Proposed Methods Outline: Modeling Basin–to–Basin Stray Rates in *Oncorhynchus tshawytscha*

1. Data Structure and Preprocessing

Okay, first things first—let's be clear on what the rows in our spreadsheet actually mean. I'm assuming we have tagged Chinook salmon ($N = n_{\text{tot}}$ individuals) that were released as juveniles in some river and then recovered as adults somewhere else (or maybe the same place). We care about the binary idea of "did you stray out of your home basin?" and nothing subtler than that—at least for now.

We observe brood years $t \in \{t_{\min}, \ldots, t_{\max}\}$ and recoveries in rivers j that live inside bigger catchments (basins) k. So for bookkeeping:

$$i = 1, \ldots, n_{\text{tot}}, \qquad j = 1, \ldots, J, \qquad k = 1, \ldots, K, \qquad t = 1, \ldots, T,$$

with the obvious mapping helpers $\text{river}(i) = j(i)$, $\text{basin}(i) = k(i)$, and $\text{year}(i) = t(i)$.

The generative statement I would propose is:

$$y_i \mid p_{j(i),t(i)} \sim \text{Bernoulli}\big(p_{j(i),t(i)}\big), \tag{1}$$

where y_i is 1 if the fish strayed and $p_{j,t}$ is the stray *probability* for river j and brood year t.

Group-level alternative (when counts exist)

Obviously I don't know, but it's worth it in my mind to coonsider the possibility where the raw files come with "25 strays out of 87 returns" for each river–year combo. If so, it's cheaper to write

$$\begin{aligned} m_{j,t}^{\text{stray}} \mid p_{j,t} &\sim \text{Binomial}\big(n_{j,t},\ p_{j,t}\big), \\ n_{j,t} &= m_{j,t}^{\text{stray}} + m_{j,t}^{\text{return}}, \end{aligned} \tag{2}$$

and avoid looping over individual fish. It's the same likelihood, you're just collapsing rows.

So what do we actually have to do to the CSV?

- **Filter the rubbish.** A bunch of tag IDs are duplicated or impossible (maybe sometimes they literally say "XXXX"). We drop those early so they never poison a model run.
- **Track missing strata.** If a basin–year has zero records we'll store a flag $r_{j,t} = 1$. Later, Stan can treat that as "no data, but the random effect still exists."
- **Quick-and-dirty exploration.** I like the idea of computing a raw basin mean

$$\hat{p}_k^{\text{raw}} = \frac{\sum_{j \in k} \sum_t m_{j,t}^{\text{stray}}}{\sum_{j \in k} \sum_t n_{j,t}} \tag{3}$$

  plus a year-to-year SD

$$\hat{\sigma}_{\text{year}} = \text{sd}_t\big(\hat{p}_{\cdot,t}^{\text{raw}}\big), \tag{4}$$

  just to see if there's any signal worth chasing.

- **Balance across basins.** Take each basin's share of the overall sample,

$$w_k = \frac{\sum_t n_{k,t}}{\sum_{k'} \sum_t n_{k',t}}, \tag{5}$$

and mentally mark anything below 1% as "this will get partial-pooled to death."

Quick pros/cons note

(a) *Fish-level Bernoulli*

Max info, super flexible for downstream covariates, but 100 k fish ⇒ chunky Stan gradients.

(b) *Binomial counts*

Much faster. Downside: we lose fish-specific stuff like fork length, but we don't have that anyway (right?).

(c) *Basin-year aggregation*

If we really just care about a map and a trend line, fine—but we'd be smearing any within-river quirks.

2. Model Formulation

I *super* don't want to over-extend my own thinking onto this, ultimately all of this is totally your call and you obviously have a MUCH clearer vision of what is going on in this system, but I trust you to just like totally go a different direction if you want to. I will be so incredibly not offended. However, for fun, here's how I would do it hehe. The general philosophy would be: break the log-odds of straying, $\eta_{j,t}$, into a few interpretable pieces—one for space, one for time, one for extra co-variates, and then some white noise because the ocean is messy.

Let $\eta_{j,t} = \text{logit}\big(p_{j,t}\big)$ and write

$$\eta_{j,t} \;=\; \alpha_{k(j)} \;+\; f(t) \;+\; \mathbf{x}_{j,t}^{\top}\boldsymbol{\beta} \;+\; \epsilon_{j,t}. \tag{6}$$

I think of this like Lego blocks we can swap in or out.

2.1 Basin (or river) intercepts

$$\begin{aligned} \alpha_k &\sim \mathcal{N}\big(\mu_\alpha,\ \sigma_\alpha^2\big), \\ \mu_\alpha &\sim \mathcal{N}(0, 2), \\ \sigma_\alpha &\sim \text{HalfNormal}(0, 1). \end{aligned} \tag{7}$$

Why? Because every basin almost certainly has its own baseline propensity to lose fish, but we don't trust the tiny basins to estimate that cleanly. The hierarchy ("partial pooling") lets big basins speak loudly and small basins borrow strength. Trade-off: if one basin is genuinely bonkers weird, shrinkage will tamp that down—so we'll watch the posterior for that.

2.2 How to handle the time trend $f(t)$

Three flavors on the menu, each with its own vibe:

B-splines

$$\begin{aligned} f(t) &= \sum_{h=1}^{H} b_h(t)\,\gamma_h, \\ \gamma_h &\sim \mathcal{N}(0, \sigma_f^2). \end{aligned} \tag{8}$$

Good when we more-or-less know how wiggly we want the curve and don't need principled uncertainty in the wiggles. Downside: picking knot locations feels a bit like art class.

Gaussian Process (GP)

$$\begin{aligned} f(t) &\sim \text{GP}\big(0, C_{\ell,\sigma_f^2}\big), \\ C_{\ell,\sigma^2}(t, t') &= \sigma_f^2 \exp\Big\{-\tfrac{(t-t')^2}{2\ell^2}\Big\}. \end{aligned} \tag{9}$$

Totally hands-off smoothness—Stan learns the length-scale ℓ. Super elegant, but the matrix algebra is T^3. We can cheapen it with a Vecchia trick if it bites us.

Random-walk ($\mathbf{RW_1}$)

$$\begin{aligned} f(t) &= f(t-1) + u_t, \\ u_t &\sim \mathcal{N}(0, \tau^2). \end{aligned} \tag{10}$$

Feels intuitive ("today's deviation is yesterday plus a nudge") and samples fast because it's basically a state-space model. But if the true curve has long gentle bends, RW1 sometimes looks like stair-steps.

No hard winner—we can fit all three and eyeball WAIC.

2.3 Optional spatial smoother (BYM2)

If K (basins) is, say, ≥ 20, we can add spatial borrowing via BYM2:

$$\begin{aligned} \alpha_k &= \sqrt{\phi}\, u_k + \sqrt{1-\phi}\, v_k, \\ \mathbf{u} &\sim \mathcal{N}\big(\mathbf{0}, \sigma_u^2 \mathbf{Q}^-\big),\ v_k \sim \mathcal{N}(0, \sigma_v^2), \\ \phi &\sim \text{Beta}(0.5, 0.5). \end{aligned} \tag{11}$$

If $K < 15$ it probably won't move the needle; if $K > 50$ it might even be *necessary* for stable tails. We'll check the adjacency matrix—if lots of basins are neighbours then spatial smoothing makes intuitive sense (fish don't respect map borders).

Residual noise

$$\begin{aligned} \epsilon_{j,t} &\sim \mathcal{N}(0, \sigma_\epsilon^2), \\ \sigma_\epsilon &\sim \text{ HalfNormal}(0, 1). \end{aligned} \tag{12}$$

This just soaks up the un-modelled mush so we don't pretend the fit is perfect.

3. Covariate Specification & Priors

I'm guessing we'll eventually throw in temperature, discharge, maybe hatchery yes/no, and latitude. All of those go into $\mathbf{x}_{j,t}$, which we z-score so Stan's scales are behaved:

$$
\begin{aligned}
x^{*}_{p_{j,t}} &= \frac{x_{p_{j,t}} - \bar{x}_p}{s_p}, \\
\beta_p &\sim \mathcal{N}(0, 1.5^2).
\end{aligned}
\tag{13}
$$

Nonlinear temperature response

I'm assuming that some people probably swear straying rises at both really warm and really cold temps—who knows. A thin-plate spline lets us find out without choosing a quadratic degree in advance:

$$
\begin{aligned}
g(T) &= \boldsymbol{B}^{\top}(T)\,\boldsymbol{\theta}, \\
\boldsymbol{\theta} &\sim \mathcal{N}(\mathbf{0}, \sigma_g^2 \mathbf{I}).
\end{aligned}
\tag{14}
$$

Interaction idea Latitude might modulate temperature stress, so just tack on

$$
\theta_{\text{int}} (T_{j,t} - \bar{T})(\lambda_j - \bar{\lambda}), \qquad \theta_{\text{int}} \sim \mathcal{N}(0, 1). \tag{15}
$$

Cheap to try, easy to drop.

Missing covariates If half the discharge records are NA we can either: (i) drop those rows (ouch), or (ii) slap on a Gaussian-copula imputer. I vote (ii); Stan will integrate over the missing bits automatically.

Pro/con

- *Splines*—super flexible; tune knot count carefully.
- *Polynomials*—easy to explain, can go crazy outside data range.
- *Standardizing*—keeps β priors sensible, but if we show results to managers we'll back-transform so axes make sense.

4. Model-Comparison Framework

We'll fit $\mathcal{M}_0$ to $\mathcal{M}_3$ (intercepts only → time trend → covariates → spatial) and rank them.

Info criteria in practice

$$\text{WAIC} = -2\sum_{i=1}^{n_{\text{tot}}}\Big[\log\Big(\tfrac{1}{S}\sum_{s=1}^{S} p\big(y_i \mid \boldsymbol{\theta}^{(s)}\big)\Big) \; - \; V_s\Big], \tag{16}$$

$$\text{LOOIC} = -2\sum_{i=1}^{n_{\text{tot}}} \log \hat{p}_{-i}(y_i), \tag{17}$$

where V_s is the posterior-sample variance across s draws.

In English: lower = better, and LOOIC is basically cross-validated log score. We'll report ΔLOOIC relative to the baseline so the direction is obvious.

Posterior-predictive loss

$$\text{PPL} = \sum_{j,t}\Big[\big(\bar{y}_{j,t} - \tilde{y}_{j,t}\big)^2 + \text{Var}_s\big(y^*_{j,t}\big)\Big], \tag{18}$$

with $\tilde{y}_{j,t}$ the observed proportion and y^* replicated data.

If a model gets the mean right but the variance wrong this metric will yell at us.

Trade-off plot A Pareto frontier of ΔLOOIC vs. effective parameters tells us if we're paying 20 extra degrees of freedom for only a one-unit bump in score (in which case, hard pass).

5. Implementation Workflow

Again, to **super** stress this – these are just my suggestions! There are always like multiple ways to do this, so feel free to treat this as motivation/inspiration.

Just re-stating the nuts & bolts so we both remember later:

1. Design matrices in R, save to JSON.
2. Pre-compute adjacency W, store sparse.
3. Non-centred parameterisations everywhere—cuts divergences.
4. Four chains, 10k iterations, aim for $\widehat{R} < 1.01$ and $< 0.1\%$ divergences. If not, bump `adapt_delta`.

Hyper-priors are gentle Half-Normals / Gammas as already written.

If the GP version chokes, we drop in a Vecchia approximation (20 nearest neighbours seems to work fine in other projects).

6. Diagnostics and Validation

Posterior-predictive checks—we'll spit out simulated y^* draws and make sure the distribution overlaps reality. Flat PPP histogram = happy.

Residuals—Dunn-Smyth trick so we can treat binomials as pseudo-normals and run an ACF. Any big spikes at lag 1? Probably means a missing temporal smoother.

Moran's I—if spatial pattern remains in the residuals, the BYM2 wasn't enough (or K was too small to learn).

SBC—always good hygiene; the rank histograms catch coding bugs we'd otherwise miss.

7. Interpretation and Inference

What will we actually tell people?

Covariate effects

Take a tidy one-unit bump in a standardised covariate, back-transform the log-odds to probability space, and quote $\Pr(\Delta_p > 0)$—super intuitive ("there's a 92% chance higher temperature boosts straying").

Time trend

Plot the median spline/GP curve with a grey 95% band; mark any year where zero isn't in the band (those are the hot spots we'll gossip about).

Maps

Posterior mean stray rate per basin, plus interval width coded as line thickness. Skinny lines = we're uncertain, fat lines = solid info.

Predictive oomph

ELPD on a leave-one-basin-year fold is our scoreboard. If $\Delta\text{ELPD} > 4$ we'll call that model a clear winner (roughly Bayes factor > 50).

And that's the tour—nothing carved in stone, but hopefully enough scaffolding that it's possible to jump into coding once everything is formalized from the analysis end.

8. Kick-off Step: Prior Predictive Simulation

Before fitting anything to the real fish, let's sanity-check that our priors don't already imply impossible worlds (e.g. a 90 stray rate or basins that never stray at all). The idea is dead simple: *sample from the full generative hierarchy with the likelihood turned on but the data left out, then look at the fake data.*

8.1 Generative recipe (mathematics)

For one simulation draw $s = 1, \ldots, S$:

1. Draw hyper-parameters $\mu_\alpha^{(s)},\ \sigma_\alpha^{(s)},\ \sigma_f^{(s)},\ \sigma_\epsilon^{(s)}$ from their priors (see Sect. 5.2).
2. Draw basin intercepts $\alpha_k^{(s)} \sim \mathcal{N}\big(\mu_\alpha^{(s)}, (\sigma_\alpha^{(s)})^2\big)$ for $k = 1, \ldots, K$.

3. Draw temporal smoother weights (pick whichever flavour you plan to start with):

$$\gamma_h^{(s)} \sim \mathcal{N}(0,1), \qquad f^{(s)}(t) = \sum_h b_h(t)\, \sigma_f^{(s)} \gamma_h^{(s)}.$$

4. Compute linear predictor

$$\eta_{j,t}^{(s)} = \alpha_{k(j)}^{(s)} + f^{(s)}(t) + \mathbf{x}_{j,t}^{\top} \boldsymbol{\beta}^{(s)} \; + \; \epsilon_{j,t}^{(s)}, \quad \epsilon_{j,t}^{(s)} \sim \mathcal{N}\big(0, (\sigma_\epsilon^{(s)})^2\big).$$

5. Transform to probabilities $p_{j,t}^{(s)} = \text{logit}^{-1}\big(\eta_{j,t}^{(s)}\big)$.

6. Finally simulate fake observations

$$y_i^{*(s)} \sim \text{Bernoulli}\big(p_{j(i),\,t(i)}^{(s)}\big) \quad \text{or} \quad m_{j,t}^{*(s)} \sim \text{Binomial}\big(n_{j,t},\; p_{j,t}^{(s)}\big),$$

depending on whether you're in fish-level or count mode.

If the histogram of $\bar{y}^{*(s)} = \dfrac{1}{n_{\text{tot}}} \displaystyle\sum_i y_i^{*(s)}$ puts a lot of mass outside, say, 0.00–0.30 (our ball-park empirical range), then the priors are too wild.

8.2 What do we eyeball?

- **Global mean**: does the simulated coast-wide stray rate land in a plausible band?
- **Basin distribution**: boxplot of $\{\hat{p}_k^{*(s)}\}$ across draws—any basins routinely at 0 or 1?
- **Time trend fan chart**: overlay 50 prior curves $f^{(s)}(t)$; do they explode off the plot?
- **Extreme-probability tail**: density of logits $\eta_{j,t}^{(s)}$—should mostly live between about ± 4 (i.e. $p \in [0.02, 0.98]$).

If any of those look bonkers, we tighten the corresponding prior scale (e.g. shrink σ_α or σ_f) *before* touching real data.

8.3 Tiny pseudocode sketch

Listing 1: Prior-predictive check driver

```
# __main.R   (early in the pipeline, before model fitting)
if (RUN_PRIOR_CHECK) {

  # 1. sample  S  prior-draw parameter sets ------------------------
  draws <- replicate(
            S,
            sample_prior_once(model = "M1_time_spline",
                              design = design_mats)
          )

  # 2. turn those into fake y* or m* ------------------------------
  fake_data <- lapply(draws, simulate_fake_observations)

  # 3. basic plots ------------------------------------------------
  plot_hist_global_means(fake_data)
  plot_box_by_basin(fake_data)
  overlay_time_trends(draws$f_time_curves)

  # 4. optional: save objects for later comparisons ---------------
  saveRDS(list(draws = draws, fake = fake_data),
          "outputs/model-objects/prior_sim.rds")
}
```

9.1 Repository Layout

Listing 2: Proposed project tree

```
chinook-stray-rates/
          __main.R                # orchestrates the entire analysis
          README.md
```

```
        .gitignore
        data/
              raw/                # original tag/recovery CSVs  (ignored)
              external/           # covariate downloads (presumably?)
              processed/          # analysis-ready files written by prep
        R/                        # modular function scripts
              prep_data.R         # ingest, clean, standardise
              build_design.R      # create Stan matrices / adjacencies
              fit_models.R        # wrapper around CmdStanR
              compare_models.R    # WAIC/LOO, rank, select
              diagnostics.R       # PPC, SBC, residual checks
              utils.R             # misc helpers (logging, plot themes)
        stan/
              models/             #  M 0 M3  .stan files
              data/               # JSON / R dump inputs
        outputs/
              model-objects/      # *.rds or *.qs fit objects
              figs/               # EDA & PPC plots
              tables/             # LOO/WAIC summaries (csv / tex)
        docs/
            methods_outline.tex # this document
```

9.2 High-Level Pseudocode (_main.R)

```
# ------------------------------------------------------------------
# 0. CONFIGURATION
# ------------------------------------------------------------------
define GLOBAL_OPTIONS           # paths, cores, model list, seeds
load_required_packages()        # renv/packrat handled elsewhere
source_all_R_scripts()          # imports functions from ./R/

# ------------------------------------------------------------------
```

```
# 1. DATA PREPARATION
# ------------------------------------------------------------------
raw_data      <- ingest_raw()                    # prep_data.R
covariates    <- fetch_external_covariates()
processed_dat <- clean_and_merge(raw_data,
                                 covariates)    # returns tibble
save_processed(processed_dat)                    # data/processed/

# ------------------------------------------------------------------
# 2. DESIGN MATRICES FOR STAN
# ------------------------------------------------------------------
design <- build_stan_inputs(processed_dat)      # build_design.R
write_json_dump(design, "stan/data/design.json")

# ------------------------------------------------------------------
# 3. MODEL FITTING LOOP
# ------------------------------------------------------------------
for each model_spec in MODEL_LIST (M0     M3):
    stan_file <- glue("stan/models/{model_spec}.stan")
    fit       <- fit_stan_model(stan_file, design)  # fit_models.R
    save_fit_object(fit, model_spec)                # outputs/model-
   objects/

# ------------------------------------------------------------------
# 4. MODEL COMPARISON
# ------------------------------------------------------------------
fit_objects <- load_all_fits()
ranking_tbl <- compare_models(fit_objects)          # compare_models.R
write_table(ranking_tbl, "outputs/tables/loo.csv")
best_model  <- extract_best(ranking_tbl)

# ------------------------------------------------------------------
# 5. DIAGNOSTICS & VISUALISATION
```

```
# ----------------------------------------------------------------------
generate_pp_checks(best_model, design)              # diagnostics.R
run_residual_acf(best_model)
compute_morans_I(best_model)
export_all_figures()                                # outputs/figs/

# ----------------------------------------------------------------------
# 6. SESSION INFO & REPRODUCIBILITY
# ----------------------------------------------------------------------
log_session_info()
```

9.3 Function-Level Pseudocode (examples)

```
function ingest_raw():
    read CSVs from data/raw/
    return combined dataframe

function clean_and_merge(raw, covars):
    filter missing keys
    derive 'stray' indicator
    standardise continuous vars (z-score)
    return cleaned tibble
```

```
function fit_stan_model(stan_file, design):
    compile if needed (CmdStanR cache)
    sample with specified chains/warmup/iter
    convert to draws_rvars
    return fit object
```

9.4 Stan Skeleton

Listing 3: Example Stan file

```
data {
  int<lower=1> N;               // observations
  int<lower=0,upper=1> y[N];    // stray indicator
  int<lower=1> K;               // n basins
  int<lower=1,upper=K> basin[N];
  int<lower=1> H;               // spline basis size
  matrix[N, H] B;               // pre-computed basis
}
parameters {
  vector[K] alpha_raw;
  real mu_alpha;
  real<lower=0> sigma_alpha;
  vector[H] gamma;
  real<lower=0> sigma_f;
}
transformed parameters {
  vector[K] alpha = mu_alpha + sigma_alpha * alpha_raw;
  vector[N] eta   = alpha[basin] + B * (sigma_f * gamma);
}
model {
  alpha_raw  ~ normal(0, 1);
  mu_alpha   ~ normal(0, 2);
  sigma_alpha~ normal(0, 1);
  gamma      ~ normal(0, 1);
  sigma_f    ~ normal(0, 1);
  y          ~ bernoulli_logit(eta);
}
generated quantities {
  vector[N] y_rep = bernoulli_logit_rng(eta);
}
```

9.5 Next Steps

1. Flesh out remaining *.stan files ($\mathcal{M}_2, \mathcal{M}_3$) by adding covariate matrix $\mathbf{X}$ and BYM2 blocks.

2. Wrap the entire pipeline in a lightweight `targets` or GNU `Make` workflow if fully automated dependency tracking is desired.

3. Integrate continuous-integration (CI) checks (e.g. GitHub Actions) running a short "toy" chain to ensure the repo stays reproducible.