SouthGreen bioinformatics platform

PACKAGES AND ENVIRONMENTS MANAGEMENT WITH

Marcel Martin
https://nbisweden.github.io/workshop-conda/

Jacques Dainat Ph.D.

REPRODUCTIBILITY CRISIS

Dependencies hell (scripts, tools, DB, ref etc)

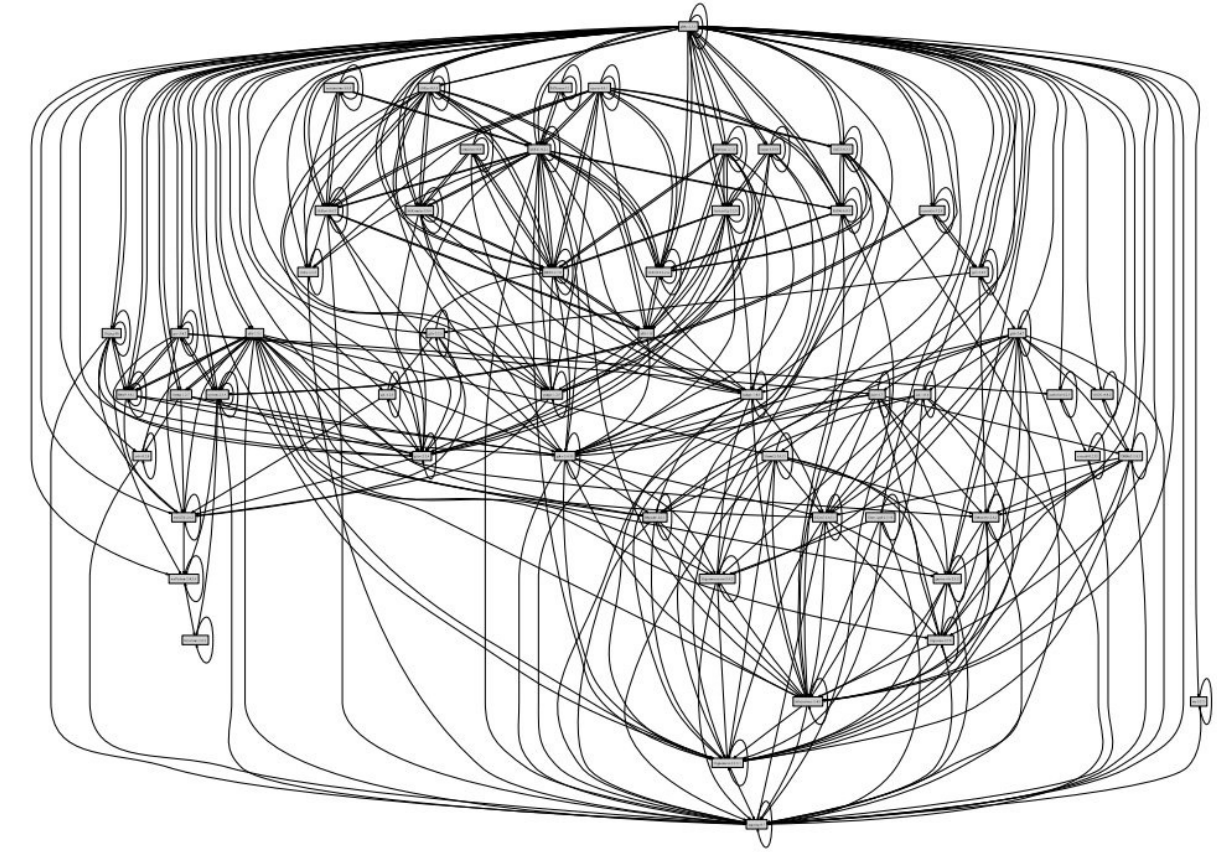
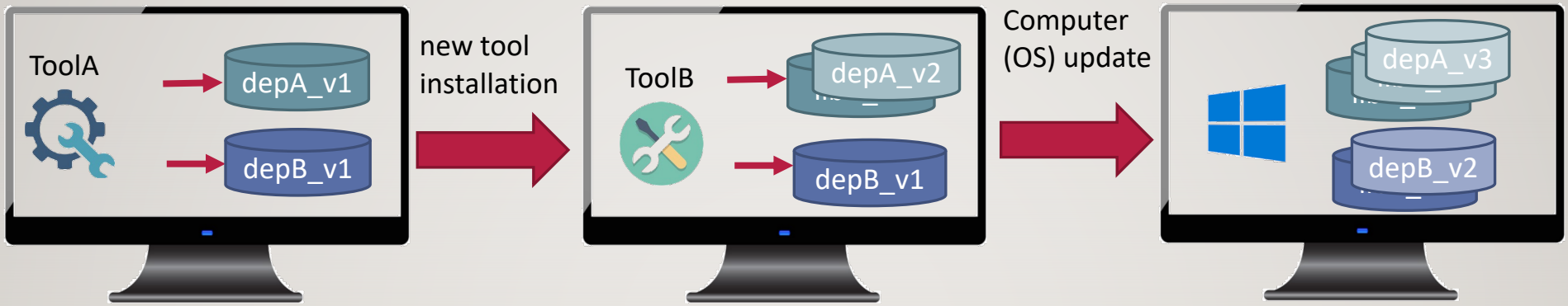


Figure 1.5.: The *dependency hell*: the runtime dependency graph of Mozilla Firefox

The purely functional software deployment model / Eelco Dolstra - [S.l.] : [s.n.], 2006 - Tekst. - Proefschrift Universiteit Utrecht

Dependencies hell

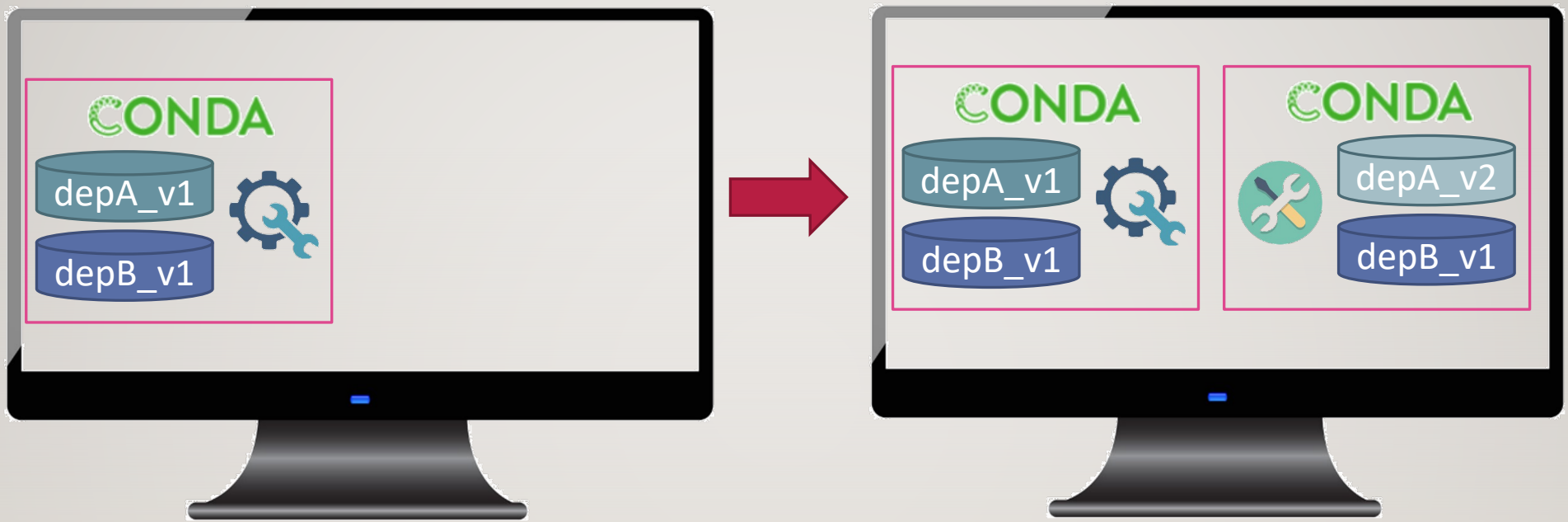


NOW

- Tools A and/or B may provide different results or stop to work
- A new Tools C cannot be installed due to compilation requirements competing with another tool

Conda dependency manager

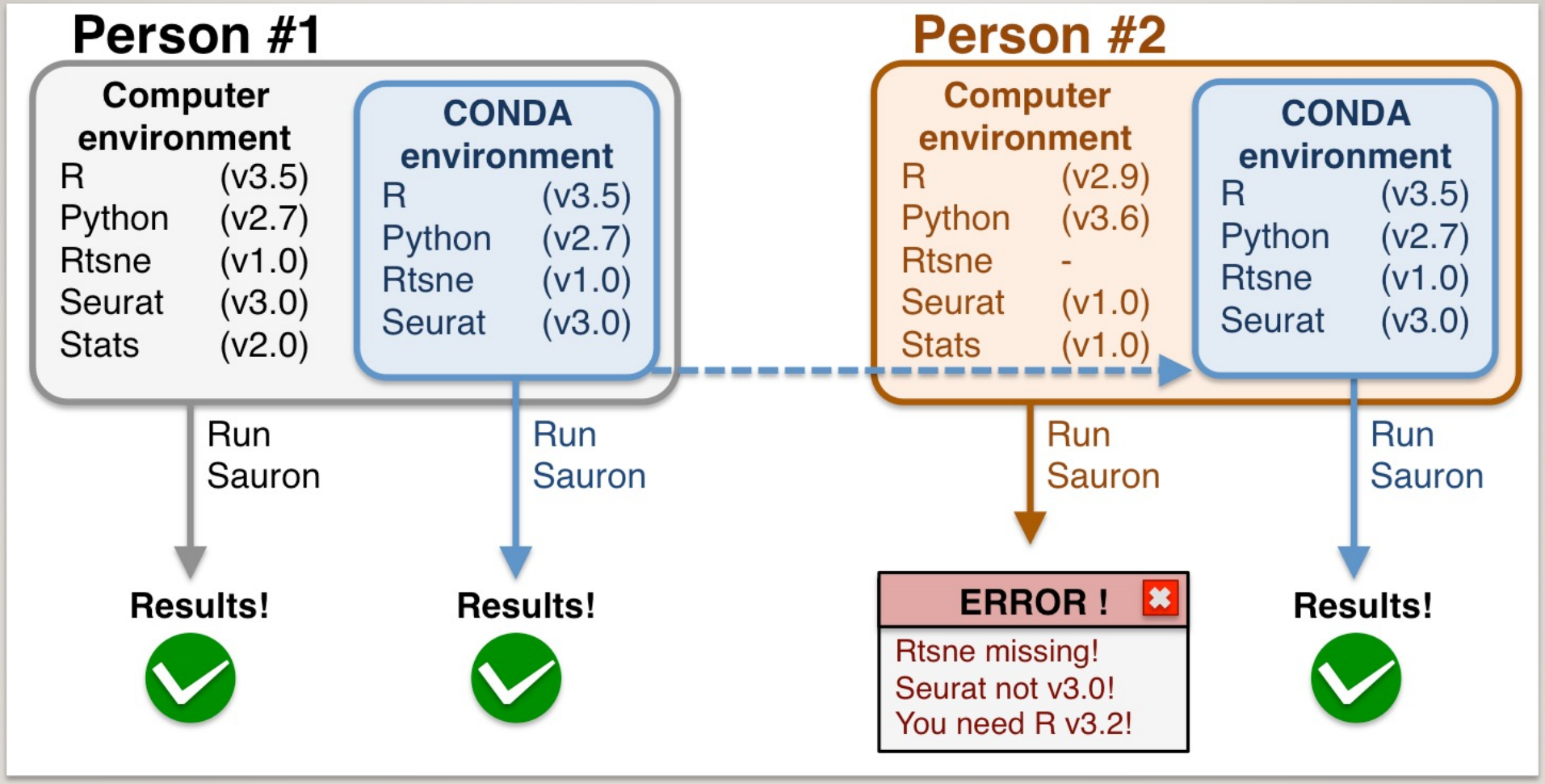
Each environment will have its own dependencies



Conda strength: - tool precompiled
- isolated environments with defined dep/software versions

Drawback: - conda environment can be heavy

Conda dependency manager



https://nbisweden.github.io/excelerate-scRNAseq/conda_instructions.html

Conda: a **package**, **dependency**, and **environment manager**

- Like `apt`, `brew`, `pip`, `yum`, etc., but with focus on data science
- Supports Linux, macOS, Windows
- Packages come pre-compiled
- On Linux, they work on most distributions
- Packages are hosted centrally at anaconda.org
- Users can contribute their own packages via ***channels*** (think YouTube)
- Most important for us :
  - The defaults channel (11 270 packages)
  - The conda-forge channel (21 455 packages)
  - The Bioconda channel (9894+ packages)

] Both are community-driven

CONDA terms

- Conda: The package manager
- conda: The command-line program
- Anaconda, Inc: The company (previously Continuum Analytics, Inc)
- Anaconda: A distribution of many data-science packages managed by conda
- miniconda: A much smaller distribution that only contains conda
- Mamba: A faster, drop-in replacement for conda
- Bioconda: A bioinformatics-focused channel for Conda packages
- conda-forge: A community-driven channel for everything else

Using Conda on your laptop

- Download Miniconda and install it:

```
$ curl -sO https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
$ bash Miniconda3-latest-Linux-x86_64.sh
Welcome to Miniconda3 ...
[...]
```

- Set up the Bioconda channel:

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
conda config --set channel_priority strict
```

Conda is sometimes slow

• Conda installs a package and all required dependencies, observing constraints on the version.

Example:

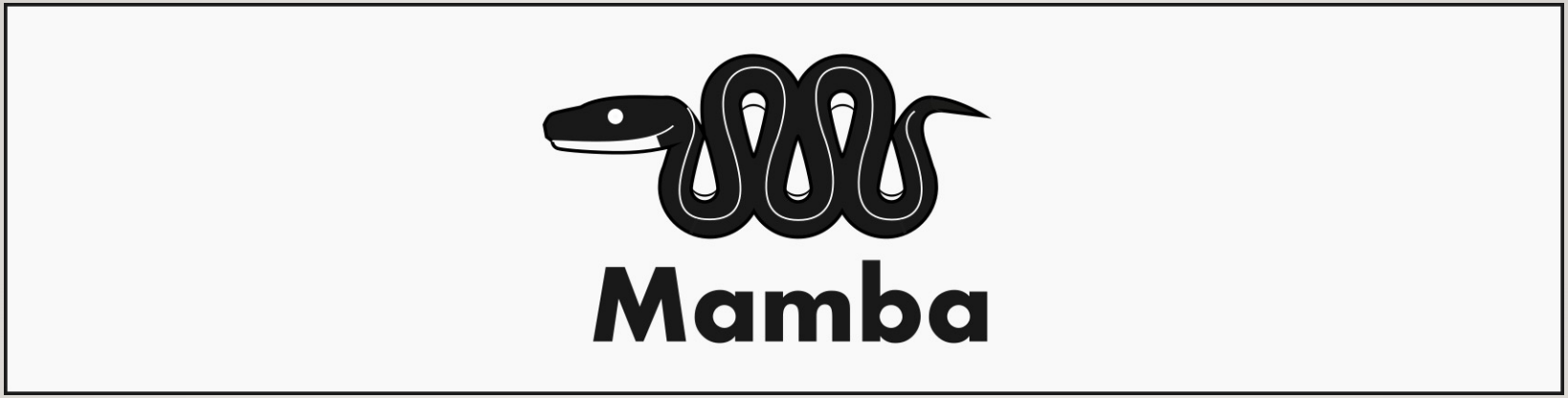
- A workflow needs tools `x` and `y`.
- `x` requires snakemake
- Recent `snakemake` versions require `python>=3.7`
- Tool `y` requires `python<=3.6`

→ The dependency resolver needs to find an older `snakemake` version still compatible with Python 3.6.

• Solving which package versions satisfy all requirements is an NP-complete problem

• For channels with many packages (conda-forge) and/or packages with many dependencies, dependency resolution can take ***hours***.

Mamba to the rescue



Mamba is a replacement for conda with a *much* faster dependency resolver

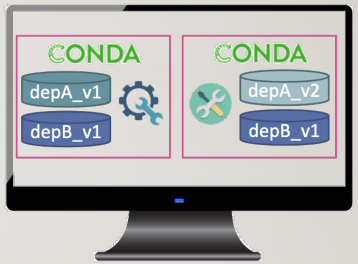
- Same command-line interface as conda (in most cases)
- Install it (from conda-forge):

```
conda install -n base mamba
```

Conda environments

Conda environment basics

- Anything installed by Conda is put into a *Conda environment*.
  - A single environment contains a consistent set of packages (compatible with each other)
  - Different environments are independent of each other



- The default environment is called base
  - It contains Conda itself and its dependencies such as Python

- The base environment is the only one that cannot easily be deleted and re-created, so keep it clean
  - Best to use it only for Conda-related tools (*mamba*, *conda-build* etc.)

- For anything else, create a new environment

Working with environments

- Example: Create an environment named mapping and install *Samtools* and *Bowtie2* into it:

```
conda create -n mapping samtools bowtie2
```

- To use the software you must activate the environment:

```
$ conda activate mapping
$ samtools --version
samtools 1.15.1
```

- To install a package into an existing environment:

```
conda install -n mapping bwa=0.7.17
```

- To install into the currently active environment:

```
conda install bwa=0.7.17
```

- You can use `=`, `>=`, `<=` to constrain package versions
- Find packages by searching anaconda.org or with `conda search`

Treat Conda environments as ephemeral

- To test a new tool, install it into a fresh Conda environment. Then delete the environment to uninstall.

```
conda env remove -n mapping
```

- To find out when the bug you're seeing in a tool was introduced, install older versions into a temporary environment.

- If your project's environment got messed up, just delete it and start over.

- Environments are just directories within your Conda installation directory
  - `~/miniconda/envs/mapping` or similar
  - Activating an environment adds its `bin/` directory to your `$PATH`

Persisting and sharing environments

- **Environment files** specify how to create an environment.
- Example `environment.yaml`:

```
name: bwa
channels:
- conda-forge
- bioconda
- defaults
dependencies:
- bwa=0.7.17
```

- To create the environment, run:

```
mamba env create -f environment.yaml # Note: "env create", not "create"
```

- The YAML file can be written manually or be generated from an existing environment:

```
conda env export --no-builds [--from-history] > environment.yaml
```

  - --no-builds is recommended in the Bioconda FAQ

Categories of environment files

Environment files can be **abstract** or **concrete** (or in between).

Abstract dependencies: List only `pysam` and Conda will pick a suitable version for it and its dependencies.

- Good at development time and when you develop something that needs to be installable alongside other tools.
- Not reproducible. If a dependency is updated, your tool or workflow may produce different results.

Concrete dependencies: List `pysam=0.19.1` and ***all dependencies and their versions***.

- This is what `conda env export` creates.
- Reproducible.
- Very unflexible. Installing another tool into the environment likely leads to conflicts.
- Probably platform-specific. You need one concrete `environment.yaml` for each platform you support.

Abstract or concrete?

For some projects, using both abstract ***and*** concrete makes sense:

- The abstract dependencies define what your software depends on
- The concrete dependencies (one file for each platform) are used for your tests (CI)
- The concrete environments are generated from the abstract one.

More ways to create reproducible environment specifications

```
$ conda list --export
# This file may be used to create an environment using:
# $ conda create --name <env> --file <this file>
# platform: linux-64
...
bwidget=1.9.14=ha770c72_1
bzip2=1.0.8=h7f98852_4
...
```

```
$ conda list --explicit
...
@EXPLICIT
...
https://conda.anaconda.org/conda-forge/linux-64/bwidget-1.9.14-ha770c72_1.tar.bz2
https://conda.anaconda.org/conda-forge/noarch/bzip2-1.0.8-h7f98852_4.tar.bz2
...
```

--explicit includes channels!

- Exactly reproduces environments (on one platform)
- It is very fast because Conda no longer resolves dependencies
- Recreating the environment can fail when files are removed from anaconda.org

A last word

Conda is designed to make your life as a bioinformatician easier!

Conda works well with workflow managers!