Handling Messy Clinical Data

Lessons from the WashU I2DB Datathon

Subrata Pal*

with Sayan Das†, Ayoushman Bhattacharya†, Subhrajyoty Roy† (Team **SASS**)

Capstone Course, Health Data Science, Saint Louis University

June 17, 2026

*DRIVES lab, Department of Neurology
†Department of Statistics and Data Science
Washington University in St. Louis

I2DB Datathon: What's your first move?

We have 2024 longitudinal EHR health records with:

- $\sim$36 000 patients, $\sim$90 / 10 % controlled / uncontrolled
- Demographics, lab values, medications, visit records
- Irregular time-series: 1–7 A1c readings per patient

Goal: Predict whether a diabetes patient's HbA1c will be uncontrolled (?!) in 2025.

I2DB Datathon: What's your first move?

We have 2024 longitudinal EHR health records with:

- ∼36 000 patients, ∼90 / 10 % controlled / uncontrolled
- Demographics, lab values, medications, visit records
- Irregular time-series: 1–7 A1c readings per patient

Goal: Predict whether a diabetes patient's HbA1c will be uncontrolled (?!) in 2025.

You're handed this dataset. What do you do first?

EDA? Feature engineering? Modeling? Google "HbA1c"? Ask ChatGPT?

What does “uncontrolled” even mean?

HbA1c = glycated hemoglobin: roughly your average blood glucose over the past 2 to 3 months.

ADA clinical intervention threshold: A1c > 7 (doctors start acting)

The competition says “predict uncontrolled diabetes.”

What threshold would you use?

What does "uncontrolled" even mean?

HbA1c = glycated hemoglobin: roughly your average blood glucose over the past 2 to 3 months.

ADA clinical intervention threshold: A1c > 7 (doctors start acting)

The competition says "predict uncontrolled diabetes."

What threshold would you use?

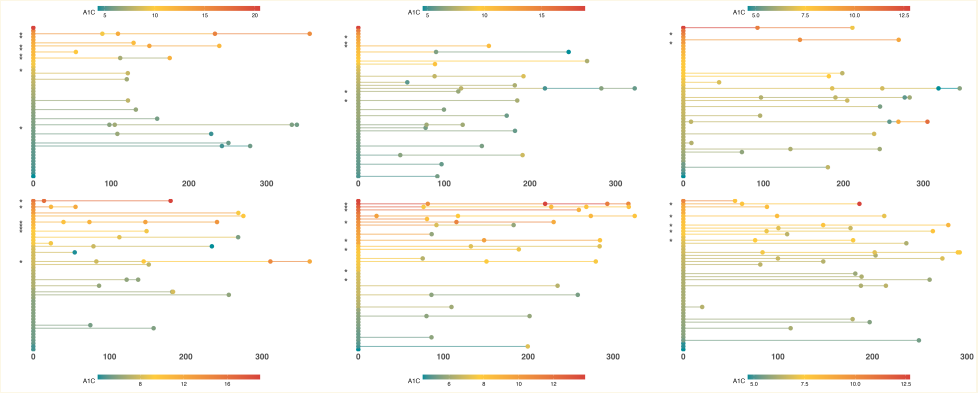
We used 7. For days.

Competition definition: A1c $> \mathbf{8}$.

Read. The. Problem. Statement.

Domain knowledge *and* problem definition, before any analysis.

The landscape: confused?



Each line = one patient's A1c trajectory. Color = A1c level. Six panels: age × gender.

What does messy data look like?

Three things the raw data threw at us in the first hour:

- **Out-of-order timestamps:** patient 43's 4th visit logged *before* the 3rd.
- **Same day, wildly different A1c:** 6.2 vs 9.1, readings a day apart.
- **Implausible BMI:** values over 300. (filter to 10–80)

Always look at the raw data before computing anything.

Missing isn't nothing

Half our patients are missing height, weight, or labs. The textbook move: **impute**. Sayan ran MICE: it fills each gap from the other columns.

Missing isn't nothing

Half our patients are missing height, weight, or labs. The textbook move: **impute**. Sayan ran MICE: it fills each gap from the other columns.

We did. Then tried something dumber: a flag for *whether* each value is missing.

The missing-ness flag beat the imputed value.

Missing isn't nothing

Half our patients are missing height, weight, or labs. The textbook move: **impute**. Sayan ran MICE: it fills each gap from the other columns.

We did. Then tried something dumber: a flag for *whether* each value is missing.

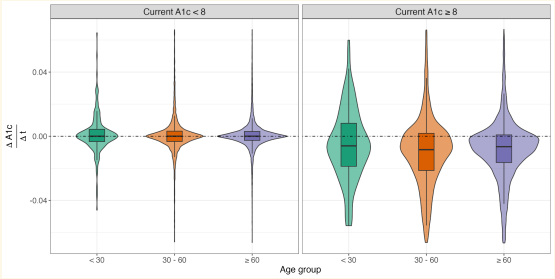
The missing-ness flag beat the imputed value.

"Didn't get measured" isn't noise: it's how *engaged* a patient is with their care. The absence *is* the signal.

And 58 % have just **one** A1c reading: sparse observation, not missing data. MICE can't invent a clinic visit that never happened.

Spot the pattern

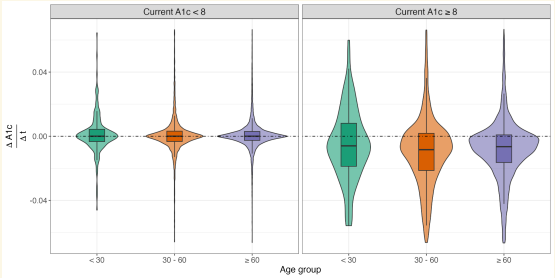
A1c change between consecutive visits, split by current A1c < 8 vs ≥ 8.



Left panel vs right: what's different?

Spot the pattern

A1c change between consecutive visits, split by current A1c < 8 vs ≥ 8.

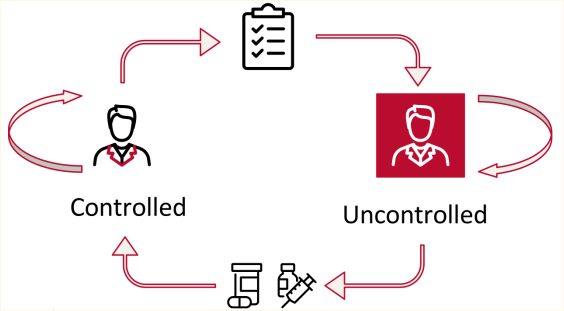


Left panel vs right: what's different?

Below 8: symmetric, up or down. $\geq$ **8:** skewed *down*, treatment pulling back.

Symmetry and spread: two internal behaviors, not one.

Two behaviors → two heads

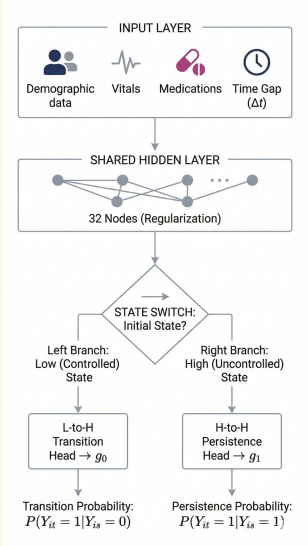


The clinical cycle: monitoring catches deterioration, treatment pulls patients back.

Two distinct pathways, two different processes:

- **L→H** (transition): controlled patient deteriorates despite monitoring
- **H→H** (persistence): uncontrolled patient stays uncontrolled despite treatment

Two heads → the model



Shared hidden layer, two separate output heads:

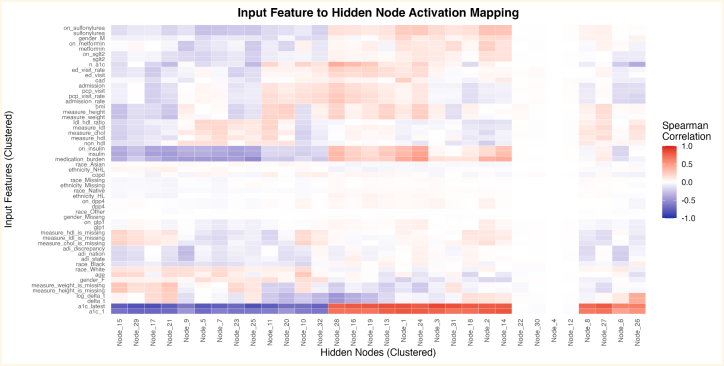
$$P(Y_{it} = 1 \mid Y_{is} = k) \;=\; \sigma\big(g_k(\mathbf{x}_i,\; t - s)\big)$$

- $k = 0$: transition!
- $k = 1$: persistence!

$\mathbf{x}_i$ = patient features. $\quad t - s$ = time gap. $\quad \sigma(z) = \frac{1}{1+e^{-z}}$.

Didn't beat the 3-line GLM. But it taught us *which features matter...*

Opening the black box



Each cell: an input feature's correlation with a hidden-node activation. Bottom rows (`a1c_latest`, `a1c_1`, `delta_t`) blaze red: the NN *independently* found that A1c history and time gap are the core signal.

Design a feature from a biological point

You have 5 A1c readings over 2 years. You need **one number** to predict the future.

What would you compute?

Think 30 seconds. Discuss with your neighbor.

Design a feature from a biological point

You have 5 A1c readings over 2 years. You need **one number** to predict the future.

What would you compute?

Think 30 seconds. Discuss with your neighbor.

Mean? Latest? Max? Slope? All reasonable.

What worked best: **EWMA**: exponentially weighted moving average.

$$a_{\text{weighted}} = \frac{\sum_{k=1}^{n} \lambda^{n-k} a_k}{\sum_{k=1}^{n} \lambda^{n-k}}, \qquad \lambda = 0.5$$

Most recent reading: weight 1. Previous: 0.5. Before that: 0.25, ... A1c = 10 two years ago but 6.5 last month, which matters more?

90 / 10: one thing to remember!

Always predict "controlled" $\rightarrow$ **90 % accuracy**, but F1 $= 0$. Accuracy is the wrong yardstick.

Three fixes for imbalance:

- **Threshold tuning:** sweep the cutoff, not 0.5
- **Class weights:** upweight the rare class in the loss
- **Resampling** (up / down / SMOTE): re-balances the classes; none beat a tuned threshold

F1 rewards *who's on top*, not the probability you assign. Ranking $>$ calibration.

The 3-line model

```
fit  <- glm(a1c_status_2025 ~ a1c_weighted,
            family = binomial, data = train)
pred <- predict(fit, val, type = "response")
```

$\mathrm{F1} = 61.4\,\%$

Three lines of R. One predictor. Logistic regression.

Your baseline: everything else must *beat* it.

Biological feature hunting

Feature	Clinical reasoning
Recency-weighted avg (EWMA)	Summary weighted toward the present
First A1c reading	Baseline severity, before treatment
Fraction of readings > 8	Chronic uncontrolled vs transient spikes
A1c per treatment load	Treatment-resistant: high A1c *despite* medications
Days since last A1c	Information staleness (ADA: 3–6 mo monitoring)
Biggest single-visit worsening	Sudden jump $\neq$ slow drift
Drift rate while controlled	Below 8: creeping up, flat, or dropping?
Number of A1c readings	Sicker or more engaged patient

Translating domain knowledge into numbers.

One feature?

Model	Features	Val F1 (imbalanced data)
GLM (EWMA only)	1	61.4 %
XGBoost (full)	74	62.2 %

One feature?

Model	Features	Val F1 (imbalanced data)
GLM (EWMA only)	1	61.4 %
XGBoost (full)	74	62.2 %

One number, a weighted average of past A1c, captures most of the achievable signal.

The question is no longer "which model?" It's "*why* is one feature quite good enough, and what would it take to go further?"

Every model, similar ceiling

We tried everything: GLM, glmnet, GAM, Decision Tree, Random Forest, XGBoost, Bayesian hierarchical, Neural Network, CTMC etc.

What's the spread in F1 across all of them? 20 percentage points? 10? 5?

Every model, similar ceiling

We tried everything: GLM, glmnet, GAM, Decision Tree, Random Forest, XGBoost, Bayesian hierarchical, Neural Network, CTMC etc.

What's the spread in F1 across all of them? 20 percentage points? 10? 5?

1.3 percentage points.

60.9 % to 62.2 %.

Three completely different statistical paradigms. Same wall.

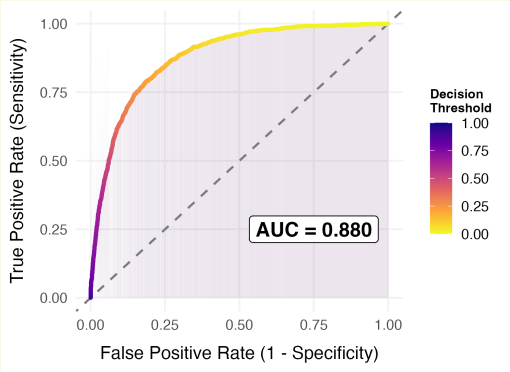
Even the other teams' very complex models like transformers hit it.

These F1s are on returned patients only (the hard ∼18 % subset); non-returned patients are trivially controlled.

What drives predictions?

#	Feature	Gain
1	Days since last A1c	35.8 %
2	EWMA of A1c	20.4 %
3	Most recent A1c	8.4 %
4	First A1c reading	6.2 %
5	Frac above 8	4.7 %
6	Num. drug classes	∼3 %
7	BMI	∼2 %
8	Insulin rate	0.7 %

Top 5 are all A1c: 75.5 % of gain.
Medications, BMI: collectively ∼25 %.



Val AUC = 0.880 (returned patients only)

Add non-returned (trivially $P = 0$): 0.938

Trust but verify

Your teammate: "Random Forest gets **63 % F1**: new best!" Do you celebrate?

Trust but verify

Your teammate: "Random Forest gets **63 % F1**: new best!" Do you celebrate?

We almost did. Then read the code: `predict()` ran on the *training* set, not validation.

Done right, the Random Forest gets 61.6 %, tied with everything else.

Audit your results. Audit your teammates' results. Especially the ones that look too good.

Imagine you know nothing about a patient except their average A1c.

How much information have you missed?

Imagine you know nothing about a patient except their average A1c.

How much information have you missed?

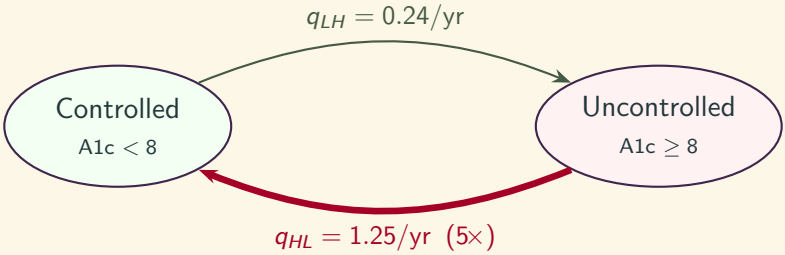
Only 8 %.

92 % of variation is *between* patients, not within.

- Who a patient *is* matters far more than what happened recently
- Fancy longitudinal models can only work with that 8 %
- EWMA is a patient-level summary. That's why one feature = 99 % of signal.

Technically: Intraclass Correlation Coefficient (ICC) = 0.924.

Each visit: "controlled" (< 8) or "uncontrolled" (≥ 8). How fast do they switch? Count crossings, divide by time in each state.



Treatment pulls people down 5× faster than they drift up.

After ≈ 8 months, old data is no better than guessing.

Saw patient 2 months ago? Informative. 18 months ago? You're back to the base rate.

What's missing?

We're stuck at 62 %. We have A1c, meds, demographics, labs.

What information would you WANT that isn't here?

60 seconds. Brainstorm.

What's missing?

We're stuck at 62 %. We have A1c, meds, demographics, labs.

What information would you WANT that isn't here?

60 seconds. Brainstorm.

What's not in the charts

- Medication adherence
- Diet and nutrition
- Physical activity
- Treatment timing (start/stop)
- Financial / life disruptions
- Mental health and stress

The wall = what the patient does *between visits*.

More on interpretability

In this case:

clean the mess $\rightarrow$ honest feature(s) $\rightarrow$ one ceiling, many models $\rightarrow$ *why*

. . . all easily or moderately interpretable.

More on interpretability

In this case:

clean the mess $\rightarrow$ honest feature(s) $\rightarrow$ one ceiling, many models $\rightarrow$ *why*

. . . all easily or moderately interpretable.

But what if the black box works very well, and you can't read it?

Can you trust a model you can't read?

What can you do?

Black box or glass box?

Detecting **near-miss car crashes** from accelerometer data. A 1.7M-parameter deep net nails it, and so does a model you can read.

GLM (your basic model)

$$g(p) = \beta_0 + \beta_1 x$$

Your 3-line model: one coefficient per feature.

Black box or glass box?

Detecting **near-miss car crashes** from accelerometer data. A 1.7M-parameter deep net nails it, and so does a model you can read.

GLM (your basic model) **EBM (glass box)**

$$g(p) = \beta_0 + \beta_1 x \longrightarrow \boldsymbol{g(p) = \beta_0 + \sum_j f_j(x_j) + \sum_{j<k} f_{jk}(x_j, x_k)}$$

Each term becomes a *learned curve*, plus a few interactions: still readable.

Black box or glass box?

Detecting **near-miss car crashes** from accelerometer data. A 1.7M-parameter deep net nails it, and so does a model you can read.

GLM (your basic model) **EBM (glass box)** deep net (black box)

$$g(p) = \beta_0 + \beta_1 x \longrightarrow \boldsymbol{g(p) = \beta_0 + \sum_j f_j(x_j) + \sum_{j<k} f_{jk}(x_j, x_k)} \rightarrow g(p) = \sigma(W_L \cdots \sigma(W_1 x))$$

Green: can read every term. **Red**: x runs through L nested layers: *no per-feature term* to plot.

Black box or glass box?

Detecting **near-miss car crashes** from accelerometer data. A 1.7M-parameter deep net nails it, and so does a model you can read.

GLM (your basic model) **EBM (glass box)** deep net (black box)

$$g(p) = \beta_0 + \beta_1 x \longrightarrow \boldsymbol{g(p) = \beta_0 + \sum_j f_j(x_j) + \sum_{j<k} f_{jk}(x_j, x_k)} \rightarrow g(p) = \sigma(W_L \cdots \sigma(W_1 x))$$

Green: can read every term. **Red**: x runs through L nested layers: *no per-feature term* to plot.

Accelerometer only	AP
Peak-g baseline	89.6
Deep net (1.7M params)	97.6
EBM glass box	**96.9**
+ gyro + speed	**97.7**

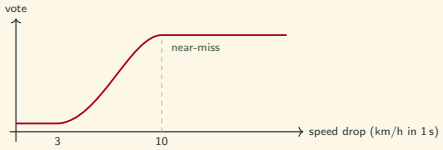
0.7 AP is the entire price of a model you can read.

Give it two channels the network ignored, and it's free.

Similar accuracy. One you can read, one you can't. Why choose the one you can't?

What the glass box teaches you

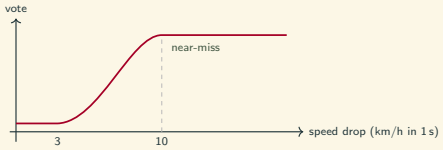
Each curve is one feature's vote: up = near-miss, down = normal.



Discovery. The model hands experts a number: > 10 km/h in 1 s is the danger line.

What the glass box teaches you

Each curve is one feature's vote: up = near-miss, down = normal.



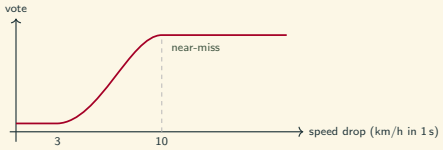
Discovery. The model hands experts a number: > 10 km/h in 1 s is the danger line.

Then read the **misclassifications**: **steering-type** near-misses slip through far more than **brake-type** (recall **48 %** vs **73 %**): a hard swerve looks like an ordinary corner.

Some domain knowledge, some common sense, some transparency: that's how you *find* a blind spot.

What the glass box teaches you

Each curve is one feature's vote: up = near-miss, down = normal.



Discovery. The model hands experts a number: > 10 km/h in 1 s is the danger line.

Then read the **misclassifications**: **steering-type** near-misses slip through far more than **brake-type** (recall **48 %** vs **73 %**): a hard swerve looks like an ordinary corner.

Some domain knowledge, some common sense, some transparency: that's how you *find* a blind spot. Then you fix that blind spot: a feature for the swerve, or better data. A black box might have the same blind spot, but never shows you where.

More data, or the right data?

That crash data was collected at **100 Hz**: 100 readings per second. Our target fleet records at only 24 Hz. Disaster?

More data, or the right data?

That crash data was collected at **100 Hz**: 100 readings per second. Our target fleet records at only 24 Hz. Disaster?

Sampling rate	Average Precision
100 Hz	0.6775
24 Hz	0.6778

Zero difference. A near-miss lasts 0.5–2 s: you don't need 100 Hz to see a 1-second event. The extra resolution was *overkill*: data the model never used.

More data, or the right data?

That crash data was collected at **100 Hz**: 100 readings per second. Our target fleet records at only 24 Hz. Disaster?

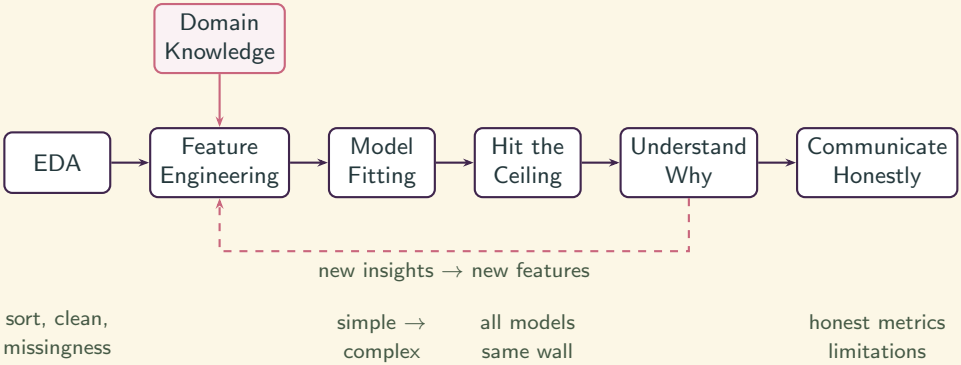
Sampling rate	Average Precision
100 Hz	0.6775
24 Hz	0.6778

Zero difference. A near-miss lasts 0.5–2 s: you don't need 100 Hz to see a 1-second event. The extra resolution was *overkill*: data the model never used.

The real question is never *how much* data: it's *which*.

And we only knew 24 Hz was enough by trying both: you often find the right scale from a pilot or the data, not before it.

The workflow



The workflow matters more than any single model.

Many textbooks start at step 3. Most of the value is in steps 1–2 and 4–6.

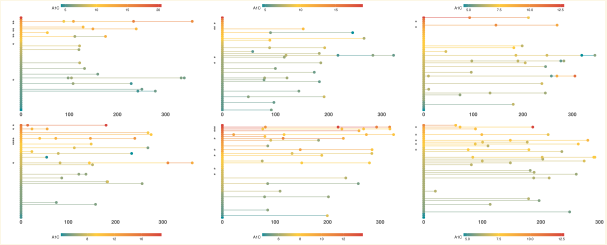
What we learned

1. **Look at the data, get a feel** before computing anything
2. **Missingness might be important**: missing readings = engagement proxy
3. **Start small, understand it**: the 3-line model
4. **The ceiling is usually the data, not the model**: every model, similar wall. (Exception: huge data and a model built for its structure.)
5. **Audit everything**: your teammates' AND YOUR OWN
6. **Read the problem statement**: 7 vs 8 cost us days
7. **Glass box over black box**: similar accuracy, and it shows its blind spots
8. **The right data beats more data most of the time**: match the feature to the signal

ggkodom: the plot that became a package

Ayoushman's EDA plot (every patient's A1c journey as one colour-coded lane) became a ggplot2 extension.

```
ggplot(a1c_long, aes(x = time, y = patient, colour = a1c)) +
  geom_kodom_line()
```



One grammar, many views:

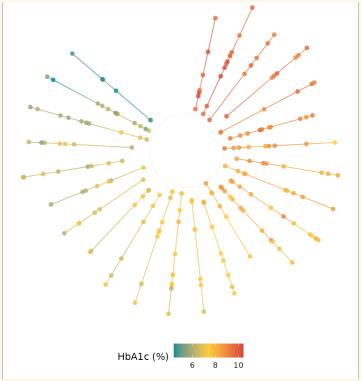
- `geom_kodom_line()`: swimlane
- `geom_kodom_heatmap()`: heatmap
- `geom_kodom_branch()`: counterfactual A1c per drug arm

. . . and the same grammar bends into a circle.

`github.com/subroy13/ggkodom` · submitted to CRAN · all four of us as authors.

Same grammar, now radial

Same patients, same data. Change one geom, and time wraps around the clock.



`geom_kodom_circular()`: each spoke a patient, sorted by control. The *Kadam* flower.

`geom_kodom_periodic()`: months around the dial, the year spirals outward.

A whole cohort's glycemic history in one figure: teal in control, red not.

SASS

Sayan Das

d.sayan@wustl.edu

Ayoushman Bhattacharya

abhattacharya@wustl.edu

Subhrajyoty Roy

roy.s@wustl.edu

Subrata Pal

s.pal@wustl.edu

github.com/subroy13/ggkodom

Appendix: two-head NN in torch

```
model <- nn_module(
  initialize = function(p, h = 64) {
    self$shared <- nn_linear(p, h)
    self$head0  <- nn_linear(h, 1)   # L->H
    self$head1  <- nn_linear(h, 1)   # H->H
  },
  forward = function(x, k) {
    z <- torch_relu(self$shared(x))
    out <- torch_where(k == 0,
              self$head0(z), self$head1(z))
    torch_sigmoid(out)
  }
)
```

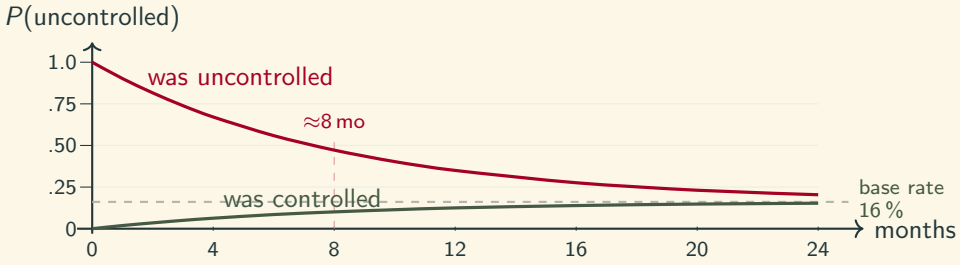
`k` selects which head fires. Same shared layer, different output weights per regime.

Appendix: Continuous-Time Markov Chain (CTMC)

Two states: L (controlled, <8) and H (uncontrolled, ≥ 8).

Rate = how many patients cross / total time spent in that state (per year):

$L \to H$ (deterioration): $q_{LH} = 0.24/\text{yr}$ $\quad$ $H \to L$ (treatment): $q_{HL} = 1.25/\text{yr}$ $\quad$ ($5\times$ faster)



After $\tau = 1/(q_{LH} + q_{HL}) \approx 8$ months, both converge to the base rate (16 %): the starting state is forgotten.

Appendix: the hero feature (EWMA)

```
# Recency-weighted A1c  (EWMA, lambda = 0.5)
# one number captures 99% of the achievable signal
a1c_weighted = case_when(
  !is.na(value_a1c_5) ~ {                  # 5 readings
    w <- 0.5^(4:0); w <- w / sum(w)        # 16:8:4:2:1, normalized
    w[1]*value_a1c_1 + w[2]*value_a1c_2 + w[3]*value_a1c_3 +
      w[4]*value_a1c_4 + w[5]*value_a1c_5
  },
  # 4 readings: same idea with 0.5^(3:0)
  !is.na(value_a1c_3) ~ 0.25*value_a1c_1 + 0.25*value_a1c_2 + 0.50*value_a1c
      _3,
  !is.na(value_a1c_2) ~ 0.50*value_a1c_1 + 0.50*value_a1c_2,
  TRUE                ~ value_a1c_1        # single reading -> itself
)
```

Most recent reading weight 1, previous 0.5, then 0.25, ... and it degrades gracefully as readings disappear.

Appendix: missingness as a feature

```
impute_and_flag <- function(df) {
  flag_cols <- c("a1c_sd", "a1c_range", "value_hdl", "value_ldl",
                 "value_height", "value_weight", "bmi", "time_gap")  # ...

  # 1. record WHAT was missing  (the engagement signal)
  for (col in flag_cols)
    df[[paste0(col, "_miss")]] <- as.integer(is.na(df[[col]]))

  # 2. THEN fill the holes with the column median
  for (col in feature_cols) {
    med <- median(df[[col]], na.rm = TRUE)
    df[[col]] <- ifelse(is.na(df[[col]]), med, df[[col]])
  }
  df
}
```

The `_miss` flags beat the imputed values: "didn't get measured" is a proxy for how engaged a patient is with their care.

Appendix: threshold tuning for F1

```
# 90/10 data: the F1-best cutoff is NOT 0.5
best_f1_threshold <- function(pred_probs, target, mask = NULL) {
  th_list   <- seq(0.01, 0.99, 0.005)
  f1_scores <- numeric(length(th_list))
  for (i in seq_along(th_list)) {
    pred <- factor(pred_probs >= th_list[i], levels = c(FALSE, TRUE))
    f1_scores[i] <- compute_metrics(pred, target, mask,
                                    verbose = FALSE)$metrics["F1"]
  }
  c(best_th = th_list[which.max(f1_scores)],
    best_f1 = max(f1_scores))
}
```

Sweep every cutoff, keep the one that maximizes positive-class F1. A default 0.5 throws away most of the rare class.

Appendix: regularized XGBoost + the mask trick

```
params_reg <- list(
  objective        = "binary:logistic",
  max_depth        = 5,       # shallow trees
  eta              = 0.05,    # slow learning
  subsample        = 0.5,     # row sampling
  colsample_bytree = 0.5,     # column sampling
  min_child_weight = 10,
  gamma            = 1        # min loss reduction to split
)
m <- xgb.train(params_reg, xgb_train, nrounds = 500,
               early_stopping_rounds = 30)

# non-returned patients are trivially controlled -> P = 0
p_test <- ifelse(df_test$mask, predict(m, xgb_test), 0)
```

Heavy regularization stops 74 features from overfitting. The mask's free true-negatives are why test AUC (0.938) tops our honest 0.880.

Appendix: Bayesian state-space (JAGS)

```
model {
  prec_Y      ~ dgamma(8, 10)       # obs precision
  alpha_delta ~ dnorm(0.4, 1)       # log treatment effect
  alpha_prec  ~ dnorm(6.0, 1)       # log latent precision
  for (i in 1:N_patients) {         # covariates drive each patient
    log(delta[i])    <- alpha_delta + inprod(X[i,], beta_delta)
    log(prec_mu[i]) <- alpha_prec  + inprod(X[i,], beta_prec)
  }
  for (j in 1:N_subseq) {           # Markov: treatment lowers A1c
    expected_mu[j] <- mu[j-1] - delta[id[j]] * med_flag[j]
    mu[j] ~ dnorm(expected_mu[j], prec_mu[id[j]] / dt[j])
  }
  for (k in 1:N_total) {            # noisy + threshold-censored
    Y[k] ~ dnorm(mu[k], prec_Y);    is_above_A[k] ~ dinterval(Y[k], A_star)
  }
}
```

Latent A1c = Markov chain, treatment lowers it, observation is threshold-censored. F1 = 0.616, the same band.