Using R to generate data for questions

Craig Alexander & Eilidh Jack

Contents

Chapter 1

Overview

In this session, we will take a look at how we can use R to generate data to create questions similar to those found in the Advanced Higher Statistics Exam papers. This tutorial will work through some examples from Paper 1 from the 2021 exam. You can access the paper and solutions by clicking on the links.

1.1 Libraries

Throughout this tutorial, we will use some libraries within R. If you would prefer to work through the examples on R, you will need to install the following libraries:

- `tidyverse`
- `truncnorm`

Chapter 2

2021 Paper 1 Example

2.1 Question 1

In this example, we will take a look at question 1 from Paper 1 in 2021. This question is a report style question based on Google AI data of times taken to draw a cat or a dog. The example contains a stem and leaf diagram with the combined data and some summary statistics for both sets of drawings. Following this, a Mann-Whitney Test is carried out to test whether both samples have different average drawing times.

We will now look at how we can draw a sample of the data from the question, by randomly sampling data for both groups using properties from their summary statistics

2.1.1 Generating a random sample of data

We can generate a random sample of data for both the categories using the summary statistics provided. As we can see from the stem and leaf diagram, the data have a lower bound, where we cannot observe any data below zero, as the data recorded are based on time elapsed.

In order to sample data of this form, we can use a variation of the Normal distribution called the truncated normal distribution, which allows us to bound a Normal distribution through a given range.

To randomly sample data from this distribution, we can use the `rtruncnorm` function from the `truncnorm` package in R as follows:

```r
sample_dog <- rtruncnorm(n=145, a=0, b=16, mean=7.5, sd=2.66)
sample_cat <- rtruncnorm(n=121, a=0, b=14, mean=5.4, sd=2.31)
```

The parameters required are defined as follows:

- `n` - the number of samples we wish to draw
- `a` - the lower bound of the distribution (here, we will set this to 0)
- `b` - the upper bound of the distribution (here, we have set this to be the ceiling of the max value for each group)
- `mean` - the mean of each group
- `sd` - the standard deviation of each group

Running the code above will produce a sample for both groups based on their relative summary statistics. We can check the summary statistics of our data using `summary()`

```
summary(sample_dog)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.260   5.420   6.979   7.172   8.846  14.280
```

```
summary(sample_cat)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5993  4.5442  5.9961  5.8839  6.9648 12.4311
```

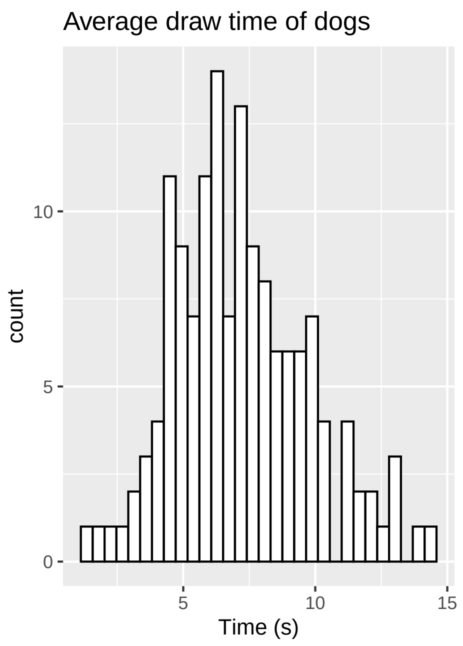
2.1.2 Visualising the data

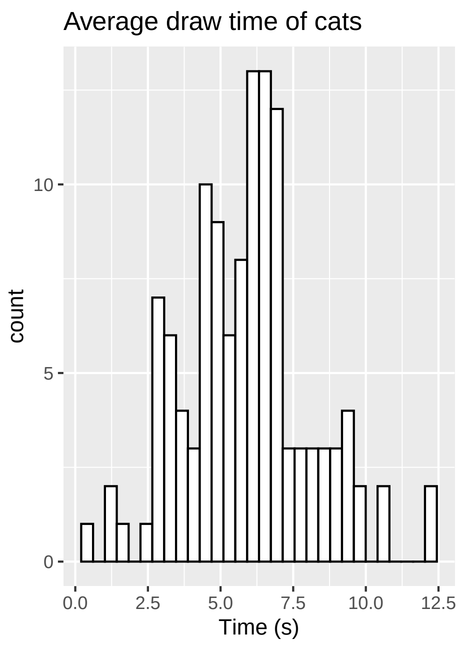
We can also check the distribution of our sampled data for comparison by visualising it using a histogram. The `ggplot2` library found in the `tidyverse` library provides several functions for data visualisation and has become more popular than base R graphics. We will use the `geom_histogram()` function from the library in this example as follows:

```
dog_hist <- ggplot(data.frame(sample_dog),aes(x=sample_dog)) + geom_histogram(color="bl
            labs(title="Average draw time of dogs",x="Time (s)")
cat_hist <- ggplot(data.frame(sample_cat),aes(x=sample_cat)) + geom_histogram(color="bl
  labs(title="Average draw time of cats",x="Time (s)")

grid.arrange(dog_hist,cat_hist,ncol=2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```





To create these histograms, the code above works in the following fashion:

- We first specify our data using `ggplot(data)`, where our data here is either group of samples.
- To specify which variables we would like to select, we use the `aes()` argument. As we only have one sample of data in each case, we specify this using `x=data`.
- We then generate the histogram using `geom_histogram()`, where we can define the line colour using `color` and the filled colour of the bars using `fill`.
- We can label our plot using the `labs` argument, where we can include a `title` and an `x` axis label

We can also alter the number of bins we use (`ggplot` will set a standard number of bins by default) using the `bins` argument. Let's alter the number of bins for the dog data to be 10

```
dog_hist <- ggplot(data.frame(sample_dog),aes(x=sample_dog)) + geom_histogram(color="black",fill=
  labs(title="Average draw time of dogs",x="Time (s)")
```

2.1.3 Setting random seeds for reproducability

When we randomly sample data each time in R, we will obtain a different sample than before. Let's run our previous code twice to see if there is any differences:

```
sample_dog1 <- rtruncnorm(n=145, a=0, b=16, mean=7.5, sd=2.66)
sample_dog2 <- rtruncnorm(n=145, a=0, b=16, mean=7.5, sd=2.66)

summary(sample_dog1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4464  5.7736  7.3936  7.5150  9.2837 13.4899
```

```
summary(sample_dog2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.239   5.346   7.562   7.454   9.250  13.457
```

We see that both samples produce different summary statistics. This can cause difficulty when you are working on a specific problem and want to design questions around the specific characteristics of the data you have sampled the first time.

We can force R to use the same random number generation by using the `set.seed()` function. Here, we specify the seed from the random number generator we want to use each time we generate samples. This number can be any number you wish to choose! The example below highlights how this works:

```
set.seed(2023)
sample_dog1 <- rtruncnorm(n=145, a=0, b=16, mean=7.5, sd=2.66)
set.seed(2023)
sample_dog2 <- rtruncnorm(n=145, a=0, b=16, mean=7.5, sd=2.66)

summary(sample_dog1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.006   6.316   7.545   7.798   9.365  14.776
```

```
summary(sample_dog2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.006   6.316   7.545   7.798   9.365  14.776
```

Here, we see we can produce the same data as the first sample by setting the seed prior to sampling.

2.2 Regression