LECTURE15 EBVALUATING MODELS

COMPARING MODEL WITH OBSERVATIONS

Why?

- to make a “case” for why you selected this model for your project
- to quantify uncertainty in your model outputs
- to choose parameters that make the model perform well
    - reduce parameter uncertainty

QANTIFYING MODEL PERFORMANCE

- Depends on type of model output
  - single value (mean energy production from a solar panel)
  - time series (streamflow, PM10, monthly energy production)
  - spatial pattern (population, income level)
  - space-time (pollution in different cites over time, forest biomass growth across different forests)
- Depends on observations
  - often not as “dense” as model output (e.g samples)
  - sometimes you have a time series (e.g. streamflow, PM10)
  - sometimes you have a spatial pattern (e.g. population, income level)
  - often you are sampling (e.g mean PM10, mean streamflow) so there is also error/uncertainty in the observation

METRICS

- what is possible given available data
- what is important to get “right” given model application

EXAMPLE (PICKING MODEL BASED ON PERFORMANCE)

Model Comparison

Staudinger, M., Stahl, K., Seibert, J., Clark, M. P., and Tallaksen, L. M.: Comparison of hydrological model structures based on recession and low flow simulations, Hydrol. Earth Syst. Sci. Discuss., 8, 6833-6866, doi:10.5194/hessd-8-6833-2011, 2011

SOMME CLASSIC METRICS

- t.test (are the means different)
- var.test (are the variances different)

Dynamic models (trends over space and time)

- RMSE (Root Mean Square Error)
- Correlation (R^2)
- NSE (Nash-Sutcliffe Efficiency) (variance weighted correlation)

But you can code your own

- make them an R function for utility

NSE

$$NSE = \frac{\sum((m_i - o_i)^2)}{\sum((o_i - mean(o))^2}$$

```r
nse = function(m, o) {
  err = m - o
  meanobs = mean(o)
  mse = sum(err^2)
  ovar = sum((o - meanobs)^2)
  return(1 - mse / ovar)
}
```

PERCENT ERROR

Useful to see if over whole time series you have a bias

Relative Error (%) = $\frac{\bar{m}-\bar{o}}{\bar{o}} \times 100$

```r
relerr = function(m, o) {
  err = m - o
  meanobs = mean(o)
  return(mean(err) / meanobs)
}
```

SOFT METRICS: FUZZY EVALUATION

- Handle uncertainty/imprecise/quantitative data
- Fuzzy membership functions

Many options - here’s one example

$$perf(x) = \begin{cases} 0 & \text{if } x \le a_1 \\ \frac{x - a_1}{a_2 - a_1} & \text{if } a_1 \le x < a_2 \\ 1 & \text{if } a_2 \le x < a_3 \\ \frac{a_4 - x}{a_4 - a_3} & \text{if } a_3 \le x < a_4 \\ 0 & \text{if } x \ge a_4 \end{cases}$$

Seibert, J., and J. J. McDonnell, On the dialog between experimentalist and modeler in catchment hydrology:

Use of soft data for multicriteria model calibration, Water Resour. Res., 38(11), 1241,

doi:10.1029/2001WR000978, 2002

EXAMPLE APPLICATION

For a given year, we just know whether the water manager reported a drought
For our watershed

- if streamflow was less than 800 mm/year then for sure it was a drought
- if streamflow was between 800 and 1000 mm/year then it was a drought with some uncertainty
- if streamflow is above 1000 mm/year then it was not a drought

But all we know is the report for each year
Can we evaluate our model performance given this uncertainty?
Two Conditions * drought * a1=0, a2=0, a3=800, a4=1000 * OK * upper bound just pick really large numbers * a1=800, a2=1000, a3=10000, a4=10000

```r
library(tidyverse)
library(here)
source(here("R/fuzzy_perf.R"))
w8_wy = readRDS(here("Data/w8_wy.RDS"))

# first year
w8_wy[1,"report"]
```

```
# A tibble: 1 × 1
  report
  <chr>
1 ok
```

```r
fuzzy_perf(w8_wy$model[1], 800, 1000, 1000
```

```
[1] 0.74021
```

```r
w8_wy[1,"report"]
```

```
# A tibble: 1 × 1
  report
  <chr>
1 ok
```

```r
fuzzy_perf(w8_wy$model[3], 0, 0, 800, 1000
```

```
[1] 1
```

```r
w8_wy$perf = 0
for (i in 1:nrow(w8_wy)) {
  w8_wy$perf[i] = ifelse(w8_wy$report[i] =
    fuzzy_perf(w8_wy$model[i], 0, 0, 800,
    fuzzy_perf(w8_wy$model[i], 800, 1000,
}
```

COMBINING METRICS

- Normalize metrics (0-1)
- Make all increase with performance
- Weighted sum or multiplicative approach

$Metric_A * Metric_B * Metric_C$
$Metric_A * weight_A + Metric_B * weight_B + Metric_C * weight_C$
where $weight_A + weight_B + weight_C = 1$

NORMALIZING

SSE = $1/n * \sum((m_i - o_i)^2)$
How do I make this 0-1? and increasing values give better performance

SOLUTION OPTIONS

$L = (SSE)^{-n}$
$L = exp(-n * SSE)$
$L = (max(SSE) - SSE)/max(SSE)$
where $max(SSE)$ is the maximum value of SSE across all models or sometimes
$L = (SSE - min(SSE))/(max(SSE) - min(SSE))$

COMBINED PERFORMANCE

Example R function

Lets go back to daily streamflow data

```r
source(here("R/nse.R"))
source(here("R/relerr.R"))
source(here("R/cper.R"))

w8 = readRDS(here("Data/w8.RDS"))
```