

# Introduction to randomisation, bootstrap, and Monte Carlo methods in R

[https://stirlingcodingclub.github.io/randomisation/  
randomisation\\_notes.html](https://stirlingcodingclub.github.io/randomisation/randomisation_notes.html)

Brad Duthie

18 September 2019

## Randomisation in R for statistical analyses

**Being able to generate random numbers or sample random subsets of a list is very useful for all types of modelling**

# Randomisation in R for statistical analyses

**Being able to generate random numbers or sample random subsets of a list is very useful for all types of modelling**

1. The R programming languages includes many base functions for random number and list sampling
  - ▶ `rnorm`, `runif`, `rbinom`, `rpois`
  - ▶ `sample`

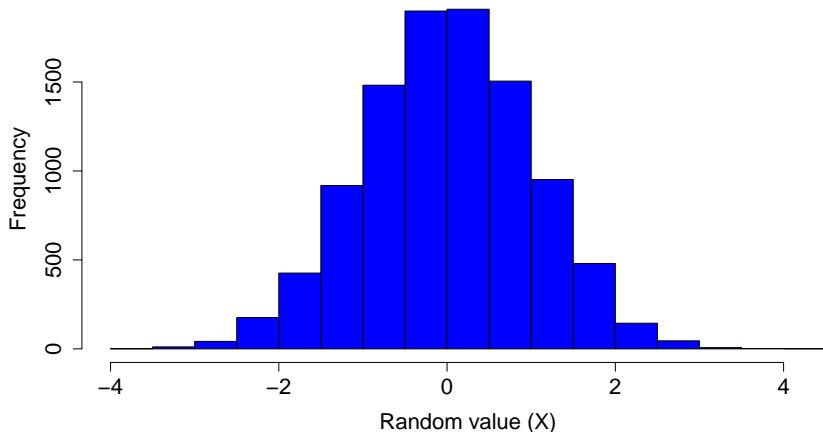
# Randomisation in R for statistical analyses

**Being able to generate random numbers or sample random subsets of a list is very useful for all types of modelling**

1. The R programming languages includes many base functions for random number and list sampling
  - ▶ `rnorm`, `runif`, `rbinom`, `rpois`
  - ▶ `sample`
2. Look at each of these functions briefly, then show how they can be used in frequentist statistics
  - ▶ Hypothesis testing (i.e., getting p-values)
  - ▶ Confidence intervals

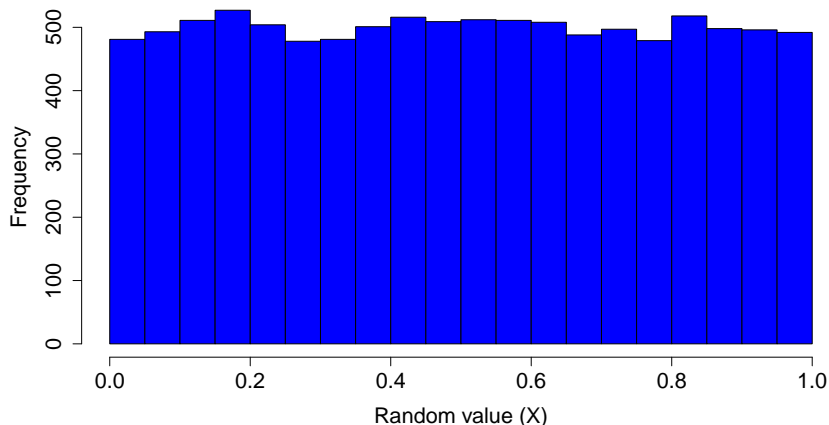
## Random sampling using base functions in R: rnorm

```
random_norms <- rnorm(n = 10000, mean = 0, sd = 1);
```



## Random sampling using base functions in R: runif

```
rand_unifs <- runif(n = 10000, min = 0, max = 1);
```



## Random sampling using base functions in R: rbinom

The `rbinom` function is a bit more tricky.

- ▶ `n`: Number of observations
- ▶ `size`: Number of independent trials *within an observation*
- ▶ `prob`: Probability that a trial is successful

If we want to flip one fair coin 1000 times:

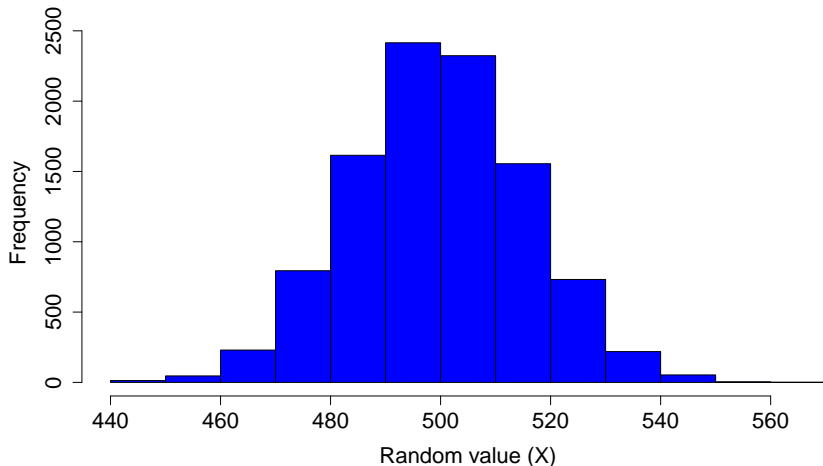
```
rand_binoms <- rbinom(n = 1, size = 1000, prob = 0.5);
```

If we want to flip 10 fair coins 1000 times:

```
rand_binoms <- rbinom(n = 10, size = 1000, prob = 0.5);
```

## Random sampling using base functions in R: rbinom

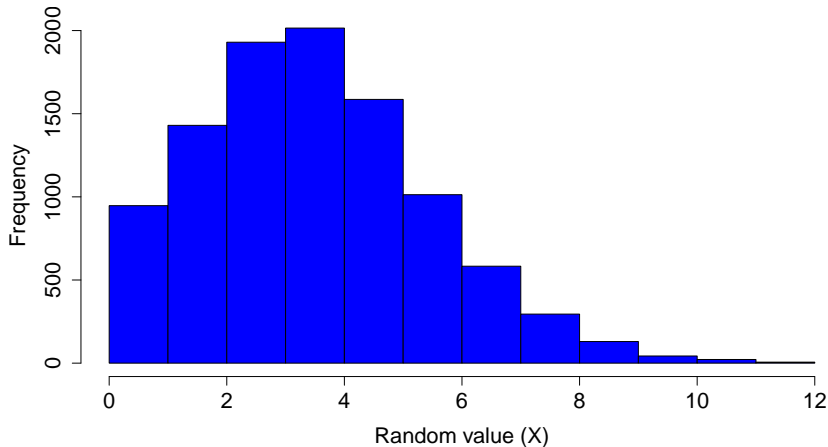
```
rand_binoms <- rbinom(n = 10000, size = 1000, prob = 0.5);
```





## Random sampling using base functions in R: rpois

```
rand_poissons <- rpois(n = 10000, lambda = 4);
```



# The logic of randomisation for frequentist statistics

**Logic of frequentist statistics becomes more intuitive when using randomisation and resampling methods**

# The logic of randomisation for frequentist statistics

**Logic of frequentist statistics becomes more intuitive when using randomisation and resampling methods**

- ▶ **P-value:** For a particular statistical model, given that the null hypothesis is true, what is the probability of getting a statistic as or more extreme than the one observed?

# The logic of randomisation for frequentist statistics

**Logic of frequentist statistics becomes more intuitive when using randomisation and resampling methods**

- ▶ **P-value:** For a particular statistical model, given that the null hypothesis is true, what is the probability of getting a statistic as or more extreme than the one observed?
- ▶ **Confidence interval:** An interval within which the parameter mean would be expected to be contained with some confidence (e.g., 95%) if the population were repeatedly resampled

# The logic of randomisation for frequentist statistics

**Logic of frequentist statistics becomes more intuitive when using randomisation and resampling methods**

- ▶ **P-value:** For a particular statistical model, given that the null hypothesis is true, what is the probability of getting a statistic as or more extreme than the one observed?
- ▶ **Confidence interval:** An interval within which the parameter mean would be expected to be contained with some confidence (e.g., 95%) if the population were repeatedly resampled

**By using randomisation and re-sampling, we can do statistical tests using a process that better reflects the logic underlying the frequentist approach.**

## The basic idea: a t-test

```
data(iris); # Remove one of the three species below  
dat <- iris[iris[,5] != "setosa",];  
dat <- dat[,c(1, 5)]; # Remove unneeded columns
```

|    | Sepal.Length | Species    |
|----|--------------|------------|
| 51 | 7.0          | versicolor |
| 52 | 6.4          | versicolor |
| 53 | 6.9          | versicolor |
| 54 | 5.5          | versicolor |
| 55 | 6.5          | versicolor |

- ▶ Two species include *Iris versicolor* and *I. virginica*
- ▶ Want to know difference between sepal lengths is significant

## The basic idea: a t-test

```
t.test(dat[,1] ~ dat[,2], alternative = "two.sided");
```

```
##
```

```
## Welch Two Sample t-test
```

```
##
```

```
## data: dat[, 1] by dat[, 2]
```

```
## t = -5.6292, df = 94.025, p-value = 1.866e-07
```

```
## alternative hypothesis: true difference in means between
```

```
## 95 percent confidence interval:
```

```
## -0.8819731 -0.4220269
```

```
## sample estimates:
```

```
## mean in group versicolor mean in group virginica
```

```
## 5.936 6.588
```

## The basic idea: a t-test

**A different way to state the null hypothesis:** *Species identity does not have any effect on the mean difference between sepal lengths*

1. If we randomly assign species to sepal lengths, then calculate the difference between species means, the size of this difference is due to chance
2. If we repeat step 1 many times, we can generate a null distribution of differences between species means (i.e., what magnitudes of differences between sepal lengths are expected by chance)
3. We can compare the *actual observed* difference between species mean sepal lengths to the null distribution generated in step 2.

Proportion of differences that are as extreme or more extreme than the *actual observed* difference in the null distribution **is a p-value**.



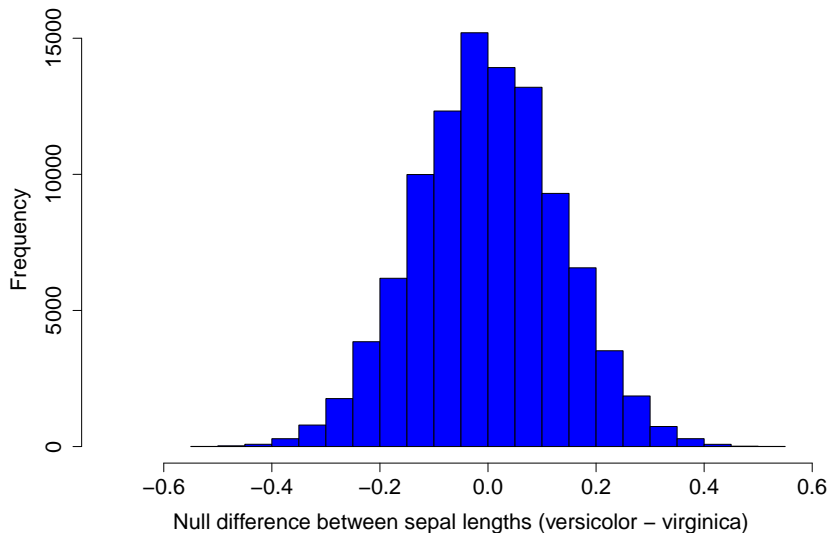
# Randomisation to test for a difference between means

We can code the algorithm 1-3 below

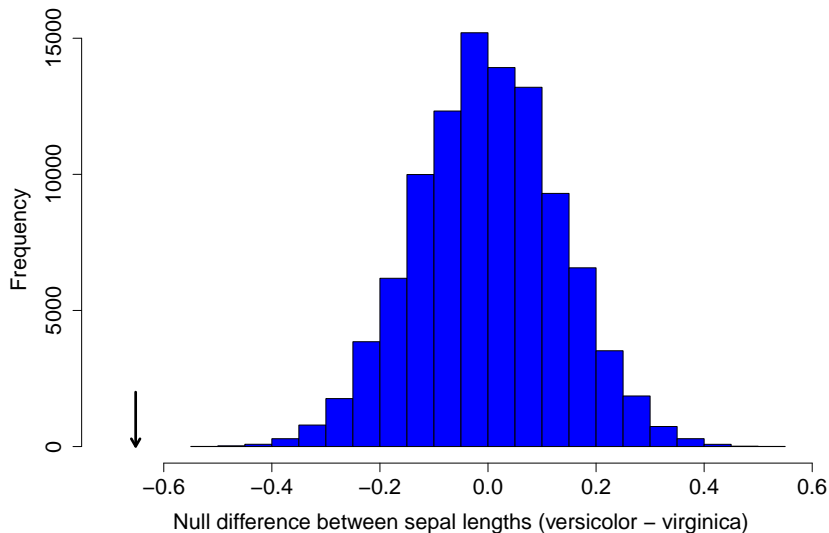
```
iter <- 99999;           # Total iterations
diff <- NULL;
N      <- dim(dat)[1];    # Total number of sepals
for(i in 1:iter){
  sepal_smp <- sample(x = dat[,2], size = N);
  versicolor <- which(sepal_smp == "versicolor");
  virginica <- which(sepal_smp == "virginica");
  mn_samp_ve <- mean(dat[versicolor, 1]);
  mn_samp_vi <- mean(dat[virginica, 1]);
  diff[i] <- mn_samp_ve - mn_samp_vi;
}
```

We now have a null distribution of mean differences between sepal lengths (diff)

## Randomisation to test for a difference between means



## Randomisation to test for a difference between means



## Randomisation to test for a difference between means

The actual difference between versicolor and virginica group means is -0.652.

```
more_extreme <- sum(abs(diff) >= abs(obs_diff));
```

There were 0 values as or more extreme in the null distribution generated with 99999 values

$$P < \frac{0 + 1}{99999 + 1} = 0.00001.$$

This is consistent with the value from `t.test`

## Practice problems

1. Generate a histogram of random numbers sampled from a gamma distribution with a shape parameter ( $k$ ) of 2 and a scale parameter ( $\theta$ ) of 3 using the `rgamma` function in base R.
2. Using the base R data set `trees`, calculate 95% confidence intervals for tree Girth, Height, and Volume. Note, to read in the data, type `data("trees")` in the console. You should then have the data table `trees`, with three columns and 31 rows ([appendix](#) in the [notes](#) has a bootstrap function).
3. Using the base R data set `mtcars`, use a randomisation test to see if there is a significant correlation between car miles per gallon (`mpg`) and car weight (`wt`)

---

**Notes:** [https://stirlingcodingclub.github.io/randomisation/randomisation\\_notes.html](https://stirlingcodingclub.github.io/randomisation/randomisation_notes.html)