STSB6816 Test 1 of 2023

Mathematical Statistics and Actuarial Science; University of the Free State

2023/04/20

Time: 180 minutes; Marks: 50

MEMORANDUM

Instructions

- Answer all questions in a single R Markdown document. Please knit to PDF or Word at the end and submit both the PDF/Word document and the ".Rmd" file for assessment, in that order.
- Label questions clearly, as it is done on this question paper.
- All results accurate to about 3 decimal places.
- Show all derivations, formulas, code, sources, and reasoning.
- Intervals should cover 95% probability unless stated otherwise.
- No communication software, devices, or communication capable websites may be accessed prior to submission. You may not (nor even appear to) attempt to communicate or pass information to another student.

Introduction

The data is provided at https://ufs.blackboard.com. **It consists of the flight times (in seconds) of paper air planes constructed and thrown by primary school learners.**

These times are assumed to follow a Gompertz distribution. You are encouraged to fit the Gompertz distribution to these times as part of this process, although other approaches will get partial credit.

The Gompertz distribution has density function

$$f(t) = \alpha\exp(\lambda t)\exp\left[-\frac{\alpha}{\lambda}(\exp(\lambda t) - 1)\right]$$

and survival function

$$S(t) = \exp\left[-\frac{\alpha}{\lambda}(\exp(\lambda t) - 1)\right]$$

In order to establish that you fit the distribution correctly, you are instructed first to generate a sample that you know is from a Gompertz distribution and apply your fitting approach to this simulated sample. The idea is that should you get the same parameters out that you put in, then your approach will have more credibility on the real data.

Question 1

1.1) Derive the inverse survival function or inverse CDF (your choice). **[4]**

```
cat("$$\\begin{aligned}
u &= \\exp\\left[-\\frac{\\alpha}{\\lambda}(\\exp(\\lambda t) - 1)\\right] \\\\
\\log (u) &= -\\frac{\\alpha}{\\lambda}(\\exp(\\lambda t) - 1) \\\\
1 - \\frac{\\lambda\\log (u)}{\\alpha} &= \\exp(\\lambda t) \\\\
\\log\\left[1-\\log (u)\\lambda\\alpha^{-1}\\right]\\lambda^{-1} &= t
\\end{aligned}$$")
```

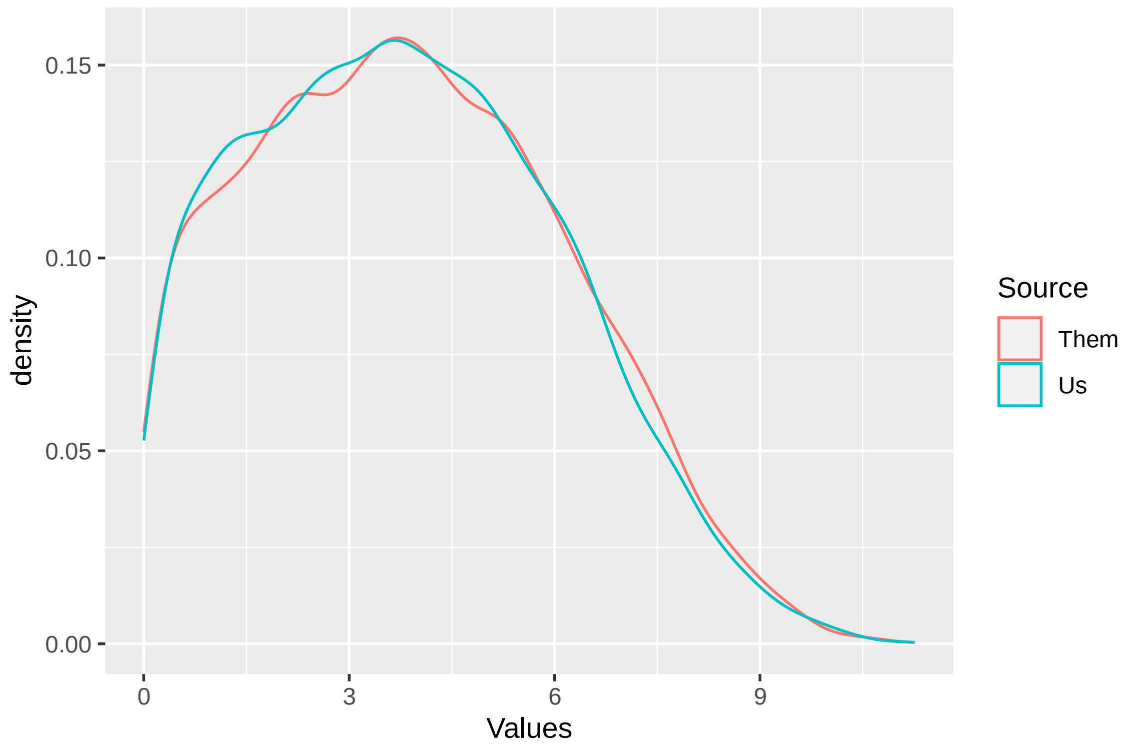
$$\begin{aligned}
u &= \exp\left[-\frac{\alpha}{\lambda}(\exp(\lambda t) - 1)\right] \\
\log (u) &= -\frac{\alpha}{\lambda}(\exp(\lambda t) - 1) \\
1 - \frac{\lambda\log (u)}{\alpha} &= \exp(\lambda t) \\
\log\left[1-\log (u)\lambda\alpha^{-1}\right]\lambda^{-1} &= t
\end{aligned}$$

Set S or F equal to u [2]. Attempt to solve for t [1]. Get the right function [1].

1.2) Using the inverse survival function or inverse CDF (your choice), generate a sample of size 500 from a Gompertz(0.3, 0.1) distribution. **[4]**

[Using an established package to generate the sample instead of your own will earn 2 out of 4 marks; while using an established package in addition to your own and showing that they match within the simulation error will earn 5 out of 4 marks.]

```
library(tidyverse)

rGomp <- function(n = 1, lambda = 1, alpha = 1) {
  log(1 - log(runif(n))*lambda/alpha)/lambda
}

nsims <- 10000
sims <- data.frame(Source = rep(c('Us', 'Them'), each = nsims),
                   Values = c(rGomp(nsims, lambda = 0.3, alpha = 0.1),
                              DescTools::rGompertz(nsims, shape = 0.3, rate = 0.1)))
sims |> ggplot(aes(x = Values, colour = Source)) + geom_density()
```



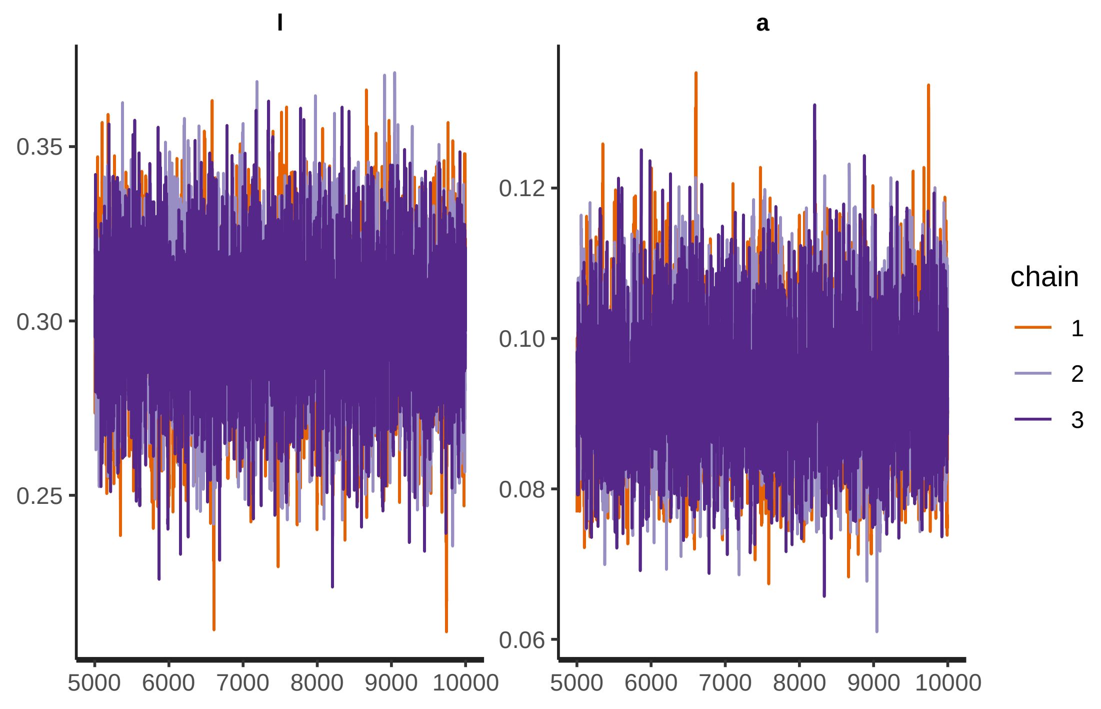
```
nsims <- 500
x <- rGomp(nsims, lambda = 0.3, alpha = 0.1)
```

Generating a sample as indicated by any means [2]. Using own function that matches derivation [2]. Showing that the established package gives the same results [1 bonus].

1.3) Fit a Gompertz distribution to the simulated times. Give parameter estimates, with uncertainty, for your fit (trace plots showing good convergence are highly recommended for simulation fits). **[11]**

Hint: As the Gompertz distribution is not one of the standard distributions in Stan, it is recommended that you add the following code to the start of your Stan model. This will allow you to sample from the Gompertz in the usual way (*i.e.* y[i] ~ Gompertz(lambda, alpha)). Alternatively, specify the full log posterior in Stan directly to the model using *target +=* and then Stan math functions. Stan's math functions are similar to R's math functions.

```
functions {
  real Gompertz_lpdf(real y, real lam, real a) {
      return log(a) + lam*y - (exp(lam*y)-1)*a/lam;
  }
}

library(rstan)
mycores <- 3
options(mc.cores = mycores)

// This Stan block defines a Gompertz model by Sean van der Merwe, UFS
functions {
  real Gompertz_lpdf(real y, real lam, real a) {
      return log(a) + lam*y - (exp(lam*y)-1)*a/lam;
  }
}
data {
  int<lower=1> n;            // number of observations
  real<lower=0> y[n];    // observations
}
// The parameters of the model
parameters {
  real<lower=0> a;       // alpha
  real<lower=0> l;       // Lambda
}
model {
  for (i in 1:n) {
    y[i] ~ Gompertz(l, a);
  }
}

saveRDS(GompertzModel, file = 'GompertzModel.Rds')

ModelFitSims <- sampling(GompertzModel, list(n=length(x), y=x), iter = 10000, chains
= mycores)

pars_of_interest <- c('l', 'a')
ModelFitSims |> traceplot(pars = pars_of_interest)
```



```
summary(ModelFitSims, pars = pars_of_interest)$summary |> kable(digits = 3)
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
l	0.299	0	0.020	0.259	0.286	0.299	0.313	0.338	3269.357	1.001
a	0.094	0	0.009	0.078	0.088	0.093	0.099	0.112	3384.431	1.001

Defining the Gompertz model correctly: function/target [1], data [1], parameters [2], likelihood [2]. Fitting the Gompertz model to the simulated data [2], giving parameter estimates with uncertainty [2] and trace plot showing convergence [1]. An alternative, sensible, and correctly implemented model can earn (data 1 + pars 1 + lik 1 + fit 1 + ests 1 + trace 1) = 6 marks; while a non-Bayesian implementation can get up to 10 marks here (but will forfeigt later marks).

1.4) For each parameter, report how many absolute standard deviations it is away from the known values used to simulate the sample. Comment on whether the values are reasonable. **[5]**

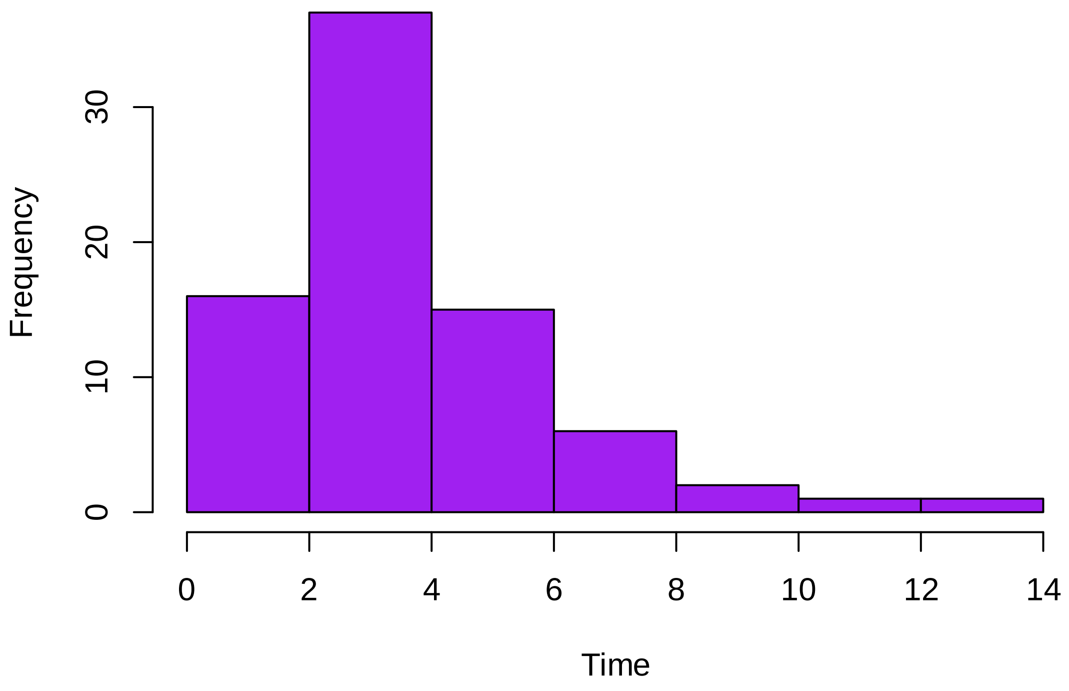
```
simssims <- rstan::extract(ModelFitSims)
c(
  lambda = abs(mean(simssims$l) - 0.3)/sd(simssims$l),
  alpha = abs(mean(simssims$a) - 0.1)/sd(simssims$a)
) |> round(4)
```

```
| lambda  alpha
| 0.0340 0.7139
```

Summarising the requested statistics neatly [3]. Comment saying that it is not multiple standard deviations (less than 2) and thus fairly reasonable [2].

1.5) Import the data set into R and explore it visually. You could use a histogram or density plot and discuss what you see. **[4]**

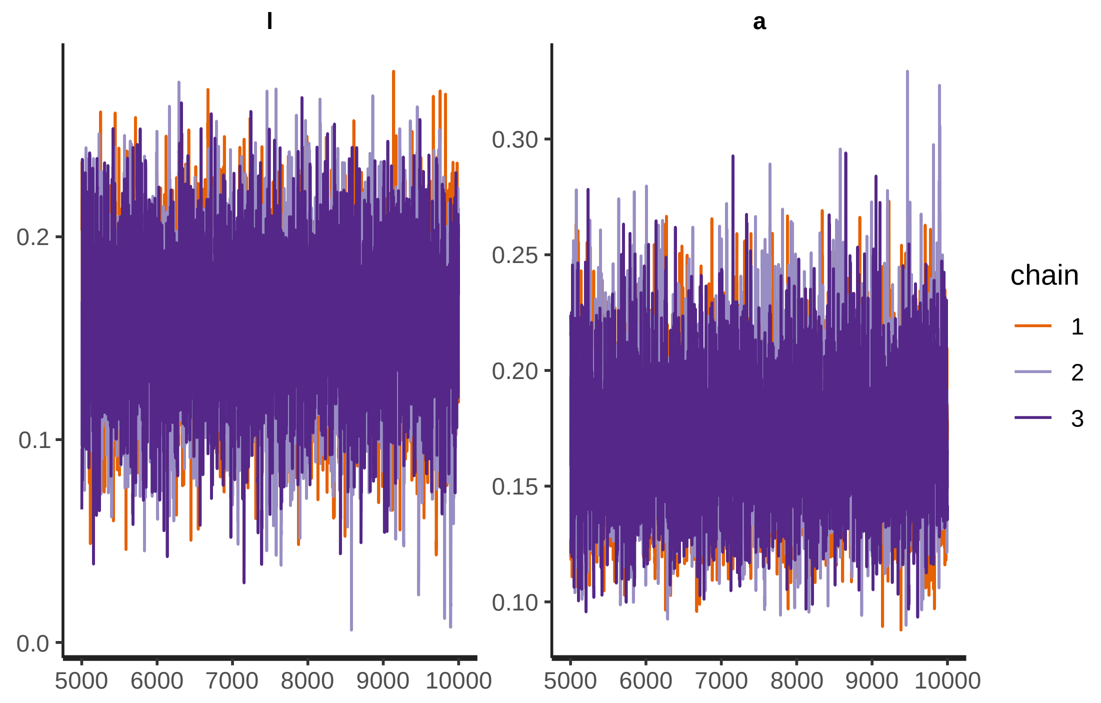
```
"STSB6816Test1Data2023.xlsx" |> openxlsx::read.xlsx("TestData") -> d

par(mar=c(5,5,1,1))
d$Time |> hist(col = 'purple', main = '', xlab = 'Time')
```



Loading data [1], histogram/density plot [1], and discussion saying something about skewness or that some planes flew really long and well while most crashed in an expected fashion [2].

1.6) Fit a Gompertz distribution to the observed times. Give parameter estimates, with uncertainty, for your fit. **[6]**

```
ModelFitData <- sampling(GompertzModel, list(n=nrow(d), y=d$Time), iter = 10000,
chains = mycores)

ModelFitData |> traceplot(pars = pars_of_interest)
```



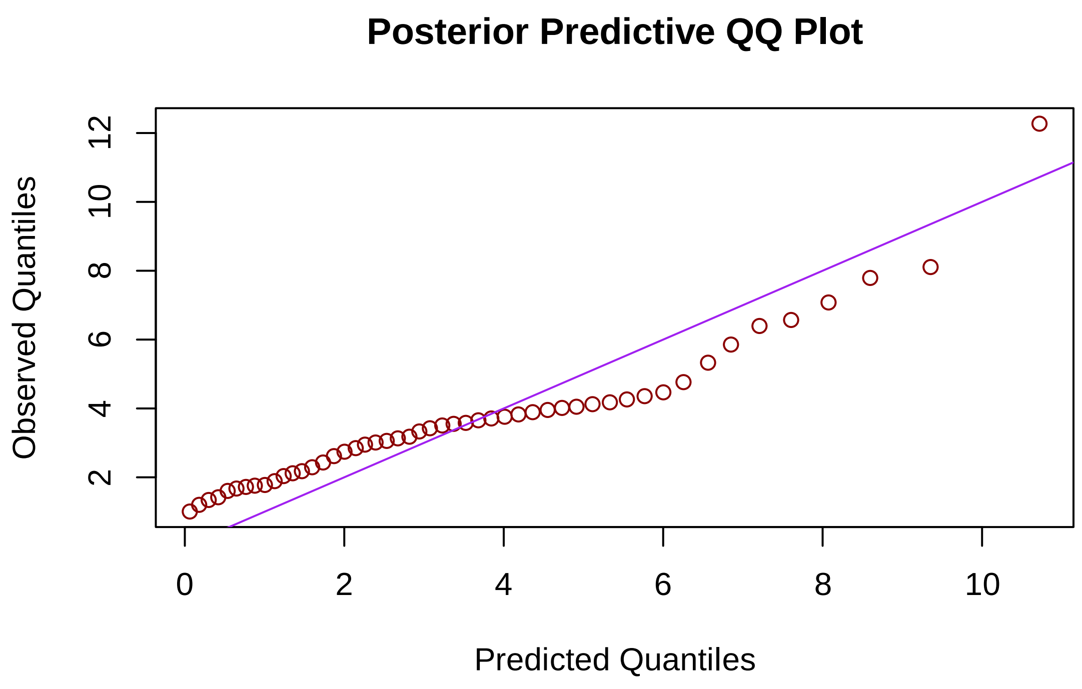
```
summary(ModelFitData, pars = pars_of_interest)$summary |> kable(digits = 3)
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
l	0.157	0.001	0.036	0.085	0.134	0.158	0.182	0.225	4702.600	1.001
a	0.170	0.000	0.029	0.118	0.149	0.168	0.188	0.232	4744.076	1.001

Fitting the Gompertz model to the observed data [3], giving parameter estimates with uncertainty [2] and trace plot showing convergence [1].

1.7) Draw a quantile-quantile plot showing the quantiles of the observed data against the quantiles of the posterior predictive distribution of the next random throw. Comment on the quality of the fit, both the discrepancies and possible sources of discrepancies. **[7]**

```
datasims <- rstan::extract(ModelFitData)
preds <- rGomp(length(datasims$l), datasims$l, datasims$a)
qseq <- seq(0.01, 0.99, 0.02)
plot(quantile(preds, qseq), quantile(d$Time, qseq),
     main='Posterior Predictive QQ Plot',
     xlab = 'Predicted Quantiles', ylab = 'Observed Quantiles',
     col = 'darkred')
lines(c(0,13), c(0,13), col = 'purple')
```



Drawing the QQplot in any sensible way [4]. Saying that the fit is not good [1]: short flights were longer than expected, and long flights were shorter than expected, except for one extreme value distorting the results [2].

1.8) Consider 10 random future throws. What is the probability that the one that flies the furthest stays in the air for more than 8 seconds? **[5]**

Hint: You must predict sets of 10 throws, then check whether the longest flight time of the 10 is longer than 8 seconds. You must do this at least 1000 times and average the results to get a probability estimate.

```
longest <- preds |> matrix(10) |> apply(2, max)
mean(longest > 8)

| [1] 0.5253333
```

Combining enough predictions in an organised fashion [2], finding the longest time of each set [1], and getting the final probability [2].

1.9) Is your probability above sensible based on the observed data? First give an instinctive answer then calculate a bootstrap/resampling estimate using the data alone and compare. **[4]**

```
longest <- replicate(2000, {d$Time |> sample(10, replace = TRUE) |> max()})
mean(longest > 8)

| [1] 0.3915
```

First, we see that the probability is near 50%, indicating an uncertain model trying to estimate something quite possible but uncertain [1]. Then we see that the bootstrap approach [2] gives a similar result [1], lending credibility to our result.

Points total

The points on the test add up to **50**