Chapter 1
Introduction

1.1 Introduction to the Training

1.1.1 Course Aims

This workshop has been designed to help Shiny developers who are looking to put their apps into production understand the range of tools now available to make this process easier. By the end of this workshop attendees will be able to build automated test suites for their applications, identify bottlenecks and be aware of solutions to common problems, and start load testing applications ahead of deployment.

1.1.2 Course Materials

Items appearing in this material are sometimes given a special appearance to set them apart from regular text. Here's how they look:

```
> This is a section of code              # This is a comment
```

A warning, typically describing non-intuitive aspects of the R language

A tip: additional features of R or "shortcuts" based on user experience

Exercises to be performed during (or after) the training course

1.1.3 Course Script and Exercise Answers

Throughout this workshop the trainer will be using a range of shiny applications to demonstrate key concepts. You will be able to get all of these files from the MangoTheCat GitHub pages.

1.2 Example Applications

In order to demonstrate the functionality that we will see throughout this workshop we have produced two small, sample shiny applications that capture some of the most common problems we encounter.

1.2.1 Average Flight Delays by Airport

This application, which we will use for examples in the materials, allows the user to select one of three US airports and then creates a visualisation of flight arrival delays against departure delays from that airport. The app also creates a table showing median arrival delays.

There are a number of features of this application that mean that it is slow to run, and would generally be considered unacceptable to end users to deal with. We will show how we can resolve these issue and make our app scalable.

```
library(shiny)
library(ggplot2)
library(dplyr)

ui <- fluidPage(

  fluidRow(
    selectInput("airport", "Select Origin Airport",
                choices = list("New York, JFK" = "JFK",
                               "San Francisco Int." = "SFO",
                               "Chicago O'Hare" = "ORD"))
  ),

  fluidRow(
    column(width = 6,
           plotOutput("dep_vs_arr")),
    column(width = 6,
           dataTableOutput("arr_delay"))
  )

)
```

```
server <- function(input, output, session){

  flights <- readr::read_csv("./data/flights08.csv")
  airports <- readr::read_csv("./data/airports.csv")

  output$dep_vs_arr <- renderPlot({

    airport_data <- flights %>%
      filter(Origin == input$airport)

    ggplot(airport_data, aes(DepDelay, ArrDelay)) +
      geom_point() +
      labs(title = input$airport)

  })

  output$arr_delay <- renderDataTable({

    flights %>%
      filter(Origin == input$airport) %>%
      group_by(Dest) %>%
      summarise(`Median Arrival Delay` = median(ArrDelay, na.rm =
TRUE)) %>%
      left_join(airports, by = c("Dest" = "iata")) %>%
      select(airport, `Median Arrival Delay`) %>%
      arrange(desc(`Median Arrival Delay`))

  })

}

shinyApp(ui, server)
```

1.2.2 Factors Causing Delays for Flights Starting in Atlanta

The second application, that will be used in the exercises, allows users to fit a simple linear model to understand how a selection of factors lead to departure delays for flights departing from Atlanta. Users can select a variable which will be plotted, and the model fit will be returned in a small table. In addition, a random selection of departure delays is also visualised.

This application, in general, runs faster than the one above, but may start to see performance issues as the number of concurrent users is significantly increased.

```
library(shiny)
library(dplyr)
library(ggplot2)
library(broom)
library(feather)

ui <- fluidPage(

  sidebarLayout(
    sidebarPanel(
      selectInput(
        "coef", "Select Model Coefficient (x-axis):",
        choices = c("DayOfWeek", "DepTime", "AirTime", "Distance"),
        selected = "Distance"
        ),
      sliderInput(
        "nsamp", "Number of Random Samples:",
        value = 10000, min = 0, max = 20000, step = 1000
        )
      ),
    mainPanel(
      fluidRow(
        column(width = 6,
          plotOutput("delay")
        ),
        column(width = 6,
          plotOutput("model")
        )
        ),
      fluidRow(
        dataTableOutput("coeffs")
        )
      )
    )
)
```

```
server <- function(input, output, session){

  atlanta <- read_feather("flights_sub.feather") %>%
    filter(Origin == "ATL")

  fit_model <- reactive({

    model <- paste0("DepDelay ~", input$coef)

    lm(as.formula(model), data = atlanta)

    })

  output$model <- renderPlot({

    hour_delay <- atlanta %>%
      filter(DepDelay > 60)

    ggplot(data = hour_delay, aes_string(input$coef, "DepDelay")) +
      geom_point() +
      geom_smooth(method = "lm")

    })

  output$delay <- renderPlot({

    random_sample <- atlanta %>%
      sample_n(input$nsamp)

    ggplot(data = random_sample, aes(DepDelay)) +
      geom_histogram()

    })

  output$coeffs <- renderDataTable({

    fit_model() %>%
      tidy()

    })

}

shinyApp(ui, server)
```