tidycreel: A Unified R Framework for Modern Creel Surveys

SCHOOL OF NATURAL RESOURCES

Christopher Chizinski[1], Sarah Ulrichsen[1], Logan Walters[1], Austin Stafford[1], Israt Zarin[1], Keith Hurley[2]

[1] School of Natural Resources, University of Nebraska-Lincoln [2] Nebraska Game and Parks Commission

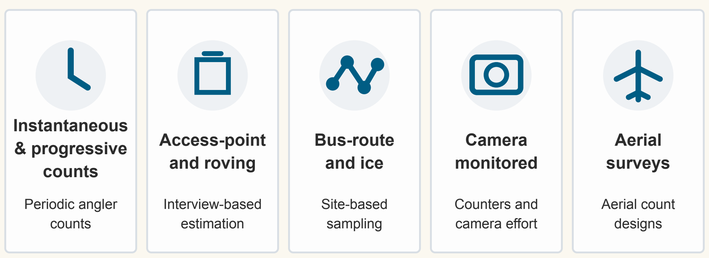
What is tidycreel?

Running a creel survey is straightforward, but analyzing one is not. The design you ran decides the estimator you are allowed to use, and getting that pairing wrong is easy to do and hard to catch.

tidycreel is an R package: one workflow from sampling plan to final estimate.

- **The right estimator, chosen for you** — from the design, not from memory
- **Uncertainty on every number** — variance and CI, carried to the report
- **The same answer next year** — a script, not a spreadsheet

Any design you actually run



You pick the design, and tidycreel picks the estimator.

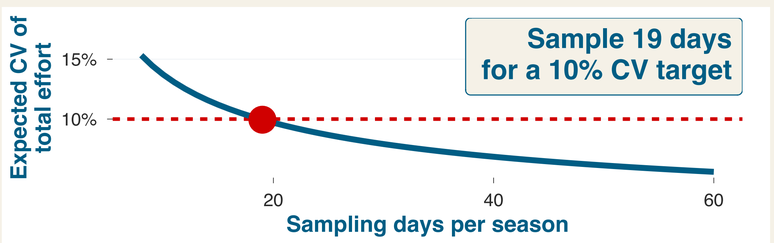
Flexible data input

The companion package **`tidycreel.connect`** imports survey data directly into the tidycreel workflow.

- Databases or CSV files (`creel_connect()`)
- REST APIs (`creel_connect_api()`)
- Returns validated tables ready for `add_*()` functions

1 · Plan the season

How many days, and which ones? Both are settled before anyone goes to the water.



Set the precision you need, read off the effort it costs, and take that number to whoever controls the budget. `cv_from_n()` and `optimal_n()` turn a target into a sample size.

Audit the strata before committing

- Check weights, inclusion probabilities, coverage (`audit_strata()`)
- Test a redesign before you commit (`simulate_strata_collapse()`)
- Move effort where it buys precision (`reallocate_strata()`)
- Confirm the plan hits your CV target (`validate_design()`)

Then build a defensible calendar

- Allocate sampling days across periods (`generate_schedule()`)
- Assign sites and crews to a route (`generate_bus_schedule()`)
- Check the calendar before anyone drives (`validate_creel_schedule()`)

2 · Build the design

One object holds the whole season, and every estimator takes it first. Change `survey_type=` and the same code runs a bus-route survey instead of a roving one.

Why a design object?

Most creel analyses hand calendar, count, interview and catch tables from one function to the next. That is how seasons get mixed and strata mismatched, with nothing to warn you. One validated object closes that door.

```
design <- creel_design(calendar, date = date,
                       strata = day_type) |>
  add_counts(counts)          |>
  add_interviews(interviews) |>
  add_catch(catch)

#> Calendar:   122 days (2025-05-01 to 2025-08-30)
#> Counts:     30 observations   Strata: day_type
#> Interviews: 476 (343 complete, 133 incomplete)
```

Where these numbers come from

Every figure and printout on this poster is one simulated 122-day reservoir season. `simulate_creel_data()` is fitted to Nebraska Game and Parks Commission creel data and ships with the package, so you can reproduce all of this, and find where it breaks, before pointing tidycreel at your own survey.

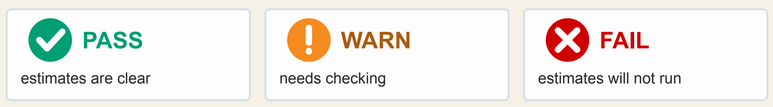
3 · Validate

Catch problems early. tidycreel refuses to estimate from data it cannot defend, and it says why:

- Calendar days with interviews but no counts
- Interviews dated outside the season
- Strata too thin to estimate a variance
- Incomplete trips that are not equivalent to complete ones

```
validation_report(counts, interviews)
#> Overall: PASS
```

Every check returns one of three verdicts:



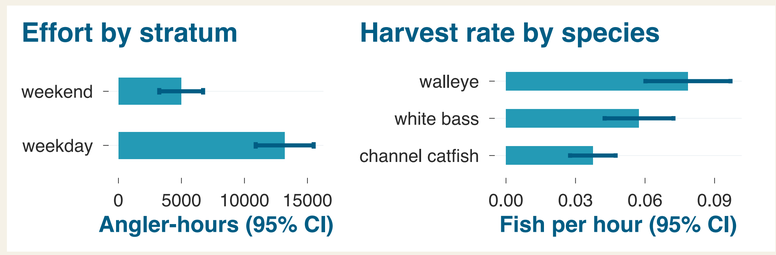
For example, should you pool the incomplete trips? In this season, no.

Pooling mid-trip interviews with completed ones is only defensible if the two give equivalent catch rates. `validate_incomplete_trips()` tests that.

```
validate_incomplete_trips(design, catch =
catch_total, effort = hours_fished)
#> Threshold: ±20% of complete trip estimate
#> ! Equivalence FAILED: use complete trips only
```

The answer is no, and those 133 trips are 28% of the sample.

4 · Estimate with uncertainty



Every number comes with its uncertainty, with no separate variance step and no post-hoc bootstrap you have to remember to run. Error bars above are design-based.

Taylor linearization, bootstrap, and jackknife share one interface, and non-response adjustment rides along.

Beyond effort and catch

- Estimate angler abundance from marks (`estimate_angler_n()`)
- Scale harvest to that population (`estimate_mr_harvest()`)
- Work exploitation from tag returns (`estimate_exploitation_rate()`)
- Weighted size and age structure (`est_length_distribution()`)

Same design-based uncertainty throughout.

One season, every total. Effort, catch and harvest over one design at `target = "period_total"` — 30 sampled days extrapolated to all 122.

```
bind_rows(lapply(totals, tidy), .id = "metric")
#> # A tibble: 3 × 5
#>   metric  estimate    se ci_lower ci_upper
#> 1 effort    18245. 1413.   15350.   21139.
#> 2 catch      8440. 1110.    6256.   10624.
#> 3 harvest    3748.  457.    2850.    4646.
```

5 · Report and communicate

One design in, shareable results out. Every summary, figure, and exported table carries the survey design and its uncertainty, so what you hand over is what someone else can check.

```
summary(effort_est)
#> Creel Survey Summary (Total | Taylor linearization |
95%)
#>  day_type Estimate     SE CI Lower CI Upper  N
#>   weekday    13280 1126.0    10970    15590 20
#>   weekend     4964  853.5     3215     6712 10
```

Re-run it next season and the numbers are comparable, because the design lives in the object rather than in someone's memory.

Continue Learning

Package docs, open source. Install it, look up any function, and follow the tutorials that build a survey from raw counts to a reported estimate.

chrischizinski.com/tidycreel

Companion book, free online. *Modern Creel Survey Analysis in R* gives the design-based reasoning behind every default, worked on real reservoirs.

modern-creel-surveys.netlify.app

An open-source framework for reproducible, design-based creel survey analysis.