ct.output.figure()
def plot_map(variable):
    """
    Application main steps:

    - set the application layout with output at the bottom
    - select a variable name from a list in the dropdown menu
    - retrieve the selected variable
    - compose a title
    - show the result on a map using the chosen title

    """

    data = ct.catalogue.retrieve(
        'reanalysis-era5-single-levels',
        {
            'variable': variables[variable],
            'product_type': 'reanalysis',
            'year': '2010',
            'month': '08',
            'day': '15',
            'time': '12:00',
        }
    )

    title = '{}'.format(' '.join([text.capitalize() for text in variable.split('_')]))
    fig = ct.cdsplot.geomap(data, title=title)

    return fig
```

Climate Change

Application Structure

Import Libraries
Define Layout Style

```
import cdstoolbox as ct

layout = {
    'output_align': 'bottom'
}

variables = {
    'Near-Surface Air Temperature': '2m_temperature',
    'Eastward Near-Surface Wind': '10m_u_component_of_wind',
    'Westward Near-Surface Wind': '10m_v_component_of_wind',
    'Sea Level Pressure': 'mean_sea_level_pressure',
    'Sea Surface Temperature': 'sea_surface_temperature',
}


@ct.application(title='Plot Map', layout=layout)
@ct.input.dropdown('variable', label='Variable', values=variables.keys())
@ct.output.figure()
def plot_map(variable):
    """
    Application main steps:

    - set the application layout with output at the bottom
    - select a variable name from a list in the dropdown menu
    - retrieve the selected variable
    - compose a title
    - show the result on a map using the chosen title

    """

    data = ct.catalogue.retrieve(
        'reanalysis-era5-single-levels',
        {
            'variable': variables[variable],
            'product_type': 'reanalysis',
            'year': '2010',
            'month': '08',
            'day': '15',
            'time': '12:00',
        }
    )

    title = '{}'.format(' '.join([text.capitalize() for text in variable.split('_')]))
    fig = ct.cdsplot.geomap(data, title=title)

    return fig
```

Climate Change

Application Structure

Import Libraries
Define Layout Style
Define Variables as str

```python
import cdstoolbox as ct

layout = {
    'output_align': 'bottom'
}

variables = {
    'Near-Surface Air Temperature': '2m_temperature',
    'Eastward Near-Surface Wind': '10m_u_component_of_wind',
    'Westward Near-Surface Wind': '10m_v_component_of_wind',
    'Sea Level Pressure': 'mean_sea_level_pressure',
    'Sea Surface Temperature': 'sea_surface_temperature',
}

@ct.application(title='Plot Map', layout=layout)
@ct.input.dropdown('variable', label='Variable', values=variables.keys())
@ct.output.figure()
def plot_map(variable):
    """
    Application main steps:

    - set the application layout with output at the bottom
    - select a variable name from a list in the dropdown menu
    - retrieve the selected variable
    - compose a title
    - show the result on a map using the chosen title

    """

    data = ct.catalogue.retrieve(
        'reanalysis-era5-single-levels',
        {
            'variable': variables[variable],
            'product_type': 'reanalysis',
            'year': '2010',
            'month': '08',
            'day': '15',
            'time': '12:00',
        }
    )

    title = '{}'.format(' '.join([text.capitalize() for text in variable.split('_')]))
    fig = ct.cdsplot.geomap(data, title=title)

    return fig
```

Climate Change

Application Structure

Import Libraries
Define Layout Style
Define Variables as str
Define input and outputs

```python
import cdstoolbox as ct

layout = {
    'output_align': 'bottom'
}

variables = {
    'Near-Surface Air Temperature': '2m_temperature',
    'Eastward Near-Surface Wind': '10m_u_component_of_wind',
    'Westward Near-Surface Wind': '10m_v_component_of_wind',
    'Sea Level Pressure': 'mean_sea_level_pressure',
    'Sea Surface Temperature': 'sea_surface_temperature',
}


@ct.application(title='Plot Map', layout=layout)
@ct.input.dropdown('variable', label='Variable', values=variables.keys())
@ct.output.figure()
def plot_map(variable):
    """
    Application main steps:

    - set the application layout with output at the bottom
    - select a variable name from a list in the dropdown menu
    - retrieve the selected variable
    - compose a title
    - show the result on a map using the chosen title

    """

    data = ct.catalogue.retrieve(
        'reanalysis-era5-single-levels',
        {
            'variable': variables[variable],
            'product_type': 'reanalysis',
            'year': '2010',
            'month': '08',
            'day': '15',
            'time': '12:00',
        }
    )

    title = '{}'.format(' '.join([text.capitalize() for text in variable.split('_')]))
    fig = ct.cdsplot.geomap(data, title=title)

    return fig
```

Climate Change

Application Structure

Import Libraries
Define Layout Style
Define Variables as str
Define input and outputs
Define Computation steps

```
import cdstoolbox as ct

layout = {
    'output_align': 'bottom'
}

variables = {
    'Near-Surface Air Temperature': '2m_temperature',
    'Eastward Near-Surface Wind': '10m_u_component_of_wind',
    'Westward Near-Surface Wind': '10m_v_component_of_wind',
    'Sea Level Pressure': 'mean_sea_level_pressure',
    'Sea Surface Temperature': 'sea_surface_temperature',
}


@ct.application(title='Plot Map', layout=layout)
@ct.input.dropdown('variable', label='Variable', values=variables.keys())
@ct.output.figure()
def plot_map(variable):
    """
    Application main steps:

    - set the application layout with output at the bottom
    - select a variable name from a list in the dropdown menu
    - retrieve the selected variable
    - compose a title
    - show the result on a map using the chosen title

    """

    data = ct.catalogue.retrieve(
        'reanalysis-era5-single-levels',
        {
            'variable': variables[variable],
            'product_type': 'reanalysis',
            'year': '2010',
            'month': '08',
            'day': '15',
            'time': '12:00',
        }
    )

    title = '{}'.format(' '.join([text.capitalize() for text in variable.split('_')]))
    fig = ct.cdsplot.geomap(data, title=title)

    return fig
```

Climate Change

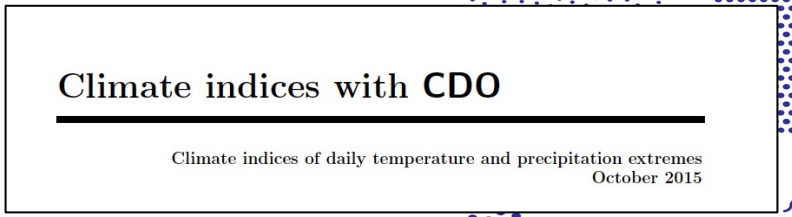
Let’s visit the Toolbox

https://cds.climate.copernicus.eu/cdsapp#!/toolbox

ECMWF Copernicus Europe's eyes on Earth European Commission

Summary & comments

- Many (other) ways to visualise climate data
  - GIS-applications
  - CDO
  - Python (and other)
  - Excel
- Point data vs. gridded data
- More and more “user-friendly” online tools
  - Very relevant for non-climate-expert users



ILMATIETEEN LAITOS
METEOROLOGISKA INSTITUTET
FINNISH METEOROLOGICAL INSTITUTE

Thank You!