

# Auswertungen automatisieren mit make

## Automatisierte, reproduzierbare Prozesse

### Problem:

- kurz vor Abgabe noch neue Korrekturen einpflegen
  - Tippfehler korrigieren, Plots bearbeiten
- $\text{\TeX}$  ausführen, ausdrucken
- vergessen, Plots neu zu erstellen

## Automatisierte, reproduzierbare Prozesse

### Lösung: Make!

- sieht, welche Dateien geändert wurden
- berechnet nötige Operationen
- führt Python-Skript aus, führt  $\text{T}_{\text{E}}\text{X}$  aus

## Makefile

- Datei heißt Makefile, keine Endung!
  - bei Windows Dateiendungen einschalten!  
(<http://support.microsoft.com/kb/865219/de>)
- besteht aus Rules:

### Rule

```
target: prerequisites  
    recipe
```

**target** Datei(en), die von dieser Rule erzeugt wird

**prerequisites** Dateien, von denen diese Rule abhängt

**recipe** Befehle: prerequisites → target (mit Tab eingerückt)

## Beispiel

```
all: report.pdf
```

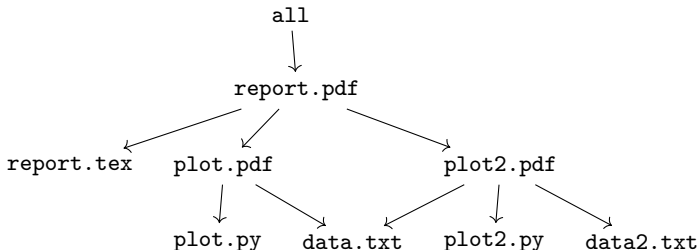
```
plot.pdf: plot.py data.txt  
python plot.py
```

```
report.pdf: report.tex  
lualatex report.tex
```

```
report.pdf: plot.pdf
```

- wenn nur `make` gestartet wird, wird der erste Target erstellt, hier `all`; um ein anderes Target zu erstellen, startet man `make target`
  - nützlich, wenn man Plot bearbeitet: `make plot.pdf`
- `all` braucht `report.pdf`, also wird `latex report.tex` ausgeführt
- `report.pdf` braucht noch `plot.pdf`, also wird Python ausgeführt

## Funktionsweise



- Abhängigkeiten bilden einen DAG (directed acyclic graph)
- Dateien werden neu erstellt, falls sie nicht existieren oder älter als ihre Prerequisites sind
- Prerequisites werden zuerst erstellt
- top-down Vorgehen

## Advanced

- make clean

```
clean:
```

```
# alles löschen, was vom Makefile erzeugt wird  
rm plot.pdf report.pdf
```

- build-Ordner: Ordner sauber halten

```
build/plot.pdf: plot.py data.txt | build  
python plot.py # savefig('build/plot.pdf')
```

```
build/report.pdf: report.tex build/plot.pdf | build  
lualatex --output-directory=build report.tex
```

```
build:
```

```
mkdir -p build
```

```
clean:
```

```
rm -rf build
```

- | build ist ein order-only Prerequisite: Alter wird ignoriert

## Expert

Mehrere unabhängige Auswertungen: könnte man sie parallel ausführen?

Ja! `make -j4` (4 Prozesse gleichzeitig)

```
plot1.pdf plot2.pdf: plot.py data.txt
python plot.py
```

Problem: `make` führt `plot.py` gleichzeitig zweimal aus

Lösung: manuell synchronisieren

```
plot1.pdf plot2.pdf: plot.sync
@: # nichts tun
```

```
plot.sync: plot.py data.txt
touch plot.sync
python plot.py
```



## Motivation

- Automatisierung verhindert Fehler
- Dient als Dokumentation
- Reproduzierbarkeit unverzichtbar in der Wissenschaft
- Idealfall: Eingabe von `make` erstellt komplettes Protokoll/Paper aus Daten
- Spart Zeit, da nur nötige Operationen ausgeführt werden