Orchestrating Reproducible, Scalable, and Transparent NextGen and NextGen-Adjacent Workflows

CIROH Developers Conference 2026 — Workshop Session 1 of 2

Workshop Leads: Darri Eythorsson, Tristan Montoya, Jordan Read, James Halgren, Raymond Spiteri and Martyn Clark

May 28th, 2026

Session 1: Agenda

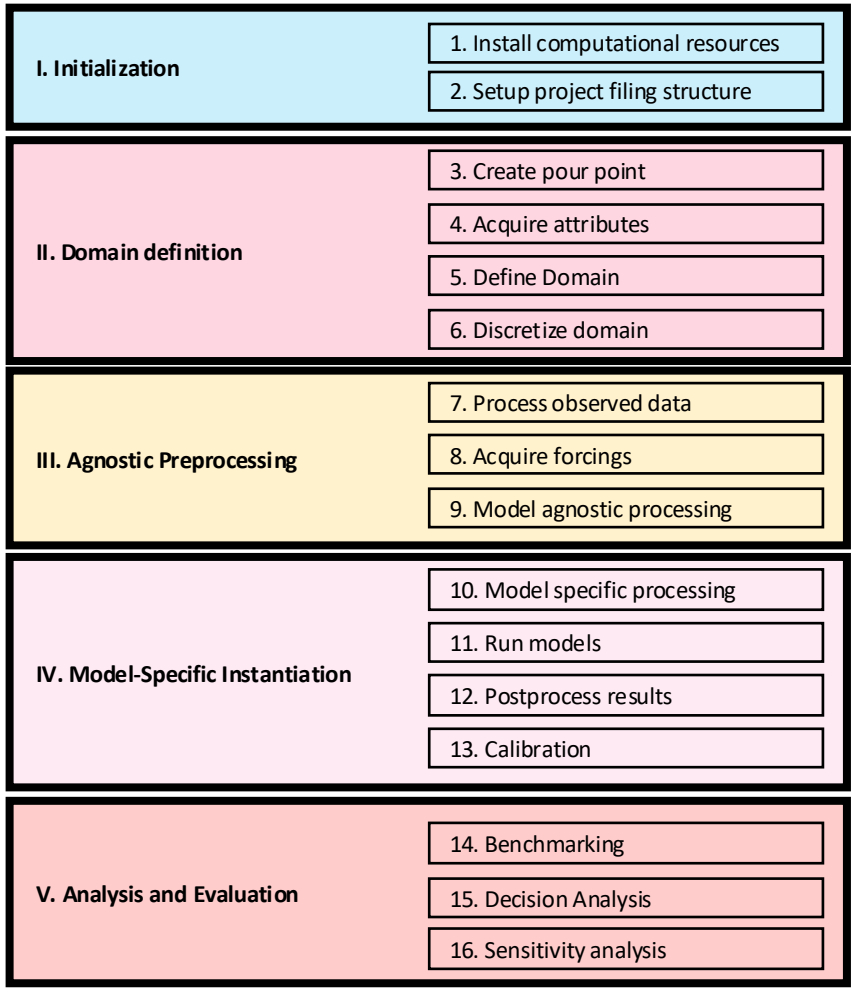
- **Part A — Context & Motivation**
  - The configuration problem
  - The uncertainty quantification problem
  - Connection to NextGen / NGIAB
- **Part B — Hands-On: Logan River jHBV Workflow**
  - Step 1: Configuration
  - Step 2: Domain definition & watershed delineation
  - Step 3: Forcing & observation data acquisition
  - Step 4: jHBV model execution
  - Step 5: Evaluation & calibration (DDS)
- **Part C — Recap and Wrap-Up**

Session 1: The Configuration Problem - Composability

Layers of composable components

Decision points

- Project initialization
- Domain spatial discretization
- Input data
- Model specific configuration
- Optimisation strategy
- Evaluation

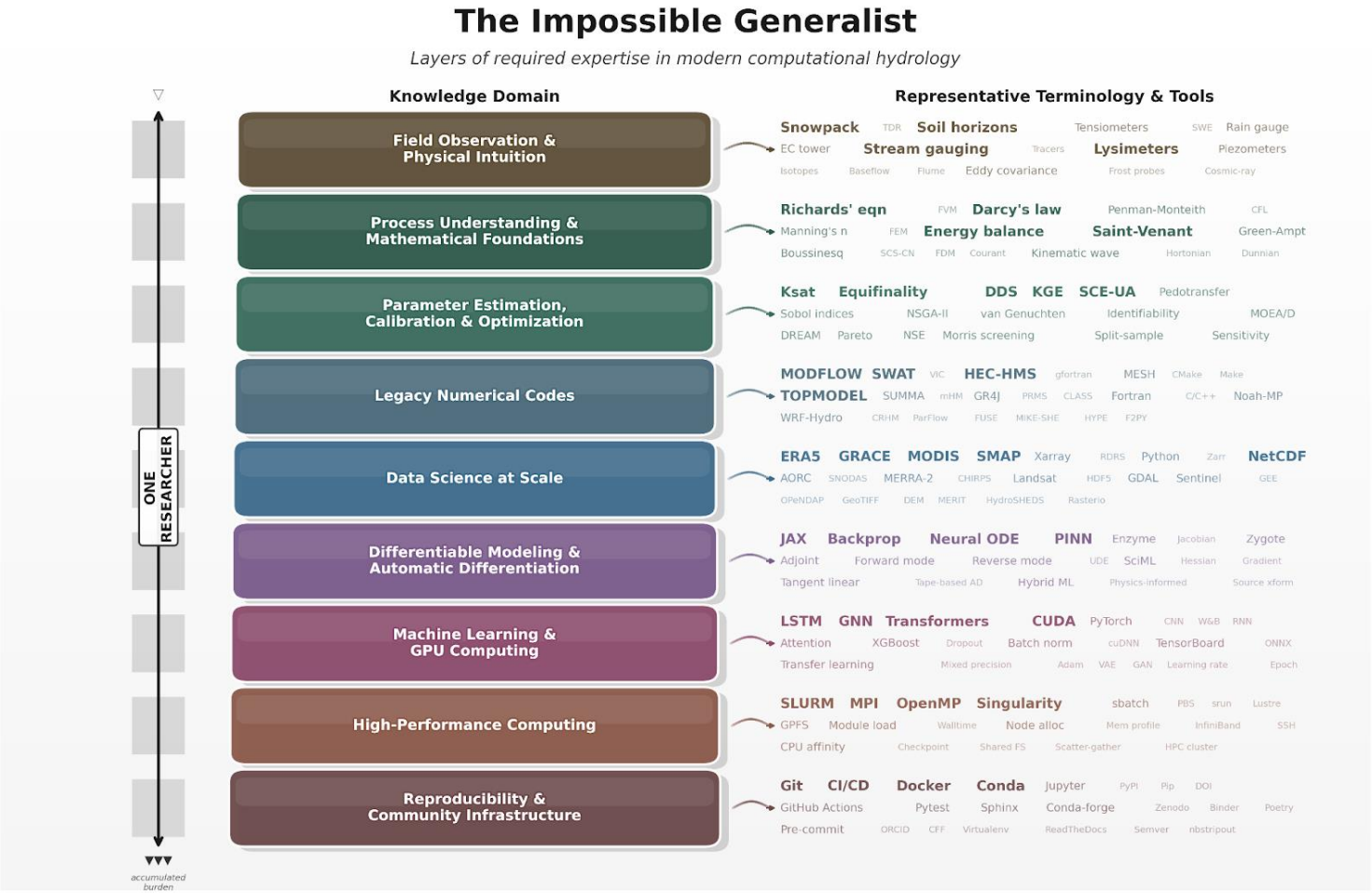


Session 1: The Configuration Problem - Technical breadth

Layers of composable components

Decision points

- Project initialization
- Domain spatial discretization
- Input data
- Model specific configuration
- Optimisation strategy
- Evaluation



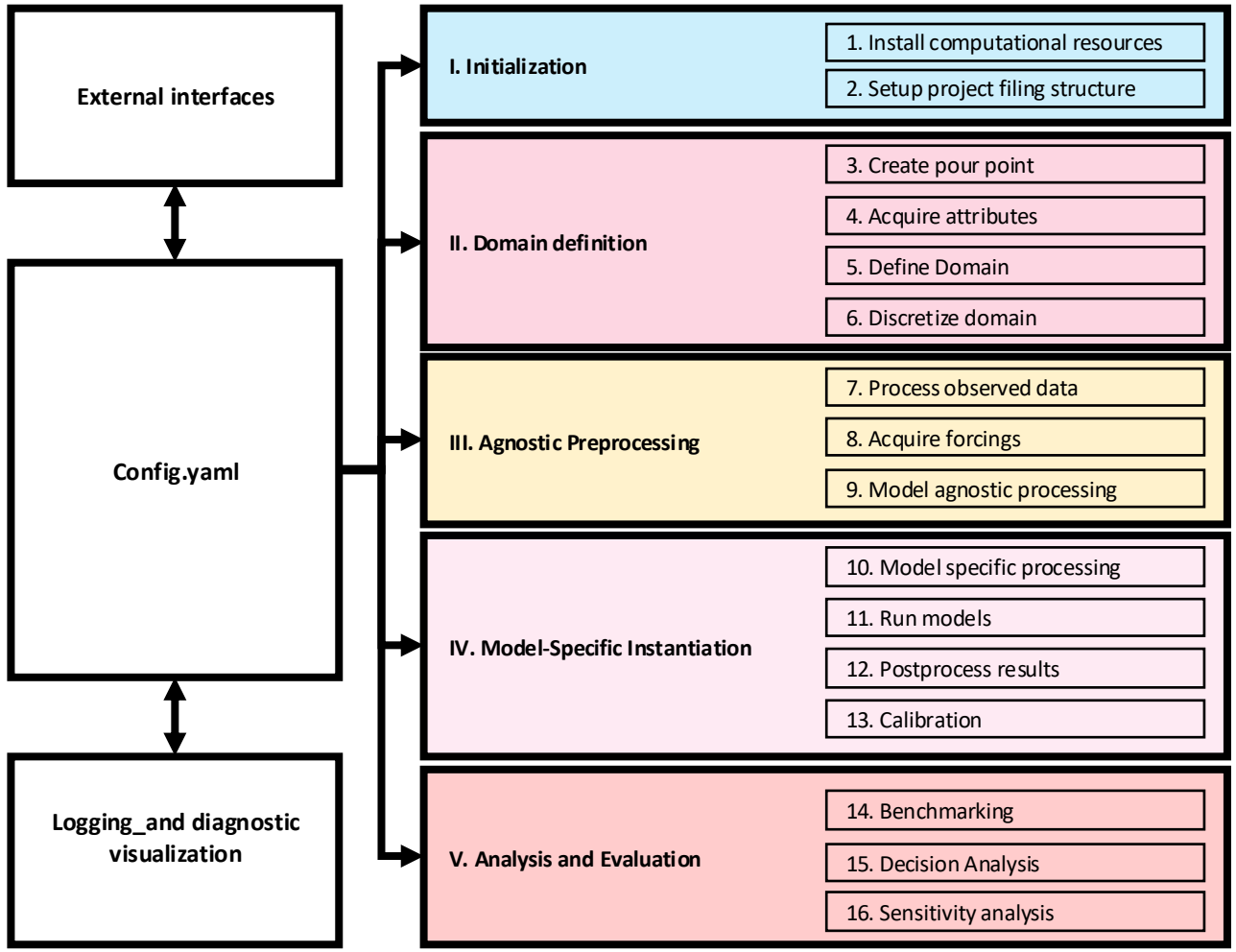
Eythorsson et al., 2026a “Out-standing in Every Field: The Vision for Shared Architecture in Computational Hydrology”, *in prep*

Session 1: The Configuration Problem – Cohesion

Layers of composable components

Decision points

- **Project initialization**
- Domain spatial discretization
- Input data
- Model specific configuration
- Optimisation strategy
- Evaluation



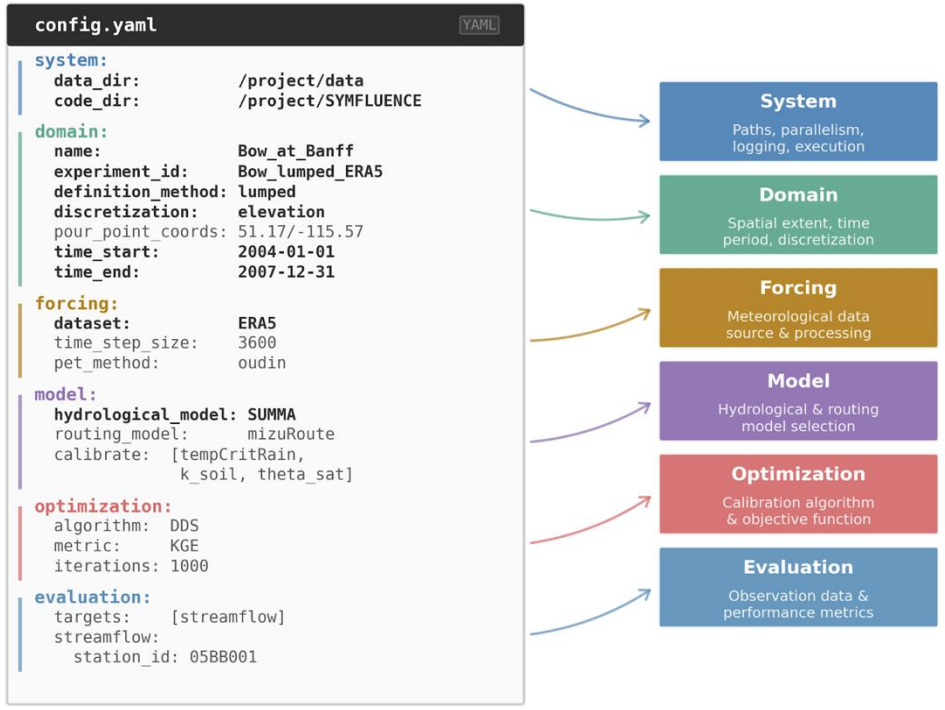
Session 1: The Configuration Problem - Settings

Layers of composable components

Decision points

- **Project initialization**
- **Domain spatial discretization**
- **Input data**
- **Model specific configuration**
- **Optimisation strategy**
- **Evaluation**

Configuration expresses scientific intent; the framework handles implementation



Minimal working example — 10 required parameters shown (bold keys); all others use validated defaults

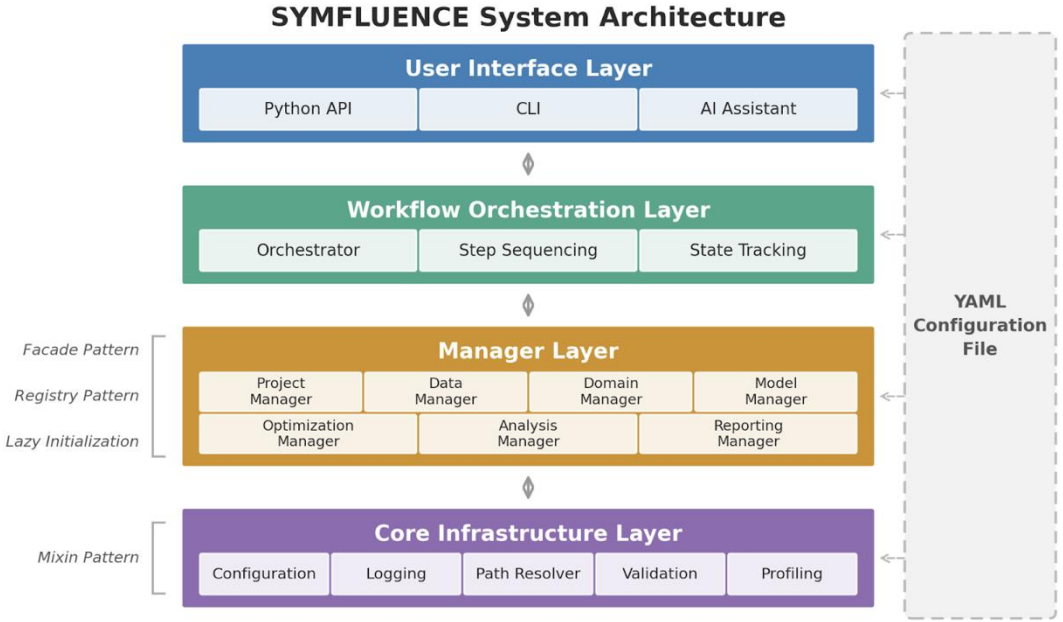
Eythorsson et al., 2026b "The Registry as Social Contract: Architectural Patterns for Community Hydrological Modeling", *in prep*

Session 1: The Configuration Problem - Software Architecture

Layers of composable components

Decision points

- **Project initialization**
- **Domain spatial discretization**
- **Input data**
- **Model specific configuration**
- **Optimisation strategy**
- **Evaluation**



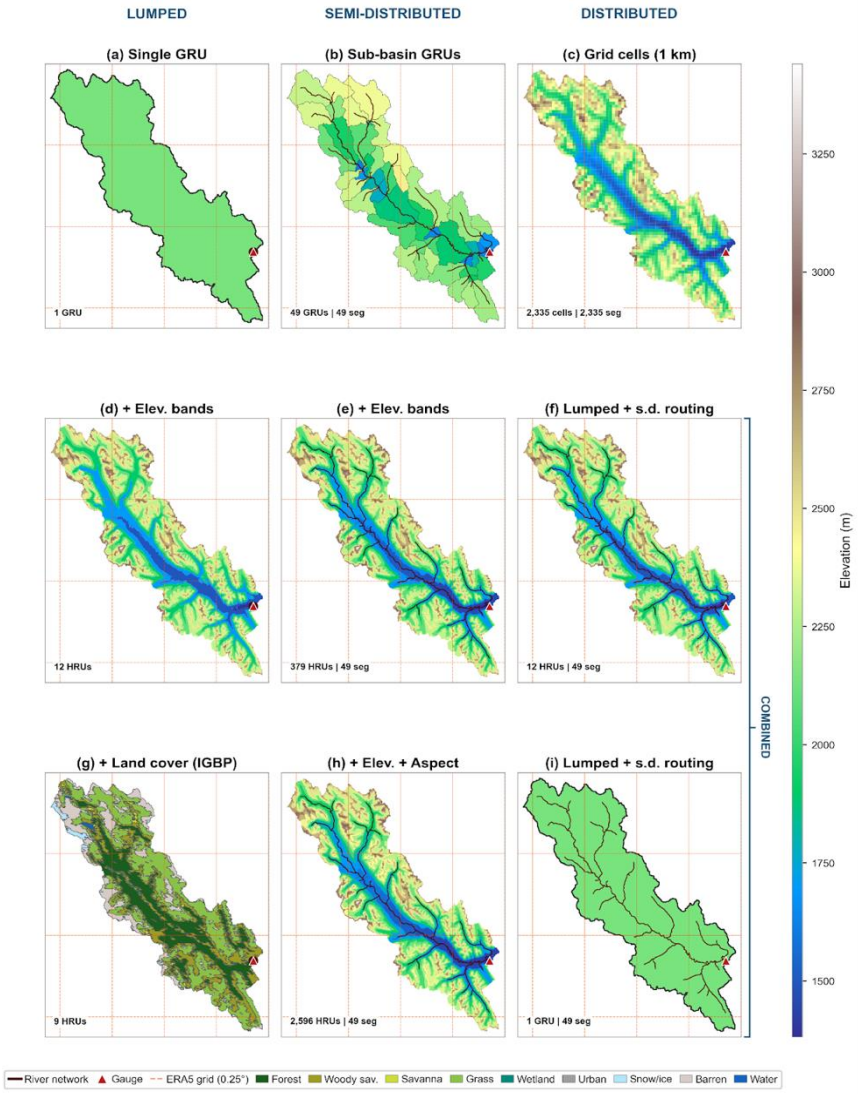
Eythorsson et al., 2026b "The Registry as Social Contract: Architectural Patterns for Community Hydrological Modeling", *in prep*

Session 1: The Configuration Problem - Resolution

Layers of composable components

Decision points

- **Project initialization**
- **Domain spatial discretization**
- **Input data**
- **Model specific configuration**
- **Optimisation strategy**
- **Evaluation**



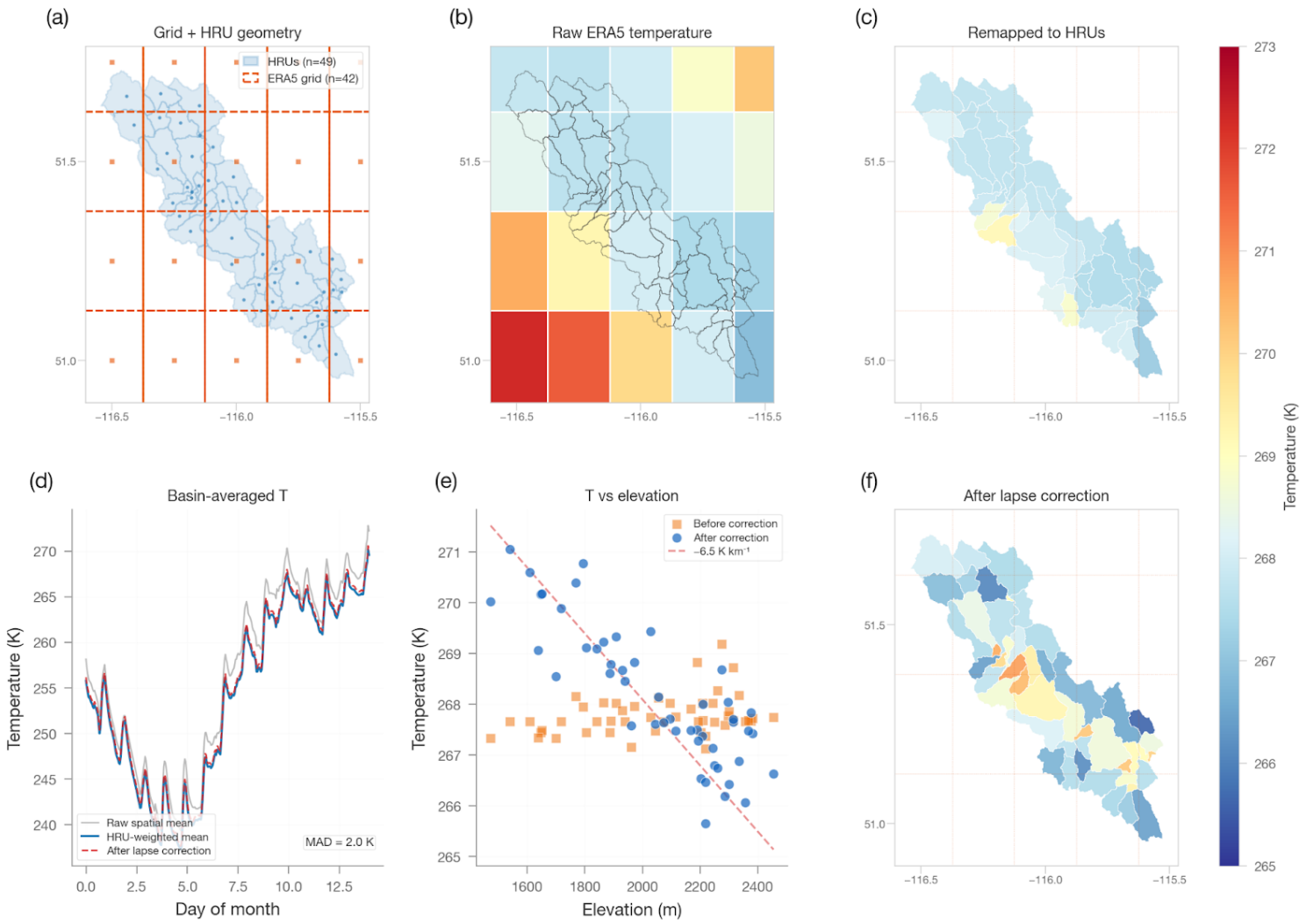
Eythorsson et al., 2026c "From Configuration to Prediction: Multi-Model, Multi-Basin Experiments with SYMFLUENCE", *in prep*

Session 1: The Configuration Problem - Processing

Layers of composable components

Decision points

- **Project initialization**
- **Domain spatial discretization**
- **Input data**
- **Model specific configuration**
- **Optimisation strategy**
- **Evaluation**



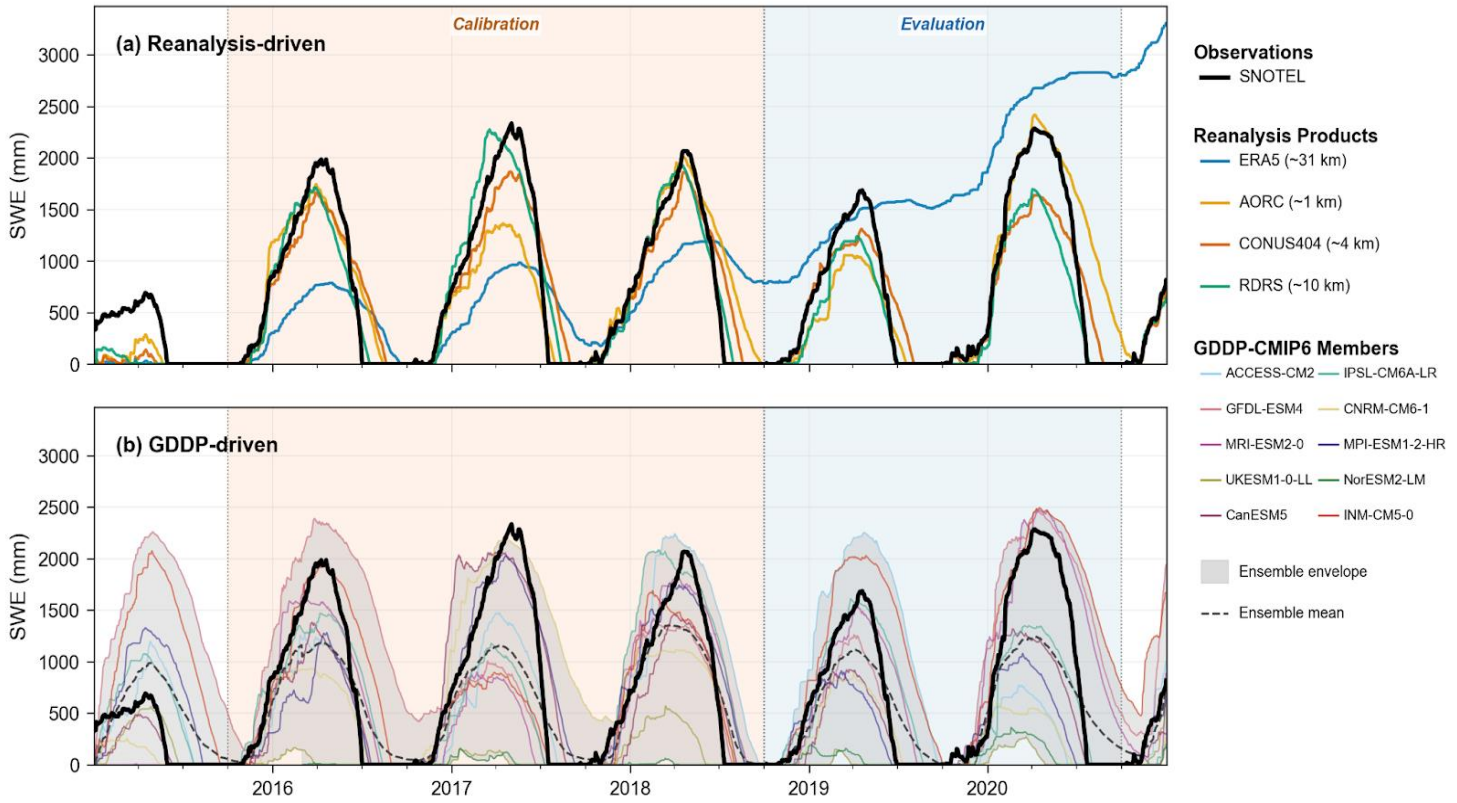
Eythorsson et al., 2026c "From Configuration to Prediction: Multi-Model, Multi-Basin Experiments with SYMFLUENCE", *in prep*

Session 1: The Configuration Problem - Forcing

Layers of composable components

Decision points

- Project initialization
- Domain spatial discretization
- **Input data**
- Model specific configuration
- Optimisation strategy
- Evaluation



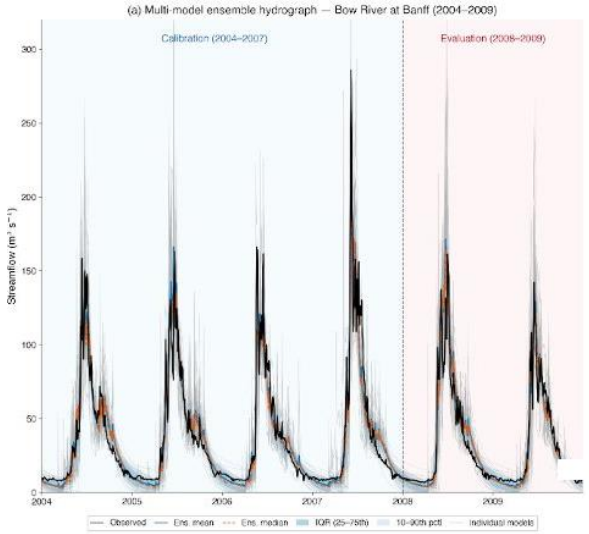
Eythorsson et al., 2026c "From Configuration to Prediction: Multi-Model, Multi-Basin Experiments with SYMFLUENCE", *in prep*

Session 1: The Configuration Problem - Orchestration

Layers of composable components

Decision points

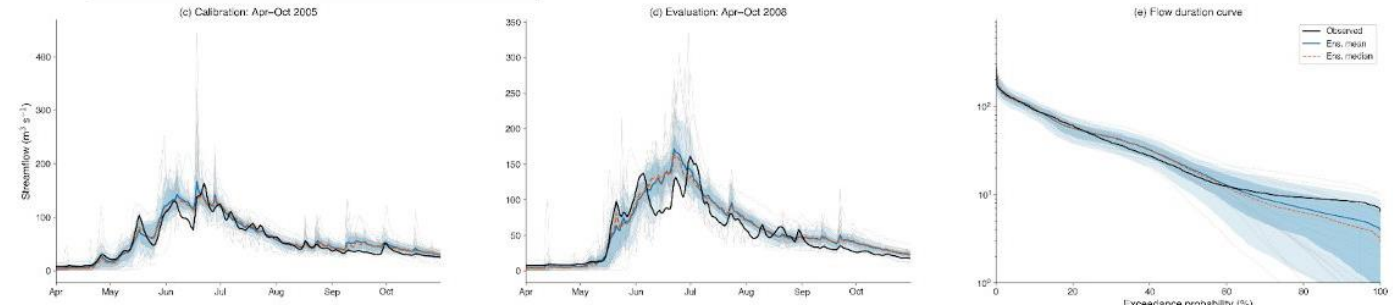
- Project initialization
- Domain spatial discretization
- Input data
- **Model specific configuration**
- Optimisation strategy
- Evaluation



(b) Model performance summary (ranked by evaluation KGE)

#	ID	nP	Cal KGE	Evaluation KGE	r	α	β	NSE	PB%	Fl%	Time calib
1	M01	115k	0.97	0.88	0.94	1.09	1.05	0.85	4.7	--	8.1h
2	M02	11	0.90	0.88	0.91	1.08	0.99	0.81	-1.3	0	4.3h
3	M03	16	0.90	0.86	0.90	1.10	1.01	0.77	0.9	0	6.2h
4	M04	17	0.92	0.86	0.94	1.13	1.00	0.84	-0.1	4	6.3h
5	M05	29	0.90	0.84	0.95	1.15	1.05	0.85	5.3	41	29.9h
6	M06	9	0.89	0.83	0.92	1.15	1.00	0.79	0.2	13	6.3h
7	M07	17	0.87	0.82	0.89	1.14	1.03	0.73	3.0	1	9.1m
8	M08	19	0.88	0.82	0.86	1.09	0.93	0.68	-6.6	0	140s
9	M09	15	0.85	0.82	0.86	0.96	1.11	0.72	11.3	0	10.1m
10	M10	10	0.87	0.81	0.91	1.16	0.97	0.77	-2.7	0	18.9m
11	M11	16	0.92	0.81	0.93	1.17	1.06	0.80	6.3	1	59.8m
12	M12	10	0.90	0.80	0.94	1.18	0.95	0.81	-5.3	19	3.8m
13	M13	21	0.94	0.80	0.95	1.19	1.01	0.85	1.5	7	12.3m
14	M14	4	0.92	0.79	0.94	1.15	1.12	0.82	12.5	0	14.1m
15	M15	14	0.94	0.79	0.94	1.19	1.07	0.81	6.5	0	10.1h
16	M16	19	0.90	0.79	0.94	1.28	1.03	0.81	3.3	0	3.7h
17	M17	14	0.89	0.78	0.89	1.18	1.06	0.70	6.1	0	103.2h
18	M18	13	0.82	0.77	0.82	1.14	1.00	0.58	0.2	0	17.7m
19	M19	14	0.88	0.76	0.94	1.22	0.93	0.79	-6.7	0	143s
20	M20	25	0.88	0.75	0.83	1.18	1.07	0.56	6.6	0	3.7m
21	M21	13	0.75	0.74	0.84	1.12	0.84	0.59	-15.9	13	7.3h
22	M22	26	0.83	0.73	0.84	1.22	0.98	0.56	-1.9	0	9.9m
23	M23	13	0.81	0.71	0.86	1.24	1.07	0.58	6.8	1	15.7h
24	M24	14	0.74	0.70	0.92	1.22	0.81	0.73	-18.6	0	149s
25	M25	7	0.88	0.66	0.85	1.30	1.08	0.53	7.8	0	28.7h
26	M26	11	0.89	0.62	0.92	1.26	1.26	0.66	25.9	2	97s
27	M27	11	0.72	0.62	0.92	1.16	1.33	0.68	33.2	0	5.3h
	Ens. mean			0.93	0.89						
	Ens. median			0.91	0.90						

nP: # params Fl%: crash rate Time: calibration time



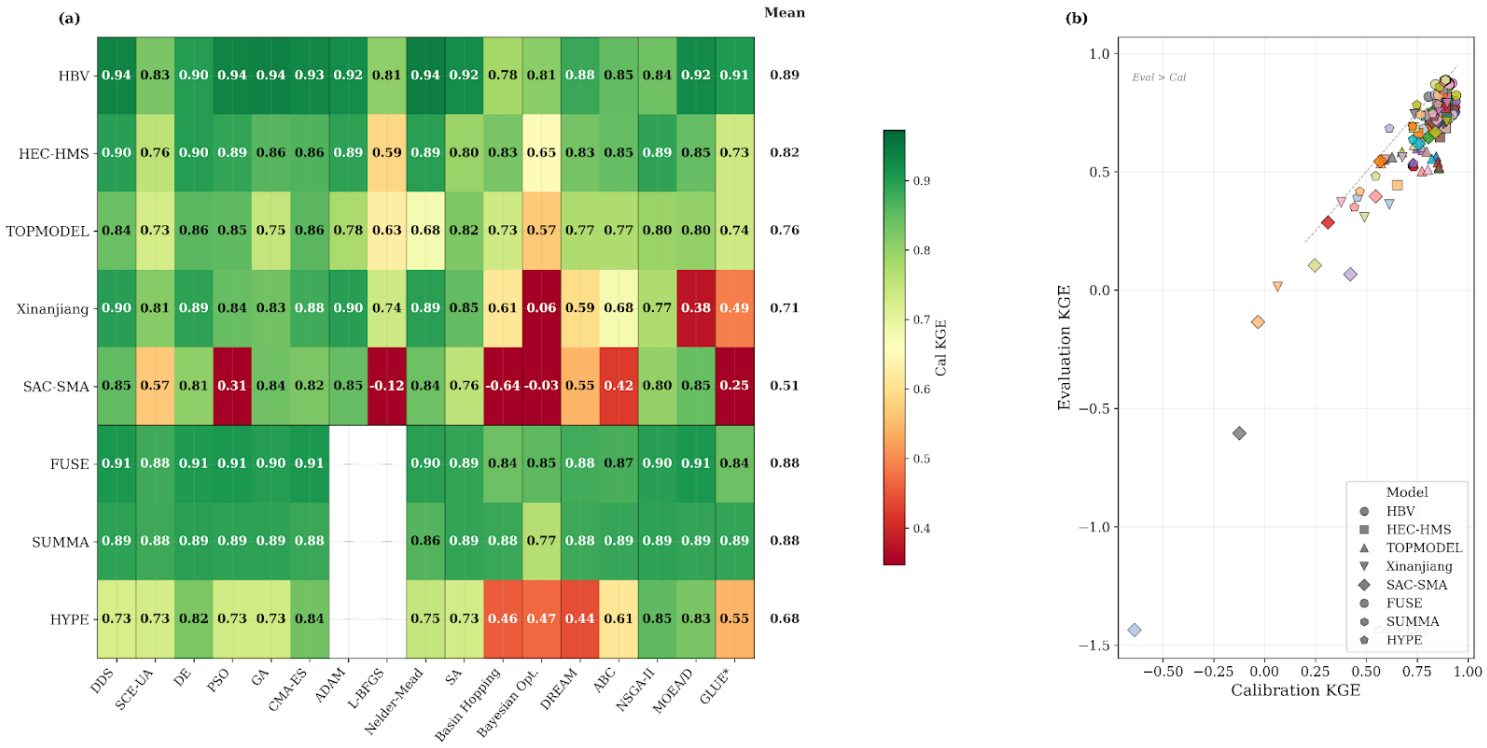
Eythorsson et al., 2026c “From Configuration to Prediction: Multi-Model, Multi-Basin Experiments with SYMFLUENCE”, *in prep*

Session 1: The Configuration Problem - Calibration

Layers of composable components

Decision points

- Project initialization
- Domain spatial discretization
- Input data
- Model specific configuration
- **Optimisation strategy**
- Evaluation



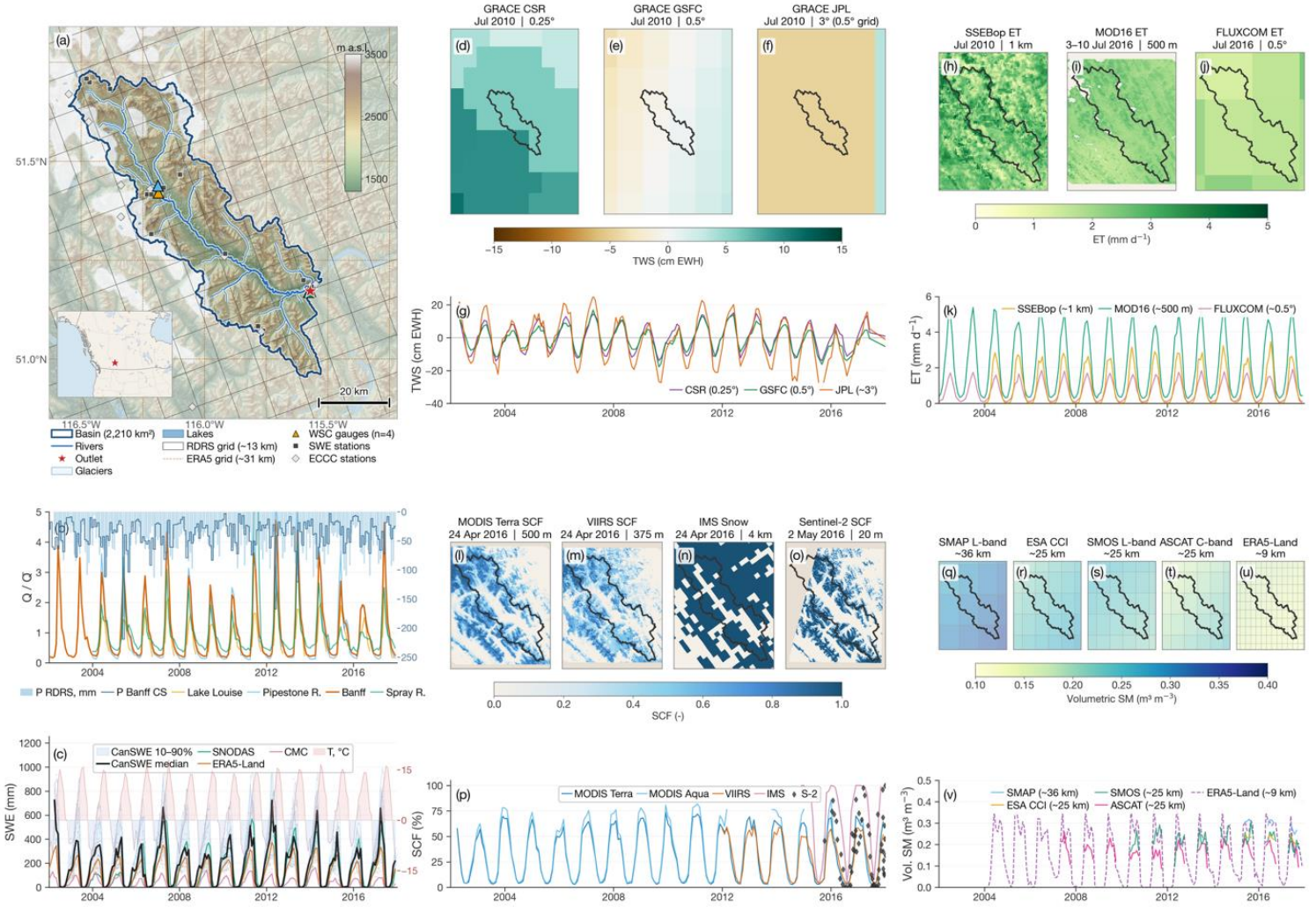
Eythorsson et al., 2026c “From Configuration to Prediction: Multi-Model, Multi-Basin Experiments with SYMFLUENCE”, *in prep*

Session 1: The Configuration Problem - Evaluation

Layers of composable components

Decision points

- **Project initialization**
- **Domain spatial discretization**
- **Input data**
- **Model specific configuration**
- **Optimisation strategy**
- **Evaluation**



Eythorsson et al., 2026c “From Configuration to Prediction: Multi-Model, Multi-Basin Experiments with SYMFLUENCE”, *in prep*

Session 1: The uncertainty-quantification problem

The binding constraint on uncertainty quantification in hydrology is not methods, data, or compute -- it is integration infrastructure.

A systematic uncertainty budget requires:

- Multiple models -- structural uncertainty
- Multiple forcings -- input uncertainty
- Multiple algorithms -- calibration uncertainty
- Multiple evaluation targets -- diagnostic depth

Current practice:

- Each group wires its own pipeline

Combinatorial integration cost

M × N

bespoke adapters for M forcings × N models

16 forcings × 27 models × 17 algorithms × 14 metrics

> 100,000 pairwise adapters

Session 1: The Registry as social contract

One generic container for every pluggable component type

The pattern

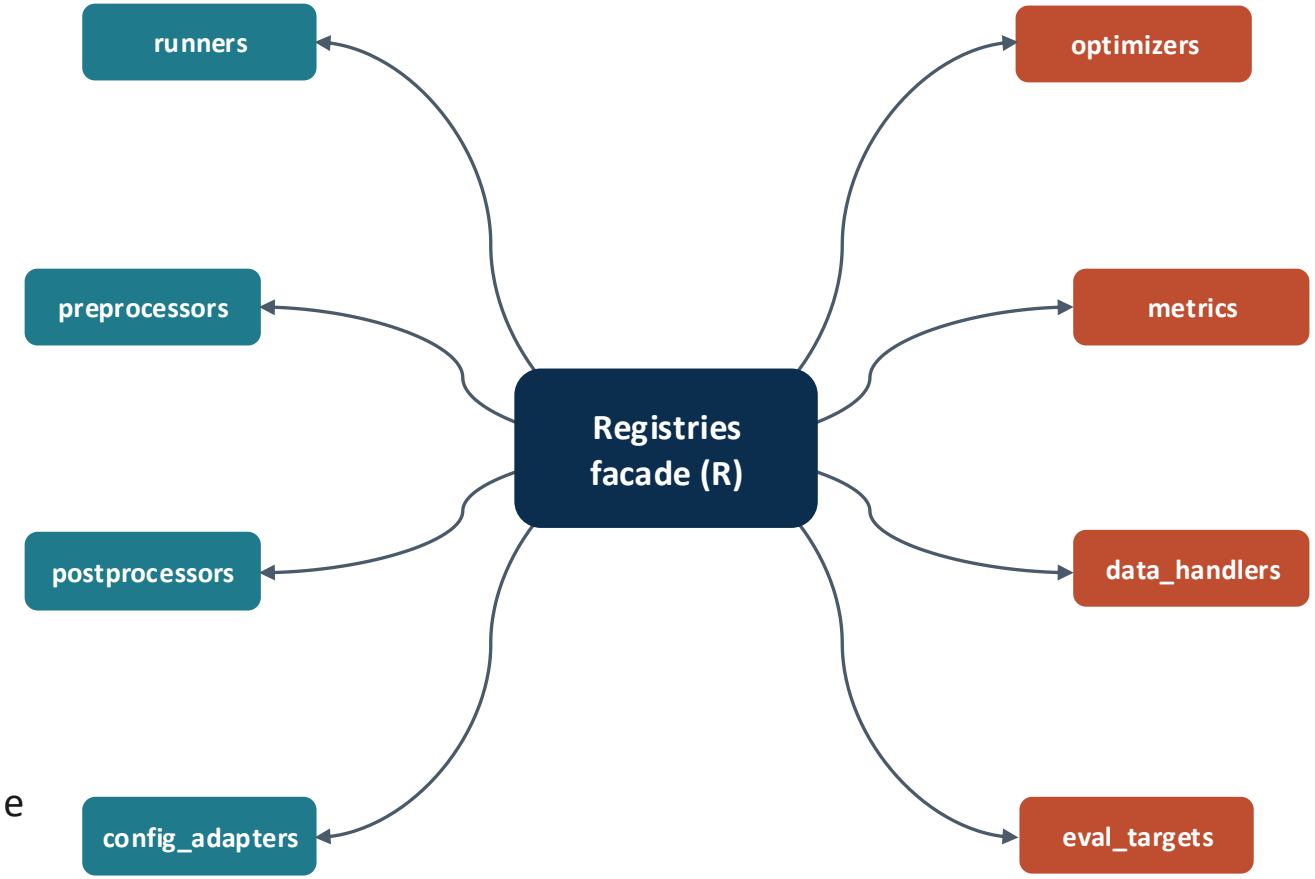
- One Registry[T] class — models, data handlers, optimizers, metrics, evaluation targets, etc.
- Dict-like API: R.runners["SUMMA"]
- Lazy imports: components resolved on first use

The procedure

- Independently extendible through python setuptools

1. Create PyPI package
2. Inherit base classes
3. Register component classes
4. Declare entry point in pyproject.toml

- SYMFLUENCE discovers registered components at runtime
- No source code modification required
    - Default components PR to pyproject.toml



30 domain registries share the same API

Session 1: The Registry: M × N become M + N

One interface per component

Thirty typed registries. Current inventory:

- **27 models** across 6 languages (Fortran, C, C++, R, Julia, Python)
- **60+ data handlers** (ERA5, AORC, RDRS, CARRA, GRACE, …)
- **17 calibration algorithms** (DDS, DE, NSGA-II, DREAM, ADAM, …)
- **14 metrics**, 12 benchmarks, routing / optimization plug-ins

The M+N reduction:

- Each forcing converts once to a CF-Intermediate Format
- Each model reads from that format once
- Adding model #28: write one adapter

Without the registry

M × N

pairwise adapters per workflow stage

↓

With typed registries

M + N

one adapter per component

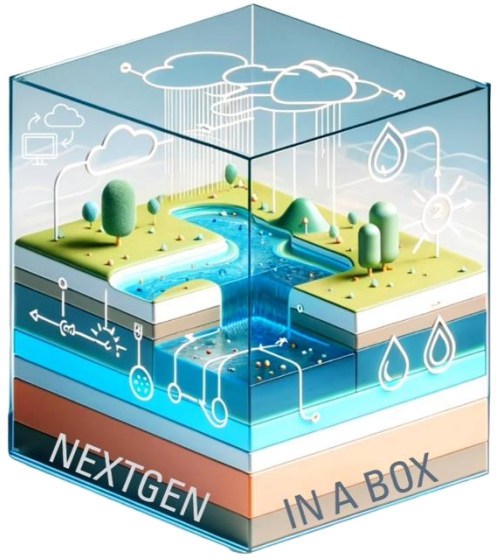
e.g. 27 models + 60 data handlers = 87 adapters
not 27 × 60 = 1,620

Session 1: Connection to NextGen / NGIAB – Orchestration

Community instrument ecosystem

- Models
- Datasets
- Geospatial
- Evaluation
- Optimization
- Benchmarks
- Parallelization
- **Orchestration (SYMFLUENCE)**

Configurations + Processed Inputs



NGIAB platform

Research

two-way pipeline

Operations

feedback — findings inform new configurations

Part B

Hands-On: Logan River with jHBV

Workshop exercise: Logan River

- **What we will build:**
  - Lumped watershed model for USGS 10109000 (Logan River, UT)
  - jHBV hydrological model
  - RDRS forcing data
  - DDS calibration optimizing KGE (200 iterations)
- Notebook: examples/04_workshop_notebooks/04a_logan_river_workshop.ipynb
- Workflow stages: Config -> Domain -> Data -> Model -> Evaluate -> Calibrate

Session 1: Test setup

- https://workshop.ciroh.awi.2i2c.cloud

Download, upload and open Jupyter notebook:

https://github.com/symfluence-org/SYMFLUENCE/tree/main/examples/04_workshop_notebooks

- **04a_logan_river_workshop.ipynb**

Step 1 — Configuration

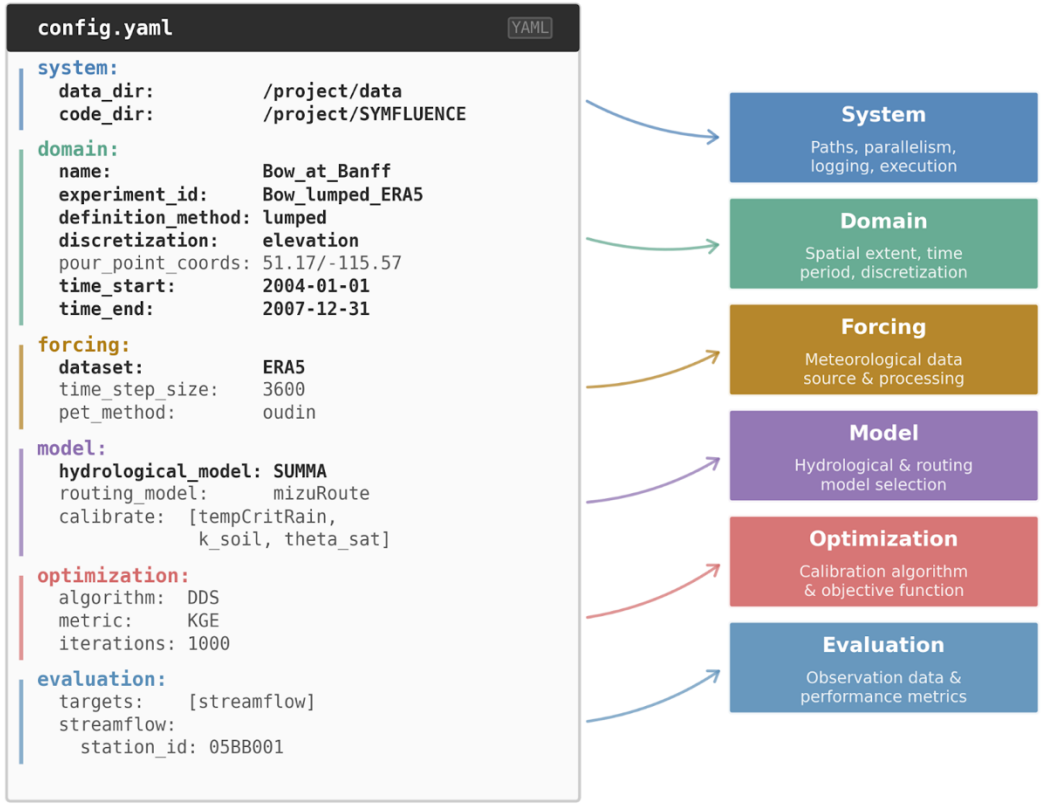
One YAML config captures the full scientific experiment:

Minimal config in Python API:

- SymfluenceConfig.from_minimal()
  - Start from defaults, override what you need

Notebook Step1

A single YAML file defines the complete experiment — 6 sections, 450+ available parameters



Minimal working example — 10 required parameters shown (bold keys); all others use validated defaults

Eythorsson et al., 2026b “The Registry as Social Contract: Architectural Patterns for Community Hydrological Modeling”, *in prep*

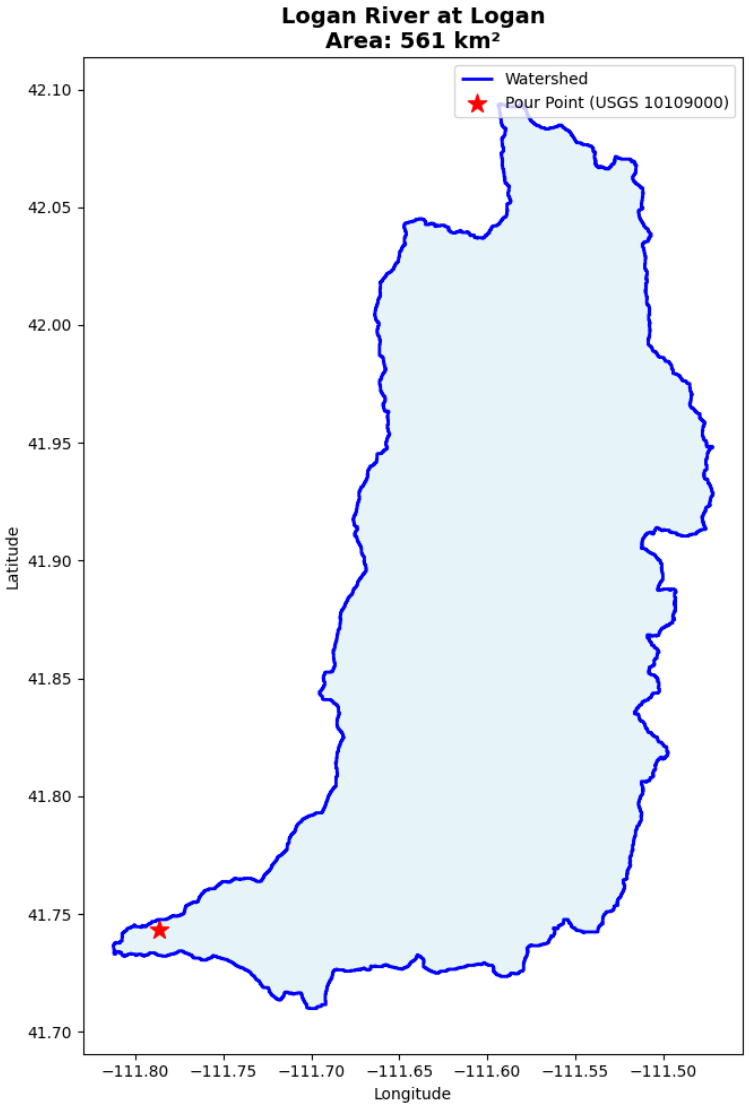
Step 2 — Domain definition

Three sub-steps, one line each:

- Geospatial attribute acquisition
  - Elevation (Copernicus DEM 90m), land cover, soil type
- Watershed delineation
  - TauDEM: DEM -> flow direction -> contributing area -> basin boundary
- Domain discretization
  - Lumped: single GRU (Grouped Response Unit)

Notebook steps 2a-2c

- symfluence.managers['data'].acquire_attributes()
- symfluence.managers['domain'].define_domain()
- symfluence.managers['domain'].discretize_domain()



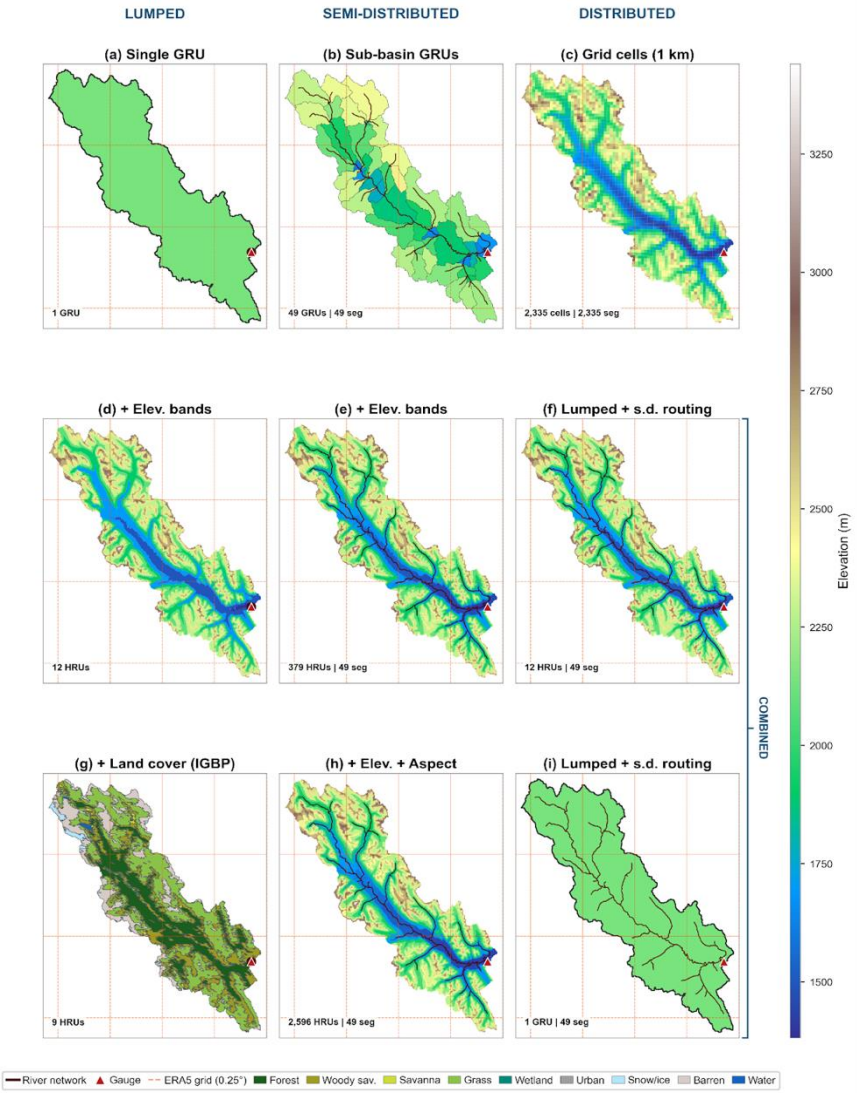
Step 2 — Domain definition

Three sub-steps, one line each:

- Geospatial attribute acquisition
  - Elevation (Copernicus DEM 90m), land cover, soil type
- Watershed delineation
  - TauDEM: DEM -> flow direction -> contributing area -> basin boundary
- Domain discretization
  - Lumped: single GRU (Grouped Response Unit)

Notebook steps 2a-2c

- symfluence.managers['data'].acquire_attributes()
- symfluence.managers['domain'].define_domain()
- symfluence.managers['domain'].discretize_domain()

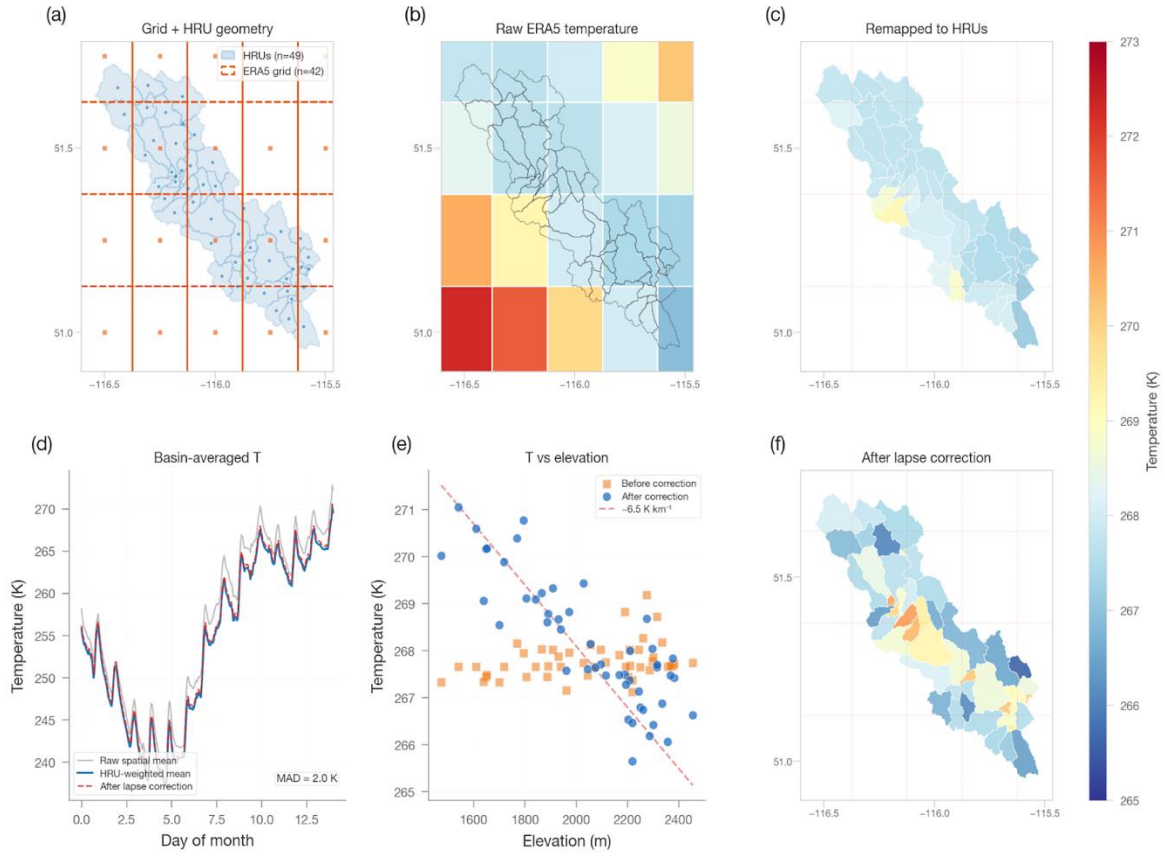


Eythorsson et al., 2026c “From Configuration to Prediction: Multi-Model, Multi-Basin Experiments with SYMFLUENCE”, *in prep*

Step 3 — Data acquisition & preprocessing

- Streamflow observations
  - USGS NWIS API -> station 10109000 -> discharge (m3/s)
- Meteorological forcing — RDRS
  - 12.5 km spatial resolution, hourly temporal resolution
  - Precipitation, temperature, humidity, wind, radiation, pressure
- Model-agnostic preprocessing
  - Standardize variable names (CF conventions)
  - Unit conversion, spatial averaging over watershed
- Output: Model ready datastore
  - The same input data feeds jHBV, SUMMA, FUSE, or NextGen

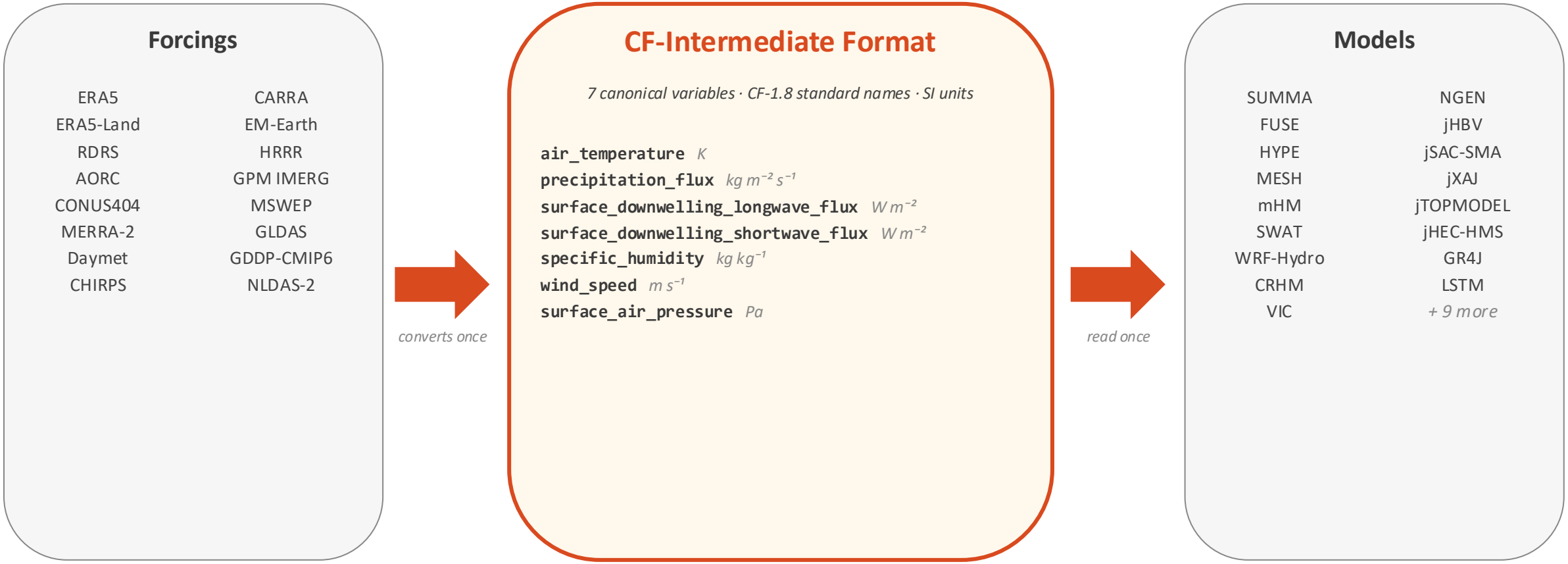
Notebook steps 3a – 3c



Eythorsson et al., 2026c “From Configuration to Prediction: Multi-Model, Multi-Basin Experiments with SYMFLUENCE”, *in prep*

Step 3 — Data acquisition & preprocessing

One format, every model



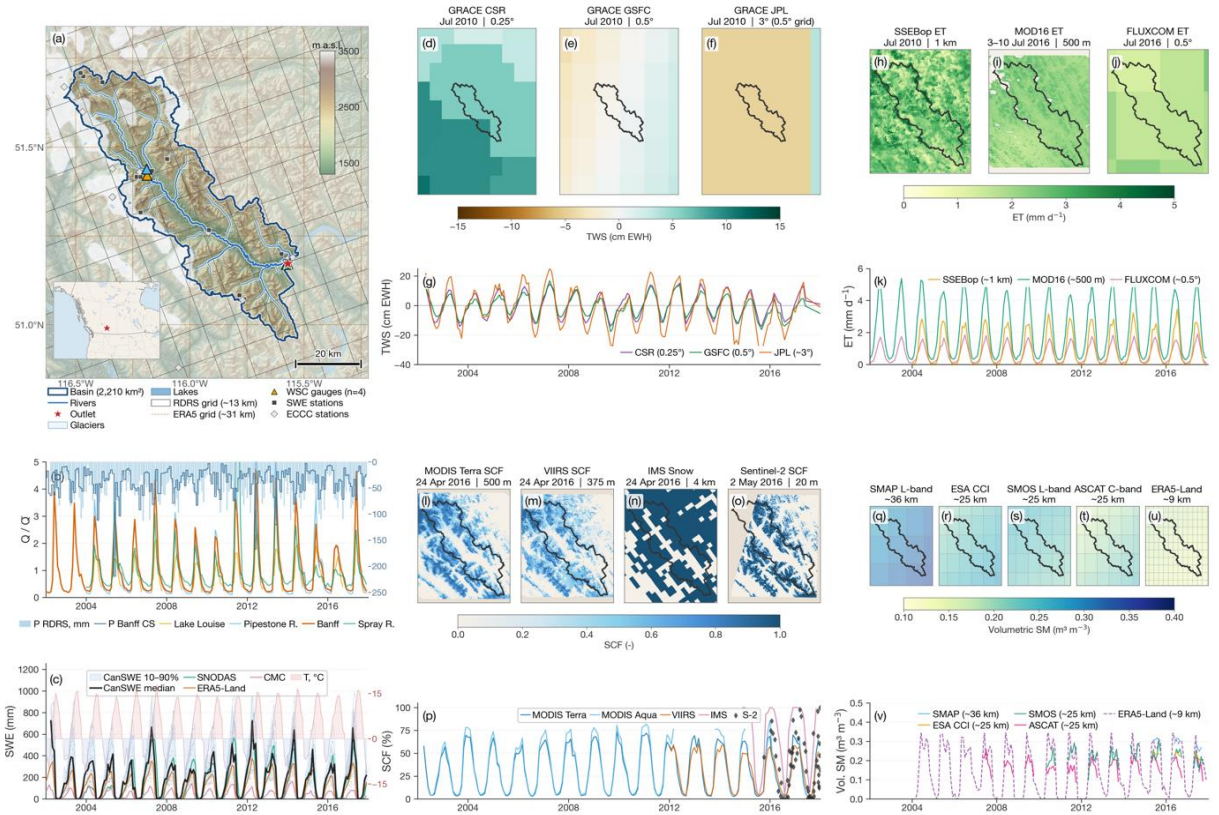
Without a shared format: **16 × 27 = 432 adapters** With CFIF: **16 + 27 = 43**

Step 3 — Data acquisition & preprocessing

One format, every model

3 classes of data

- **Forcing (dynamic)**
  - **Re-analysis**
  - **Forecast**
  - **GCM**
- **Geospatial attributes (“static”)**
  - **Configuration (elevation, soil class, land cover, etc.,)**
  - **Optimization (signatures, stratification, climate, etc.,)**
- **Observations**
  - **Point scale (streamflow, snow, soil moisture, etc.,)**
  - **Distributed (GRACE, MODIS, SMAP, etc.,)**



Eythorsson et al., 2026c “From Configuration to Prediction: Multi-Model, Multi-Basin Experiments with SYMFLUENCE”, *in prep*

Step 3 — Data acquisition & preprocessing

One store, every input

Forcing

16 datasets

ERA5 · RDRS · AORC
CONUS404 · CARRA
MERRA-2 · Daymet · …

Observations

6 families · 30+ sources

USGS · WSC · GRDC
SNOTEL · CanSWE · MODIS
SMAP · GRACE · FLUXNET

Attributes

15 datasets

Copernicus DEM · MERIT
SoilGrids · POLARIS
MODIS LC · HydroLAKES · RGI

Model-Ready Data Store

NetCDF-4 · CF-1.8 metadata · per-HRU indexed

Forcing variables

```
air_temperature(time, hru)
precipitation_flux(time, hru)
surface_downwelling_longwave_flux
```

Observation series

```
discharge_cms(time)  ·  swe(time, station)
et(time, hru)  ·  soil_moisture(time, hru)
sca(time, hru)  ·  tws_anomaly(time)
```

Static attributes

```
hru_area(hru)  ·  elev_mean(hru)
soil_fraction(hru, soil_class)  ·
land_fraction(hru, land_class)
```

Simulation

any of 27 models
reads forcing + attrs
produces fluxes & states

Calibration & evaluation

any of 17 algorithms
any of 14 metrics
multivariate-ready

Configuration & regionalization

transfer functions
parameter regionalization
geofabric coupling

Any model. Any metric. Any optimizer. **One store.**

Step 3 — Data acquisition & preprocessing

Table C1. Meteorological forcing datasets (16 handlers).

Dataset	Provider	Coverage	Resolution	Access Mode	Reference
ERA5	ECMWF/Copernicus	Global	0.25°, hourly	Cloud (CDS API)	Hersbach et al. (2020)
ERA5-Land	ECMWF/Copernicus	Global	0.1°, hourly	Cloud (CDS API)	Muñoz-Sabater et al. (2021)
RDRS v2.1	ECCC	North America	10 km, hourly	Cloud / MAF	Gasset et al. (2021)
AORC	NOAA	CONUS	0.01°, hourly	Cloud (AWS S3)	Benjamin et al. (2016)
CONUS404	NCAR/USGS	CONUS	4 km, hourly	Cloud / User	Rasmussen et al. (2023)
MERRA-2	NASA GMAO	Global	0.5°×0.625°, hourly	Cloud (GES DISC)	Gelaro et al. (2017)
Daymet v4	ORNL DAAC	North America	1 km, daily	Cloud	Thornton et al. (2022)
CHIRPS	UCSB	50°S–50°N	0.05°, daily	Cloud	Funk et al. (2015)
CARRA	Copernicus	Arctic/N. Atlantic	2.5 km, 3-hourly	Cloud (CDS API)	—
EM-Earth	Tang et al.	Global	0.1°, daily	Cloud	Tang et al. (2022)
HRRR	NOAA	CONUS	3 km, hourly	Cloud (AWS S3)	Benjamin et al. (2016)
GPM IMERG	NASA	Global (60°S–60°N)	0.1°, 30-min	Cloud	Hou et al. (2014)
MSWEP v2	Beck et al.	Global	0.1°, 3-hourly	User	Beck et al. (2019)
GLDAS	NASA	Global	0.25°, 3-hourly	Cloud	Rodell et al. (2004)
NEX-GDDP-CMIP6	NASA	Global	0.25°, daily	Cloud (OpenDAP/Zarr)	Thrasher et al. (2022)
NLDAS-2	NASA	CONUS	0.125°, hourly	Cloud	Rodell et al. (2004)

Table C2. Streamflow observation sources (6 handlers).

Source	Provider	Coverage	Variables	Temporal Res.	Reference
USGS NWIS	U.S. Geological Survey	USA	Discharge	15-min, daily	—
WSC Hydat	Water Survey of Canada	Canada	Discharge	Daily	—
GRDC	BfG Germany	Global	Discharge	Daily, monthly	—
SMHI	Swedish Met. & Hydrol. Institute	Sweden	Discharge	Daily	—
Hub'Eau	Ministère de l'Écologie (France)	France	Discharge	Daily	—
IMO	Icelandic Meteorological Office	Iceland	Discharge	Daily	—

Table C3. Snow observation products (8 handlers).

Product	Provider	Coverage	Resolution	Variables	Reference
SNODAS	NOAA NSIDC	CONUS	1 km, daily	SWE, snow depth, melt	—
MODIS MOD10A1	NASA LP DAAC	Global	500 m, daily	Fractional snow cover	Hall & Riggs (2021)
VIIRS VNP10A1	NASA NSIDC	Global	375 m, daily	Fractional snow cover	—
IMS	NOAA NESDIS	N. Hemisphere	4 km, daily	Binary snow extent	—
SNOTEL	USDA NRCS	Western USA	Point, daily/hourly	SWE, snow depth, T, P	—
CanSWE	ECCC / Universities	Canada	Point, daily–weekly	SWE, snow depth	—
GlobSnow v3	FMI/ESA	N. Hemisphere	25 km, daily	SWE	Luojus et al. (2021)
CMC Snow Depth	CMC Canada	N. Hemisphere	24 km, daily	Snow depth	Brown & Brasnett (2010)

Eythorsson et al., 2026b "The Registry as Social Contract: Architectural Patterns for Community Hydrological Modeling", *in prep*

Step 3 — Data acquisition & preprocessing

Table C4. Soil moisture, evapotranspiration, groundwater, and TWS products (15 handlers).

Product	Provider	Coverage	Resolution	Variables	Reference
SMAP L3	NASA NSIDC	Global	:36 km (9 km enh.), daily	Surface soil moisture	O'Neill et al. (2021)
SMOS L3	ESA/CATDS	Global	:25 km, daily	Surface soil moisture	Kerr et al. (2012)
ASCAT	EUMETSAT	Global	25 km, daily	Surface soil moisture (index)	Wagner et al. (2013)
ESA CCI SM	ESA	Global	0.25°, daily	Surface soil moisture (merged)	Dorigo et al. (2017)
Sentinel-1	ESA Copernicus	Global	1 km, 6–12 days	Surface soil moisture	—
ISMN	TU Wien/GEWEX	Global (stations)	Point, hourly–daily	Soil moisture (multi-depth)	Dorigo et al. (2021)
MODIS MOD16A2	NASA LP DAAC	Global	500 m, 8-day	ET, PET	Running et al. (2019)
GLEAM v3	Ghent University	Global	0.25°, daily	ET, transpiration, interception	Martens et al. (2017)
FLUXCOM	MPI-BGC	Global	0.25°–0.0833°, daily–monthly	ET (ML upscaled)	Jung et al. (2019)
FLUXNET	Regional networks	Global (towers)	Point, 30-min	ET, sensible heat, NEE	—
SSEBop	USGS EROS	Global	1 km, monthly	Actual ET	Senay et al. (2013)
OpenET	OpenET Inc.	Western USA	30 m, monthly	ET (ensemble)	Melton et al. (2022)
GRACE/GRACE-FO	NASA PO.DAAC	Global	:300 km (mascon), monthly	TWS anomaly	Tapley et al. (2019)
Well observations	National agencies	Variable	Point, hourly–monthly	Groundwater level	—
GGMN	IGRAC	Global	Point, variable	Groundwater level	—

Table C5. Static geospatial and attribute datasets (15 handlers).

Product	Provider	Coverage	Resolution	Variables	Reference
Copernicus DEM	ESA/Copernicus	Global	30 m, 90 m	Elevation	—
MERIT DEM	Yamazaki et al.	Global	:90 m	Elevation (hydrol. conditioned)	Yamazaki et al. (2019)
SRTM	NASA	60°S–60°N	30 m, 90 m	Elevation	—
SoilGrids 2.0	ISRIC	Global	250 m	Sand/silt/clay %, bulk density, SOC, pH	Poggio et al. (2021)
POLARIS	Chaney et al.	CONUS	30 m	Soil properties (probabilistic)	Chaney et al. (2019)
GLHYMPS 2.0	Huscroft et al.	Global	:1 km	Permeability, porosity	Huscroft et al. (2018)
Soil thickness	Pelletier et al.	Global	:1 km	Soil, regolith, sediment depth	Pelletier et al. (2016)
MODIS MCD12Q1	NASA LP DAAC	Global	500 m, annual	Land cover type (IGBP)	Friedl & Sulla-Menashe (2019)
MODIS MCD15A2H	NASA LP DAAC	Global	500 m, 8-day	LAI, FPAR	Myneni et al. (2021)
MODIS MOD13A2	NASA LP DAAC	Global	1 km, 16-day	NDVI, EVI	Didan (2021)
GEDI L2B	NASA	52°S–52°N	25 m footprint	Canopy height, cover	Dubayah et al. (2020)
RGI v7.0	NSIDC/GLIMS	Global	Vector outlines	Glacier extent, area	RGI Consortium (2023)
Global Surface Water	JRC/Pekel et al.	Global	30 m, monthly	Water occurrence, change	Pekel et al. (2016)
HydroLAKES	Messager et al.	Global	Vector polygons	Lake area, volume, depth	Messager et al. (2016)
NLCD	USGS	CONUS	30 m, multi-year	Land cover, impervious surface	—

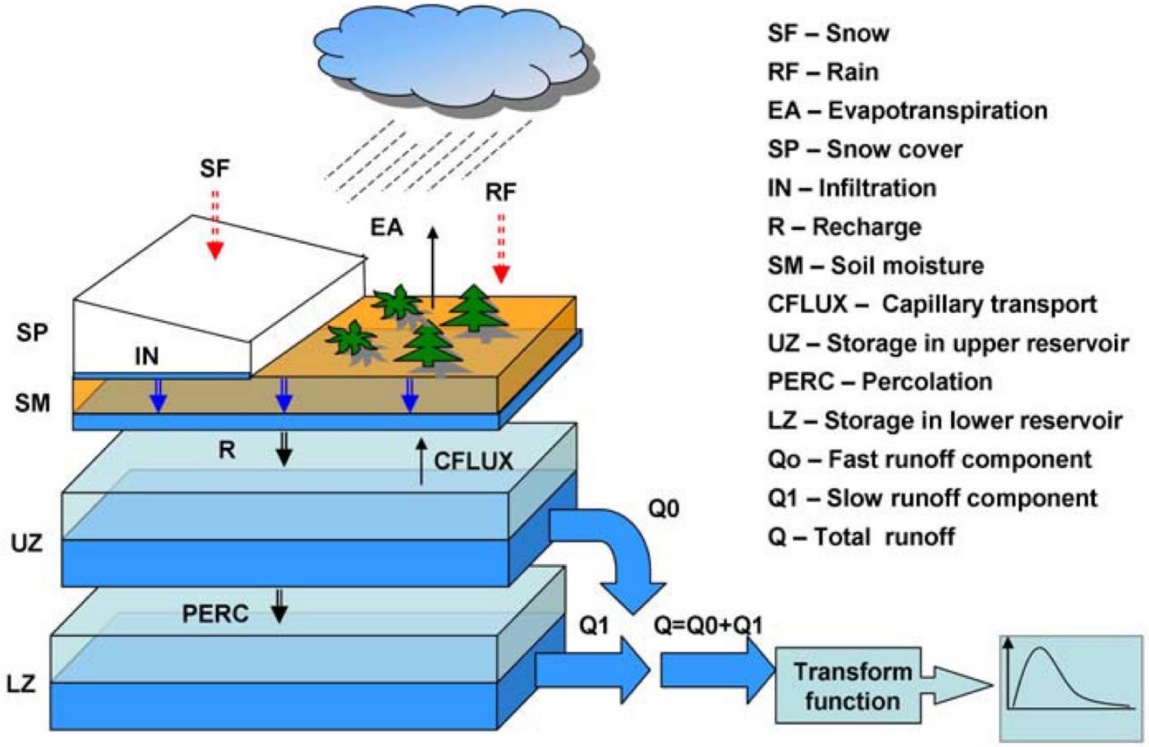
Eythorsson et al., 2026b "The Registry as Social Contract: Architectural Patterns for Community Hydrological Modeling", *in prep*

Step 4 — Model execution: jHBV

- jHBV: JAX-AD enabled HBV conceptual model
  - Classic HBV structure:
    - Snow routine (degree-day)
    - Soil moisture accounting
    - Response function (upper + lower reservoir)
    - Routing (triangular weighting / gamma)

Notebook steps 4a-4b

- symfluence.managers['model'].preprocess_models()
- symfluence.managers['model'].run_models()



CUAHSI, Hydroinformatics: Chapter 17 Intro to Modelling

Step 4 — Model execution: jHBV

Table A1. Hydrological and land-surface models supported in SYMFLUENCE.

Model	Language	Key Reference
SUMMA	Fortran	Clark et al. (2015a,b)
FUSE	Fortran	Clark et al. (2008)
HYPE	Fortran	Lindström et al. (2010)
MESH	Fortran	Pietroniro et al. (2007)
mHM	Fortran	Samaniego et al. (2010)
SWAT	Fortran	Arnold et al. (1998)
WRF-Hydro	Fortran	Gochis et al. (2020)
PRMS	Fortran	Markstrom et al. (2015)
CRHM	Fortran	Pomeroy et al. (2007)
WATFLOOD	Fortran	Kouwen (2018)
Wflow (SBM)	Julia	van Verseveld et al. (2024)
RHESSys	C	Tague and Band (2004)
NGEN	C++	Ogden et al. (2026)
HBV	JAX/Python	Lindström et al. (1997)
HEC-HMS	JAX/Python	Feldman (2000)
TOPMODEL	JAX/Python	Beven and Kirkby (1979)
Xinanjiang	JAX/Python	Zhao (1992)
SAC-SMA	JAX/Python	Burnash (1995)
GR4J/GR6J	R (airGR)	Perrin et al. (2003)
Snow17	JAX/Python	Anderson (1973, 2006).
VIC	C	Liang et al. (1994)
CLM5	Fortran	Lawrence et al. (2019)
LSTM	PyTorch	Kratzert et al. (2018)
GNN	PyTorch	—
WMFire	C++	Kennedy et al. (2017)

Table A2. Integrated groundwater and subsurface models.

Model	Type	Language	Subsurface Method	Calib. Params	Key Reference
MODFLOW 6	Groundwater	Fortran	Finite difference	3	Langevin et al. (2017)
ParFlow	Integrated 3-D	C++	Richards (variably sat.)	5	Kollet and Maxwell (2006), (Kuffour et al., 2020)
PIHM	Coupled SW–GW	C	Richards + diffusion wave	7	Qu and Duffy (2007)
CLM-ParFlow	Coupled LSM–GW	Fortran/C++	Richards (ParFlow)	14+	Maxwell et al. (2015)
GSFLOW	Coupled PRMS–MODFLOW	Fortran	MODFLOW-NWT	10+	Markstrom et al. (2008)

Table A3. Routing models.

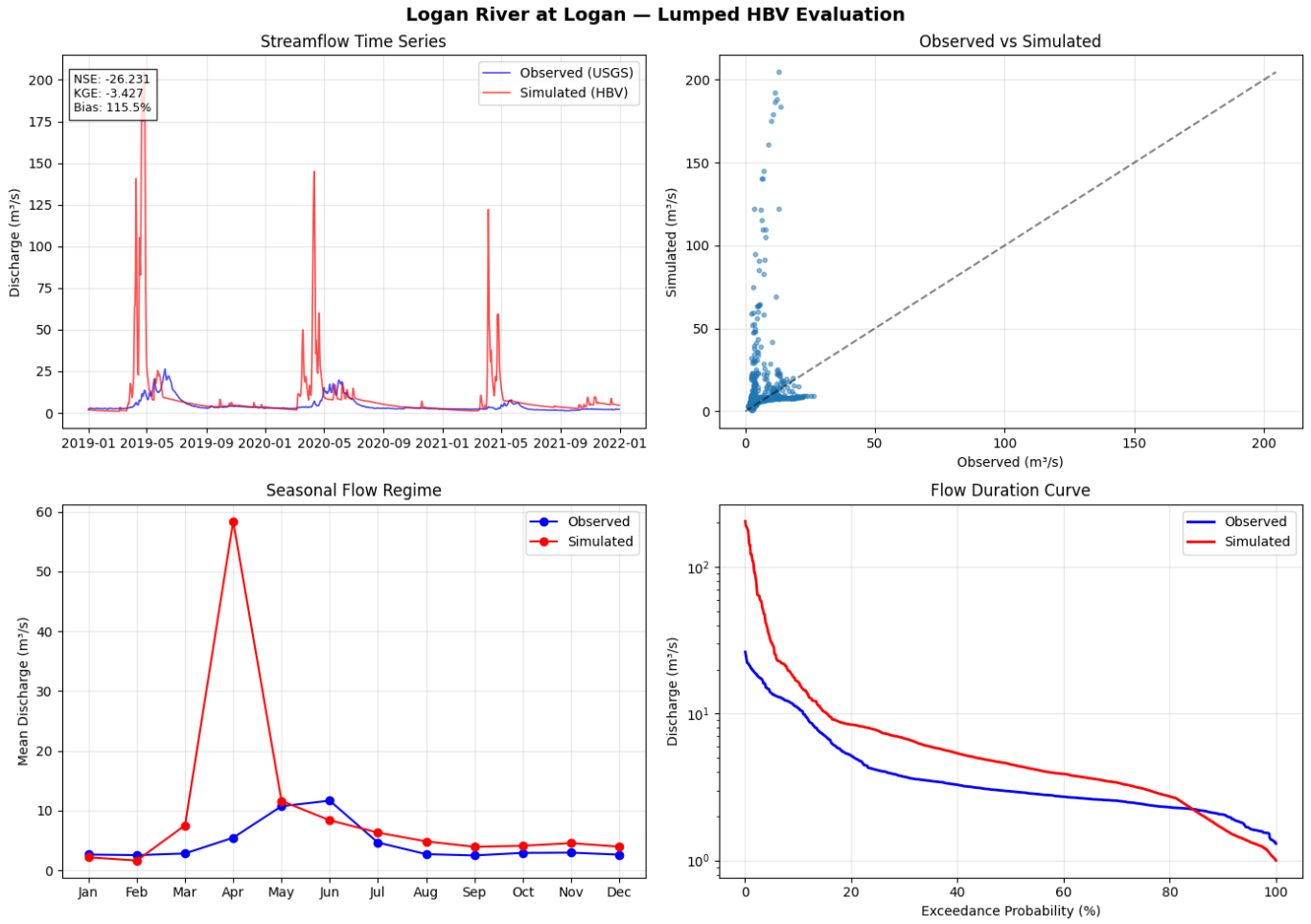
Model	Language	Methods	Calib. Params	Key Reference
mizuRoute	Fortran	IRF, KWT, diffusive wave	2	Mizukami et al. (2016)
T-Route	Python	Muskingum-Cunge, diffusive wave	1	Johnson et al. (2023)

Eythorsson et al., 2026b “The Registry as Social Contract: Architectural Patterns for Community Hydrological Modeling”, *in prep*

Step 5a — Streamflow evaluation (uncalibrated)

- Simulated vs. observed discharge:
  - Metrics (post-spinup):
    - NSE, KGE, PBIAS
  - Diagnostic plots:
- Default parameters
- How much can calibration improve?

Notebook step 5a

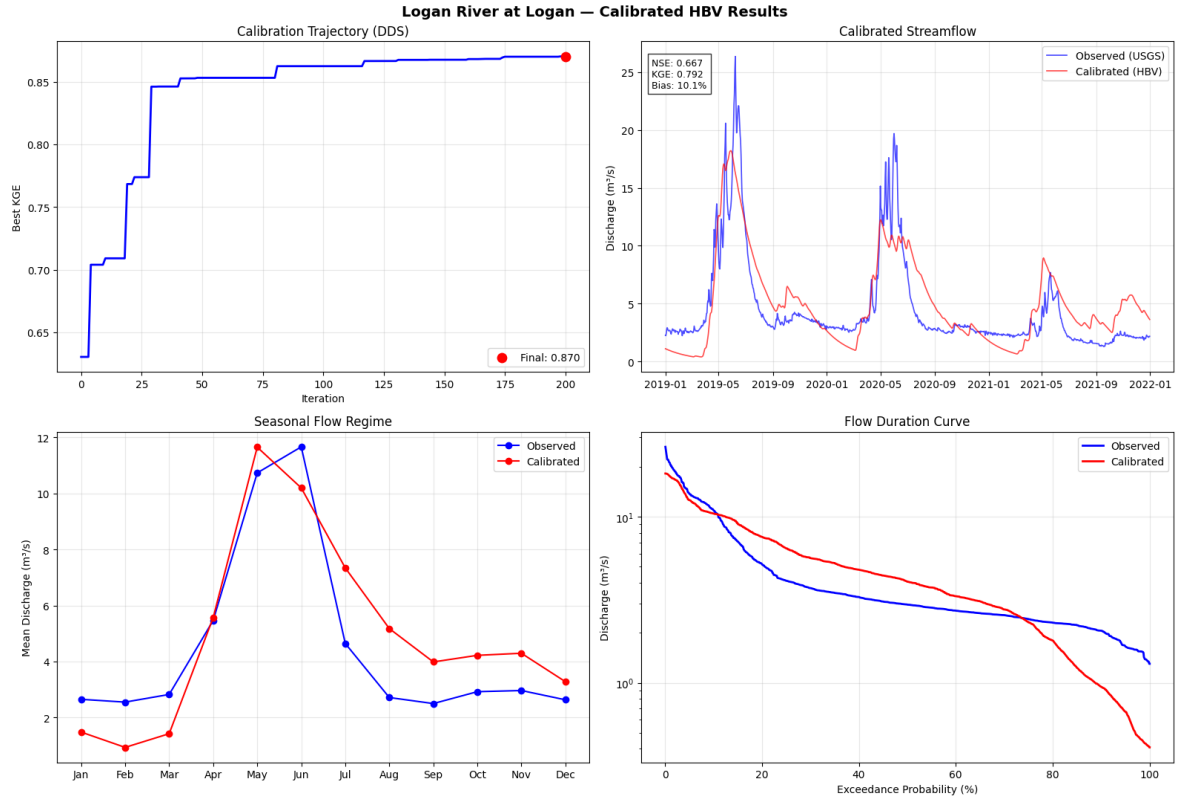


Step 5b — Calibration with DDS

- DDS Dynamically Dimensioned Search (Tolson & Shoemaker, 2007)
  - Global optimization for computationally expensive models
  - Automatically reduces search dimensionality with iterations
- Calibration setup:
  - Objective: maximize streamflow KGE
  - calibration period (2019)
  - Iterations: 200

Notebook step 5b

- symfluence.managers['optimization'].calibrate_model()



Step 5b — Calibration with DDS

Table B2. Evaluation metrics registered in the metrics registry.

Metric	Type	Range	Direction	Key Reference
NSE	Efficiency	(–∞, 1]	Maximize	Nash and Sutcliffe (1970)
log NSE	Efficiency (low flows)	(–∞, 1]	Maximize	Krause et al. (2005)
KGE	Composite (r, α, β)	(–∞, 1]	Maximize	Gupta et al. (2009)
KGE′	Modified composite	(–∞, 1]	Maximize	Kling et al. (2012)
KGE_np	Non-parametric composite	(–∞, 1]	Maximize	Pool et al. (2018)
VE	Volumetric efficiency	(–∞, 1]	Maximize	Criss and Winston (2008)
RMSE	Error magnitude	[0, ∞)	Minimize	—
NRMSE	Normalized error	[0, ∞)	Minimize	—
MAE	Absolute error	[0, ∞)	Minimize	—
MARE	Relative error	[0, ∞)	Minimize	—
Bias	Systematic offset	(–∞, ∞)	0	—
PBIAS	Percent bias	(–100, 100)	0	—
Pearson r	Correlation	[–1, 1]	Maximize	—
R^2	Determination	(–∞, 1]	Maximize	—

Table B3. Variable-specific evaluators and supported observation sources.

Evaluator	Variables	Observation Sources
Streamflow	Discharge	USGS, WSC, GRDC, SMHI, Hub'Eau
Snow	SWE, snow cover	SNODAS, MODIS SCA, SNOTEL, CanSWE
ET	Evapotranspiration	MODIS MOD16, GLEAM, FLUXCOM, FLUXNET
Soil Moisture	Vol. water content	SMAP, ASCAT, ESA CCI, ISMN
Groundwater	Water table depth	Well observations, GRACE-derived
TWS	Total water storage	GRACE/GRACE-FO (JPL, CSR, GSFC)

Table B1. Optimization algorithms available in SYMFLUENCE.

Algorithm	Type	Key Reference
DDS	Derivative-free, single-objective	Tolson and Shoemaker (2007)
SCE-UA	Evolutionary, adaptive	Duan et al. (1992)
PSO	Population-based, metaheuristic	Kennedy and Eberhart (1995)
DE	Differential evolution	Storn and Price (1997)
GA	Genetic algorithm	Holland (1975)
CMA-ES	Covariance matrix adaptation	Hansen (2006)
Nelder-Mead	Simplex	Nelder and Mead (1965)
Basin Hopping	Global + local hybrid	Wales and Doye (1997)
Simulated Annealing	Stochastic	Kirkpatrick et al. (1983)
Bayesian Optimization	Surrogate-based	Snoek et al. (2012)
NSGA-II	Multi-objective evolutionary	Deb et al. (2002)
MOEA/D	Multi-objective decomposition	Zhang and Li (2007)
DREAM	MCMC with differential evolution	Vrugt et al. (2009)
ABC-SMC	Approximate Bayesian Computation	Sisson et al. (2018)
GLUE	Generalized Likelihood Uncertainty	Beven and Binley (1992)
Adam	Gradient-based, adaptive LR	Kingma and Ba (2015)
L-BFGS	Quasi-Newton, gradient-based	Liu and Nocedal (1989)

Eythorsson et al., 2026b "The Registry as Social Contract: Architectural Patterns for Community Hydrological Modeling", *in prep*

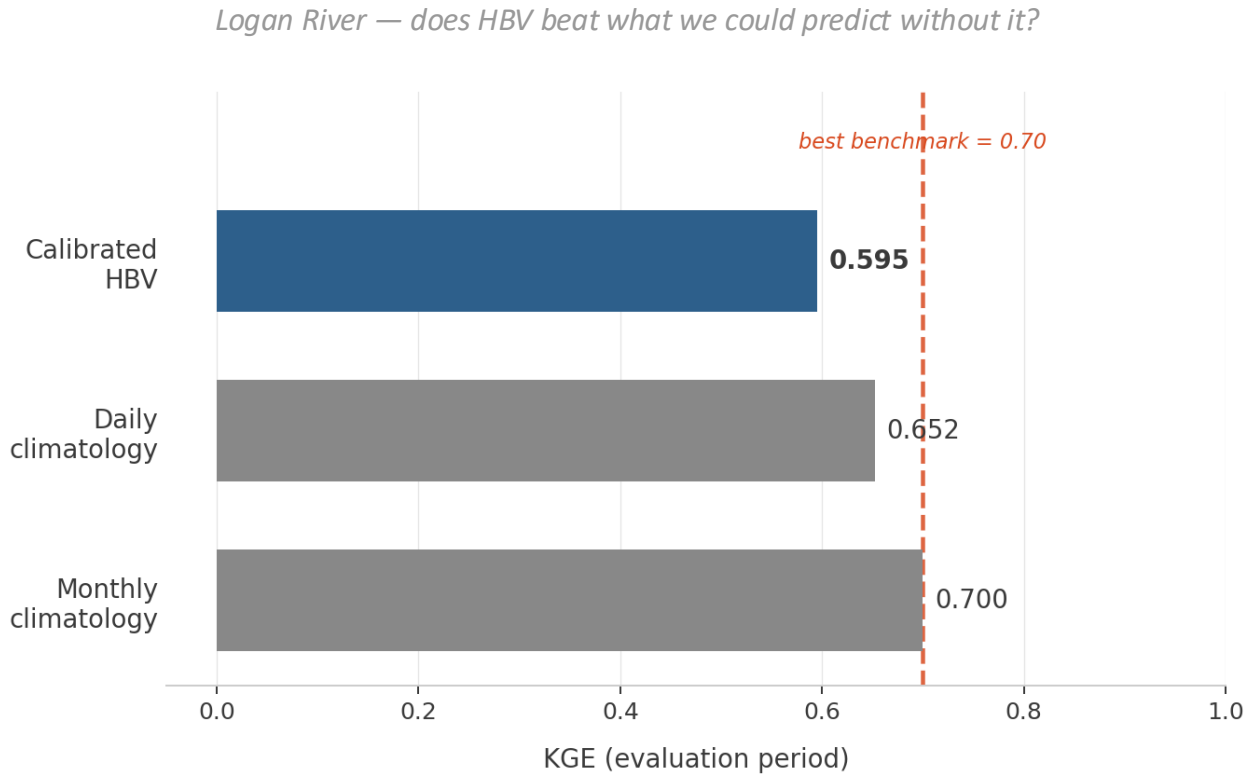
Step 5c — Benchmarking with HydroBM

On Logan River:

Calibrated HBV **0.595**

lost to monthly climatology

Monthly climatology **0.700**



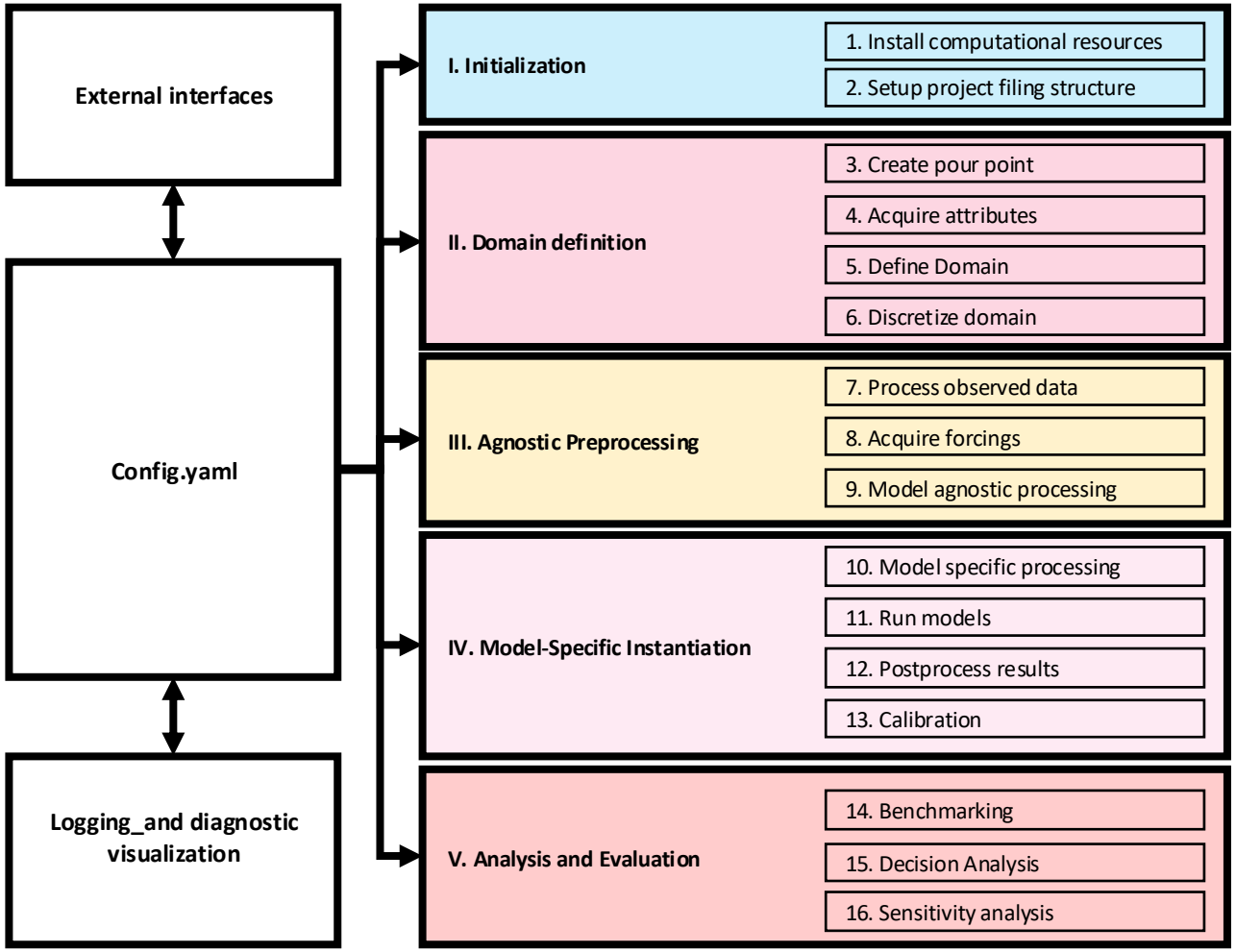
KGE alone tells you fit. Benchmarks tell you whether that fit is real skill — or whether the basin's signal was already in the seasonal cycle.

Part C

Recap and Wrap-Up

Key decision points in the workflow

- Each decision introduces uncertainty:
- 1. Model structure
- 2. Forcing data
- 3. Calibration algorithm
- 4. Spatial discretization
- 5. Evaluation metric



Reproducibility & provenance

- SYMFLUENCE tracks provenance at every stage:
  - Configuration is the experiment specification
  - Data provenance
  - Model provenance
  - Reproducibility = re-run from config alone
- Per domain _workLog documents every step:

```
2026-05-16 12:33:11 - symfluence - INFO - logging_manager - setup_logging - ============================================================
2026-05-16 12:33:11 - symfluence - INFO - logging_manager - setup_logging - SYMFLUENCE Logging Initialized
2026-05-16 12:33:11 - symfluence - INFO - logging_manager - setup_logging - Domain: Logan_River_at_Logan
2026-05-16 12:33:11 - symfluence - INFO - logging_manager - setup_logging - Experiment ID: workshop_run_1
2026-05-16 12:33:11 - symfluence - INFO - logging_manager - setup_logging - Log Level: INFO
2026-05-16 12:33:11 - symfluence - INFO - logging_manager - setup_logging - Log File: /Users/darri.eythorsson/compHydro/SYMFLUENCE_data/domain_Logan_River_at_Logan/
_workLog_Logan_River_at_Logan/symfluence_general_Logan_River_at_Logan_20260516_123311.log
2026-05-16 12:33:11 - symfluence - INFO - logging_manager - setup_logging - ============================================================
2026-05-16 12:33:11 - symfluence - INFO - logging_manager - log_configuration - Configuration logged to: /Users/darri.eythorsson/compHydro/SYMFLUENCE_data/
domain_Logan_River_at_Logan/_workLog_Logan_River_at_Logan/config_Logan_River_at_Logan_20260516_123311.yaml
2026-05-16 12:33:11 - symfluence - INFO - system - __init__ - SYMFLUENCE initialized
2026-05-16 12:33:12 - symfluence - INFO - project_manager - setup_project - Setting up project for domain: Logan_River_at_Logan
2026-05-16 12:33:12 - symfluence - INFO - project_manager - setup_project - Project directory created at: /Users/darri.eythorsson/compHydro/SYMFLUENCE_data/
domain_Logan_River_at_Logan
2026-05-16 12:33:12 - symfluence - INFO - project_manager - create_pour_point - Pour point shapefile created successfully: /Users/darri.eythorsson/compHydro/
SYMFLUENCE_data/domain_Logan_River_at_Logan/shapefiles/pour_point/Logan_River_at_Logan_pourPoint.shp
2026-05-16 12:33:12 - symfluence - INFO - acquisition_service - acquire_attributes - Starting attribute acquisition
2026-05-16 12:33:12 - symfluence - INFO - acquisition_service - acquire_attributes - Cloud data access enabled for attributes (DEM_SOURCE: copdem90)
2026-05-16 12:33:12 - symfluence - INFO - acquisition_service - _run_parallel_tasks - Acquiring attributes: 3 tasks (serial)
2026-05-16 12:33:12 - symfluence - INFO - acquisition_service - _run_parallel_tasks - Starting: DEM
2026-05-16 12:33:12 - symfluence - INFO - base - _skip_if_exists - Using existing file: /Users/darri.eythorsson/compHydro/SYMFLUENCE_data/domain_Logan_River_at_Logan/
data/attributes/elevation/dem/domain_Logan_River_at_Logan_elv.tif
2026-05-16 12:33:12 - symfluence - INFO - acquisition_service - _run_parallel_tasks - Completed: DEM
2026-05-16 12:33:12 - symfluence - INFO - acquisition_service - _run_parallel_tasks - Starting: soil
2026-05-16 12:33:12 - symfluence - INFO - base - _skip_if_exists - Using existing file: /Users/darri.eythorsson/compHydro/SYMFLUENCE_data/domain_Logan_River_at_Logan/
data/attributes/soilclass/domain_Logan_River_at_Logan_soil_classes.tif
2026-05-16 12:33:12 - symfluence - INFO - acquisition_service - _run_parallel_tasks - Completed: soil
2026-05-16 12:33:12 - symfluence - INFO - acquisition_service - _run_parallel_tasks - Starting: landcover
2026-05-16 12:33:12 - symfluence - INFO - landcover - download - Fetching MODIS Land Cover (MCD12Q1 v061) from Zenodo
2026-05-16 12:33:12 - symfluence - INFO - landcover - download - MODIS landcover bbox: {'lat_min': 41.7, 'lat_max': 42.15, 'lon_min': -111.9, 'lon_max': -111.4}
2026-05-16 12:33:12 - symfluence - INFO - acquisition_service - _run_parallel_tasks - Completed: landcover
2026-05-16 12:33:12 - symfluence - INFO - lumped_delineator - delineate_lumped_watershed - Delineating lumped watershed: Logan_River_at_Logan
2026-05-16 12:33:14 - symfluence - INFO - gdal_processor - run_gdal_processing - Completed GDAL polygonization using direct method
2026-05-16 12:33:14 - symfluence - INFO - lumped_delineator - _select_lumped_basin_from_streamnet - Selected outlet stream: WSNO=24 (of 2 candidate segment(s) within 200
m of snapped gauge)
```

A single YAML file defines the complete experiment — 6 sections, 450+ available parameters

config.yaml (YAML)

```yaml
system:
  data_dir:           /project/data
  code_dir:           /project/SYMFLUENCE

domain:
  name:               Bow_at_Banff
  experiment_id:      Bow_lumped_ERA5
  definition_method:  lumped
  discretization:     elevation
  pour_point_coords:  51.17/-115.57
  time_start:         2004-01-01
  time_end:           2007-12-31

forcing:
  dataset:            ERA5
  time_step_size:     3600
  pet_method:         oudin

model:
  hydrological_model: SUMMA
  routing_model:      mizuRoute
  calibrate:  [tempCritRain,
               k_soil, theta_sat]

optimization:
  algorithm:  DDS
  metric:     KGE
  iterations: 1000

evaluation:
  targets:    [streamflow]
  streamflow:
    station_id: 05BB001
```

- **System** — Paths, parallelism, logging, execution
- **Domain** — Spatial extent, time period, discretization
- **Forcing** — Meteorological data source & processing
- **Model** — Hydrological & routing model selection
- **Optimization** — Calibration algorithm & objective function
- **Evaluation** — Observation data & performance metrics

Minimal working example — 10 required parameters shown (bold keys); all others use validated defaults

Eythorsson et al., 2026b “The Registry as Social Contract: Architectural Patterns for Community Hydrological Modeling”, *in prep*

Contributing

Open commons. Structured stewardship.

A USE IT

- pip install symfluence
- Docs: symfluence.readthedocs.io
- GPL-3.0 with commercial licensing option
- Stable on main, integration on develop
- 24 releases — semantic versioning

B EXTEND IT

- Plugin architecture: add models, handlers, and algorithms without forking the core
- PRs welcome to develop
- 99+ test files — CI green required before merge
- Branching: feature/* → develop → main

C SHAPE IT

- Issues: bug reports and feature requests
- Discussions: questions, use cases, design
- CONTRIBUTING.md, GOVERNANCE.md, Code of Conduct

Preview: Session 2 — Parallel Scaling

- In Session 2, we will:
  - Take the setup from Session 1 and scale it
  - Parallel calibration
  - Distributed modelling

num_processes > 1

Thank you.

Every step — acquisition, preprocessing, simulation across models, calibration, evaluation, benchmarking

— designed for portability from laptop to HPC

TRY IT

`pip install symfluence`

Run on your own basin.

Define a pour point.

Build your ensemble

TELL US

Issues: bugs, edges, surprises.

Discussions: design questions, use cases.

PRs: new defaults, new handlers, new algos.

STAY IN TOUCH

Session 2 (afternoon): regionalization and the model-ready data store.

`darri.eythorsson@ucalgary.ca`

`github.com/symfluence-org/SYMFLUENCE` · `symfluence.org` · `symfluence.readthedocs.io` · `darri@symfluence.org`

Appendix I

Registry extendability

Registration mechanisms

Three equivalent ways to hand a class to the registry

1. Direct call

```
from symfluence.core.registries import R

R.runners.add(
    "SUMMA", SummaRunner,
    runner_method="run",
)
```

2. Decorator

```
@R.runners.add(
    "SUMMA",
    runner_method="run",
)
class SummaRunner:
    ...
```

3. Lazy import

```
R.bmi_adapters.add_lazy(
    "SUMMA",
    "symfluence.coupling."
    "adapters.SUMMAProcess"
    "Component",
)
```

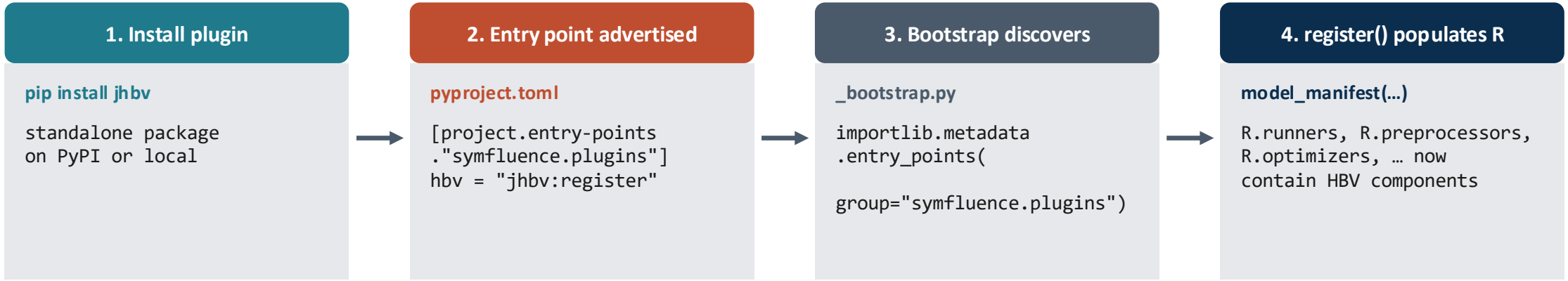
Shortcut for model plugins: model_manifest()

One call registers every base component of a model family —
preprocessor, runner, postprocessor, config adapter,
result extractor, optimizer, worker, parameter manager, …

```
model_manifest(
    "HBV",
    preprocessor    = HBVPreProcessor,
    runner          = HBVRunner,
    runner_method   = "run_hbv",
    postprocessor   = HBVPostprocessor,
    config_adapter  = HBVConfigAdapter,
    optimizer       = HBVModelOptimizer,
    worker          = HBVWorker,
    ...
)
```

How plugins are discovered

Entry points → bootstrap → register() → populated registries



Key properties

- Zero modifications to the SYMFLUENCE core — plugins live in their own repos.
- Plugin failures are logged, never crash the framework bootstrap.
- Works for built-ins too: SUMMA, FUSE, HYPE, NGEN … all register the same way.
- Lazy imports keep startup fast — heavy deps (JAX, torch) load only when used.

Case study — jHBV

A JAX-based HBV-96 model as a standalone SYMFLUENCE plugin: https://github.com/symfluence-org/jhbv/

Package layout

```
jHBV/
├─ pyproject.toml        ← entry point
└─ src/jhbv/
    ├─ __init__.py        ← register()
    ├─ preprocessor.py
    ├─ runner.py
    ├─ postprocessor.py
    ├─ config.py
    ├─ extractor.py
    └─ calibration/
        ├─ worker.py
        ├─ optimizer.py
        └─ parameter_manager.py
```

Inherits from SYMFLUENCE base classes

```
HBVPreProcessor   ← BaseModelPreProcessor
HBVRunner         ← BaseModelRunner + mixins
HBVPostprocessor  ← StandardModelPostprocessor
HBVConfigAdapter  ← AutoGeneratedConfigAdapter
HBVWorker         ← InMemoryModelWorker
```

src/jhbv/__init__.py

```
def register() -> None:
    from symfluence.core.registry import model_manifest
    from .runner           import HBVRunner
    from .preprocessor     import HBVPreProcessor
    from .postprocessor    import HBVPostprocessor
    from .config           import HBVConfigAdapter
    from .extractor        import HBVResultExtractor
    from .calibration.optimizer         import HBVModelOptimizer
    from .calibration.worker            import HBVWorker
    from .calibration.parameter_manager import HBVParameterManager

    model_manifest(
        "HBV",
        preprocessor      = HBVPreProcessor,
        runner            = HBVRunner,
        runner_method     = "run_hbv",
        postprocessor     = HBVPostprocessor,
        config_adapter    = HBVConfigAdapter,
        result_extractor  = HBVResultExtractor,
        optimizer         = HBVModelOptimizer,
        worker            = HBVWorker,
        parameter_manager = HBVParameterManager,
    )
```

Building your own plugin

Four steps — no SYMFLUENCE fork required

1 Create the package

```
my_model/
├─ pyproject.toml
└─ src/my_model/
    └─ __init__.py
```

Standard src-layout Python package.

2 Inherit base classes

```
class MyRunner(
    BaseModelRunner,
    SpatialOrchestrator,
):
    def run(self):
        ...
```

Pick the hooks you need: runner, preprocessor, config adapter, worker.

3 Write register()

```
def register():
  from symfluence.core\
    .registry import \
    model_manifest
  model_manifest(
    "MYMODEL",
    runner=MyRunner,
    ...)
```

Called once, by SYMFLUENCE, at bootstrap.

4 Declare entry point

```
[project.entry-points
."symfluence.plugins"]
mymodel = \
  "my_model:register"
```

pip install . — the model is now discoverable as "MYMODEL" everywhere.

Your model behaves like a built-in: `R.runners["MYMODEL"]` • `symfluence workflow run --model MYMODEL`