The Biostatistician's Guide to "Skeptic Reviewer"-Proof Multivariable Modeling

A Practical Guide for Clinical Researchers and Biostatisticians

Covering: Linear, Logistic, Cox, and Fine-Gray Models

With complete R code, mathematical proofs, and copy-paste-ready methods paragraphs

Table of Contents

1. Introduction

1.1 Who This Guide Is For

This guide is written for biostatisticians and clinical researchers who need to build multivariable regression models that will survive peer review. Whether you are a beginner learning variable selection or an experienced analyst defending your methods, this document provides the mathematical justification, code, and exact language you need.

Clinical Studies Focus: This guide emphasizes methods appropriate for clinical research, including: observational cohort studies, randomized controlled trials with secondary analyses, prognostic model development, and risk factor identification studies. All examples, code, and recommendations are tailored for clinical outcome data including continuous endpoints (e.g., blood pressure, quality of life scores), binary outcomes (e.g., treatment response, mortality), and time-to-event outcomes (e.g., survival, disease recurrence).

1.2 Why Reviewers Attack Variable Selection

Stepwise variable selection has been heavily criticized in the statistical literature. Harrell (2015) called it *"one of the most controversial and misused techniques in applied statistics."* The main criticisms include:

1. Inflated Type I error: Bootstrap analyses reveal substantial instability in variable selection (Altman & Andersen, 1989).

2. Biased coefficient estimates: Selected coefficients are biased away from zero (Miller, 1984).

3. Underestimated standard errors: The selection process is ignored in variance estimation (Hurvich & Tsai, 1990).

4. Instability: Small data changes lead to completely different models (Derksen & Keselman, 1992).

Reference: *Harrell FE. Regression Modeling Strategies. 2nd ed. Springer; 2015. Chapter 4.*

1.3 When Stepwise IS Defensible

Despite these criticisms, automated variable selection remains common and is defensible when:

1. The goal is prediction, not inference (Shmueli, 2010)

2. You use information criteria (AIC/BIC) rather than p-value thresholds (Burnham & Anderson, 2002)

3. You report sensitivity analyses showing model stability

4. You pre-specify candidate variables based on clinical knowledge (Heinze et al., 2018)

5. Sample size is adequate: at least 10-20 events per candidate variable (Peduzzi et al., 1996; Vittinghoff & McCulloch, 2007)

Reference: *Heinze G, Wallisch C, Dunkler D. Variable selection - A review and recommendations for the practicing statistician. Biom J. 2018;60(3):431-449.*

2. The Four Regression Workhorses

Before discussing variable selection, you must choose the appropriate regression model for your outcome type.

Model	Outcome Type	R Function	Effect Measure
Linear	Continuous (e.g., BP, BMI)	lm()	Mean difference (beta)
Logistic	Binary (yes/no)	glm(..., family=binomial)	Odds Ratio (OR)
Cox	Time-to-event	survival::coxph()	Hazard Ratio (HR)
Fine-Gray	Competing risks	tidycmprsk::crr()	Subdistribution HR (sHR)

2.1 Linear Regression

When to use: Continuous outcome that is approximately normally distributed. Examples: systolic blood pressure, HbA1c, quality of life scores.

Key assumption: Residuals are normally distributed with constant variance (homoscedasticity).

Reference: *Vittinghoff E, Glidden DV, Shiboski SC, McCulloch CE. Regression Methods in Biostatistics. 2nd ed. Springer; 2012.*

2.2 Logistic Regression

When to use: Binary outcome (yes/no, success/failure). Examples: 30-day mortality, disease recurrence.

Key assumption: Observations are independent; linear relationship between predictors and log-odds.

Reference: *Hosmer DW, Lemeshow S, Sturdivant RX. Applied Logistic Regression. 3rd ed. Wiley; 2013.*

2.3 Cox Proportional Hazards

When to use: Time-to-event outcome. Examples: overall survival, time to disease progression.

Key assumption: Proportional hazards - the hazard ratio is constant over time. Test with cox.zph().

Reference: *Therneau TM, Grambsch PM. Modeling Survival Data: Extending the Cox Model. Springer; 2000.*

2.4 Fine-Gray Competing Risks

When to use: Time-to-event with competing risks that prevent the event of interest.

Key distinction: Cox models cause-specific hazard; Fine-Gray models subdistribution hazard (cumulative incidence).

Reference: *Fine JP, Gray RJ. A proportional hazards model for the subdistribution of a competing risk. J Am Stat Assoc. 1999;94(446):496-509.*

3. Variable Selection Philosophy

3.1 Pre-specification vs. Data-Driven Selection

Pre-specified models: Variables chosen BEFORE looking at the data, based on clinical knowledge. This preserves nominal Type I error rates.

Data-driven selection: Variables chosen based on the data (stepwise, LASSO). This inflates Type I error but may improve prediction.

Recommendation: "For confirmatory analyses, pre-specification is strongly recommended. For exploratory analyses, data-driven methods may be used but should be clearly labeled as hypothesis-generating." (Heinze et al., 2018)

3.2 The Principle of Parsimony (Occam's Razor)

Definition: Parsimony (from Latin 'parcere' = to spare) is the principle that simpler explanations are preferable to complex ones, all else being equal. In statistics, this translates to: prefer models with fewer parameters that adequately explain the data.

3.2.1 Why Parsimony Matters

1. Overfitting Prevention

Complex models with many parameters can fit the training data perfectly but fail to generalize to new data. Each additional parameter 'uses up' information from the data, leaving less for genuine signal detection.

2. Bias-Variance Tradeoff

Adding variables reduces bias (the model can capture more patterns) but increases variance (estimates become unstable). Parsimonious models find the optimal balance.

```
Total Error = Bias² + Variance + Irreducible Error


Too few variables → High bias, low variance (underfitting)
Too many variables → Low bias, high variance (overfitting)
Optimal → Balance that minimizes total error
```

3. Interpretability

A model with 3 predictors is easier to explain to clinicians and implement in practice than one with 15. Parsimonious models facilitate clinical decision-making.

4. Reproducibility

Complex models are more likely to capitalize on sample-specific noise. Parsimonious models are more likely to replicate in independent datasets.

5. Statistical Power

Each parameter estimated 'costs' degrees of freedom. Parsimonious models preserve power for testing the effects that truly matter.

3.2.2 Parsimony in Practice

The fundamental question: Does adding this variable improve the model enough to justify the added complexity?

Information criteria formalize this question:

```
IC = -2 log L + penalty × k

where:
  -2 log L = measure of model fit (lower = better fit)
  k = number of parameters
  penalty = cost per parameter

The penalty term implements parsimony:
  Higher penalty → More parsimonious models preferred
  Lower penalty → Better-fitting models preferred
```

3.2.3 AIC vs BIC: Different Parsimony Philosophies

Criterion	Penalty	Philosophy	Parsimony Level
AIC	2 per parameter	Find best predictive model	Moderate
BIC	log(n) per parameter	Find true generating model	Strong
$p < 0.05$	~3.84 per parameter	Traditional threshold (not selection-valid)	Weak
LASSO	Continuous shrinkage	Automatic selection + shrinkage	Adaptive

When n > 8: BIC penalty > AIC penalty, so BIC produces more parsimonious models.

As n → ∞: BIC penalty grows without bound, eventually selecting the true model (consistency).

3.2.4 The Parsimony-Completeness Tension

Caution: Excessive parsimony can be harmful!

There is tension between parsimony and capturing all important effects:

• **Omitted variable bias:** Excluding a confounder that affects both exposure and outcome biases the exposure effect estimate.

• **Loss of precision:** Excluding a strong predictor of the outcome (even if unrelated to exposure) inflates residual variance and widens confidence intervals.

• **Clinical incompleteness:** A model missing a clinically important predictor may fail in subgroups where that predictor matters.

Best Practice: Start with clinically motivated candidate variables, then apply parsimony within that set.

Reference: *Burnham KP, Anderson DR. Model Selection and Multimodel Inference. 2nd ed. Springer; 2002. Chapter 2: Information and Likelihood Theory.*

Reference: *Harrell FE. Regression Modeling Strategies. 2nd ed. Springer; 2015. Section 4.3: Parsimony.*

3.3 Information Criteria

Information criteria formalize the parsimony vs. prediction tradeoff:

AIC = -2 log L + 2k (Akaike, 1974) - smaller penalty, favors complex models

BIC = -2 log L + k log(n) (Schwarz, 1978) - larger penalty, favors parsimonious models

where L = likelihood, k = number of parameters, n = sample size.

Reference: *Burnham KP, Anderson DR. Model Selection and Multimodel Inference. 2nd ed. Springer; 2002.*

3.3 Events Per Variable (EPV) Rule

For binary and survival outcomes, inadequate sample size leads to overfitting. The classic rule-of-thumb requires at least 10 events per candidate variable (Peduzzi et al., 1996), though 20+ may be needed for reliable selection. Note: This is a rough heuristic - modern guidance suggests EPV requirements depend on outcome fraction, effect sizes, and shrinkage methods (Riley et al., 2020).

Example: With 80 events, use at most 8 candidate variables (80/10 = 8).

Reference: *Peduzzi P, et al. A simulation study of the number of events per variable in logistic regression analysis. J Clin Epidemiol. 1996;49(12):1373-1379.*

4. Essential Statistical Concepts Glossary

This section defines key terms every biostatistician must understand for defensible modeling.

4.1 Regularization and Shrinkage

Regularization: A technique that adds a penalty term to the loss function to prevent overfitting by constraining coefficient magnitudes. The model trades some bias for reduced variance.

```
Regularized Loss = Original Loss + λ × Penalty


Three main types:
 Ridge (L2):    Penalty = Σ β²   → Shrinks coefficients toward zero
 LASSO (L1):    Penalty = Σ |β|   → Shrinks AND sets some to exactly zero
 Elastic Net:   Penalty = α|β| + (1-α)β²  → Combines both
```

Shrinkage: The phenomenon where estimated coefficients are pulled toward zero (or another target). Occurs naturally in regularization, but also recommended post-hoc to correct for optimism in stepwise selection.

```
# Shrinkage factor estimation (van Houwelingen method)
shrinkage_factor <- (model_chi_sq - df) / model_chi_sq


# Apply shrinkage to coefficients
shrunken_coef <- original_coef * shrinkage_factor


# In R with rms package:
library(rms)
model <- lrm(y ~ x1 + x2, data = mydata, x = TRUE, y = TRUE)
validate(model)  # Shows optimism-corrected performance
```

Penalization: Synonym for regularization. The penalty parameter λ (lambda) controls the strength: λ=0 gives OLS/MLE; λ→∞ shrinks all coefficients to zero.

Reference: *Tibshirani R. Regression shrinkage and selection via the lasso. J R Stat Soc B. 1996;58(1):267-288.*

4.2 Calibration

Calibration: The agreement between predicted probabilities and observed outcomes. A well-calibrated model that predicts 30% risk should see ~30% events in that group.

Why it matters: A model can have excellent discrimination (separating high vs low risk) but poor calibration (predicted probabilities are wrong). Both are needed for clinical utility.

```
# Calibration assessment methods:

# 1. Calibration plot (graphical)
library(rms)
model <- lrm(y ~ x1 + x2, data = mydata, x = TRUE, y = TRUE)
cal <- calibrate(model, B = 200)  # Bootstrap calibration
plot(cal)

# 2. Hosmer-Lemeshow test (formal test - but has limitations)
library(ResourceSelection)
hoslem.test(y, fitted(model), g = 10)

# 3. Calibration slope and intercept (preferred)
# CORRECT calibration intercept/slope assessment:
# lp = linear predictor on logit scale
lp <- predict(model, type = 'lp')  # linear predictor

# Calibration-in-the-large (intercept only, slope fixed at 1)
fit_int <- glm(y ~ offset(lp), family = binomial)
cal_intercept_only <- coef(fit_int)[1]  # Ideal: 0

# Calibration slope AND intercept (both estimated)
fit_slope <- glm(y ~ lp, family = binomial)
cal_intercept <- coef(fit_slope)[1]  # Ideal: 0
cal_slope <- coef(fit_slope)[2]      # Ideal: 1
# NOTE: Slope < 1 indicates overfitting; slope > 1 indicates underfitting

# 4. Brier score decomposition
# Brier = Reliability - Resolution + Uncertainty
```

Calibration-in-the-large: Compares mean predicted probability to observed event rate. Detects systematic over/under-prediction.

Calibration slope: Ideal = 1.0. Slope < 1 indicates overfitting (predictions too extreme). Slope > 1 indicates underfitting.

Reference: *Steyerberg EW, Vickers AJ, et al. Assessing the performance of prediction models. Epidemiology. 2010;21(1):128-138.*

4.3 Discrimination

Discrimination: The model's ability to separate subjects who experience the outcome from those who do not. High discrimination = good ranking of risk.

```
# Key discrimination metrics:

# 1. AUC / C-statistic (concordance)
# Probability that a random case ranks higher than a random control
# Range: 0.5 (random) to 1.0 (perfect)
library(pROC)
roc_obj <- roc(y, predicted_prob)
auc(roc_obj)  # AUC with 95% CI

# For survival models: Harrell's C
library(survival)
concordance(coxph_model)

# 2. Sensitivity and Specificity (at a threshold)
# Sensitivity = TP / (TP + FN) = true positive rate
# Specificity = TN / (TN + FP) = true negative rate
coords(roc_obj, 'best', ret = c('threshold', 'sensitivity', 'specificity'))

# 3. ROC curve
plot(roc_obj, print.auc = TRUE)

# Interpretation guidelines (Hosmer & Lemeshow):
# AUC 0.5-0.6: No discrimination
# AUC 0.6-0.7: Poor
# AUC 0.7-0.8: Acceptable
# AUC 0.8-0.9: Excellent
# AUC > 0.9: Outstanding (rare in medicine)
```

Reference: *Harrell FE, Lee KL, Mark DB. Multivariable prognostic models. Stat Med. 1996;15(4):361-387.*

4.4 Validation

Internal Validation: Assessing model performance using the same data used for development. Corrects for optimism (overfitting).

External Validation: Assessing performance in completely independent data. The gold standard for generalizability.

```
# Internal validation methods:

# 1. Bootstrap validation (preferred - Harrell)
library(rms)
model <- lrm(y ~ x1 + x2, data = mydata, x = TRUE, y = TRUE)
v <- validate(model, B = 200)  # 200 bootstrap samples
print(v)  # Shows index.orig, training, test, optimism, index.corrected

# 2. K-fold cross-validation
library(caret)
ctrl <- trainControl(method = 'cv', number = 10,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
cv_model <- train(y ~ x1 + x2, data = mydata, method = 'glm',
                  family = 'binomial', trControl = ctrl, metric = 'ROC')

# 3. Leave-one-out cross-validation (LOOCV)
# Special case of k-fold where k = n
# Computationally expensive, high variance

# 4. Split-sample (train/test split)
# Simple but wasteful - loses power
# Use only with very large datasets (n > 10,000)
set.seed(123)
train_idx <- sample(nrow(mydata), 0.7 * nrow(mydata))
train_data <- mydata[train_idx, ]
test_data <- mydata[-train_idx, ]
```

Optimism: The difference between apparent performance (on training data) and true performance (on new data). Stepwise selection increases optimism substantially.

```
# Optimism-corrected performance:
Corrected_AUC = Apparent_AUC - Optimism

# Rule of thumb for expected optimism:
# - Pre-specified model: ~0.01-0.02 AUC optimism
# - Stepwise selection: can produce substantial optimism (quantify via
bootstrap)
# - LASSO with CV: typically lower optimism than stepwise
# Always quantify optimism empirically rather than assuming universal ranges
```

Reference: *Steyerberg EW. Clinical Prediction Models. 2nd ed. Springer; 2019. Chapter 5: Overfitting and Optimism.*

4.5 Cross-Validation

K-Fold Cross-Validation: Data split into k equal parts. Model trained on k-1 parts, tested on remaining part. Repeated k times. Final performance = average across folds.

```
# Common choices:
# k = 5:  Good bias-variance tradeoff, fast
# k = 10: Standard choice, slightly more variance
# k = n:  LOOCV - low bias, high variance, slow

# For variable selection within CV:
library(caret)

# WRONG: Select variables first, then CV (data leakage!)
# selected <- step(full_model)  # Uses ALL data
# cv_results <- train(selected_formula, ...)  # Optimistic!

# CORRECT: Selection WITHIN each fold
ctrl <- trainControl(method = 'cv', number = 10)
model <- train(y ~ ., data = mydata,
               method = 'glmStepAIC',  # Stepwise within each fold
               trControl = ctrl,
               trace = FALSE)
```

Reference: *Hastie T, Tibshirani R, Friedman J. Elements of Statistical Learning. 2nd ed. Springer; 2009. Chapter 7.*

4.6 Multicollinearity

Multicollinearity: High correlation among predictors. Causes unstable coefficient estimates (large SEs) even when the overall model fits well.

```
# Detection methods:

# 1. Variance Inflation Factor (VIF)
library(car)
vif(model)
# VIF > 5: Moderate concern
# VIF > 10: Serious problem
```

```
# 2. Condition number
library(perturb)
colldiag(model)
# Condition index > 30: Problematic

# 3. Correlation matrix
cor(model.matrix(model)[, -1])  # Exclude intercept
# |r| > 0.7: Concerning

# Solutions:
# - Remove one of correlated variables
# - Combine into composite/index
# - Use ridge regression or PCA
# - Center continuous predictors
```

Impact on hybrid method: High multicollinearity inflates Wald SEs disproportionately, potentially causing divergence from LRT-based methods.

Reference: *Belsley DA, Kuh E, Welsch RE. Regression Diagnostics. Wiley; 1980.*

4.7 Goodness of Fit

Goodness of fit: Overall assessment of how well the model describes the data. Different from discrimination and calibration.

```
# Metrics by model type:

# Linear regression:
summary(lm_model)$r.squared      # R² (0-1)
summary(lm_model)$adj.r.squared  # Adjusted R²

# Logistic regression:
# Pseudo-R² (many versions - use cautiously)
library(pscl)
pR2(glm_model)  # McFadden, Cox-Snell, Nagelkerke

# Deviance and likelihood ratio test
anova(null_model, full_model, test = 'Chisq')

# Hosmer-Lemeshow (calibration, not overall fit)
# Note: Sensitive to group number; not recommended as primary metric
```

```
# Cox regression:
summary(cox_model)$concordance   # C-statistic
summary(cox_model)$rsq           # R² (Kent & O'Quigley)
```

4.8 Model Comparison Metrics

Net Reclassification Improvement (NRI): Quantifies how a new model improves risk classification compared to an old model.

Integrated Discrimination Improvement (IDI): Average improvement in predicted probabilities for cases vs. controls.

```
library(nricens)  # For survival data
library(PredictABEL)  # For binary outcomes

# Binary outcome NRI/IDI
reclassification(data = mydata,
                 cOutcome = 'outcome',
                 predrisk1 = pred_old,
                 predrisk2 = pred_new,
                 cutoff = c(0, 0.1, 0.2, 1))

# Continuous NRI (category-free)
# Preferred over categorical NRI
nri <- improveProb(pred_old, pred_new, outcome)

# Interpretation:
# NRI > 0: New model improves classification
# Report event NRI and non-event NRI separately
```

Decision Curve Analysis: Evaluates clinical utility by quantifying net benefit across threshold probabilities.

```
library(dcurves)

# Decision curve
dca(outcome ~ pred_model1 + pred_model2, data = mydata)

# Interpretation:
# Model is useful if its curve is above 'treat all' and 'treat none'
# Higher net benefit = more clinical utility
```

Reference: *Pencina MJ, D'Agostino RB, Vasan RS. Evaluating the added predictive ability of a new marker. Stat Med. 2008;27(2):157-172.*

Reference: *Vickers AJ, Elkin EB. Decision curve analysis. Med Decis Making. 2006;26(6):565-574.*

4.9 Separation and Convergence Issues

Complete Separation: A predictor (or combination) perfectly predicts the outcome. MLE does not exist; coefficients → ±∞.

Quasi-complete Separation: Predictor almost perfectly predicts outcome. MLE may exist but is unstable with huge SEs.

```
# Warning signs:
# - Coefficients > 10 or < -10
# - Standard errors > 5
# - Warning: 'fitted probabilities 0 or 1 occurred'
# - Warning: 'algorithm did not converge'

# Detection:
library(detectseparation)
glm(y ~ x, family = binomial, data = mydata, method = 'detect_separation')

# Solutions:
# 1. Firth's penalized likelihood (recommended)
library(logistf)
logistf(y ~ x1 + x2, data = mydata)

# 2. Exact logistic regression (small samples)
library(elrm)

# 3. Bayesian with informative priors
library(rstanarm)
stan_glm(y ~ x, family = binomial, prior = normal(0, 2.5))
```

In separation: Wald p-values are WRONG (too large). LRT and score tests may be more reliable but standard asymptotics are still questionable when MLE doesn't exist - use Firth penalization or exact methods.

Reference: *Heinze G, Schemper M. A solution to the problem of separation in logistic regression. Stat Med. 2002;21(16):2409-2419.*

4.10 Concordance (C-statistic / C-index)

Concordance: The probability that, for a randomly selected pair of subjects where one has the outcome and one does not, the model correctly assigns a higher risk to the subject with the outcome.

```
# Interpretation:
# C = 0.5: Model is no better than chance (coin flip)
# C = 1.0: Perfect discrimination
# C = 0.7-0.8: Acceptable for most clinical models

# For logistic regression (equivalent to AUC):
library(pROC)
roc_obj <- roc(outcome ~ predicted_prob, data = mydata)
auc(roc_obj)

# For Cox regression (Harrell's C):
library(survival)
cox_model <- coxph(Surv(time, status) ~ x1 + x2, data = mydata)
concordance(cox_model)

# Alternative with survcomp package:
library(survcomp)
concordance.index(predicted_risk, time, status)

# For competing risks:
library(riskRegression)
Score(list('Model' = fg_model), formula = Hist(time, status) ~ 1,
      data = mydata, times = c(1, 3, 5))
```

Somers' D: Related measure: D = 2(C - 0.5). Ranges from -1 to +1, with 0 = no association.

Clinical Interpretation Guidelines for Concordance:

C = 0.50-0.60: Poor discrimination - model barely better than chance; unlikely to be clinically useful

C = 0.60-0.70: Weak discrimination - may identify high-risk groups but individual predictions unreliable

C = 0.70-0.80: Acceptable discrimination - suitable for most clinical prediction models

C = 0.80-0.90: Excellent discrimination - model can reliably separate high and low risk patients

C > 0.90: Outstanding discrimination - rare in clinical practice; verify for overfitting

Important Clinical Note: Good discrimination (high C) does not guarantee good calibration. A model can perfectly rank patients by risk (C=1.0) while systematically over- or under-estimating absolute risk. Always report both discrimination AND calibration.

Reference: *Harrell FE, Lee KL, Mark DB. Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. Stat Med. 1996;15(4):361-387.*

4.11 Condition Number and Collinearity Diagnostics

Condition Number: A measure of numerical stability of the design matrix. High values indicate near-collinearity that can cause computational problems and unstable estimates.

```
# Condition number calculation:
X <- model.matrix(model)[, -1]  # Remove intercept
kappa(X)  # Base R - condition number

# Interpretation:
# κ < 10: No problem
# κ = 10-30: Moderate collinearity
# κ > 30: Severe collinearity - results may be unstable

# Detailed collinearity diagnostics:
library(perturb)
colldiag(model, scale = TRUE, center = TRUE)

# Output shows:
# - Condition indices for each dimension
# - Variance decomposition proportions
# - High proportion (>0.5) on high index = problematic variable

# Alternative with car package:
library(car)
vif(model)  # Variance Inflation Factors

# Eigenvalue analysis:
eigen(cor(X))$values
# Small eigenvalues (< 0.01) indicate collinearity
```

Relationship to VIF: VIF = 1/(1-R^2) where R^2 is from regressing each predictor on all others. VIF and condition number both detect collinearity but measure different aspects.

Clinical Context for Collinearity:

In clinical studies, collinearity often occurs with: (1) related biomarkers (e.g., creatinine and BUN), (2) polynomial terms (age and age²), (3) dummy variables from categorical

variables with many levels, (4) indices that share components (e.g., BMI includes weight). While severe collinearity inflates standard errors and makes individual coefficients uninterpretable, it does NOT bias predictions. If prediction is your goal, moderate collinearity may be acceptable. For inference, consider combining collinear variables or removing redundant predictors.

Reference: *Belsley DA, Kuh E, Welsch RE. Regression Diagnostics: Identifying Influential Data and Sources of Collinearity. Wiley; 1980.*

4.12 drop1() and anova() - Understanding the Difference

drop1(): Tests each term by comparing the full model to a model with that term removed. Gives TYPE III tests (each term adjusted for all others).

anova(): Sequential (TYPE I) tests - each term tested after adjusting only for terms already in the model. Order matters!

```
# CRITICAL DIFFERENCE:

model <- lm(y ~ A + B + C, data = mydata)

# drop1() - Type III (what you usually want)
drop1(model, test = 'F')  # For lm
drop1(model, test = 'Chisq')  # For glm, coxph

# Tests:
# A | adjusting for B and C
# B | adjusting for A and C
# C | adjusting for A and B
# Order doesn't matter - each term tested fairly

# anova() - Type I (sequential)
anova(model)

# Tests:
# A | adjusting for nothing
# B | adjusting for A only
# C | adjusting for A and B
# ORDER MATTERS! Different order = different p-values

# Demonstrating the difference:
anova(lm(y ~ A + B, data = mydata))  # B tested after A
```

```
anova(lm(y ~ B + A, data = mydata))  # A tested after B - DIFFERENT!


# For Type II tests (each term adjusted for all others at same level):
library(car)
Anova(model, type = 'II')  # Usually same as Type III for main effects
Anova(model, type = 'III')  # Requires sum-to-zero contrasts for interactions
```

RULE: Use drop1() for backward elimination. Use anova() only for comparing nested models.

Clinical Applications:

When to use drop1(): Backward elimination, identifying which predictors to remove, testing categorical variables as a whole unit (crucial for variables like tumor stage with multiple levels).

When to use anova(): Comparing specific nested model pairs, testing if adding a specific set of variables improves model fit (e.g., testing if biomarkers add value beyond clinical predictors).

Common Mistake: Using individual coefficient p-values from summary() to decide variable importance. For categorical variables, this tests only individual dummy levels against the reference, not whether the entire variable is significant. Always use drop1() for variable-level decisions in clinical models.

```
# Comparing nested models with anova():
model_full <- glm(y ~ x1 + x2 + x3, family = binomial, data = mydata)
model_reduced <- glm(y ~ x1 + x2, family = binomial, data = mydata)


anova(model_reduced, model_full, test = 'Chisq')  # LRT for x3


# This is equivalent to:
drop1(model_full, test = 'Chisq')['x3', ]
```

Reference: *Fox J, Weisberg S. An R Companion to Applied Regression. 3rd ed. Sage; 2019. Chapter 4.*

4.13 Interpreting Model Output

4.13.1 Linear Regression Output

```
model <- lm(sbp ~ age + bmi + smoking, data = mydata)
summary(model)


# Key components:
# Coefficients table:
#              Estimate Std.Error t value Pr(>|t|)
```

```
# (Intercept)   85.234    5.123   16.64  <2e-16  <- Mean SBP when all X=0
# age            0.523    0.089    5.88  4.2e-09 <- SBP increase per year
# bmi            1.234    0.312    3.96  8.1e-05 <- SBP increase per BMI unit
# smokingYes     5.678    1.456    3.90  1.1e-04 <- SBP difference: smokers
vs non


# Interpretation:
# 'Each 1-year increase in age is associated with 0.52 mmHg higher SBP
#  (95% CI: 0.35-0.70, p < 0.001), adjusting for BMI and smoking.'


# Model fit:
# Multiple R-squared:  0.234  <- 23.4% variance explained
# Adjusted R-squared:  0.229  <- Penalized for number of predictors
# F-statistic: 45.67 on 3 and 450 DF, p < 2.2e-16  <- Overall model test


# 95% Confidence intervals:
confint(model)
```

4.13.2 Logistic Regression Output

```
model <- glm(death ~ age + stage + treatment, family = binomial, data =
mydata)
summary(model)


# Coefficients are LOG ODDS RATIOS:
#              Estimate Std.Error z value Pr(>|z|)
# (Intercept)   -3.456    0.567   -6.10  1.1e-09
# age            0.045    0.012    3.75  1.8e-04  <- log(OR) per year
# stageIII       0.892    0.234    3.81  1.4e-04  <- log(OR) III vs I
# stageIV        1.567    0.289    5.42  6.0e-08  <- log(OR) IV vs I
# treatmentB    -0.654    0.198   -3.30  9.6e-04  <- log(OR) B vs A


# Convert to Odds Ratios:
exp(coef(model))  # Point estimates
exp(confint(model))  # 95% CI


# Interpretation:
# 'Stage IV was associated with 4.79 times higher odds of death compared to
#  Stage I (OR 4.79, 95% CI: 2.72-8.45, p < 0.001), adjusting for age and
#  treatment.'
```

```
# Model fit (no R² equivalent - use pseudo-R²):
library(pscl)
pR2(model)  # McFadden, Cox-Snell, Nagelkerke

# Overall model test:
null_model <- glm(death ~ 1, family = binomial, data = mydata)
anova(null_model, model, test = 'Chisq')
```

4.13.3 Cox Regression Output

```
library(survival)
model <- coxph(Surv(time, status) ~ age + stage + treatment, data = mydata)
summary(model)

# Coefficients are LOG HAZARD RATIOS:
#            coef exp(coef) se(coef)     z Pr(>|z|)
# age       0.034     1.035    0.011  3.09  0.0020  <- log(HR) per year
# stageIII  0.756     2.130    0.201  3.76  0.0002  <- log(HR) III vs I
# stageIV   1.234     3.435    0.245  5.04  <0.001  <- log(HR) IV vs I
# treatB   -0.543     0.581    0.178 -3.05  0.0023  <- log(HR) B vs A

# exp(coef) IS the Hazard Ratio:
# 'Stage IV was associated with 3.44 times higher hazard of death compared
#  to Stage I (HR 3.44, 95% CI: 2.13-5.55, p < 0.001).'

# Model fit:
# Concordance = 0.723 (se = 0.018)  <- C-statistic
# Likelihood ratio test = 45.67 on 4 df, p = 2.3e-09  <- Overall model
# Wald test = 42.34 on 4 df, p = 1.4e-08
# Score (logrank) test = 44.12 on 4 df, p = 5.6e-09
```

4.13.4 Fine-Gray Output

```
library(tidycmprsk)
model <- crr(Surv(time, factor(status)) ~ age + stage + treatment,
             data = mydata, failcode = 1)
summary(model)

# Coefficients are LOG SUBDISTRIBUTION HAZARD RATIOS:
# Interpretation similar to Cox, but for CUMULATIVE INCIDENCE:
```

```
# 'Treatment B was associated with 42% lower subdistribution hazard of
#  the primary event (sHR 0.58, 95% CI: 0.41-0.82, p = 0.002), accounting
#  for the competing risk of other-cause mortality.'

# Key difference from Cox:
# Cox HR: instantaneous rate among those still at risk
# Fine-Gray sHR: effect on cumulative incidence function
```

4.14 Assumption Checking and Diagnostics

4.14.1 Linear Regression Assumptions

```
model <- lm(y ~ x1 + x2 + x3, data = mydata)

# 1. LINEARITY - residuals vs fitted should show no pattern
plot(model, which = 1)

# 2. NORMALITY of residuals
plot(model, which = 2)  # Q-Q plot
shapiro.test(residuals(model))  # Formal test (use cautiously)

# 3. HOMOSCEDASTICITY (constant variance)
plot(model, which = 3)  # Scale-location plot
library(lmtest)
bptest(model)  # Breusch-Pagan test

# 4. INDEPENDENCE (no autocorrelation)
dwtest(model)  # Durbin-Watson test

# 5. NO INFLUENTIAL OUTLIERS
plot(model, which = 4)  # Cook's distance
plot(model, which = 5)  # Residuals vs leverage

# Influential observations:
influence.measures(model)
# DFBETAS: change in each coefficient if observation removed
# DFFITS: change in fitted value
# Cook's D: overall influence (>4/n is concerning)

# 6. NO MULTICOLLINEARITY
```

```
library(car)
vif(model)  # VIF > 5 concerning, > 10 serious
```

4.14.2 Logistic Regression Assumptions

```
model <- glm(y ~ x1 + x2 + x3, family = binomial, data = mydata)

# 1. LINEARITY in the logit (for continuous predictors)
# Method: Box-Tidwell test
library(car)
boxTidwell(y ~ x1 + x2, other.x = ~ x3, data = mydata)  # x3 is categorical

# Alternative: plot logit vs predictor using grouped means
mydata$x1_group <- cut(mydata$x1, breaks = 10)
group_means <- aggregate(y ~ x1_group, data = mydata, mean)
group_means$logit <- log(group_means$y / (1 - group_means$y))
plot(group_means$x1_group, group_means$logit)

# 2. NO SEPARATION
# Check for warnings in model output
# Or use:
library(detectseparation)
glm(y ~ x1 + x2, family = binomial, data = mydata, method =
'detect_separation')

# 3. GOODNESS OF FIT
library(ResourceSelection)
hoslem.test(model$y, fitted(model), g = 10)
# Note: HL test has limitations - use calibration plots instead

# 4. INFLUENTIAL OBSERVATIONS
plot(model, which = 4)  # Cook's distance
influencePlot(model)  # From car package

# 5. MULTICOLLINEARITY
vif(model)
```

4.14.3 Cox Regression Assumptions

```
library(survival)
model <- coxph(Surv(time, status) ~ x1 + x2 + x3, data = mydata)
```

```
# 1. PROPORTIONAL HAZARDS (most important!)
cox.zph(model)
# p < 0.05 suggests PH violation for that variable
# GLOBAL p-value tests overall PH assumption

plot(cox.zph(model))  # Schoenfeld residuals vs time
# Flat line = PH satisfied
# Trend = PH violated - consider:
#   - Stratification: coxph(... + strata(x3))
#   - Time-varying coefficient: tt() function
#   - Different model (e.g., accelerated failure time)

# 2. LINEARITY for continuous predictors
# Martingale residuals vs predictor
plot(mydata$x1, residuals(model, type = 'martingale'))
lines(lowess(mydata$x1, residuals(model, type = 'martingale')), col = 'red')
# Flat line = linear relationship OK

# 3. INFLUENTIAL OBSERVATIONS
# dfbeta residuals
dfbeta <- residuals(model, type = 'dfbeta')
plot(dfbeta[, 1])  # For first coefficient

# 4. OVERALL FIT
# Cox-Snell residuals should follow Exp(1) if model fits
cs_resid <- mydata$status - residuals(model, type = 'martingale')
surv_cs <- survfit(Surv(cs_resid, mydata$status) ~ 1)
plot(surv_cs, fun = 'cumhaz')  # Should be straight line through origin
```

4.14.4 Fine-Gray Assumptions

```
# Fine-Gray has similar assumptions to Cox:

# 1. PROPORTIONAL SUBDISTRIBUTION HAZARDS
# Test using Schoenfeld-type residuals
library(crrSC)
# Or visually compare CIF curves across groups

library(cmprsk)
```

```
cif <- cuminc(mydata$time, mydata$status, mydata$treatment)
plot(cif)  # Curves should not cross if PH holds

# 2. Check model specification
# Compare observed vs predicted CIF
library(riskRegression)
Score(list('Model' = fg_model),
      formula = Hist(time, status) ~ 1,
      data = mydata, times = c(1, 3, 5),
      plots = 'calibration')
```

4.15 Quick Reference Table

Concept	What It Measures	Key Metric	Ideal Value
Calibration	Predicted vs observed probabilities	Calibration slope	1.0
Discrimination	Separation of outcomes	AUC / C-statistic	> 0.7
Regularization	Complexity penalty	λ (lambda)	CV-selected
Multicollinearity	Predictor correlation	VIF	< 5
Optimism	Overfitting amount	Bootstrap optimism	< 0.05 AUC
Parsimony	Model simplicity	Number of parameters	Minimum adequate
Validation	Generalizability	Corrected AUC	Close to apparent

5. Step-by-Step Modeling Workflows

This section provides a complete checklist for each model type. Follow these steps in order.

QUICK REFERENCE: Universal Modeling Scheme (All Model Types)

STEP 1 - Data Preparation: Inspect data structure, check missing values, handle NA appropriately

STEP 2 - Exploratory Analysis: Examine outcome distribution, bivariate relationships, correlation matrix

STEP 3 - Fit Full Model: Include all clinically relevant candidate predictors

STEP 4 - Check Assumptions: Verify model-specific assumptions BEFORE variable selection

STEP 5 - Variable Selection: Use drop1() for Type III tests, then BIC-based backward elimination

STEP 6 - Validate Final Model: Re-check assumptions, influential observations, bootstrap validation

STEP 7 - Report Results: Coefficients with 95% CI, model performance metrics, calibration/discrimination

Model-Specific Key Functions

Linear: lm() | drop1(test='F') | step() | R^2 and Adjusted R^2

Logistic: glm(family=binomial) | drop1(test='Chisq') | stepAIC() | C-statistic/AUC

Cox: coxph() | drop1(test='Chisq') | cox.zph() for PH assumption | Harrell's C

Fine-Gray: crr() or FGR() | Gray's test for PH | Time-dependent AUC

5.1 Linear Regression: Complete Workflow

STEP 1: Data Preparation

```
# Load and inspect data
str(mydata)
summary(mydata)


# Check for missing values
colSums(is.na(mydata))


# Handle missing data (complete case or imputation)
mydata_complete <- na.omit(mydata)  # Simple approach
# Or use mice package for multiple imputation
```

STEP 2: Exploratory Analysis

```
# Outcome distribution
hist(mydata$y, main = 'Outcome Distribution')
# If skewed, consider transformation: log(y), sqrt(y)

# Bivariate relationships
pairs(mydata[, c('y', 'x1', 'x2', 'x3')])

# Correlation matrix
cor(mydata[, sapply(mydata, is.numeric)], use = 'complete.obs')
```

STEP 3: Fit Full Model

```
full_model <- lm(y ~ x1 + x2 + x3 + x4 + x5, data = mydata)
summary(full_model)
```

STEP 4: Check Assumptions BEFORE Variable Selection

```
# Diagnostic plots
par(mfrow = c(2, 2))
plot(full_model)

# Multicollinearity
library(car)
vif(full_model)  # All should be < 5

# If VIF > 10, remove one of the correlated variables FIRST
```

STEP 5: Variable Selection

```
# Check overall variable importance (Type III tests)
drop1(full_model, test = 'F')

# Automated BIC backward elimination
final_model <- step(full_model, direction = 'backward',
                    k = log(nrow(mydata)), trace = 1)
```

STEP 6: Validate Final Model Assumptions

```
par(mfrow = c(2, 2))
plot(final_model)

# Formal tests
library(lmtest)
bptest(final_model)  # Homoscedasticity
shapiro.test(residuals(final_model))  # Normality (if n < 5000)
```

```
# Influential observations
which(cooks.distance(final_model) > 4/nrow(mydata))
```

STEP 7: Report Results

```
summary(final_model)
confint(final_model)  # 95% CIs

# Effect sizes
# Report: coefficient (95% CI), p-value
# R-squared for overall model fit
```

5.2 Logistic Regression: Complete Workflow

STEP 1: Data Preparation

```
# Ensure outcome is 0/1 or factor
mydata$outcome <- as.numeric(mydata$outcome == 'Yes')  # If character
table(mydata$outcome)  # Check distribution

# Check event rate
mean(mydata$outcome)  # Ideally 0.1-0.9; extreme = separation risk
```

STEP 2: Check Events Per Variable (EPV)

```
n_events <- sum(mydata$outcome)
n_predictors <- 5  # Number of candidate predictors
epv <- n_events / n_predictors
cat('EPV:', epv, '\n')
# EPV >= 10 minimum, >= 20 preferred for selection
```

STEP 3: Exploratory Analysis

```
# Outcome by predictors
library(tableone)
CreateTableOne(vars = c('x1', 'x2', 'x3'),
               strata = 'outcome', data = mydata)
```

STEP 4: Fit Full Model

```
full_model <- glm(outcome ~ x1 + x2 + x3 + x4 + x5,
                  family = binomial, data = mydata)
summary(full_model)

# Check for separation warnings!
```

STEP 5: Check Assumptions

```
# Multicollinearity
library(car)
vif(full_model)

# Linearity in logit (for continuous predictors)
# Box-Tidwell test
boxTidwell(outcome ~ x1, other.x = ~ x2 + x3, data = mydata)
# If p < 0.05, consider transformation or spline
```

STEP 6: Variable Selection

```
# Type III tests for all variables
```

```
drop1(full_model, test = 'Chisq')


# BIC backward elimination
final_model <- step(full_model, direction = 'backward',
                    k = log(nrow(mydata)), trace = 1)
```

STEP 7: Assess Model Performance

```
# Discrimination (AUC)
library(pROC)
pred_prob <- predict(final_model, type = 'response')
roc_obj <- roc(mydata$outcome, pred_prob)
auc(roc_obj)
plot(roc_obj, print.auc = TRUE)


# Calibration
library(rms)
# Refit with lrm for calibration
dd <- datadist(mydata); options(datadist = 'dd')
model_rms <- lrm(outcome ~ x1 + x2 + x3, data = mydata, x = TRUE, y = TRUE)
cal <- calibrate(model_rms, B = 200)
plot(cal)


# Hosmer-Lemeshow (use cautiously)
library(ResourceSelection)
hoslem.test(mydata$outcome, pred_prob, g = 10)
```

STEP 8: Internal Validation

```
# Bootstrap validation for optimism-corrected performance
validate(model_rms, B = 200)
```

STEP 9: Report Results

```
# Odds Ratios with 95% CI
exp(cbind(OR = coef(final_model), confint(final_model)))


# Report: OR (95% CI), p-value
# Model discrimination: AUC = X.XX
# Calibration: slope = X.XX, Hosmer-Lemeshow p = X.XX
```

5.3 Cox Proportional Hazards: Complete Workflow

STEP 1: Data Preparation

```
library(survival)

# Check time and event variables
summary(mydata$time)  # Should be > 0
table(mydata$status)  # 0 = censored, 1 = event

# Event rate
n_events <- sum(mydata$status)
cat('Number of events:', n_events, '\n')
cat('Censoring rate:', 1 - mean(mydata$status), '\n')
```

STEP 2: Check Events Per Variable

```
n_events <- sum(mydata$status)
n_predictors <- 5
epv <- n_events / n_predictors
cat('EPV:', epv, '\n')  # Need >= 10
```

STEP 3: Kaplan-Meier Exploration

```
# Overall survival
km_fit <- survfit(Surv(time, status) ~ 1, data = mydata)
plot(km_fit, xlab = 'Time', ylab = 'Survival Probability')

# By key categorical predictors
km_by_group <- survfit(Surv(time, status) ~ treatment, data = mydata)
plot(km_by_group, col = 1:2, xlab = 'Time', ylab = 'Survival')
legend('bottomleft', legend = levels(mydata$treatment), col = 1:2, lty = 1)

# Log-rank test
survdiff(Surv(time, status) ~ treatment, data = mydata)
```

STEP 4: Fit Full Model

```
full_model <- coxph(Surv(time, status) ~ x1 + x2 + x3 + x4 + x5,
                    data = mydata)
summary(full_model)
```

STEP 5: Check Proportional Hazards Assumption (CRITICAL!)

```
# Schoenfeld residuals test
ph_test <- cox.zph(full_model)
```

```
print(ph_test)  # p < 0.05 = PH violation

# Visual inspection
plot(ph_test)  # Should be flat horizontal lines

# If PH violated for variable X:
# Option 1: Stratify
model_strat <- coxph(Surv(time, status) ~ x1 + x2 + strata(x_violating),
                     data = mydata)

# Option 2: Time-varying coefficient
model_tv <- coxph(Surv(time, status) ~ x1 + tt(x_violating),
                  data = mydata, tt = function(x, t, ...) x * log(t))
```

STEP 6: Check Other Assumptions

```
# Linearity for continuous predictors
# Martingale residuals vs predictor
resid_mart <- residuals(full_model, type = 'martingale')
plot(mydata$x1, resid_mart)
lines(lowess(mydata$x1, resid_mart), col = 'red')
# Should be approximately flat

# Multicollinearity
# Use VIF from linear model with same predictors as proxy
library(rms)
vif(ols(time ~ x1 + x2 + x3 + x4 + x5, data = mydata))
```

STEP 7: Variable Selection

```
# Type III tests
drop1(full_model, test = 'Chisq')

# BIC backward elimination
library(MASS)
final_model <- stepAIC(full_model, direction = 'backward',
                       k = log(nrow(mydata)), trace = 1)

# Re-check PH assumption on final model
cox.zph(final_model)
```

STEP 8: Assess Model Performance

```
# Concordance (C-statistic)
summary(final_model)$concordance

# Calibration (using rms)
library(rms)
dd <- datadist(mydata); options(datadist = 'dd')
model_cph <- cph(Surv(time, status) ~ x1 + x2 + x3,
                 data = mydata, x = TRUE, y = TRUE, surv = TRUE)
cal <- calibrate(model_cph, u = 365, B = 200)  # At 1 year
plot(cal)
```

STEP 9: Internal Validation

```
validate(model_cph, B = 200)
```

STEP 10: Report Results

```
# Hazard Ratios with 95% CI
summary(final_model)

# Or formatted:
exp(cbind(HR = coef(final_model), confint(final_model)))

# Report: HR (95% CI), p-value
# Concordance: C = X.XX
# PH assumption: Global test p = X.XX
```

5.4 Fine-Gray Competing Risks: Complete Workflow

IMPORTANT: BIC for Fine-Gray Models

The standard BIC() function does NOT work with tidycmprsk::crr() objects. You must calculate BIC manually using the formula: BIC = -2 × logLik + k × log(n), where k is the number of parameters and n is the number of events. The helper function below handles this automatically.

```
# CRITICAL: Pseudo-BIC for Fine-Gray (Wald-based variable selection)
# WARNING: cmprsk uses a PSEUDO log-likelihood, not a true likelihood.
# The pseudo-LRT is NOT asymptotically chi-square (see cmprsk docs).
# Use Wald tests for variable selection instead.
# car::Anova() does NOT support tidycmprsk::crr() objects.

# Helper: Wald joint test for multi-df terms (categorical variables)
wald_test_crr <- function(model, coef_indices) {
  # Joint Wald test for specified coefficient indices
  b <- coef(model)[coef_indices]
  V <- vcov(model)[coef_indices, coef_indices, drop = FALSE]
  W <- as.numeric(t(b) %*% solve(V) %*% b)  # Wald statistic
  df <- length(coef_indices)
  pchisq(W, df, lower.tail = FALSE)
}

# Helper: Pragmatic pseudo-BIC for Fine-Gray (use with caution)
# This is NOT true BIC - it's based on pseudo-likelihood
calc_pseudo_bic_crr <- function(model, n_events) {
  # For tidycmprsk::crr(), extract pseudo log-likelihood
  ll <- model$cmprsk$loglik  # Note: Object structure may vary by tidycmprsk
version
  k <- length(coef(model))
  -2 * ll + k * log(n_events)  # Pseudo-BIC approximation
}
```

STEP 1: Data Preparation

```
library(tidycmprsk)
library(cmprsk)

# Status coding:
# 0 = censored
# 1 = event of interest
```

```
# 2 = competing event (e.g., death from other cause)
table(mydata$status)

# Event counts
cat('Primary events:', sum(mydata$status == 1), '\n')
cat('Competing events:', sum(mydata$status == 2), '\n')
cat('Censored:', sum(mydata$status == 0), '\n')
```

STEP 2: Check Events Per Variable

```
n_primary_events <- sum(mydata$status == 1)
n_predictors <- 5
epv <- n_primary_events / n_predictors
cat('EPV for primary event:', epv, '\n')  # Need >= 10
```

STEP 3: Cumulative Incidence Exploration

```
# Overall cumulative incidence
library(cmprsk)
cif_overall <- cuminc(mydata$time, mydata$status)
plot(cif_overall, xlab = 'Time', ylab = 'Cumulative Incidence')

# By groups
cif_by_group <- cuminc(mydata$time, mydata$status, mydata$treatment)
plot(cif_by_group)

# Gray's test (competing risks equivalent of log-rank)
print(cif_by_group$Tests)
```

STEP 4: Fit Full Model

```
full_model <- crr(Surv(time, factor(status)) ~ x1 + x2 + x3 + x4 + x5,
                  data = mydata, failcode = 1)
summary(full_model)
```

STEP 5: Check Proportional Subdistribution Hazards

```
# Visual check: CIF curves by groups should not cross
# If they cross substantially, consider time-varying effects

# Compare CIF curves
cif_check <- cuminc(mydata$time, mydata$status, mydata$x_categorical)
plot(cif_check)

# More formal: compare early vs late HRs
```

```
# Split data at median follow-up and fit separate models
```

STEP 6: Variable Selection (Wald-Based - No Automated Function)

WARNING: car::Anova() does NOT support tidycmprsk::crr() objects. Use Wald tests from model summary or implement joint Wald tests manually for multi-df terms.

```
# Use model summary for individual Wald p-values
tidy(full_model)  # Wald p-values for each coefficient


# Manual backward elimination with Wald tests and pseudo-BIC:
# NOTE: This uses pseudo-BIC based on pseudo-likelihood - interpret
cautiously
finegray_backward_wald <- function(data, time_var, status_var,
                                   predictors, failcode = 1) {
  current_vars <- predictors
  n_events <- sum(data[[status_var]] == failcode)


  repeat {
    formula_str <- paste0('Surv(', time_var, ', factor(', status_var,
                          ')) ~ ', paste(current_vars, collapse = ' + '))
    model <- tidycmprsk::crr(as.formula(formula_str), data = data, failcode =
failcode)
    current_bic <- calc_pseudo_bic_crr(model, n_events)


    # Get Wald p-values from model summary (for continuous predictors)
    model_tidy <- broom::tidy(model)
    pvalues <- setNames(model_tidy$p.value, model_tidy$term)
    worst_var <- names(which.max(pvalues))


    test_vars <- setdiff(current_vars, worst_var)
    if (length(test_vars) == 0) break


    formula_test <- paste0('Surv(', time_var, ', factor(', status_var,
                           ')) ~ ', paste(test_vars, collapse = ' + '))
    test_model <- crr(as.formula(formula_test), data = data,
                      failcode = failcode)


    if (calc_pseudo_bic_crr(test_model, n_events) >= current_bic) break
    current_vars <- test_vars
```

```
    cat('Removed:', worst_var, '| BIC:',
round(calc_pseudo_bic_crr(test_model, n_events), 2), '\n')
  }

  # Return final model
  formula_final <- paste0('Surv(', time_var, ', factor(', status_var,
                          ')) ~ ', paste(current_vars, collapse = ' + '))
  crr(as.formula(formula_final), data = data, failcode = failcode)
}

final_model <- finegray_backward_wald(mydata, 'time', 'status',
                                      c('x1', 'x2', 'x3', 'x4', 'x5'))
```

STEP 7: Assess Model Performance

```
# C-statistic for competing risks
library(riskRegression)

score_result <- Score(
  list('Final Model' = final_model),
  formula = Hist(time, status) ~ 1,
  data = mydata,
  times = c(365, 730, 1095),  # 1, 2, 3 years
  metrics = c('auc', 'brier'),
  plots = 'calibration'
)
print(score_result)
```

STEP 8: Calibration

```
# Calibration plot
plotCalibration(score_result, times = 365)  # At 1 year

# Compare predicted vs observed CIF
# Use Score() with plots = 'calibration'
```

STEP 9: Report Results

```
# Subdistribution Hazard Ratios with 95% CI
summary(final_model)

# Report: sHR (95% CI), p-value
# Time-dependent AUC at key timepoints
# Note: Interpret as effect on cumulative incidence, not instantaneous rate
```

5.5 Universal Workflow Summary Table

Step	Linear	Logistic	Cox	Fine-Gray
1. Data prep	Check outcome distribution	Check outcome 0/1	Check time>0, status	Check status 0/1/2
2. EPV check	N/A (use n/p > 10)	Events/predictors ≥10	Events/predictors ≥10	Primary events/pred ≥10
3. Explore	Pairs plot, correlations	Table 1 by outcome	KM curves, log-rank	CIF curves, Gray test
4. Fit full	lm()	glm(family=binomial)	coxph()	crr()
5. Assumptions	Residual plots, VIF	VIF, linearity in logit	PH test (cox.zph)	Proportional sHR
6. Selection	drop1(test='F'), step()	drop1(test='Chisq'), step()	drop1(), stepAIC()	Anova(), manual BIC
7. Performance	R^2, residual SE	AUC, calibration	C-statistic, calibration	Time-dep AUC, calibration
8. Validation	Adj R^2, cross-validation	Bootstrap validate()	Bootstrap validate()	Bootstrap Score()
9. Report	β (95% CI), R^2	OR (95% CI), AUC	HR (95% CI), C-stat	sHR (95% CI), AUC(t)

6. CRITICAL PITFALL: The Categorical Variable Trap

This is the #1 mistake made by beginners.

6.1 The Problem

When R fits a model with a categorical variable (e.g., 4-level 'stage'), it creates 3 dummy coefficients.

summary() shows individual p-values per dummy: These test each level vs. reference - NOT the overall variable!

```
# MISLEADING output from summary():
            Estimate  Pr(>|z|)
stageII       0.08    0.90    <- Stage II vs I: not significant
stageIII      0.25    0.04    <- Stage III vs I: significant!
stageIV       0.19    0.06    <- Stage IV vs I: borderline

# WRONG conclusion: 'I should keep only stageIII'
# CORRECT: Test the WHOLE variable with drop1()
```

6.2 The Solution: drop1() for Joint Tests

```
drop1(model, test = 'Chisq')

# CORRECT output:
#         Df Deviance  LRT   Pr(>Chi)
# <none>     403.21
# sex     1  404.12  0.91   0.34    <- 1 df (2 levels)
# stage   3  405.89  2.68   0.44    <- 3 df (4 levels)

# Stage overall p = 0.44 > sex p = 0.34
# Remove STAGE first, not sex!
```

6.3 Code for Each Model Type

Linear: `drop1(model, test = 'F')`

Logistic/Cox: `drop1(model, test = 'Chisq')`

Fine-Gray: Use Wald tests from tidy(model); car::Anova() NOT supported for crr()

6.4 Common Mistakes to Avoid

MISTAKE 1: Removing one dummy level ('stageII is p=0.90, remove it')

Why wrong: Stage is a clinical construct. You cannot remove stage II patients from existence!

MISTAKE 2: Keeping only 'significant' levels

Why wrong: This is post-hoc data dredging. The result won't replicate.

MISTAKE 3: Using coefficient p-values for elimination

Why wrong: You end up with a mutilated variable with missing levels.

THE RULE: Categorical variables enter and leave the model AS A WHOLE.

Reference: *Greenland S. Avoiding power loss associated with categorization. Epidemiology. 1995;6(4):450-454.*

7. Model-Specific Implementation

7.1 Linear Regression

```
full_model <- lm(y ~ x1 + x2 + x3 + x4, data = mydata)

# Check assumptions
par(mfrow = c(2, 2))
plot(full_model)

# Variable importance (overall tests)
drop1(full_model, test = 'F')

# BIC-based backward elimination
final_model <- step(full_model, direction = 'backward', k =
log(nrow(mydata)))
```

7.2 Logistic Regression

```
full_model <- glm(y ~ x1 + x2 + x3 + x4, data = mydata, family = binomial)

# Variable importance
drop1(full_model, test = 'Chisq')

# BIC-based backward elimination
final_model <- step(full_model, direction = 'backward', k =
log(nrow(mydata)))
```

7.3 Cox Proportional Hazards

```
library(survival)
full_model <- coxph(Surv(time, status) ~ x1 + x2 + x3 + x4, data = mydata)

# Check PH assumption
cox.zph(full_model)

# Variable importance
drop1(full_model, test = 'Chisq')

# BIC-based backward elimination
library(MASS)
```

```
final_model <- stepAIC(full_model, direction = 'backward', k =
log(nrow(mydata)))
```

7.4 Fine-Gray Competing Risks

```
library(tidycmprsk)
library(car)


full_model <- crr(Surv(time, factor(status)) ~ x1 + x2 + x3 + x4,
                  data = mydata, failcode = 1)


# Overall variable tests (Wald-based - car::Anova NOT supported)
tidy(full_model)  # Use Wald p-values from tidy()


# Manual backward elimination required - see Code Appendix
```

8. Automatic Variable Selection Libraries in R

These packages provide automated backward, forward, and other selection methods:

8.1 Built-in Functions

step() - Base R

```
# Works with lm(), glm(), coxph()
# k = 2 for AIC, k = log(n) for BIC

# BIC backward elimination
final <- step(full_model, direction = 'backward', k = log(nrow(data)))

# AIC backward elimination
final <- step(full_model, direction = 'backward', k = 2)

# Forward selection
null_model <- glm(y ~ 1, data = data, family = binomial)
final <- step(null_model, scope = list(lower = ~1, upper = ~x1+x2+x3+x4),
              direction = 'forward', k = log(nrow(data)))

# Bidirectional (stepwise)
final <- step(full_model, direction = 'both', k = log(nrow(data)))
```

MASS::stepAIC() - More Robust

```
library(MASS)

# Works better with survival models
# Same syntax as step()
final <- stepAIC(full_model, direction = 'backward', k = log(nrow(data)))

# With trace to see steps
final <- stepAIC(full_model, direction = 'backward', k = log(nrow(data)),
trace = 1)
```

8.2 Specialized Packages

rms::fastbw() - Fast Backward Elimination

```
library(rms)
```

```
# Setup data distribution
dd <- datadist(mydata)
options(datadist = 'dd')

# Fit with rms
full_model <- lrm(y ~ x1 + x2 + x3 + x4, data = mydata)

# Fast backward elimination
# rule = 'aic' or 'p' (p-value based)
fastbw(full_model, rule = 'aic')

# For Cox models
full_cox <- cph(Surv(time, status) ~ x1 + x2 + x3 + x4, data = mydata)
fastbw(full_cox, rule = 'aic')
```

Reference: *Harrell FE. rms: Regression Modeling Strategies. R package.*

glmulti - Exhaustive Search

```
library(glmulti)

# Exhaustive search of all possible models
# WARNING: Slow with many predictors!
result <- glmulti(y ~ x1 + x2 + x3 + x4,
                  data = mydata,
                  level = 1,           # Main effects only
                  method = 'h',        # Exhaustive
                  crit = 'bic',        # Use BIC
                  family = binomial)

# Best model
summary(result@objects[[1]])

# For faster genetic algorithm search
result <- glmulti(y ~ x1 + x2 + x3 + x4,
                  data = mydata,
                  level = 1,
                  method = 'g',        # Genetic algorithm
                  crit = 'bic',
                  family = binomial)
```

Reference: *Calcagno V, de Mazancourt C. glmulti: An R Package for Easy Automated Model Selection. J Stat Softw. 2010;34(12):1-29.*

bestglm - Best Subset Selection

```
library(bestglm)

# Requires specific data format: predictors first, outcome last
Xy <- mydata[, c('x1', 'x2', 'x3', 'x4', 'y')]

# Best subset with BIC
result <- bestglm(Xy, IC = 'BIC', family = binomial)

# Best model
result$BestModel

# All models ranked
result$Subsets
```

Reference: *McLeod AI, Xu C. bestglm: Best Subset GLM. R package.*

MuMIn - Multi-Model Inference

```
library(MuMIn)

# Fit full model with na.action = na.fail
options(na.action = 'na.fail')
full_model <- glm(y ~ x1 + x2 + x3 + x4, data = mydata, family = binomial)

# Generate all subsets and rank by IC
all_models <- dredge(full_model, rank = 'BIC')

# View top models
head(all_models)

# Get best model
best <- get.models(all_models, 1)[[1]]
summary(best)

# Model averaging (for prediction)
avg_model <- model.avg(all_models, subset = delta < 2)
```

Reference: *Barton K. MuMIn: Multi-Model Inference. R package.*

glmnet - LASSO/Elastic Net

```
library(glmnet)

# Prepare matrix input
x <- model.matrix(y ~ x1 + x2 + x3 + x4, data = mydata)[, -1]
y <- mydata$y

# LASSO with cross-validation
cv_fit <- cv.glmnet(x, y, family = 'binomial', alpha = 1)

# Coefficients at optimal lambda
coef(cv_fit, s = 'lambda.min')

# More parsimonious (1 SE rule)
coef(cv_fit, s = 'lambda.1se')

# For survival
library(survival)
cv_fit <- cv.glmnet(x, Surv(time, status), family = 'cox', alpha = 1)
```

Reference: *Friedman J, Hastie T, Tibshirani R. Regularization Paths for Generalized Linear Models via Coordinate Descent. J Stat Softw. 2010;33(1):1-22.*

8.3 Summary Table: Automatic Selection Libraries

Package	Function	Methods	Model Types	Best For
base R	step()	Forward/Backward/Both	lm, glm, coxph	General use
MASS	stepAIC()	Forward/Backward/Both	All + survival	Survival models
rms	fastbw()	Backward	lrm, cph, ols	Speed, validation
glmulti	glmulti()	Exhaustive/Genetic	glm	Small p, all subsets
bestglm	bestglm()	Best subset	glm	Small p, exhaustive
MuMIn	dredge()	All subsets	lm, glm, etc.	Model averaging

glmnet	cv.glmnet()	LASSO/Ridge/ Elastic	All + survival	High-dimensional

9. Method Comparison Cheat Sheet

9.1 Selection Criteria

Criterion	Formula	Penalty/param	Best For	R Code
AIC	-2logL + 2k	2	Prediction	step(..., k=2)
BIC	-2logL + k*log(n)	log(n)	True model ID	step(..., k=log(n))
$p < 0.05$	Wald/LRT	~3.84	Traditional	manual with drop1()
$p < 0.157$	Wald/LRT	= 2 (≈AIC)	AIC-equivalent	threshold at 0.157

9.2 When to Use AIC vs BIC

Use AIC: Goal is prediction; want to capture all relevant predictors; moderate n.

Use BIC: Goal is identifying true model; want parsimony; large n; Bayesian perspective.

Reference: *Burnham KP, Anderson DR. Model Selection and Multimodel Inference. 2nd ed. Springer; 2002.*

10. Defending Your Choices to Reviewers

10.1 Common Attacks and Rebuttals

ATTACK: "Stepwise selection is invalid."

REBUTTAL: "We used backward elimination with BIC, not p-value thresholds. This approach is recommended by Heinze et al. (2018) for exploratory analyses. We report sensitivity analyses showing model stability."

ATTACK: "Why BIC not AIC?"

REBUTTAL: "BIC was chosen because our goal was identifying clinically important predictors rather than optimizing prediction. BIC provides stronger penalty for complexity, favoring parsimonious models (Burnham & Anderson, 2002)."

ATTACK: "You should have used LASSO."

REBUTTAL: "While LASSO can be useful for variable selection, our goal was interpretable effect estimates. LASSO produces shrunk coefficients that are not directly interpretable as unpenalized effect sizes, and require additional inference procedures for valid confidence intervals."

ATTACK: "Standard errors are invalid after selection."

REBUTTAL: "We acknowledge post-selection inference limitations (Berk et al., 2013). We addressed this via: (1) pre-specified candidates, (2) bootstrap validation, (3) interpreting results as exploratory."

10.2 Essential References

Methodology: *Heinze et al. Biom J. 2018;60(3):431-449.*

Information criteria: *Burnham & Anderson. Model Selection and Multimodel Inference. Springer; 2002.*

Post-selection: *Berk et al. Ann Stat. 2013;41(2):802-837.*

11. Template Methods Paragraphs

Copy and modify for your papers.

11.1 Backward Elimination with BIC

"Candidate predictors were selected based on clinical relevance and prior literature. Multivariable [logistic/Cox] regression was performed using backward elimination with the Bayesian Information Criterion (BIC) as the stopping rule. At each step, the variable whose removal resulted in the largest decrease in BIC was eliminated. Elimination continued until no removal decreased BIC. This approach is recommended for identifying parsimonious models (Heinze et al., 2018; Burnham & Anderson, 2002). Categorical variables were evaluated using joint tests (drop1 function) rather than individual coefficient tests. All analyses were performed in R version [X.X.X]."

11.2 Pre-specified Model

"Candidate predictors were pre-specified based on clinical relevance, biological plausibility, and prior literature, without reference to the current dataset. Variables included [list]. No data-driven selection was performed to preserve Type I error rates (Harrell, 2015). All pre-specified variables were retained regardless of statistical significance."

11.3 LASSO Selection

"Given the high-dimensional nature of our data, we used LASSO regression for variable selection (Tibshirani, 1996). The regularization parameter lambda was selected via 10-fold cross-validation. Variables with non-zero coefficients at the optimal lambda were selected. To obtain unpenalized, interpretable coefficients, we then fit standard [logistic/Cox] regression with LASSO-selected variables, acknowledging post-selection inference limitations."

12. Complete Code Appendix

12.1 Universal BIC Backward Elimination

```
backward_bic <- function(model) {
  step(model, direction = 'backward', k = log(nobs(model)), trace = 1)
}
```

12.2 Correct Variable Testing

```
get_variable_pvalues <- function(model) {
  if (inherits(model, 'lm') && !inherits(model, 'glm')) {
    drop1(model, test = 'F')
  } else {
    drop1(model, test = 'Chisq')
  }
}
```

12.3 Bootstrap Stability Check

```
bootstrap_stability <- function(data, formula, B = 200) {
  vars <- all.vars(formula)[-1]
  counts <- setNames(rep(0, length(vars)), vars)

  for (i in 1:B) {
    boot_data <- data[sample(nrow(data), replace = TRUE), ]
    full <- glm(formula, data = boot_data, family = binomial)
    final <- step(full, k = log(nrow(boot_data)), trace = 0)
    selected <- all.vars(formula(final))[-1]
    counts[selected] <- counts[selected] + 1
  }
  sort(counts / B * 100, decreasing = TRUE)
}
```

12.4 Fine-Gray Manual Backward Elimination

Note: Standard BIC() does not work with tidycmprsk::crr(). Uses PSEUDO-likelihood (not true BIC). car::Anova() NOT supported - use Wald tests.

```
library(tidycmprsk); library(broom)

# Helper: Pseudo-BIC for Fine-Gray (WARNING: based on pseudo-likelihood)
calc_pseudo_bic_crr <- function(model, n_events) {
```

```
  ll <- model$cmprsk$loglik  # Note: Object structure may vary by tidycmprsk
version
  k <- length(coef(model))
  -2 * ll + k * log(n_events)
}


# Wald-based backward elimination (uses pseudo-BIC)
finegray_backward_wald <- function(data, time_var, status_var, predictors,
failcode = 1) {
  current_vars <- predictors
  n_events <- sum(data[[status_var]] == failcode)


  repeat {
    formula_str <- paste0('Surv(', time_var, ', factor(', status_var, ')) ~
',
                          paste(current_vars, collapse = ' + '))
    model <- crr(as.formula(formula_str), data = data, failcode = failcode)
    current_bic <- calc_pseudo_bic_crr(model, n_events)


    # Use Wald p-values from model summary (car::Anova not supported)
    model_tidy <- broom::tidy(model)
    pvalues <- setNames(model_tidy$p.value, model_tidy$term)
    worst_var <- names(which.max(pvalues))


    test_vars <- setdiff(current_vars, worst_var)
    if (length(test_vars) == 0) break


    formula_test <- paste0('Surv(', time_var, ', factor(', status_var, ')) ~
',
                           paste(test_vars, collapse = ' + '))
    test_model <- crr(as.formula(formula_test), data = data, failcode =
failcode)


    if (calc_pseudo_bic_crr(test_model, n_events) >= current_bic) break
    current_vars <- test_vars
    cat('Removed:', worst_var, '| BIC:',
round(calc_pseudo_bic_crr(test_model, n_events), 2), '\n')
  }


  formula_final <- paste0('Surv(', time_var, ', factor(', status_var, ')) ~
',
```

```
                          paste(current_vars, collapse = ' + '))
  crr(as.formula(formula_final), data = data, failcode = failcode)
}
```

References

Akaike H. A new look at the statistical model identification. IEEE Trans Automat Contr. 1974;19(6):716-723.

Altman DG, Andersen PK. Bootstrap investigation of the stability of a Cox regression model. Stat Med. 1989;8(7):771-783.

Austin PC, Fine JP. Practical recommendations for reporting Fine-Gray model analyses. Stat Med. 2017;36(27):4391-4400.

Austin PC, Lee DS, Fine JP. Introduction to the Analysis of Survival Data in the Presence of Competing Risks. Circulation. 2016;133(6):601-609.

Barton K. MuMIn: Multi-Model Inference. R package version 1.47.5. 2023.

Berk R, Brown L, Buja A, Zhang K, Zhao L. Valid post-selection inference. Ann Stat. 2013;41(2):802-837.

Burnham KP, Anderson DR. Model Selection and Multimodel Inference. 2nd ed. Springer; 2002.

Calcagno V, de Mazancourt C. glmulti: An R Package for Easy Automated Model Selection. J Stat Softw. 2010;34(12):1-29.

Derksen S, Keselman HJ. The relative power of the ordinary least squares and James-Stein procedures. Educ Psychol Meas. 1992;52:583-591.

Engle RF. Wald, likelihood ratio, and Lagrange multiplier tests in econometrics. Handbook of Econometrics. 1984;2:775-826.

Fine JP, Gray RJ. A proportional hazards model for the subdistribution of a competing risk. J Am Stat Assoc. 1999;94(446):496-509.

Friedman J, Hastie T, Tibshirani R. Regularization Paths for Generalized Linear Models. J Stat Softw. 2010;33(1):1-22.

Greenland S. Avoiding power loss associated with categorization. Epidemiology. 1995;6(4):450-454.

Harrell FE. Regression Modeling Strategies. 2nd ed. Springer; 2015.

Heinze G, Wallisch C, Dunkler D. Variable selection - A review and recommendations. Biom J. 2018;60(3):431-449.

Hosmer DW, Lemeshow S, Sturdivant RX. Applied Logistic Regression. 3rd ed. Wiley; 2013.

Hurvich CM, Tsai CL. The impact of model selection on inference. Am Stat. 1990;44(3):214-217.

McLeod AI, Xu C. bestglm: Best Subset GLM. R package version 0.37.3. 2020.

Miller AJ. Selection of subsets of regression variables. J R Stat Soc Ser A. 1984;147(3):389-425.

Peduzzi P, et al. A simulation study of the number of events per variable. J Clin Epidemiol. 1996;49(12):1373-1379.

Riley RD, Ensor J, Snell KIE, et al. Calculating the sample size required for developing a clinical prediction model. BMJ. 2020;368:m441.

Schwarz G. Estimating the dimension of a model. Ann Stat. 1978;6(2):461-464.

Shmueli G. To explain or to predict? Stat Sci. 2010;25(3):289-310.

Steyerberg EW. Clinical Prediction Models. 2nd ed. Springer; 2019.

Therneau TM, Grambsch PM. Modeling Survival Data: Extending the Cox Model. Springer; 2000.

Tibshirani R. Regression shrinkage and selection via the lasso. J R Stat Soc B. 1996;58(1):267-288.

Vittinghoff E, et al. Regression Methods in Biostatistics. 2nd ed. Springer; 2012.

Vittinghoff E, McCulloch CE. Relaxing the rule of ten events per variable. Am J Epidemiol. 2007;165(6):710-718.