Practical 2

Melvil Deleage, Jeff Macaraeg

May 20, 2025

Table of contents

Part 1 - Venice

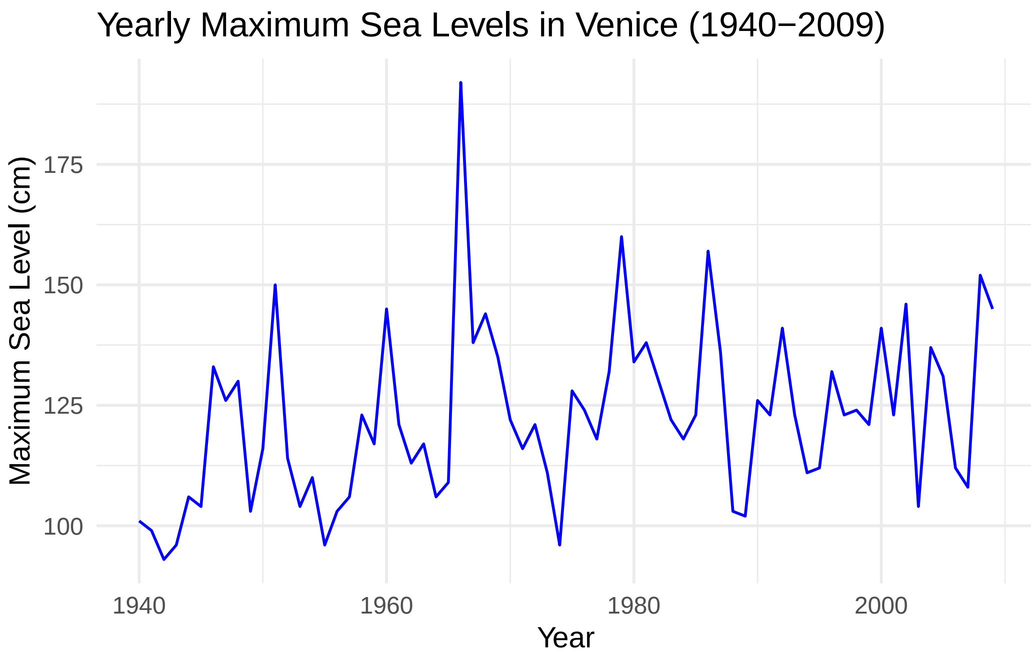
The `venice90` dataset can be found in the `VGAM` package.

Question a)

> Read in the data. Extract and represent the yearly max values from 1940 to 2009. What do you observe ?

```r
library(VGAM)
data(venice90)

# Transform venice90 into a data frame
venice90_df <- as.data.frame(venice90)

# Group by year and extract the maximum sea level per year between 1940 to 2009
yearly_max <- venice90_df %>%
  group_by(year) %>%
  summarise(max_sealevel = max(sealevel))

# Plot the yearly maximum sea levels
ggplot(yearly_max, aes(x = year, y = max_sealevel)) +
  geom_line(color = "blue") +
  labs(
    x = "Year",
```

```
    y = "Maximum Sea Level (cm)",
    title = "Yearly Maximum Sea Levels in Venice (1940-2009)"
  ) +
  theme_minimal()
```



We can observe some variability over the years and a slight upward trend, so the maximum levels in Venice seem to be increasing.

Question b)

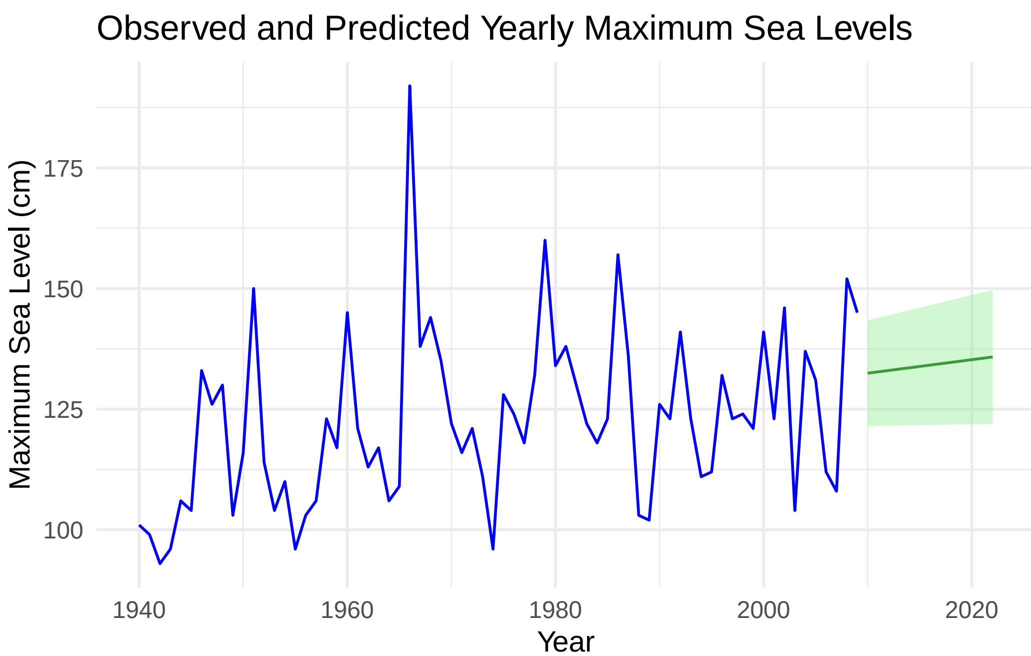
> We are end of 2009 and would like to predict the yearly maximum values over the next 13 years (from 2010 to 2022). A naive approach consists of fitting a linear model on the observed yearly maxima and predict their values for 2010–2022. Proceed to this prediction and provide confidence intervals.

```
# Fit linear model
model <- lm(max_sealevel ~ year, data = yearly_max)

# Predict for 2010-2022 with confidence intervals
future_years <- data.frame(year = 2010:2022)
pred <- predict(model, newdata = future_years, interval = "confidence", level = 0.99)

# Show predictions
cbind(future_years, pred)
```

```
   year      fit      lwr      upr
1  2010 132.4522 121.4683 143.4361
2  2011 132.7321 121.5137 143.9505
3  2012 133.0121 121.5576 144.4665
4  2013 133.2920 121.6002 144.9838
5  2014 133.5719 121.6414 145.5025
```

```
6  2015 133.8519 121.6813 146.0225
7  2016 134.1318 121.7200 146.5436
8  2017 134.4118 121.7576 147.0659
9  2018 134.6917 121.7942 147.5892
10 2019 134.9716 121.8298 148.1135
11 2020 135.2516 121.8644 148.6387
12 2021 135.5315 121.8982 149.1649
13 2022 135.8115 121.9311 149.6919
```

```r
# Combine predictions with years
pred_df <- cbind(future_years, as.data.frame(pred))

# Plot observed and predicted with confidence intervals
ggplot() +
  geom_line(data = yearly_max, aes(x = year, y = max_sealevel), color = "blue") +
  geom_line(data = pred_df, aes(x = year, y = fit), color = "darkgreen") +
  geom_ribbon(data = pred_df, aes(x = year, ymin = lwr, ymax = upr), fill = "lightgreen", alpha =
  labs(x = "Year", y = "Maximum Sea Level (cm)",
       title = "Observed and Predicted Yearly Maximum Sea Levels") +
  theme_minimal()
```



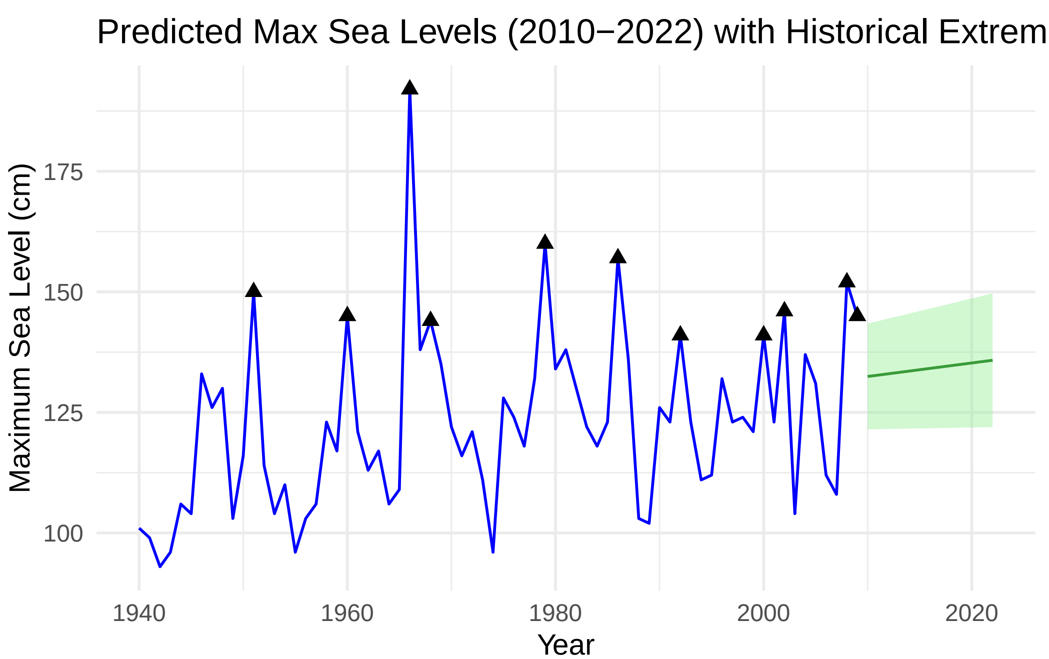
We used a confidence interval of 99% to predict for the years 2010 to 2022.

Question c)

> Represent in the same graph the predicted yearly max values for the period 2010–2022, their pointwise confidence bounds and the observed values greater than 140 cm from the table below.

```r
# Observed values > 140 cm
extreme_vals <- yearly_max %>% filter(max_sealevel > 140)
```

```
# Plot everything together
ggplot() +
  # Historical data
  geom_line(data = yearly_max, aes(x = year, y = max_sealevel), color = "blue") +

  # Predictions
  geom_line(data = pred_df, aes(x = year, y = fit), color = "darkgreen") +
  geom_ribbon(data = pred_df, aes(x = year, ymin = lwr, ymax = upr),
              fill = "lightgreen", alpha = 0.4) +

  # Highlight points > 140 cm
  geom_point(data = extreme_vals, aes(x = year, y = max_sealevel),
             color = "black", size = 2, shape = 17) +  # triangle shape

  labs(x = "Year", y = "Maximum Sea Level (cm)",
       title = "Predicted Max Sea Levels (2010-2022) with Historical Extremes (>140 cm)") +
  theme_minimal()
```



This plot provides all the necessary information, from the historical data in the blue line, to the yearly maximum values with the red points, the dark green line being the prediction for 2010 to 2022, the light green area being the confidence intervals and finally, the black triangles being the values greater than 140cm.

Now we perform a risk analysis and because we are interested in the period 2010–2022, we want to calculate the 13-years return level., for each year.

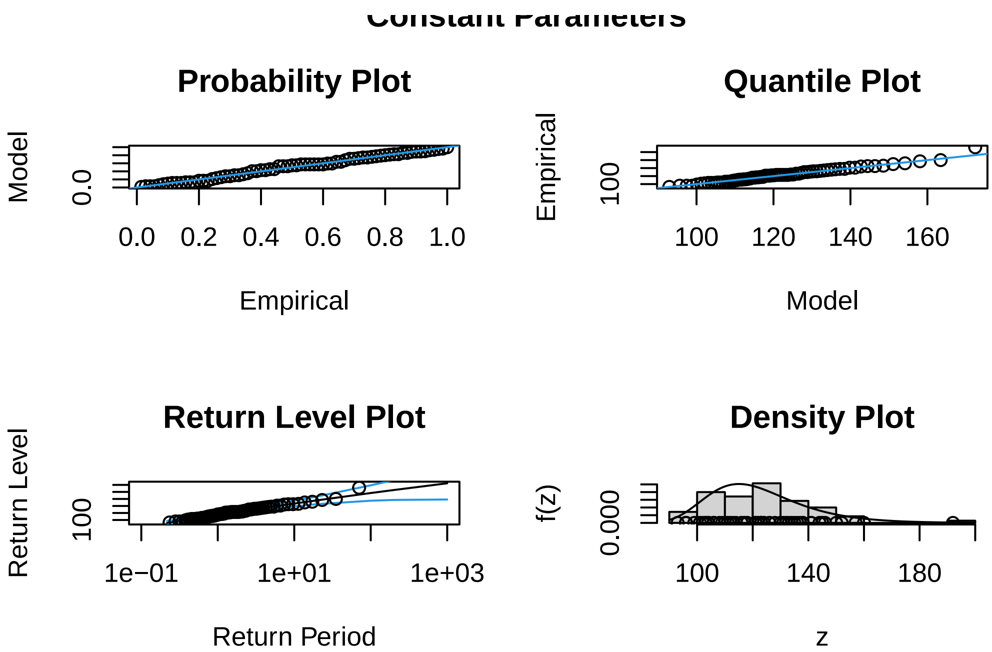
Question d)

Fit a GEV a with constant parameters to the historical yearly max values. Fit a GEV with time varying location parameter. Compare the two embedded models using likelihood ratio test (LRT). Show diagnostic plots.

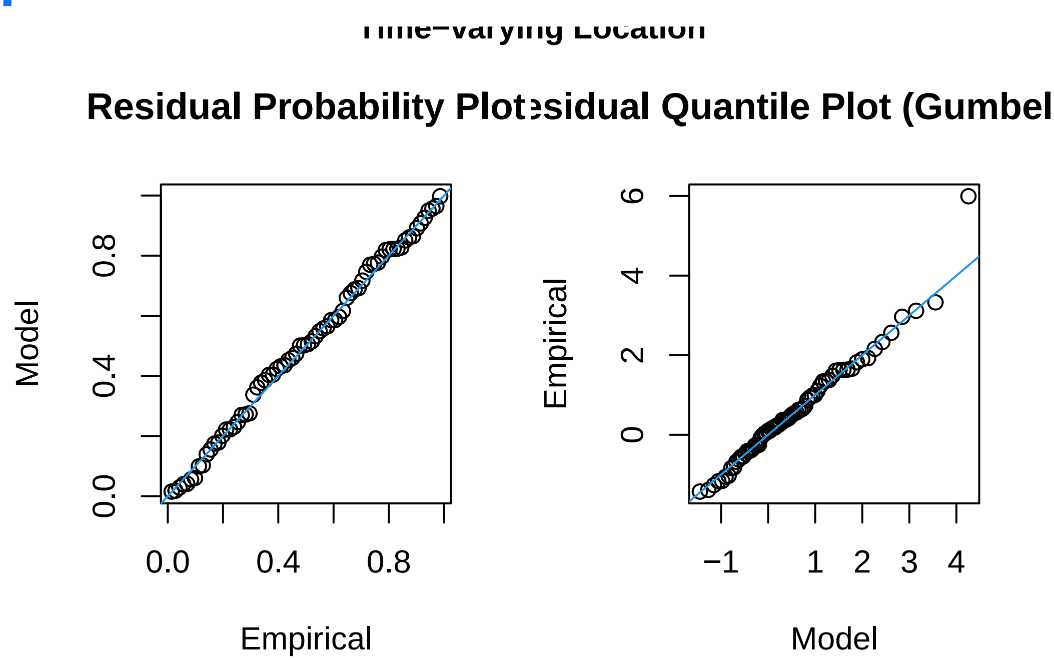
```r
# Prepare the data (1940-2009), extract yearly maxima and center the year
venice90_df <- as_tibble(venice90) %>%
  filter(year >= 1940, year <= 2009) %>%
  group_by(year) %>%
  summarise(max_sealevel = max(sealevel), .groups = "drop") %>%
  mutate(year_centered = year - mean(year))  # mean-centred year

sea_levels <- venice90_df$max_sealevel                       # response
year_covariate <- matrix(venice90_df$year_centered, ncol = 1)  # 1-column covariate matrix

# Helper to rename GEV parameter estimates
name_gev_par <- function(fit, location_trend = FALSE, scale_trend = FALSE, shape_trend = FALSE) {
  names_vec <- c("location0")
  if (location_trend) names_vec <- c(names_vec, "location1")
  names_vec <- c(names_vec, "scale0")
  if (scale_trend) names_vec <- c(names_vec, "scale1")
  names_vec <- c(names_vec, "shape0")
  if (shape_trend) names_vec <- c(names_vec, "shape1")

  stopifnot(length(names_vec) == length(fit$mle))
  names(fit$mle) <- names(fit$se) <- names_vec
  fit
}

# Fit GEV with constant parameters
fit_const <- gev.fit(sea_levels, show = FALSE) |> name_gev_par()

# Fit GEV with time-varying location (trend on location)
fit_trend <- gev.fit(sea_levels, ydat = year_covariate, mul = 1, show = FALSE) |> name_gev_par(lo

# Likelihood Ratio Test
LRT <- 2 * (-fit_trend$nllh + fit_const$nllh)
pval <- pchisq(LRT, df = 1, lower.tail = FALSE)

# Select best model
if (pval < 0.05) {
  best_fit <- fit_trend
  best_model <- "Time-varying Location"
} else {
  best_fit <- fit_const
  best_model <- "Constant Parameters"
}
cat(sprintf("\n--- d) Likelihood-ratio test ---\nLRT = %.2f,  p = %.3f\nSelected model: %s\n",
            LRT, pval, best_model))
```

```
--- d) Likelihood-ratio test ---
LRT = 11.62,  p = 0.001
Selected model: Time-varying Location
```

```r
# Diagnostic plots
par(mfrow = c(2,2)); gev.diag(fit_const); title("Constant Parameters", outer = TRUE)
```




```r
par(mfrow = c(2,2)); gev.diag(fit_trend); title("Time-varying Location", outer = TRUE)
```



We fitted a constant and a time-varying model. The latter is better thanks to the low p-value and the log-likelihood of 11.62.The model looks overall okay, despite having some outliers which might influence it. There are no major pattern nor heteroskedasticity.

Question e)

Add if necessary a time varying scale and or shape GEV parameter. Select the best model according to LRT.

```
# Fit GEV models with additional time-varying parameters
fit_loc_scale <- gev.fit(sea_levels, ydat = year_covariate, mul = 1, sigl = 1,
                         show = FALSE) |> name_gev_par(location_trend = TRUE, scale_trend = TRUE)

fit_loc_shape <- gev.fit(sea_levels, ydat = year_covariate, mul = 1, shl = 1,
                         show = FALSE) |> name_gev_par(location_trend = TRUE, shape_trend = TRUE)

fit_loc_scale_shape <- gev.fit(sea_levels, ydat = year_covariate, mul = 1, sigl = 1, shl = 1,
                               show = FALSE) |> name_gev_par(location_trend = TRUE, scale_trend =

# Collect log-likelihoods for comparison
log_likelihoods <- purrr::map_dbl(
  list(fit_const, fit_trend, fit_loc_scale, fit_loc_shape, fit_loc_scale_shape),
  \(fit) -fit$nllh
)

names(log_likelihoods) <- c("const", "location", "location+scale", "location+shape", "location+sc

# Likelihood Ratio Test Table
lrt_tbl <- tibble(
  comparison = c("location vs const",
                 "location+scale vs location",
                 "location+shape vs location",
                 "location+scale+shape vs location+scale"),
  LR = c(2 * (log_likelihoods["location"] - log_likelihoods["const"]),
         2 * (log_likelihoods["location+scale"] - log_likelihoods["location"]),
         2 * (log_likelihoods["location+shape"] - log_likelihoods["location"]),
         2 * (log_likelihoods["location+scale+shape"] - log_likelihoods["location+scale"])),
  df = 1,
  p = pchisq(LR, df, lower.tail = FALSE)
)

print(lrt_tbl, digits = 3)
```

```
# A tibble: 4 x 4
  comparison                                LR    df        p
  <chr>                                  <dbl> <dbl>    <dbl>
1 location vs const                     11.6       1 0.000651
2 location+scale vs location             0.892     1 0.345
3 location+shape vs location             5.03      1 0.0250
4 location+scale+shape vs location+scale 5.94      1 0.0148
```

```
# Start with location trend model as baseline
best_fit <- fit_trend
best_model_name <- "location"
```

```r
# Check if adding scale improves the model significantly
if (lrt_tbl$p[2] < 0.05) {
  best_fit <- fit_loc_scale
  best_model_name <- "location+scale"
}

# If scale was not added, check if shape improves the model
if (best_model_name == "location" && lrt_tbl$p[3] < 0.05) {
  best_fit <- fit_loc_shape
  best_model_name <- "location+shape"
}

# If both location and scale are in the model, check if adding shape improves it further
if (best_model_name == "location+scale" && lrt_tbl$p[4] < 0.05) {
  best_fit <- fit_loc_scale_shape
  best_model_name <- "location+scale+shape"
}

cat("\nSelected model:", best_model_name, "\n")
```

```
Selected model: location+shape
```

The best model includes time-varying location and shape parameters. The addition of a time-varying scale is not necessary based on the LRT. This model provides the best fit and should be used for further analysis or prediction.

Question f) + g)

f) Predict the 13-years return level, each year from 2010 to 2022.
g) Calculate confidence bands for these predictions.

```r
# Extract parameter value by name (return 0 if not found)
get_param <- function(params, name) {
  ifelse(name %in% names(params), params[[name]], 0)
}

# Compute model parameters at covariate value z
get_gev_parameters <- function(fit, z) {
  p <- fit$mle
  location0 <- get_param(p, "location0"); location1 <- get_param(p, "location1")
  scale0    <- get_param(p, "scale0");    scale1    <- get_param(p, "scale1")
  shape0    <- get_param(p, "shape0");    shape1    <- get_param(p, "shape1")

  list(
    location = location0 + location1 * z,
    scale    = scale0    + scale1    * z,
    shape    = shape0    + shape1    * z
  )
}
```

```r
# Compute 13-year return level using GEV parameters
return_level <- function(location, scale, shape, m = 13) {
  p <- 1 - 1/m
  if (abs(shape) < 1e-6) {
    location - scale * log(-log(p))  # Gumbel case
  } else {
    location + (scale / shape) * ((-log(p))^(-shape) - 1)
  }
}

# Delta method standard error for return level at covariate z
return_level_se <- function(fit, z, m = 13) {
  params <- get_gev_parameters(fit, z)
  location <- params$location
  scale    <- params$scale
  shape    <- params$shape
  p        <- 1 - 1/m

  if (abs(shape) < 1e-6) {
    dloc <- 1
    dsca <- -log(-log(p))
    dshp <- 0
  } else {
    A    <- (-log(p))^(-shape) - 1
    dloc <- 1
    dsca <- A / shape
    dshp <- -scale / shape^2 * A + scale / shape * (-log(p))^(-shape) * log(-log(p))
  }

  # Gradient vector
  grad <- setNames(numeric(length(fit$mle)), names(fit$mle))
  grad["location0"] <- dloc
  if ("location1" %in% names(grad)) grad["location1"] <- dloc * z
  grad["scale0"]    <- dsca
  if ("scale1" %in% names(grad)) grad["scale1"] <- dsca * z
  grad["shape0"]    <- dshp
  if ("shape1" %in% names(grad)) grad["shape1"] <- dshp * z

  # Standard error via delta method
  sqrt(as.numeric(t(grad) %*% fit$cov %*% grad))
}

# Predictions for 2010 to 2022
years_future <- 2010:2022
z_future     <- years_future - mean(venice90_df$year)  # center future years

predicted_return_levels <- map2_dfr(years_future, z_future, \(year, z) {
  params <- get_gev_parameters(best_fit, z)
  rl     <- return_level(params$location, params$scale, params$shape, m = 13)
  se     <- return_level_se(best_fit, z, m = 13)
```

```
  tibble(
    year = year,
    return_level = rl,
    lower_bound  = rl - qnorm(0.975) * se,
    upper_bound  = rl + qnorm(0.975) * se
  )
})

# Print the predictions
print(as.data.frame(predicted_return_levels), digits = 5)
```

```
   year return_level lower_bound upper_bound
1  2010       147.78      137.25      158.32
2  2011       147.87      137.72      158.02
3  2012       147.96      138.17      157.75
4  2013       148.06      138.60      157.52
5  2014       148.17      139.00      157.33
6  2015       148.27      139.38      157.17
7  2016       148.39      139.73      157.04
8  2017       148.51      140.07      156.94
9  2018       148.63      140.39      156.87
10 2019       148.75      140.68      156.82
11 2020       148.88      140.96      156.81
12 2021       149.02      141.22      156.82
13 2022       149.16      141.47      156.85
```

```
# For better visualization
predicted_return_levels %>%
  select(year, return_level) %>%
  mutate(return_level = round(return_level, 2)) %>%
  rename(`Year` = year, `13-Year Return Level (cm)` = return_level) %>%
  kable(caption = "13-Year Return Levels for Venice (2010-2022)")
```

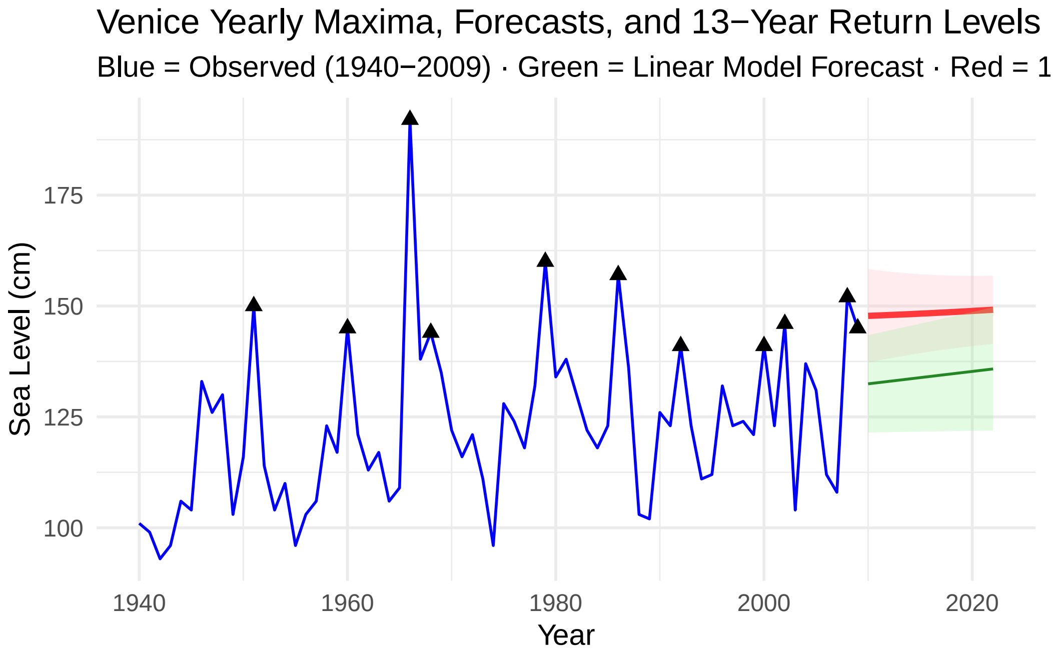
Table 1: 13-Year Return Levels for Venice (2010–2022)

Year	13-Year Return Level (cm)
2010	147.78
2011	147.87
2012	147.96
2013	148.06
2014	148.17
2015	148.27
2016	148.39
2017	148.51
2018	148.63
2019	148.75
2020	148.88
2021	149.02
2022	149.16

Question h)

Represent in the same graph your predictions of the 13-years return levels, their pointwise confidence intervals, the predicted yearly max values from the linear model and the observed values greater than 140 cm from the table below.

```
# (1) Observed yearly maxima already stored in venice90_df
# (2) Linear model forecasts from part b
linear_forecast_df <- pred_df %>%
  rename(lower_ci_linear = lwr, upper_ci_linear = upr, predicted_linear = fit)

# (3) Observed extremes > 140 cm
extreme_values <- venice90_df %>% filter(max_sealevel > 140)

# (4) Plot observed, linear forecast, and GEV return levels
ggplot() +
  # Observed data
  geom_line(data = venice90_df, aes(x = year, y = max_sealevel), color = "blue") +

  # GEV-based 13-year return level + CI
  geom_line(data = predicted_return_levels, aes(x = year, y = return_level), color = "red", linew
  geom_ribbon(data = predicted_return_levels, aes(x = year, ymin = lower_bound, ymax = upper_boun

  # Linear model predictions + CI
  geom_line(data = linear_forecast_df, aes(x = year, y = predicted_linear), color = "darkgreen")
  geom_ribbon(data = linear_forecast_df, aes(x = year, ymin = lower_ci_linear, ymax = upper_ci_li

  # Extreme observed points > 140 cm
  geom_point(data = extreme_values, aes(x = year, y = max_sealevel), shape = 17, size = 2) +

  # Labels and theme
  labs(
    x = "Year", y = "Sea Level (cm)",
    title = "Venice Yearly Maxima, Forecasts, and 13-Year Return Levels",
    subtitle = "Blue = Observed (1940-2009) · Green = Linear Model Forecast · Red = 13-Year Retur
  ) +
  theme_minimal()
```



Question i)

> Broadly speaking, each year, there is a chance of 1/13 that the observed value is above the 13-years return level. Comment the results for both the linear model prediction and GEV approach. Note that 12 of the 20 events occurred in the 21st century.

While both models provide useful insights, the linear model clearly underestimates extremes and provides overly narrow confidence intervals. The GEV approach, especially with time-varying parameters, is more suited for modeling extremes and gives a more realistic picture of sea level risk. However, even the GEV predictions fall short of the most recent high events, such as 2.04m in 2022, indicating that the system is non-stationary and that risk is increasing over time. This shift is emphasized by the concentration of extreme events in the 21st century, suggesting that return periods are shortening and that what was once a 13-year event may now be happening more frequently.