A General Race-Model Likelihood: Structure, Notation, and Examples

Cindy

Abstract

Race models explain choices and response times by positing latent *accumulators* that finish at different times and compete to determine the observed outcome. This note presents a single, general likelihood recipe that covers common constructions (pooling, AND/OR logic, inhibition/exclusion, mixtures, terminals, censors, and deadlines) without enumerating many special cases. We express model structure with simple Boolean-like expressions over *pools* of accumulators and show how each expression defines an *event CDF* and its *first-hit density*. Small DSL examples illustrate how specifications map directly to the mathematics. The development assumes only that finishing-time densities and cumulative distribution functions are available.

1 Overview

Decisions are represented as contests among *events* that become true over time. Events are built from *pools* of accumulators using AND/OR/NOT-like operators and optional guards (inhibitors and protectors). An observed response occurs when its event becomes true *first*. This view yields a compact, general per-trial likelihood that applies regardless of how events are composed.

2 Objects and Notation

Each accumulator i has finishing-time CDF $F_i(t)$, PDF $f_i(t)$, and survival $S_i(t) = 1 - F_i(t)$. A *pool* P aggregates one or more accumulators under a completion rule. The most common rule is k-of-n: the pool completes when the k-th member finishes. For i.i.d. members with CDF F and PDF f,

$$F_{(k)}(t) = \sum_{m=k}^{n} \binom{n}{m} [F(t)]^m [1-F(t)]^{n-m}, \qquad f_{(k)}(t) = \binom{n}{k-1} f(t) [F(t)]^{k-1} [1-F(t)]^{n-k}. \tag{1}$$

For heterogeneous members $i = 1..n$, the first-hit density is

$$f_{(k)}(t) = \sum_{j=1}^{n} f_j(t) \sum_{\substack{S \subseteq \{1..n\} \setminus \{j\} \\ |S|=k-1}} \prod_{i \in S} F_i(t) \prod_{i \notin S,\ i \neq j} \big[1 - F_i(t)\big], \tag{2}$$

with $F_{(k)}(t)$ the corresponding CDF. We build *events* from pools using a small expression calculus. Every event E has a completion probability $C_E(t)$ (true by t) and a first-hit density $h_E(t)$ (becomes true *at* t). Throughout we assume independence between individual accumulators.

3 Expression Calculus (Events from Pools)

Let E_1, E_2 be events. The following rules define C_E and h_E recursively.

Event from a pool. If E is the event that pool P completes, then $C_E(t) = F_P(t)$ and $h_E(t) = f_P(t)$.

AND (all-of). $E = E_1 \wedge E_2$ becomes true when the last required part finishes:

$$C_\wedge(t) = C_{E_1}(t)\, C_{E_2}(t), \qquad h_\wedge(t) = h_{E_1}(t)\, C_{E_2}(t) + h_{E_2}(t)\, C_{E_1}(t). \tag{3}$$

OR (first-of). $E = E_1 \vee E_2$ becomes true when the first part finishes:

$$C_\vee(t) = 1 - \big[1 - C_{E_1}(t)\big]\big[1 - C_{E_2}(t)\big], \qquad h_\vee(t) = h_{E_1}(t)\, \big[1 - C_{E_2}(t)\big] + h_{E_2}(t)\, \big[1 - C_{E_1}(t)\big]. \tag{4}$$

NOT / exclusion. Absence constraints do not create new first-hits; instead, they multiply the likelihood by survival factors of the excluded elements. If outcome R must exclude a set $\mathcal{X}_R$ and is inhibited by $\mathcal{I}_R$ (blockers), define

$$M_R(t) = \prod_{X \in \mathcal{X}_R} S_X(t) \prod_{I \in \mathcal{I}_R} S_I(t). \tag{5}$$

Guards (`inhibit`/`guard` in the DSL) are encoded as such absence multipliers, optionally disabled by protectors that have finished earlier.

4 Unified Per-Trial Likelihood

Let the model offer responses $\{R\}$ (each an event expression), terminals $\{T\}$ that yield no choice, and censors $\{C\}$ that end observation. For a trial with observed outcome o at time t, the contribution is

$$\ell_o(t) = h_o(t)\left\{\prod_{S\neq o}[1-C_S(t)]\right\}\left\{\prod_{I\in\mathcal{I}_o} S_I(t)\right\}\left\{\prod_{X\in\mathcal{X}_o} S_X(t)\right\}\left\{\prod_{T} S_T(t)\right\}\left\{\prod_{C} S_C(t)\right\}. \tag{6}$$

If a terminal T or censor C is observed instead, use their densities

$$\tau_T(t) = f_T(t)\prod_R[1-C_R(t)]\prod_{T'\neq T} S_{T'}(t)\prod_C S_C(t), \qquad \gamma_C(t) = f_C(t)\prod_R[1-C_R(t)]\prod_T S_T(t)\prod_{C'\neq C} S_{C'}(t). \tag{7}$$

With a deadline D, densities are truncated to $t \leq D$ and the no-response probability collects remaining mass

$$\Pr(\text{no response by } D) = \prod_R\big[1 - C_R(D)\big]\prod_k\big[1 - F_{\text{censor},k}(D)\big]. \tag{8}$$

The full log-likelihood sums $\log L = \sum_j \log \ell_{o_j}(t_j)$ over trials (with τ/γ replacing ℓ when appropriate).

5 Mixtures, Mapping, and Shared Parameters

Mixtures average component-wise likelihoods: for components c with weights w_c,

$$\ell(t, o) = \sum_c w_c\,\ell_c(t, o). \tag{9}$$

Fixed relabelling or guessing maps the vector of response densities $\boldsymbol{p}(t)$ to $\boldsymbol{p}'(t) = \boldsymbol{M}\,\boldsymbol{p}(t)$ for a column-stochastic matrix $\boldsymbol{M}$. Shared parameters constrain subsets of $\boldsymbol{\theta}$ so two or more pools share distributional parameters; this changes F/f but not the structural forms in Equations (3)–(6).

6 DSL Examples (specification `examples/new_API.R`)

We show how the DSL maps to the mathematics above. In the snippets, `add_pool` declares a pool, `add_outcome` names an outcome and attaches an event expression, and `inhibit`/`all_of`/`first_of` correspond to NOT/AND/OR semantics.

Simple two-response race. Two single-member pools compete.

```
example_1_simple <- race_spec() |>
  add_accumulator("go1", "lognormal", meanlog = log(0.30), sdlog = 0.18) |>
  add_accumulator("go2", "lognormal", meanlog = log(0.32), sdlog = 0.18) |>
  add_pool("R1", "go1") |>
  add_pool("R2", "go2") |>
  add_outcome("R1", "R1") |>
  add_outcome("R2", "R2") |>
  build_model()
```

Math at a glance: $h_{R1}(t) = f_{go1}(t)$, $C_{R2}(t) = F_{go2}(t)$; use Equation (6) for $\ell_{R1}(t)$ and swap roles for $R2$.

Go/Stop with inhibition and gating. One response is inhibited by a stop signal; another requires the stop as a gate.

```
example_2_stop_mixture <- race_spec() |>
  add_pool("GO1", "go1") |>
  add_pool("STOP", "stop") |>
  add_pool("GO2", "go2") |>
  add_outcome("R1", inhibit("GO1", by = "STOP")) |>
  add_outcome("R2", all_of("GO2", "STOP")) |>
  build_model()
```

Here $h_{R1}(t) = h_{GO1}(t)$ with absence multiplier $M_{R1}(t) = S_{STOP}(t)$; and $h_{R2}(t) = h_{\wedge}(t)$ from Equation (3) with $E_1 = GO2$, $E_2 = STOP$.

k-of-n pool. Outcome completes when two of three members finish.

```
example_9_advanced_k <- race_spec() |>
  add_pool("A", c("a1","a2","a3"), k = 2L) |>
  add_outcome("A", "A") |>
  build_model()
```

Use Equation (1) (or (2)) to obtain F_A and f_A, then plug into Equation (6).

Dual path via shared gate. Two outcomes share a last pool (a gate), requiring tie resolution if the gate finishes at t.

```
example_6_dual_path <- race_spec() |>
  add_pool("TaskA", "acc_taskA") |>
```

```
  add_pool("TaskB", "acc_taskB") |>
  add_pool("GateC", "acc_gateC") |>
  add_outcome("Outcome_via_A", all_of("TaskA", "GateC")) |>
  add_outcome("Outcome_via_B", all_of("TaskB", "GateC")) |>
  build_model()
```

When a group of outcomes shares the same last-required pool, split the at-t mass from that pool across outcomes using time-varying weights based on which non-shared cores would have finished first; the engine handles this automatically.

Censoring and deadline. Add a censor stream and a hard deadline.

```
example_11_censor_deadline <- race_spec() |>
  add_pool("L", "go_left") |>
  add_pool("R", "go_right") |>
  add_pool("CENSOR", "censor_watch") |>
  add_outcome("Left", "L") |>
  add_outcome("Right", "R") |>
  add_outcome("NR_CENSOR", "CENSOR", options = list(class = "censor")) |>
  set_metadata(deadline = 0.55) |>
  build_model()
```

Use Equation (7) for $\gamma_{\mathrm{CENSOR}}(t)$ and Equation (8) for the no-response mass beyond D.

Mixture across components. Average component-wise likelihoods.

```
example_7_mixture <- race_spec() |>
  add_pool("TARGET", c("target_fast", "target_slow")) |>
  add_pool("COMP", "competitor") |>
  add_outcome("R1", "TARGET") |>
  add_outcome("R2", "COMP") |>
  set_metadata(mixture = list(components = list(
    component("fast", weight = 0.2), component("slow", weight = 0.8)
  ))) |>
  build_model()
```

Evaluate Equation (6) per component and average via Equation (9).

7 Implementation Recipe

For each trial: (1) parse each outcome's expression into a tree of events; (2) compute $C_E(t)$ and $h_E(t)$ recursively using Equations (1)–(4); (3) evaluate the per-trial contribution with Equation (6); (4) include terminals/censors via Equation (7) or deadlines via Equation (8) if present; (5) apply mixtures and any label mappings; (6) sum logs. Absence constraints multiply in as survival factors and do not change first-hit forms.

Relationships and special cases. (i) Gates are just ANDs over required pools; (ii) Excluders and inhibitors are absence multipliers; (iii) Terminals/censors are symmetric to responses in the competition, differing only in labelling; (iv) Shared last-pool ties redistribute the at-t mass from the shared pool across tied outcomes; without shared last pools, no tie handling is needed.

References