

Data Analysis and Statistical Simulation using R

Dr. Kapil Kumar

Department of Statistics
Chaudhary Charan Singh University, Meerut, India
kapilstats@gmail.com

One-Week Faculty Development Programme / Short Term Course on *Data Analysis
using Open Source Tools* (22–27 April 2024)

Departments of ICT and Mathematics, Chaudhary Bansi Lal University, Bhiwani

23 April 2024 · Online

Plan of the session

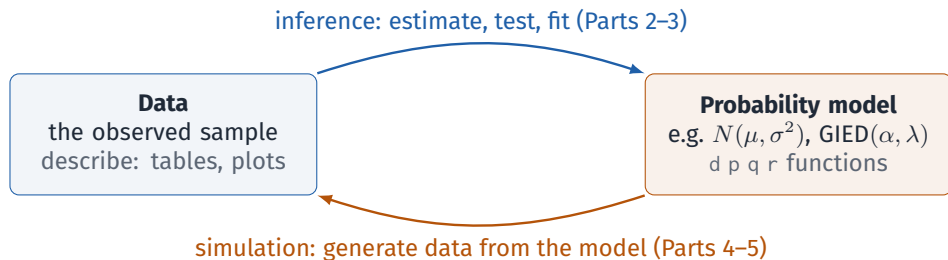
1. Getting started
2. Exploring a district data set
3. Tests and a regression model
4. Simulating from probability distributions
5. Inverse transform and Monte Carlo studies

By the end you can check and explore a real data file, test and model it with diagnostics, generate random numbers from any distribution, simulate a sampling distribution, fit a lifetime model by maximum likelihood and check it, and run a small Monte Carlo study.

PART 1

Getting started

Data analysis and simulation: two directions



Why both Simulation shows what a model implies when the mathematics is hard, and it checks the methods we use for inference: sampling distributions, goodness of fit, size of a test.

Working in a project; packages used

- Install R from CRAN, then RStudio Desktop (posit.co).
- *File* → *New Project*: the project folder becomes the working directory, so "data/data21.csv" works on every computer. No `setwd` with a path such as `D:/...` is needed.
- **lmtest**: `bptest()`; **car**: `vif()`; **goftest**: `ad.test()` (Anderson–Darling).

```
# install once:
# install.packages(c("lmtest", "car",
#                    "goftest"))
library(lmtest)
library(car)
library(goftest)

col_main <- "#1F5FA8" # plot colours
col_alt  <- "#B45309"
```

PART 2

Exploring district data

A district-level crop data set

```
d <- read.csv("data/data21.csv")
dim(d)
head(d[, c("Dist", "Yield", "NetArea", "Netirrig", "Rainfall", "Fert", "HH_SIZE")], 3)

## [1] 58 10
##   Dist Yield NetArea Netirrig Rainfall   Fert HH_SIZE
## 1    1 19575  274837   255465    1077 226891    6.35
## 2    2 23483  326387   323474     767 342817    6.70
## 3    3 19442  335831   310080    1284 208480    6.49
```

About these data Illustrative data supplied with the lecture: crop, units and source are not recorded. Further columns: Croppedarea, GrossIrrigated, MARG_HH_F.

Plausibility checks before any analysis

```
c(missing = sum(is.na(d)), duplicated = sum(duplicated(d)))  
with(d, c(net_le_gross = all(NetArea <= Croppedarea),  
          irr_le_net    = all(Netirrig <= NetArea),  
          girr_le_gross = all(GrossIrrigated <= Croppedarea),  
          irr_le_girr   = all(Netirrig <= GrossIrrigated)))  
  
##      missing duplicated  
##           0           0  
## net_le_gross  irr_le_net girr_le_gross  irr_le_girr  
##           TRUE           TRUE           TRUE           TRUE
```

Reading it No missing values or duplicated rows, and all four logical constraints between the area variables hold in every district.

New variables: ratios that compare districts fairly

```
d$irr_pct <- 100 * d$Netirrig / d$NetArea      # % of net sown area irrigated
d$crop_int <- 100 * d$Croppedarea / d$NetArea  # cropping intensity, %
d$fert_ha <- d$Fert / d$Croppedarea           # fertiliser per unit cropped area
round(sapply(d[, c("irr_pct", "crop_int", "fert_ha")], quantile,
             probs = c(0.1, 0.5, 0.9)), 2)

##      irr_pct crop_int fert_ha
## 10%   62.68   139.4   0.27
## 50%   86.05   156.1   0.42
## 90%   94.64   174.6   0.64
```

Why ratios Totals mostly measure district size. Cropping intensity 150% = half the net area is sown twice a year. The median district irrigates 86% of its net area.

Summaries with `apply()`, `sapply()` and our own function

```
vars <- c("Yield", "Rainfall", "irr_pct",  
         "fert_ha", "HH_SIZE")  
cv <- function(x) 100 * sd(x) / mean(x)  
round(apply(d[, vars], 2, median), 2)  
round(sapply(d[, vars], cv), 1)      # CV in %  
  
##      Yield Rainfall  irr_pct  fert_ha  HH_SIZE  
## 15412.50   906.50    86.05    0.42    6.50  
##      Yield Rainfall  irr_pct  fert_ha  HH_SIZE  
##      29.5     39.1     20.3     39.0     7.5
```

Reading it Rainfall (CV 39%) and fertiliser per unit area (39%) vary most between districts; household size varies least (8%).

`apply()` works over the columns of a matrix or data frame;
`sapply()` over the elements of a list (a data frame is a list of columns).

Control flow: classifying districts by rainfall

```
r <- d$Rainfall
rain_class <- character(length(r))
for (i in seq_along(r)) {
  if (r[i] < 750) rain_class[i] <- "low"
  else if (r[i] < 1050) rain_class[i] <- "medium"
  else rain_class[i] <- "high"
}
table(rain_class)

## rain_class
##      high      low medium
##       20       20      18
```

```
d$rain <- cut(d$Rainfall,
  breaks = c(0, 750, 1050, Inf),
  labels = c("low", "medium", "high"),
  right = FALSE)
table(d$rain)
all(d$rain == rain_class) # same?

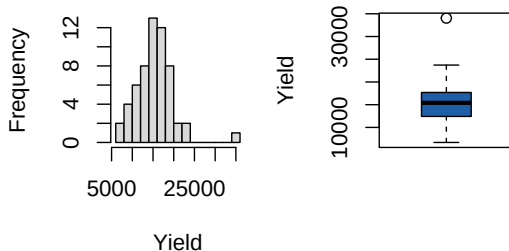
##
##      low medium      high
##       20      18       20
## [1] TRUE
```

`cut()`: the same in one vectorised call.

The distribution of yield

```
quantile(d$Yield)
par(mfrow = c(1, 2), mar = c(4, 4, 1, 1))
hist(d$Yield, breaks = 15, main = "",
     col = "grey85", xlab = "Yield")
boxplot(d$Yield, col = col_main,
        ylab = "Yield")
d$Dist[d$Yield > 30000] # the outlier

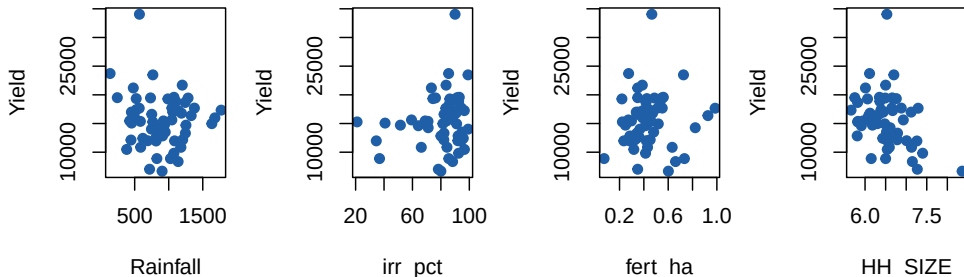
##      0%    25%    50%    75%   100%
##  6693 12494 15412 17685 34050
## [1] 24
```



Reading it Roughly symmetric around 15,412, except district 24 at 34,050, twice the median.

How is yield related to the inputs?

```
par(mfrow = c(1, 4), mar = c(4, 4, 1, 1), cex = 0.9) # one panel per input
for (v in vars[-1]) plot(d[[v]], d$Yield, xlab = v, ylab = "Yield", pch = 19, col = col_main)
```



Correlations with yield: rainfall -0.11, irrigation 0.17, fertiliser 0.08, household size -0.37.

Reading the exploration

What the data show No scatter plot shows a strong pattern: yield is only weakly correlated with each input. The strongest correlation is with household size ($r = -0.37$); irrigation ($r = 0.17$) and rainfall ($r = -0.11$) are close to zero.

One unusual district District 24 has by far the highest yield; it may drive any model we fit.

Questions for Part 3 Do yields differ between rainfall classes? Which inputs predict yield when considered together, and how much does district 24 matter?

Keep in mind District-level correlations describe districts, not individual farms.

PART 3

Tests and a regression model

Association in a small table

```
sugar <- read.csv("data/sugar_patients_data.csv")
tab <- table(age = sugar$age, diabetes = sugar$diabetes)
tab
round(chisq.test(tab)$expected, 1) # two cells < 5
```

```
##          diabetes
## age      no yes
##  old      2   6
##  young  11   1
##          diabetes
## age      no yes
##  old    5.2 2.8
##  young  7.8 4.2
```

The data 20 hypothetical patients: 6 of 8 older and 1 of 12 younger patients have diabetes.

Problem Two expected counts are below 5, so the χ^2 distribution is a poor guide to the p -value (R warns). Simulation offers a way out.

A Monte Carlo p -value for the same table

```
set.seed(2024)
p_mc <- chisq.test(tab, B = 10000,
  simulate.p.value = TRUE)$p.value
p_mc
fisher.test(tab)$p.value

## [1] 0.004
## [1] 0.004438
```

Simulate B tables with the observed margins under H_0 ; p = share of simulated X^2 at least as large as the observed one.

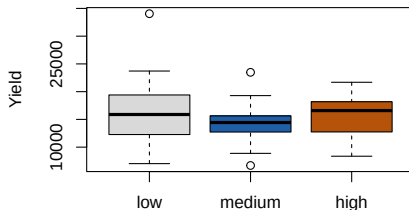
Reading it The simulated $p = 0.004$ and Fisher's exact $p = 0.0044$ agree: in these hypothetical data, diabetes is associated with age group.

A first taste of Part 4: when the theory is doubtful, simulate the null distribution.

Does yield differ between rainfall classes?

```
fit_aov <- aov(Yield ~ rain, data = d)
summary(fit_aov)
kruskal.test(Yield ~ rain, data = d)$p.value
round(tapply(d$Yield, d$rain, median))
```

##	Df	Sum Sq	Mean Sq	F value	Pr(>F)
## rain	2	51905729	25952865	1.26	0.29
## Residuals	55	1136259169	20659258		
## [1]	0.2768				
##	low	medium	high		
##	15884	14440	16590		



Reading it F -test $p = 0.29$,
Kruskal-Wallis $p = 0.28$: no
evidence that mean yield
differs between the classes.

A multiple regression for yield

```
fit <- lm(Yield ~ irr_pct + fert_ha + Rainfall + HH_SIZE, data = d)
coef(summary(fit))
```

##	Estimate	Std. Error	t value	Pr(> t)
## (Intercept)	38813.88	7746.66	5.010	0.000006402
## irr_pct	58.08	35.26	1.647	0.105387070
## fert_ha	4347.57	3593.82	1.210	0.231748905
## Rainfall	-2.30	1.61	-1.429	0.158924124
## HH_SIZE	-4288.37	1173.99	-3.653	0.000595086

$$\text{Yield}_i = \beta_0 + \beta_1 \text{irr}_i + \beta_2 \text{fert}_i + \beta_3 \text{rain}_i + \beta_4 \text{hhsz}_i + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2) \text{ independent}$$

Reading the regression

```
round(vif(fit), 2)          # car
round(confint(fit)["HH_SIZE", ])
fs <- summary(fit)$fstatistic
pf(fs[1], fs[2], fs[3], lower.tail = FALSE)

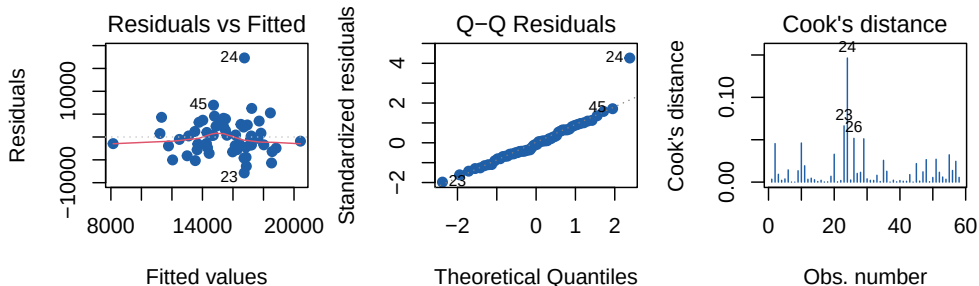
## irr_pct  fert_ha Rainfall  HH_SIZE
##      1.13      1.23      1.05      1.10
## 2.5 % 97.5 %
## -6643 -1934
##      value
## 0.005698
```

In plain words The inputs explain 24% of the variation (overall F -test $p = 0.006$). With the others fixed, one more person per household goes with a mean yield about 4,288 lower. The other inputs are not significant.

Careful VIFs all below 2: no collinearity. An association, not a causal effect.

Diagnostic plots

```
par(mfrow = c(1, 3), mar = c(4, 4, 2, 1), cex = 0.9)
plot(fit, which = c(1, 2, 4), pch = 19, col = col_main) # residuals, Q-Q, Cook's distance
```



One point lies far above the rest: the Q-Q plot bends at the top; Cook's D singles out 24.

Tests on the residuals and influence

```
bptest(fit)$p.value           # constant variance
shapiro.test(residuals(fit))$p.value # normality
r <- rstudent(fit)
round(sort(abs(r), decreasing = TRUE)[1:3], 2)
cd <- cooks.distance(fit)
round(sort(cd, decreasing = TRUE)[1:3], 3)

##      BP
## 0.5863
## [1] 0.0009086
##   24   23   45
## 5.21 2.04 1.75
##   24   23   26
## 0.146 0.066 0.052
```

Reading it Constant variance is plausible ($p = 0.59$), but the residuals are not normal ($p = 0.0009$). District 24 has a studentized residual of 5.2 and the largest Cook's distance (above the usual $4/n = 0.069$).

Sensitivity: refit without district 24

```
fit24 <- update(fit, data = subset(d, Dist != 24))
round(cbind(all = coef(fit),
             without_24 = coef(fit24)), 1)
round(c(all = summary(fit)$r.squared,
        without_24 = summary(fit24)$r.squared), 3)

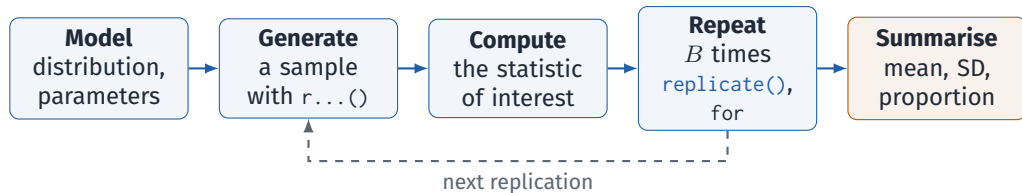
##           all without_24
## (Intercept) 38813.9    37992.7
## irr_pct      58.1      48.6
## fert_ha     4347.6    3866.3
## Rainfall     -2.3     -1.4
## HH_SIZE     -4288.4   -4183.5
##           all without_24
##           0.237    0.289
```

Reading it The household-size effect is almost unchanged and no other input becomes significant; the residuals now look normal (Shapiro-Wilk $p = 0.73$). Report both fits.

PART 4

Simulating from distributions

Why simulate, and how a simulation is built



- See sampling distributions that theory gives only approximately.
- Check estimators, tests and goodness-of-fit p -values.

Two habits Fix the seed with `set.seed()`; report the Monte Carlo error of every simulated number.

Four functions for every distribution

Prefix	Returns
d	density $f(x)$ or $P(X = x)$
p	cdf $F(q) = P(X \leq q)$
q	quantile $F^{-1}(p)$
r	random sample of size n

Example: `dnorm()`, `pnorm()`, `qnorm()`, `rnorm()`.

Distribution	Suffix
binomial, Poisson, geometric	binom, pois, geom
hypergeometric, neg. binomial	hyper, nbinom
normal, log-normal, uniform	norm, lnorm, unif
exponential, gamma, beta	exp, gamma, beta
Weibull, Cauchy	weibull, cauchy
t , χ^2 , F	t, chisq, f

Pattern prefix + suffix: `pbinom()` is the binomial cdf, `qchisq()` a chi-square quantile, `rweibull()` a Weibull sample. Our own distributions can follow the same pattern.

The four functions in action

```
dbinom(3, size = 10, prob = 0.2)    # P(X = 3)
pbinom(3, size = 10, prob = 0.2)    # P(X <= 3)
qnorm(0.975)                        # 97.5% point
pnorm(1.96)                         # and back
qt(0.975, df = 10)                  # t critical value
pexp(1, rate = 2)                   # P(X <= 1)

## [1] 0.2013
## [1] 0.8791
## [1] 1.96
## [1] 0.975
## [1] 2.228
## [1] 0.8647
```

Reading it `qnorm()` and `pnorm()` are inverses: the 97.5% point 1.96 gives back probability 0.975. With 10 df the t critical value (2.23) is larger than 1.96: wider intervals for small samples.

These replace printed statistical tables.

Random samples and the seed

```
set.seed(23)
x1 <- rbinom(n = 50, size = 20, prob = 0.8)
mean(x1)           # theory: 20 x 0.8 = 16
x2 <- rexp(n = 50, rate = 2)
1 / mean(x2)       # estimates the rate 2
set.seed(1); runif(3)
set.seed(1); runif(3) # same seed, same numbers

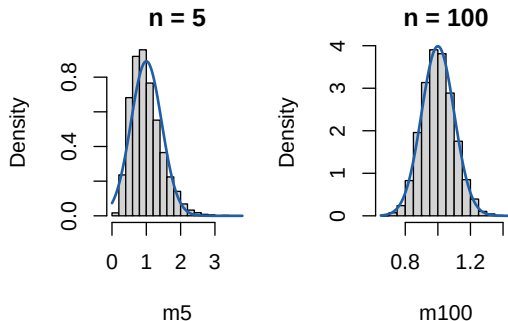
## [1] 15.64
## [1] 2.117
## [1] 0.2655 0.3721 0.5729
## [1] 0.2655 0.3721 0.5729
```

Reading it The sample mean 15.64 is close to $np = 16$, and $1/\bar{x}$ estimates the exponential rate. Each sample is random, but a fixed seed reproduces it exactly.

R's generator (Mersenne-Twister) is deterministic but behaves like independent $U(0, 1)$ draws.

Sampling distribution of the mean, by simulation

```
sim1 <- function(n) mean(rexp(n)) # Exp(1)
set.seed(2024)
m5 <- replicate(10000, sim1(5))
m20 <- replicate(10000, sim1(20))
m100 <- replicate(10000, sim1(100))
par(mfrow = c(1, 2), mar = c(4, 4, 2, 1))
hist(m5, freq = FALSE, main = "n = 5")
curve(dnorm(x, 1, 1 / sqrt(5)), add = TRUE,
      col = col_main, lwd = 2)
hist(m100, freq = FALSE, main = "n = 100")
curve(dnorm(x, 1, 1 / sqrt(100)), add = TRUE,
      col = col_main, lwd = 2)
```



Blue: $N(1, 1/n)$. Skewed at $n = 5$, close to normal at $n = 100$.

Reading it: the central limit theorem at work

```
sims <- list(n5 = m5, n20 = m20, n100 = m100)
round(rbind(mean = sapply(sims, mean),
                  sd = sapply(sims, sd),
                  sd_theory = 1 / sqrt(c(5, 20, 100))), 4)

##              n5      n20      n100
## mean         0.9997  1.0024  1.0007
## sd           0.4464  0.2234  0.0989
## sd_theory    0.4472  0.2236  0.1000
```

$X_i \sim \text{Exp}(1)$: $E(\bar{X}) = 1$, $\text{SD}(\bar{X}) = 1/\sqrt{n}$;
 $\bar{X} \approx N(1, 1/n)$ for large n .

Reading it The simulated means and SDs match theory at every n . The *shape* is the slow part: clearly skewed at $n = 5$, close to normal by $n = 100$.

With $B = 10,000$ the Monte Carlo error of each simulated mean is about $\text{SD}/100$.

PART 5

Inverse transform and Monte Carlo

The inverse-transform method

If $U \sim U(0, 1)$ and F is a continuous cdf, then $X = F^{-1}(U)$ has cdf F :
 $P(X \leq x) = P\{U \leq F(x)\} = F(x).$

- Needs a closed-form F^{-1} ; otherwise use other methods (e.g. acceptance-rejection).
- Exponential: $F(x) = 1 - e^{-\lambda x}$ gives $x = -\log(1 - u)/\lambda$.

```
set.seed(5)
u <- runif(4)
-log(1 - u) / 2      # by hand, rate 2
qexp(u, rate = 2)    # R's quantile function

## [1] 0.1117 0.5779 1.2437 0.1673
## [1] 0.1117 0.5779 1.2437 0.1673
```

Same numbers: `qexp()` is F^{-1} .

Example: the generalized inverted exponential distribution

For $x, \alpha, \lambda > 0$:

$$f(x) = \frac{\alpha\lambda}{x^2} e^{-\lambda/x} (1 - e^{-\lambda/x})^{\alpha-1}$$

$$F(x) = 1 - (1 - e^{-\lambda/x})^\alpha$$

$$F^{-1}(u) = \frac{-\lambda}{\log\{1 - (1 - u)^{1/\alpha}\}}$$

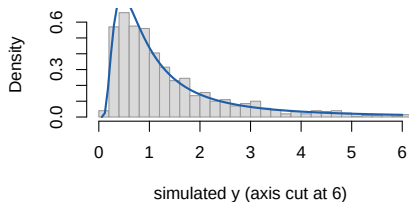
```
dgied <- function(x, alpha, lambda)
  alpha * lambda / x^2 * exp(-lambda / x) *
  (1 - exp(-lambda / x))^(alpha - 1)
pgied <- function(q, alpha, lambda)
  1 - (1 - exp(-lambda / q))^alpha
qgied <- function(p, alpha, lambda)
  -lambda / log(1 - (1 - p)^(1 / alpha))
rgied <- function(n, alpha, lambda)
  qgied(runif(n), alpha, lambda) # inversion
```

A flexible lifetime model (GIED) used in reliability studies.

Simulating GIED data and checking them

```
set.seed(2024)
y_sim <- rgied(1000, alpha = 1.5, lambda = 1)
p3 <- c(0.25, 0.5, 0.75)
rbind(sim = quantile(y_sim, p3),
      theory = qgied(p3, 1.5, 1))
c(KS = ks.test(y_sim, pgied, 1.5, 1)$p.value,
  AD = ad.test(y_sim, "pgied", 1.5, 1)$p.value)
```

```
##           25%   50%   75%
## sim      0.5951 1.040 2.011
## theory 0.5728 1.006 1.978
##      KS      AD
## 0.3141 0.4466
```



Reading it Quartiles match theory, and with the true parameters neither KS nor Anderson–Darling rejects: the generator works.

Fitting the GIED to fibre-strength data

```
y <- read.csv("data/fibre_strength.csv")$strength
n <- length(y)
negll <- function(theta, x) { # theta: log scale
  a <- exp(theta[1]); l <- exp(theta[2])
  -sum(log(a * l) - 2 * log(x) - l / x +
       (a - 1) * log(1 - exp(-l / x)))
}
mle <- nlm(negll, p = log(c(16, 5)), x = y)
est <- exp(mle$estimate) # back to alpha, lambda
a_hat <- est[1]; l_hat <- est[2]
c(alpha = a_hat, lambda = l_hat, logLik = -mle$minimum)

## alpha lambda logLik
## 205.88 13.88 -49.17
```

$$\begin{aligned}\ell(\alpha, \lambda) = & n \log(\alpha \lambda) \\ & - 2 \sum \log x_i \\ & - \lambda \sum 1/x_i \\ & + (\alpha - 1) \\ & \times \sum \log(1 - e^{-\lambda/x_i})\end{aligned}$$

`nlm()` minimises $-\ell$; using $\log \alpha$ and $\log \lambda$ keeps both parameters positive. Data: 69 fibre strengths (GPa) used in the lecture.

Is the GIED a good fit?

```
D_obs <- ks.test(y, pgied, a_hat, l_hat)$statistic
c(D_obs,
  KS_p = ks.test(y, pgied, a_hat, l_hat)$p.value,
  AD_p = ad.test(y, "pgied", alpha = a_hat,
    lambda = l_hat)$p.value)

##          D      KS_p      AD_p
## 0.04141 0.99978 0.99201
```

Reading it The largest gap between the empirical and fitted cdf is only $D = 0.041$; both p -values are large.

But these p -values are too large

They assume the parameters were known in advance; here they were estimated from the same data, so the fit looks better than it is. Three tied values also make the KS p approximate. Remedy: simulate the null distribution of D .

A parametric bootstrap p -value

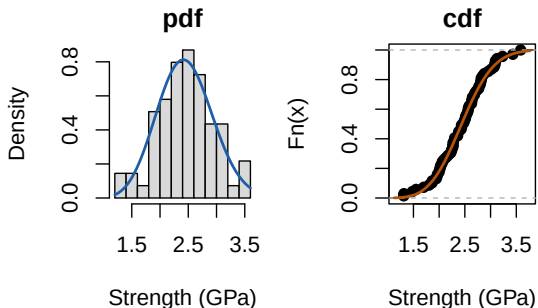
```
set.seed(2024)
B <- 500
D_boot <- numeric(B)
for (b in 1:B) {
  yb <- rgied(n, a_hat, l_hat)      # data from the fit
  fb <- nlm(negll, p = mle$estimate, x = yb,
            stepmax = 1)           # re-estimate
  D_boot[b] <- ks.test(yb, pgied, exp(fb$estimate[1]),
                      exp(fb$estimate[2]))$statistic
}
mean(D_boot >= D_obs)              # bootstrap p

## [1] 0.996
```

Reading it Each bootstrap sample repeats the whole procedure, estimation included. The observed $D = 0.041$ is far below the simulated 5% critical value 0.11; bootstrap $p = 0.996$. The GIED fits these data well.

Fitted versus empirical: density and cdf

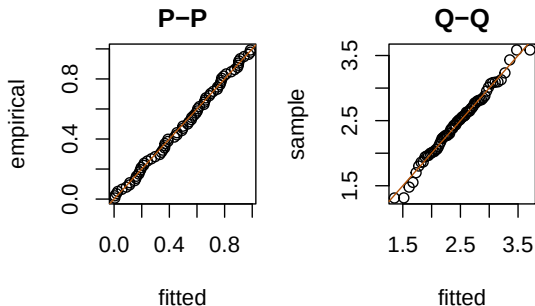
```
par(mfrow = c(1, 2), mar = c(4, 4, 2, 1))
hist(y, breaks = 10, freq = FALSE,
     main = "pdf", xlab = "Strength (GPa)",
     col = "grey85")
curve(dgied(x, a_hat, l_hat), add = TRUE,
      col = col_main, lwd = 2)
plot(ecdf(y), main = "cdf",
     xlab = "Strength (GPa)")
curve(pgied(x, a_hat, l_hat), add = TRUE,
      col = col_alt, lwd = 2)
```



The fitted density follows the histogram; the fitted cdf tracks the empirical steps.

Fitted versus empirical: P-P and Q-Q plots

```
pp <- ((1:n) - 0.5) / n # plotting positions
par(mfrow = c(1, 2), mar = c(4, 4, 2, 1))
plot(pgied(sort(y), a_hat, l_hat), pp,
     main = "P-P", xlab = "fitted",
     ylab = "empirical")
abline(0, 1, col = col_alt)
plot(qgied(pp, a_hat, l_hat), sort(y),
     main = "Q-Q", xlab = "fitted",
     ylab = "sample")
abline(0, 1, col = col_alt)
```



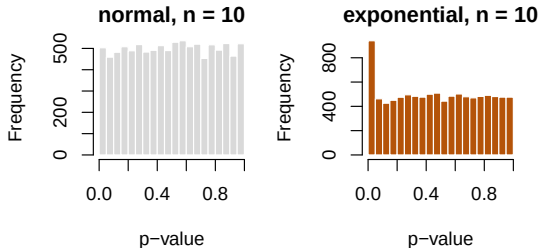
Points hug the diagonal in both plots: the fit is good in the centre (P-P) and in the tails (Q-Q).

A Monte Carlo study: is the t -test's size right?

```
set.seed(2024)
B <- 10000                                # H0 is true in every case below
p_norm10 <- replicate(B, t.test(rnorm(10), mu = 0)$p.value)
p_exp10  <- replicate(B, t.test(rexp(10), mu = 1)$p.value)
p_exp30  <- replicate(B, t.test(rexp(30), mu = 1)$p.value)
p_exp100 <- replicate(B, t.test(rexp(100), mu = 1)$p.value)
size <- c(normal_10 = mean(p_norm10 < 0.05), exp_10 = mean(p_exp10 < 0.05),
          exp_30 = mean(p_exp30 < 0.05), exp_100 = mean(p_exp100 < 0.05))
size
round(sqrt(0.05 * 0.95 / B), 4)           # Monte Carlo standard error at 0.05

## normal_10    exp_10    exp_30    exp_100
##    0.0503    0.0940    0.0766    0.0546
## [1] 0.0022
```

Reading the Monte Carlo study



Size $\hat{\alpha} = \frac{1}{B} \sum_b I(p_b < 0.05)$, Monte Carlo SE $\sqrt{\hat{\alpha}(1 - \hat{\alpha})/B}$. Under H_0 valid p -values are uniform.

Reading it Normal data: size 0.0503, on target. Exponential data: 0.094 at $n = 10$ and 0.0766 at $n = 30$, many Monte Carlo SEs (about 0.002) above 0.05; 0.0546 at $n = 100$.

Lesson For small, skewed samples the t -test rejects a true H_0 too often; the CLT repairs this only slowly.

Key take-aways

1. **Check before you analyse:** missing values, duplicates and logical constraints; build ratios that compare units fairly.
2. **Read models with their diagnostics:** a significant coefficient, a low R^2 , non-normal residuals and one influential district are all part of the story.
3. **d, p, q, r** give density, cdf, quantile and random numbers for every standard distribution; `set.seed()` makes runs repeatable.
4. **Inversion** turns uniform numbers into any distribution with a closed-form quantile function, as for the GIED.
5. **Simulation checks inference:** sampling distributions, bootstrap goodness-of-fit p -values, and the real size of a test, each with its Monte Carlo error.

References

- Abouammoh, A. M. and Alshingiti, A. M. (2009). Reliability estimation of generalized inverted exponential distribution. *Journal of Statistical Computation and Simulation*, 79(11), 1301–1315.
- Breusch, T. S. and Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica*, 47(5), 1287–1294.
- Fox, J. and Weisberg, S. (2019). *An R Companion to Applied Regression*, 3rd ed. Sage.
- Jones, O., Harden, S. and Crawley, M. J. (2022). *The R Book*, 3rd ed. Wiley.
- Rizzo, M. L. (2019). *Statistical Computing with R*, 2nd ed. CRC Press.
- Ross, S. M. (2013). *Simulation*, 5th ed. Academic Press.
- Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4), 591–611.
- R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna. <https://www.R-project.org>

Thank you

Questions and discussion