OptSurvCutR: Validated Cut-point Selection for Survival Analysis

Payton TO Yau

School of Science and Technology, Nottingham Trent University, Nottingham, NG11 8QX

ABSTRACT

The stratification of subjects based on continuous predictors is a common yet challenging task in time-to-event analysis (or survival), particularly when relationships are non-linear and require multiple thresholds. Arbitrary cut-point selection can lead to overfitting and spurious conclusions, highlighting the need for a statistically rigorous and reproducible framework.

We present *OptSurvCutR* (Optimal Survival Cut-Points in R programming), an R package designed to provide a complete, end-to-end workflow for robust and interpretable cut-point analysis. The package's methodology is structured into a logical three-step process. First, the *find_cutpoint_number()* function provides a data-driven approach to determine the optimal number of cut-points by comparing models using information criteria (e.g., AIC, AICc or BIC). Second, the *find_cutpoint()* function identifies the precise locations of these thresholds by optimising survival metrics, such as the log-rank statistic or hazard ratio, using either an exhaustive systematic search or an efficient genetic algorithm. Finally, the *validate_cutpoint()* function assesses the stability of the identified thresholds by generating 95% confidence intervals through bootstrap resampling.

We demonstrate the package's full workflow using two case studies: a plant science example modelling the non-monotonic effect of temperature on rapeseed germination and a clinical bioinformatics analysis stratifying colorectal cancer patients by a microbial biomarker. These examples show how *OptSurvCutR* can uncover complex survival patterns that would be missed by traditional dichotomisation. *OptSurvCutR* provides a transparent and extensible framework for validated cut-point selection, making it broadly applicable for researchers working with time-to-event data. The package, including source code and documentation, is freely available from GitHub at https://github.com/paytonyau/OptSurvCutR.

Keywords: cut-point selection, multiple cut-points, survival analysis, time-to-event, biomarker, risk stratification, genetic algorithm, bootstrapping, bioinformatics, R package

BACKGROUND

The identification of meaningful thresholds within continuous data is a ubiquitous task in data-driven research, from clinical diagnostics to environmental and food science. The conversion of a continuous variable into a categorical one by defining an optimal cut-point is a powerful method for risk stratification and decision-making (Mayeux, 2004). In high-throughput omics data, such as gene expression, for example, it often relies on this cut-off approach to stratify patients into low- and high-risk groups for survival outcomes. However, identifying a truly optimal and robust cut-point is non-trivial. The process is susceptible to statistical artefacts, such as the multiple comparisons problem, which can lead to the selection of a statistically significant but clinically irrelevant threshold (Altman et al., 1994). Moreover, biological and environmental systems rarely adhere to simple linear relationships. A single cut-point often does not capture non-monotonic, U-shaped relationships, such as when both very low and very high levels of a biomarker (i.e., temperature and pH) are associated with increased risk (Royston et al., 2006). In these cases, forcing such complex relationships into a simple high/low dichotomy can lead to significant patient misclassification and obscure the true underlying biological patterns. Identifying multiple thresholds to define low-, intermediate-, and high-risk groups is therefore essential for accurately modelling such complex systems.

While several R packages exist to address this challenge, they are often specialised for specific scenarios. Packages like *survminer* provide excellent tools for finding a single optimal cut-point in survival contexts (Alboukadel Kassambara et al., 2024). Foundational methods for outcome-based biomarker assessment were established by early graphical bioinformatics tools like *X-tile*, which uses the log-rank test to find multiple cut-points (Camp et al., 2004), and later by web applications like Cutoff Finder (Budczies et al., 2012). More recent statistical approaches have also been described for determining both the optimal number and location of cut-points, for instance, by using the Akaike Information Criterion (AIC) to select the best model (Chang et al., 2017). Despite the power of these individual tools, a significant methodological gap remains for a single, well-integrated R package that provides a complete and flexible workflow—from determining the appropriate number of cuts to finding their locations and validating the result—within a reproducible software environment.

To address these limitations, we have developed *OptSurvCutR*, a versatile R package designed specifically for robust cut-point analysis in a time-to-event context. *OptSurvCutR* consolidates the entire analytical process into a single toolkit, offering (1) a data-driven function to determine the optimal number of cut-points, (2) flexible algorithms for identifying both single and multiple thresholds, (3) an integrated validation step to ensure the robustness of the findings, and (4) support for parallel processing to accelerate computationally intensive tasks.

IMPLEMENTATION

Package Architecture and Workflow

The *OptSurvCutR* package is implemented in the R statistical programming language and is designed to provide a cohesive and logical workflow for cut-point analysis in a survival context. The workflow is structured to guide a researcher from initial data exploration to final model validation and is centred around three core functions, which are summarised in **Table 1**.

Table 1. The *OptSurvCutR* analysis workflow. This table outlines the three core functions and their recommended use in a typical analysis. AIC, Akaike Information Criterion; AICc, Corrected Akaike Information Criterion; BIC, Bayesian Information Criterion.

Step & Function	Purpose	Recommended Use Case
1 *find_cutpoint_number()*	To determine the statistically optimal number of cut-points (e.g., is it better to have one, two, or zero cuts?).	**method**: "genetic" is recommended for flexibility. "systematic" is faster but limited to max_cuts = 2. **criterion**: "BIC" (default), "AIC", or "AICc".
2 *find_cutpoint()*	To find the precise location of the optimal cut-point(s).	**num_cuts**: Set this based on the result from Step 1. **method**: "genetic" is recommended for 2+ cuts. "systematic" is generally faster for 1 cut. **criterion**: "logrank" (default) is a robust choice for assessing significance. "hazard_ratio" can be used to maximise effect size (for 1 cut-point only).
3 *validate_cutpoint()*	To assess the stability and precision of the discovered cut-point(s) by generating 95% confidence intervals via bootstrapping.	Apply the object returned from Step 2. **num_replicates**: 500 or higher is recommended for reliable confidence intervals.

CORE FUNCTIONS AND ALGORITHMS

The *OptSurvCutR* package is implemented in the R statistical programming language and is designed to provide a cohesive and logical workflow for cut-point analysis in a time-to-event context. The workflow is structured to guide a researcher from initial data exploration to final model validation and is centred around three core functions: *find_cutpoint_number()*, *find_cutpoint()*, and *validate_cutpoint()*. These functions employ one of two distinct search algorithms for optimisation: a "systematic" search (Altman et al., 1994), and a "genetic" algorithm (Mebane Jr & Sekhon, 2011). The systematic search exhaustively evaluates all possible thresholds, guaranteeing a globally optimal solution but becoming computationally infeasible for multiple cut-points. In contrast, the genetic

algorithm uses a heuristic evolutionary approach to efficiently search the vast parameter space, making it a practical and powerful tool for identifying multiple thresholds in larger datasets.

Step 1: Determining the Number of Cut-points with *find_cutpoint_number()*

This function serves as the initial, exploratory step to help researchers make a data-driven decision on the most plausible number of cut-points. It leverages the search algorithms to compare models with different numbers of thresholds (e.g., 0 vs. 1 vs. 2 cuts) and provides statistical evidence using information criteria. The function supports the Akaike Information Criterion (AIC, $AIC = -2log(L) + 2k$) (Akaike, 1974), Bayesian Information Criterion (BIC, $BIC = -2cdotlog(L) + kcdotlog(n)$) (Gideon Schwarz, 1978) or Corrected Akaike Information Criterion (AICc, $AICc = AIC + frac2k(k+1)n-k-1$) (Hurvich & Tsai, 1989). These criteria balance model fit (measured by the log-likelihood, log(L)) with model complexity (the number of parameters, k) and sample size (n). The model with the lowest information criterion score is considered the most plausible.

Step 2: Identifying Cut-point Locations with *find_cutpoint()*

This is the main workhorse function for identifying the precise location of the optimal cut-point(s). It uses one of two search algorithms—either a systematic or genetic search—to find the cut-point(s) that optimise a user-selected survival metric. The available criteria are based on established statistical methods for survival analysis:

- **"logrank"**: This criterion finds the cut-point that maximises the chi-squared statistic from the standard non-parametric log-rank test (Mantel, 1966). The test compares the observed (O) and expected (E) number of events across groups at each event time.
- **"hazard_ratio"**: This criterion finds the cut-point that maximises the Hazard Ratio (HR) derived from a Cox Proportional-Hazards model (Cox, 1972). The HR quantifies the magnitude of the difference in risk between the groups created by the cut-point, with a higher value indicating a more clinically significant separation.
- **"p_value"**: This criterion seeks the cut-point that minimises the p-value from the log-rank test component of a Cox Proportional-Hazards model (Cox, 1972). This method directly optimises for the highest level of statistical significance in the difference between the survival curves of the groups.

Step 3: Assessing Stability with *validate_cutpoint()*

After a cut-point has been identified, this function assesses its stability and robustness. It performs a non-parametric bootstrap resampling procedure (Efron, 1979). This process involves creating a large

number of new datasets by sampling subjects with replacement from the original data. To ensure a fair validation, the entire *find_cutpoint()* algorithm, using the original settings (e.g., method and criterion), is then re-applied to each of these bootstrap samples. The resulting collection of newly discovered optimal cut-points is used to build an empirical distribution, which provides a direct assessment of how sensitive the optimal cut-point is to sampling variability. This distribution is then used to calculate 95% confidence intervals, which indicate the reliability of the chosen threshold.

The package also includes a suite of S3 methods for *print()*, *summary()*, and *plot()* to ensure that the results from each step are user-friendly and easy to interpret visually.

Software Implementation and Availability

The *OptSurvCutR* package is implemented in R (R Core Team, 2025). Its core statistical functionality relies on the *survival* package for time-to-event modelling (Therneau, 2024) and the *rgenoud* package for the genetic algorithm optimisation (Mebane Jr & Sekhon, 2011). For computational efficiency, parallel processing is implemented via the *foreach* and *doParallel* packages (Corporation & Weston, 2022). Data manipulation and visualisation are handled by packages from the *tidyverse* ecosystem (Wickham et al., 2019), primarily *dplyr* (Wickham et al., 2014) and *ggplot2* (Wickham, 2016), with survival-specific plots enhanced by *survminer* (Kassambara et al., 2016). User-facing messages and progress bars are generated by the *cli* package (Csárdi, 2017). Reproducibility for the stochastic genetic algorithm is ensured through base R's random seed generation. The package, including all source code and documentation, is freely available from GitHub at https://github.com/paytonyau/OptSurvCutR.

CASE STUDIES

Case Study 1: Identifying Temperature Thresholds for Rapeseed Germination

To demonstrate the full exploratory workflow of *OptSurvCutR* in a time-to-event context, we analysed a simulated dataset inspired by research on the germination of rapeseed (*Brassica napus L.*), where germination time is expected to have a non-monotonic relationship with temperature (Haj Sghaier et al., 2022). To create a realistic dataset for analysis, we simulated data in two parts to mimic the findings of the original study. First, we generated data based on the seven constant temperature points reported in the manuscript. Then, to create a more continuous predictor variable for finding optimal thresholds, we interpolated the germination parameters for six intermediate temperature points and generated a second part of the dataset. The combination of these simulated datasets is used to demonstrate the capabilities of the package (**Supplementary information 1**).

First, an exploratory analysis was performed using *find_cutpoint_number()* to determine the most plausible number of cut-points. A comparison of models with 0 to 4 cuts showed that the BIC was minimised for a model with 3 cut-points, providing statistical support for stratifying the data into four distinct temperature zones (**Figure 1A**).

```
R> library(OptSurvCutR)
R> data(”germination”)
R> number_result_bic <- find_cutpoint_number(
+  data = analysis_data,
+  predictor = “temperature”,
+  outcome_time = “time”,
+  outcome_event = “germinated”,
+  method = “genetic”,
+  criterion = “BIC”,
+  use_parallel = TRUE,
+  max_cuts = 5,
+  nmin = 0.1,
+  maxiter = 500,
+  seed = 42
+  )oi8934iuhjmnn
```

Next, *find_cutpoint()* was run with method = “genetic”, criterion = “logrank”, and num_cuts = 3 to identify the specific temperature thresholds. The algorithm identified three optimal cut-points at 8.7°C, 12.8°C, and 31.8°C. These thresholds stratified the seeds into four groups: “Cool” (≤12.8°C), “Sub-Optimal” (8.7°C-12.8°C), “Optimal” (12.8°C-31.8°C) and “Warm” (>31.8°C). The Kaplan-Meier analysis of the four groups revealed that the “Optimal” group had the fastest germination rate, as indicated by an early and steep drop in its curve. In contrast, the “Cool”, “Sub-Optimal”, and “Warm” groups showed similar patterns to each other, with their curves dropping several days later, confirming that germination was inhibited under these conditions. The p-value (< 0.0001; **Figure 1B**) indicates the differences between these groups are highly statistically significant.

```
R> multi_cut_result <- find_cutpoint(
+  data = analysis_data,
+  predictor = “temperature”,
+  outcome_time = “time”,
+  outcome_event = “germinated”,
+  method = “genetic”,
+  criterion = “logrank”,
+  use_parallel = TRUE,
+  num_cuts = 3,
+  nmin = 0.1,
+  maxiter = 500,
+  seed = 123
+  )
```

Finally, the stability of these three thresholds was assessed using *validate_cutpoint()* with 500 bootstrap replicates. The analysis yielded narrow 95% confidence intervals for all three cut-points, indicating that the identified thresholds were stable and not artefacts of the specific data sample (**Figure 1C**).

```
R> validation_result <- validate_cutpoint(
+  cutpoint_result = multi_cut_result,
+  num_replicates = 500,
+  use_parallel = TRUE,
+  seed = 456,
+  maxiter = 100
+  )
```

Analysis using a Cox proportional-hazards model showed that seeds in the “Cool”, “Sub-Optimal”, and “Warm” temperature groups had a significantly slower germination time compared to the “Optimal” reference group. The result was highly significant ($p < 0.0001$), as visualised in the corresponding forest plot (**Figure 1D**).

```
R> analysis_data$Temp_Group <- relevel(analysis_data$Temp_Group, ref = “Optimal”)
R> cox_model <- coxph(Surv(time, germinated) ~ Temp_Group, data = analysis_data)
R> ggforest(cox_model, data = analysis_data, main = “Hazard Ratios Relative to
+  Optimal Temperature”)
```

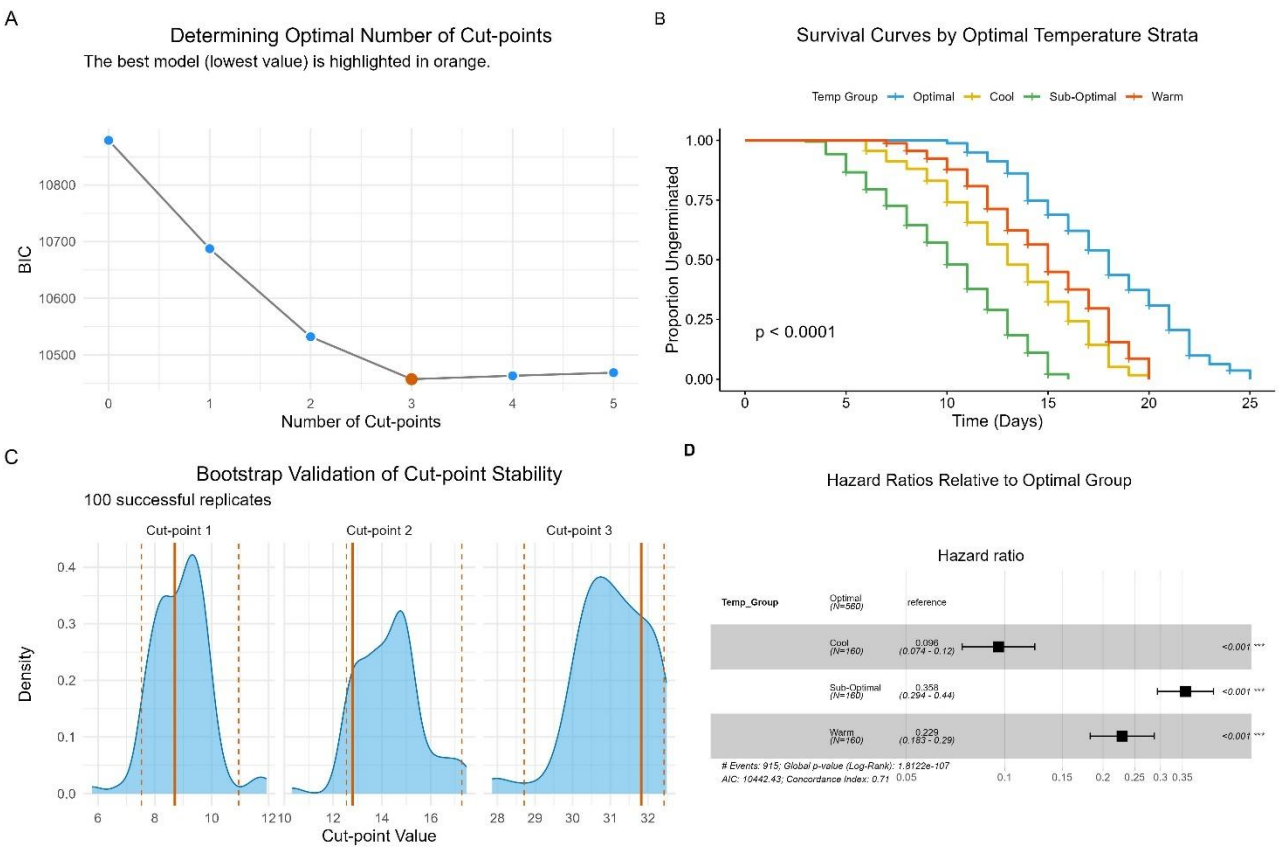

Figure 1. A comprehensive workflow demonstrating the statistical identification, validation, and

interpretation of optimal temperature thresholds for rapeseed germination time. (A) Optimal Model Selection: Bayesian Information Criterion (BIC) scores plotted against the number of proposed cut-points. The model with 3 cut-points was selected as optimal due to the lowest BIC score, justifying the stratification of the data into four distinct groups. **(B) Survival Analysis:** Kaplan-Meier curves illustrating the time-to-germination profiles for the four statistically derived temperature groups: “Cool”, “Sub-Optimal”, “Optimal”, and “Warm”. The “Optimal” group exhibits the fastest germination and is set as the reference for comparison in the hazard ratio analysis. Shaded areas represent 95% confidence intervals, and the displayed p-value is from a log-rank test assessing the overall difference between the groups. **(C) Cut-point Stability Assessment:** Density plots showing the empirical distribution of the three optimal cut-points derived from 500 bootstrap replicates. The narrow peaks indicate high stability of the originally identified thresholds (solid vertical lines). **(D) Effect Size Quantification:** A forest plot displaying the Hazard Ratios (HR) and 95% confidence intervals from a Cox proportional-hazards model. The analysis quantifies the germination rate of the non-optimal groups relative to the “Optimal” reference group. Error bars represent 95% confidence intervals.

Case Study 2: Stratifying Colorectal Cancer Patients by a Microbial Biomarker

To demonstrate the application of *OptSurvCutR* in a clinical bioinformatics context, we utilised a TCGA gut virome dataset in colorectal cancer (CRC) (Smyth et al., 2024). We aimed to identify the relative abundance of the genus Enterovirus in patients across several distinct survival groups for a 5-year overall survival analysis, with patient follow-up time censored at 60 months.

First, an exploratory study using *find_cutpoint_number()* was performed to identify the optimal number of prognostic groups. A comparison of models with 0 to 4 cuts indicated that a model with two cut-points was optimal, as determined by the BIC (BIC = 1251.69) with using genetic and a minimum of 20% of the total number for the group to avoid low numbers in any one of the groups (**Figure 2A**).

Next, *find_cutpoint()* was executed with num_cuts = 2 to identify the precise abundance thresholds that best separated these three groups, optimising for the log-rank statistic. The two optimal cut-points were used to categorise patients into “Low”, “Medium”, and “High” Enterovirus genus abundance groups. A Kaplan-Meier analysis revealed a highly significant difference in 5-year overall survival among these three groups (log-rank, p = 0.0018; **Figure 2B**).

The stability of these thresholds was then assessed using *validate_cutpoint()* with 500 bootstrap replicates. The analysis yielded narrow 95% confidence intervals for both cut points (Cut 1: 1.506 – 1.893; Cut 2: 1.749 – 2.357) (**Figure 2C**). The narrow confidence intervals confirmed the stability of both thresholds, although their slight overlap suggests the boundary between the “Low”, “Medium”, and “High” groups may represent a continuous gradient of risk rather than an abrupt change.

A subsequent Cox proportional-hazards model was fitted to quantify the risk associated with each group, using the “Medium” abundance group as the reference. The “High” abundance group was associated with a significantly increased hazard of mortality (Hazard Ratio = 2.9; 95% CI, 1.5 - 5.5),

while the “Low” group showed a risk profile similar to the reference group (Hazard Ratio = 2.6; 95% CI, 1.4 - 4.8)(**Figure 2D**).

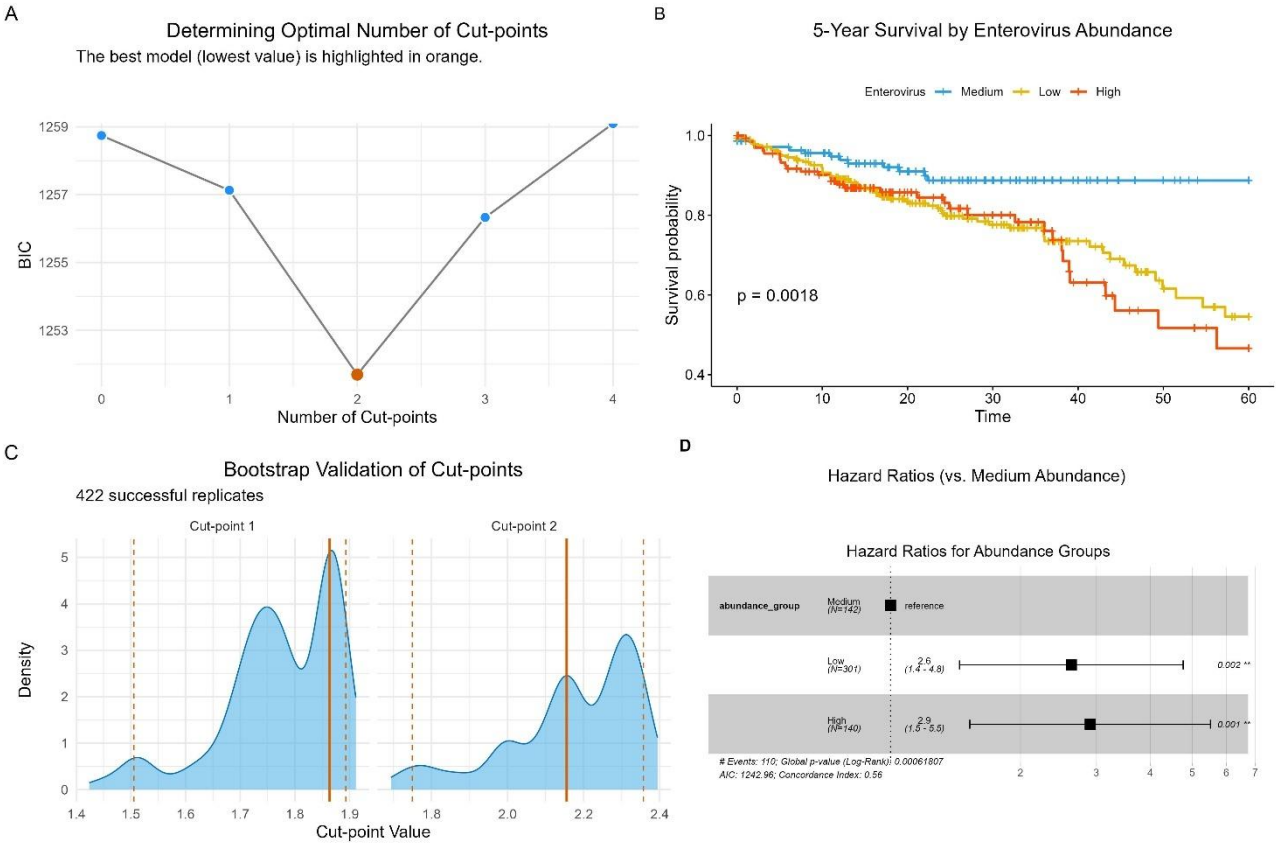

Figure 2. Complete workflow for the prognostic analysis of *Enterovirus* abundance in colorectal cancer. (A) Optimal Model Selection: Bayesian Information Criterion (BIC) analysis to determine the optimal number of prognostic groups based on *Enterovirus* abundance. The model with 2 cut-points (creating three groups) was selected due to the lowest BIC score. **(B) Survival Analysis:** Kaplan-Meier curves for 5-year overall survival for the three patient groups (”Low”, “Medium”, “High”). The analysis reveals a significant difference in survival profiles (log-rank $p < 0.001$), with the “Medium” abundance group showing the most favourable outcome. **(C) Cut-point Stability:** Density plots from a bootstrap analysis (500 replicates) assessing the stability of the two abundance thresholds. The narrow distributions confirm the robustness of the identified cut-points (solid vertical lines). **(D) Risk Quantification:** Forest plot displaying the Hazard Ratios (HR) for mortality from a Cox proportional-hazards model. The analysis quantifies the increased risk for the “Low” and “High” abundance groups relative to the “Medium” reference group. Error bars represent 95% confidence intervals.

These case studies demonstrated how *OptSurvCutR* can be used to uncover a complex, non-linear prognostic relationship, identifying a specific high-risk patient subgroup that would be obscured by traditional dichotomisation methods.

Discussion

The identification of optimal cut-points is a common yet challenging task in data analysis, with significant implications for biomarker discovery and patient stratification. In this paper, we have introduced *OptSurvCutR*, a novel R package designed to provide a comprehensive and statistically robust workflow for this purpose in a time-to-event context. Through a case study using a simulated dataset inspired by real-world ecological research, we have demonstrated its key capabilities for identifying and validating multiple cut-points.

A primary advantage of *OptSurvCutR* is its integrated workflow, which guides the user from initial model selection to final validation. The *find_cutpoint_number()* function addresses a critical, often-overlooked step by providing a data-driven method to justify the number of cut-points, protecting against arbitrary data dichotomisation. Furthermore, the package's most unique contribution is the implementation of a genetic algorithm (method = "genetic") to identify multiple cut-points simultaneously. As demonstrated with the rapeseed germination data, this feature is essential for uncovering complex, non-linear relationships between a biomarker and an outcome, such as the U-shaped effect of temperature on germination time, which standard dichotomisation methods would miss.

Comparison with Existing Software

The development of *OptSurvCutR* builds upon a long history of tools designed for cut-point optimisation. Foundational graphical tools like *X-tile* (Camp et al., 2004), developed nearly 20 years ago, established the importance of log-rank based optimisation by using an exhaustive systematic search to find up to two optimal cut-points. While pioneering, this approach was computationally limited to stratifying data into a maximum of three groups. Later, web applications like Cutoff Finder provided an accessible platform for single cut-point analysis, but are no longer updated and available online (Budczies et al., 2012). While these tools were pioneering, a need exists for a modern, extensible, and reproducible solution within the R programming environment.

Within the current R ecosystem, *OptSurvCutR* complements existing packages by providing a unique, integrated workflow (**Table 2**). Packages like *survminer* and *maxstat* are powerful tools for finding a single optimal cut-point, typically using a systematic search. *OptSurvCutR* not only includes this exhaustive search capability for ensuring global optimality in single cut-point scenarios but also crucially extends beyond these packages by providing a powerful genetic algorithm to efficiently discover multiple thresholds for more complex problems. Furthermore, few packages offer an integrated function to provide statistical justification for the number of cut-points to be used. By combining these features—justification, discovery of multiple cut-points, and bootstrap-based

validation—into a single, cohesive package, OptCutSurvR provides a more complete analytical workflow.

Table 2. Feature Comparison of *OptSurvCutR* with Other Software for Time-to-Event Analysis. The table contrasts the key capabilities of the proposed *OptSurvCutR* package against other R packages (*survminer*, *maxstat*) and foundational bioinformatics tools (*X-tile*, *Cutoff Finder*). **Platform** refers to the software's operating environment. **Multiple Cut-points** indicate the ability to identify more than one threshold simultaneously. **Cut-point Number Selection** refers to a data-driven function to determine the optimal number of cut-points. **Validation** indicates a built-in function, such as bootstrapping, to assess the stability of the thresholds. **Covariate Adjustment** refers to the ability to account for confounding variables during the optimisation process.

Feature	OptCutSurvR	survminer	maxstat	X-tile	Cutoff Finder
Platform	R Package	R Package	R Package	Standalone Software	Web App (Defunct)
Multiple Cut-points	Yes (Genetic Alg.)	No	No	Yes (Systematic, max 3 groups)	No
Cut-point Number Selection	Yes	No	No	No	No
Validation	Yes (Bootstrapping)	No	No	No	No
Covariate Adjustment	Yes (Genetic Alg.)	No	No	No	No

The Importance of a Standardised Workflow

The development of *OptSurvCutR* also addresses the need for reproducible research in biomarker studies. By encapsulating these complex statistical procedures within a single, well-documented R package, we provide a transparent and standardised tool that enhances the reproducibility of cut-point analyses compared to the use of ad-hoc scripts. When compared to “black-box” machine learning models, the outputs of OptCutSurvR—such as Kaplan-Meier curves and forest plots—are highly interpretable, which is a significant advantage in clinical and biological research where understanding the “why” is as important as the prediction itself.

Limitations and Future Directions

Despite its strengths, *OptSurvCutR* has limitations that present opportunities for future development. The systematic search for multiple cut-points (*max_cuts* > 1) is computationally intensive and not recommended for large datasets; the genetic algorithm is a more practical alternative in these cases. While the current implementation of covariate adjustment is a key feature, future versions could expand this to the systematic search method for a more comprehensive analysis. Finally, while the bootstrap validation is robust, the addition of a formal permutation testing method to *find_cutpoint_number()* would provide an even more

rigorous assessment of statistical significance and further protect against the multiple comparisons problem. To broaden the package's impact and make it accessible to researchers who are not proficient in R, a key future direction is the development of a graphical user interface (GUI) using the R Shiny framework. This would provide an interactive, code-free platform that could be bundled within the package for local use or deployed as a public web application.

Conclusions

In conclusion, *OptSurvCutR* is a powerful and versatile R package that provides a complete, end-to-end workflow for optimal cut-point analysis in a time-to-event context. By integrating tools for determining the number of cuts, finding their locations with multiple algorithms, and validating the results, it enables researchers to perform more rigorous, reproducible, and nuanced analyses across diverse scientific disciplines.

DECLARATIONS

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

The *OptSurvCutR* package is freely available on GitHub at https://github.com/paytonyau/OptSurvCutR.

The rapeseed germination dataset used in Case Study 1 was computationally simulated. The R code to generate this dataset and to reproduce the complete analysis is provided within the package's vignette, which is accessible via the command browseVignettes(package = "OptSurvCutR").

Competing interests

The authors declare that they have no competing interests.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

REFERENCES

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*(6), 716–723. https://doi.org/10.1109/TAC.1974.1100705

Alboukadel Kassambara, Marcin Kosinski, Przemyslaw Biecek, & Scheipl Fabian. (2024). *survminer: Drawing survival curves using 'ggplot2'* (Version 0.5.0) [R package]. https://CRAN.R-project.org/package=survminer

Altman, D. G., Lausen, B., Sauerbrei, W., & Schumacher, M. (1994). Dangers of Using "Optimal" Cutpoints in the Evaluation of Prognostic Factors. *JNCI: Journal of the National Cancer Institute*, *86*(11), 829–835. https://doi.org/10.1093/jnci/86.11.829

Budczies, J., Klauschen, F., Sinn, B. V., Győrffy, B., Schmitt, W. D., Darb-Esfahani, S., & Denkert, C. (2012). Cutoff Finder: A Comprehensive and Straightforward Web Application Enabling Rapid Biomarker Cutoff Optimization. *PLOS ONE*, *7*(12), e51862. https://doi.org/10.1371/journal.pone.0051862

Camp, R. L., Dolled-Filhart, M., & Rimm, D. L. (2004). X-Tile: A New Bio-Informatics Tool for Biomarker Assessment and Outcome-Based Cut-Point Optimization. *Clinical Cancer Research*, *10*(21), 7252–7259. https://doi.org/10.1158/1078-0432.CCR-04-0713

Chang, C., Hsieh, M.-K., Chang, W.-Y., Chiang, A. J., & Chen, J. (2017). Determining the optimal number and location of cutoff points with application to data of cervical cancer. *PLOS ONE*, *12*(4), e0176231. https://doi.org/10.1371/journal.pone.0176231

Corporation & Weston. (2022). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package* (Version 1.0.17) [R package]. https://CRAN.R-project.org/package=doParallel

Cox, D. R. (1972). Regression Models and Life-Tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, *34*(2), 187–202. https://doi.org/10.1111/j.2517-6161.1972.tb00899.x

Csárdi, G. (2017). *cli: Helpers for Developing Command Line Interfaces* (p. 3.6.5) [Data set]. https://doi.org/10.32614/CRAN.package.cli

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, *7*(1), 1–26. https://doi.org/10.1214/aos/1176344552

Gideon Schwarz. (1978). Estimating the dimension of a model. *The Annals of Statistics*, *6*(2), 461–464. https://doi.org/10.1214/aos/1176344136

Haj Sghaier, A., Tarnawa, Á., Khaeim, H., Kovács, G. P., Gyuricza, C., & Kende, Z. (2022). The Effects of Temperature and Water on the Seed Germination and Seedling Development of Rapeseed (Brassica napus L.). *Plants*, *11*(21), 2819. https://doi.org/10.3390/plants11212819

Hurvich, C. M., & Tsai, C.-L. (1989). Regression and time series model selection in small samples. *Biometrika*, *76*(2), 297–307. https://doi.org/10.1093/biomet/76.2.297

Kassambara, A., Kosinski, M., & Biecek, P. (2016). *survminer: Drawing Survival Curves using 'ggplot2'* (p. 0.5.1) [Data set]. https://doi.org/10.32614/CRAN.package.survminer

Mantel, N. (1966). Evaluation of survival data and two new rank order statistics arising in its consideration. *Cancer Chemotherapy Reports*, *50*(3). https://pubmed.ncbi.nlm.nih.gov/5910392/

Mayeux, R. (2004). Biomarkers: Potential uses and limitations. *NeuroRX*, *1*(2), 182–188. https://doi.org/10.1602/neurorx.1.2.182

Mebane Jr, W. R., & Sekhon, J. S. (2011). Genetic Optimization Using Derivatives: The rgenoud Package for R. *Journal of Statistical Software*, *42*, 1–26. https://doi.org/10.18637/jss.v042.i11

R Core Team. (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Royston, P., Altman, D. G., & Sauerbrei, W. (2006). Dichotomizing continuous predictors in multiple regression: A bad idea. *Statistics in Medicine*, *25*(1), 127–141. https://doi.org/10.1002/sim.2331

Smyth, J., Godet, J., Choudhary, A., Das, A., Gkoutos, G. V., & Acharjee, A. (2024). Microbiome-Based Colon Cancer Patient Stratification and Survival Analysis. *Cancer Medicine*, *13*(22), e70434. https://doi.org/10.1002/cam4.70434

Therneau, T. (2024). *A Package for Survival Analysis in R* (Version 3.8-3) [R package]. https://CRAN.R-project.org/package=survival

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T., Miller, E., Bache, S., Müller, K., Ooms, J., Robinson, D., Seidel, D., Spinu, V., … Yutani, H. (2019). Welcome to the Tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Wickham, H., François, R., Henry, L., Müller, K., & Vaughan, D. (2014). *dplyr: A Grammar of Data Manipulation* (p. 1.1.4) [Data set]. https://doi.org/10.32614/CRAN.package.dplyr

Supplementary information 1

Generation of the germination Dataset

The germination dataset is a simulated dataset created to rigorously test the functionalities of the *OptSurvCutR* package, particularly its ability to identify multiple cut-points in data with non-linear relationships. The simulation process is grounded in the empirical findings reported by Haj Sghaier *et al*. (2022) on rapeseed germination and was conducted in two main parts.

Part 1: Simulation from Manuscript Parameters. The first part of the dataset was generated using parameters derived directly from the source manuscript. This included data for seven constant experimental temperatures (5, 10, 15, 20, 25, 30, and 35°C). For each temperature, we defined the key parameters that govern the germination process:

- **Time Window:** The *start_day* and *end_day* of the germination period.
- **Growth Rate:** The slope and intercept for a linear growth model.
- **Success Rate:** The final *germination_rate* observed at that temperature.

Using these parameters, a dataset was simulated to reflect the outcomes at these specific temperature points.

Part 2: Simulation from Interpolated Data. To provide a more continuous predictor variable and create a more challenging optimisation problem, a second dataset was simulated. This was achieved by linearly interpolating the key parameters (*slope*, *intercept*, *germination_rate*, etc.) for six intermediate temperature points (7.5, 12.5, 17.5, 22.5, 27.5, and 32.5°C) that were not present in the original study.

The final germination data object included in the package is the combination of these two simulated subsets. This hybrid approach yields a dataset that honours the original experimental findings while also providing the continuity needed to robustly test and demonstrate cut-point finding algorithms.

```
# ====================================================================
# DATA GENERATION SCRIPT for the ‘germination’ dataset.
# This script simulates rapeseed germination data based on parameters from
# Haj Sghaier et al. (2022) and saves the final object to data/germination.rda
# ====================================================================

# Set a global seed for full reproducibility of the simulation
set.seed(2025)

# --- 1. Define Parameters & Core Simulation Function ---

# Original parameters from the manuscript
original_params <- data.frame(
  temperature = c(5, 10, 15, 20, 25, 30, 35),
  start_day = c(13, 7, 4, 4, 3, 4, 9),
  end_day = c(25, 20, 16, 16, 15, 16, 20),
  slope = c(0.9846, 3.6848, 4.9765, 3.8693, 6.5766, 1.1384, 0.1560),
  intercept = c(-13.627, -34.076, -28.116, -17.012, -35.621, -1.2833, -1.0571),
  r_squared = c(0.9585, 0.9501, 0.9307, 0.9524, 0.9564, 0.9237, 0.9574),
  germination_rate = c(0.88, 0.90, 0.92, 0.98, 0.94, 0.91, 0.55)
```

```
)

# Core data generation function (unchanged)
generate_hybrid_data <- function(params) {
  set.seed(as.integer(params$temperature * 10))
  n_replicates <- 4
  n_seeds_per_dish <- 20
  n_samples_total <- n_replicates * n_seeds_per_dish

  time_data <- round(runif(n_samples_total, min = params$start_day, max = params$end_day))

  perfect_growth <- params$intercept + params$slope * time_data
  perfect_growth_variance <- var(perfect_growth)
  if (params$r_squared < 1 && params$r_squared > 0) {
    residual_variance <- perfect_growth_variance * (1 - params$r_squared) / params$r_squared
  } else {
    residual_variance <- 0
  }
  residual_std_dev <- sqrt(residual_variance)
  residuals <- rnorm(n = n_samples_total, mean = 0, sd = residual_std_dev)
  actual_growth <- perfect_growth + residuals
  actual_growth[actual_growth < 0] <- 0

  total_germinated <- round(n_samples_total * params$germination_rate)
  status_vector <- c(rep(1, total_germinated), rep(0, n_samples_total - total_germinated))
  shuffled_status <- sample(status_vector)

  replicate_vector <- rep(1:n_replicates, each = n_seeds_per_dish)
  actual_growth[shuffled_status == 0] <- 0

  data.frame(
    temperature = params$temperature,
    replicate = replicate_vector,
    time = time_data,
    growth = actual_growth,
    germinated = shuffled_status
  )
}


# --- 2. Create Interpolated Parameters ---

# Define the new temperature points and the parameters to interpolate
new_temps <- c(7.5, 12.5, 17.5, 22.5, 27.5, 32.5)
params_to_interp <- c("slope", "intercept", "r_squared", "germination_rate", "start_day", "end_day")

# Use sapply to apply the interpolation to all specified parameters at once
interpolated_values <- sapply(params_to_interp, function(param) {
  approx(original_params$temperature, original_params[[param]], xout = new_temps)$y
})

interpolated_params <- as.data.frame(interpolated_values)
interpolated_params$temperature <- new_temps
```

```
# --- 3. Combine Parameters and Run Full Simulation ---

# Combine original and interpolated parameters into a single data frame
all_params <- rbind(original_params, interpolated_params)

# Apply the generation function to every row of the combined parameter set
all_data_list <- lapply(1:nrow(all_params), function(i) generate_hybrid_data(all_params[i, ]))

# Combine the list of data frames into one final object named ‘germination’
germination <- do.call(rbind, all_data_list)


# --- 4. Save the Final Data Object for the Package ---
usethis::use_data(germination, overwrite = TRUE)
```