3rd Webinar on Climate Risk
5th December 2025

Catastrophe Modelling for Informed Decision Making & Its Intersection with Climate Science

Srishti Grover
Assistant Director – Moody's

Institute of Actuaries of India

Kerela 2018

- 23% higher rainfall in the monsoon season with August seeing 96% above normal rainfall
- Worst flood in Kerela in over a century!
- 480+ deaths, 33000 people rescued and Rs.40,000 crore worth of property was damaged.
- Days after the flood, severe drought caught Kerela! Top layer of soil was run-off, up to 2 meters in some places! Worsened by climate change and man-made activities/situations like – deforestation, sand-mining etc.

Contents

Institute of Actuaries of India

- Linking traditional vendor models with modern climate-aligned analytics
- Transforming large hazard/exposure/ loss datasets into actionable insights
- Understanding regulatory pressures shaping the Indian insurance market
- Building a Catastrophe Model- Demo

Why This Topic Now?

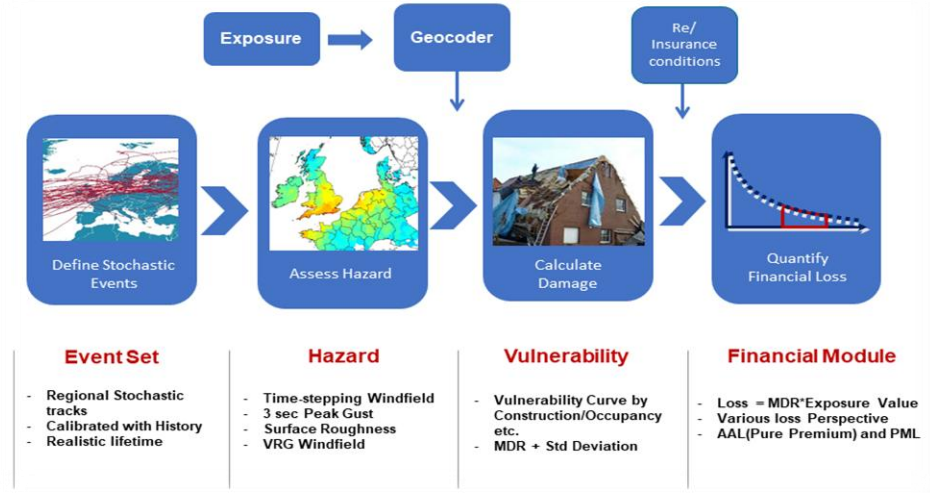
- The shift in global risk landscape: Higher hazard volatility (ENSO cycles, warming oceans, monsoon destabilization)
- Rising severity of "clash events" — SCS + flood, cyclone + rainfall + surge
- Insurer solvency pressure → need for deeper risk intelligence
- IFRS S2, BRSR Core, EU CSRD pushing climate scenario disclosure
- India's insurance penetration increasing → demand for better pricing & reserving

Goal for actuarial students: To understand how catastrophe models + climate science + open data can directly improve underwriting, reserving, portfolio optimization, and regulatory compliance.

What Is Catastrophe Modelling?

- Cat models are giant simulation engines that ask:
  - What if a big disaster hits your portfolio tomorrow? Next year? Once in 200 years? And they estimate your losses
  - Insurer solvency pressure → need for deeper risk intelligence

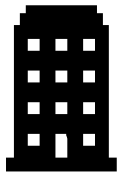
- A catastrophe model combines scientific hazard simulations, engineering vulnerability curves, financial terms, and stochastic event catalogues to estimate probabilistic and scenario losses for natural and man-made catastrophes.



What Is Catastrophe Modelling?

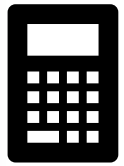
Hazard Module

- Cyclone, flood, earthquake, SCS/Hail, wildfire
- ~50,000–1,000,000 stochastic events
- Historical vs Stochastic Catalogues



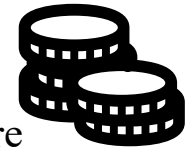
Exposure Module

- Location | Construction | Height | Year Built | Occupancy
- TIV, limits, deductibles



Vulnerability Module

- Damage ratios
- Fragility curves: engineering + empirical loss data
- Secondary modifiers



Financial Module

- Policy terms, occurrence & aggregate structures
- Facultative & treaty terms

Loss Outputs

- EP curves
- Return periods
- TVaR
- Scenario losses

Why Actuaries Must Understand Cat Models?

Cat modelling drives many major actuarial decisions:

- Pricing (adequacy of premium)
- Solvency, RBC, reinsurance optimization
- Capital allocation (1-in-200, tail risk, TVaR)
- Emerging risk assessment
- Climate transition vs physical impacts
- Strategic underwriting (geo-mix, peril-mix, accumulation)

Climate-Related Perils

Institute of Actuaries of India

1. Cyclone

- More east-coast intensification (Bay of Bengal Warming Trend)
- Slower translation speeds → heavier rainfall → compound flood

2. Flood

- Increasing monsoon variance
- Riverine + pluvial + coastal compound flooding
- Kerala 2018: 2,600 mm rainfall in 3 months

3. Heatwaves & Wildfires

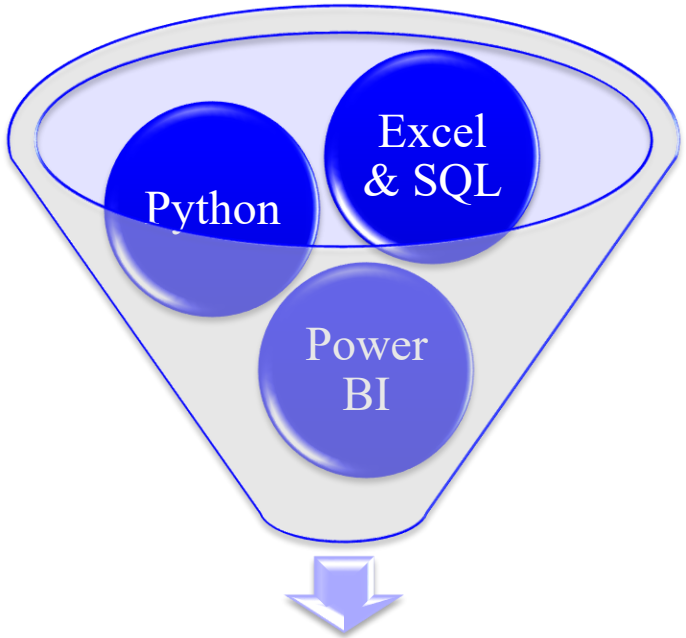
- Under-modelled in India
- North India seeing hail + Mortality + business interruption risk is rising

4. Severe Convective Storms (SCS)

- Straight-line wind clusters
- Major clash peril for insurers due to urban expansion

Working with Large Datasets

- Actuarial students will often work with:
  - Millions of locations
  - 20–50 attributes per record
  - Multiple portfolios ranging from multi year, multi geography etc.
  - Multiple vendor outputs/inputs per scenario
  - Hazard intensities, vulnerability fields, lookup keys
  - Large return-period loss tables
  - Unstructured climate data (IMD rainfall, CMIP6 projections)



Common tools

Statistical Approaches for Actionable Insights

Institute of Actuaries of India

EP Curves (Occ & Agg)

- Mean, Median, PML, TVaR
- Stress & reverse-stress testing

Return Period Analytics

- 1-in-200 (Solvency)
- 1-in-50 (Pricing Stability)
- 1-in-20 (Operational risk)

Attribution & Contribution Analysis

- Per peril, state, occupancy, construction
- Pareto loss drivers, Lorenz curves

Climate-conditioned adjustments

- Using IMD extreme rainfall trends
- Parametric adjustments to ARI/return period
- Vendor scenario overlays

Regulatory Requirements

India:

- IRDAI Catastrophe Reserves (typhoon, flood, EQ)
- BRSR Core
- ESG Chapter in annual reports
- Stress testing (upcoming)
- Climate-risk integration (2025–2030 roadmap)

Global – relevant for cross-border reinsurers:

- IFRS S1 & S2
- EU CSRD
- NAIC climate scenario guidance
- PRA (UK) biennial exploratory scenario (BES)

Leveraging Public Open-Source Flood Datasets (India)

Institute of Actuaries of India

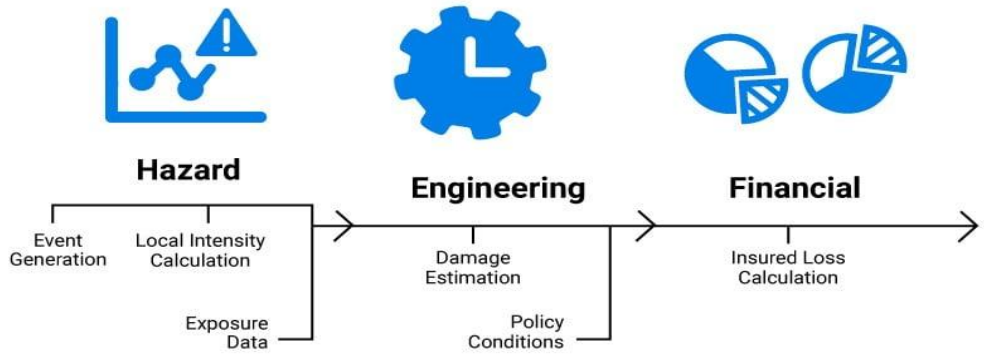
Available datasets

- IMD gridded rainfall (0.25°)
- GLOFAS (Global Flood Awareness System)
- NASA DEM (Digital Elevation Models)
- SRTM 30 m resolution
- CWC flood forecast
- IBTrACS (cyclone tracks)
- NRSC Bhuvan Flood Layers

How to use them for modelling

- Create rainfall anomaly maps → flood susceptibility layers
- Combine DEM + river networks → inundation depth models
- Assign depth-damage curves by occupancy
- Build socio-economic resilience index (state-wise HDI, GDP, poverty)
- Run “what-if” scenarios for 50-year, 100-year, 200-year rainfall

Building a Catastrophe Model- Demo

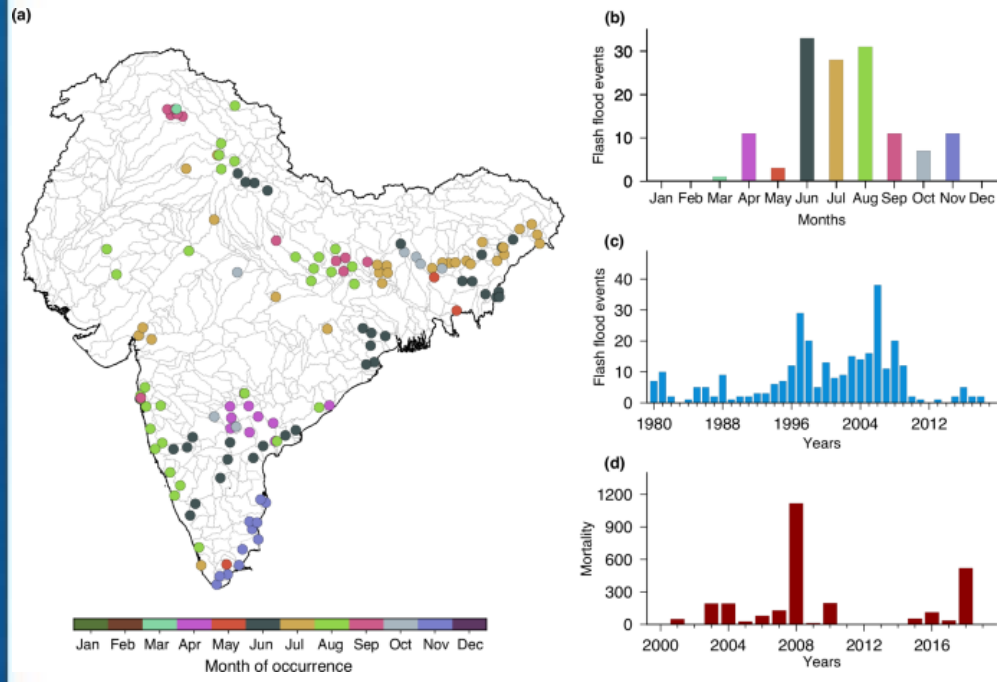



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

np.random.seed(42)  # reproducibility
```

```python
# ------------------------------------------------------------
# 1. CREATE A HISTORICAL EVENT CATALOGUE (500 EVENTS)
# ------------------------------------------------------------
n_events = 500

events = pd.DataFrame({
    "event_id": np.arange(1, n_events+1),
    # Random "epicentre" over a box (say Kerala-like)
    "lat": np.random.uniform(8, 13, n_events),
    "lon": np.random.uniform(74, 78, n_events),
    # Event severity index (e.g., peak rainfall / wind / flood index)
    "severity": np.random.gamma(shape=2.0, scale=1.5, size=n_events)
})

# Assign an annual occurrence rate λ_i for each event
# Here: a few big events with higher rate, many small ones with lower rate
base_rate = 5.0  # total expected events per year across portfolio
relative_weight = events["severity"] / events["severity"].sum()
events["lambda"] = base_rate * relative_weight  # λ_i sum to base_rate
```

Each event is assigned a severity and a frequency. Large events get slightly higher weight

Building a Catastrophe Model- Demo




```python
# ------------------------------------------------------------
# 2. BUILD EXPOSURE: 100 LOCATIONS WITH TIV + OCCUPANCY
# ------------------------------------------------------------
n_locations = 100

occupancy_types = ["Residential", "Commercial", "Industrial"]
occupancy_probs = [0.6, 0.3, 0.1]

locations = pd.DataFrame({
    "loc_id": np.arange(1, n_locations+1),
    "lat": np.random.uniform(8, 13, n_locations),
    "lon": np.random.uniform(74, 78, n_locations),
    "TIV": np.random.uniform(5e6, 5e7, n_locations),  # total insured value
    "occupancy": np.random.choice(occupancy_types,
                                  size=n_locations, p=occupancy_probs)
})
```

How portfolios are represented spatially before Hazard simulation

Building a Catastrophe Model- Demo

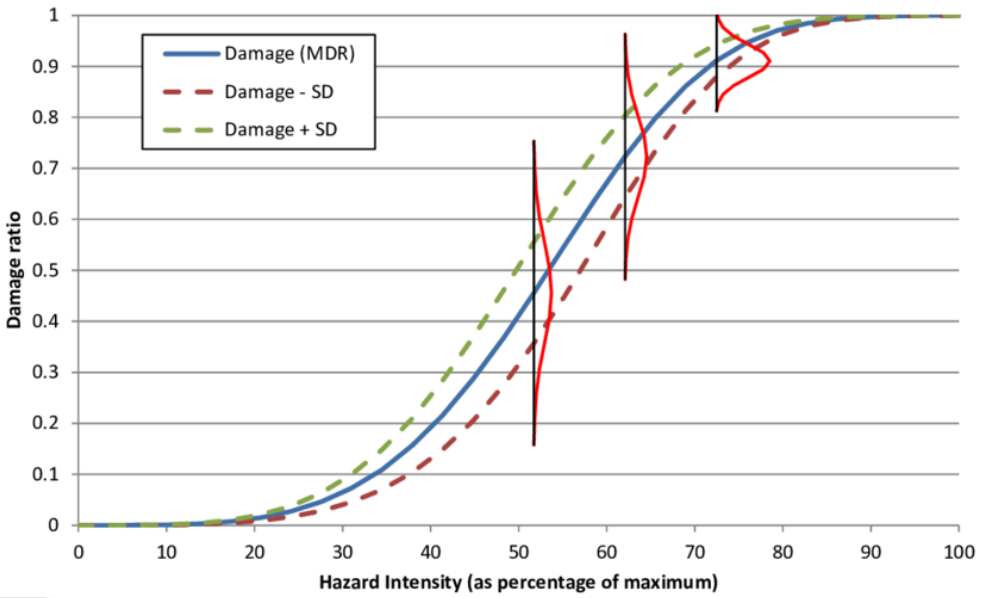
Radial Decay function: Intensity = Severity * e^-distance
The closer the building is to the event epicenter, the higher the wind speed or flood depth.

```
# ------------------------------------------------------------
# 3. SIMPLE HAZARD MODEL: INTENSITY DECAYS WITH DISTANCE
# ------------------------------------------------------------
from scipy.spatial.distance import cdist

# Precompute location coordinates
loc_coords = locations[["lat", "lon"]].values

def compute_intensity(event_lat, event_lon, severity, loc_coords,
                      r0=0.3, decay_km=150.0):
    """
    Simple radial decay: intensity at location
    = severity * exp(-distance / decay_scale)
    distance computed crudely in degrees -> km.
    """
    event_coord = np.array([[event_lat, event_lon]])
    # degrees to "km-ish" (very rough, fine for demo)
    dist_deg = cdist(event_coord, loc_coords)[0]
    dist_km = dist_deg * 110.0
    intensity = severity * np.exp(-dist_km / decay_km)
    return intensity
```

Building a Catastrophe Model- Demo

Institute of Actuaries of India

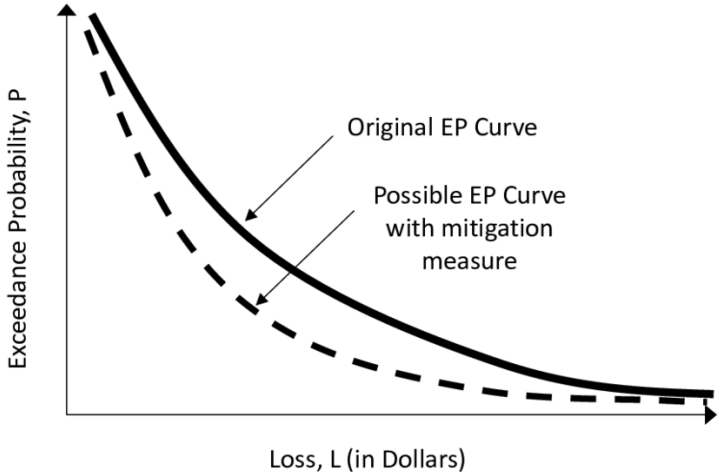
```python
# ------------------------------------------------------------
# 4. VULNERABILITY MODULE: DAMAGE RATIO CURVES
# ------------------------------------------------------------
def vuln_curve(intensity, occupancy):
    """
    Deterministic mean damage ratio as function of hazard intensity.
    Use different curves by occupancy type.
    intensity ~ 0-something (unitless flood/wind index).
    """
    # Base coefficients per occupancy
    if occupancy == "Residential":
        a, b = 0.30, 1.2
    elif occupancy == "Commercial":
        a, b = 0.35, 1.3
    else:  # Industrial
        a, b = 0.40, 1.4

    mean_dr = a * np.power(intensity, b)
    return np.clip(mean_dr, 0.0, 1.0)

def simulate_damage_ratios(intensity_vec, occ_vec, sigma=0.4, n_mc=100
    """
    Secondary uncertainty via lognormal scatter around vuln_curve mean
    Returns:
        mean_damage_ratio (per location),
        std_dev_damage_ratio (per location)
    """
    n_loc = len(intensity_vec)
    mean_dr = np.zeros(n_loc)
    std_dr = np.zeros(n_loc)

    for i in range(n_loc):
        m = vuln_curve(intensity_vec[i], occ_vec[i])
        # avoid log problems
        m = max(m, 1e-4)
        # lognormal with mean m, approximate sigma in log-space
        mu_ln = np.log(m) - 0.5 * sigma**2
        samples = np.exp(np.random.normal(mu_ln, sigma, size=n_mc))
        samples = np.clip(samples, 0.0, 1.0)
        mean_dr[i] = samples.mean()
```



Building a Catastrophe Model- Demo

Institute of Actuaries of India

```
# ------------------------------------------------------------
# 5. BUILD ELT: EXPECTED LOSS PER EVENT (ACROSS PORTFOLIO)
# ------------------------------------------------------------
elt_records = []

for idx, ev in events.iterrows():
    intensity = compute_intensity(ev["lat"], ev["lon"],
                                  ev["severity"], loc_coords)
    occ_vec = locations["occupancy"].values
    mean_dr, std_dr = simulate_damage_ratios(intensity, occ_vec,
                                             sigma=0.5, n_mc=200)

    tiv = locations["TIV"].values
    loc_mean_loss = mean_dr * tiv
    loc_var_loss = (std_dr * tiv)**2

    event_mean_loss = loc_mean_loss.sum()
    event_std_loss = np.sqrt(loc_var_loss.sum())  # assuming loc independence

    elt_records.append({
        "event_id": ev["event_id"],
        "lambda": ev["lambda"],
        "mean_loss": event_mean_loss,
        "std_loss": event_std_loss
    })

elt = pd.DataFrame(elt_records)
elt = elt[elt["mean_loss"] > 0]  # drop events with zero impact
elt.sort_values("mean_loss", ascending=False, inplace=True)
```




```
# ------------------------------------------------------------
# 6. OCCURRENCE EP CURVE & RETURN PERIODS FROM ELT
# ------------------------------------------------------------
# Sort events by descending loss
elt_sorted = elt.sort_values("mean_loss", ascending=False).reset_index(drop=True)

# Occurrence exceedance probability: P(Loss >= x) in a given year
# Under Poisson assumption, occurrence EP at threshold x is:
# EP(x) = 1 - exp(- sum_{events with loss>=x} lambda_i )

losses = elt_sorted["mean_loss"].values
lambdas = elt_sorted["lambda"].values

cum_lambdas = np.cumsum(lambdas)  # as we go down the list
ep_occ = 1.0 - np.exp(-cum_lambdas)

# Return period (years) = 1 / EP
rp_occ = 1.0 / ep_occ

ep_curve = pd.DataFrame({
    "mean_loss": losses,
    "EP_occurrence": ep_occ,
    "ReturnPeriod_occ": rp_occ
})
```

Building a Catastrophe Model- Demo

Institute of Actuaries of India

```python
# ------------------------------------------------------------
# 7. TVaR (TAIL VaR) FROM ELT
# ------------------------------------------------------------
def tvar_from_elt(elt_df, prob_level):
    """
    TVaR at probability level p (e.g. p=0.01 for 1% tail).
    Uses severity distribution implied by ELT lambdas.
    """
    df = elt_df.copy()
    # weight of each event in severity distribution ~ lambda_i
    df["weight"] = df["lambda"] / df["lambda"].sum()
    df.sort_values("mean_loss", inplace=True)
    df["cumprob"] = df["weight"].cumsum()

    # losses in tail where cumprob > 1 - p
    tail = df[df["cumprob"] > 1 - prob_level]
    if tail.empty:
        return 0.0

    # approximate conditional mean in tail
    tvar = np.average(tail["mean_loss"], weights=tail["weight"])
    # find VaR at that level (quantile)
    var = df.loc[df["cumprob"] >= 1 - prob_level, "mean_loss"].iloc[0]
    return var, tvar

for rp in [10, 20, 50, 100, 200]:
    p = 1.0 / rp  # exceedance probability
    var, tvar = tvar_from_elt(elt, p)
    print(f"{rp}-year Occurrence:")
    print(f"  VaR  (PML) ≈ {var:,.0f}")
    print(f"  TVaR (Tail VaR) ≈ {tvar:,.0f}\n")
```