Assignment 03 - Solutions

Statistical Computing and Empirical Methods

YOUR_NAME (YOUR_STUDENT_ID)

A word of advice

Think of the SCEM labs as going to the gym: if you pay a gym membership, but instead of working out you use a machine to lift the weights for you, you won't get the benefits.

ChatGPT, DeepSeek, Claude and other GenAI tools can provide answers to most of the questions below. Before you try that, please consider the following: answering the specific questions below is not the point of this assignment. Instead, the questions are designed to give you the chance to develop a better understanding of estimation concepts and a certain level of ***statistical thinking***. These are essential skills for any data scientist, even if they end up using generative AI - to write an effective prompt and to catch the common (often subtle) errors that AI produces when trying to solve anything non-trivial.

A very important part of this learning involves not having the answers ready-made for you, but instead taking the time to actually search for the answer, trying something, getting it wrong, and trying again.

So, make the best use of this session. The assignments are not marked, so it is much better to try the yourself even if you get incorrect answers (you'll be able to correct yourself later when you receive feedback) than to submit a perfect, but GPT'd solution.

IMPORTANT NOTES:

- **DO NOT** change the code block names. Enter your solutions to each question into the predefined code blocks.
- **DO NOT** add calls to `install.packages()` into your solutions. Some questions may require you to load packages using `library()`. Please do not use any other packages except the ones explicitly listed in the "setup" code block below.

Setup

This code block below sets up your session. The only think you should change in it is to replace `ABCDEF` by your student ID number, which will be used as the seed for your random number generators.

Part I: Point estimation of parameters

Q1. Simulating data and estimating the mean

```r
## Question 1.a
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
#
```

```r
# ...
# x.mean <- ...
# x.var  <- ...
xunif <- runif(12, min = 50, max = 80)
xunif
```

```
##  [1] 63.37659 72.80904 60.95964 50.65372 65.14843 59.56130 71.64306 62.79967
##  [9] 54.20282 76.35271 69.49369 54.75461
```

```r
xunif.mean <- mean(xunif)
xunif.mean
```

```
## [1] 63.47961
```

```r
xunif.var <- var(xunif)
xunif.var
```

```
## [1] 64.29231
```

```r
## Question 1.b
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
n <- 200
p_true <- 0.25

xbern <- rbinom(n, size = 1, prob= p_true)
xbern
```

```
##   [1] 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0
##  [38] 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 0 1 1 0 0 0 1 0 0 1 0 0 1 0 0
##  [75] 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0
## [112] 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0
## [149] 0 1 1 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 0
## [186] 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0
```

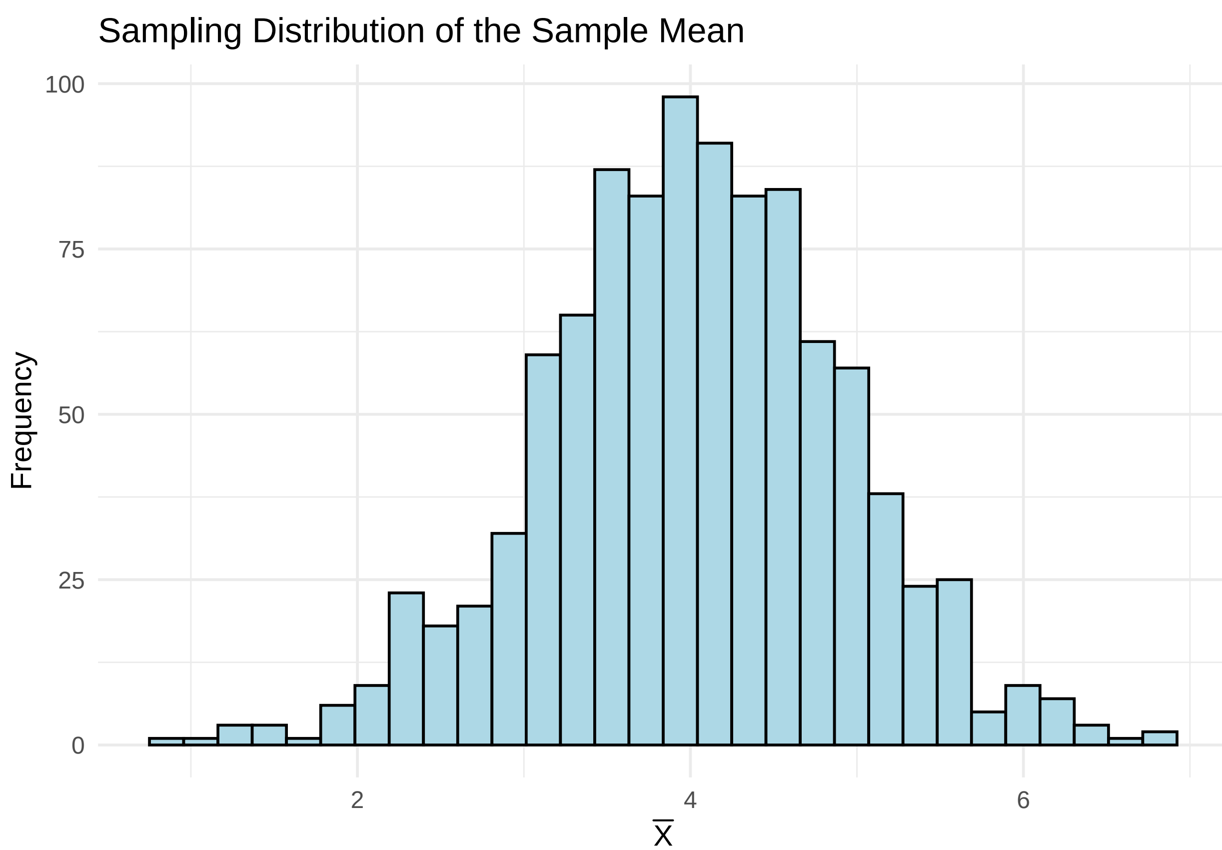
```r
xbern.p <- mean(xbern)
c(true_p = p_true, sample_p = xbern.p)
```

```
##   true_p sample_p
##    0.250    0.255
```

Q2. Sampling distribution of the mean

```r
## Question 2.a
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
n <- 5
mu <- 4
sigma <- 2
reps <- 1000

xbar.vector <- numeric(reps)
```

```r
for (i in 1:reps) {
  sample_data <- rnorm(n, mean=mu, sd=sigma)
  xbar.vector[i] <-mean(sample_data)
}

library(ggplot2)
ggplot(data.frame(xbar = xbar.vector), aes(x=xbar)) +
  geom_histogram(bins = 30, color= "black", fill="lightblue") +
  labs(
    title = "Sampling Distribution of the Sample Mean",
    x = expression(bar(X)),
    y = "Frequency"
  ) +
  theme_minimal()
```




```r
## Question 2.b
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
xbar.mean <- mean(xbar.vector)
xbar.mean
```

```
## [1] 4.01869
```

```r
xbar.sd <- sd(xbar.vector)
xbar.sd
```

```
## [1] 0.9049617
```

```r
c(samples_mean = xbar.mean, mu = mu)
```

```
## samples_mean           mu
##      4.01869      4.00000
```

```r
c(sd_samples_mean = xbar.sd, se_means = sigma/sqrt(n))
```

```
## sd_samples_mean        se_means
##       0.9049617       0.8944272
```

Q3. Bias of an estimator

```r
## Question 3
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
n <- 20
num_samples <- 2000
true_variance <- 1

xvar.vector <- numeric(num_samples)

for (i in 1:num_samples) {
  sample_data <- rnorm(n, 0, 1)
  sample_mean <- mean(sample_data)
  mle_variance <- sum((sample_data - sample_mean)^2)/n
  xvar.vector[i] <- mle_variance
}

avg_mle_variance <- mean(xvar.vector)
avg_mle_variance
```

```
## [1] 0.9388511
```

```r
xvar.bias <- avg_mle_variance - true_variance
xvar.bias
```

```
## [1] -0.06114892
```

```r
print("Negatively biased")
```

```
## [1] "Negatively biased"
```

Discussion: Maximum Likelihood Estimation (MLE)

Example of using the R function `optim()` to estimate the MLE values for the mean and variance of a normal distribution. *You don't need to change anything in the code block below.*

```r
## This is just an example

set.seed(1)
n    <- 100000 # sample size
mu   <- 12     # true mean
var  <- 9      # true variance


# Generate sample
x <- rnorm(n, mean = mu, sd = sqrt(var))

# log-likelihood for Normal(mu, sigma^2), given a sample X
# (Check lecture slides for the formula)
llik <- function(par, X) {
  mu     <- par[1]
  sigma2 <- par[2]
  n      <- length(X)
  if (sigma2 <= 0) return(Inf)  # variance must be positive
  llik <- -n * log(sqrt(2 * pi * sigma2)) - 0.5 * sum((X - mu)^2 / sigma2)
  return(llik)
}

# Initial guesses for mu and sigma^2
# (any finite Real values of mu and of sigma2 > 0 should work)
init <- c(mu = 0, sigma2 = 1)

# Run optimization
fit <- optim(par = init, fn = llik, X = x,
             method = "L-BFGS-B",         # optimisation method to use
             lower = c(-Inf, 1e-9),       # Enforce positive variance (minimal allowed value: 10^-9)
             control = list(fnscale = -1) # To make it a maximisation problem
             )

fit$par
```

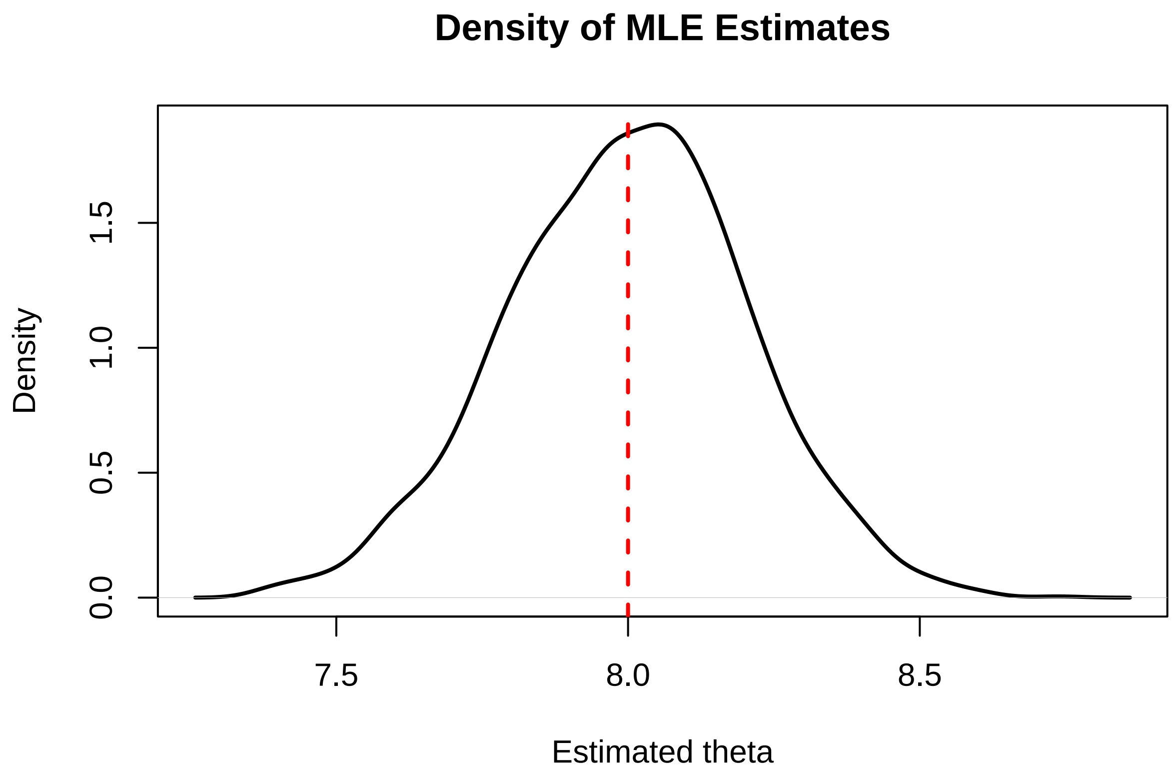
```
##        mu    sigma2
## 11.993268  9.063433
```

Q4: Numerical computation of MLE value

```r
## Question 4.a
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
theta_true <- 5
sample_data <- rcauchy(10000, location = theta_true)
neg_loglik <- function(theta, x) {
  sum(log(1+(x-theta)^2))
}

result <- optim(par=median(sample_data), fn=neg_loglik, x= sample_data, method = "BFGS")
result$par
```

```
## [1] 4.989948
```

```r
## Question 4.b
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
n <- 50
num_samples <- 2000
theta_true <- 8
cauchy.vector <- numeric(num_samples)

mle_cauchy <- function(x) {
  neg_loglik <- function(theta) sum(log(1+(x-theta)^2))
  result <- optim(par = median(x), fn=neg_loglik, method = "BFGS")
  return(result$par)
}

for (i in 1:num_samples) {
  x <- rcauchy(n, location = theta_true)
  cauchy.vector[i] <- mle_cauchy(x)
}

plot(density(cauchy.vector),
     main = "Density of MLE Estimates",
     xlab = "Estimated theta",
     lwd=2
     )
abline(v= theta_true, col= "red", lty=2, lwd=2)
```



Part II: Data visualisation

Q5: Basic plotting

```
## Question 5.a
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
library(nycflights13)
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
```

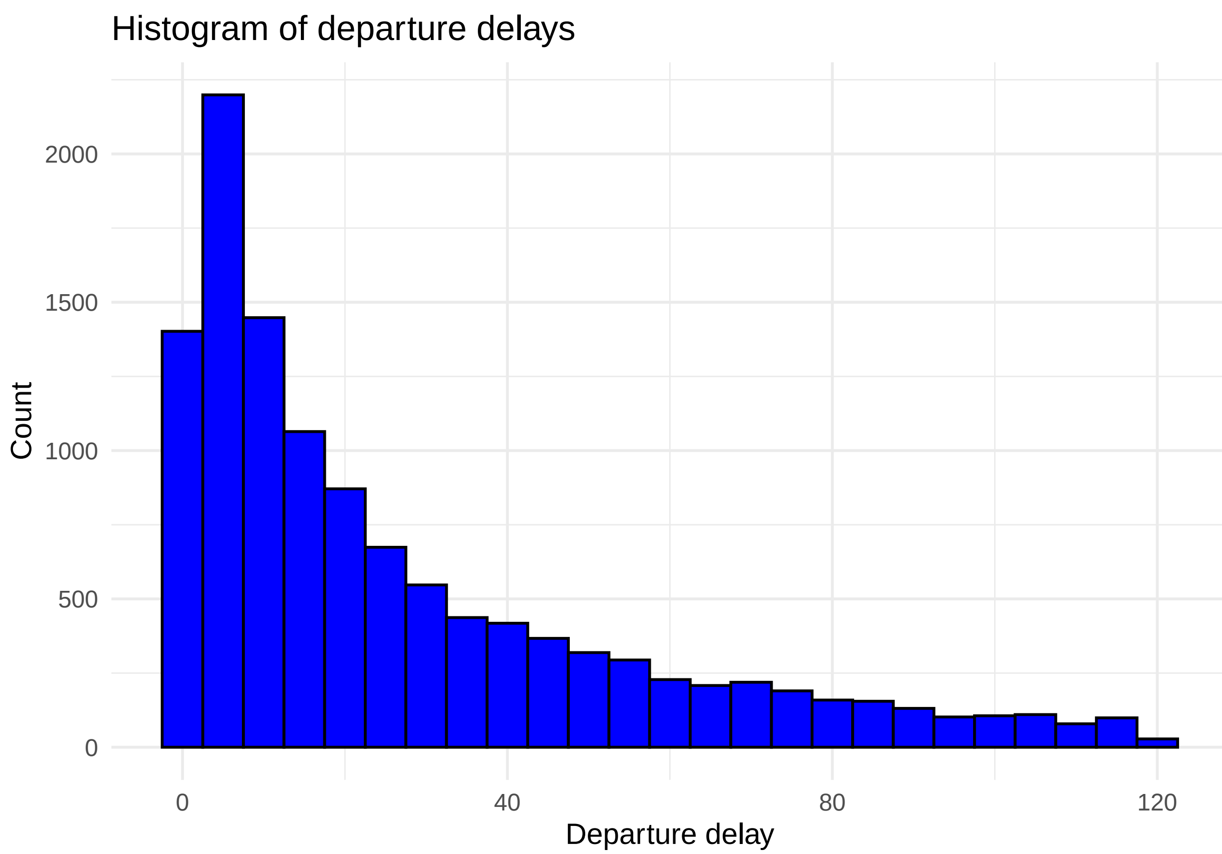
```
library(ggplot2)
ls()
```

```
##  [1] "avg_mle_variance"   "cauchy.vector"      "fit"
##  [4] "i"                  "init"               "llik"
##  [7] "mle_cauchy"         "mle_variance"       "mu"
## [10] "MY_STUDENT_ID"      "n"                  "neg_loglik"
## [13] "num_samples"        "p_true"             "permitted.packages"
## [16] "reps"               "result"             "sample_data"
## [19] "sample_mean"        "sigma"              "theta_true"
## [22] "true_variance"      "var"                "x"
## [25] "xbar.mean"          "xbar.sd"            "xbar.vector"
## [28] "xbern"              "xbern.p"            "xunif"
## [31] "xunif.mean"         "xunif.var"          "xvar.bias"
## [34] "xvar.vector"
```

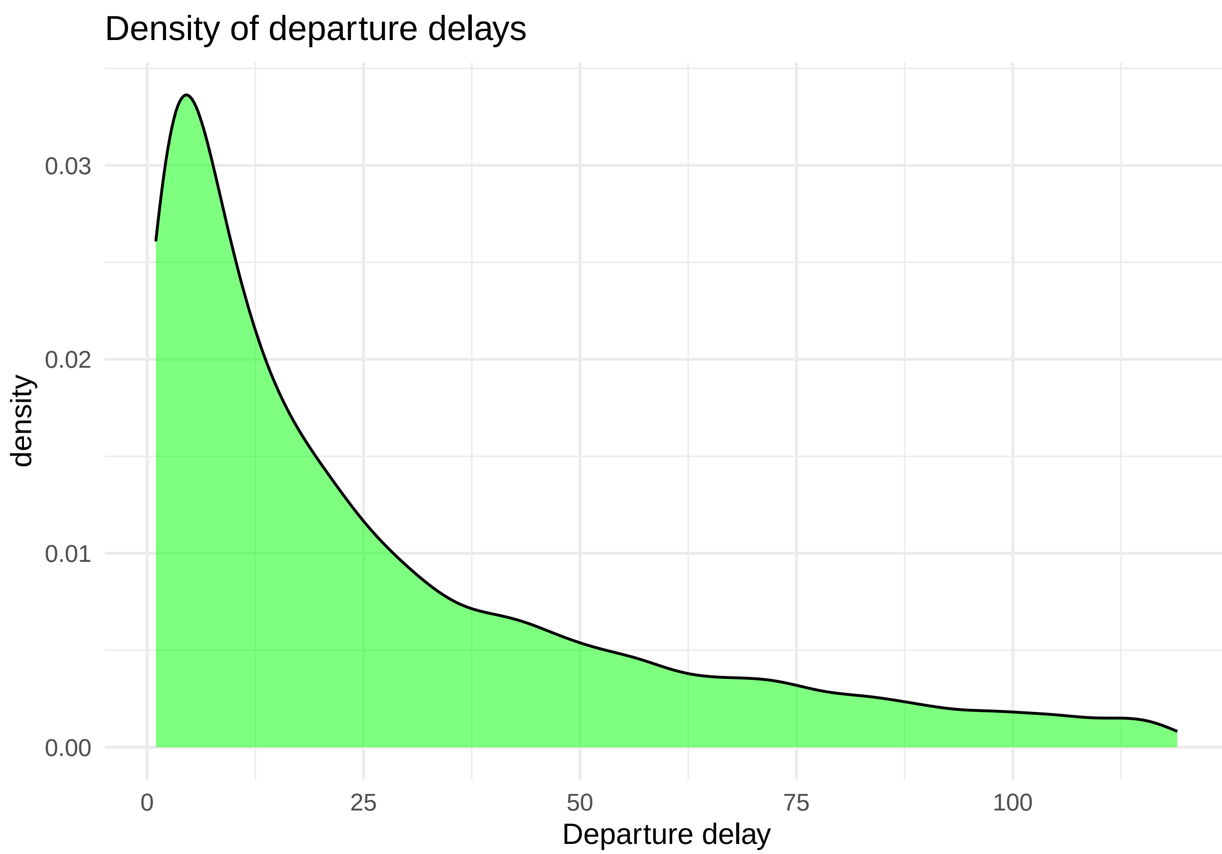
```r
head(flights)
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

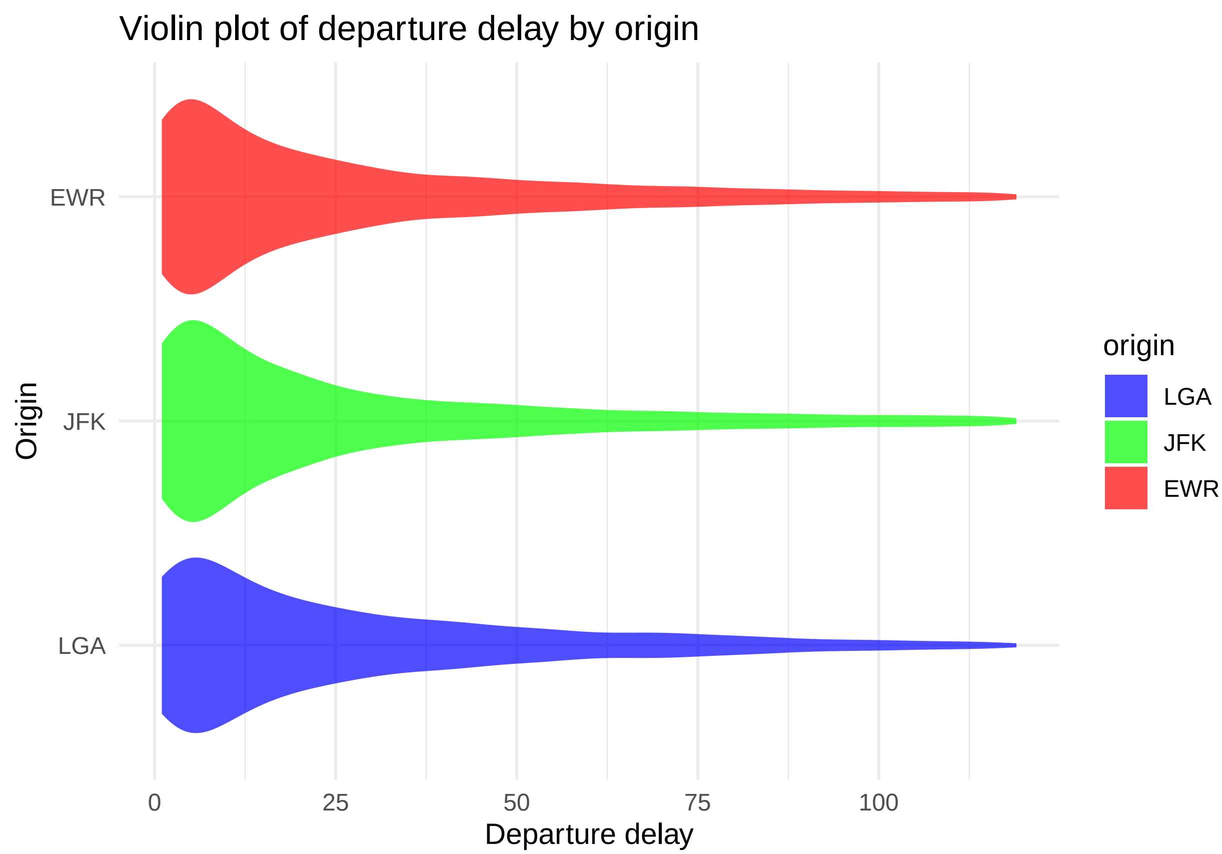
```r
# ?flights

delayed_flights_2h <- flights %>%
  filter(dep_delay>0, dep_delay<120) %>%
  slice_sample(prop = 0.1)

ggplot(delayed_flights_2h, aes(x=dep_delay)) +
  geom_histogram(binwidth = 5, color = "black", fill="blue") +
  labs(title="Histogram of departure delays",
       x = "Departure delay", y = "Count") +
  theme_minimal()
```



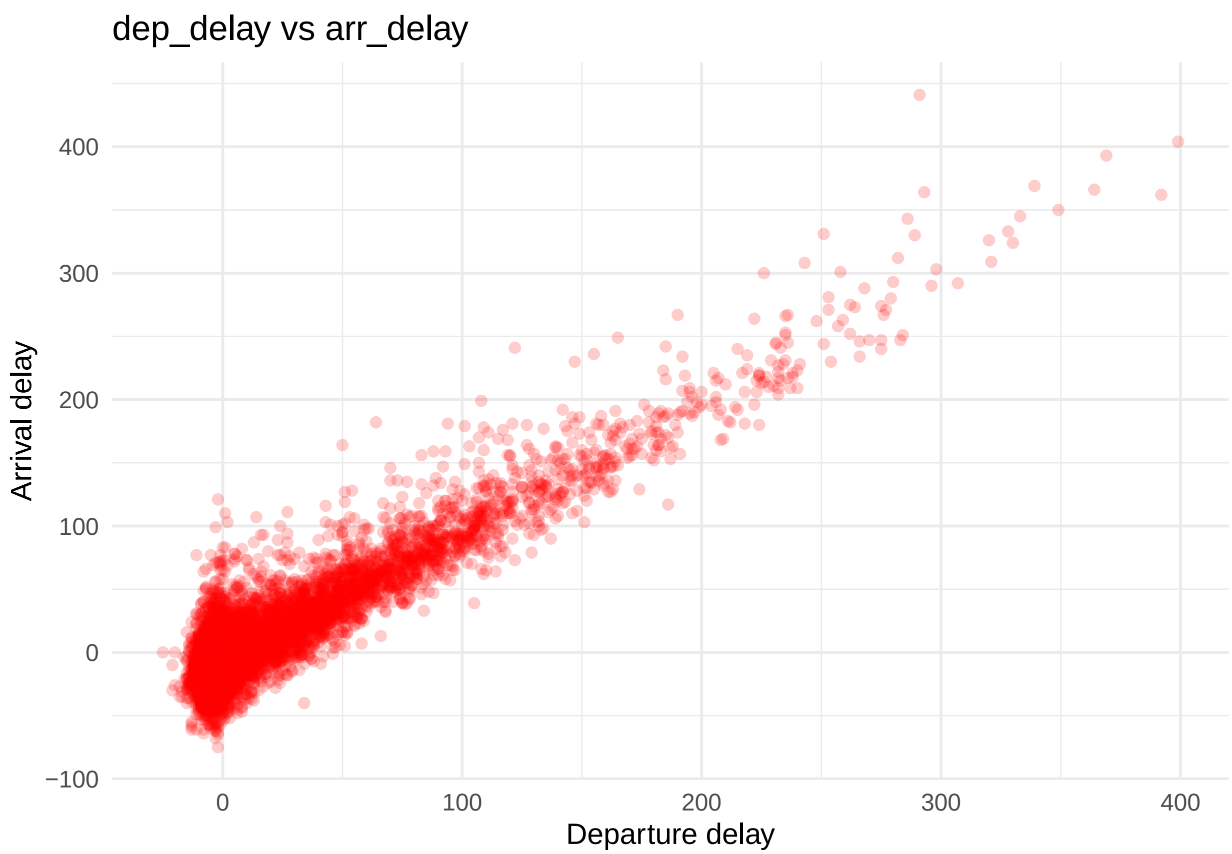

```r
## Question 5.b
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
ggplot(delayed_flights_2h, aes(x=dep_delay))+
  geom_density(fill = "green", alpha=0.5) +
  labs(title = "Density of departure delays",
       x = "Departure delay") +
  theme_minimal()
```




```r
## Question 5.c
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
delayed_grouped <- flights %>%
  filter(!is.na(dep_delay), dep_delay>0, dep_delay<120) %>%
  slice_sample(prop=0.1) %>%
  mutate(origin = factor(origin, levels = c("LGA", "JFK", "EWR")))

air_colors <- c("EWR" = "red", "JFK" = "green", "LGA"="blue")


ggplot(delayed_grouped, aes(x=dep_delay, y=origin, fill=origin))+
  geom_violin(alpha=0.7, trim=TRUE, color = NA) +
  scale_fill_manual(values = air_colors, name="origin")+
  labs(title = "Violin plot of departure delay by origin",
       x = "Departure delay", y = "Origin")+
  theme_minimal()+
  theme(legend.position = "right")
```



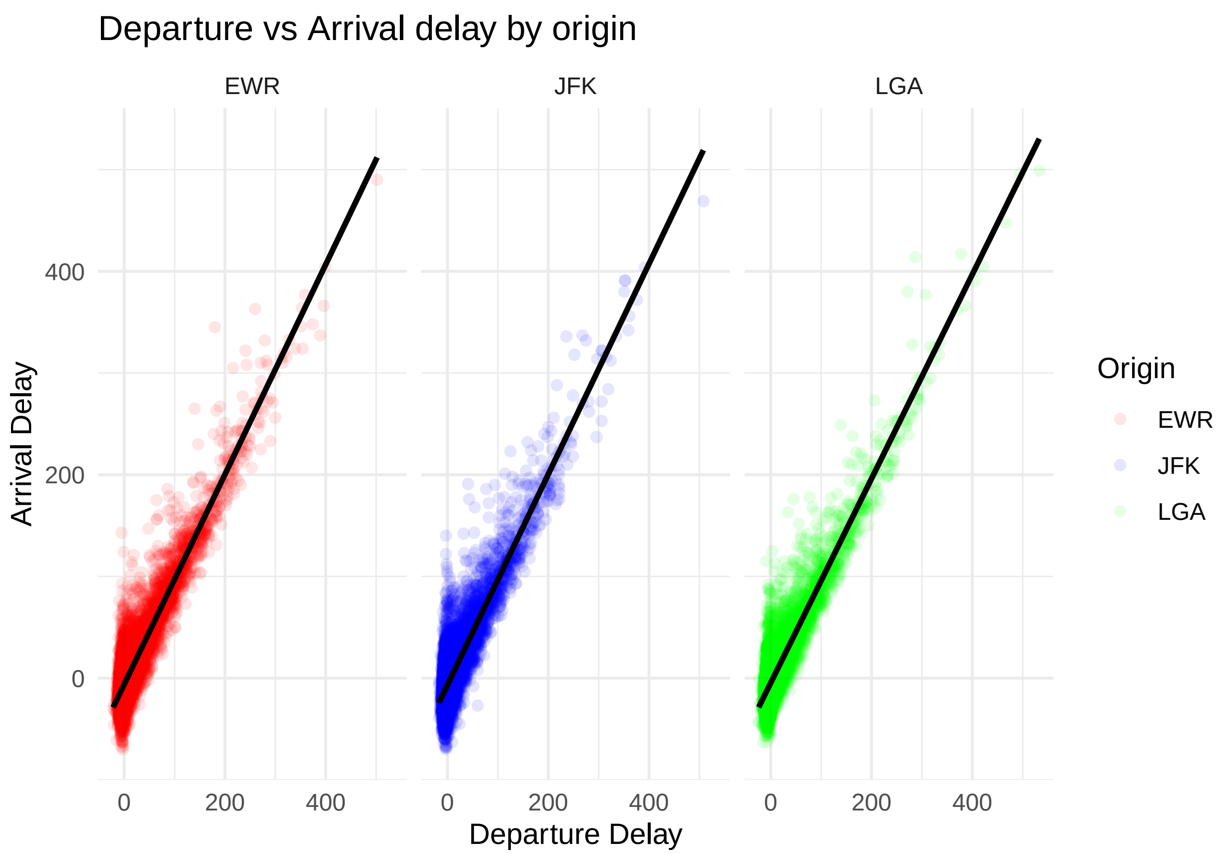

```r
## Question 5.d
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
ewr_sample <- flights %>%
  filter(origin=="EWR", !is.na(dep_delay), !is.na(arr_delay)) %>%
  slice_sample(prop = 0.1)

ggplot(ewr_sample, aes(x=dep_delay, y=arr_delay))+
  geom_point(color = "red", alpha=0.2) +
  labs(title = "dep_delay vs arr_delay",
       x = "Departure delay", y = "Arrival delay")+
  theme_minimal()
```



Q6: Facetting and annotation

```
## Question 6.a
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
sample_df <- flights %>%
  filter(!is.na(dep_delay), !is.na(arr_delay)) %>%
  slice_sample(prop = 0.10)
origin_colors <- c("EWR" = "red", "JFK" = "blue", "LGA" = "green")

p <- ggplot(sample_df, aes(x = dep_delay, y = arr_delay, color=origin))+
  geom_point(alpha=0.10)+
  geom_smooth(method="lm", se=FALSE, color="black")+
  facet_wrap(~ origin) +
  scale_color_manual(values = origin_colors, name = "Origin") +
  labs(title = "Departure vs Arrival delay by origin",
       x="Departure Delay", y="Arrival Delay")+
  theme_minimal()
p
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



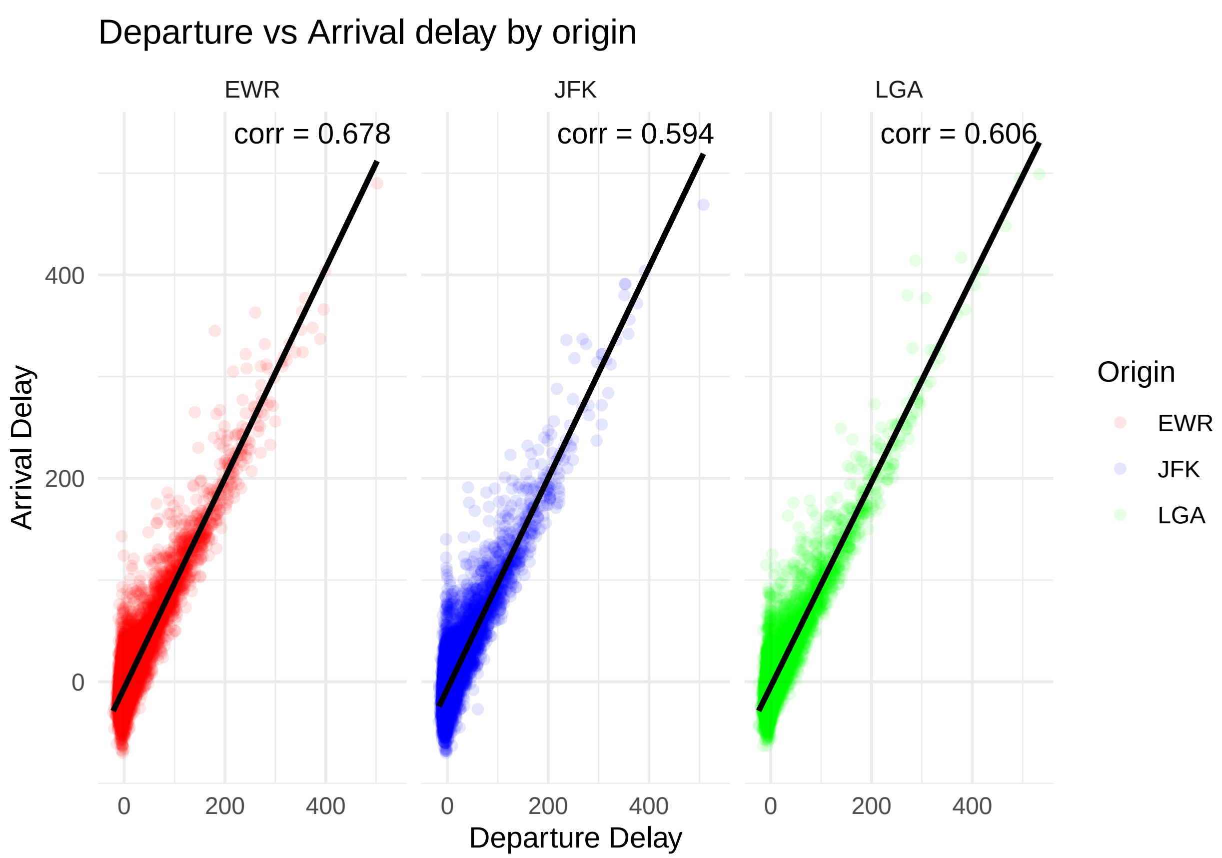

```r
## Question 6.b
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
cor_df <-flights %>%
  filter(!is.na(dep_delay), !is.na(arr_delay)) %>%
  group_by(origin) %>%
  summarize(
    cor_spearman = cor(dep_delay, arr_delay, method="spearman", use = "pairwise.complete.obs"),
)%>%
  arrange(desc(cor_spearman))
cor_df
```

```
## # A tibble: 3 x 2
##   origin cor_spearman
##   <chr>         <dbl>
## 1 EWR           0.678
## 2 LGA           0.606
## 3 JFK           0.594
```

```r
## Question 6.c
set.seed(MY_STUDENT_ID) # <--- Don't change this

# Your code here
##
label_df <- cor_df %>%
```

```r
  mutate(
    label=paste0("corr = ", round(cor_spearman,3)),
    x=Inf,
    y=Inf
  )

p+geom_text(
  data=label_df,
  aes(x=x, y=y, label=label),
  inherit.aes = FALSE,
  hjust=1.1, vjust=1.5,
  size=4, color="black"
)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Part III: Conceptual/Theory Questions

Q7

- **Point estimator:** A rule or formula that gives an estimate of a population parameter from sample data (e.g., the sample mean for the population mean).
- **Point estimate:** The actual numerical value obtained when the estimator is applied to a data set.
- **Sampling distribution:** The probability distribution of an estimator when repeated samples of the same size are taken.

Q8

- **Bias:** Difference between the expected value of an estimator and the true parameter.
- **Variance:** How much the estimator's values vary between different samples.
- We may prefer a *slightly biased estimator* if it has much lower variance — giving smaller overall error (bias–variance tradeoff).

Q9

- **Mean Squared Error (MSE):**

$$\mathrm{MSE}(\hat{\theta}) = \mathbb{E}[(\hat{\theta} - \theta)^2]$$

It measures overall estimation error and can be written as:

$$\mathrm{MSE} = \mathrm{Variance} + (\mathrm{Bias})^2$$

Q10

- **Central Limit Theorem (CLT):** For large samples, the sampling distribution of the mean $\bar{X}$ is approximately normal:

$$\frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \sim N(0, 1)$$

- **Importance:** It allows us to use normal-based confidence intervals and hypothesis tests for many estimators, even when the data are not normally distributed.