ACTL30007 - Omar Amin - Assignment 1

Omar

2025-06-14

R Markdown

```
library(readxl)
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

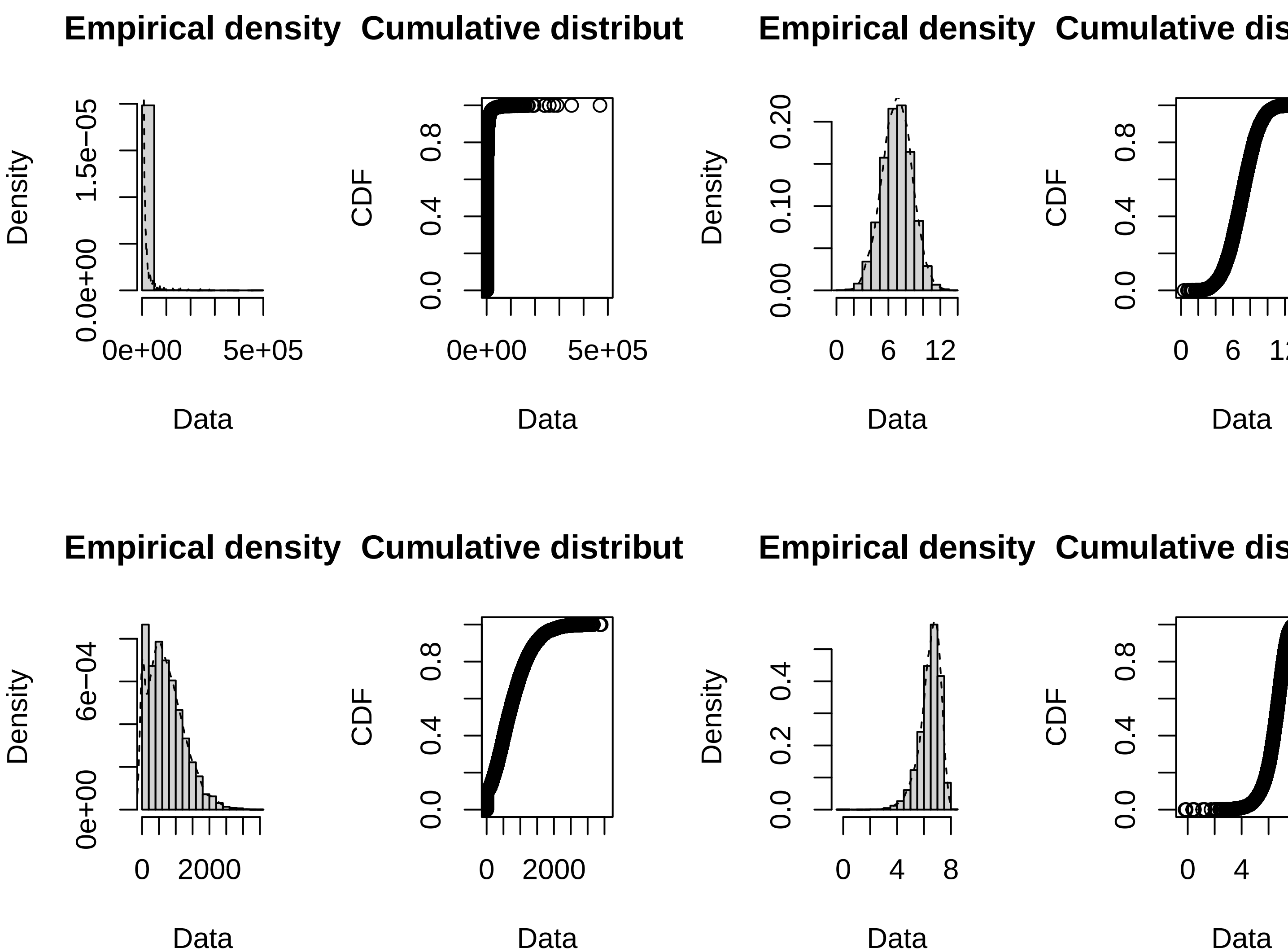
```
# Load data
df1 = read_excel("1463797.xlsx", sheet=1)
df2 = read_excel("1463797.xlsx", sheet=2)
df3 = read_excel("1463797.xlsx", sheet=3)

# Filter strictly positive claims
sev1 <- df1$claim_size[df1$claim_size > 0]
sev2 <- df2$claim_size[df2$claim_size > 0]
sev3 <- df3$claim_size[df3$claim_size > 0]

data <- list(df1, df2, df3)
```

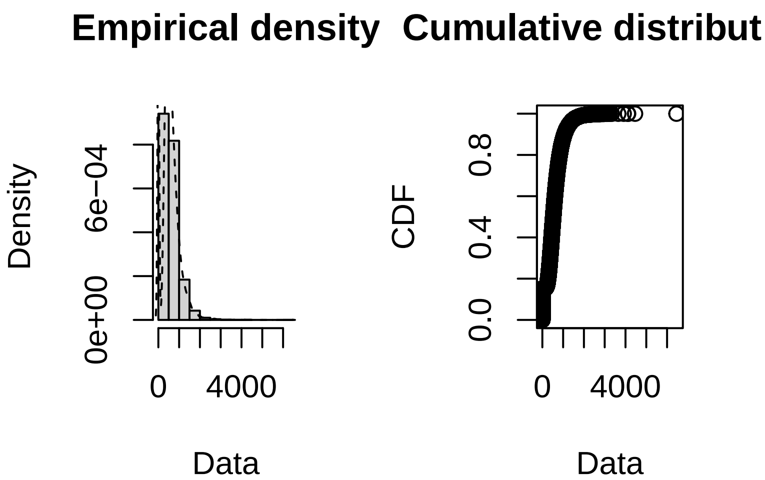
Including Plots

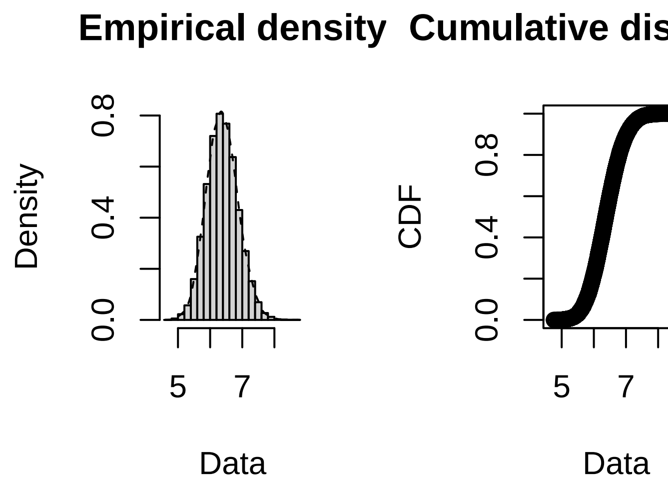
You can also embed plots, for example:

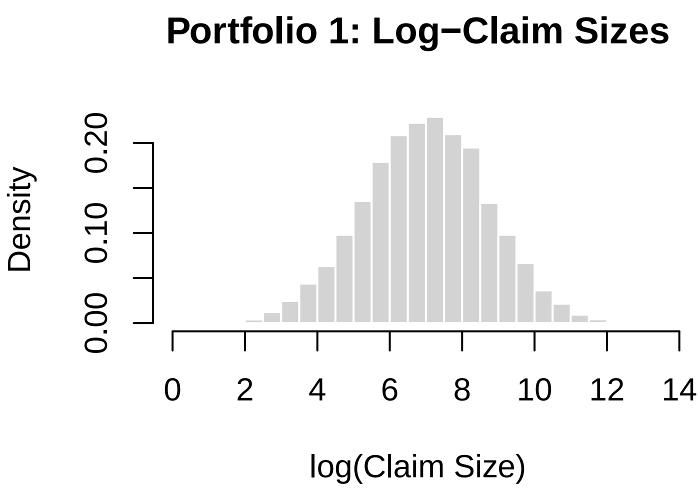
```
## Loading required package: MASS

##
## Attaching package: 'MASS'
```

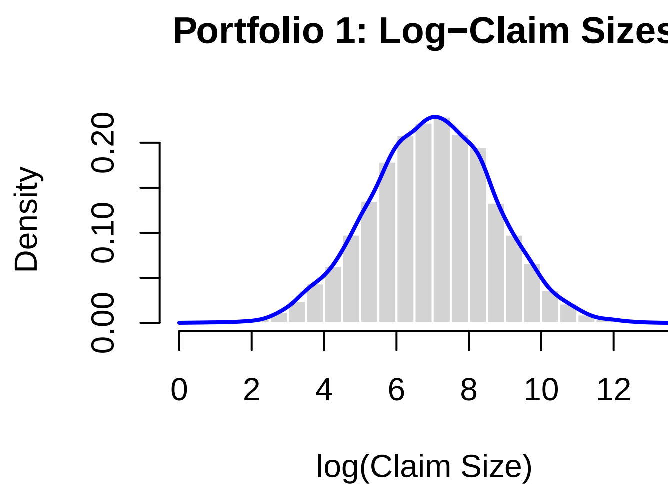
```
## The following object is masked from 'package:dplyr':
##
##     select

## Loading required package: survival

## Portfolio 1: 68.641%

## Portfolio 2: 90.432%

## Portfolio 3: 84.787%
```

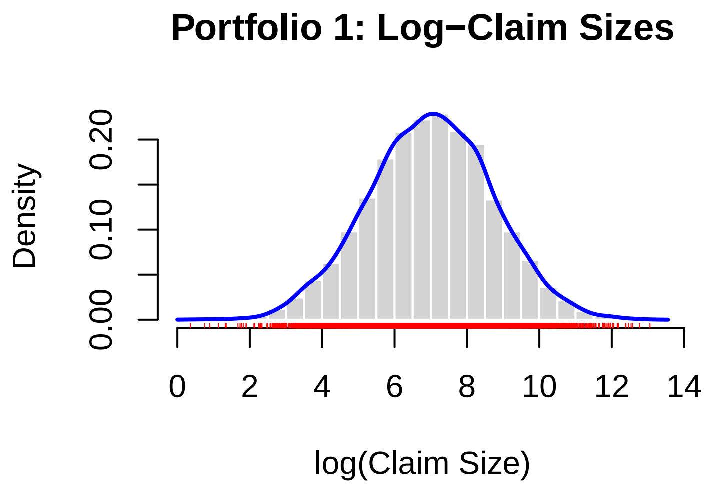



Empirical density
Cumulative distribut
Density
CDF
Data


Empirical density
Cumulative dis
Density
CDF
Data


Portfolio 1: Log−Claim Sizes
Density
log(Claim Size)


Portfolio 1: Log−Claim Sizes
Density
log(Claim Size)



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
summarize_portfolio <- function(df) {
  count_data <- df %>%
    group_by(id) %>%
    summarise(n_claims = ifelse(length(claim_size) == 1 & all(claim_size == 0), 0L, n()))

  zero_prop <- mean(count_data$n_claims == 0)
  list(
    zero_prop = zero_prop,
    mean = mean(count_data$n_claims),
    var = var(count_data$n_claims),
    data = count_data
  )
}

stats1 <- summarize_portfolio(df1)
stats2 <- summarize_portfolio(df2)
stats3 <- summarize_portfolio(df3)
stats1; stats2; stats3
```

```
## $zero_prop
## [1] 0.413
##
## $mean
## [1] 0.904
##
## $var
## [1] 0.9426783
##
## $data
## # A tibble: 10,000 x 2
##       id n_claims
##    <dbl>    <int>
```

```
##  1     1        0
##  2     2        2
##  3     3        2
##  4     4        0
##  5     5        0
##  6     6        1
##  7     7        0
##  8     8        1
##  9     9        0
## 10    10        0
## # i 9,990 more rows

## $zero_prop
## [1] 0.1692
##
## $mean
## [1] 1.5992
##
## $var
## [1] 1.299889
##
## $data
## # A tibble: 10,000 x 2
##       id n_claims
##    <dbl>    <int>
##  1     1        2
##  2     2        2
##  3     3        2
##  4     4        1
##  5     5        3
##  6     6        3
##  7     7        2
##  8     8        1
##  9     9        2
## 10    10        2
## # i 9,990 more rows

## $zero_prop
## [1] 0.2923
##
## $mean
## [1] 1.6291
##
## $var
## [1] 2.179151
##
## $data
## # A tibble: 10,000 x 2
##       id n_claims
##    <dbl>    <int>
##  1     1        2
##  2     2        0
##  3     3        4
##  4     4        3
```

```
## 5     5        6
## 6     6        0
## 7     7        0
## 8     8        1
## 9     9        0
## 10   10        0
## # i 9,990 more rows
```

```
cat("Index of Dispersions P1, P2, P3: ", c(stats1$var/stats1$mean, stats2$var/stats2$mean, stats3$var/s
```

```
## Index of Dispersions P1, P2, P3:  1.042786 0.8128373 1.337641
```

```
# Fit Poisson and (if appropriate) Negative Binomial for claim counts
library(MASS)
df1_counts <- df1 %>% group_by(id) %>% summarise(count = if_else(n()==1 & any(claim_size==0), 0L, as.in
df2_counts <- df2 %>% group_by(id) %>% summarise(count = if_else(n()==1 & any(claim_size==0), 0L, as.in
df3_counts <- df3 %>% group_by(id) %>% summarise(count = if_else(n()==1 & any(claim_size==0), 0L, as.in

# Frequency models (intercept-only, since no covariates)
model1_pois <- glm(count ~ 1, data = df1_counts, family = "poisson")
model2_pois <- glm(count ~ 1, data = df2_counts, family = "poisson")
model3_pois <- glm(count ~ 1, data = df3_counts, family = "poisson")
# Negative Binomial fit for portfolios showing overdispersion
model3_nb   <- glm.nb(count ~ 1, data = df3_counts)       # (only portfolio3 needs NB significantly)
coef(model3_nb); model3_nb$theta                          # view NB parameters (intercept log-mean, the
```

```
## (Intercept)
##   0.4880277
```

```
## [1] 3.862073
```

```
library(fitdistrplus)
library(actuar)   # for Pareto distribution functions
```

```
##
## Attaching package: 'actuar'
```

```
## The following objects are masked from 'package:stats':
##
##     sd, var
```

```
## The following object is masked from 'package:grDevices':
##
##     cm
```

```
sev1 <- df1$claim_size[df1$claim_size > 0]   # Portfolio1 claim sizes > 0
sev2 <- df2$claim_size[df2$claim_size > 0]
sev3 <- df3$claim_size[df3$claim_size > 0]

# Fit parametric severity distributions for Portfolio1 (heavy-tailed)
```

```r
fit1_gamma <- fitdist(sev1, "gamma", start=list(shape=1, scale=mean(sev1)))   # Gamma
fit1_lnorm <- fitdist(sev1, "lnorm")                                          # Lognormal
fit1_pareto<- fitdist(sev1, "pareto", start=list(shape=2, scale=500))         # Pareto

gofstat(list(fit1_gamma, fit1_lnorm, fit1_pareto), fitnames = c("Gamma","Lognormal","Pareto"))$aic
```

```
##     Gamma Lognormal    Pareto
##  164796.8  161740.5  161931.7
```

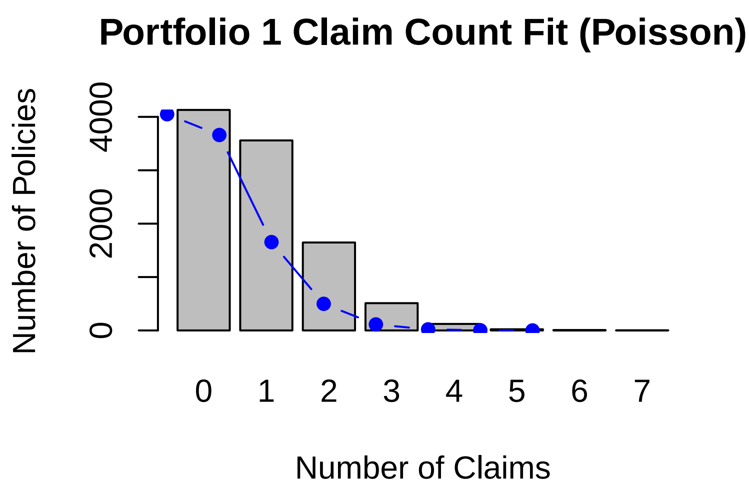
```r
fit1_gamma$estimate; fit1_lnorm$estimate; fit1_pareto$estimate              # view parameter estimates
```

```
##        shape        scale
##    0.4543333 9873.8802195
```

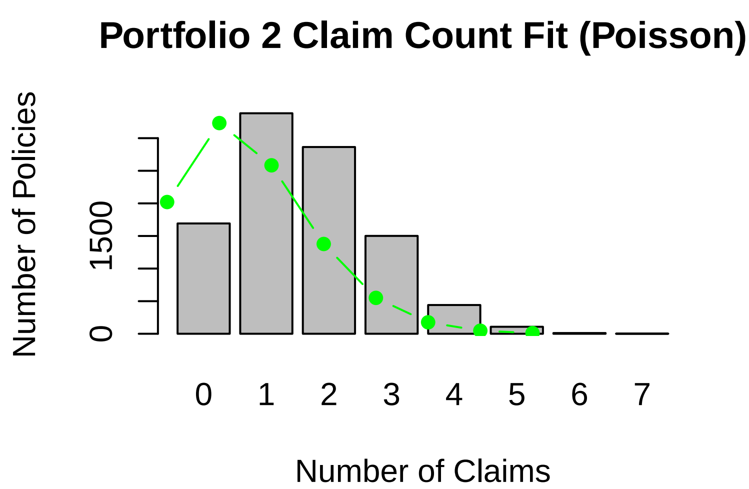
```
##  meanlog    sdlog
## 6.989846 1.710560
```

```
##       shape       scale
##    1.096714 1257.609778
```

```r
# --------- Frequency models for Portfolios 1 and 2 ---------
# Prepare claim counts per policy
counts_df <- function(df) {
  df %>%
    group_by(id) %>%
    summarise(n_claims = ifelse(length(claim_size) == 1 & all(claim_size == 0), 0L, n()))
}

counts1 <- counts_df(df1)
counts2 <- counts_df(df2)

# Portfolio 1
fit_pois1 <- glm(n_claims ~ 1, family = poisson(), data = counts1)
lambda1 <- exp(coef(fit_pois1))
expected1 <- dpois(0:max(counts1$n_claims), lambda1) * nrow(counts1)
obs1 <- table(counts1$n_claims)

# Portfolio 2
fit_pois2 <- glm(n_claims ~ 1, family = poisson(), data = counts2)
lambda2 <- exp(coef(fit_pois2))
expected2 <- dpois(0:max(counts2$n_claims), lambda2) * nrow(counts2)
obs2 <- table(counts2$n_claims)

# Plot for Portfolio 1
barplot(obs1, col = "gray", names.arg = names(obs1),
        main = "Portfolio 1 Claim Count Fit (Poisson)",
        xlab = "Number of Claims", ylab = "Number of Policies")
lines(0:max(counts1$n_claims), expected1, col = "blue", type = "b", pch = 16)
```

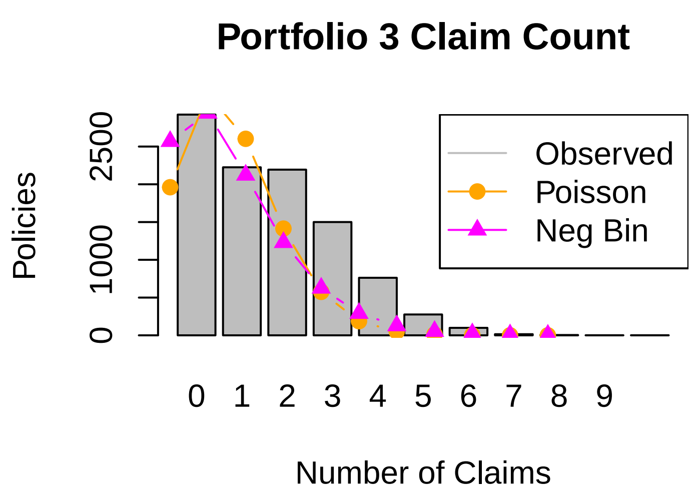



```r
# Plot for Portfolio 2
barplot(obs2, col = "gray", names.arg = names(obs2),
        main = "Portfolio 2 Claim Count Fit (Poisson)",
        xlab = "Number of Claims", ylab = "Number of Policies")
lines(0:max(counts2$n_claims), expected2, col = "green", type = "b", pch = 16)
```

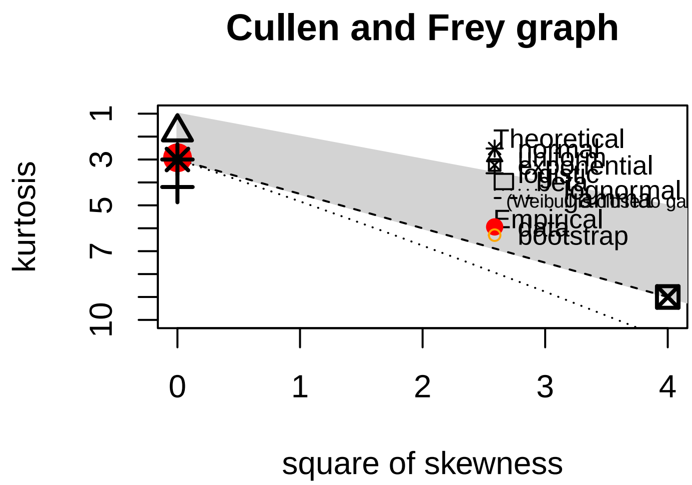



```r
# --------- Frequency models for Portfolio 3 ---------

# Fit Poisson and Negative Binomial
fit_pois3 <- glm(n_claims ~ 1, family = poisson(), data = stats3$data)
fit_nb3   <- glm.nb(n_claims ~ 1, data = stats3$data)

# Visualize model fit
obs_counts <- table(stats3$data$n_claims)
lambda3 <- exp(coef(fit_pois3))
```

```r
mu3 <- lambda3
theta3 <- fit_nb3$theta
expected_pois <- dpois(0:max(stats3$data$n_claims), lambda3) * nrow(stats3$data)
expected_nb   <- dnbinom(0:max(stats3$data$n_claims), mu = mu3, size = theta3) * nrow(stats3$data)

barplot(obs_counts, col = "gray", names.arg = names(obs_counts),
        main = "Portfolio 3 Claim Count", xlab = "Number of Claims", ylab = "Policies")
lines(0:max(stats3$data$n_claims), expected_pois, type = "b", col = "orange", pch = 19)
lines(0:max(stats3$data$n_claims), expected_nb,   type = "b", col = "magenta", pch = 17)
legend("topright", legend = c("Observed", "Poisson", "Neg Bin"),
       col = c("gray", "orange", "magenta"), pch = c(NA, 19, 17), lty = 1)
```

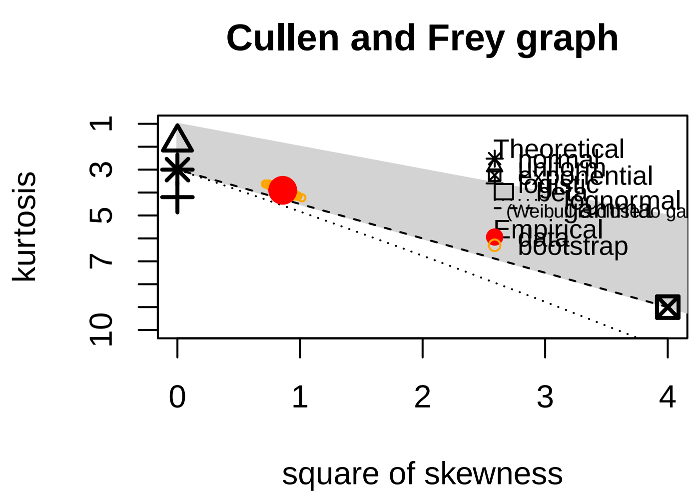



```r
library(fitdistrplus)

# Portfolio 1 - log-transformed claim sizes
sev1 <- df1$claim_size[df1$claim_size > 0]
descdist(log(sev1), boot = 1000)
```




```
## summary statistics
## ------
## min:  0.3577616   max:  13.05511
## median:  7.012698
## mean:  6.989846
## estimated sd:  1.710654
## estimated skewness:  -0.02848522
## estimated kurtosis:  2.923099
```

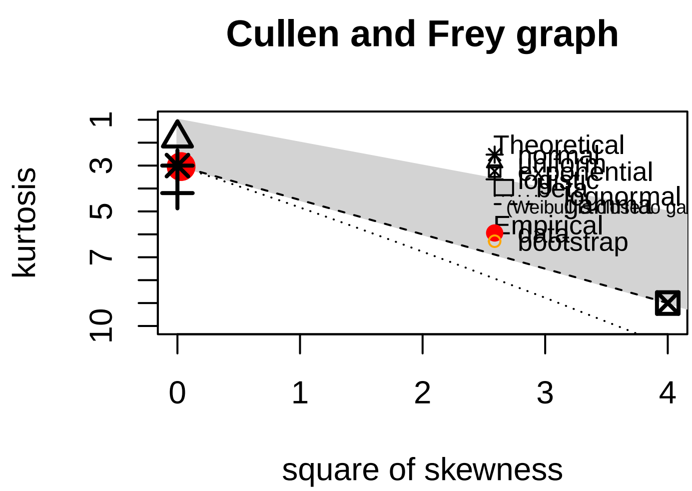
```r
sev2 <- df2$claim_size[df2$claim_size > 0]
descdist(sev2, boot = 1000)
```




```
## summary statistics
## ------
## min:  0.8236877   max:  3401.825
```

```
## median:  717.4712
## mean:  802.7218
## estimated sd:  506.1836
## estimated skewness:  0.9267448
## estimated kurtosis:  3.901759
```

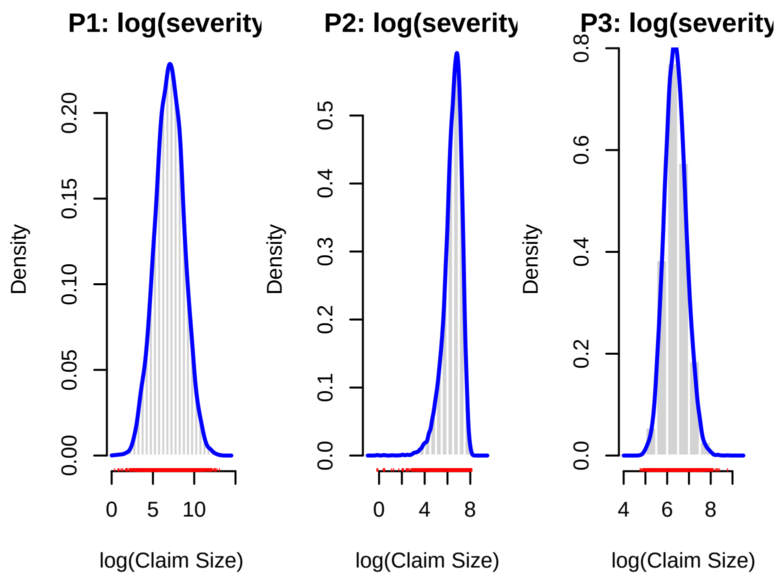
```r
sev3 <- df3$claim_size[df3$claim_size > 0]
descdist(log(sev3), boot = 1000)
```




```
## summary statistics
## ------
## min:  4.75679   max:  8.772254
## median:  6.368575
## mean:  6.382347
## estimated sd:  0.4890409
## estimated skewness:  0.173933
## estimated kurtosis:  3.042674
```

```r
#--- 1. Prepare log-severity vectors
logs1 <- log(df1$claim_size[df1$claim_size > 0])
logs2 <- log(df2$claim_size[df2$claim_size > 0])
logs3 <- log(df3$claim_size[df3$claim_size > 0])

#--- 2. A plotting function
plot_log_severity <- function(logs, title) {
  x_min <- floor(min(logs))
  x_max <- ceiling(max(logs)) + 0.5

  hist(logs,
       breaks = seq(x_min, x_max, by = 0.5),
       freq   = FALSE,
       xlim   = c(x_min, x_max),
       col    = "lightgray",
       border = "white",
```

```r
      main   = title,
      xlab   = "log(Claim Size)"
  )
  dens <- density(logs, from = x_min, to = x_max)
  lines(dens, col="blue", lwd=2)
  rug(logs, col="red", ticksize=0.02)
}

#--- 3. Plot side by side
par(mfrow = c(1,3), mar = c(4,4,2,1))
plot_log_severity(logs1, "P1: log(severity)")
plot_log_severity(logs2, "P2: log(severity)")
plot_log_severity(logs3, "P3: log(severity)")
```




```r
par(mfrow = c(1,1))
```

Question 1: Summarize the recommended modelling parameters for each portfolio of risk.

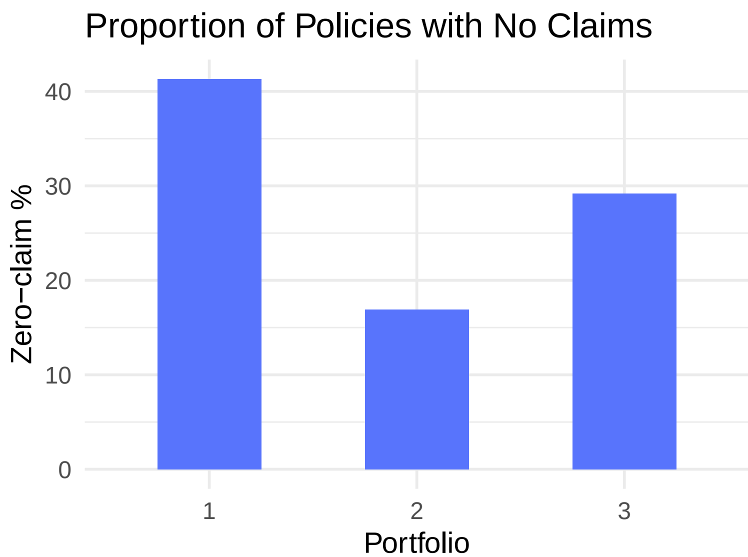
```r
# Load libraries & data
library(readxl); library(dplyr); library(fitdistrplus); library(MASS)

df_list <- lapply(1:3, function(i) read_excel("1463797.xlsx", sheet = i))

# Summarize frequency & choose model
freq_summary <- lapply(df_list, function(df) {
  counts <- df %>% group_by(id) %>% summarise(
    zero     = all(claim_size==0),
    claims   = ifelse(zero,0L,n())
  )
  m <- mean(counts$claims); v <- var(counts$claims)
  zero_pct <- 100*mean(counts$claims==0)
```

```r
  if (v/m > 1.05) {
    size <- m^2/(v-m); model <- "Negative Binomial"; params <- c(mu=m, size=size)
  } else {
    model <- "Poisson"; params <- c(lambda=m)
  }
  list(zero_pct=round(zero_pct,1), model=model, params=round(params,3))
})

# Summarize severity & choose model
sev_summary <- lapply(df_list, function(df) {
  sev <- df$claim_size[df$claim_size > 0]

  # fit both
  gfit <- fitdist(sev, "gamma", start=list(shape=1, scale=mean(sev)))
  lfit <- fitdist(sev, "lnorm")

  # pull out AIC
  aic_g <- gofstat(gfit)$aic
  aic_l <- gofstat(lfit)$aic

  if (aic_l < aic_g) {
    model  <- "Lognormal"
    params <- lfit$estimate[c("meanlog","sdlog")]
  } else {
    model  <- "Gamma"
    params <- gfit$estimate[c("shape","scale")]
  }
  list(model = model, params = round(params,3))
})


# Build and print summary table
summary_tbl <- tibble::tibble(
  Portfolio    = 1:3,
  ZeroClaimPct = sapply(freq_summary, `[[`, "zero_pct"),
  FreqModel    = sapply(freq_summary, `[[`, "model"),
  FreqParams   = sapply(freq_summary, function(x) paste(names(x$params),"=",x$params,collapse=", ")),
  SevModel     = sapply(sev_summary, `[[`, "model"),
  SevParams    = sapply(sev_summary, function(x) paste(names(x$params),"=",x$params,collapse=", "))
)
print(summary_tbl)
```

```
## # A tibble: 3 x 6
##   Portfolio ZeroClaimPct FreqModel         FreqParams         SevModel SevParams
##       <int>        <dbl> <chr>             <chr>              <chr>    <chr>
## 1         1         41.3 Poisson           lambda = 0.904     Lognorm~ meanlog ~
## 2         2         16.9 Poisson           lambda = 1.599     Gamma    shape = ~
## 3         3         29.2 Negative Binomial mu = 1.629, size ~ Lognorm~ meanlog ~
```

```r
# Barplot of zero-claim %
library(ggplot2)

ggplot(summary_tbl, aes(x = factor(Portfolio), y = ZeroClaimPct)) +
```

```r
  geom_col(width = 0.5, fill = "#5874fc") +
  labs(
    x = "Portfolio",
    y = "Zero-claim %",
    title = "Proportion of Policies with No Claims"
  ) +
  theme_minimal()
```

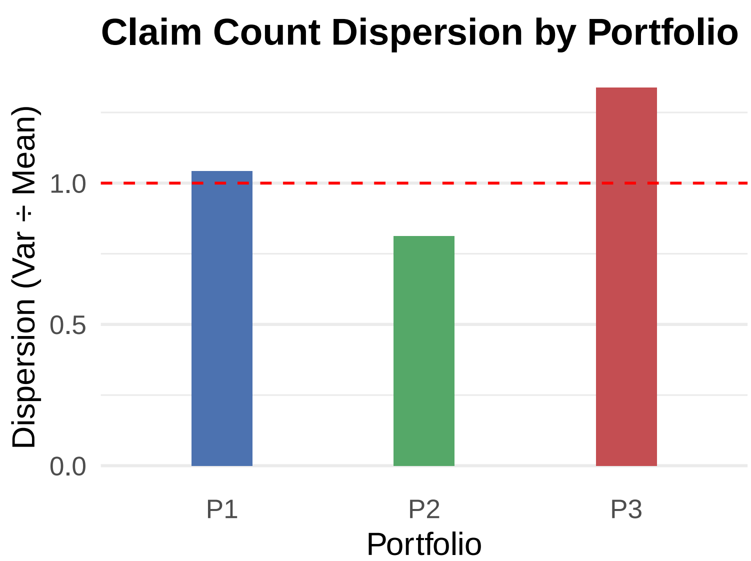


Question 2

Part 1: Dispersion of the policies

```r
library(dplyr)
library(ggplot2)

get_disp <- function(df, name) {
  counts <- df %>% group_by(id) %>%
    summarise(claims = ifelse(all(claim_size==0), 0L, n())) %>% ungroup()
  m <- mean(counts$claims); v <- var(counts$claims)
  data.frame(Portfolio = name, Dispersion = v/m)
}

disp1 <- get_disp(df1, "P1")
disp2 <- get_disp(df2, "P2")
disp3 <- get_disp(df3, "P3")
disp_df <- bind_rows(disp1, disp2, disp3)

ggplot(disp_df, aes(x = Portfolio, y = Dispersion, fill = Portfolio)) +
  geom_col(width = 0.3, show.legend = FALSE) +
  scale_x_discrete(
    expand = c(.3,0)
  ) +
```

```r
  scale_fill_manual(values = c("P1" = "#4C72B0",
                               "P2" = "#55A868",
                               "P3" = "#C44E52")) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "red") +
  labs(
    x = "Portfolio",
    y = "Dispersion (Var ÷ Mean)",
    title = "Claim Count Dispersion by Portfolio"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    axis.ticks.x       = element_blank(),
    panel.grid.major.x = element_blank(),
    plot.title         = element_text(face = "bold", size = 14)
  )
```



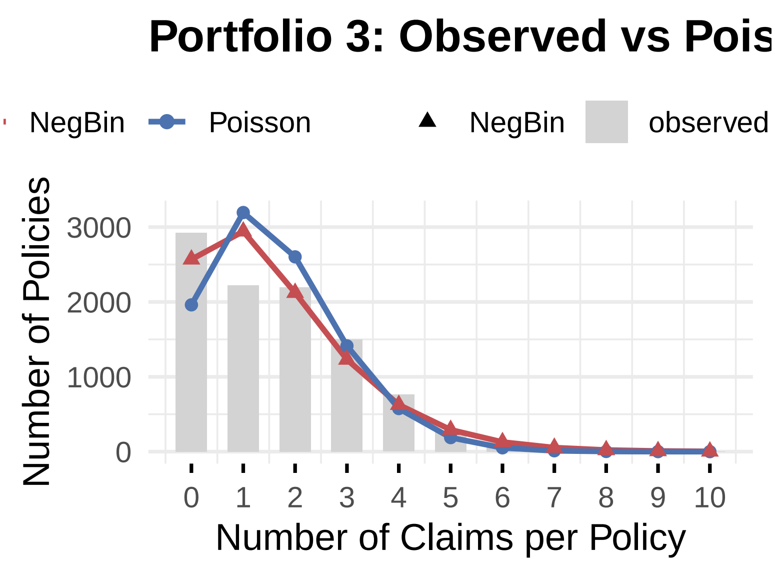
Part 2: Demonstrating heterogeneity in portfolio 3

```r
library(dplyr)
library(ggplot2)
library(MASS)

# 1. Prepare the data
cts3 <- df3 %>%
  group_by(id) %>%
  summarise(claims = ifelse(all(claim_size == 0), 0L, n())) %>%
  ungroup()

obs_df <- cts3 %>% count(claims) %>% rename(observed = n)

pois3 <- glm(claims ~ 1, family = poisson(), data = cts3)
nb3   <- glm.nb(claims ~ 1, data = cts3)
```

```r
lam3  <- exp(coef(pois3))
mu3   <- exp(coef(nb3)); th3 <- nb3$theta

maxc <- max(cts3$claims)
model_df <- tibble(
  claims   = 0:maxc,
  observed = obs_df$observed,
  Poisson  = dpois(claims, lam3) * nrow(cts3),
  NegBin   = dnbinom(claims, size = th3, mu = mu3) * nrow(cts3)
)

# 2. Pivot longer for ggplot
plot_df <- model_df %>%
  pivot_longer(cols = c("observed","Poisson","NegBin"),
               names_to = "Model", values_to = "Count")

# 3. Plot
ggplot(plot_df, aes(x = claims, y = Count, fill = Model, color = Model, shape = Model)) +
  # observed as bars
  geom_col(data = filter(plot_df, Model == "observed"),
           aes(x = claims, y = Count, fill = Model),
           color = NA, width = 0.6) +
  # Poisson & NegBin as lines + points
  geom_line(data = filter(plot_df, Model != "observed"),
            aes(x = claims, y = Count, color = Model), size = 1) +
  geom_point(data = filter(plot_df, Model != "observed"),
             aes(x = claims, y = Count, color = Model, shape = Model), size = 2) +
  # manual scales
  scale_fill_manual(
    name  = "",
    values = c("observed" = "lightgray", "Poisson" = NA, "NegBin" = NA)
  ) +
  scale_color_manual(
    name   = "",
    values = c("observed" = NA, "Poisson" = "#4C72B0", "NegBin" = "#C44E52")
  ) +
  scale_shape_manual(
    name   = "",
    values = c("observed" = NA, "Poisson" = 16,      "NegBin" = 17)
  ) +
  scale_x_continuous(breaks = 0:maxc) +
  labs(
    x     = "Number of Claims per Policy",
    y     = "Number of Policies",
    title = "Portfolio 3: Observed vs Poisson & NegBin Fits"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title         = element_text(face = "bold"),
    panel.grid.major.x = element_blank(),
    axis.ticks.x       = element_line(),
    axis.text.x        = element_text(margin = margin(t = 5)),
    legend.position    = "top"
```

```
  )
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```



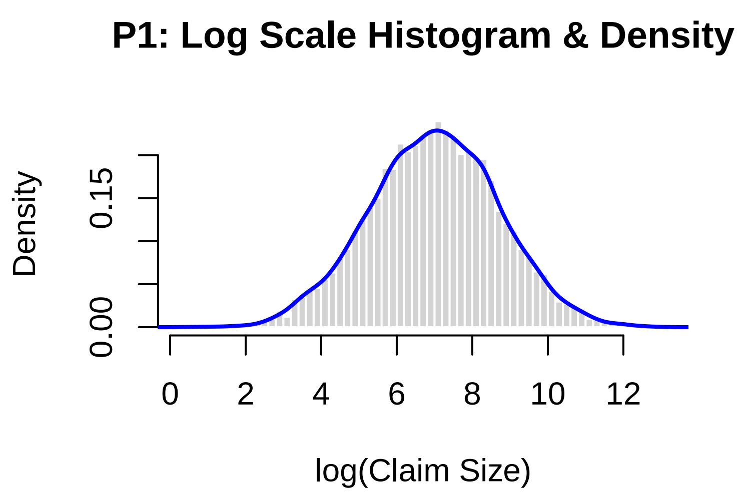
Question 3

```r
library(readxl)
library(fitdistrplus)
library(actuar)     # for Pareto
library(extRemes)   # for mrlplot
```

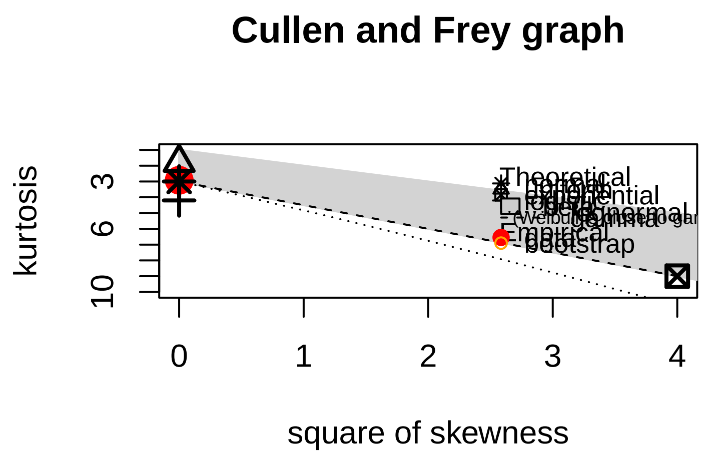
```
## Loading required package: Lmoments

## Loading required package: distillery

##
## Attaching package: 'extRemes'

## The following objects are masked from 'package:stats':
##
##     qqnorm, qqplot
```

```r
# 1. Data
df1   <- read_excel("1463797.xlsx", sheet="Sheet1")
sev1  <- df1$claim_size[df1$claim_size > 0]
n <- length(sev1)
```

```r
logs1 <- log(sev1)

# 2. Fit candidates (for Q-Q)
fit_gamma  <- fitdist(sev1, "gamma",  start=list(shape=1,scale=mean(sev1)))
fit_lnorm  <- fitdist(sev1, "lnorm")
fit_pareto <- fitdist(sev1, "pareto", start=list(shape=2,scale=min(sev1)))

# Emprical density (logged data)
par(mar = c(5,4,4,2) + 0.1)  # extra top margin for title
hist(logs1,
     breaks = 50,
     freq   = FALSE,
     col    = "lightgray",
     border = "white",
     xlab   = "log(Claim Size)",
     ylab   = "Density",
     main   = "P1: Log Scale Histogram & Density"
)
lines(density(logs1), col = "blue", lwd = 2)
```

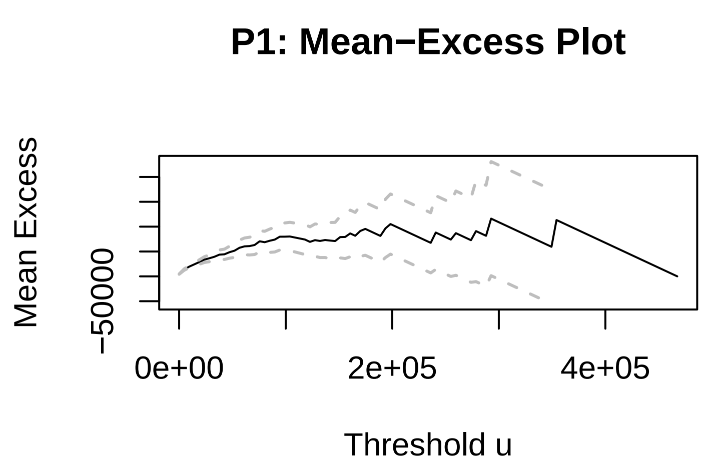



```r
# 3a) Cullen-Frey on log-data
par(mar = c(5,4,6,2))      # bottom, left, top, right margins
descdist(logs1, boot=1000, discrete=FALSE)
```

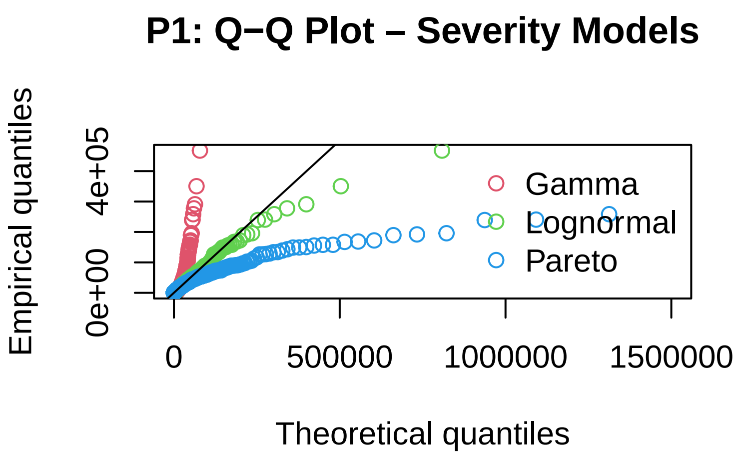



```
## summary statistics
## ------
## min:  0.3577616   max:  13.05511
## median:  7.012698
## mean:  6.989846
## estimated sd:  1.710654
## estimated skewness:  -0.02848522
## estimated kurtosis:  2.923099
```

```r
# Default title "Cullen and Frey graph" sits neatly in that top margin.

# 3b) Mean-excess plot
par(mar = c(5,4,6,2))
mrlplot(sev1,
        main = "P1: Mean-Excess Plot",
        xlab = "Threshold u")
```



```r
# 3c) Q-Q comparison
par(mar = c(5,4,6,2))
qqcomp(
  list(fit_gamma, fit_lnorm, fit_pareto),
  legendtext = c("Gamma","Lognormal","Pareto"),
  main = "P1: Q-Q Plot - Severity Models",
  xlab = "Theoretical quantiles",
  ylab = "Empirical quantiles",
  xlim = c(0, 1500000)
)
```




```r
# 3d) AIC and BIC comparison
fits <- list(
  Gamma = fit_gamma,
  Lognormal = fit_lnorm,
  Pareto = fit_pareto
)

ic_tab <- t(sapply(fits, function(fit) {
  k <- length(fit$estimate)
  ll     <- fit$loglik
  AIC    <- 2*k - 2*ll
  BIC    <- log(n)*k - 2*ll
  c(AIC = round(AIC,1), BIC = round(BIC,1))
}))
print(ic_tab)
```

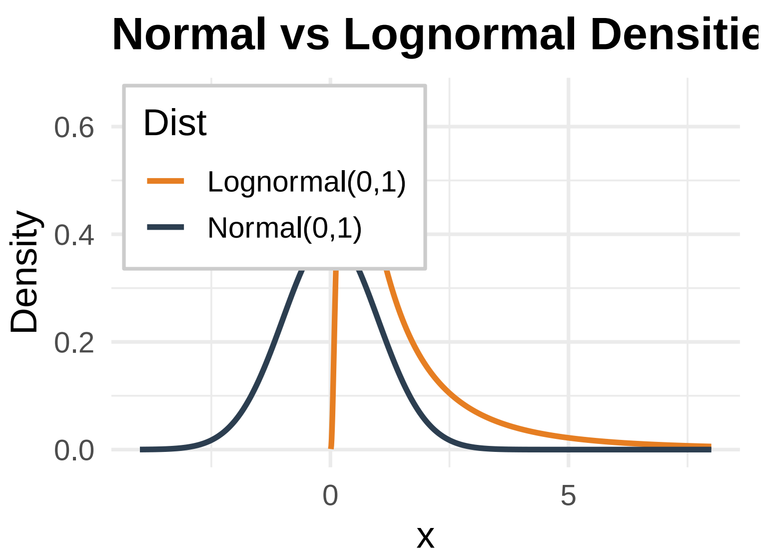
```
##                AIC      BIC
## Gamma     164796.8 164811.0
## Lognormal 161740.5 161754.8
## Pareto    161931.7 161945.9
```

```r
# Estimating paramters of log normal distribution
fit_lnorm <- fitdist(sev1, "lnorm")
meanlog <- fit_lnorm$estimate["meanlog"]
sdlog   <- fit_lnorm$estimate["sdlog"]
meanlog; sdlog
```

```
##  meanlog
## 6.989846

##   sdlog
## 1.71056
```

```r
library(ggplot2)

# Parameters
mu    <- 0
sigma <- 1

# Data frames
norm_df <- data.frame(
  x = seq(-4, 8, length.out = 1000),
  y = dnorm(seq(-4, 8, length.out = 1000), mean = mu, sd = sigma),
  Dist = "Normal(0,1)"
)

log_df <- data.frame(
  x = seq(0.01, 8, length.out = 1000),
  y = dlnorm(seq(0.01, 8, length.out = 1000), meanlog = mu, sdlog = sigma),
  Dist = "Lognormal(0,1)"
)

# Combine
plot_df <- rbind(norm_df, log_df)

# Plot
ggplot(plot_df, aes(x = x, y = y, color = Dist)) +
  geom_line(size = 1) +
  scale_color_manual(
    values = c("Normal(0,1)" = "#2C3E50", "Lognormal(0,1)" = "#E67E22")
  ) +
  labs(
    title = "Normal vs Lognormal Densities",
    x     = "x",
    y     = "Density"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title      = element_text(face = "bold"),
    legend.position   = c(0.02, 0.98),       # near top-left
    legend.justification = c(0, 1),          # left/top corner of legend at that point
    legend.background = element_rect(fill = "white", color = "gray80")
  )
```

```
## Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
## 3.5.0.
## i Please use the `legend.position.inside` argument of `theme()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Question 4

```r
library(dplyr)

# Aggregate per-policy totals (example for Portfolio 1)
agg <- df2 %>%
  group_by(id) %>%
  summarise(total = sum(claim_size)) %>%
  ungroup()

n     <- nrow(agg)
p     <- mean(agg$total > 0)              # ~59%
meanN <- n * p                            # 5 900
varN  <- n * p * (1 - p)                  # 2 419
p
```

```
## [1] 0.8308
```

```r
tibble(
  Model      = c("Binomial","Poisson"),
  Mean       = c(meanN,      meanN),
  Variance   = c(varN,       meanN),
  Dispersion = c(varN/meanN, 1)
)
```

```
## # A tibble: 2 x 4
##   Model     Mean Variance Dispersion
##   <chr>    <dbl>    <dbl>      <dbl>
## 1 Binomial  8308    1406.      0.169
## 2 Poisson   8308    8308       1
```

```r
# Binomial Dispersion = 0.41  vs Poisson Dispersion = 1.0
```

```r
library(fitdistrplus)

cts3 <- df3 %>%
  group_by(id) %>%
  summarise(
    # count how many non-zero claim_size entries each policy has
    claims = sum(claim_size > 0)
  ) %>%
  ungroup()

fit_nb3b <- fitdist(cts3$claims, "nbinom")

fit_nb3b$estimate
```

```
##     size       mu
## 3.860821 1.629247
```