Introduction to R Programming / Rstudio and R Markdown / R Notebooks

Scott K. Hyde
Mathematics Department

Brigham Young University – Hawaii
Laie, Hawaii

Winter 2024

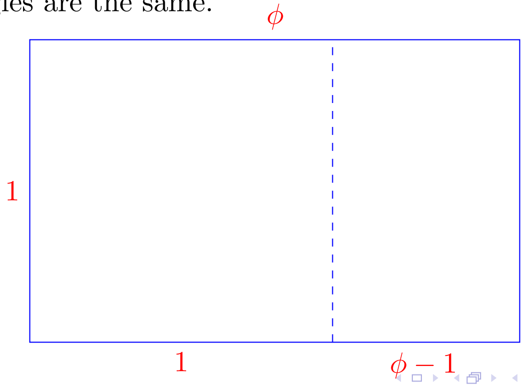
Outline

- Since ancient times, the Golden Rectangle has been believed to be the most pleasing rectangle to look at.

- Since ancient times, the Golden Rectangle has been believed to be the most pleasing rectangle to look at.
- It is composed of a rectangle that can be cut up into a square and another rectangle similar to the original one.

- Since ancient times, the Golden Rectangle has been believed to be the most pleasing rectangle to look at.
- It is composed of a rectangle that can be cut up into a square and another rectangle similar to the original one.
- In other words, the ratio of the dimensions of both rectangles are the same.

- Since ancient times, the Golden Rectangle has been believed to be the most pleasing rectangle to look at.
- It is composed of a rectangle that can be cut up into a square and another rectangle similar to the original one.
- In other words, the ratio of the dimensions of both rectangles are the same.



Golden Ratio

The Golden Rectangle is defined by

$$\frac{1}{\phi} = \frac{\phi - 1}{1},$$

Golden Ratio

The Golden Rectangle is defined by

$$\frac{1}{\phi} = \frac{\phi - 1}{1},$$

which simplifies to

$$\phi^2 - \phi - 1 = 0.$$

Golden Ratio

The Golden Rectangle is defined by

$$\frac{1}{\phi} = \frac{\phi - 1}{1},$$

which simplifies to

$$\phi^2 - \phi - 1 = 0.$$

The positive solution to the equation is

$$\phi = \frac{1 + \sqrt{5}}{2}$$

R Basics

- Numbers in R:

R Basics

- Numbers in R:

```
> phi = (1+sqrt(5))/2;phi
```

R Basics

- Numbers in R:

```
> phi = (1+sqrt(5))/2;phi
[1] 1.618034
```

R Basics

- Numbers in R:

```
> phi = (1+sqrt(5))/2;phi
[1] 1.618034
```

  - Default is 6 significant digits.

R Basics

- Numbers in R:

```
> phi = (1+sqrt(5))/2;phi
[1] 1.618034
```

  - Default is 6 significant digits.
  - Obtain 22 significant digits with `options(digits=22)`

R Basics

- Numbers in R:

```
> phi = (1+sqrt(5))/2;phi
[1] 1.618034
```

  - Default is 6 significant digits.
  - Obtain 22 significant digits with `options(digits=22)`
  - However, the last 6 digits are NOT reliable and are only useful to make calculations more accurate (but should be ignored). `options(digits=16)` will give the reliable digits, but use 22 right now for illustrations.

R Basics

- Numbers in R:

```
> phi = (1+sqrt(5))/2;phi
[1] 1.618034
```

  - Default is 6 significant digits.
  - Obtain 22 significant digits with `options(digits=22)`
  - However, the last 6 digits are NOT reliable and are only useful to make calculations more accurate (but should be ignored). `options(digits=16)` will give the reliable digits, but use 22 right now for illustrations.
  - Try `> 1/3`. It will produce the strange

```
[1] 0.3333333333333333148296
```

R Basics

- Numbers in R:

  ```
  > phi = (1+sqrt(5))/2;phi
  [1] 1.618034
  ```

  - Default is 6 significant digits.
  - Obtain 22 significant digits with `options(digits=22)`
  - However, the last 6 digits are NOT reliable and are only useful to make calculations more accurate (but should be ignored). `options(digits=16)` will give the reliable digits, but use 22 right now for illustrations.
  - Try `> 1/3`. It will produce the strange

    ```
    [1] 0.3333333333333333148296
    ```

- We can solve $x^2 - x - 1 = 0$. First, write the coefficients for increasing powers. This means $-1 + -1x + 1x^2$.

R Basics

- Numbers in R:

  ```
  > phi = (1+sqrt(5))/2;phi
  [1] 1.618034
  ```

  - Default is 6 significant digits.
  - Obtain 22 significant digits with `options(digits=22)`
  - However, the last 6 digits are NOT reliable and are only useful to make calculations more accurate (but should be ignored). `options(digits=16)` will give the reliable digits, but use 22 right now for illustrations.
  - Try `> 1/3`. It will produce the strange

    ```
    [1] 0.3333333333333333148296
    ```

- We can solve $x^2 - x - 1 = 0$. First, write the coefficients for increasing powers. This means $-1 + -1x + 1x^2$.

  ```
  > p = c(-1,-1,1)
  ```

R Basics

- Numbers in R:

```
> phi = (1+sqrt(5))/2;phi
[1] 1.618034
```

  - Default is 6 significant digits.
  - Obtain 22 significant digits with `options(digits=22)`
  - However, the last 6 digits are NOT reliable and are only useful to make calculations more accurate (but should be ignored). `options(digits=16)` will give the reliable digits, but use 22 right now for illustrations.
  - Try `> 1/3`. It will produce the strange

```
[1] 0.3333333333333333148296
```

- We can solve $x^2 - x - 1 = 0$. First, write the coefficients for increasing powers. This means $-1 + -1x + 1x^2$.

```
> p = c(-1,-1,1)
> polyroot(p)
```

R Basics

- Numbers in R:

```
> phi = (1+sqrt(5))/2;phi
[1] 1.618034
```

  - Default is 6 significant digits.
  - Obtain 22 significant digits with `options(digits=22)`
  - However, the last 6 digits are NOT reliable and are only useful to make calculations more accurate (but should be ignored). `options(digits=16)` will give the reliable digits, but use 22 right now for illustrations.
  - Try `> 1/3`. It will produce the strange

```
[1] 0.3333333333333333148296
```

- We can solve $x^2 - x - 1 = 0$. First, write the coefficients for increasing powers. This means $-1 + -1x + 1x^2$.

```
> p = c(-1,-1,1)
> polyroot(p)
[1] -0.6180339887498948-0i 1.6180339887498949+0i
```

Plotting in R

- Create a function in R:

Plotting in R

- Create a function in R:

```
> f = function(x) {1/x - (x-1)}
```

Plotting in R

- Create a function in R:

```
> f = function(x) {1/x - (x-1)}
```

- Create a plot w/default settings:

Plotting in R

- Create a function in R:

```
> f = function(x) {1/x - (x-1)}
```

- Create a plot w/default settings:

```
> plot(f)
```

Plotting in R

- Create a function in R:

```
> f = function(x) {1/x - (x-1)}
```

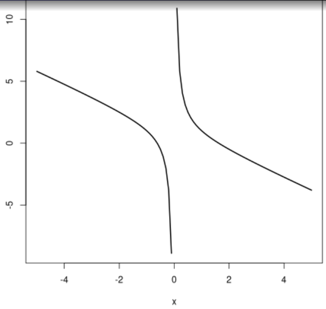
- Create a plot w/default settings:

```
> plot(f)
```

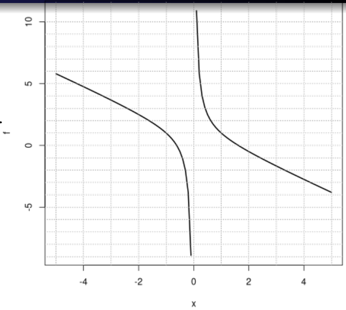
- Create a better plot with a grid:

Plotting in R

- Create a function in R:
  ```
  > f = function(x) {1/x - (x-1)}
  ```
- Create a plot w/default settings:
  ```
  > plot(f)
  ```
- Create a better plot with a grid:
  ```
  > plot(f,-5,5)
  ```



Plotting in R



- Create a function in R:

```
> f = function(x) {1/x - (x-1)}
```

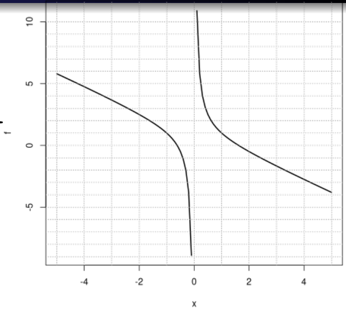
- Create a plot w/default settings:

```
> plot(f)
```

- Create a better plot with a grid:

```
> plot(f,-5,5)
> abline(h=-10:10,v=-5:5,lty=3,col="gray")
```

Plotting in R



- Create a function in R:

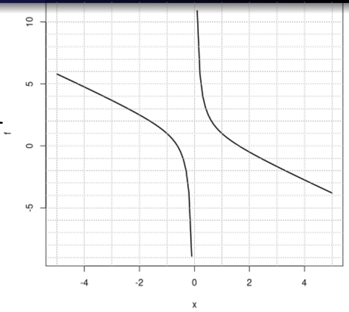
  `> f = function(x) {1/x - (x-1)}`
- Create a plot w/default settings:

  `> plot(f)`
- Create a better plot with a grid:

  `> plot(f,-5,5)`

  `> abline(h=-10:10,v=-5:5,lty=3,col="gray")`
- Type `help(plot)` for more options.

Plotting in R



- Create a function in R:

```
> f = function(x) {1/x - (x-1)}
```

- Create a plot w/default settings:

```
> plot(f)
```

- Create a better plot with a grid:

```
> plot(f,-5,5)
> abline(h=-10:10,v=-5:5,lty=3,col="gray")
```

- Type `help(plot)` for more options.
- There are other commands like `curve`

Fibonacci Numbers

Fibonacci Numbers appear often in nature, such as petals of flowers, organization of sunflower seeds, pine cones, branches on trees, shell spirals, reproduction, and even more.

Leonardo Fibonacci originally posed the following question:

> *A newly born breeding pair of rabbits are put in a field; each breeding pair mates at the age of one month, and at the end of their second month, they always produce another pair of rabbits; and rabbits never die, but continue breeding forever. Fibonacci posed the puzzle: how many pairs will there be in one year?*

Fibonacci's problem introduces a delay for maturation, which made the problem more complicated.

Fibonacci Numbers

If f_n dentoes the number of pairs of rabbits after n months, then the number of pairs is the number at the beginning plus the number of births:

$$f_n = f_{n-1} + f_{n-2}$$

with initial conditions of $f_0 = f_1 = 1$. So we see

$$\begin{aligned} f_2 &= f_1 + f_0 = 1 + 1 = 2 \\ f_3 &= f_2 + f_1 = 2 + 1 = 3 \\ f_4 &= f_3 + f_2 = 3 + 2 = 5 \\ &\dots \end{aligned}$$

Fibonacci Program

Here is a R function for the Fibonacci Sequence (Uses the for loop construct)

Fibonacci Program

Here is a R function for the Fibonacci Sequence (Uses the for loop construct)

```
fib = function(n) {
  ## Fibonacci sequence
  ## Generates the first n Fibonacci Numbers
  m = c(1,2,rep(0,n-2)) #start with 1 and 2
  for (k in 3:n) {
    m[k] = m[k-1] + m[k-2]
  }
  m
}
```

Fibonacci Program

Here is a R function for the Fibonacci Sequence (Uses the for loop construct)

```
fib = function(n) {
  ## Fibonacci sequence
  ## Generates the first n Fibonacci Numbers
  m = c(1,2,rep(0,n-2)) #start with 1 and 2
  for (k in 3:n) {
    m[k] = m[k-1] + m[k-2]
  }
  m
}
```

How many rabbits at one year? Use $n = 12$ months

Fibonacci Program

Here is a R function for the Fibonacci Sequence (Uses the for loop construct)

```
fib = function(n) {
  ## Fibonacci sequence
  ## Generates the first n Fibonacci Numbers
  m = c(1,2,rep(0,n-2)) #start with 1 and 2
  for (k in 3:n) {
    m[k] = m[k-1] + m[k-2]
  }
  m
}
```

How many rabbits at one year? Use $n = 12$ months

```
> fib(12)
[1]   1   2   3   5   8  13  21  34  55  89 144 233
```

Fibonacci Program

Suppose you want to know the sequence until it passes a specific number. We use the while loop construct to answer that question:

Fibonacci Program

Suppose you want to know the sequence until it passes a specific number. We use the while loop construct to answer that question:

```
fibmax = function(M) {
  ## Fibonacci Maximum - returns months and
  ## fibonaccinumbers until supasses M.
  m = c(1,2) #start with 1 and 2
  k=3
  while (m[k-1] < M) {
    m = c(m,m[k-1] + m[k-2])
    k = k + 1
  }
  list(sequence=m,months=length(m))
}
```

Fibonacci Program

Suppose you run `fibmax(10000)`. It would return

Fibonacci Program

Suppose you run `fibmax(10000)`. It would return

```
$sequence
 [1]     1     2     3     5     8    13    21
 [8]    34    55    89   144   233   377   610
[15]   987  1597  2584  4181  6765 10946

$months
[1] 20
```

Fibonacci Program

Suppose you run `fibmax(10000)`. It would return

```
$sequence
 [1]     1     2     3     5     8    13    21
 [8]    34    55    89   144   233   377   610
[15]   987  1597  2584  4181  6765 10946

$months
[1] 20
```

So it follows that after 20 months, the Fibonacci sequence has surpassed 10000. In the 19^{th} month, the number was 6765, and the 20^{th} Fibonacci number was 10946.

Fibonacci Program

Suppose you run `fibmax(10000)`. It would return

```
$sequence
 [1]     1     2     3     5     8    13    21
 [8]    34    55    89   144   233   377   610
[15]   987  1597  2584  4181  6765 10946

$months
[1] 20
```

So it follows that after 20 months, the Fibonacci sequence has surpassed 10000. In the 19^{th} month, the number was 6765, and the 20^{th} Fibonacci number was 10946.

While loops are valuable for stopping a loop when a criterion has been met.

Fibonacci and the Golden Ratio

The ratio of successive Fibonacci numbers converge to the Golden Ratio! Recall that the Golden Ratio is

$$\phi = \frac{1+\sqrt{5}}{2} = 1.61803398874989.$$

Fibonacci and the Golden Ratio

The ratio of successive Fibonacci numbers converge to the Golden Ratio! Recall that the Golden Ratio is

$$\phi = \frac{1+\sqrt{5}}{2} = 1.61803398874989.$$

We can compute these ratios with two calls to `fib`:

```
> n=40; ## The -1 below means "remove the 1st element"
> fib(n)[-1]/fib(n)[-n]
```

Fibonacci and the Golden Ratio

The ratio of successive Fibonacci numbers converge to the Golden Ratio! Recall that the Golden Ratio is

$$\phi = \frac{1+\sqrt{5}}{2} = 1.61803398874989.$$

We can compute these ratios with two calls to `fib`:

```
> n=40; ## The -1 below means "remove the 1st element"
> fib(n)[-1]/fib(n)[-n]
```

You can also get the same result with

Fibonacci and the Golden Ratio

The ratio of successive Fibonacci numbers converge to the Golden Ratio! Recall that the Golden Ratio is

$$\phi = \frac{1+\sqrt{5}}{2} = 1.61803398874989.$$

We can compute these ratios with two calls to `fib`:

```
> n=40; ## The -1 below means "remove the 1st element"
> fib(n)[-1]/fib(n)[-n]
```

You can also get the same result with

```
> exp(diff(log(fib(n))))   ## Figure it out!
```

Fibonacci and the Golden Ratio

The ratio of successive Fibonacci numbers converge to the Golden Ratio! Recall that the Golden Ratio is

$$\phi = \frac{1+\sqrt{5}}{2} = 1.61803398874989.$$

We can compute these ratios with two calls to `fib`:

```
> n=40; ## The -1 below means "remove the 1st element"
> fib(n)[-1]/fib(n)[-n]
```

You can also get the same result with

```
> exp(diff(log(fib(n))))  ## Figure it out!
```

Here's a few of the ratios:

$$f_6/f_5 = 1.625$$
$$f_{26}/f_{25} = 1.61803398878024$$
$$f_{40}/f_{39} = 1.61803398874989$$

3n+1 sequence

The Collatz Problem is an unsolved problem in Number Theory. We can explore this problem using R. The problem is stated as:

3n+1 sequence

The Collatz Problem is an unsolved problem in Number Theory. We can explore this problem using R. The problem is stated as:

1. Pick a number, `n`

3n+1 sequence

The Collatz Problem is an unsolved problem in Number Theory. We can explore this problem using R. The problem is stated as:

1. Pick a number, `n`
2. If `n=1`, stop.

3n+1 sequence

The Collatz Problem is an unsolved problem in Number Theory. We can explore this problem using R. The problem is stated as:

1. Pick a number, `n`
2. If `n=1`, stop.
3. If `n` is even, then let `n=n/2`, goto step 2

3n+1 sequence

The Collatz Problem is an unsolved problem in Number Theory. We can explore this problem using R. The problem is stated as:

1. Pick a number, `n`
2. If `n=1`, stop.
3. If `n` is even, then let `n=n/2`, goto step 2
4. If `n` is odd, then let `n=3n+1`, goto step 2

3n+1 sequence

The Collatz Problem is an unsolved problem in Number Theory. We can explore this problem using R. The problem is stated as:

1. Pick a number, `n`
2. If `n=1`, stop.
3. If `n` is even, then let `n=n/2`, goto step 2
4. If `n` is odd, then let `n=3n+1`, goto step 2

The conjecture is that this sequence will ALWAYS terminate after a finite number of steps.

3n+1 sequence

The Collatz Problem is an unsolved problem in Number Theory. We can explore this problem using R. The problem is stated as:

1. Pick a number, `n`
2. If `n=1`, stop.
3. If `n` is even, then let `n=n/2`, goto step 2
4. If `n` is odd, then let `n=3n+1`, goto step 2

The conjecture is that this sequence will ALWAYS terminate after a finite number of steps.

There are limitations to computer arithmetic so, we will have roundoff error if an element is too big.

3n+1 sequence

This problem is iterative with conditional statements, so we will use while and if-then-else statements.

3n+1 sequence

This problem is iterative with conditional statements, so we will use while and if-then-else statements.

```
collatz = function(n) { ## Show the 3n+1 sequence.
  s = n #start with n
  while (n > 1) {
    if ( n %% 2 == 0) {
      n=n/2
    } else {
      n = 3*n+1
    }
    s=c(s,n) #Add n to the s vector
  }
  s
}
```

3n+1 sequence

This program will produce a sequence that ends in 1. Try it with several values. Can you find one that is very long?

3n+1 sequence

This program will produce a sequence that ends in 1. Try it with several values. Can you find one that is very long?

```
> collatz(10)
```

3n+1 sequence

This program will produce a sequence that ends in 1. Try it with several values. Can you find one that is very long?

```
> collatz(10)
[1] 10 5 16 8 4 2 1
```

3n+1 sequence

This program will produce a sequence that ends in 1. Try it with several values. Can you find one that is very long?

```
> collatz(10)
[1] 10 5 16 8 4 2 1
> collatz(7)
```

3n+1 sequence

This program will produce a sequence that ends in 1. Try it with several values. Can you find one that is very long?

```
> collatz(10)
[1] 10 5 16 8 4 2 1
> collatz(7)
[1] 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

3n+1 sequence

This program will produce a sequence that ends in 1. Try it with several values. Can you find one that is very long?

```
> collatz(10)
[1] 10 5 16 8 4 2 1
> collatz(7)
[1] 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

- Why program this? Why not? It helps us to understand this problem. Maybe we can find patterns in them! Besides, it's really easy to just type `collatz` See if you can find patterns in them.

3n+1 sequence

This program will produce a sequence that ends in 1. Try it with several values. Can you find one that is very long?

```
> collatz(10)
[1] 10 5 16 8 4 2 1
> collatz(7)
[1] 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

- Why program this? Why not? It helps us to understand this problem. Maybe we can find patterns in them! Besides, it's really easy to just type `collatz` See if you can find patterns in them.
- Try and find the longest sequence for numbers from 1-100!

3n+1 sequence

This program will produce a sequence that ends in 1. Try it with several values. Can you find one that is very long?

```
> collatz(10)
[1] 10 5 16 8 4 2 1
> collatz(7)
[1] 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

- Why program this? Why not? It helps us to understand this problem. Maybe we can find patterns in them! Besides, it's really easy to just type `collatz` See if you can find patterns in them.
- Try and find the longest sequence for numbers from 1-100!
- I wrote a program and found the longest from 1–10,000. It is 6171 with a length of 262! Try 27!

Bessel Functions

Some functions are defined in terms of an infinite sum. For example, the Bessel Function is a solution to Bessel's Equation:

$$x^2\frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - m^2)y = 0$$

Bessel Functions

Some functions are defined in terms of an infinite sum. For example, the Bessel Function is a solution to Bessel's Equation:

$$x^2\frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - m^2)y = 0$$

- This equation appears naturally in applied problems (e.g. vibrating drums)

Bessel Functions

Some functions are defined in terms of an infinite sum. For example, the Bessel Function is a solution to Bessel's Equation:

$$x^2\frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - m^2)y = 0$$

- This equation appears naturally in applied problems (e.g. vibrating drums)
- To find the solution to this, take Differential Equations.

Bessel Functions

Some functions are defined in terms of an infinite sum. For example, the Bessel Function is a solution to Bessel's Equation:

$$x^2\frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - m^2)y = 0$$

- This equation appears naturally in applied problems (e.g. vibrating drums)
- To find the solution to this, take Differential Equations.
- A solution to the equation is the m^{th} order Bessel function of the 1^{st} kind (m is an integer). Its series is

$$J_m(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{n!(m+n)!}\left(\frac{x}{2}\right)^{m+2n}$$

Bessel Functions

Techniques to use

- We want a recursive formula, where new terms build off old terms.

Bessel Functions

Techniques to use

- We want a recursive formula, where new terms build off old terms.
- Each iteration add the new term to the sum.

Bessel Functions

Techniques to use

- We want a recursive formula, where new terms build off old terms.
- Each iteration add the new term to the sum.
- Use a while command to iterate and build the sum.

Bessel Functions

Techniques to use

- We want a recursive formula, where new terms build off old terms.
- Each iteration add the new term to the sum.
- Use a while command to iterate and build the sum.
- If the new term is larger than the tolerance, then repeat.

Bessel Functions

Techniques to use

- We want a recursive formula, where new terms build off old terms.
- Each iteration add the new term to the sum.
- Use a while command to iterate and build the sum.
- If the new term is larger than the tolerance, then repeat.
- If it is smaller, then quit.

Bessel Functions

Techniques to use

- We want a recursive formula, where new terms build off old terms.
- Each iteration add the new term to the sum.
- Use a while command to iterate and build the sum.
- If the new term is larger than the tolerance, then repeat.
- If it is smaller, then quit.
- The input to the function is a vector, so it will evaluate the function at each value in the vector.

Bessel Functions

The n^{th} and $(n+1)^{\text{th}}$ terms are

Bessel Functions

The n^{th} and $(n+1)^{\text{th}}$ terms are

$$t_n = \frac{(-1)^n}{n!(m+n)!}\left(\frac{x}{2}\right)^{m+2n} \quad t_{n+1} = \frac{(-1)^{n+1}}{(n+1)!(m+n+1)!}\left(\frac{x}{2}\right)^{m+2(n+1)}$$

Bessel Functions

The n^{th} and $(n+1)^{\text{th}}$ terms are

$$t_n = \frac{(-1)^n}{n!(m+n)!}\left(\frac{x}{2}\right)^{m+2n} \quad t_{n+1} = \frac{(-1)^{n+1}}{(n+1)!(m+n+1)!}\left(\frac{x}{2}\right)^{m+2(n+1)}$$

Dividing t_{n+1} by t_n will help develop the recursion. So

Bessel Functions

The n^{th} and $(n+1)^{\text{th}}$ terms are

$$t_n = \frac{(-1)^n}{n!(m+n)!}\left(\frac{x}{2}\right)^{m+2n} \qquad t_{n+1} = \frac{(-1)^{n+1}}{(n+1)!(m+n+1)!}\left(\frac{x}{2}\right)^{m+2(n+1)}$$

Dividing t_{n+1} by t_n will help develop the recursion. So

$$\frac{t_{n+1}}{t_n} = \frac{\frac{(-1)^{n+1}}{(n+1)!(m+n+1)!}\left(\frac{x}{2}\right)^{m+2n+2}}{\frac{(-1)^n}{n!(m+n)!}\left(\frac{x}{2}\right)^{m+2n}} = \frac{-1}{(n+1)(m+n+1)}\left(\frac{x}{2}\right)^2$$

Bessel Functions

The n^{th} and $(n+1)^{\text{th}}$ terms are

$$t_n = \frac{(-1)^n}{n!(m+n)!}\left(\frac{x}{2}\right)^{m+2n} \qquad t_{n+1} = \frac{(-1)^{n+1}}{(n+1)!(m+n+1)!}\left(\frac{x}{2}\right)^{m+2(n+1)}$$

Dividing t_{n+1} by t_n will help develop the recursion. So

$$\frac{t_{n+1}}{t_n} = \frac{\frac{(-1)^{n+1}}{(n+1)!(m+n+1)!}\left(\frac{x}{2}\right)^{m+2n+2}}{\frac{(-1)^n}{n!(m+n)!}\left(\frac{x}{2}\right)^{m+2n}} = \frac{-1}{(n+1)(m+n+1)}\left(\frac{x}{2}\right)^2$$

So

$$t_{n+1} = \frac{-x^2}{4(n+1)(m+n+1)}t_n$$

Bessel Functions

So the R code for $J_m(x)$ is

Bessel Functions

So the R code for $J_m(x)$ is

```
bessel = function(x,m,tol=1e-10) {
  ## Approx to J_m(x) to a given tolerance
  k=1
  summa = term = (x/2)^m/factorial(m)
  while (max(abs(term)) >= tol) {
    term = term*(-x^2)/(4*k*(k+m))
    summa = summa + term;
    k=k+1
  }
  summa
}
```

Bessel Functions

To graph the function, issue the commands

Bessel Functions

To graph the function, issue the commands

```
> curve(bessel(x,m=2),0,20,col="blue")
```

Bessel Functions

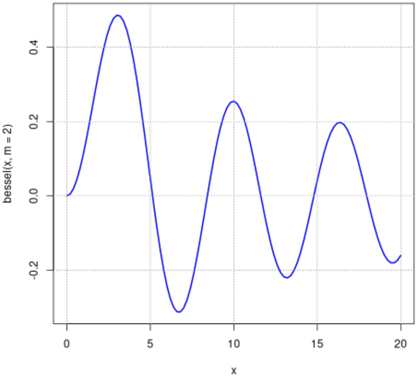
To graph the function, issue the commands

```
> curve(bessel(x,m=2),0,20,col="blue")
> abline(h=0.2*(-2:3),v=5*(1:4),lty=3,col="gray")
```

Bessel Functions

To graph the function, issue the commands

```
> curve(bessel(x,m=2),0,20,col="blue")
> abline(h=0.2*(-2:3),v=5*(1:4),lty=3,col="gray")
```



Several Bessel Functions

We can also plot several on top of each other! This code produces the plot on the next slide

Several Bessel Functions

We can also plot several on top of each other! This code produces the plot on the next slide

```
> colors=c("blue","green","red","purple","orange")
```

Several Bessel Functions

We can also plot several on top of each other! This code produces the plot on the next slide

```
> colors=c("blue","green","red","purple","orange")
> x=seq(0,20,length=1000)
```

Several Bessel Functions

We can also plot several on top of each other! This code produces the plot on the next slide

```
> colors=c("blue","green","red","purple","orange")
> x=seq(0,20,length=1000)
> plot(x,bessel(x,m=0),type="l",col=colors[1],lwd=2,
   ylab=expression(J[m](x)),main=expression(
  paste("Bessel Functions ",J[0](x)," to ",J[4](x))))
```

Several Bessel Functions

We can also plot several on top of each other! This code produces the plot on the next slide

```
> colors=c("blue","green","red","purple","orange")
> x=seq(0,20,length=1000)
> plot(x,bessel(x,m=0),type="l",col=colors[1],lwd=2,
   ylab=expression(J[m](x)),main=expression(
  paste("Bessel Functions ",J[0](x)," to ",J[4](x))))
> for (i in 1:4)
  points(x,bessel(x,m=i),type="l",col=colors[i+1],lwd=2)
```

Several Bessel Functions

We can also plot several on top of each other! This code produces the plot on the next slide

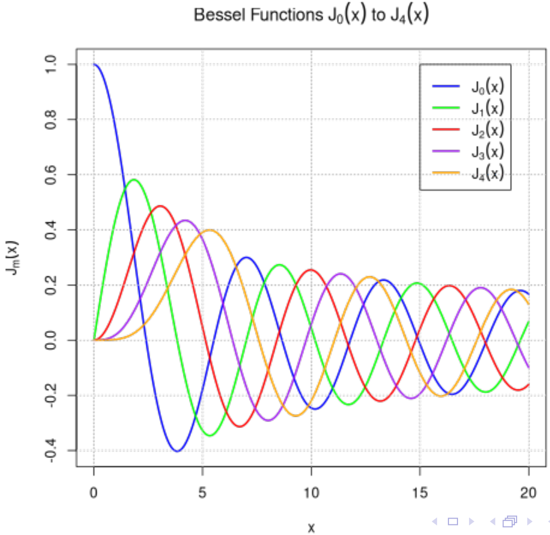
```
> colors=c("blue","green","red","purple","orange")
> x=seq(0,20,length=1000)
> plot(x,bessel(x,m=0),type="l",col=colors[1],lwd=2,
    ylab=expression(J[m](x)),main=expression(
   paste("Bessel Functions ",J[0](x)," to ",J[4](x))))
> for (i in 1:4)
   points(x,bessel(x,m=i),type="l",col=colors[i+1],lwd=2)
> abline(h=0.2*(-2:5),v=5*(0:4),lty=3,col="gray")
```

Several Bessel Functions

We can also plot several on top of each other! This code produces the plot on the next slide

```
> colors=c("blue","green","red","purple","orange")
> x=seq(0,20,length=1000)
> plot(x,bessel(x,m=0),type="l",col=colors[1],lwd=2,
    ylab=expression(J[m](x)),main=expression(
   paste("Bessel Functions ",J[0](x)," to ",J[4](x))))
> for (i in 1:4)
   points(x,bessel(x,m=i),type="l",col=colors[i+1],lwd=2)
> abline(h=0.2*(-2:5),v=5*(0:4),lty=3,col="gray")
> legend(15,1,legend=c(expression(J[0](x),J[1](x),J[2](x),
    J[3](x),J[4](x))),col=colors,lty=1,lwd=2)
```

Several Bessel Functions



R scripts

- A “script” file in R that consists of functions, commands, statements, etc.

R scripts

- A "script" file in R that consists of functions, commands, statements, etc.
- It is a list of commands that allow you to reproduce your results at any time.

R scripts

- A "script" file in R that consists of functions, commands, statements, etc.
- It is a list of commands that allow you to reproduce your results at any time.
- An improved method of intermingling code and output is through R Markdown.

R scripts

- A “script” file in R that consists of functions, commands, statements, etc.
- It is a list of commands that allow you to reproduce your results at any time.
- An improved method of intermingling code and output is through R Markdown.
- In particular R-Notebook can be used interactively.

R scripts

- A "script" file in R that consists of functions, commands, statements, etc.
- It is a list of commands that allow you to reproduce your results at any time.
- An improved method of intermingling code and output is through R Markdown.
- In particular R-Notebook can be used interactively.
- What is an R-Notebook?

R-Notebooks / R-Markdown

- It is a mode Rstudio in which R code is "tangled" with your explanation.

R-Notebooks / R-Markdown

- It is a mode Rstudio in which R code is “tangled” with your explanation.
- Using a simple markup language (called R-Markdown), you write a document just like you would a Word document (but with simple mark-up symbols).

R-Notebooks / R-Markdown

- It is a mode Rstudio in which R code is "tangled" with your explanation.
- Using a simple markup language (called R-Markdown), you write a document just like you would a Word document (but with simple mark-up symbols).
- What's new (compared to R-scripts) is that you can insert R code snippets in the middle of the document.

R-Notebooks / R-Markdown

- It is a mode Rstudio in which R code is “tangled” with your explanation.
- Using a simple markup language (called R-Markdown), you write a document just like you would a Word document (but with simple mark-up symbols).
- What’s new (compared to R-scripts) is that you can insert R code snippets in the middle of the document.
- These snippets are executed and the results are inserted into the document where they appear.

R-Notebooks / R-Markdown

- It is a mode Rstudio in which R code is “tangled” with your explanation.
- Using a simple markup language (called R-Markdown), you write a document just like you would a Word document (but with simple mark-up symbols).
- What’s new (compared to R-scripts) is that you can insert R code snippets in the middle of the document.
- These snippets are executed and the results are inserted into the document where they appear.
- This allows you to “knit” your document and code together.

R-Notebooks / R-Markdown

- It is a mode Rstudio in which R code is “tangled” with your explanation.
- Using a simple markup language (called R-Markdown), you write a document just like you would a Word document (but with simple mark-up symbols).
- What’s new (compared to R-scripts) is that you can insert R code snippets in the middle of the document.
- These snippets are executed and the results are inserted into the document where they appear.
- This allows you to “knit” your document and code together.
- The markup language is called “R-Markdown”.

R-Notebooks / R-Markdown

- It is a mode Rstudio in which R code is “tangled” with your explanation.
- Using a simple markup language (called R-Markdown), you write a document just like you would a Word document (but with simple mark-up symbols).
- What’s new (compared to R-scripts) is that you can insert R code snippets in the middle of the document.
- These snippets are executed and the results are inserted into the document where they appear.
- This allows you to “knit” your document and code together.
- The markup language is called “R-Markdown”.
- In our class, you will be required to create your problem set solutions using “R-Notebooks” (or R-Markdown).

R-Notebooks / R-Markdown

- Two modes: R-Notebook and R-Markdown (KnitR).

R-Notebooks / R-Markdown

- Two modes: R-Notebook and R-Markdown (KnitR).
- Both are the same but R-Notebook is meant to be run interactively whereas R-Markdown's purpose is to create a document and show how it was obtained.

R-Notebooks / R-Markdown

- Two modes: R-Notebook and R-Markdown (KnitR).
- Both are the same but R-Notebook is meant to be run interactively whereas R-Markdown's purpose is to create a document and show how it was obtained.
- Think of it like homework, but rather the answers appearing right after the code. This makes "reproducible homework". Anyone looking at this document will see your answers, but they also will know how you obtained the results!

R-Notebooks / R-Markdown

- Two modes: R-Notebook and R-Markdown (KnitR).
- Both are the same but R-Notebook is meant to be run interactively whereas R-Markdown's purpose is to create a document and show how it was obtained.
- Think of it like homework, but rather the answers appearing right after the code. This makes "reproducible homework". Anyone looking at this document will see your answers, but they also will know how you obtained the results!
- Here is a link to a tutorial on R Markdown online. Spend a few minutes on it and learn it a little better. Please ask questions when you can!

R-Notebooks / R-Markdown

- Two modes: R-Notebook and R-Markdown (KnitR).
- Both are the same but R-Notebook is meant to be run interactively whereas R-Markdown's purpose is to create a document and show how it was obtained.
- Think of it like homework, but rather the answers appearing right after the code. This makes "reproducible homework". Anyone looking at this document will see your answers, but they also will know how you obtained the results!
- Here is a link to a tutorial on R Markdown online. Spend a few minutes on it and learn it a little better. Please ask questions when you can!
- You can also go to the link which has the output from this presentation in R-Notebook mode. Click on it and learn!