

CHAP-DEVOPS

# Evaluating forecasting models

A hands-on workshop with Chap and chap-devops

---

From an empty laptop to a comparison of three models, in about one hour

# What we do today

---

1. Install chap-devops, and check the machine.
2. Get a dataset: dengue cases and climate, for districts of Laos.
3. Learn what a **backtest** is.
4. Evaluate a model with a backtest, and plot it.
5. Evaluate two more models, and compare the three.
6. Change the backtest, and see what changes.

You need a laptop with **Linux or macOS** (or Windows with WSL), **Docker**, and about **25 GB** of free disk space.

# Before the workshop: the downloads

**The images are large, and a room of laptops on one network downloads them slowly.** Download them at home, the day before.

| Image                          | Size on disk   | Used for                           |
|--------------------------------|----------------|------------------------------------|
| chap-worker                    | about 12 GB    | the model from GitHub              |
| chap-core                      | about 2.4 GB   | plots, metrics, the model services |
| auto_arima_chapkit             | about 6.2 GB   | the second model                   |
| chapkit_simple_multistep_model | less than 1 GB | the third model                    |

```
docker pull --platform linux/amd64 ghcr.io/dhis2-chap/chap-worker:v2.5.0
docker pull --platform linux/amd64 ghcr.io/dhis2-chap/chap-core:v2.5.0
chap-devops run auto_arima_chapkit && chap-devops stop auto_arima_chapkit
chap-devops run chapkit_simple_multistep_model && chap-devops stop chapkit_simple_multistep_model
```

# The plan for the hour

---

| Minutes  | Step                                     | While it runs                         |
|----------|------------------------------------------|---------------------------------------|
| 0 to 5   | Install, <code>chap-devops doctor</code> | Explain what Chap and chap-devops are |
| 5 to 10  | Get the data, exercise 1                 | Read the CSV together                 |
| 10 to 20 | What a backtest is                       | No command runs                       |
| 20 to 30 | The first backtest (step 3)              | Explain splits and periods            |
| 30 to 35 | Plot it, exercise 2                      | Read the plot together                |
| 35 to 45 | Two more models (step 5)                 | Explain the metrics                   |
| 45 to 55 | Compare, exercises 3 and 4               | Discuss the answers                   |
| 55 to 60 | The quiz, and next steps                 |                                       |

Start each long command first, and explain while it runs.

# Words we use

---

| Word               | What it means                                                                    |
|--------------------|----------------------------------------------------------------------------------|
| <b>Chap</b>        | The platform that trains forecasting models and makes forecasts of disease cases |
| <b>model</b>       | A program that learns from past cases and climate, and forecasts cases           |
| <b>backtest</b>    | Forecasts made from points in the past, compared with what happened              |
| <b>split</b>       | One point in the past from which the model trains and forecasts                  |
| <b>period</b>      | One time step of the data, here one month                                        |
| <b>horizon</b>     | How many periods ahead a forecast goes                                           |
| <b>chap-devops</b> | The tool that runs Chap and its models for us, with Docker                       |

## Step 1: Install and check

---

```
curl -fsSL https://raw.githubusercontent.com/winterop-com/chap-devops/main/install.sh | sh
chap-devops --version
chap-devops doctor
```

**It worked when** `chap-devops doctor` starts the `docker cli`, `docker daemon` and `docker compose` lines with `ok`.

On an Apple silicon Mac, a `warn` on `os` and `arch` is correct: the images run under emulation, which is slower.

## Step 2: The data

---

```
mkdir -p ~/chap-workshop && cd ~/chap-workshop
curl -fsS0 https://raw.githubusercontent.com/dhis2-chap/chap-core/master/example_data/laos_subset.csv
curl -fsS0 https://raw.githubusercontent.com/dhis2-chap/chap-core/master/example_data/laos_subset.geojson
head -3 laos_subset.csv
```

```
time_period,rainfall,mean_temperature,disease_cases,population,location
2010-01,37.965,20.04,1.0,75049.5636160714,Bokeo
2010-02,8.527,22.22,1.0,75049.5636160714,Bokeo
```

# The data: what is in it

---

| Column                        | What it is                                       |
|-------------------------------|--------------------------------------------------|
| <code>time_period</code>      | The month, <code>YYYY-MM</code>                  |
| <code>location</code>         | The district                                     |
| <code>disease_cases</code>    | The dengue cases in that month: what we forecast |
| <code>rainfall</code>         | The rainfall in that month                       |
| <code>mean_temperature</code> | The mean temperature in that month               |
| <code>population</code>       | The population of the district                   |

- 3 districts: Bokeo, Savannakhet and Vientiane (prefecture).
- 36 months each: January 2010 to December 2012.
- The GeoJSON has the shape of each district. Its name is the same as the CSV's.

## Exercise 1: Look at the data

---

```
chap-devops chap validate laos_subset.csv
```

**It worked when** chap says `Validation passed: no issues found.`

1. Open the CSV in a spreadsheet. In which months are the most cases?
2. Do the months with many cases come after months with much rain?
3. Which district has the most people? Does it also have the most cases?

# What a backtest is

---

We want to know how good a model is at forecasting. We cannot wait for the future, so we **pretend that it is the past**:

1. Hide the last months of the data.
2. Let the model learn from the months before them.
3. Let it forecast the hidden months.
4. Compare the forecast with what really happened.

One time is not sufficient: a model can be lucky. So we do it from several points in the past. Each point is a **split**.

# A backtest, as a timeline

A sketch for `n_splits 3`, `n_periods 3`, `stride 1`. `T` is the last month of the data, and `chap` picks the exact months:

| Split | Trains on                         | Forecasts                                              |
|-------|-----------------------------------|--------------------------------------------------------|
| 1     | all months up to <code>T-5</code> | <code>T-4</code> , <code>T-3</code> , <code>T-2</code> |
| 2     | all months up to <code>T-4</code> | <code>T-3</code> , <code>T-2</code> , <code>T-1</code> |
| 3     | all months up to <code>T-3</code> | <code>T-2</code> , <code>T-1</code> , <code>T</code>   |

- **`n-periods`** : how far ahead each forecast goes (the horizon).
- **`n_splits`** : how many splits.
- **`stride`** : how many months between two splits.

The model never sees the months it forecasts.

# The backtest options

---

```
--backtest-params.n-periods    the forecast horizon          (default 3)
--backtest-params.n-splits     the number of train/test splits (default 7)
--backtest-params.stride       the step between two splits      (default 1)
--backtest-params.n-retrain    how many times the model trains again
```

- More splits: a fairer score, and a longer run.
- A longer horizon: a more difficult forecast.
- The splits must fit in the data: 36 months is not much.

## Step 3: The first backtest

---

```
chap-devops chap eval \  
  --model-name https://github.com/dhis2-chap/minimalist_example_r \  
  --dataset-csv laos_subset.csv \  
  --output-file out/minimalist_r.nc \  
  --backtest-params.n-splits 3 \  
  --backtest-params.n-periods 3
```

- `minimalist_example_r` is a small R model from chap-core's examples.
- `chap-devops` makes `out/` for you.

**It worked when** `chap-devops` writes `chap` finished; it wrote `out/minimalist_r.nc`.

# What happened

---

```
running `chap eval` in ghcr.io/dhis2-chap/chap-worker:v2.5.0
files: chap reads and writes in /home/me/chap-workshop
...
chap finished; it wrote out/minimalist_r.nc
`chap-devops chap plot-backtest out/minimalist_r.nc --output-file out/minimalist_r.html` plots it
```

- chap ran in a container, with R and the model's packages in it.
- The model came from GitHub. Its packages are now in a cache, so the next run is faster.
- The `.nc` file holds the forecasts and the real cases for each split.

## Step 4: Plot it

---

```
chap-devops chap plot-backtest out/minimalist_r.nc --output-file out/minimalist_r.html  
open out/minimalist_r.html          # macOS  
xdg-open out/minimalist_r.html      # Linux
```

The plot shows the real cases and the forecasts of each split. A forecast is a range, not one number: the model says how sure it is.

## Exercise 2: Read the plot

---

1. For which district is the forecast closest to the real cases?
2. Is the real value inside the forecast range most of the time?
3. Is the forecast better one month ahead than three months ahead?
4. Is a wide range a good or a bad sign? When can it be good?

## Step 5: Two more models, from the marketplace

---

```
chap-devops chap eval --model-name auto_arima_chapkit \  
  --dataset-csv laos_subset.csv --output-file out/auto_arima.nc \  
  --backtest-params.n-splits 3 --backtest-params.n-periods 3  
chap-devops chap --stop eval --model-name chapkit_simple_multistep_model \  
  --dataset-csv laos_subset.csv --output-file out/multistep.nc \  
  --backtest-params.n-splits 3 --backtest-params.n-periods 3
```

- These models are **services**: containers with an API.
- chap-devops starts each one for us, and tells chap where it answers.
- `--stop` stops the third model after its run.

## While the models run: three kinds of model

| Model                                       | How it forecasts                       | What it reads                                 |
|---------------------------------------------|----------------------------------------|-----------------------------------------------|
| <code>minimalist_example_r</code>           | A linear model ( <code>lm</code> in R) | rain and temperature                          |
| <code>auto_arima_chapkit</code>             | ARIMA, with automatic order selection  | the past cases only                           |
| <code>chapkit_simple_multistep_model</code> | A RandomForest, one month at a time    | the cases of earlier months; climate if given |

ARIMA reads no climate at all, so it is a good **baseline**: a model that uses climate must do better than it.

`chap-devops models list` lists every model in the marketplace.

## Step 6: Compare the three

---

```
chap-devops chap export-metrics out/minimalist_r.nc out/auto_arima.nc out/multistep.nc \  
--output-file out/comparison.csv
```

`comparison.csv` has one row for each model. chap can print `RuntimeWarning` lines about `log1p`. They do not stop the run.

# What the metrics mean

| Column         | What it measures                                                                | Better when |
|----------------|---------------------------------------------------------------------------------|-------------|
| mae            | The mean absolute error: on average, how many cases the forecast is off         | lower       |
| rmse           | The root mean square error: like <code>mae</code> , but large errors count more | lower       |
| crps           | How far the whole forecast range is from the real value                         | lower       |
| coverage_10_90 | How often the real value is inside the 10-90% range                             | near 0.8    |
| coverage_25_75 | The same, for the 25-75% range                                                  | near 0.5    |

A model with a low `mae` and a bad `coverage` is sure of itself, and wrong about how sure it is.

## Exercise 3: Which model is best?

---

1. Which model has the lowest `crps` ? The lowest `mae` ?
2. Is the same model best on both? Why can they differ?
3. Which model has a `coverage_10_90` nearest to 0.8? What does that tell you about its forecast range?
4. Would you use the best model for real decisions? What more would you want to know?

## Exercise 4: Change the backtest

---

```
chap-devops chap eval --model-name auto_arima_chapkit \  
  --dataset-csv laos_subset.csv --output-file out/arima_6.nc \  
  --backtest-params.n-splits 3 --backtest-params.n-periods 6  
chap-devops chap eval --model-name auto_arima_chapkit \  
  --dataset-csv laos_subset.csv --output-file out/arima_6splits.nc \  
  --backtest-params.n-splits 6 --backtest-params.n-periods 3  
chap-devops chap export-metrics out/auto_arima.nc out/arima_6.nc out/arima_6splits.nc \  
  --output-file out/backtests.csv
```

1. What happens to `mae` when the horizon is 6 months, not 3?
2. Do 6 splits give a different score from 3 splits? Which one do you trust more?

## Exercise 5: Your own data

---

Make a CSV with the same columns, and a GeoJSON with the same name:

```
mydata.csv      time_period,rainfall,mean_temperature,disease_cases,population,location
mydata.geojson  one feature for each location
```

```
chap-devops chap validate mydata.csv
chap-devops chap eval --model-name auto_arima_chapkit \
  --dataset-csv mydata.csv --output-file out/mine.nc
```

Each `location` in the CSV must be a feature in the GeoJSON.

# When a step fails

---

| You see                                                                              | What to do                                                                       |
|--------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| <code>docker daemon</code> is not <code>ok</code> in <code>chap-devops doctor</code> | Start Docker Desktop, or <code>sudo systemctl start docker</code>                |
| The first run waits a long time                                                      | It downloads an image. Wait, or use a laptop that has it                         |
| <code>the model server at ... is not running</code>                                  | Give the model id, not a URL                                                     |
| <code>did not answer ... within 300s</code>                                          | Run again with <code>--timeout 900</code>                                        |
| <code>chap exited with status 1</code>                                               | Read chap's lines above: often a typo in a file name                             |
| No disk space                                                                        | <code>docker image ls 'ghcr.io/dhis2-chap/*'</code> , remove what you do not use |

# Quiz

---

1. Why do we hide the last months in a backtest?
2. What does `--backtest-params.n-periods 6` change?
3. Two models have the same `mae`. One has `coverage_10_90` of 0.8, the other of 0.3. Which one do you prefer, and why?
4. Why does the second run of the R model start faster than the first?
5. Why must a model never see the months it forecasts?

# Clean up

---

```
chap-devops stop auto_arima_chapkit --purge    # the model service
rm -rf ~/.local/share/chap-devops/chap        # the caches
docker image ls 'ghcr.io/dhis2-chap/*'        # the images, to remove with docker image rm
```

Your `.nc`, `.html` and `.csv` files stay in `~/chap-workshop/out`.

## Where to go next

---

- **Your own data:** the same columns, a GeoJSON of the same name.
- **Your own model:** `chap-devops models new my-model` makes the project, and `chap-devops run` starts the model service.
- **DHIS2 and the Modeling App:** `chap-devops init mychap --with dhis2`, then `chap-devops dhis2 connect`.
- **More about the chap CLI in chap-devops:** the deck "The chap CLI, without Python".

The book: <https://winterop-com.github.io/chap-devops/>