06051540-MATH70076-assessment-1

MSc in Statistics 2025/26, Imperial College London

Justin Upson

Question 1

For $\xi \neq 0$:

Assuming $\sigma > 0$, and ξ in the real numbers. We know from the cumulative distribution function that:

$$F(x; \sigma, \xi, u) = 1 - \left(1 + \frac{\xi(x-u)}{\sigma}\right)_+^{-1/\xi} \tag{1}$$

For now, we can also drop the max function part (as it pertains to the range) and we can consider that separately later. Setting $p = F(x; \sigma, \xi, u) \in [0, 1]$, and rewriting to make x the subject:

$$\left(1 + \frac{\xi(x-u)}{\sigma}\right)^{-1/\xi} = (1-p) \tag{2}$$

$$1 = (1-p)\left(1 + \frac{\xi(x-u)}{\sigma}\right)^{1/\xi} \tag{3}$$

$$(1-p)^{-1} = \left(1 + \frac{\xi(x-u)}{\sigma}\right)^{1/\xi} \tag{4}$$

$$\tag{5}$$

For $\xi > 0$, we see that $\left(1+\frac{\xi(x-u)}{\sigma}\right)^{1/\xi} > 0$, so we get $u \leq x$ with slowly decaying tails and:

$$(1-p)^{-\xi} = 1 + \frac{\xi(x-u)}{\sigma} \tag{6}$$

$$(1-p)^{-\xi} - 1 = \frac{\xi(x-u)}{\sigma} \tag{7}$$

$$\left((1-p)^{-\xi} - 1\right) \times \frac{\sigma}{\xi} = x - u \tag{8}$$

$$u + \left((1-p)^{-\xi} - 1\right) \times \frac{\sigma}{\xi} = x \tag{9}$$

$$\tag{10}$$

So $F_X^{-1}(p;\sigma,\xi,u) = u + \left((1-p)^{-\xi} - 1\right) \times \frac{\sigma}{\xi} = x$

For $\xi < 0$, we know that we have quickly decaying tails with finite upper endpoint. With $x > u$, this finite endpoint is met when

$$1 + \frac{\xi(x-u)}{\sigma} = 0 \tag{11}$$

$$\frac{\xi(x-u)}{\sigma} = -1 \tag{12}$$

$$x - u = -\frac{\sigma}{\xi} \tag{13}$$

$$x = u - \frac{\sigma}{\xi} \tag{14}$$

$$\tag{15}$$

So, in the event of $\xi < 0$, the range of our function is such that $u \leq x \leq \frac{u-\sigma}{\xi}$, for the function $F_X^{-1}(p;\sigma,\xi,u) = u + \left((1-p)^{-\xi} - 1\right) \times \frac{\sigma}{\xi} = x$

It is given in the question that for $\xi \to 0$, the GPD reduces to an exponential distribution, hence continuous.

For $\xi = 0$:

$$p = 1 - exp\left(-\frac{x-u}{\sigma}\right) \tag{16}$$

$$exp\left(-\frac{x-u}{\sigma}\right) = 1 - p \tag{17}$$

$$-\frac{x-u}{\sigma} = ln(1-p) \tag{18}$$

$$-x + u = \sigma ln(1-p) \tag{19}$$

$$u - \sigma ln(1-p) = x \tag{20}$$

$$\tag{21}$$

So for $\xi = 0$, we get that $F_X^{-1}(p;\sigma,u) = u - \sigma ln(1-p) = x$ and the range of our function is $u \leq x$

Question 2a

Defining the quantile function in the style of qnorm() and qunif(), we need out code which produces the inverse CDF whilst providing similar error messages for inadmissable inputs. From this, we get that:

```
qgpd <- function (p, sigma=1, xi=0, u=0){
  len <- max(length(p), length(sigma), length(xi), length(u))
  p <- rep_len(p, len)
  sigma <- rep_len(sigma, len)
  xi <- rep_len(xi, len)
  u <- rep_len(u, len)

  safe_xi <- ifelse(xi==0, 1, xi)

  value <- ifelse(
    xi == 0,
    u - sigma * log1p(-p),
    u + ((1 - p)^(-xi) - 1) * sigma / safe_xi)
  maxp <- u - sigma/safe_xi

  # Ensuring our range is correct
  value <- ifelse(xi<0 & p==1, maxp, value)
  value <- ifelse(xi<0 & p<1 & maxp <= value, NaN, value)

  # Handling forbidden inputs
  inadmissable <-
  (is.na(p))|(is.na(sigma))|(is.na(xi))|(is.na(u))|
  (!is.numeric(p))|(!is.numeric(sigma))|(!is.numeric(xi))|(!is.numeric(u))|
  (p < 0)|(p > 1)|(sigma <= 0)

  if (any(inadmissable, na.rm = TRUE)) warning("NaNs produced")
  value[inadmissable] <- NaN

  value

}
```

By default, the expected inputs for the function are sigma=1, xi=0, u=0. The code also prevents inputs where p values are less than 0, where p values exceed 1, or where sigma is less than or equal to 0.

The expected output is a real number greater than u that is unbounded if xi is greater than or equal to 0. The expected output is less than u - sigma/xi if xi is less than 0.

Regarding the behaviours of the quantile function: The larger the value of xi, the slower the tail decays. In this for xi >= 0 the functions output approaches infinity as p -> 1.

For xi < 0, the functions output has a maximum at u - sigma/xi (when p -> 1). For larger values of sigma, the slower the tail decays.

Question 2b

```
qgpd(0.5,2,-0.4,1.5)
```

```
[1] 2.710709
```

```
qgpd(0.75,2,-0.4,1.5)
```

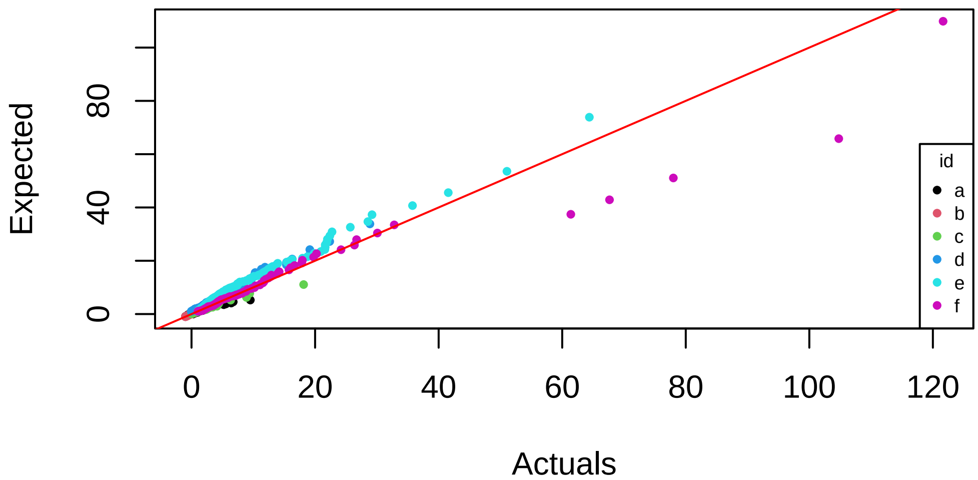
```
[1] 3.628254
```

```
qgpd(0.99,2,-0.4,1.5)
```

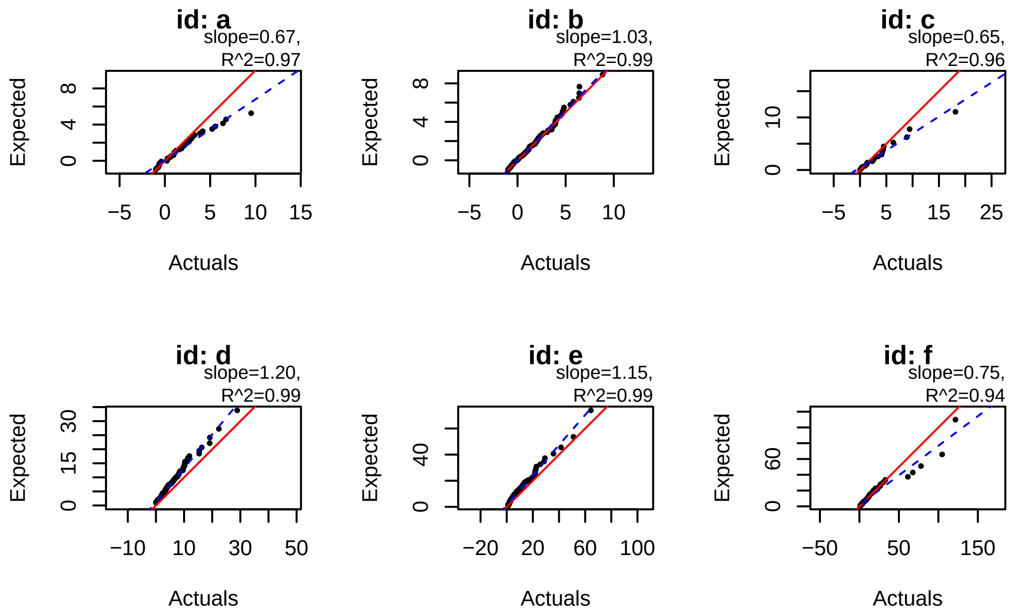
```
[1] 5.707553
```

Question 3

Graphing our actual vs expected split by id, we see that:



From the figure above, we see some interesting results – although most actuals seem in line with the expecteds (being close to the figures red line), this is not always the case. Ultimately, however, the different ids need to be separated so as to provide a more granular analysis of the distributions. In the below, the red line indicates the points for which $y = x$, whereas the blue dashed line indicates where the line of best fit through the various points:



In addition, below is the calculated Pearson correlation coefficient for each of the set ids.

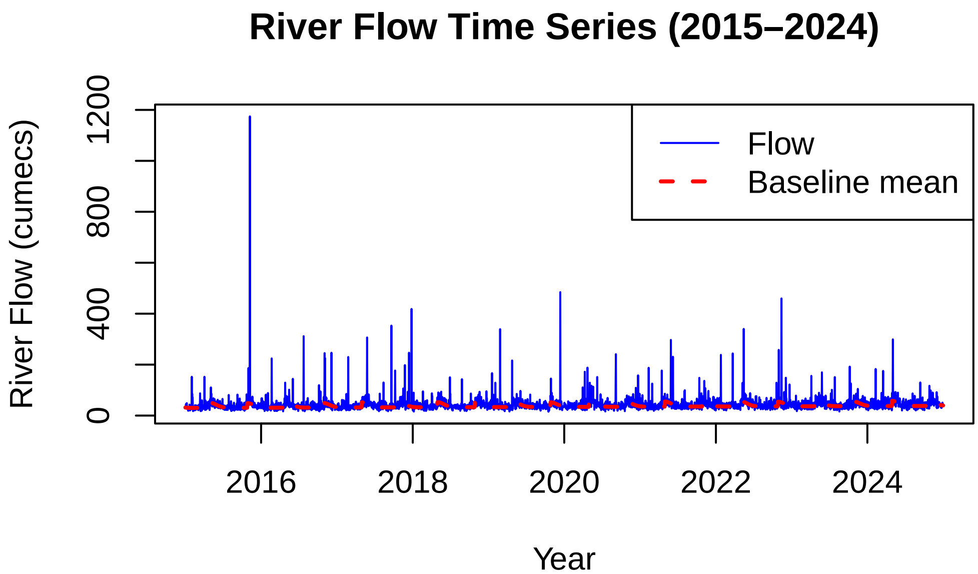
```
        a         b         c         d         e         f
0.9823689 0.9968773 0.9819495 0.9955422 0.9926756 0.9718414
```

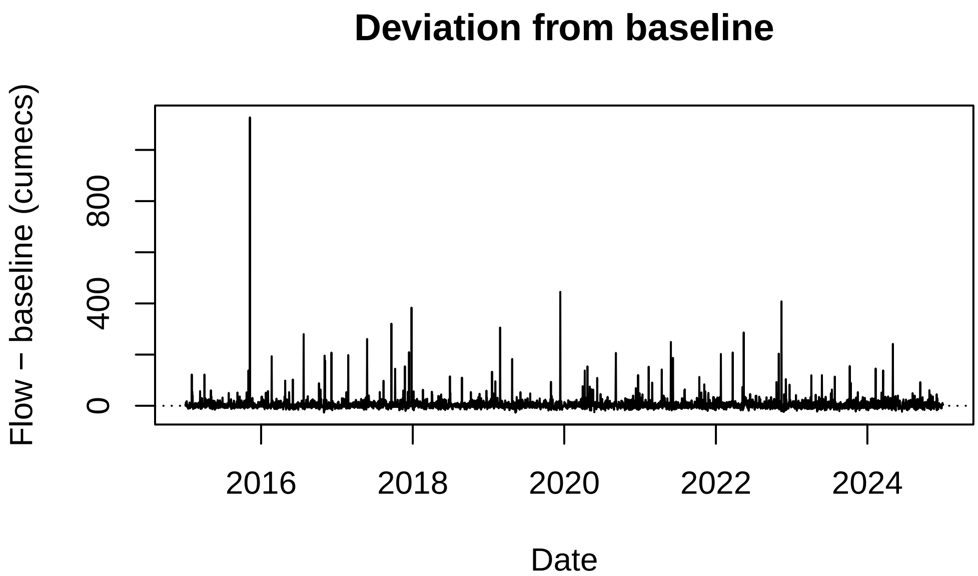
This analysis split by id shows provides a clearer insight into whether the observed data aligns with that expected from the stated distributions:

- Values taken from a are slightly beneath the $y = x$ line for smaller values and significantly beneath it for larger values - this suggests questionable alignment between the dataset and the stated distribution.
- Values taken from b are consistently on the $y = x$ line suggesting strong alignment between the dataset and the distribution.
- Values taken from c are beneath the $y = x$ line for larger values, it is also notable that c has only 20 samples within its dataset so outliers would contribute more to any fitted line.
- Values taken from d are consistently above the $y = x$ line across the entirety of the displayed points. This suggests with high likelihood that the dataset for the id is inconsistent with the stated distribution.
- Values taken from e are consistently above the $y = x$ line despite this dataset being the largest of those observed. This suggests that the dataset for this id has questionable alignment with the stated distribution.

- Values taken from f are on the $y = x$ line for smaller values, but are significantly beneath the line for larger values. This, however, must be caveated by the fact that of the 98 samples in the dataset, only 4 noticably deviate from the $y = x$ line, a result which can be explained by outliers.

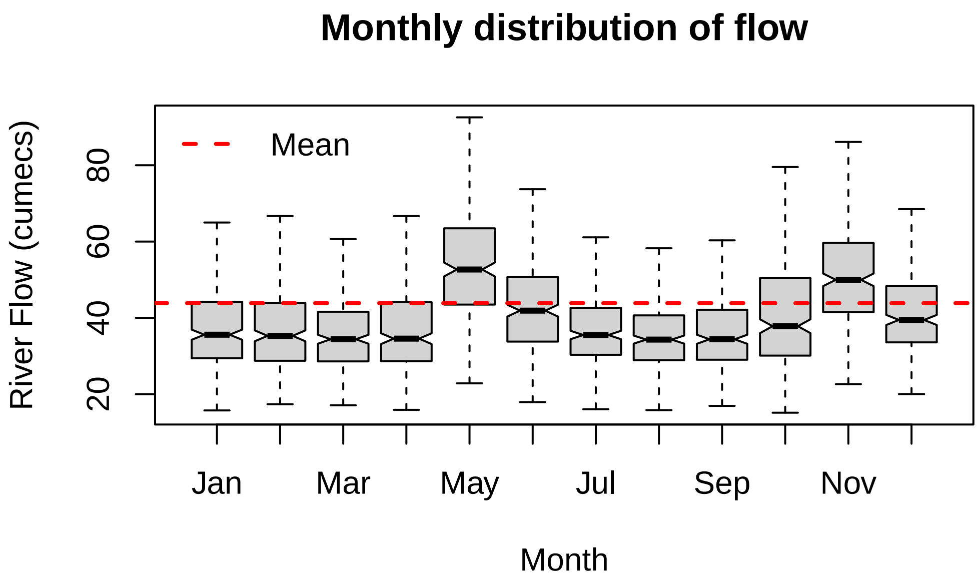
Question 4





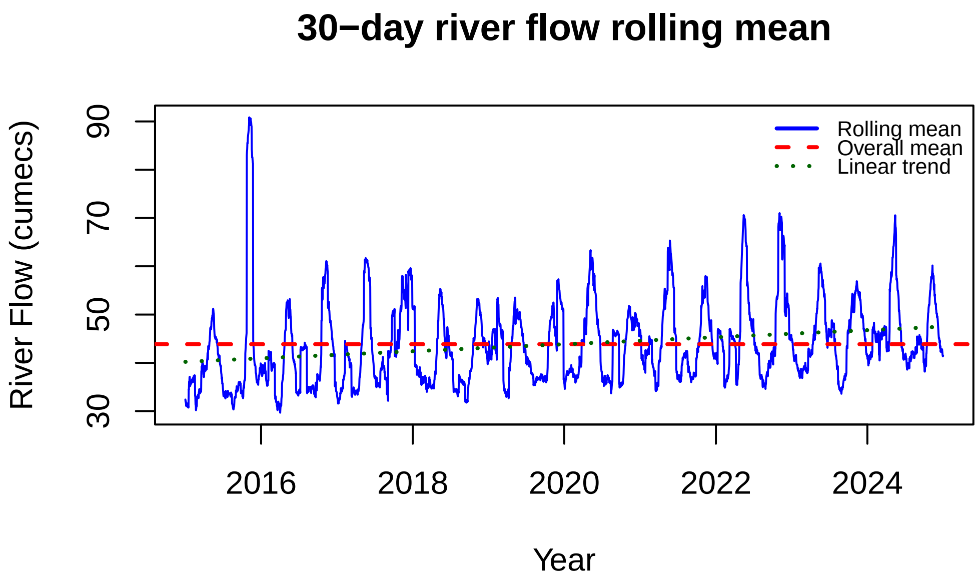
Determining whether the distribution of river flows is constant over time, we can see from the above graphs that an assumption of constant flow is flawed. This is because we see that the speed of the river's flow varies tremendously - at numerous (albeit brief) intervals over the ten year period, the speed of the river is many multiples of the mean river speed. This is something we can demonstrably see when comparing the mean red dashed line with the time series plot above.

In addition, although our boxplot below removes outliers (notably results 1.5 times or more away from the edge of the first or third quantiles), it is demonstrably useful in explaining how the interquartile range varies depending on the time of the year.



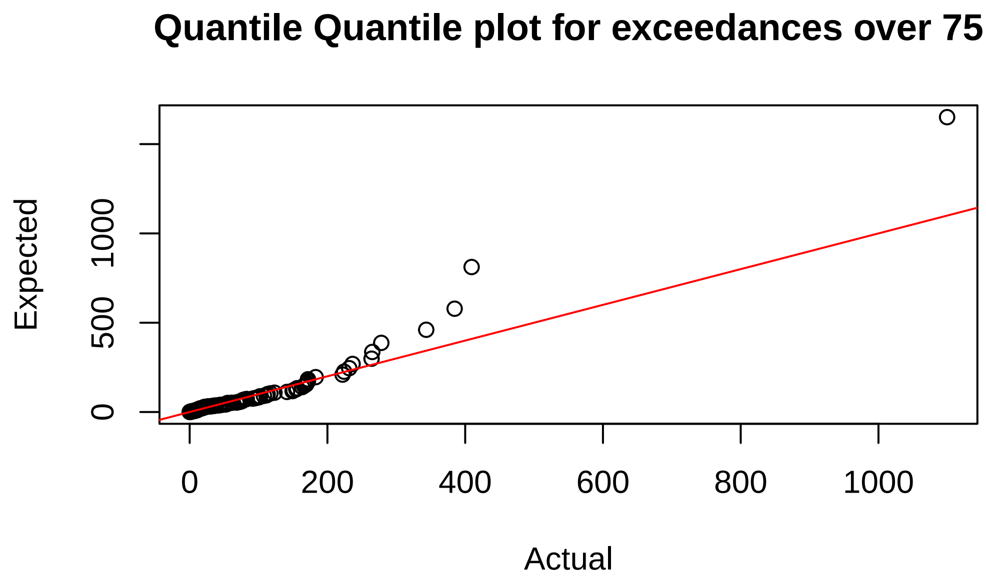
From the above, we can see the seasonality of the interquartile range – to provide one example, the entire interquartile range of March's riverflow is less than that experienced in May.

Similarly, from the rolling window of means plotted below, we can see that the riverflow over the course of years both has noise and that it slowly trends upwards.



To determine whether it is appropriate to model large data flows with the stated GPD, I constructed a quantile quantile plot contrasting the actual dataset, and the expected data. In this, the expected data was calculated using the inverse cumulative distribution.

From the graph, we can see that it is not appropriate to model large data flows with the GPD provided in the question. This is because we see a significant deviation of the plotted points from the $y = x$ line – the expected values don't align with the actual values when dealing with the largest river flows.



Code appendix

```
qgpd <- function (p, sigma=1, xi=0, u=0){
  len <- max(length(p), length(sigma), length(xi), length(u))
  p <- rep_len(p, len)
  sigma <- rep_len(sigma, len)
  xi <- rep_len(xi, len)
  u <- rep_len(u, len)

  safe_xi <- ifelse(xi==0, 1, xi)

  value <- ifelse(
    xi == 0,
    u - sigma * log1p(-p),
    u + ((1 - p)^(-xi) - 1) * sigma / safe_xi)
  maxp <- u - sigma/safe_xi

  # Ensuring our range is correct
  value <- ifelse(xi<0 & p==1, maxp, value)
  value <- ifelse(xi<0 & p<1 & maxp <= value, NaN, value)
```

```r
  # Handling forbidden inputs
  inadmissable <-
  (is.na(p))|(is.na(sigma))|(is.na(xi))|(is.na(u))|
  (!is.numeric(p))|(!is.numeric(sigma))|(!is.numeric(xi))|(!is.numeric(u))|
  (p < 0)|(p > 1)|(sigma <= 0)

  if (any(inadmissable, na.rm = TRUE)) warning("NaNs produced")
  value[inadmissable] <- NaN

  value

}

qgpd(0.5,2,-0.4,1.5)
qgpd(0.75,2,-0.4,1.5)
qgpd(0.99,2,-0.4,1.5)
# Reading our inputs:
gpd_parameters <- read.csv("gpd_parameters.csv")
gpd_samples <- read.csv("gpd_samples.csv")

# Mapping from our samples vector to our parameters table:
row <- match(gpd_samples$set_id, gpd_parameters$id)

# Ranking our samples:
class_ranking <- ave(gpd_samples$value, gpd_samples$set_id, FUN = rank)

# Using our samples to generate our p-values:
n <- gpd_parameters$size[row]
p <- (class_ranking - 0.5) / n

# Formatting (necessary for the legend):
f   <- factor(gpd_samples$set_id)
cols <- as.integer(f)
ptch <- 16
pt.cex <- 0.6

expected_q <- qgpd(
  p,
  gpd_parameters$sigma[row],
  gpd_parameters$xi[row],
  gpd_parameters$u[row]
)
```

```r
# Graphing our actual vs expected (using our p-values to generate our expected):
plot(
x = gpd_samples$value,
y = expected_q,
xlab = "Actuals",
ylab = "Expected",
col = cols,
pch = ptch,
cex = pt.cex
)

# Adding our y=x line:
abline(0,1,col="red")

# Adding our legend:
legend(
  "bottomright",
  title = "id",
  legend = levels(f),
  col = seq_along(levels(f)),
  pch = ptch,
  cex = pt.cex,
  bty = "o")

# Making six QQ plots (with one per id):
ids <- levels(f)

# Setting up the grid for our 6 smaller QQ plots:
op <- par(mfrow = c(2, 3))

# Building our plots:
for(id in ids){

  # Separating our various ids
  filt <- gpd_samples$set_id == id

  # Ensuring our plots don't appear too small when plotted
  limits <- range(c(gpd_samples$value[filt], expected_q[filt]), finite = TRUE)

  # Plotting the graph
  plot(
    x = gpd_samples$value[filt],
```

```r
    y = expected_q[filt],
    xlab = "Actuals",
    ylab = "Expected",
    main = paste("id:", id),
    pch  = ptch,
    cex  = pt.cex,
    xlim = limits,
    ylim = limits,
    asp  = 1
  )

  # Fitting a y=x line
  abline(0, 1, col = "red")

  # Adding a line of best fit through our plotted samples
  bestfit <- lm(expected_q[filt] ~ gpd_samples$value[filt])
  abline(bestfit, col = "blue", lty = 2)

  # Annotating our figure
  mtext(
    sprintf(
      "slope=%.2f,
      R^2=%.2f",
      coef(bestfit)[2],
      summary(bestfit)$r.squared),
      side = 3,
      adj = 1,
      cex = 0.6)
  }

# Reverting from grids back to normal:
par(op)

cor_results <- tapply(
  seq_along(expected_q),
  gpd_samples$set_id,
  function(filt) {
    cor(gpd_samples$value[filt], expected_q[filt])
  }
)

# Displaying correlations by id:
```

```
cor_results

# Reading our inputs:
riverflow <- read.csv("riverflow_2015_2024.csv")

# Formating dates:
riverflow$date <- as.Date(riverflow$date, format = "%Y-%m-%d")

# Time Series of flow data:
plot(
x = riverflow$date,
y = riverflow$flow,
type = "l",
col = "blue",
xlim = range(riverflow$date, na.rm = TRUE),
ylim = range(riverflow$flow, na.rm = TRUE),
xlab = "Year",
ylab = "River Flow (cumecs)",
main = "River Flow Time Series (2015-2024)"
)

# Time Series of baseline mean:
lines(
x = riverflow$date,
y = riverflow$baseline_mean,
col = "red",
lty = 2,
lwd = 2)

# Legend:
legend(
"topright",
legend = c("Flow", "Baseline mean"),
col = c("blue", "red"),
lty = c(1, 2),
lwd = c(1, 2),
bty = "o")


# Time series plot of flow minus baseline mean.
plot(
x = riverflow$date,
```

```
y = riverflow$flow - riverflow$baseline_mean,
type = "l",
xlab = "Date",
ylab = "Flow - baseline (cumecs)",
main = "Deviation from baseline"
)
abline(h = 0, lty = 3)



riverflow$month <- factor(format(riverflow$date, "%b"), levels = month.abb)

# Box and Whisker plot:
boxplot(
flow ~ month,
data = riverflow,
ylab = "River Flow (cumecs)",
xlab = "Month",
main = "Monthly distribution of flow",
notch = TRUE,
outline = FALSE)
abline(h = mean(riverflow$flow), col = "red", lwd = 2, lty = 2)
legend("topleft", legend = "Mean", col = "red", lwd = 2, lty = 2, bty = "n")
window <- 30
row_number <- nrow(riverflow)
rolling_mean <- rep(NA, row_number)

for(i in seq_len(row_number)){
  rolling_mean[i] <- mean(riverflow$flow[max(1,i-floor(window/2)):min(row_number,i+floor(winc
}

# Plotting the rolling mean
plot(
  x=riverflow$date,
  y=rolling_mean,
  type="l",
  col="blue",
  xlab="Year",
  ylab="River Flow (cumecs)",
  main="30-day river flow rolling mean")

trend <- lm(rolling_mean ~ as.numeric(riverflow$date))
```

```r
# Overall mean line and fitted trend line
abline(h = mean(riverflow$flow, na.rm = TRUE), col = "red", lty = 2, lwd = 2)
lines(riverflow$date, predict(trend), col = "darkgreen", lwd = 2, lty = 3)

# Legend
legend("topright",
       legend = c("Rolling mean", "Overall mean", "Linear trend"),
       col = c("blue", "red", "darkgreen"),
       lty = c(1, 2, 3),
       lwd = 2,
       bty = "n",
       cex = 0.7,
       y.intersp = 0.7)
# Set up, so that we can use the formula from question 2:
over75 <- riverflow$flow[riverflow$flow > 75]
p <- (rank(over75)-0.5)/length(over75)

# Quantile Quantile Plot comparing the actual versus expected data:
plot(
x = sort(over75)-75,
y = sort(qgpd(p,29.7,0.62,0)),
xlab = "Actual",
ylab = "Expected",
main = "Quantile Quantile plot for exceedances over 75")
abline(0, 1, col = "red")
```

References