Assignment 03

Statistical Computing and Empirical Methods

A word of advice

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

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 them 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.

Introduction

Before starting, make sure you have reviewed the Week 3 lecture slides and watched all videos for the week.

This assignment will be done in **RStudio**. To enter your solutions, please use the pre-structured R markdown template provided for this week. From now on, we will use this format, which will also be used in the final coursework.

You can **optionally** submit your .Rmd solutions file by 15:00h Friday 10 October. This will help us understand your work but will not count towards your final grade.

If you want to upload your .Rmd file, use the "Assignment 03" link under the *Assessments, Submission and Feedback* tab in the course Blackboard page.

Part I: Point estimation of parameters

In this part we will work with simulated data and basic R functions to explore point estimation concepts.

Q1. Simulating data and estimating the mean

a. Generate a random sample of size `n = 12` from a uniform distribution between `min = 50` and `max = 80`, and store it as variable `xunif`. Compute the sample mean and sample variance, and store their values as variables `xunif.mean` and `xunif.var`.

b. Simulate 200 Bernoulli trials with probability of success `p = 0.25` and store it as variable `xbern`. Estimate the population proportion of successes using the sample proportion and store it as variable `` `xbern.p ``. Compare it with the true value.

(*Tip: Remember that the Binomial distribution is the distribution of K Bernoulli trials - there are at least 2 ways of solving this question using R's Binomial random number generator*)

Q2. Sampling distribution of the mean

a. Run **1000 repeats** of generating $n = 5$ observations from a normal variable with $\mu = 4, \sigma = 2$, storing the sample mean each time. This should generate a vector with 1000 values, each representing the mean of a sample of size 5 (call this vector `xbar.vector`). Plot a histogram of the resulting sampling distribution. You can use either R's basic histogram or ggplot2.

b. Compute the mean and standard deviation of the sample means stored in `xbar.vector`, and store them as variables `xbar.mean` and `xbar.sd`. Compare the observed values with the theoretical ones, i.e., the mean μ and the standard error of the means, $\sigma_{\bar{X}} = \sigma/\sqrt{n}$.

Q3. Bias of an estimator

Suppose we estimate the population variance using the MLE formula,

$$\hat{\sigma}^2_{MLE} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2.$$

Simulate 2000 samples of size `n = 20` a standard normal variable (i.e., with $\mu = 0, \sigma = 1$). For each sample, compute the MLE estimator above. Store your variance estimates into a vector `xvar.vector`. Estimate the *bias* of this estimator by taking the difference between the average of estimated variances stored in `xvar.vector` and the actual value of the parameter σ. Store the result in a variable `xvar.bias`. Is this estimator positively or negatively biased?

Discussion: Maximum Likelihood Estimation (MLE)

Sometimes there is no analytical solution to the optimisation problem modelling a maximum likelihood estimator. In those cases, it is common to use numeric optimisation strategies to iteratively solve the optimisation problem, instead of relying on an exact formula. While the details of optimisation theory are (quite unfortunately) more than we can cover in this course, you can use built-in R optimisation functions to tackle some simple problems.

As an example, here is how to use the R function `optim()` to estimate the MLE values for the mean and variance of a normal distribution:

Q4: Numerical computation of MLE value

A random variable X has a Cauchy distribution with location parameter θ if its density is

$$f(X|\theta) := \frac{1}{\pi + \pi(x-\theta)^2}.$$

The Cauchy log-likelihood function given a sample $\mathbf{X} = \{X_1, \ldots, X_n\}$ is given by

$$l(\theta|\mathbf{X}) = -n\log(\pi) - \sum_{i=1}^{n} \log\left(1 + (X_i - \theta)^2\right),$$

for which there is no analytic solution.

a. Based on the earlier example, compute the MLE estimate of the Cauchy parameter θ using a sample with 10000 observations. Use $\theta = 5$ as the real value of the parameter.

b. Simulate 2000 samples of size `n = 50` of $Cauchy(\theta = 8)$. For each sample, compute the MLE estimator as you did in the previous item. Store your MLE estimates into a vector `cauchy.vector`. Plot the estimated density of the distribution of MLE values. You can use the base R `plot(density(...))` or ggplot2's `geom_density()` for this.

Part II: Data visualisation

This part is mainly about data visualisation using ggplot2.

We will use mainly the `nycflights` dataset that you used last week, which you can load using `library(nycflights13)`.

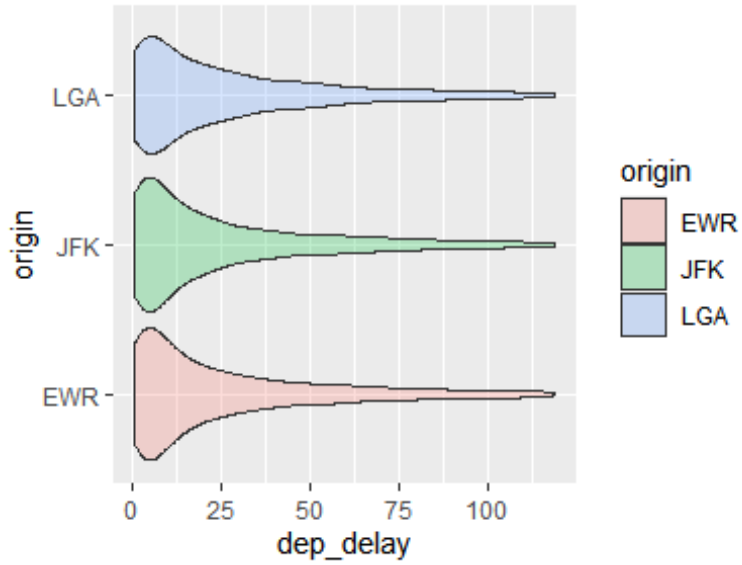
Q5.

Load the `nycflights13` package. Check that the `flights` dataset was loaded either by using `ls()` or trying to print its first rows with `head(flights)`. Read the description of the columns in this data frame using `?flights`.

a. Create a new variable called `delayed_flights_2h` containing only the rows with a **departure delay longer than 0 and shorter than 120 minutes**, randomly sampling 10% of the rows (you can use dplyr's `filter()` and `slice_sample()` functions for this). Then use a combination of the functions `ggplot()` and `geom_histogram()` to create a histogram plot of the departure delays.

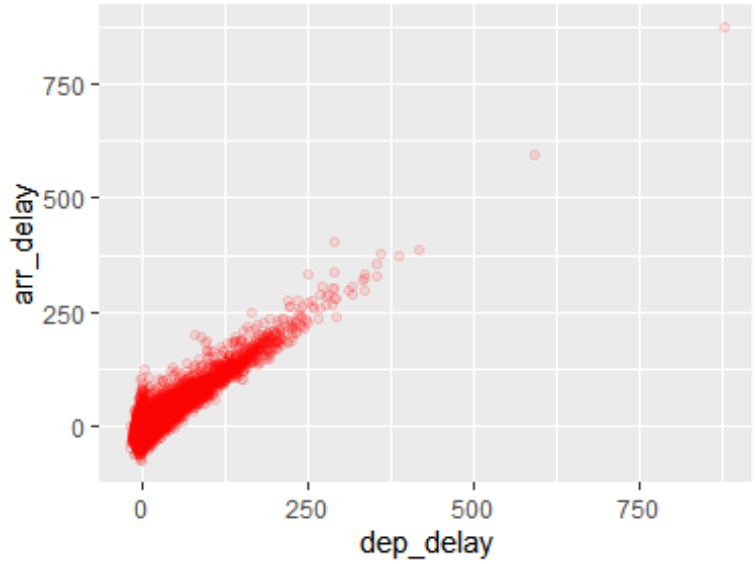
b. Now use `geom_density()` to create a density plot of the departure delays.

c. *Violin plots* are a useful alternative to visualise the overall density of points, when there are too many points to plot individually. Use `ggplot()` and `geom_violin()` to create the following plot:
(tip: you can use the parameter alpha to control the transparency. alpha = 1 is fully opaque, alpha = 0 is fully transparent)



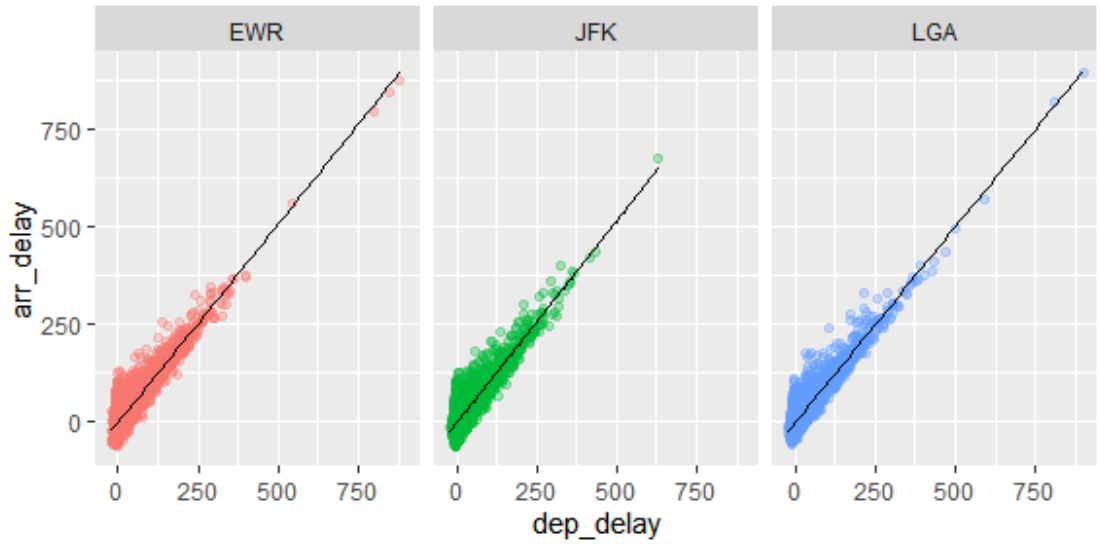
d. Scatter plots are also useful to visualise things such as correlations. Use a combination of filtering and sampling (from `dplyr`), `ggplot()` and `geom_point()` to generate a scatterplot of all departure delays (not only the filtered ones from the previous items) originating only from the airport "EWR". Randomly sample only 10%

of this data. Colour your points in red and use a low value of `alpha` to create a plot similar to the one below.



Q6. Trend lines and facetting

a. Generate the following plot using `ggplot()`, `geom_point()`, `geom_smooth(..., method="lm", se = FALSE)` and `facet_wrap()` functions. What are the visual cues being used within this plot? Based on the plot below, what can we say about the relationship between the departure and arrival delays? (This is easier to do with smaller data sizes, so feel free to use a downsampled dataset like you did in earlier questions)

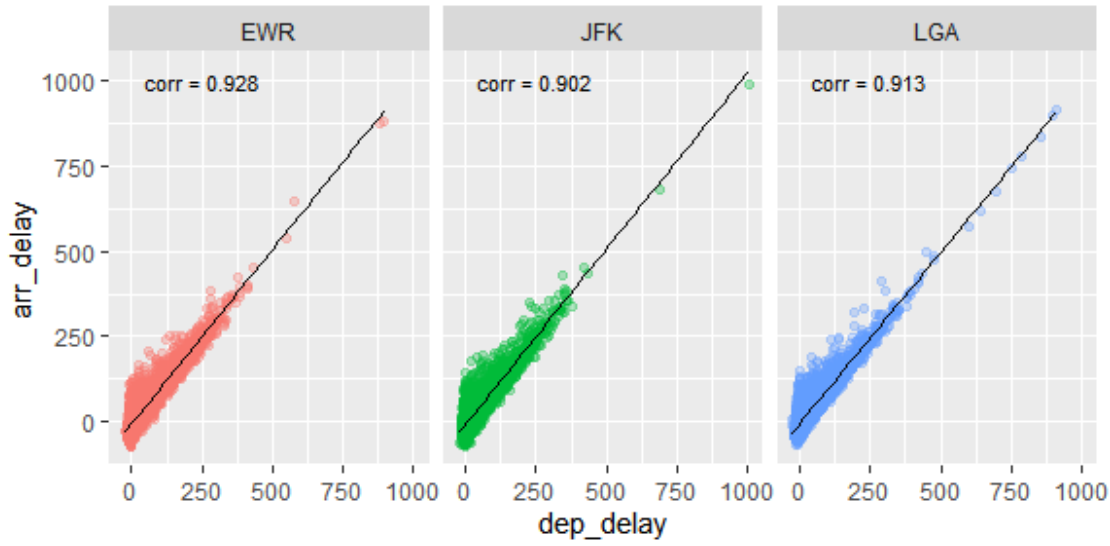


b. Compute the Spearman's rank correlation between the departure and arrival delays for each airport of origin. You can do that using `cor(..., method = "spearman",`

use = "pairwise.complete.obs") in combination with dplyr's group_by(). You should get something like this:

```
## # A tibble: 3 × 2
##   origin   cor
##   <chr>  <dbl>
## 1 EWR    0.928
## 2 JFK    0.902
## 3 LGA    0.913
```

c. Now adapt the code you produced in Q6 to add an annotation of the correlation values to each of the the facets. In the end you should get something like the plot below: *(Tip: this is trickier than it may sound. Remember that you can add new geoms to a plot using different dataframes as sources, if they have consistent column names.)*



(Extra)

Take 10 minutes to explore the R graph Gallery – there is a wealth of ideas for data visualisation, together with example ggplot2 code that you can adapt for your own projects.

Part III: Conceptual/theory Questions (optional)

Q7.

Define the following terms clearly:

- Point estimator
- Point estimate
- Sampling distribution

Q8.

State the difference between **bias** and **variance** of an estimator. Why might we sometimes prefer a slightly biased estimator over an unbiased one?

Q9.

Explain what the **Mean Squared Error (MSE)** of an estimator is, and show how it can be decomposed into variance and squared bias.

Q10.

What is the **Central Limit Theorem (CLT)**, and why is it so important for point estimation?
