DRC via HR para atendimento ao FRTB

Implementação via Regressão Heurística (HR)

Walter C Neto

2 de janeiro de 2026

1 Introdução

O *Default Risk Charge* (DRC) é o componente do capital regulatório introduzido pelo *Fundamental Review of the Trading Book* (FRTB) para capturar perdas associadas a eventos de *default* em instrumentos de crédito mantidos no *Trading Book*.

Regulatoriamente, o DRC é definido como o *Value at Risk* (VaR) ao nível de 99,9% em horizonte de um ano, considerando exclusivamente eventos de *default*. A estimação direta desse quantil extremo por métodos tradicionais de Monte Carlo apresenta desafios relevantes: custo computacional elevado, instabilidade estatística e baixa explicabilidade para fins de validação e supervisão.

Este trabalho apresenta uma aplicação prática do modelo de *Heuristic Regression* (HR) para estimação do DRC, baseada em:

- cálculo exato do DRC em carteiras de pequena dimensão;
- extração de características estruturais da carteira;
- extrapolação determinística para carteiras reais de grande porte.

2 Descrição da Carteira de Trading Book

Inicialmente, foi construída uma carteira hipotética de *Trading Book*, com 200 emissores, contendo instrumentos típicos do FRTB:

- títulos soberanos;
- títulos corporativos;
- instrumentos financeiros do setor bancário;
- posições em *bond* e *equity*.

Para cada emissor j, a carteira contém:

- exposição ao *default*;
- *Loss Given Default* (LGD);
- perda em caso de default L_j;
- probabilidade anual de default PD_j.

Essa base representa o ponto de partida típico para a aplicação do DRC sob o FRTB.

3 Formulação Matemática do DRC

Considere um portfólio com J emissores. Define-se um cenário de default como:

$$\omega = (\omega_1, \ldots, \omega_J) \in \{0,1\}^J,$$

onde $\omega_j = 1$ indica default do emissor j.

A perda do portfólio no cenário ω é:

$$\mathrm{Loss}(\omega) = \sum_{j=1}^{J} \omega_j L_j. \tag{1}$$

Assumindo independência entre emissores, a probabilidade do cenário é:

$$\mathbb{P}(\omega) = \prod_{j=1}^{J} \mathrm{PD}_j^{\omega_j} (1 - \mathrm{PD}_j)^{1-\omega_j}. \tag{2}$$

O DRC regulatório é então:

$$\mathrm{DRC}(99{,}9\%) = \inf\{x \in \mathbb{R}_+ : \mathbb{P}(\mathrm{Loss} \leq x) \geq 99{,}9\%\}. \tag{3}$$

4 Motivação do Modelo Heurístico

O cálculo exato do DRC exige a enumeração de 2^J cenários. Para carteiras reais, com centenas ou milhares de emissores, essa abordagem torna-se inviável.

A ideia central do modelo HR é:

- calcular o DRC exato apenas em carteiras pequenas;
- identificar como a estrutura da carteira afeta o DRC relativo;
- extrapolar essa relação para carteiras grandes.

5 Índices de Concentração

Ordenando as perdas de forma crescente:

$$L^{\uparrow} = (L_{(1)}, \ldots, L_{(J)}),$$

e reordenando as PDs de forma consistente, definem-se os índices:

$$Q_1(p) = \frac{\sum_{i=1}^{\lfloor pJ \rfloor} L_{(i)}}{\sum_{j=1}^{J} L_j}, \tag{4}$$

$$Q_2(p) = \frac{\sum_{i=1}^{\lfloor pJ \rfloor} \mathrm{PD}_{(i)}}{\sum_{j=1}^{J} \mathrm{PD}_j}. \tag{5}$$

Neste estudo, utilizam-se:

$$Q_1(90\%) \quad \text{e} \quad Q_2(75\%),$$

consistentes com evidências empíricas de forte concentração de risco.

6 Modelo Heurístico por Regressão Logística

Define-se o DRC normalizado:

$$y = \frac{\mathrm{DRC}(99{,}9\%)}{\sum_{j=1}^{J} L_j}. \tag{6}$$

O modelo HR assume:

$$\log\left(\frac{y}{1-y}\right) = \beta_0 + \beta_1 Q_1(90\%) + \beta_2 Q_2(75\%) + \varepsilon. \tag{7}$$

Essa forma garante que $y \in (0, 1)$, respeitando limites naturais do problema.

7 Calibração do Modelo

Foram simuladas 50 carteiras com $J = 12$, para as quais o DRC foi calculado de forma exata. Após remoção de outliers, os parâmetros estimados foram:

Tabela 1: Parâmetros estimados do modelo HR

Parâmetro	Valor
β_0	-8.404204
β_1	-2.855728
β_2	8.789292

8 Aplicação à Carteira de Trading Book

Para a carteira com 200 emissores, obteve-se:

Tabela 2: Resultados do DRC via HR

Métrica	Valor
Número de emissores	200
$Q_1(90\%)$	0.591403
$Q_2(75\%)$	0.940473
DRC normalizado y^*	0.138596
DRC(99,9%) total	447.197712

9 Interpretação Regulatória

O resultado indica que o capital de DRC corresponde a aproximadamente 14% da perda máxima teórica do portfólio. Esse valor é consistente com:

- forte concentração de perdas em poucos emissores grandes;
- ampla dispersão de emissores pequenos com PDs não desprezíveis.

O modelo produz uma estimativa estável, explicável e alinhada ao racional prudencial do FRTB.

10 Limitações e Trabalhos Futuros

- inclusão de dependência entre emissores via cópulas fatoriais;
- extensão para múltiplos horizontes regulatórios;
- uso como *challenger model* em validação independente;
- integração com exercícios de *stress testing*.

11 Referência

Frangieh, G. (2025). *FRTB & Default Risk Charge (DRC) Modeling: Insights*. Risk Management, Finance and Banking – Senior Advisor, September 2025.

A Código-Fonte

A.1 Geração da Carteira

```python
# generate_portfolio.py
import numpy as np
import pandas as pd

np.random.seed(42)

N_ISSUERS = 200

issuers = []
sectors = ["GOV", "FIN", "CORP"]
instr_types = ["BOND", "EQUITY"]

for i in range(N_ISSUERS):
    sector = np.random.choice(sectors, p=[0.6, 0.25, 0.15])
    instr = np.random.choice(instr_types, p=[0.8, 0.2])

    # Mark-to-market (concentra  o forte)
    mtm = np.random.lognormal(mean=2.5, sigma=1.0)

    # Recovery por tipo
    if instr == "BOND":
        rr = np.random.uniform(0.35, 0.45)
    else:
        rr = 0.0

    lgd = 1.0 - rr
    loss = mtm * lgd

    # PD inversamente relacionada ao tamanho
    pd_1y = np.clip(0.08 / (1 + loss), 0.0005, 0.10)

    issuers.append({
        "issuer_id": f"I{i+1:03d}",
        "issuer_name": f"Issuer_{i+1}",
        "sector": sector,
        "instrument_type": instr,
        "mtm": round(mtm, 2),
        "recovery_rate": round(rr, 2),
        "lgd": round(lgd, 2),
        "loss_default": round(loss, 4),
        "pd_1y": round(pd_1y, 5)
    })

```

```
df = pd.DataFrame(issuers)

path = r"C:\Users\Lenovo\Desktop\Desktop\Mestrado FGV\FRTB\trading_book.csv"
df.to_csv(path, index=False)

print("Carteira Trading Book criada")
print(df.head())
```

A.2 Funções Auxiliares

```
# hr_utils.py
import numpy as np

# ------------------------
# Fun   es  b sicas
# ------------------------
def sigmoid(x):
    return 1.0 / (1.0 + np.exp(-x))


def logit(y, eps=1e-9):
    y = np.clip(y, eps, 1 - eps)
    return np.log(y / (1 - y))


# ------------------------
#  ndices  de  concentra  o
# ------------------------
def concentration_indices(L, DP, p1=0.9, p2=0.75):
    J = len(L)
    idx = np.argsort(L)  # crescente por perda

    Ls = L[idx]
    DPs = DP[idx]

    k1 = max(1, int(np.floor(p1 * J)))
    k2 = max(1, int(np.floor(p2 * J)))

    Q1 = Ls[:k1].sum() / Ls.sum()
    Q2 = DPs[:k2].sum() / DPs.sum()

    return Q1, Q2


# ------------------------
```

```
# Simula  o estrutural
# ------------------------
def simulate_losses(J, rng):
    """
    Simula perdas com forte efeito de concentra  o
    """
    L = rng.beta(1/15, 5, size=J)
    L = np.clip(L, 1e-12, None)
    L = L / L.sum()
    return L


def simulate_pd(L, rng):
    """
    PD inversamente relacionada    perda
    """
    inv = 1.0 / (L + 1e-12)
    inv = (inv - inv.min()) / (inv.max() - inv.min())
    DP = 0.001 + 0.099 * inv
    DP *= (1 + rng.normal(0, 0.03, size=len(L)))
    return np.clip(DP, 1e-6, 0.10)


# ------------------------
# DRC exato (enumera  o)
# ------------------------
def exact_drc(L, DP, q=0.999):
    J = len(L)
    if J > 20:
        raise ValueError("Enumera  o exata invi vel para J > 20")

    n = 2 ** J
    losses = np.zeros(n)
    probs = np.zeros(n)

    for m in range(n):
        loss = 0.0
        prob = 1.0
        for j in range(J):
            if (m >> j) & 1:
                loss += L[j]
                prob *= DP[j]
            else:
                prob *= (1 - DP[j])
        losses[m] = loss
        probs[m] = prob

```

```
    idx = np.argsort(-losses)
    losses = losses[idx]
    probs = probs[idx]

    tail = 1 - q
    cum = np.cumsum(probs)
    k = np.searchsorted(cum, tail, side="right")

    return losses[max(0, k - 1)]


# ------------------------
# Regress o log stica (OLS no logit)
# ------------------------
def estimate_hr_parameters(Q1, Q2, y):
    z = logit(y)
    X = np.column_stack([np.ones(len(Q1)), Q1, Q2])
    beta, *_ = np.linalg.lstsq(X, z, rcond=None)
    return beta
```

A.3 Calibração do Modelo HR

```
# hr_calibration.py
import os
import sys

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
if BASE_DIR not in sys.path:
    sys.path.insert(0, BASE_DIR)

import numpy as np
from hr_utils import (
    simulate_losses, simulate_pd,
    concentration_indices, exact_drc,
    estimate_hr_parameters
)

rng = np.random.default_rng(123)

# ------------------------
# Configura  o
# ------------------------
N_SIM = 50        # n mero de carteiras sint ticas
J_CAL = 12        # dimens o pequena (enumera  o vi vel)

Q1_list, Q2_list, y_list = [], [], []
```

```python

# ------------------------
# Simula  es
# ------------------------
for _ in range(N_SIM):
    L = simulate_losses(J_CAL, rng)
    DP = simulate_pd(L, rng)

    Q1, Q2 = concentration_indices(L, DP)
    drc = exact_drc(L, DP)

    y = drc / L.sum()

    Q1_list.append(Q1)
    Q2_list.append(Q2)
    y_list.append(y)

Q1 = np.array(Q1_list)
Q2 = np.array(Q2_list)
y  = np.array(y_list)

# ------------------------
# Estima  o dos par metros
# ------------------------
B0, B1, B2 = estimate_hr_parameters(Q1, Q2, y)

print("=== Calibra  o HR conclu da ===")
print(f" 0  = {B0:.6f}")
print(f" 1  = {B1:.6f}")
print(f" 2  = {B2:.6f}")

# Salvar par metros
np.savez(
    "hr_parameters.npz",
    B0=B0, B1=B1, B2=B2
)
```

A.4 Aplicação do Modelo à Carteira

```python
# hr_application.py
import numpy as np
import pandas as pd
from hr_utils import concentration_indices, sigmoid

# ------------------------
# Ler carteira real
```

```
# ------------------------
path = r"C:\Users\Lenovo\Desktop\Desktop\Mestrado FGV\FRTB\trading_book.csv"
df = pd.read_csv(path)

L = df["loss_default"].values
DP = df["pd_1y"].values

# ------------------------
#  ndices  estruturais
# ------------------------
Q1_star, Q2_star = concentration_indices(L, DP)

# ------------------------
# Ler par metros calibrados
# ------------------------
params = np.load("hr_parameters.npz")
B0, B1, B2 = params["B0"], params["B1"], params["B2"]

# ------------------------
# Previs o do DRC
# ------------------------
z = B0 + B1 * Q1_star + B2 * Q2_star
y_star = sigmoid(z)

DRC_star = y_star * L.sum()

# ------------------------
# Resultados
# ------------------------
print("\n=== DRC via Heuristic Regression ===")
print(f"N mero de emissores: {len(df)}")
print(f"Q1*(90%): {Q1_star:.6f}")
print(f"Q2*(75%): {Q2_star:.6f}")
print(f"DRC normalizado y*: {y_star:.6f}")
print(f"DRC(99.9%) total: {DRC_star:.6f}")

# ------------------------
# Top contribuidores
# ------------------------
df["loss_share"] = df["loss_default"] / L.sum()
top = df.sort_values("loss_default", ascending=False).head(10)

print("\nTop 10 emissores por perda:")
print(top[[
    "issuer_name", "sector", "instrument_type",
    "loss_default", "pd_1y", "loss_share"
```

```
54 ]])
```