Figure 10. Data assimilation results. In panel (a) “no assimilation” refers to the RMSE of the initial forecast (the average of the initial forecast ensemble) propagated forward in time over the 100 cycles without assimilating observations into it. The rank histogram of where the truth ranks among analysis ensemble members is shown in panel (b). The ranks are evaluated for every $13^{th}$ variable in the state vector (past the correlation bound) after 100 assimilation cycles.

Upon termination of a run in DATeS, executable files can be cleaned up by calling the function `clean_executable_files()` from the utility module:

```python
# cleanup executables and temporary modules
import dates_utility as utility
utility.clean_executable_files()
```

Figure 11. Cleanup DATeS executable files.

5 Extending DATeS

DATeS aims at being a collaborative environment, and is designed such that adding DA components to the package is as easy and flexible as possible. This section describes how new implementations of components such as numerical models and assimilation methodologies can be added to DATeS.

The most direct approach is to write the new implementation completely in Python. This, however, may sacrifice efficiency, or may not be feasible when existing code in other languages needs to be reused. One of the main characteristics of DATeS is the possibility of incorporating code written in low level languages. There are several strategies that can be followed to interface existing C or Fortran code with DATeS. Amongst the most popular tools are SWIG, and F2Py for interfacing Python code with existing implementations written in C and Fortran, respectively.

Whether the new contribution is written in Python, in C, or in Fortran, an appropriate Python class that inherits the corresponding base class, or a class derived from it, has to be created. The goal is to design new classes those are conformable with the existing structure of DATeS and can interact appropriately with new as well as existing components.