R Stats Bootcamp

2.8 - Distributions

Megan Lewis

2025-02-06

R stats bootcamp - Module 2

Schedule:

- ~~Session 7: Explore data~~
- **Session 8: Distributions**
- Session 9: Correlation
- Session 10: Regression
- Session 11: T-test
- Session 12: ANOVA

Session 8 objectives:

- Use of the histogram
- Gaussian: that ain’t normal
- Poisson
- Binomial
- Diagnosing the distribution
- Practice exercises

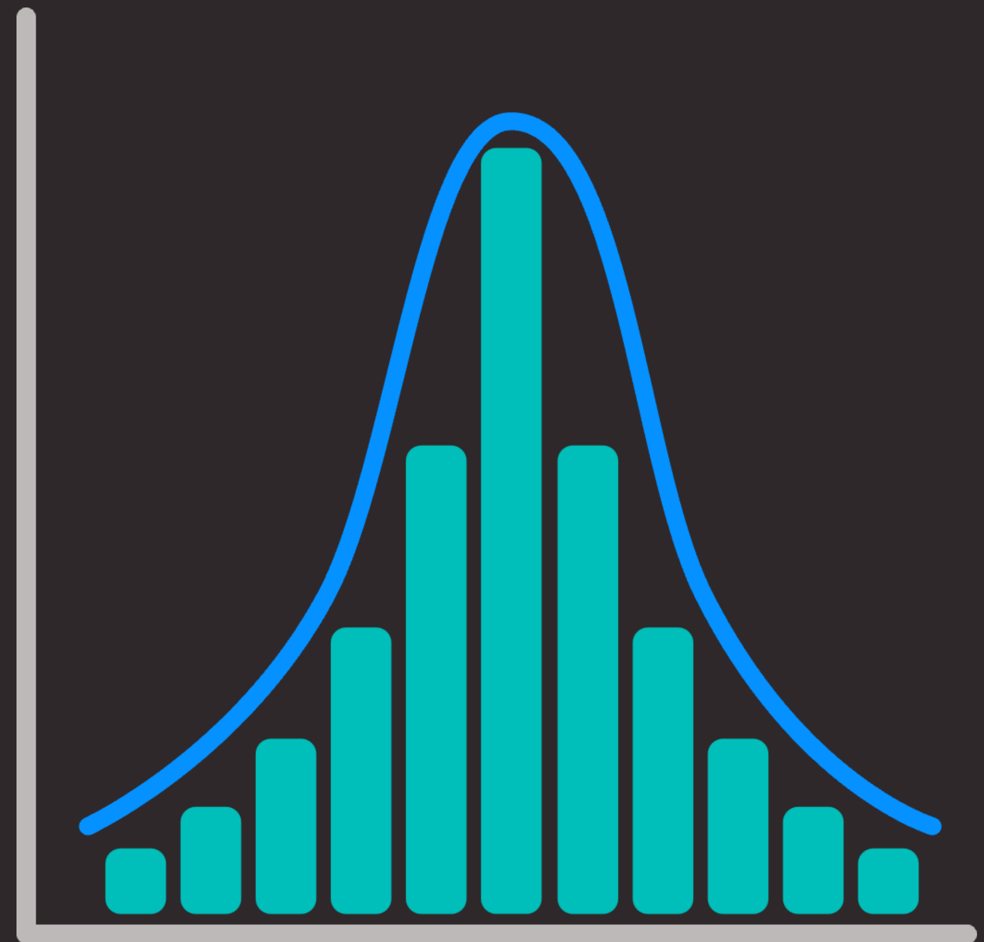
Describing the shape of data

- Sampling underpins traditional statistics
- Population of interest –> Cannot directly measure
- Use of **sampling**
  - Introduces error
  - Real variation
  - Which subjects are in the sample
  - Size of sample
- NHST exploits estimates of these errors
- See bootcamp page for further reading

Exploring diagnostic tools and data distributions

Use of the histogram

- Numeric variable: x axis
- Frequency of observations: y axis
- Can sometimes be proportion of observation



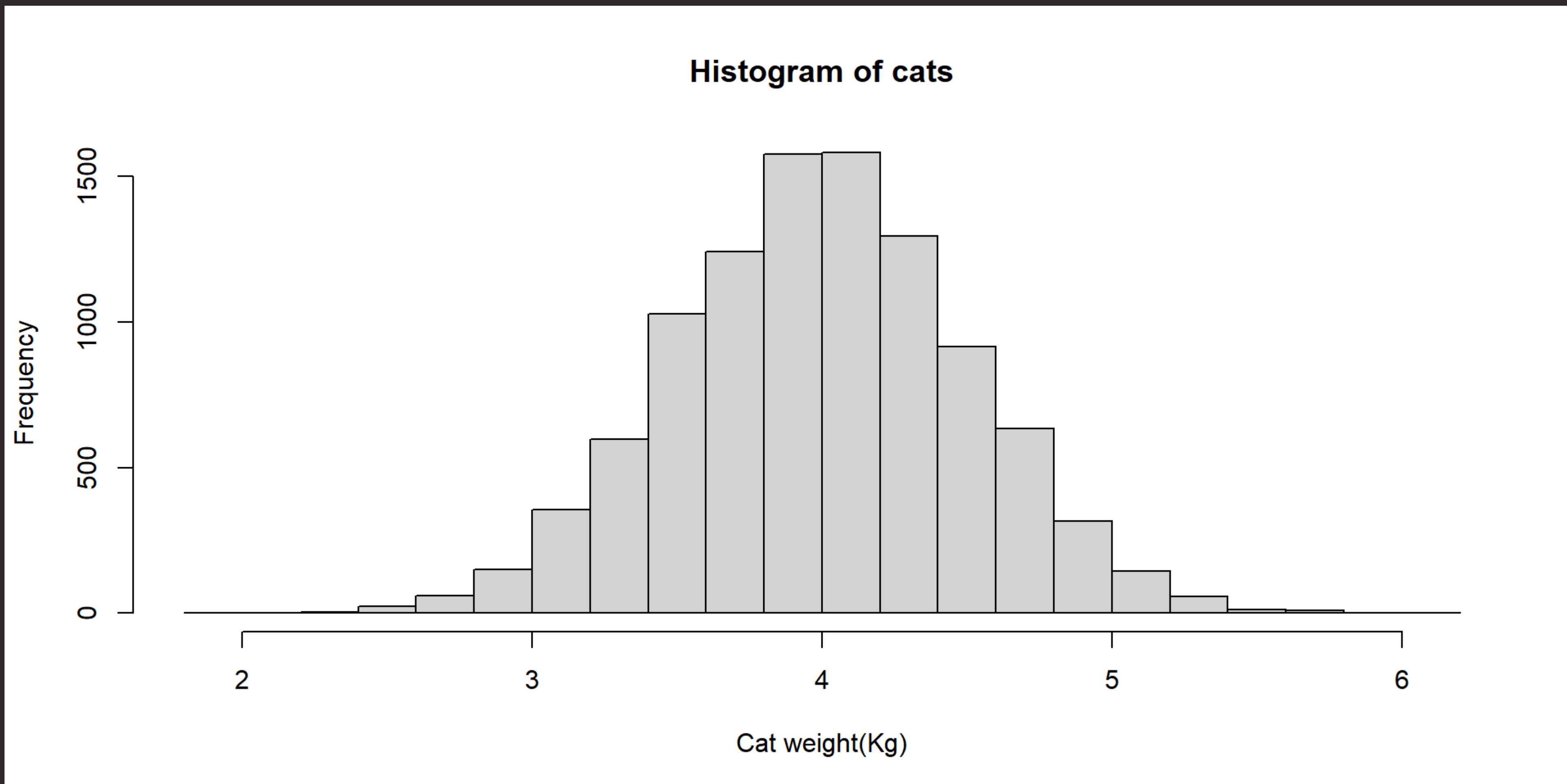
Histogram demo

```r
# Try this:

set.seed(42)

cats <- rnorm(n = 10000, # 10,000 cats
              mean = 4, #mean cat weight of 4kg
              sd = 0.5 # standard deviation of cat weight
              )

cats[1:10] # first ten cats
```

```
 [1] 4.685479 3.717651 4.181564 4.316431 4.202134 3.946938 4.755761 3.952670
 [9] 5.009212 3.968643
```

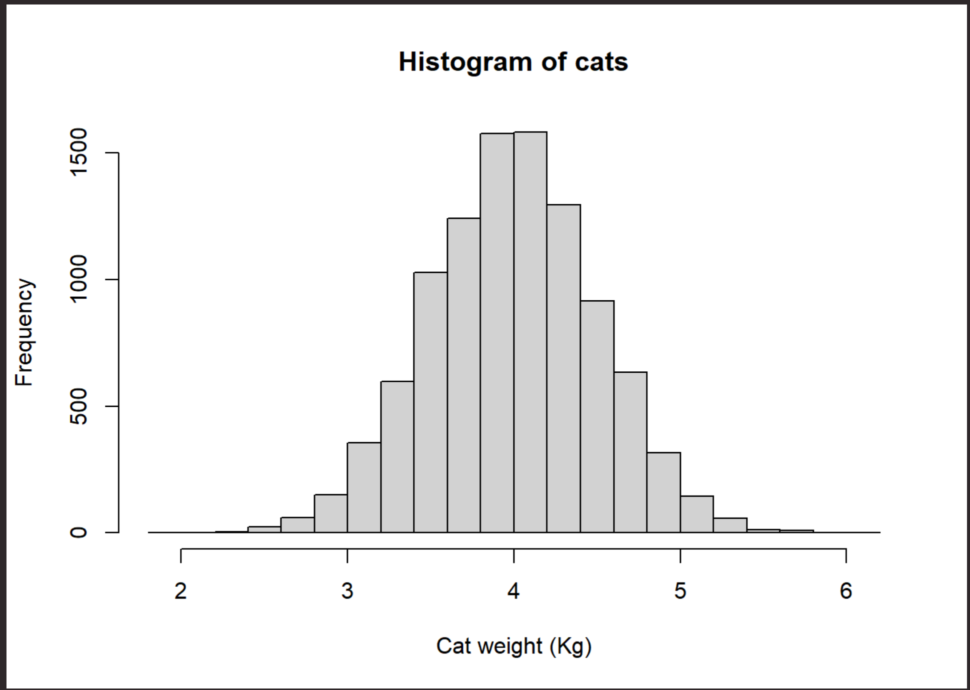
Histogram demo

```r
# Try this:

hist(x = cats,
     xlab = "Cat weight(Kg)")
```



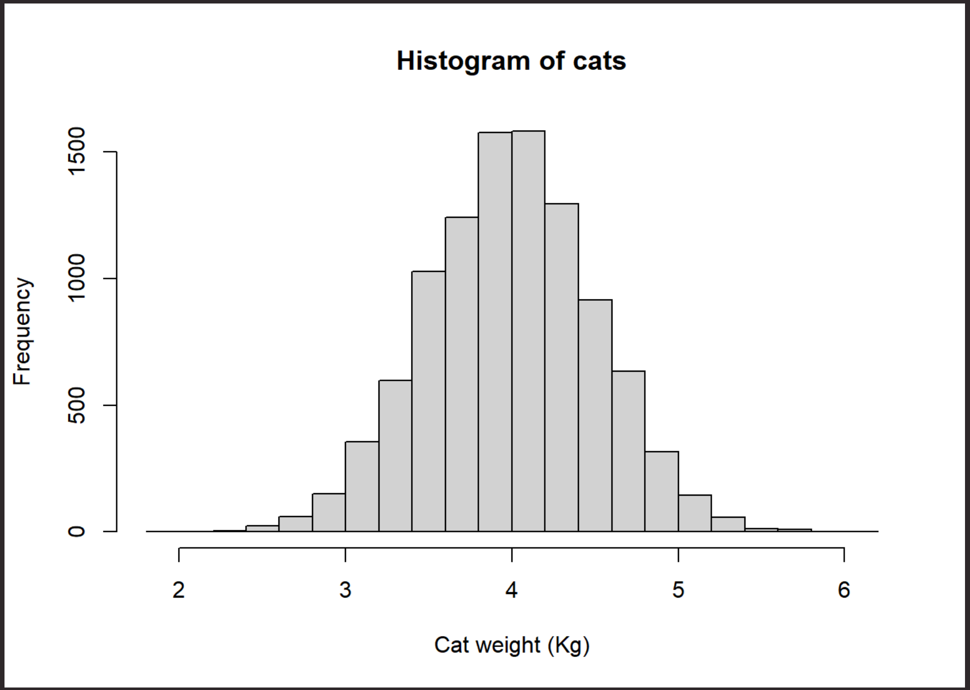
Histogram demo - Domestic cat weight

- Bars are count of number of cat at each weight on x axis
- Width of each column is a range of weights
  - Called “bins”
  - Can define, but usually done automatically based on the data



Histogram demo - Domestic cat weight

- For count data, each bar is usually one or more continuous values
- Shape of histogram can be used to infer the distribution of the data



Sampling and populations

- Imagine:
  - 10,000 cats in whole world & measured all of them
  - Could measure the exact population mean
- Sample of 100 cats:
  - Sample mean likely to differ from real population mean randomly
  - With a bunch of samples, most would be close to real mean
- Can examine with a **simulation of examples**

Sample simulation demo

```r
# Try this:
# We will do a "for loop" with for()

mymeans <- vector(mode = "numeric",
                  length = 100)
mymeans # Empty
```

```
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
 [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
 [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

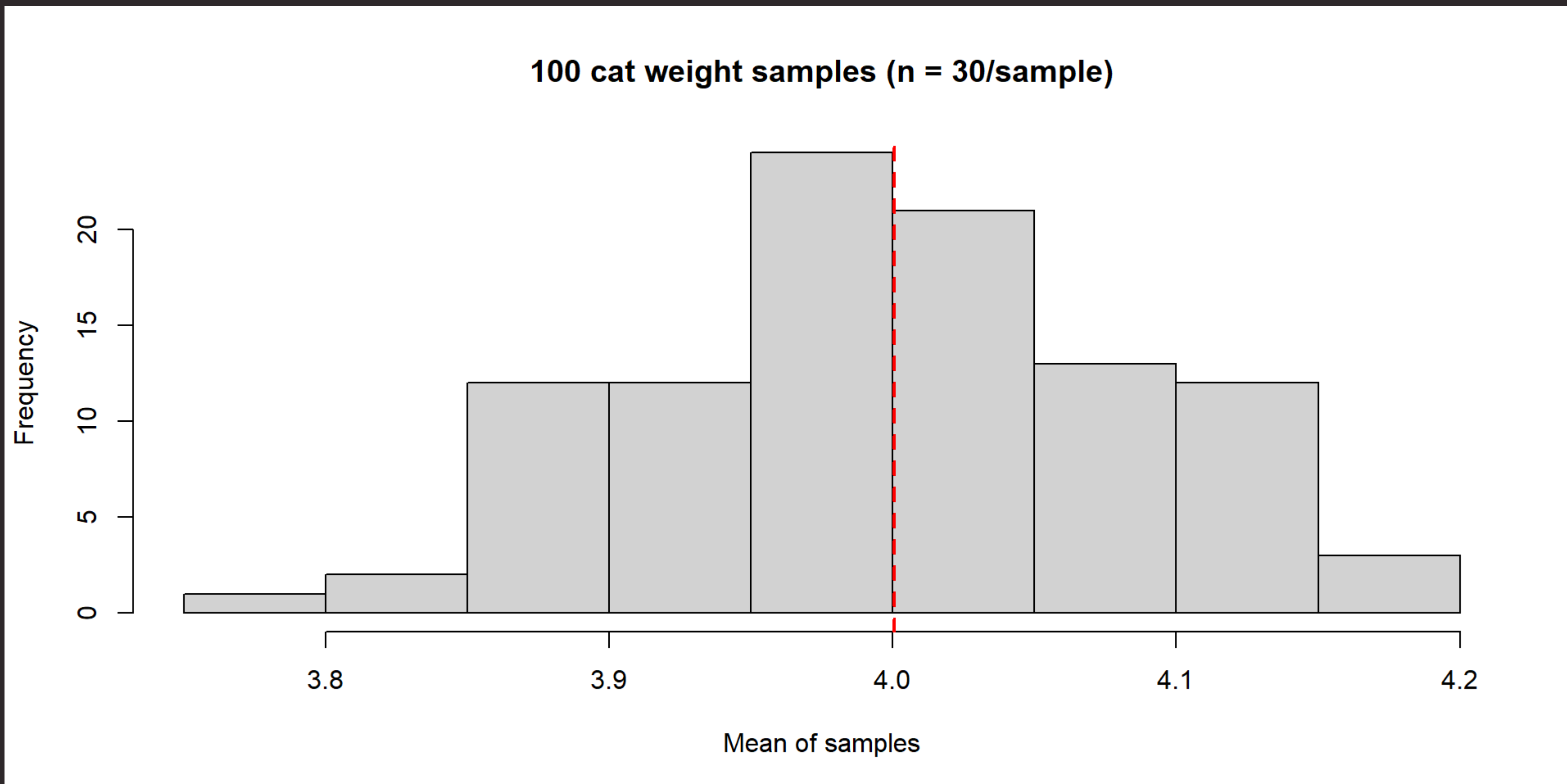
Sample simulation demo

```r
for(i in 1:100){
  mysample <- sample(x = cats, # Takes a random sample
                     size = 30)

  mymeans[i] <- mean(mysample) # stores sample mean in ith vector address
  }

mymeans # Our samples
```

```
 [1] 3.899662 3.963986 4.019242 3.975304 3.903256 3.877346 3.911380 4.013805
 [9] 3.989836 3.950881 4.002299 4.024990 4.091362 4.074493 3.989800 3.974894
[17] 4.000338 3.873104 3.945492 4.061428 4.080559 4.011375 4.023977 3.863940
[25] 4.047250 3.921947 4.049521 4.085961 3.853068 4.081517 3.987747 4.039110
[33] 3.940955 3.954955 4.008512 3.942036 3.955110 3.968722 3.896042 3.979187
[41] 3.957636 4.021170 4.107460 3.989197 3.931964 3.981774 4.125465 4.031625
[49] 4.081076 3.939582 4.185512 3.997635 3.986411 3.817746 4.075256 4.074309
[57] 4.136248 3.926092 3.976513 4.008376 3.984264 3.900717 4.138209 3.913901
[65] 4.123326 3.894216 4.087260 4.145020 3.896286 4.142604 3.865085 4.014336
[73] 4.053620 3.767552 3.981785 4.130483 4.165097 4.046661 4.077928 4.041611
[81] 3.873046 4.100438 4.015098 3.947361 4.029464 4.123772 3.860191 3.820483
[89] 4.071399 4.145585 3.982339 3.950332 3.987406 4.167345 4.126462 4.047683
[97] 4.035958 3.987601 3.960099 3.869092
```

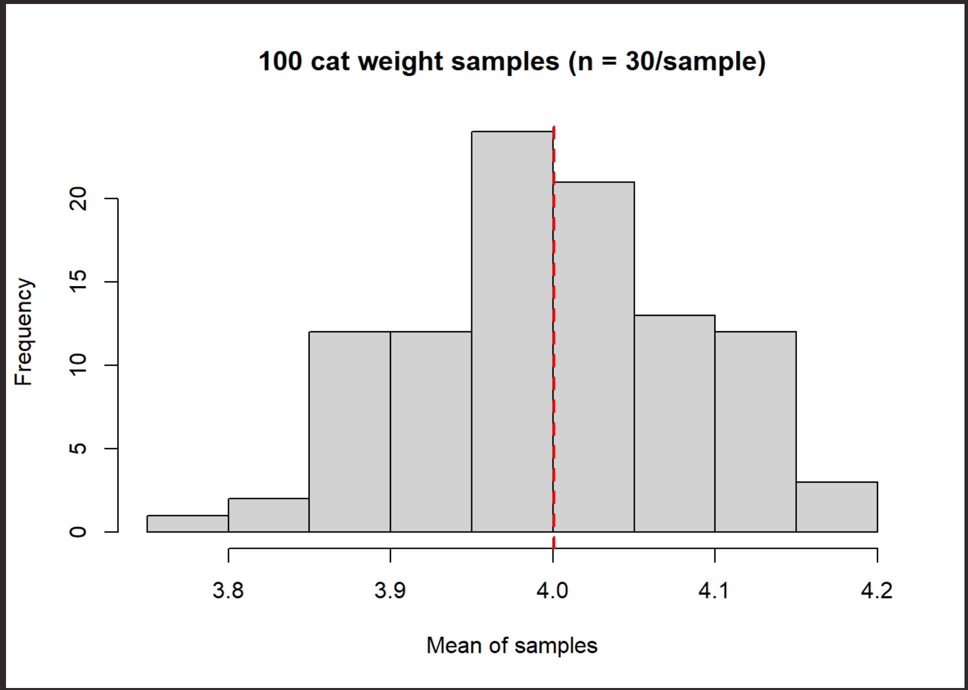
Sample simulation demo

```r
hist(x = mymeans,
     xlab = "Mean of samples",
     main = "100 cat weight samples (n = 30/sample)")
abline(v = mean(mymeans), col = "red", lty = 2, lwd = 2)
```



Sample simulation demo

- Samples vary around the true mean of 4.0kg
- Most samples are pretty close to 4.0, fewer are farther away
- Mean of the means is close to our true population
- *Try simulation a few more times, but vary settings*

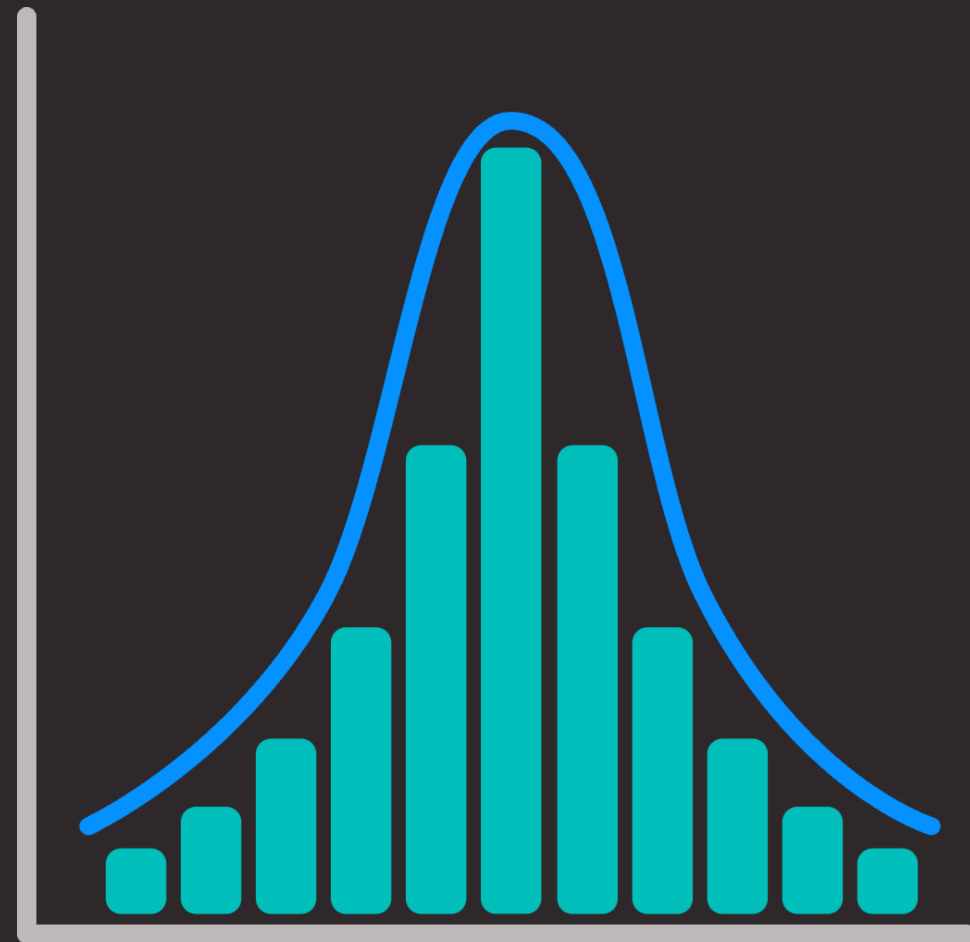


Gaussian: That ain’t necessarily normal

> **Gaussian distribution sometimes referred to as the normal distribution. This is not good practice. Referring to the Gaussian distribution as the normal distribution implies that Gaussian is “typical”, which is patently untrue**
>
> The Comic Book Guy, The Simpsons

Gaussian Distribution

- Classic “Bell curve” shaped distribution
- Probably most important distribution to master: Important in several ways
- Expect **continuous numeric variables**



The Gaussian Assumption

- For linear models like regression and ANOVA
- Assume the *residuals* to be Gaussian distributed
- Must often test and evaluate this assumption
- Described by 2 quantities: the **mean** and the **variance**

Residuals

The difference between each observation and the mean

The Gaussian Assumption

```
# Example data
(myvar <- c(1,4,8,3,5,3,8,4,5,6))
```

```
 [1] 1 4 8 3 5 3 8 4 5 6
```

```
# Mean the "hard" way
(myvar.mean <- sum(myvar)/length(myvar))
```

```
[1] 4.7
```

```
# Mean the easy way
mean(myvar)
```

```
[1] 4.7
```

```
# Variance the "hard" way
# (NB this is the sample variance with [n-1])
(sum((myvar-myvar.mean)^2 / (length(myvar)-1)))
```

```
[1] 4.9
```

```
# Std dev the easy way
sqrt(var(myvar))
```

```
[1] 2.213594
```

Gaussian histograms

- Can describe the expected “perfect” (i.e. theoretical) Gaussian distribution based on just the mean and variance

$$Gaussian\ Parameters : N(\bar{x}, S^2)$$

$$Sample\ mean = \bar{x} = \frac{(x_1 + x_2, + \ldots x_n)}{n}$$

$$Sample\ Variance = S^2 = \frac{\sum(x_i - \bar{x})^2}{n-1} = (std.\,dev.\,)^2$$

$$Sample\ Size = n$$

More Gaussian fun

```r
## Gaussian variations ####
# Try this:

# 4 means
(meanvec <- c(10, 7, 10, 10))
```

```
[1] 10  7 10 10
```

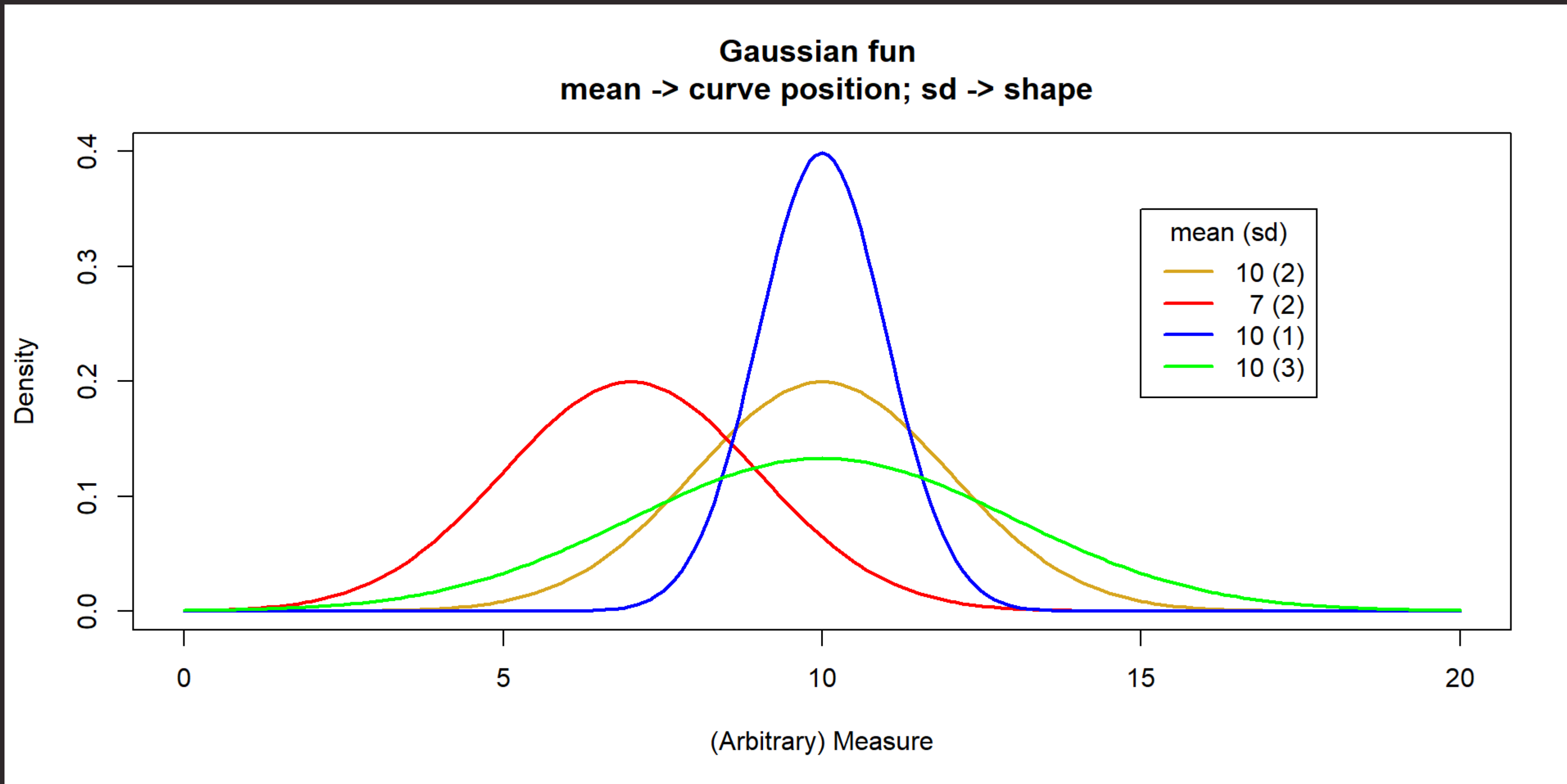
```r
# 4 standard deviations
(sdvec <- c(2, 2, 1, 3))
```

```
[1] 2 2 1 3
```

More Gaussian fun

```r
# Make a baseline plot
x <- seq(0,20, by = .1)

# Probabilities for our first mean and sd
y1 <- dnorm(x = x,
              mean = meanvec[1],
              sd = sdvec[1])

# Baseline plot of 1st mean and sd
plot(x = x, y = y1, ylim = c(0, .4),
     col = "goldenrod", lwd = 2, type = "l",
     main = "Gaussian fun \n mean -> curve position; sd -> shape",
     ylab = "Density", xlab = "(Arbitrary) Measure")

# Make distribution lines
mycol <- c("red", "blue", "green")
for(i in 1:3){
  y <- dnorm(x = x,
                  mean = meanvec[i+1],
                  sd = sdvec[i+1])
  lines(x = x, y = y, col = mycol[i], lwd = 2, type = "l")
}

```

More Gaussian fun



Quartile-Quartile (Q-Q) plots

- Diagnostic tool to test if residuals adhere to assumption of Gaussian distribution
- Plots your data against theoretical expectation of the "quantile" or percentile were your data "perfectly" Gaussian
- Samples are not necessarily expected to perfectly conform to Gaussian (due to sampling error)
- Way to **confront your data with a model**
- Degree to which your data deviates from the line is the degree to which it deviates from Gaussian
  - Especially at the ends of the lines - the tails

Quartile-Quartile (Q-Q) plots

```r
library(car) # Might need to install {car}
# Set graph output to 2 x 2 grid
# (we will set it back to 1 x 1 later)
par(mfrow = c(2,2))
```

Quartile-Quartile (Q-Q) plots

```r
# Small Gaussian sample
set.seed(42)
sm.samp <- rnorm(n = 10,
                 mean = 10, sd = 2)

qqPlot(x = sm.samp,
       dist = "norm", # C'mon guys, Gaussian ain't normal!
       main = "Small sample Gaussian")
```

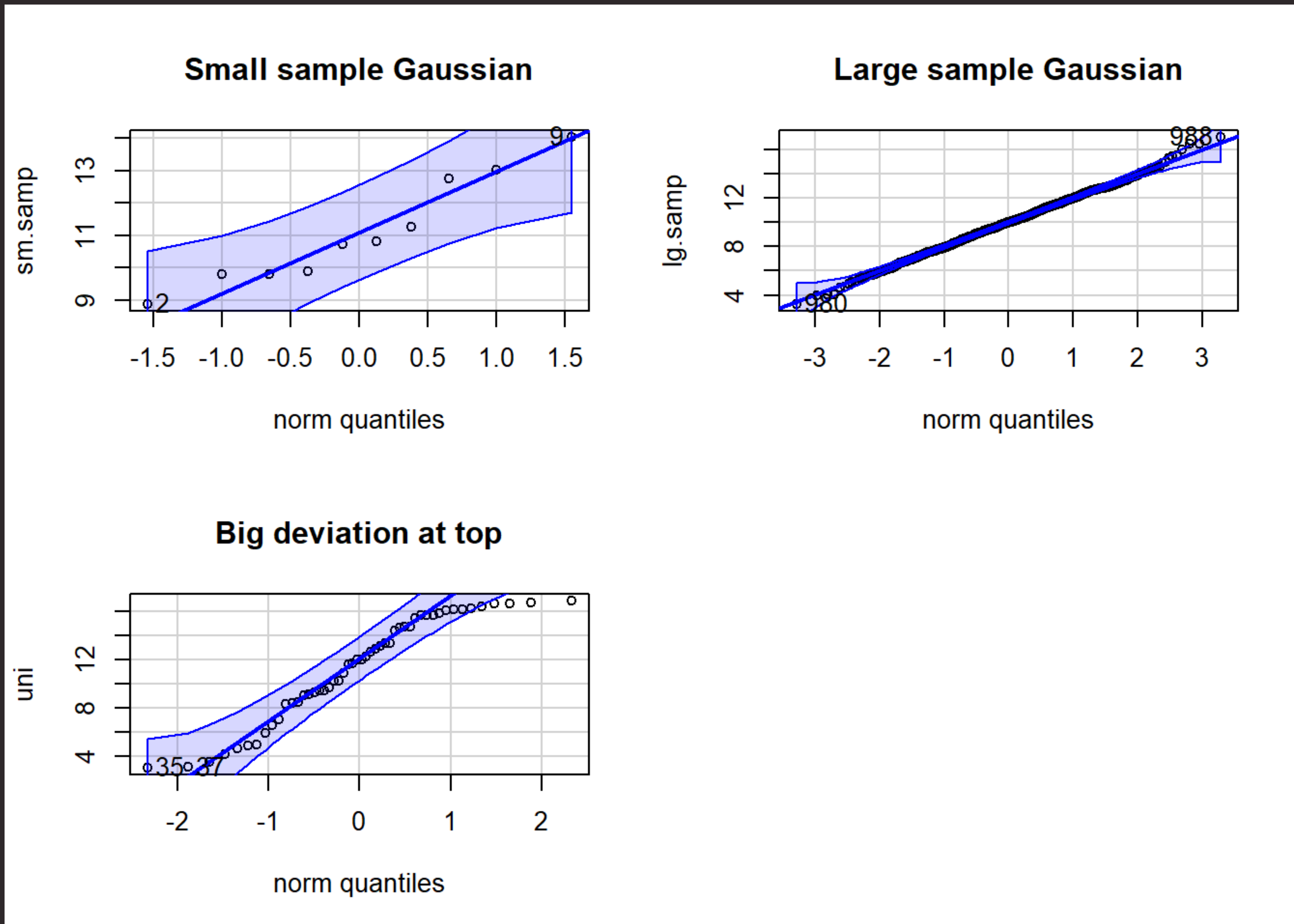
Quartile-Quartile (Q-Q) plots

```r
# Large Gaussian sample
set.seed(42)
lg.samp <- rnorm(n = 1000,
                 mean = 10, sd = 2)

qqPlot(x = lg.samp,
       dist = "norm",
       main = "Large sample Gaussian")
```

Quartile-Quartile (Q-Q) plots

```r
# Non- Gaussian sample
set.seed(42)
uni <- runif(n = 50,
                  min = 3, max = 17)

qqPlot(x = uni,
       dist = "norm",
       main = "Big deviation at top")
```

Quartile-Quartile (Q-Q) plots



Poisson Distribution

Life is good for only two things, discovering mathematics and teaching mathematics

– Simeon-Denis Poisson

Poisson Distribution

- Classic example for use is count data
- Famously exemplified by the number of Prussian soldiers who were killed by being kicked by a horse in a particular year.

Possion Distribution

- Count data of discrete events
  - i.e., Objects etc.
- Integers
  - i.e., Number of beetles caught each day in a pitfall trap

Poisson Distribution

```
rpois(20, 4)
```

```
[1] 3 3 3 5 1 5 5 2 3 4 5 9 5 4 6 2 3 6 5 3
```

- Poisson data typically skewed to the right
- Described by a single parameter, λ (lambda)
  - λ describes the mean **and** the variance

The Poisson parameter

$$Poisson\ Distribution : Pois(\lambda)$$

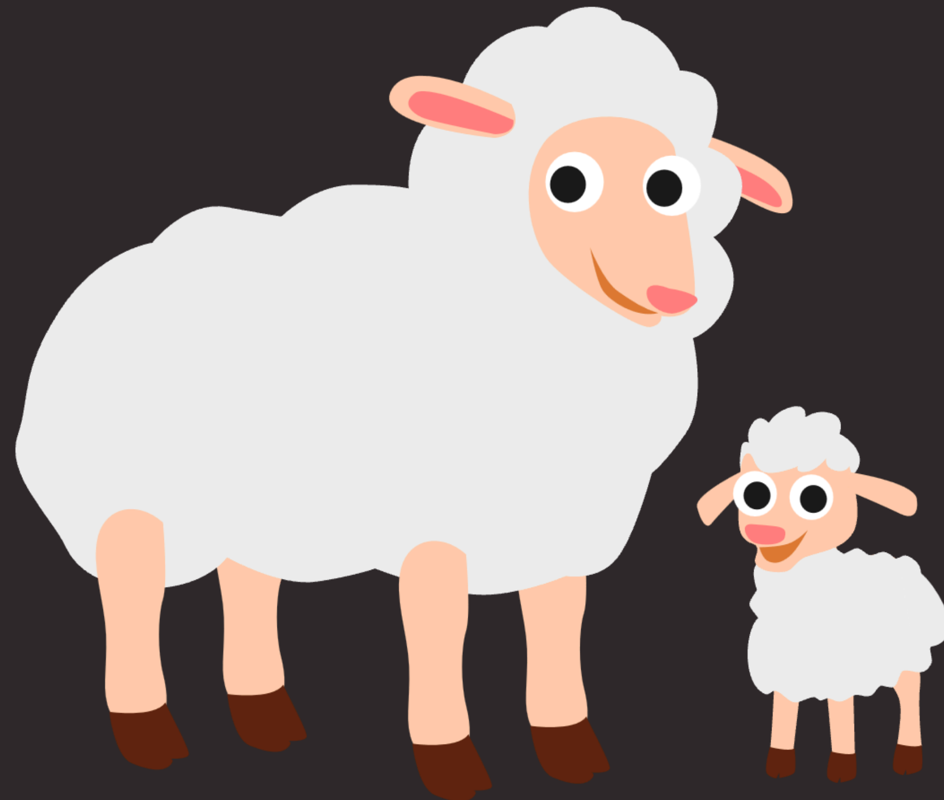
$$Poisson\ parameter \lambda = \bar{x} = S^2$$

$$Sample\ mean = \bar{x}$$

$$Sample\ variance = S^2$$

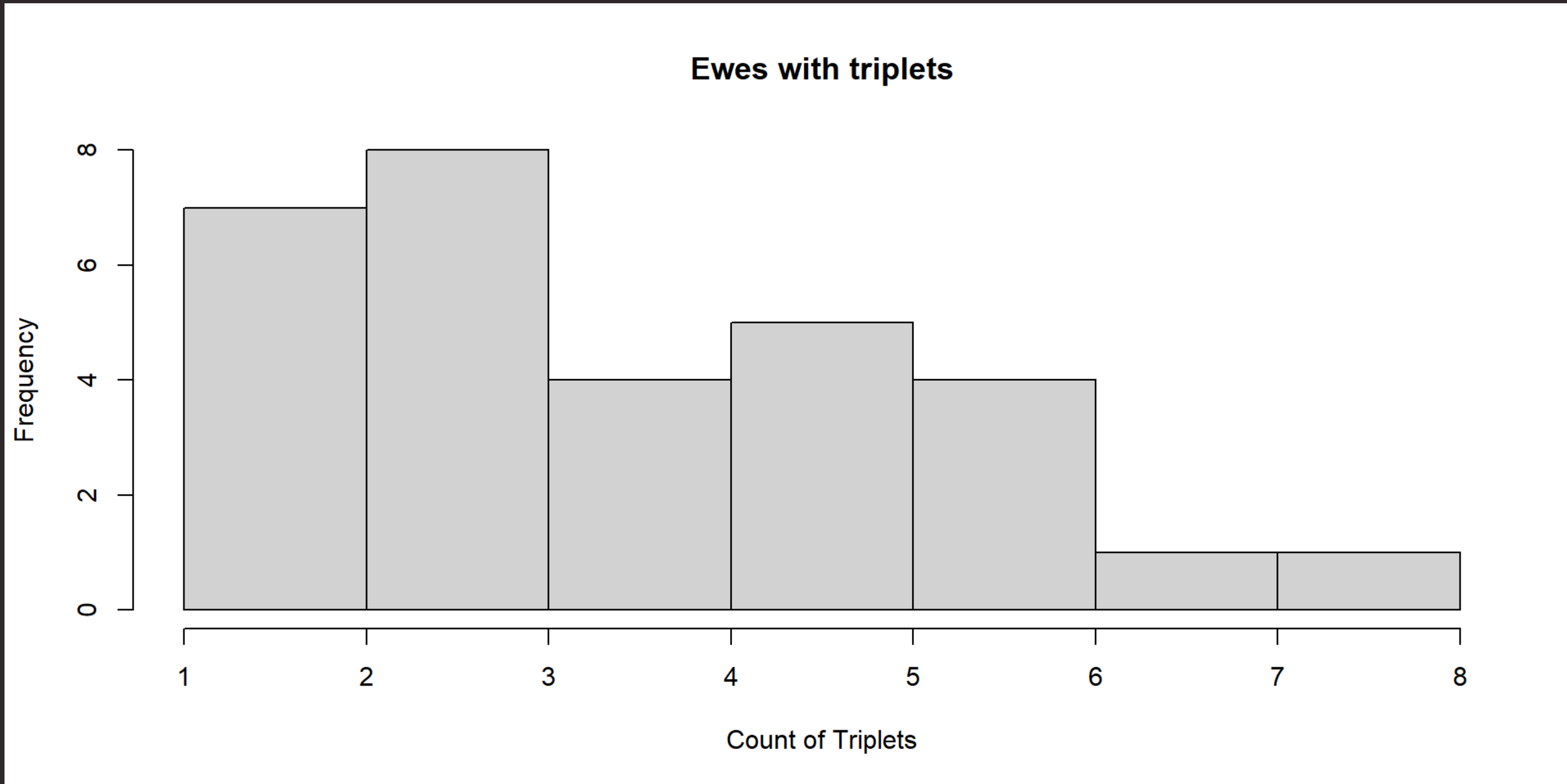
Poisson Data R Demo

- Number of ewes giving birth to triplets (Simulated)
- The counts were made in one year 1n 100 similar flocks (<20 ewes each)



Poisson Data Demo

```r
set.seed(42)
mypois <- rpois(n = 30, lambda = 3)

hist(mypois,
     main = "Ewes with triplets",
     xlab = "Count of Triplets")
```



Ewes with triplets
Frequency
0
2
4
6
8
1
2
3
4
5
6
7
8
Count of Triplets

Density plot for different Poisson lambda values

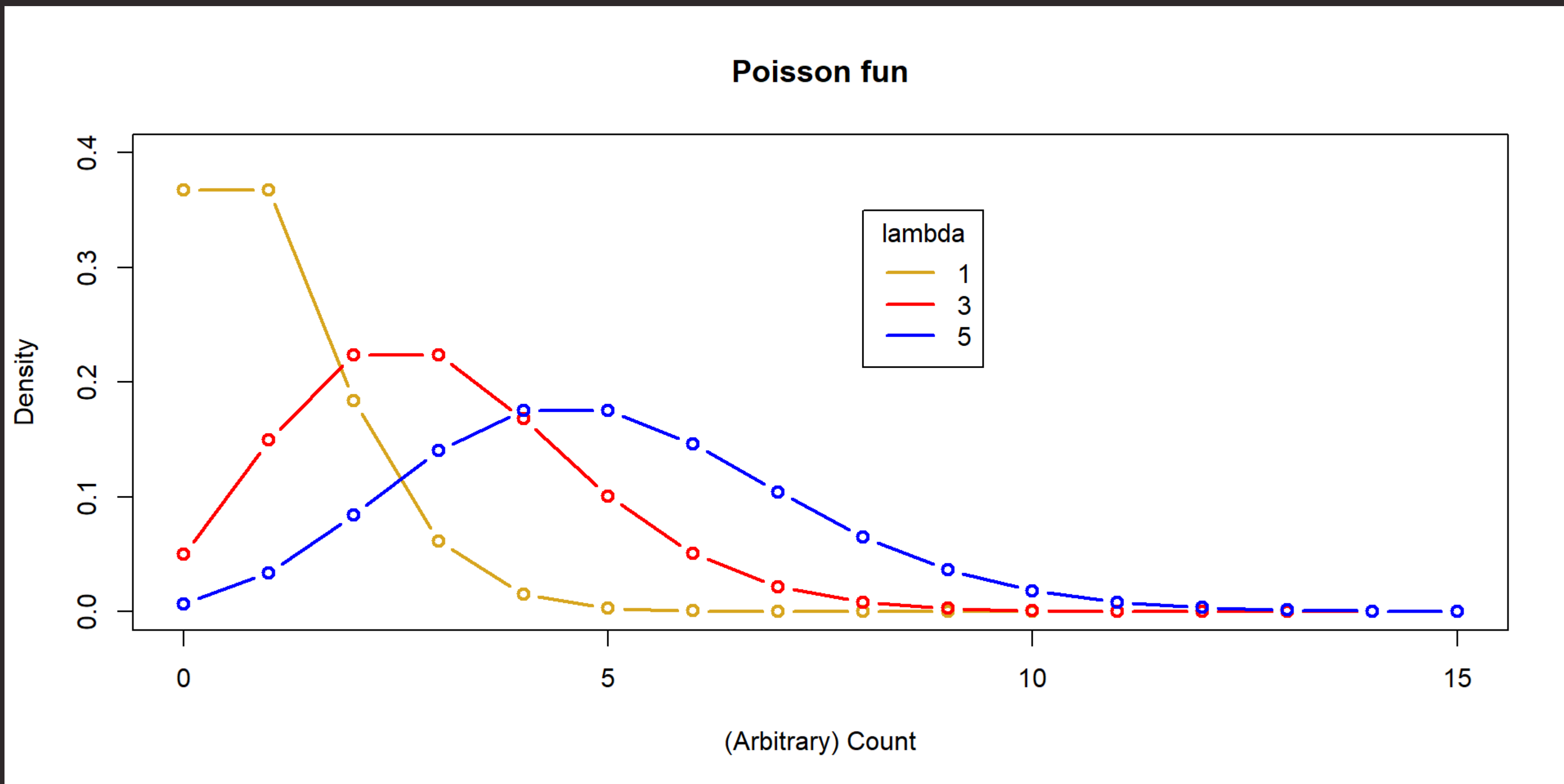
```r
# Try this:
# 3 lambdas
(lambda <- c(1, 3, 5))
```

```
[1] 1 3 5
```

```r
# Make a baseline plot
x <- seq(0, 15, by = 1)

# Probabilities for our first lambda
y1 <- dpois(x = x, lambda = lambda[1])

# Baseline plot Pois
plot(x = x, y = y1, ylim = c(0, .4),
     col = "goldenrod", lwd = 2, type = "b",
     main = "Poisson fun",
     ylab = "Density", xlab = "(Arbitrary) Count")

# Make distribution lines
mycol <- c("red", "blue")
for(i in 1:2){
```

```r
  y <- dpois(x = x, lambda = lambda[i+1])
  lines(x = x, y = y, col = mycol[i], lwd = 2, type = "b")
}

# Add a legend
legend(title = "lambda",
       legend = c("1", "3", "5"),
       lty = c(1,1,1,1), lwd = 2, col = c("goldenrod", "red", "blue"),
```

Density plot for different Poisson lambda values



Binomial Distribution

When faced with 2 choices, simply toss a coin. It works not because it settles the question for you, but because in that brief moment when the coin is in the air you suddenly know what you are hoping for

Binomial Distribution

- Describes data that has exactly **two** outcomes
  - 0 and 1, Yes and No, True and False etc.
- Examples include:
  - Flipping a coin (heads or tails)
  - Successful germination of seeds (success or failure)
  - Binary behavioral decisions (remain or disperse)

Binomial Distribution

- Data are the count of “success” in (binary) outcomes of a series of independent events
- Data coding can be variable
- Example would be success or failure while surveying for wildlife…

Binomial Nest Box Example

- Say you check 50 nest boxes
- One results per nest box
  - i.e., is the box occupied or not (yes or no)
- Probability of occupancy is 30%

Binomial Example: Nest Box Occupancy

```
# Try this:

#dormouse presence:
set.seed(42)
(my_occ <- rbinom(n = 50, # number of "experiments" - here number of nestbo
                  size = 1, # number of checks, one check per box
                  prob = 0.3 # probability of occupation
                  ))
```

```
 [1] 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1 0 0 0 1 0 1 1 1 0 0 0 1 0
0
[39] 1 0 0 0 0 1 0 1 1 0 1 0
```

Binomial Example: Nestbox Occupancy

```r
mosaicplot(table(my_occ), col = c(2,'goldenrod'))
```

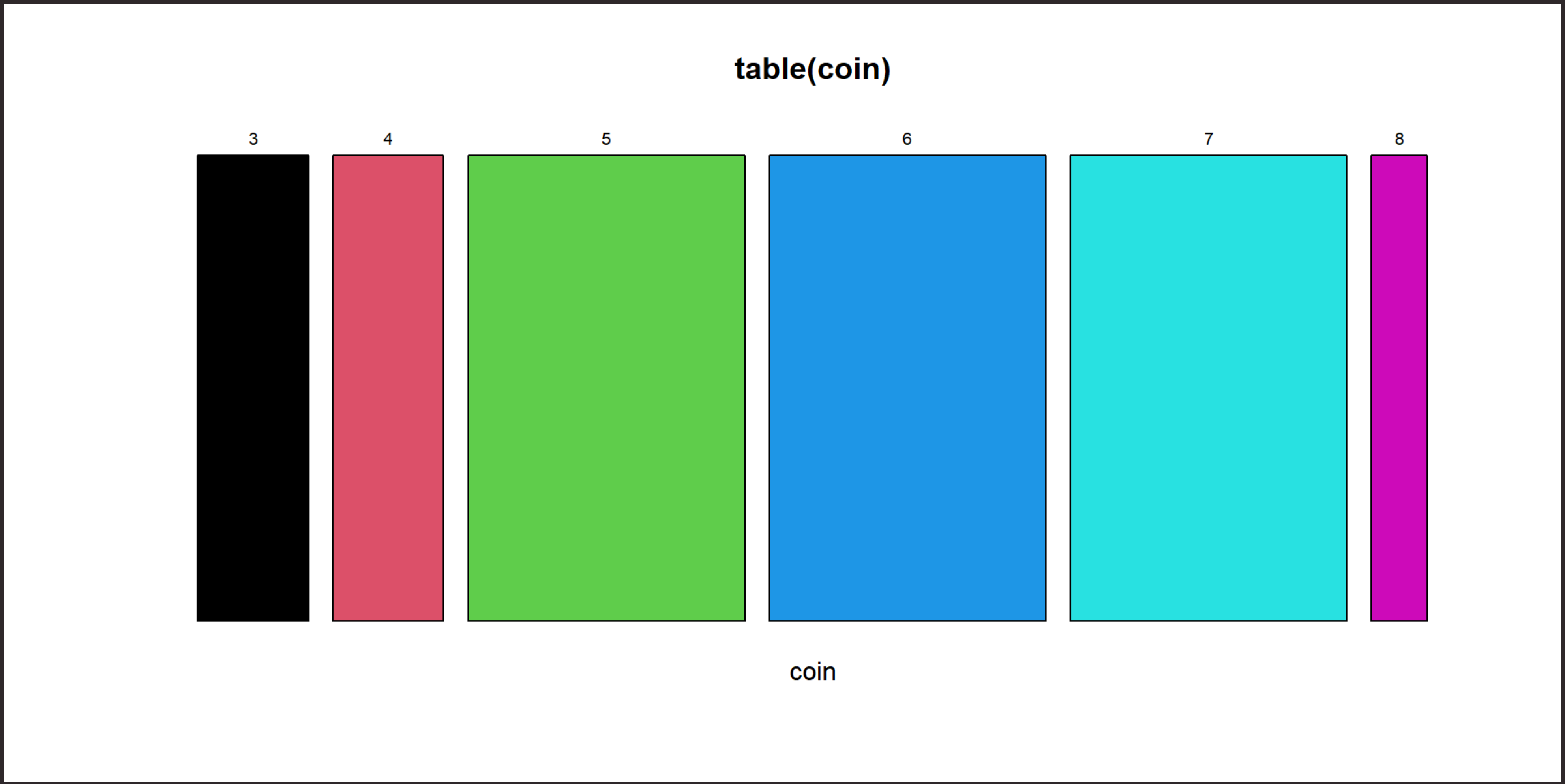
Binomial example: Flipping a coin

```
# Try this:
# Flip a fair coin:
set.seed(42)
(coin <- rbinom(n = 20, # Number of "experiments", 20 people flipping a coi
        size = 10, # Number of coin flips landing on "heads" out of 10 flips
        prob = .5)) # Probability of "heads"
```

```
[1] 7 7 4 7 6 5 6 3 6 6 5 6 7 4 5 7 8 3 5 5
```

Binomial example: Flipping a coin

```r
mosaicplot(table(coin), col = 1:unique(coin))
```



Binomial parameters

$$Binomial\ distribution : Binom(n, p)$$

$$Number\ of\ trials :\ n$$

$$Probability\ of\ success = p$$

Density plot for different binomial parameters

```r
# Try this:

# Binomial parameters
# 3 n of trial values
(n <- c(10, 10, 20))
```

```
[1] 10 10 20
```

```r
# 3 probability values
(p <- c(.5, .8, .5))
```

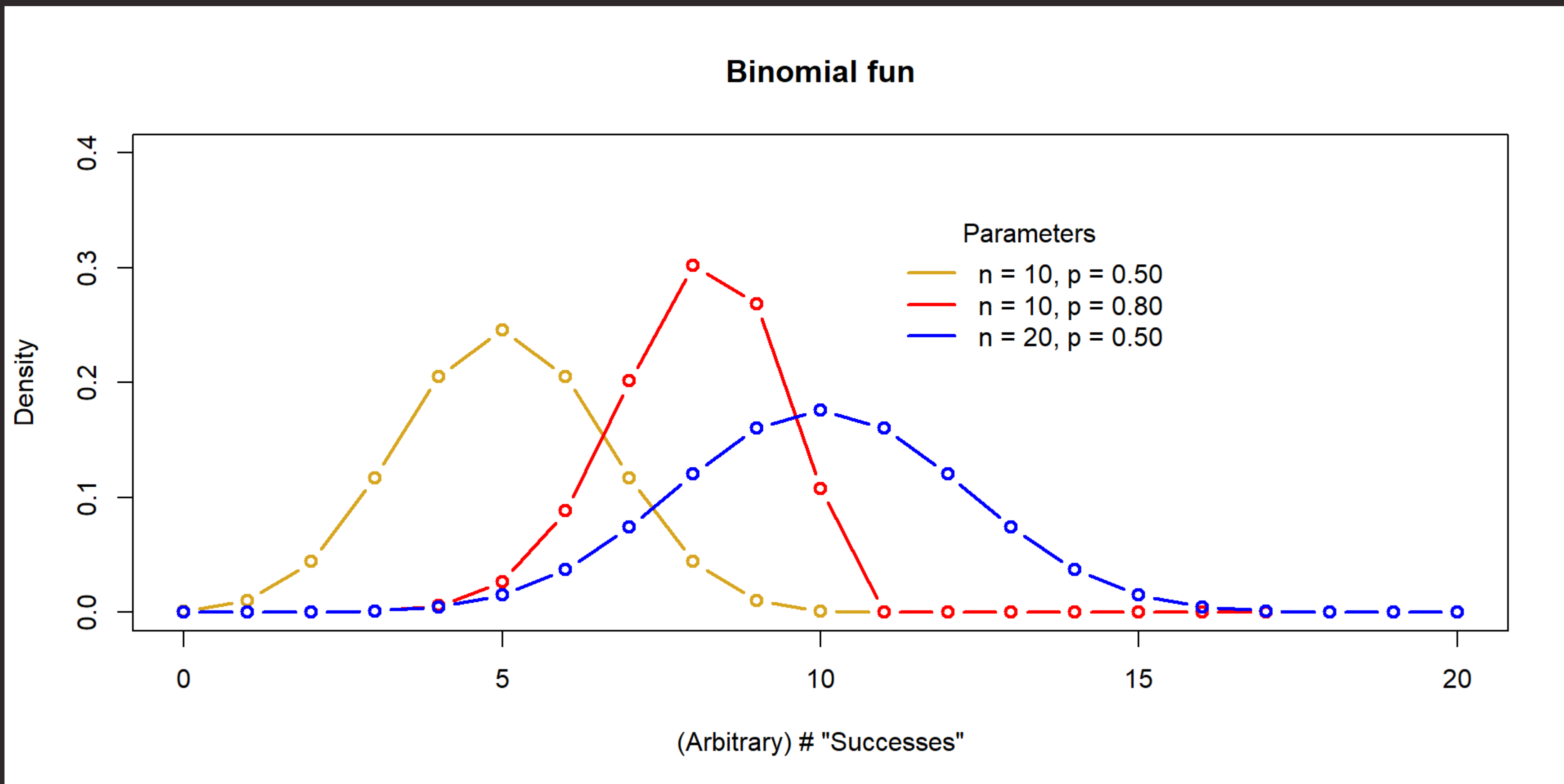
```
[1] 0.5 0.8 0.5
```

Density plot for different binomial parameters

```
# Make a baseline plot
x <- seq(0, 20, by = 1)

# Probabilities for our first set of parameters
y1 <- dbinom(x = x,
             size = n[1],
             prob = p[1])

# Baseline plot Binom
plot(x = x, y = y1, ylim = c(0, .4),
     col = "goldenrod",lwd = 2, type = "b",
     main = "Binomial fun",
     ylab = "Density", xlab = "(Arbitrary) # \"Successes\"")

# Make distribution lines
mycol <- c("red", "blue")
for(i in 1:2){
  y <- dbinom(x = x, size = n[i+1], prob = p[i+1])
  lines(x = x, y = y,
        col = mycol[i], lwd = 2, type = "b")
```

```
}

# Add a legend
legend(title = "Parameters"
```

Density plot for different binomial parameters



Diagnosing the distribution

A very common task faced when handling data is "diagnosing the distribution". Just like a human doctor diagnosing an ailment, you examine the evidence, consider the alternatives, judge the context, take an educated guess.

Diagnosing the distribution

- Statistical tests to compare data to a theoretical model
- Can be useful
- But diagnosis is principally a subjective endeavor
- Good practice is to have a set of steps to adhere to when diagnosing a distribution…

Diagnosing the distribution

- Develop an **expectation of the distribution**, based on type of data
- **Graph the data**
  - Almost always with a histogram and q-q plot with theoretical quartile line for comparison
- Compare q-q plots with different distributions for comparison if in doubt
- If the assumption of a particular distribution is important (like Gaussian residuals), try transforming your data and compare
  - i.e., `log(your_data)`, `sqrt(your_data)` etc.

Diagnosing the distribution

Visit bootcamp page for some further reading and resources

Practice exercises

Practice exercise 01

- Do you expect weight to be Gaussian distributed? How about ked? Explain your answer for each.

Practice exercise 02

- Show the code to graphically diagnose and decide whether weight is Gaussian and explain your conclusion.

Practice exercise 03

- Show the code to graphically diagnose and decide whether ked is Gaussian and explain your conclusion. If you choose another likely distribution, test it as well and similarly diagnose.

Practice exercise 04

- Explore whether trough is Gaussian, and explain whether you expect it to be so. If not, does transforming the data “persuade it” to conform to Gaussian? Discuss.