Guidelines for writing and plotting

Team Climate Dynamics and Modeling

September 19, 2024

Writing guidelines

A scientific text is not a school essay. Most rules for writing that you learned in school do not apply to scientific writing.

General concepts:

- Keep it short and simple: This applies on different levels. For the whole thesis, for paragraphs, and for sentences. Cut fluffy, flowery language that does not carry actual information; this increases the signal-to-noise ratio of your writing and hence the chances to successfully communicate to the reader. Shorter sentences are easier to read and understand. Be specific and concrete.
- Create a coherent storyline: Your text needs to have a "story" and follow a "Roter Faden". This means that the different sections are building on each other. Consider putting parts of the text in the appendix or supplement if it doesn't fit the main text but still is important.
- Have a clear structure. The classical structure of scientific texts is: Introduction - Methods - Results - Discussion.
- Write and revise: Writing is an iterative process. Do not expect the first draft to be perfect. Instead of trying to write perfect sentences right away, start with writing the most important points down. Later, you can reread the document, identify and fill gaps, add details and improve the writing style in multiple revisions. This helps to overcome writer's block and increases the quality of writing.
- One message per paragraph: this prevents the text from getting too convoluted. Information is best presented in digestible bites.
- The better the draft you hand to your advisors, the better the feedback you will receive. It is fine and in fact encouraged to use AI tools to correct typos and improve grammar and sentence structure.

- Only put the text in the final layout relatively late. When you use the final layout right at the beginning, you might be tricked by the visual beauty of your text and it is harder to focus on its actual content.

Checklist:

Make sure that:

- ☐ All acronyms are defined when they appear in the text for the first time. Use acronyms only for words that are used repeatedly in the text.
- ☐ One word is used for one term. Don't use synonyms when you are talking about the same thing. This creates ambiguity.
- ☐ Tables have headers above and figures have captions below.
- ☐ You use simple English and avoid jargon.
- ☐ You use an active voice.
- ☐ You describe your own work in present tense, previous work in past tense.
- ☐ You write conclusions in past tense.
- ☐ You are consistent with spelling conventions (American vs British English).
- ☐ You are consistent with the Bibliography and citation style.
- ☐ All figures and tables are referenced and described in order.
- ☐ There is no list of figures and tables (unless explicitly required).
- ☐ There are no broken links or broken citations.
- ☐ The figure caption is for description of the technical figure content. The scientific interpretation is given in the main text.
- ☐ You describe your methodology completely and explain why you are using it.

You can deviate from the above recommendations, but you need a reason to do so.

Further literature

Schimel, J., 2012: Writing science: how to write papers that get cited and proposals that get funded. Oxford University Press. Available online from the University Library

Plotting guidelines

Plots are one of the most important parts of a scientific text and presentation. A well crafted plot can make it much easier to get your message across, so it pays off to think deeply about plotting.

General concepts:

- A plot has to have a purpose. It should be developed iteratively as your understanding evolves. The purpose defines the type of plot (e.g. line, colormap, scatter, …).
- A plot communicates results, not that you have been productive. Don't create plots just to prove that you have done work.
- Default options from plotting software, such as python's matplotlib, are rarely optimal. Take time to adjust the plot for style, and unify the style across all plots of your work. An example is given below, including the python code. An example is given below.
- Maximize the data-to-ink ratio. Remove unnessary components such as grids, boxes around plots, boxes around legends etc. Apply Occam's razor to decide about the content of your plot - if a component of the plot is not necessary to convey the message of the plot, leave it out.
- Plots should be easily reproducible by yourself. They should be fully generated by scripts. Avoid using image editing software, such as inkscape or paint.
- Not all plots you make need to be perfect. There is a difference between the plots you produce to develop your understanding (these do not need to be polished) and the final plots that will go in your thesis/project/presentation (these need to be polished).
- The final polishing of the plot should happen when your message is settled (this mean at a late point in the science work). Don't waste time perfecting a plot that might change substantially with your understanding of its content.

Checklist:

Make sure that:

- ☐ You have a uniform style across all plots.
- ☐ Each plot has legends, axis- and tick labels, and correct units.
- ☐ No plot has a grid, a title, or a box around the legend.
- ☐ You generally use (6,4) figure size, a font size of 12 for axis labels, and a font size of 10 for tick labels.
- ☐ For subplots, information is shared as much as possible between subplots; e.g. use only one legend and use shared axis labels.
- ☐ Subplots have labels (a,b,c…).
- ☐ For colour maps:
  - You DON'T use the jet or rainbow colormap.
  - You use diverging colormaps only for anomalies, with zero as white.
  - You use only one colour map per variable.
  - You consider colour-blind friendly options (there's more than standard matplotlib colour map).

- ☐ You think about the axes intersection. Axes typically should only intersect at (0,0); in other cases you can use broken axes.

You can deviate from these rules, but be sure that you have a reason for doing so.

List of further literature

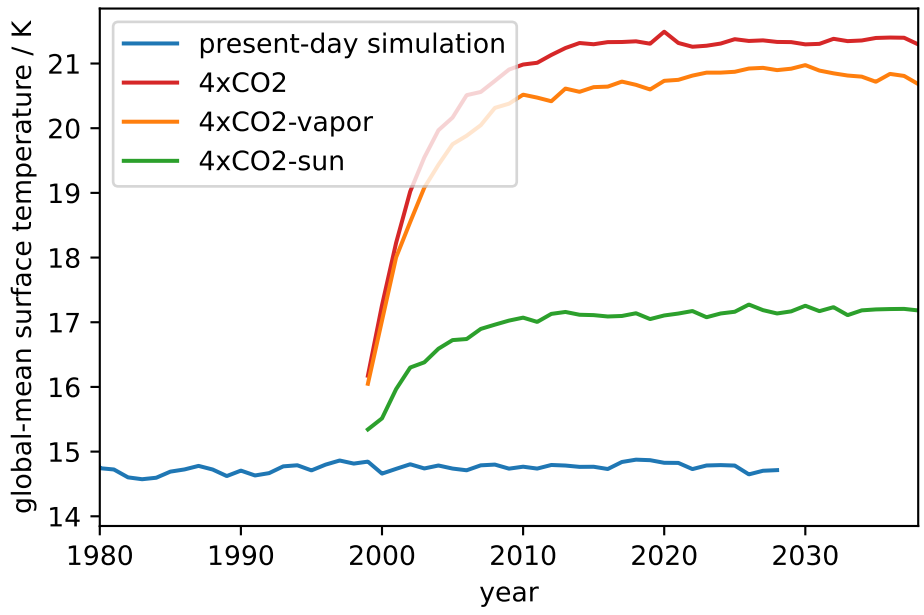
- IMG presentation guide
- https://www.principiae.be/
- Bergstrom, C. T., & West, J. D. (2021). Calling bullshit : the art of skepticism in a data-driven world. [London]: Penguin Books.

Example plot: default options of matplotlib versus manual adjustments

This is a version using the default settings of matplotlib.

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

def load_with_correct_timeaxis(fname, startyear):
    """
    Loads data from file fname and manually sets time axis.

    Input:
     - fname: input file
     - startyear: first year in data, we assume yearly data
    Output:
     - xarray dataset
    """
    _ds = xr.open_dataset(fname).squeeze()
    _ds["time"] = np.arange(startyear, startyear+_ds.time.size)
    return _ds

slabctr = load_with_correct_timeaxis(fname="./slabctr_ts.nc"   ,
 ↪  startyear=1979)
slab4x  = load_with_correct_timeaxis(fname="./slab4x_ts.nc"    ,
 ↪  startyear=1999)
slabvap = load_with_correct_timeaxis(fname="./slab4x-vap_ts.nc",
 ↪  startyear=1999)
```

```
slabsun = load_with_correct_timeaxis(fname="./slab4x-sun_ts.nc",
 ↪  startyear=1999)

plt.figure()
plt.plot(slabctr.time, slabctr["ts"] - 273.15, color="tab:blue"  ,
 ↪  label="present-day simulation")
plt.plot(slab4x.time , slab4x["ts"]  - 273.15, color="tab:red"   ,
 ↪  label="4xCO2")
plt.plot(slabvap.time, slabvap["ts"] - 273.15, color="tab:orange",
 ↪  label="4xCO2-vapor")
plt.plot(slabsun.time, slabsun["ts"] - 273.15, color="tab:green" ,
 ↪  label="4xCO2-sun")
plt.ylim(287 - 273.15,295 - 273.15)
plt.xlim(1980,2038)

plt.xlabel("year")
plt.ylabel("global-mean surface temperature / K")

plt.legend()
```



This is a version with adjusted axes, legends, font size etc.

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

def load_with_correct_timeaxis(fname, startyear):
    """
    Loads data from file fname and manually sets time axis.

    Input:
     - fname: input file
     - startyear: first year in data, we assume yearly data
    Output:
     - xarray dataset
    """
    _ds = xr.open_dataset(fname).squeeze()
    _ds["time"] = np.arange(startyear, startyear+_ds.time.size)
    return _ds

def beautify_timeseries(ax, yaxis0):
    """
    Makes plots of time series nicer in terms of axes, labeling etc.
    """
    # adjust spines
    ax = plt.gca()
    ax.spines['top'].set_color('none')
    ax.spines['right'].set_color('none')
    ax.xaxis.set_ticks_position('bottom')
    ax.spines['bottom'].set_position(('data',yaxis0))


slabctr = load_with_correct_timeaxis(fname="./slabctr_ts.nc"   ,
 ↪  startyear=1979)
slab4x  = load_with_correct_timeaxis(fname="./slab4x_ts.nc"    ,
 ↪  startyear=1999)
slabvap = load_with_correct_timeaxis(fname="./slab4x-vap_ts.nc",
 ↪  startyear=1999)
slabsun = load_with_correct_timeaxis(fname="./slab4x-sun_ts.nc",
 ↪  startyear=1999)

plt.figure(figsize=(6,4))
ax=plt.subplot(1,1,1)
```

```
plt.plot(slabctr.time, slabctr["ts"] - 273.15, color="tab:blue")
plt.plot(slab4x.time , slab4x["ts"]  - 273.15, color="tab:red")
plt.plot(slabvap.time, slabvap["ts"] - 273.15, color="tab:orange")
plt.plot(slabsun.time, slabsun["ts"] - 273.15, color="tab:green")
plt.ylim(287 - 273.15,295 - 273.15)
plt.xlim(1980,2038)

beautify_timeseries(ax, yaxis0=286.5 - 273.15)

plt.xlabel("year", loc="right", size=12)
plt.ylabel("global-mean surface temperature / K", loc="top", size=12)

plt.text(2038,288.2 - 273.15, "present-day simulation: 14.8 deg C",
 ↪  ha="right", color="tab:blue", size=12);
plt.text(2038,294.7 - 273.15, "4xCO2: +6.5 deg C", ha="right",
 ↪  color="tab:red", size=12);
plt.text(2038,293.2 - 273.15, "4xCO2-vapor: +6.1 deg C", ha="right",
 ↪  color="tab:orange", size=12);
plt.text(2038,290.6 - 273.15, "4xCO2-sun: +2.4 deg C", ha="right",
 ↪  color="tab:green", size=12);
```