Advanced Computational Statistics & Risk Modelling

The Master Exam Protocol

Candidate Reference

Contents

1 Module 1: Aggregate Loss Modelling (Simulation)

1.1 Objective

Simulate the aggregate loss $S = \sum_{i=1}^{N} X_i$ using a Compound Poisson-Gamma model. This approach is required to estimate tail risk metrics (e.g., 99.5th percentile) which analytically intractable methods cannot easily provide.

1.2 The "Easiest Efficient" Code Pattern

Key Efficiency: We use `numeric(M)` for pre-allocation. This avoids growing vectors inside a loop, reducing time complexity from $O(n^2)$ to $O(n)$.

```r
# ---------------------------------------------------------------------------
# Module 1: Efficient Monte Carlo Simulation (Base R)
# ---------------------------------------------------------------------------
set.seed(2025)

# 1. Parameters
M <- 100000            # Number of simulations
lambda_freq <- 50      # Poisson parameter (Expected Frequency)
alpha_sev <- 2         # Gamma shape
beta_sev  <- 0.1       # Gamma rate (Mean = alpha/beta = 20)

# 2. Pre-allocation (Crucial for speed)
S <- numeric(M)        # Creates a vector of 100,000 zeros

# 3. Simulation Loop
# Simulate N (freq) once per trial, then simulate that many X's (severity).
for(i in 1:M) {
  n_claims <- rpois(1, lambda = lambda_freq) # Step 1: How many claims?

  if(n_claims > 0) {
    # Step 2: Sum of 'n_claims' random severity amounts
    S[i] <- sum(rgamma(n_claims, shape = alpha_sev, rate = beta_sev))
  } else {
    S[i] <- 0
  }
}

# 4. Statistics
# 99.5th Percentile (VaR) - Solvency II Capital Requirement
var_995 <- quantile(S, probs = 0.995)

# Moments
mean_S <- mean(S)
median_S <- median(S)
# Skewness: E[(x-mu)^3] / sigma^3
skewness_S <- mean((S - mean_S)^3) / (sd(S)^3)

# 5. Output
results <- c(VaR_99.5 = var_995, Median = median_S, Skewness = skewness_S)
print(results)

# 6. Visualization
hist(S, breaks=50, main="Aggregate Loss Distribution", col="lightblue")
abline(v=var_995, col="red", lwd=2)
```

1.3 Interpretation Key

- **Skewness > 0:** The tail extends to the right. This invalidates the Normal approximation; you *must* use simulation or Gamma models to capture tail risk.
- **Mean $>$ Median:** Confirms right-skew, typical for insurance claims.

- **VaR (99.5%):** Represents the capital required to survive a 1-in-200 year event. If this value is high relative to the mean, the portfolio is volatile.

2 Module 2: The “Menu” of Diagnostics (Theory & Critique)

2.1 Error Metrics: MSE, RMSE, and MAE

Use this table to answer questions asking you to critique a student’s choice of error metric.

Metric	Theory & Interpretation	Critique Strategy (If This $\rightarrow$ Then That)
MSE	**Target:** Conditional Mean (E). **Pros:** Penalizes large errors heavily (squaring). Good for solvency focus.	**If High:** Model fails on outliers. Suggest log-transforming target. **If Low:** Good fit, but check for overfitting on small claims.
RMSE	**Target:** Mean. **Pros:** Same as MSE but in original units (€). Easier to explain to business stakeholders.	**If RMSE $\gg$ MAE:** Confirms data is highly skewed (heavy tail). Justifies Gamma GLM over Normal OLS.
MAE	**Target:** Conditional **Median**. **Pros:** Robust to outliers. **Cons:** Ignores tail risk.	**If used for Pricing: CRITICAL FAIL.** “Minimizing MAE targets the median. In insurance, Mean > Median. This model will systematically underprice risk and lead to insolvency.”

2.2 Model Selection: AIC vs. BIC

Metric	Penalty	Philosophy	Critique Strategy
AIC	$+2k$	**Prediction.** Approximates CV error. Tends to keep variables.	**Use When:** Goal is predictive accuracy for next year’s premium. **Risk:** May overfit (include weak variables) in large datasets.
BIC	$+k\ln(n)$	**Truth.** Penalizes complexity heavily if n is large.	**Use When:** Goal is identifying key risk drivers (explanation). **Risk:** Will underfit in insurance (rejects small but real risks).

2.3 GLM Diagnostics: Deviance & Dispersion

1. **Deviance (D):**
   - **Null Deviance:** Fit of model with *only* intercept (worst baseline).
   - **Residual Deviance:** Fit of *your* proposed model.
   - **Interpretation:** We want Residual Dev $\ll$ Null Dev.
2. **Overdispersion (ϕ):**
   - **Check:** Calculate $\phi = \frac{\text{Residual Deviance}}{\text{Degrees of Freedom}}$.
   - **If $\phi \approx 1$:** Poisson assumption holds (Mean $\approx$ Variance).
   - **If $\phi \gg 1$: Overdispersion** (Variance > Mean).
   - **Fix:** Critique use of Poisson. Recommend **Quasi-Poisson** or **Negative Binomial**.

3 Module 3: GLM Age Analysis (Master Script)

3.1 Objective

Compare Age as a **Covariate** (Linear), **Factor** (Bands), and **Spline**. Generate a clean overlay plot of the empirical means and the model fits.

```r
# ----------------------------------------------------------------------------
# Module 3: Comprehensive Gamma GLM Age Analysis
# ----------------------------------------------------------------------------
library(splines) # Required for ns()

# --- STEP 0: SETUP MOCK DATA (SKIP IN EXAM if 'dat' exists) ---
set.seed(42)
n <- 2000
age <- sample(18:90, n, replace = TRUE)
# True process: Convex U-shape + Noise
mu_true <- exp(6 + 0.04 * ((age - 55)^2) / 50)
gross_amount <- rgamma(n, shape = 1, scale = mu_true)
dat <- data.frame(age, gross_amount)
# -------------------------------------------------------------

# --- STEP 1: PRE-PROCESSING & EMPIRICAL ANALYSIS ---

# A. Create 10-Year Age Bands (Factor Approach)
dat$age_band <- cut(dat$age, breaks = seq(10, 100, by = 10))

# B. Calculate Empirical Mean Severity at each Age
# This aggregates the raw data to find the average cost per age
emp_means <- aggregate(gross_amount ~ age, data = dat, FUN = mean)

# --- PLOT 1: Empirical Mean Severity Only ---
plot(emp_means$age, emp_means$gross_amount,
     type = "b",        # 'b' for both points and lines
     pch = 19,          # Solid circle points
     col = "black",
     lwd = 2,
     main = "Empirical Mean Severity by Age",
     xlab = "Age", ylab = "Average Severity (EUR)")
grid()

# --- STEP 2: FIT THREE GLM CANDIDATES ---

# Model 1: Age as Continuous Covariate (Linear)
mod_linear <- glm(gross_amount ~ age, family = Gamma(link = "log"), data = dat)

# Model 2: Age as Factor (10-Year Bands)
mod_band <- glm(gross_amount ~ age_band, family = Gamma(link = "log"), data = dat)

# Model 3: Age as Natural Spline (df=4 allows flexibility for U-shape)
mod_spline <- glm(gross_amount ~ ns(age, df = 4), family = Gamma(link = "log"), data
    = dat)

# --- STEP 3: PREDICT & COMPARE ---

# Create a prediction grid (ages 18 to 90)
pred_data <- data.frame(age = 18:90)
# Map ages to bands for the band model
pred_data$age_band <- cut(pred_data$age, breaks = seq(10, 100, by = 10))

# Generate predictions (on the response scale, i.e., in Euros)
pred_data$fit_linear <- predict(mod_linear, newdata = pred_data, type = "response")
pred_data$fit_band   <- predict(mod_band,   newdata = pred_data, type = "response")
pred_data$fit_spline <- predict(mod_spline, newdata = pred_data, type = "response")

# --- PLOT 2: Model Comparison Overlay ---
```

```r
# 1. Plot the Empirical Means again (as the "truth" baseline)
plot(emp_means$age, emp_means$gross_amount,
     pch = 16, col = "grey60", cex = 0.8,
     main = "Comparing GLM Fits: Linear vs Banded vs Spline",
     xlab = "Age", ylab = "Expected Severity (EUR)",
     ylim = c(0, max(emp_means$gross_amount) * 1.1))

# 2. Overlay Models
lines(pred_data$age, pred_data$fit_linear, col = "red", lwd = 2, lty = 2) # Linear
lines(pred_data$age, pred_data$fit_band, col = "blue", lwd = 2)           # Banded
lines(pred_data$age, pred_data$fit_spline, col = "darkgreen", lwd = 3)    # Spline

# 3. Add Legend
legend("topright",
       legend = c("Empirical Data", "Linear", "Banded", "Spline"),
       col = c("grey60", "red", "blue", "darkgreen"),
       lty = c(NA, 2, 1, 1), pch = c(16, NA, NA, NA), lwd = 2)

# --- STEP 4: STATISTICAL CRITIQUE ---
print(AIC(mod_linear, mod_band, mod_spline))
anova(mod_linear, mod_spline, test = "Chisq")
```

3.2 Interpretation Key

- **Linear (Red Line):** Smooth monotonic line. **Critique:** "Too rigid. Fails to capture U-shape (risk for young/old)."
- **Banded (Blue Steps):** Staircase. **Critique:** "Creates artificial 'Cliff-Edges' (price drops instantly at age 30). Wastes degrees of freedom."
- **Spline (Green Curve):** Smooth, flexible. **Critique:** "Gold Standard. Captures non-linearity smoothly without artificial jumps."

4 Module 4: Machine Learning Master Class

4.1 Objective

Implement `rpart` Decision Trees, perform **Manual Cross-Validation** to demonstrate stability, and conduct Unsupervised Learning (PCA, K-Means).

```r
# ---------------------------------------------------------------------------
# Module 4: Master Script - Trees, Manual CV, PCA, Clustering
# ---------------------------------------------------------------------------
library(rpart)
library(rpart.plot)
library(cluster)

# Load Data (Using Iris as placeholder)
data(iris)
dat <- iris

# ===========================================================================
# PART 1: DECISION TREE WITH MANUAL CROSS-VALIDATION
# ===========================================================================

# A. Fit the Single Tree
tree_model <- rpart(Species ~., data = dat, method = "class", cp = 0.01)
rpart.plot(tree_model, main = "Classification Tree")

# B. Confusion Matrix (On Training Data)
preds <- predict(tree_model, type = "class")
conf_matrix <- table(Predicted = preds, Actual = dat$Species)
print(conf_matrix)
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
print(paste("Training Accuracy:", round(accuracy, 4)))

# C. MANUAL 10-FOLD CROSS-VALIDATION (The "Stability" Test)
set.seed(123)
K <- 10
folds <- sample(rep(1:K, length.out = nrow(dat)))
cv_accuracies <- numeric(K)

for(k in 1:K) {
  # a. Split Data
  test_idx <- which(folds == k)
  train_data <- dat[-test_idx, ]
  test_data  <- dat[test_idx, ]

  # b. Train Model
  model_k <- rpart(Species ~., data = train_data, method = "class", cp = 0.01)

  # c. Predict & Evaluate
  pred_k <- predict(model_k, newdata = test_data, type = "class")
  cv_accuracies[k] <- mean(pred_k == test_data$Species)
}

# Visualization of Stability
barplot(cv_accuracies, ylim = c(0, 1),
        main = "Stability Check: Accuracy per Fold",
        xlab = "Fold Number", ylab = "Accuracy", col = "lightblue")
abline(h = mean(cv_accuracies), col = "red", lwd = 2)

# ===========================================================================
# PART 2: UNSUPERVISED LEARNING (PCA & K-MEANS)
# ===========================================================================

X <- dat[, -5] # Remove target

# A. PCA
```

```r
# scale. = TRUE is CRITICAL.
pca_res <- prcomp(X, scale. = TRUE)

# 1. Scree Plot (Find the Elbow)
screeplot(pca_res, type = "lines", main = "PCA Scree Plot")
abline(h = 1, col = "red", lty = 2) # Kaiser's Rule

# 2. Biplot (Arrows and Angles)
biplot(pca_res, cex = 0.7, main = "PCA Biplot")

# B. K-Means Clustering
set.seed(42)
wss <- numeric(10)
for (k in 1:10) {
  wss[k] <- kmeans(scale(X), centers = k, nstart = 20)$tot.withinss
}

# Plot WSS (Look for the "kink")
plot(1:10, wss, type = "b", pch = 19,
     xlab = "K", ylab = "Total Within-Cluster SS", main = "Elbow Plot")

# Fit Final Cluster Model
final_km <- kmeans(scale(X), centers = 3, nstart = 20)

# Silhouette Plot
sil <- silhouette(final_km$cluster, dist(scale(X)))
plot(sil, main = "Silhouette Plot (k=3)")
```

4.2 Interpretation Key

- **Silhouette Plot:**
  - **Close to +1:** Ideal (point is in correct cluster).
  - **Near 0:** Borderline.
  - **Negative:** Wrong cluster.
- **PCA Biplot:**
  - **Arrows:** Length = Variance/Importance. Direction = Correlation.
  - **Angles:** Acute ($< 90°$) = Pos. Correlation. Obtuse ($> 90°$) = Neg. Correlation.
- **CV Stability:**
  - **Stable:** Bars roughly equal height.
  - **Unstable:** Bars jump wildly. Decision Trees are high-variance; consider Random Forest.

5 Module 5: Optimization & Regularization

5.1 Objective

Use `optim()` for custom Maximum Likelihood Estimation and `glmnet` for Lasso/Ridge regression.

```r
# ----------------------------------------------------------------------------
# Module 5: Optimization & Regularization
# ----------------------------------------------------------------------------

# A. Optimization (optim) - Custom MLE for Gamma
# Define Negative Log-Likelihood
nll_gamma <- function(pars, x) {
  alpha <- pars[1] # Shape
  beta  <- pars[2] # Rate
  if(alpha <= 0 |

| beta <= 0) return(Inf) # Constraints
  -sum(dgamma(x, shape = alpha, rate = beta, log = TRUE))
}

# Optimize
obs_data <- rgamma(100, 2, 0.5)
opt <- optim(par = c(1, 1), fn = nll_gamma, x = obs_data, hessian = TRUE)

# Standard Errors (Inverse Hessian)
fisher_info <- solve(opt$hessian)
se <- sqrt(diag(fisher_info))
print(opt$par)
print(se)

# B. Lasso Regression (glmnet)
library(glmnet)

# Setup Data (Matrix format required)
X_mat <- model.matrix(Species ~. - 1, data = iris)
y_vec <- as.numeric(iris$Species)

# Fit Lasso (alpha = 1)
# cv.glmnet does k-fold Cross Validation automatically
cv_fit <- cv.glmnet(X_mat, y_vec, alpha = 1)

# Plot Coefficient Path
plot(cv_fit)

# Get Lambda that minimizes error
best_lambda <- cv_fit$lambda.min
coef(cv_fit, s = "lambda.min")
```

5.2 Takeaway

- **Lasso ($\alpha = 1$):** Shrinks coefficients to exactly **zero**. Performs feature selection.
- **Ridge ($\alpha = 0$):** Shrinks coefficients **towards** zero but keeps them all. Handles multicollinearity.