Module6

Joe Vargovich

9/18/2020

Exercise 1 - Work with the political candidate dataset.

```
#Import the data from GH.
prez <- readr::read_csv('https://raw.githubusercontent.com/dereksonderegger/444/master/data-raw/Prez_Ca
```

```
## Parsed with column specification:
## cols(
##   Candidate = col_character(),
##   Gender = col_character(),
##   Birthday = col_date(format = ""),
##   Party = col_character(),
##   AgeOnElection = col_double()
## )
```

```
#a. Re-code the Gender column to have Male and Female levels.
  #Similarly convert the party variable to be Democratic or Republican.
prez = prez %>%
  mutate(Party = ifelse(Party == "D", "Democrat", "Republican" ))
head(prez)
```

```
## # A tibble: 6 x 5
##   Candidate      Gender Birthday   Party    AgeOnElection
##   <chr>          <chr>  <date>     <chr>            <dbl>
## 1 Pete Buttigieg M      1982-01-19 Democrat            38
## 2 Andrew Yang    M      1975-01-13 Democrat            45
## 3 Juilan Castro  M      1976-09-16 Democrat            44
## 4 Beto O'Rourke  M      1972-09-26 Democrat            48
## 5 Cory Booker    M      1969-04-27 Democrat            51
## 6 Kamala Harris  F      1964-10-20 Democrat            56
```

```
#b. Change Bernie Sanders to "Independent"
# Select the row where the candidate is Bernie Sanders
# and set just his party to Independent.
prez$Party[prez$Candidate == "Bernie Sanders"] = "Independent"
tail(prez)
```

```
## # A tibble: 6 x 5
##   Candidate        Gender Birthday   Party       AgeOnElection
##   <chr>            <chr>  <date>     <chr>               <dbl>
## 1 Kamala Harris    F      1964-10-20 Democrat               56
## 2 Amy Klobucher    F      1960-05-25 Democrat               60
## 3 Elizabeth Warren F      1949-06-22 Democrat               71
## 4 Donald Trump     M      1946-06-14 Republican             74
```

```
## 5 Joe Biden        M      1942-11-20 Democrat              77
## 6 Bernie Sanders   M      1941-09-08 Independent           79
```

Exercise 2 - Density function of X and conditionals for valid values of x.

Sample code for built in density function

```
a <- 4      # The min and max values we will use for this example
b <- 10     # Could be anything, but we need to pick something

x <- runif(n=1, 0,10)  # one random value between 0 and 10

# what is value of f(x) at the randomly selected x value?
dunif(x, a, b)
```

```
## [1] 0
```

a. We need a <= x <= b to return 1/b-a, otherwise return 0.

```
a <- 4
b <- 10
x <- runif(n=1, 0,10)  # one random value between 0 and 10

if( x < a ){
  result <- 0
}else if( x <= b ){
  result <- 1/(b-a) #Parenthesis are important here! Order of ops.
}else{
  result <- 0
}

print(paste('x=',round(x,digits=3), '  result=', round(result,digits=3)))
```

```
## [1] "x= 3.552   result= 0"
```

b. Create conditionals with the "and" and the "or" operators to calculate the density function properly.

```
#i.
x <- runif(n=1, 0,10)  # one random value between 0 and 10
if( (a<=x) & (x<=b) ){ # AND (&) is necessary here as both bounds apply
  result <- 1/(b-a)
}else{
  result <- 0
}
print(paste('x=',round(x,digits=3), '  result=', round(result,digits=3)))
```

```
## [1] "x= 6.574   result= 0.167"
```

```
#ii.
x <- runif(n=1, 0,10)  # one random value between 0 and 10
if( (x<a) | (b<x) ){ # OR (|) is needed here as either condition would break
  result <- 0          # the bounds of the density function
```

```
}else{
  result <- 1/(b-a)
}
print(paste('x=',round(x,digits=3), '  result=', round(result,digits=3)))
```

```
## [1] "x= 3.918   result= 0"
```

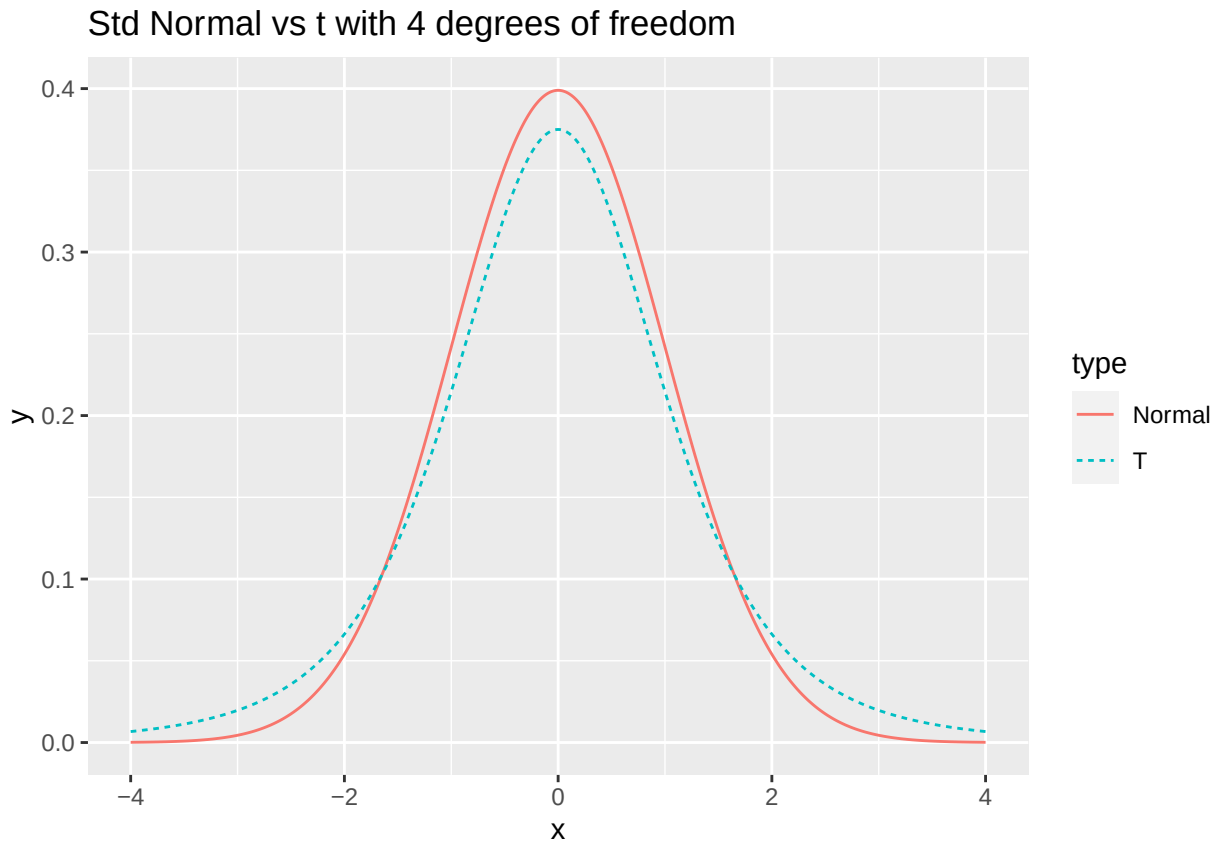
```
#iii.
x <- runif(n=1, 0,10)  # one random value between 0 and 10
result <- ifelse( a<x & x<b, 1/(b-a), 0 )
print(paste('x=',round(x,digits=3), '  result=', round(result,digits=3)))
```

```
## [1] "x= 0.876   result= 0"
```

Exercise 3 - For loops to create multiple graphs concisely

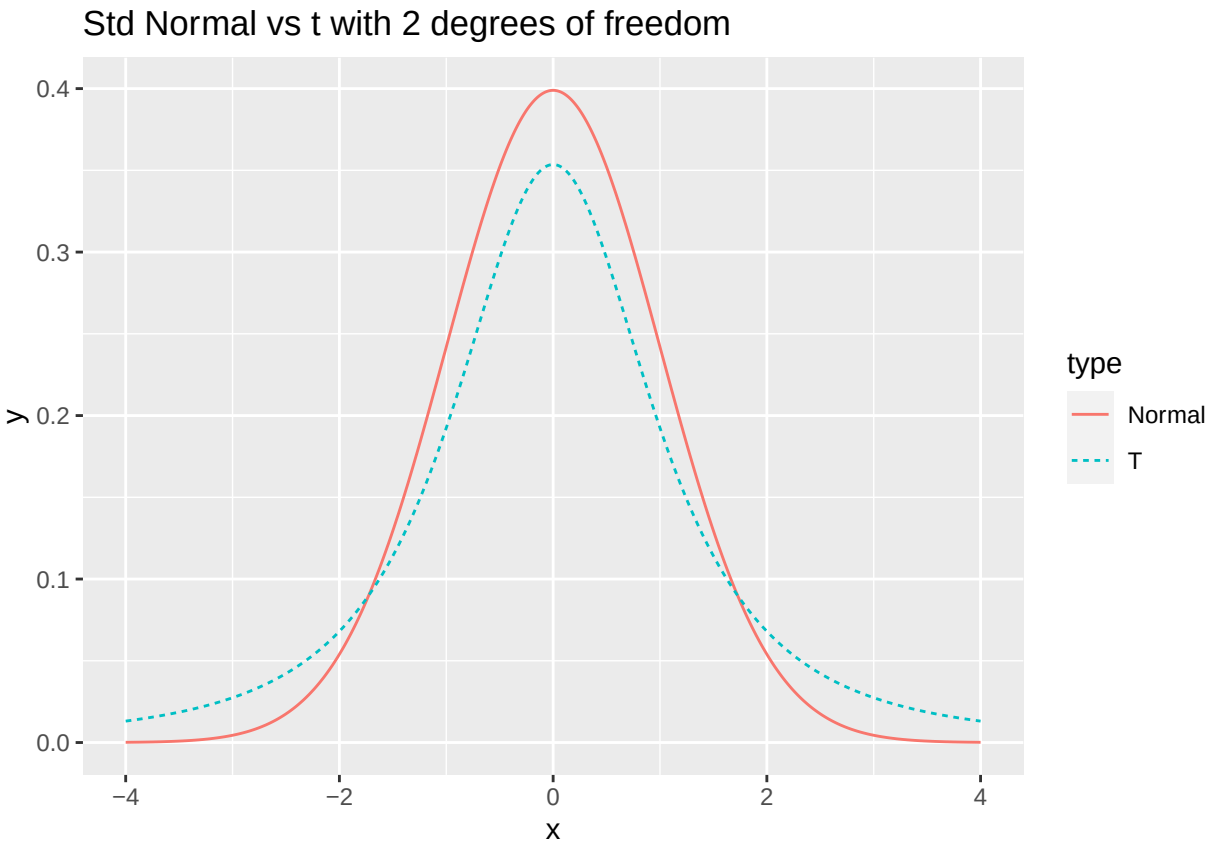
Example code for the desired graph

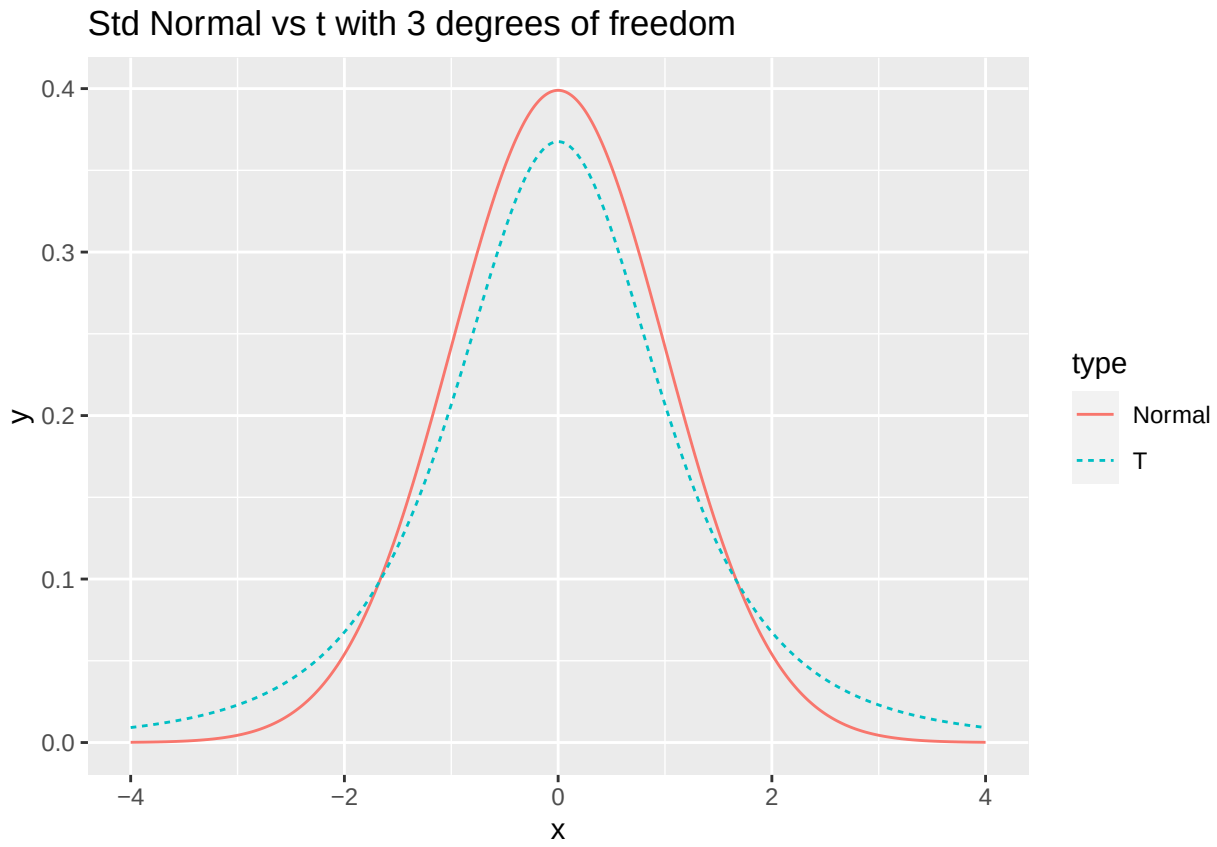
```
df <- 4
N <- 1000
x <- seq(-4, 4, length=N)

data <- data.frame(
  x = c(x,x),
  y = c(dnorm(x), dt(x, df)),
  type = c( rep('Normal',N), rep('T',N) ) )

write.csv(data, "degreeFreedomGraph.csv")

# make a nice graph
myplot <- ggplot(data, aes(x=x, y=y, color=type, linetype=type)) +
  geom_line() +
  labs(title = paste('Std Normal vs t with', df, 'degrees of freedom'))

# actually print the nice graph we made
print(myplot)
```

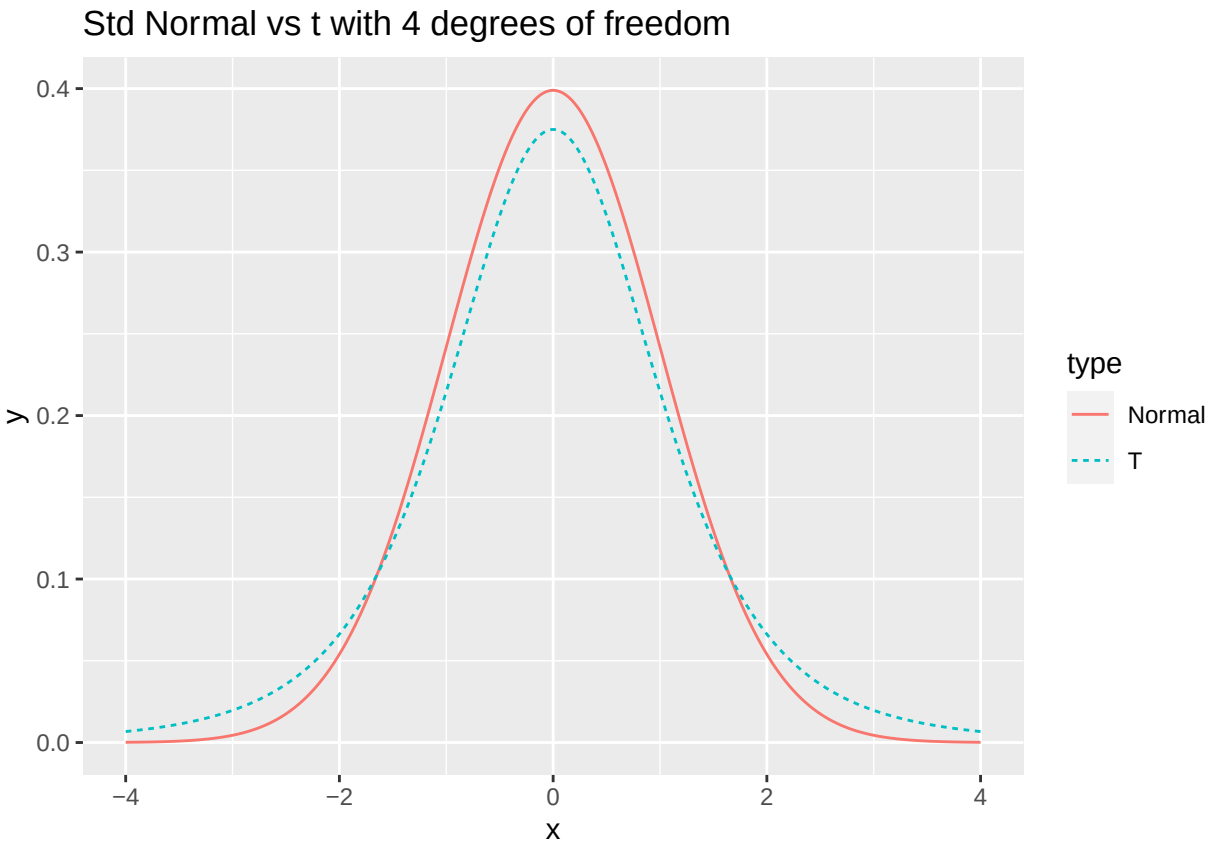


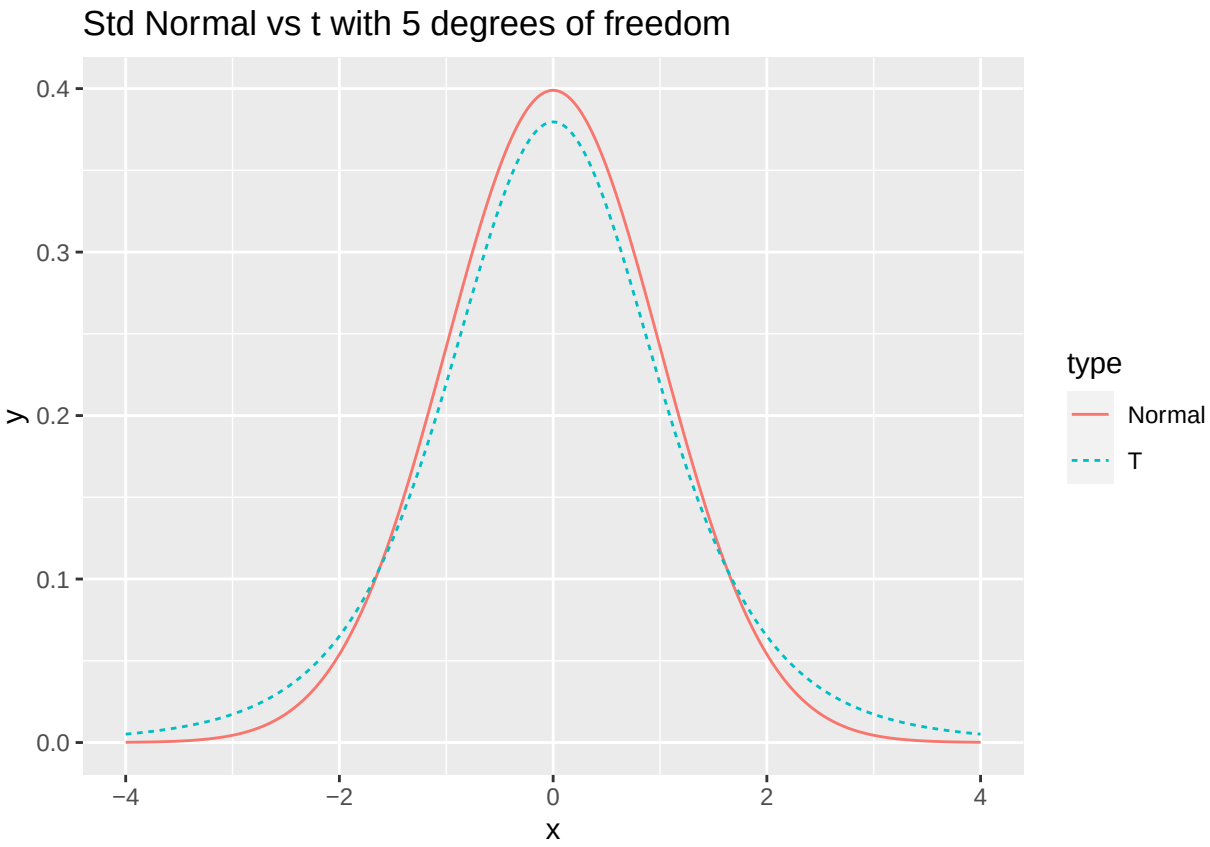
a. Use a for loop to create similar graphs for degrees of freedom 2,3,4 . . . , 29,30

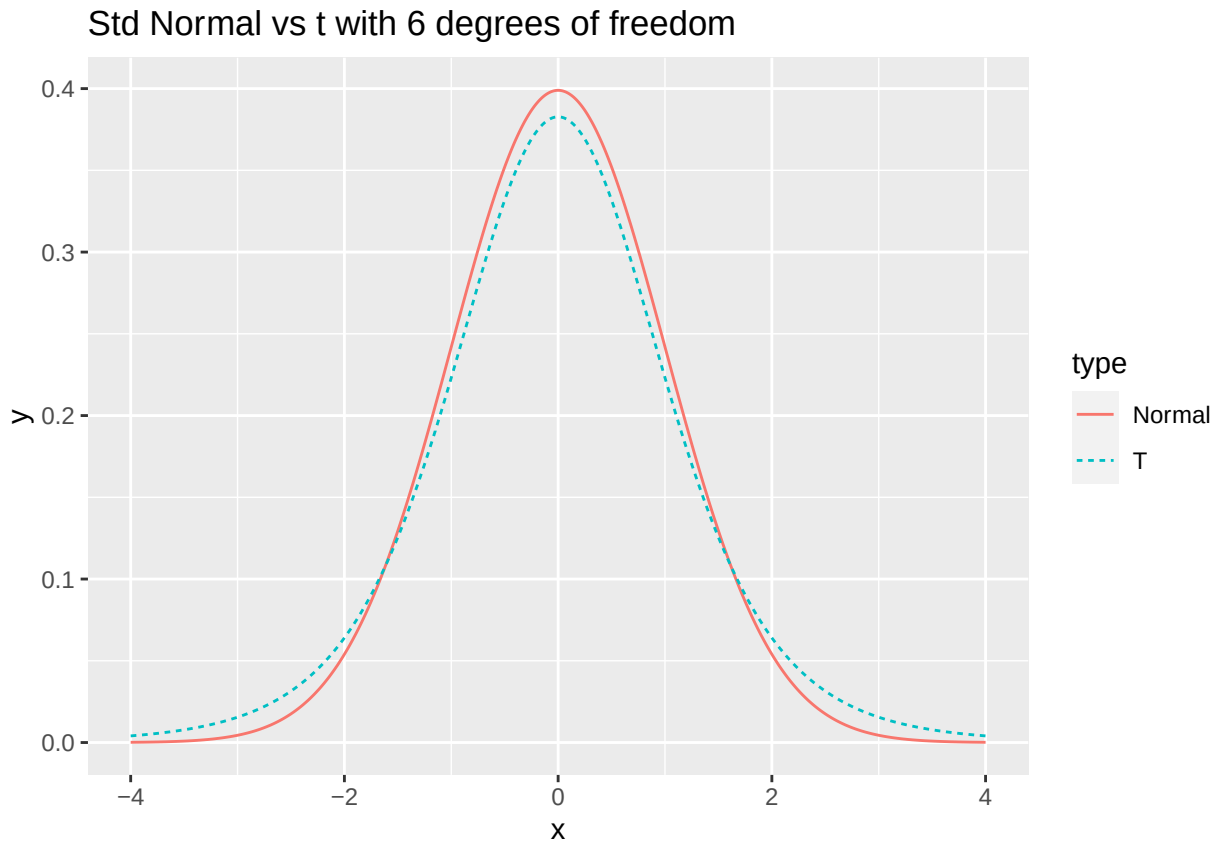
```
df = 4
# make a nice graph
  data <- data.frame(
  x = c(x,x),
  y = c(dnorm(x), dt(x, df)),
  type = c( rep('Normal',N), rep('T',N) ) )


for(df in 2:30){
  #Update our y column for each df value
  data$y <- c(dnorm(x), dt(x, df))

  #Create a new plot for each new y column
  myplot <- ggplot(data, aes(x=x, y=y, color=type, linetype=type)) +
    geom_line() +
    labs(title = paste('Std Normal vs t with', df, 'degrees of freedom'))

  print(myplot)
}
```

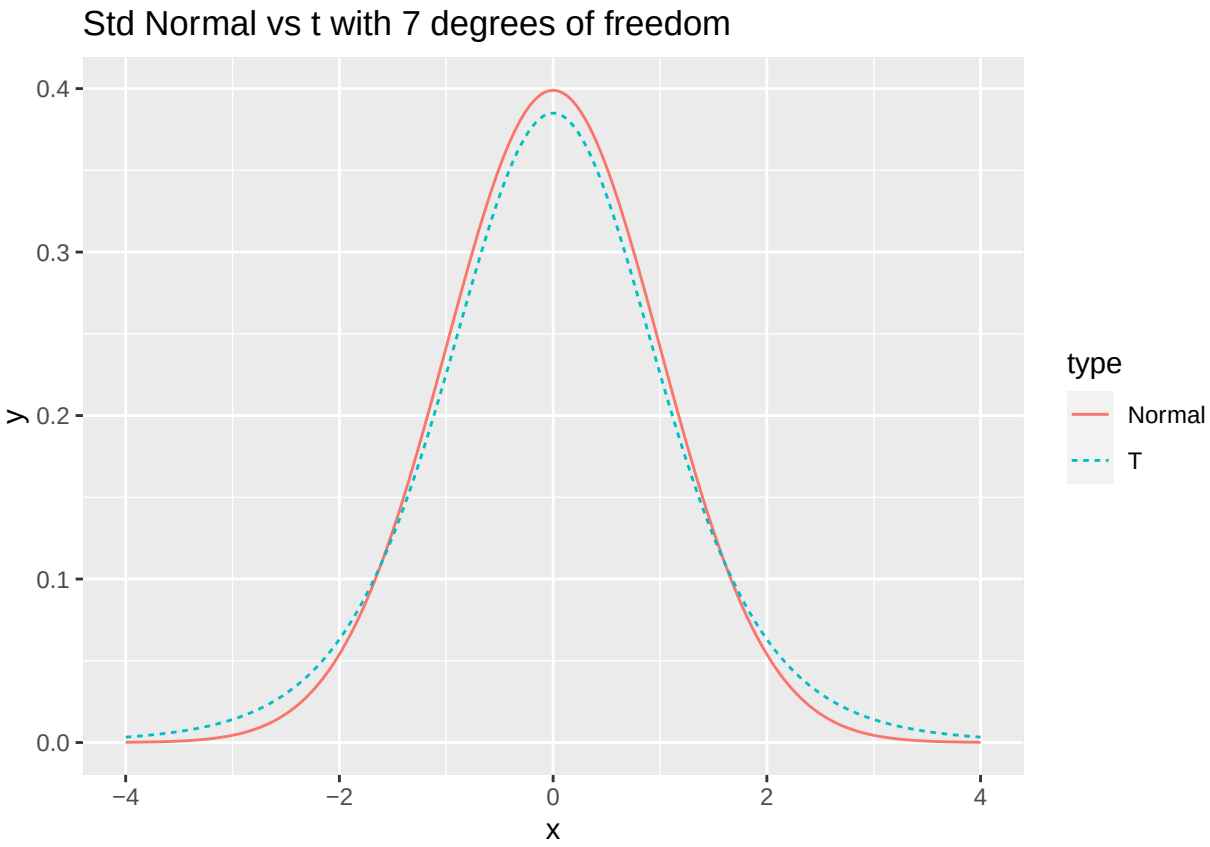

Std Normal vs t with 2 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

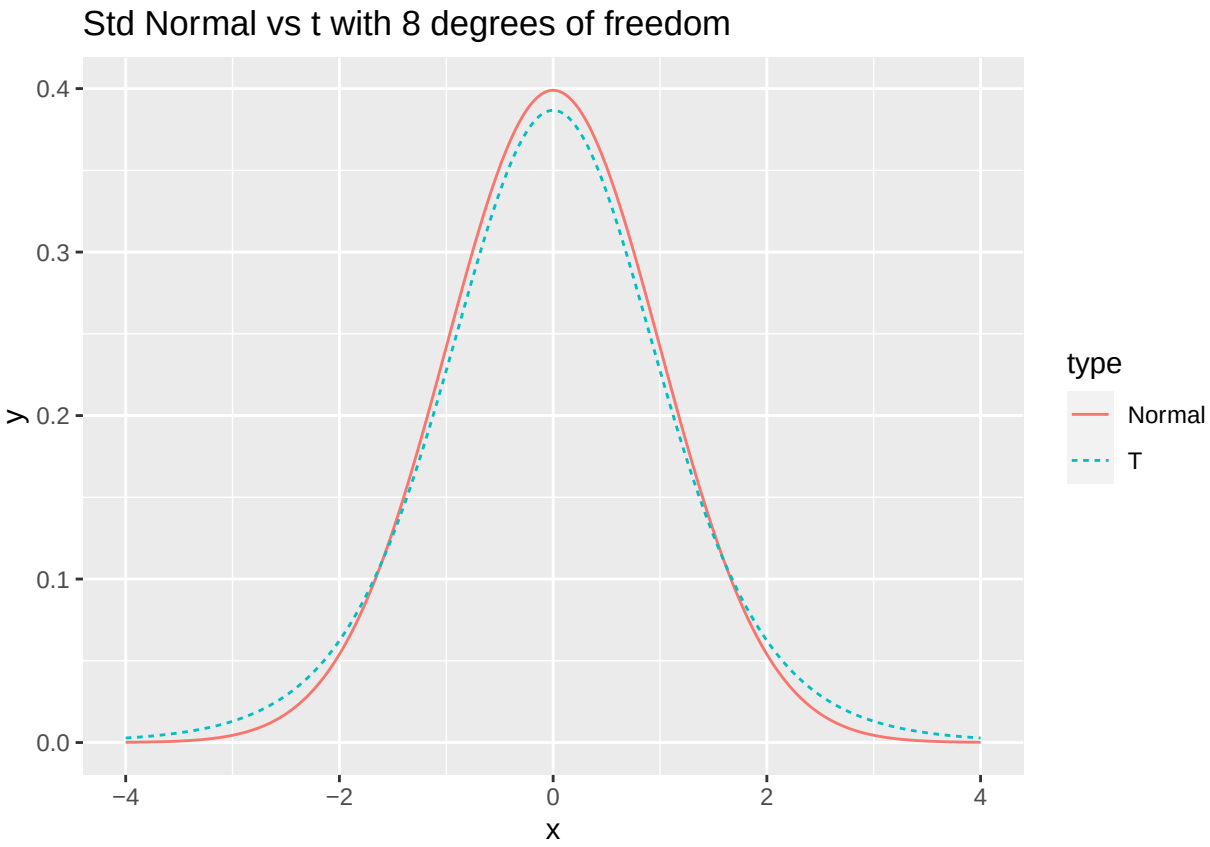

Std Normal vs t with 3 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

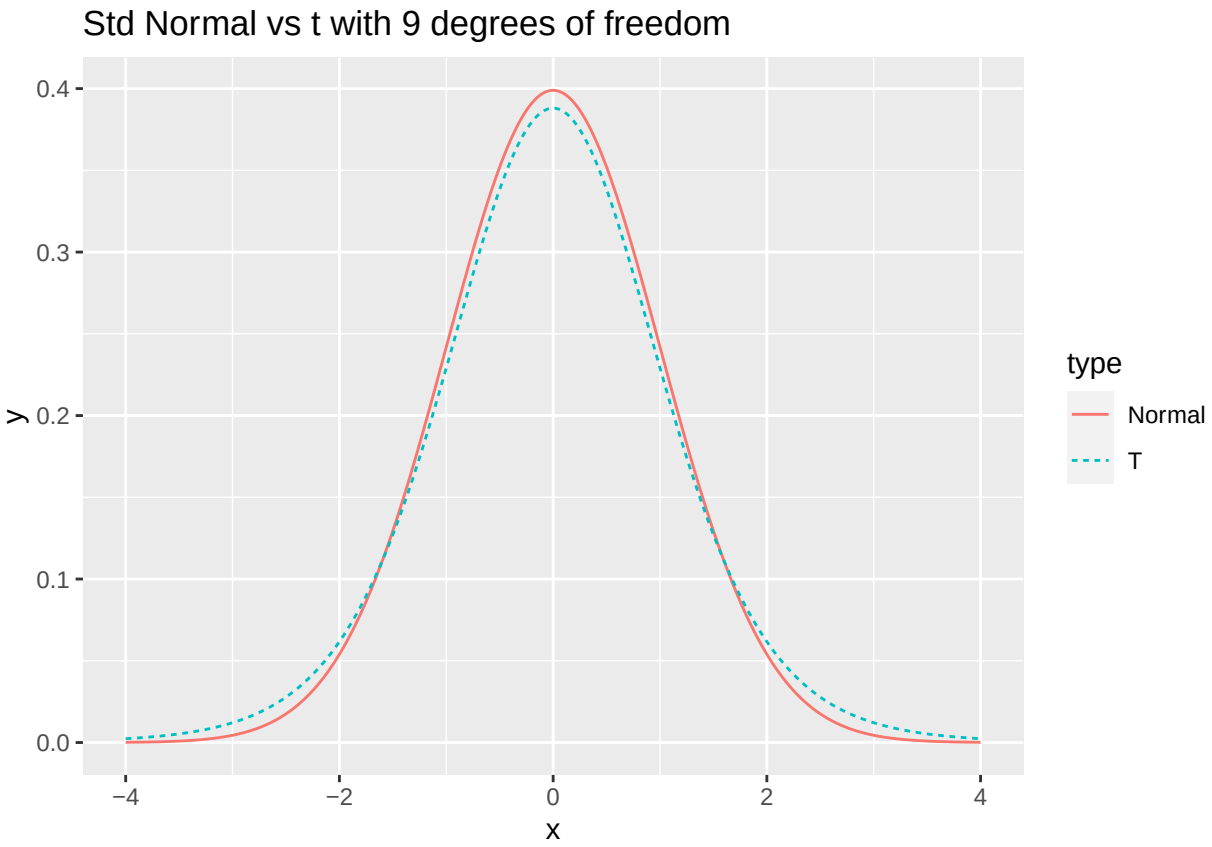

Std Normal vs t with 4 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

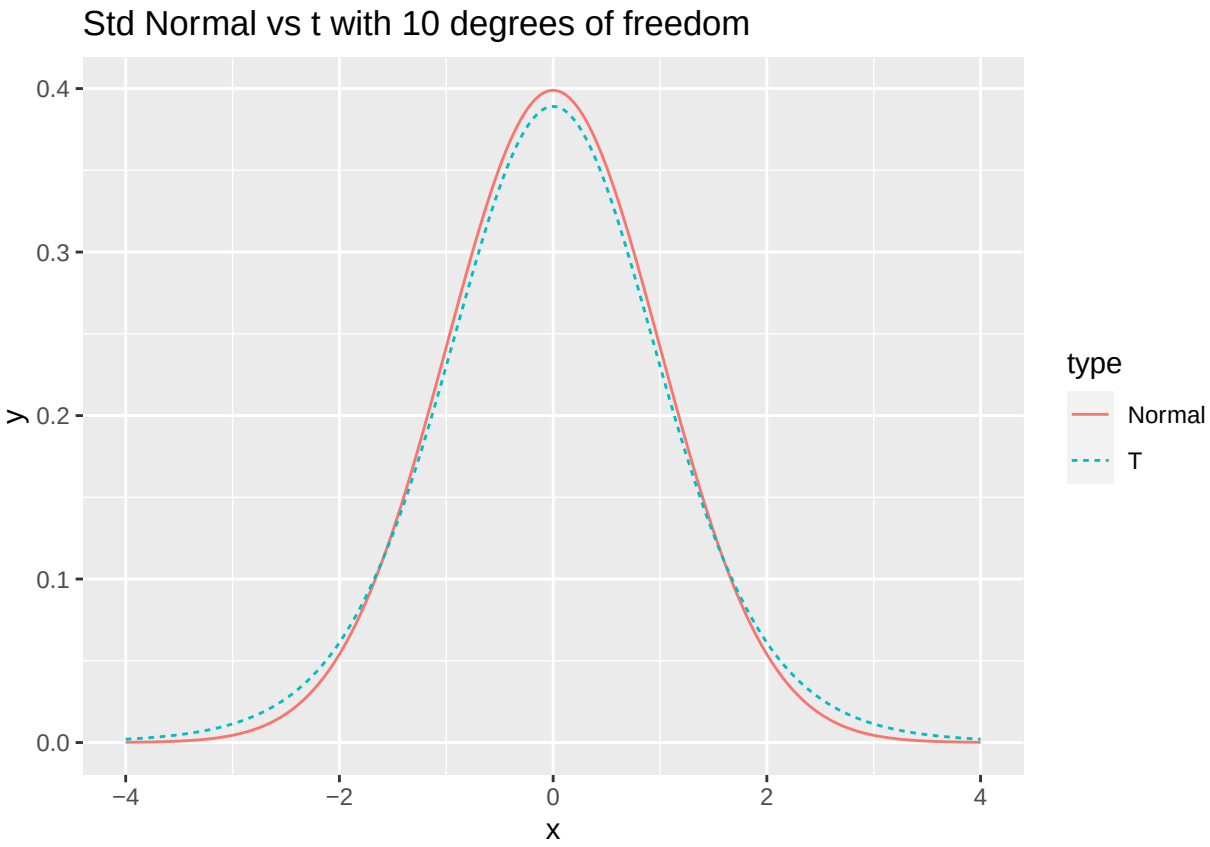

Std Normal vs t with 5 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

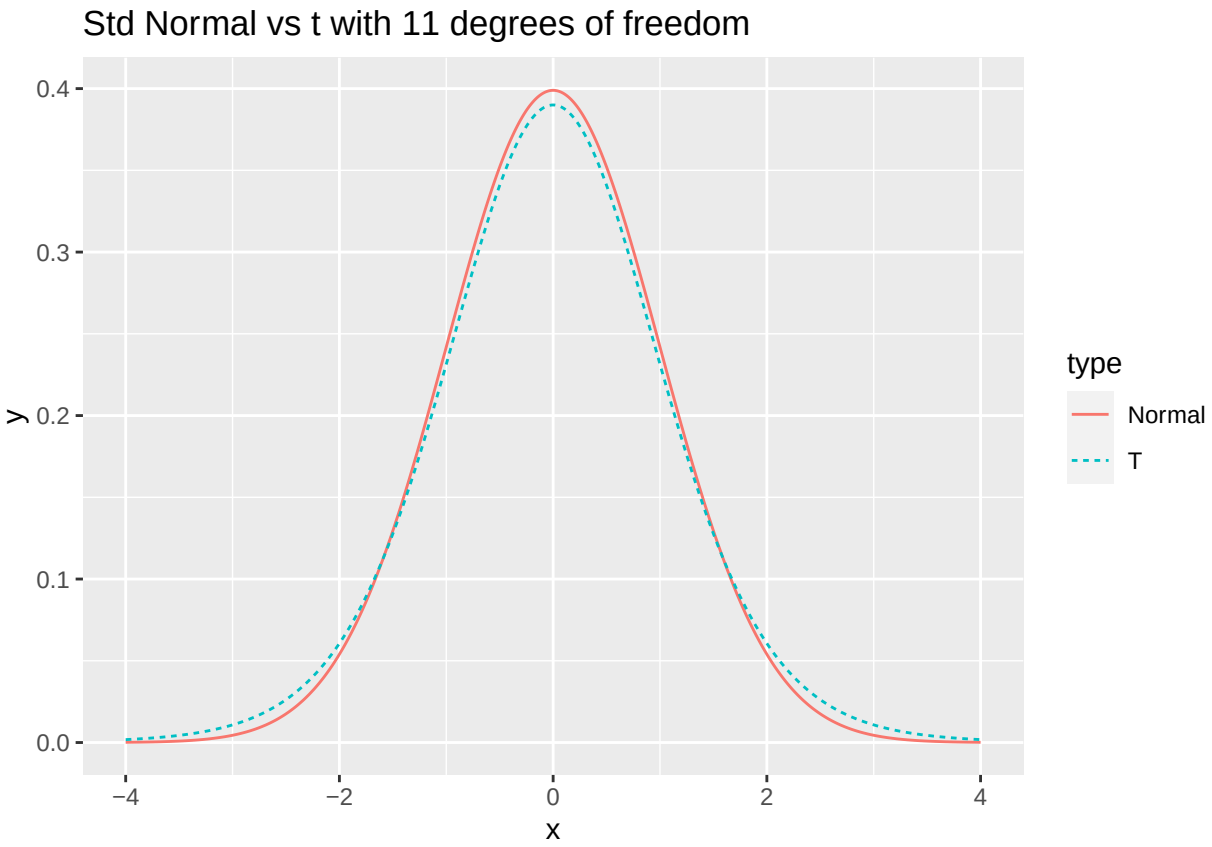

Std Normal vs t with 6 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

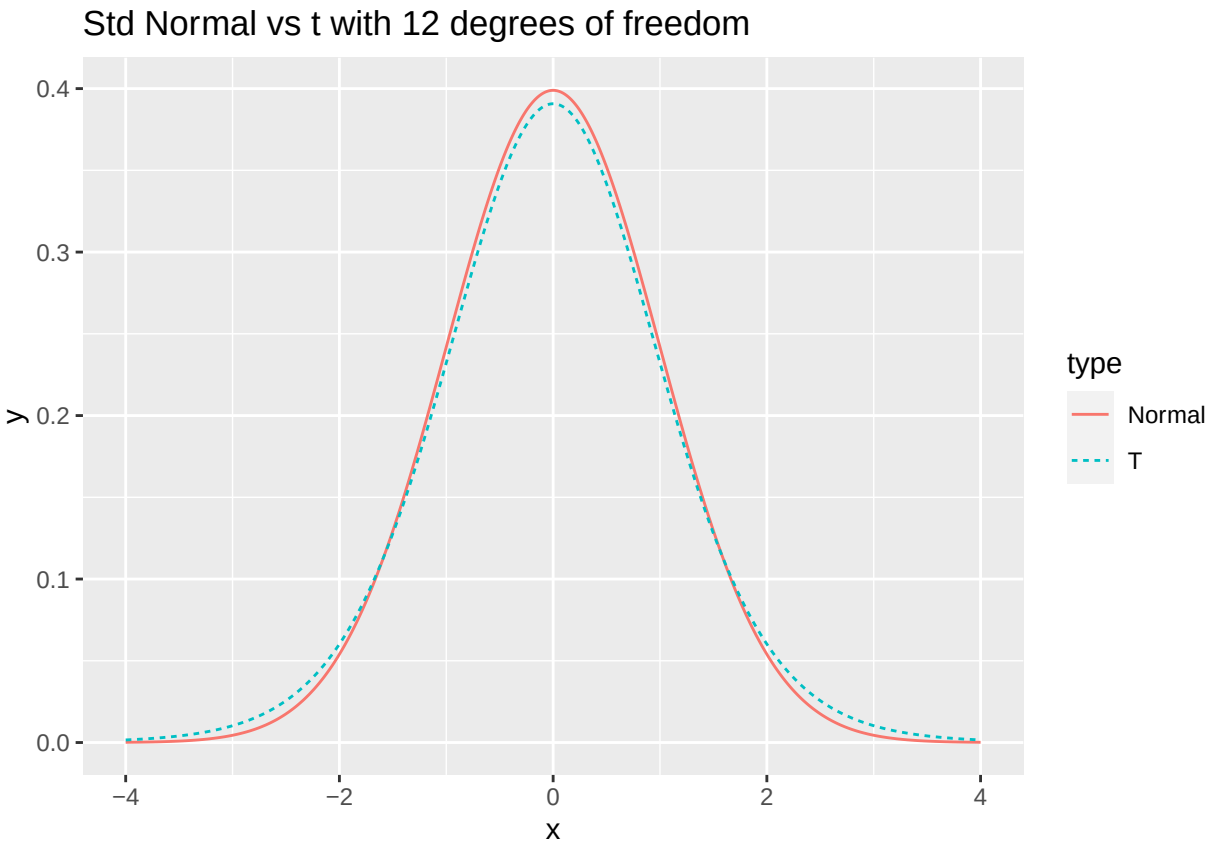

Std Normal vs t with 7 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

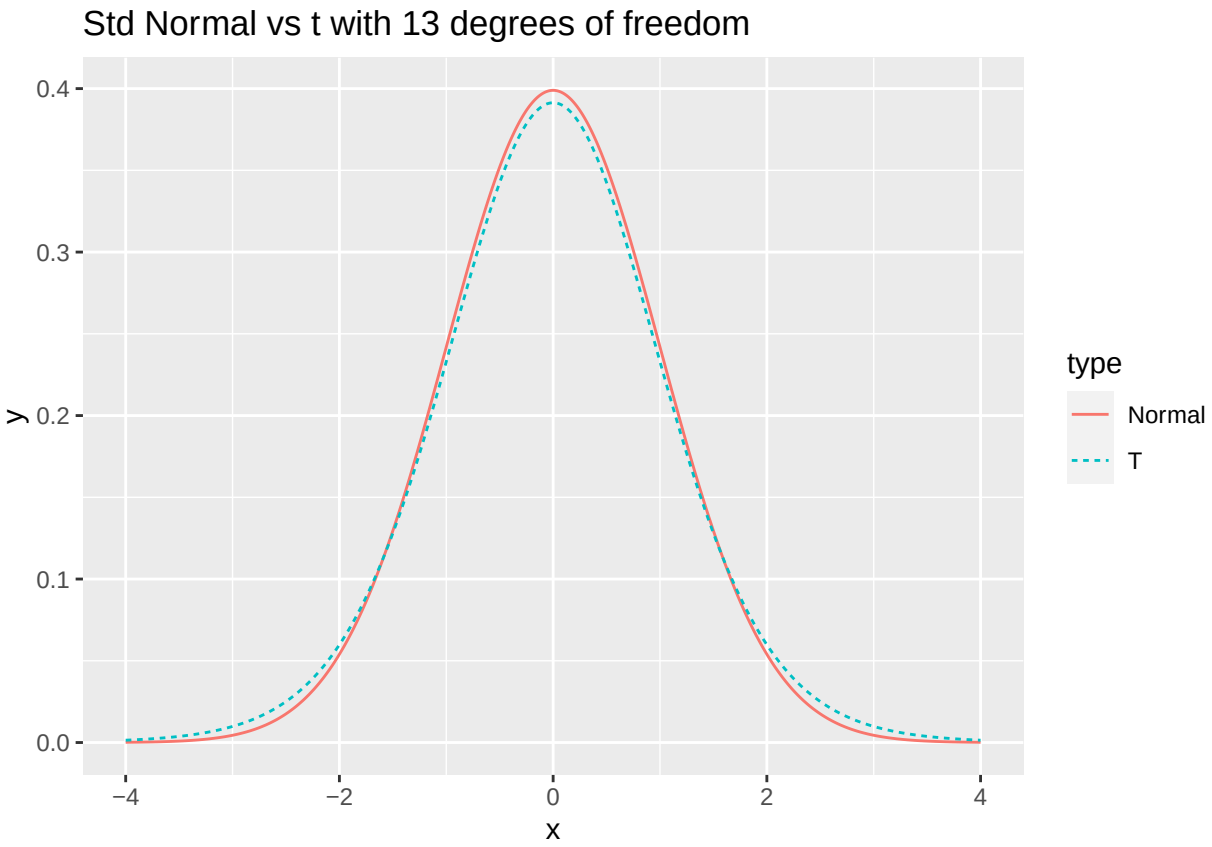

Std Normal vs t with 8 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

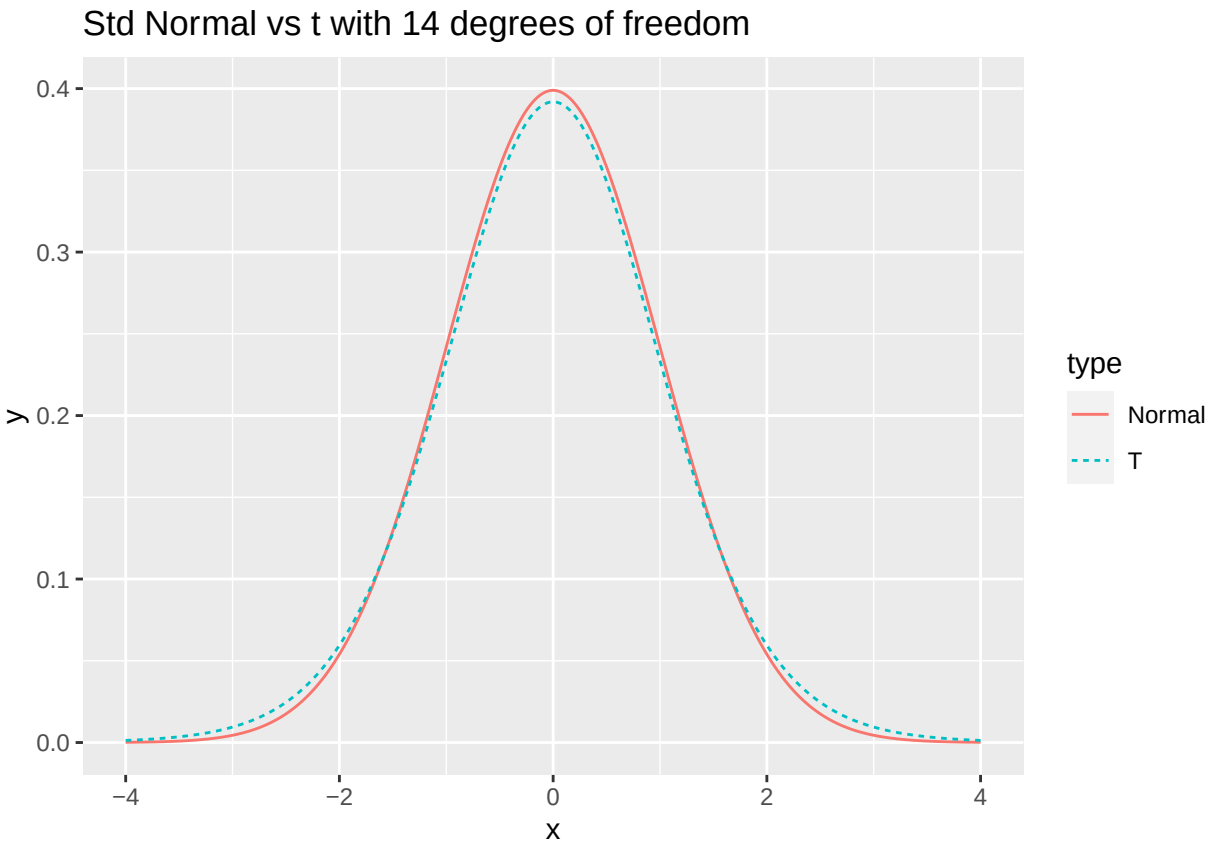

Std Normal vs t with 9 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

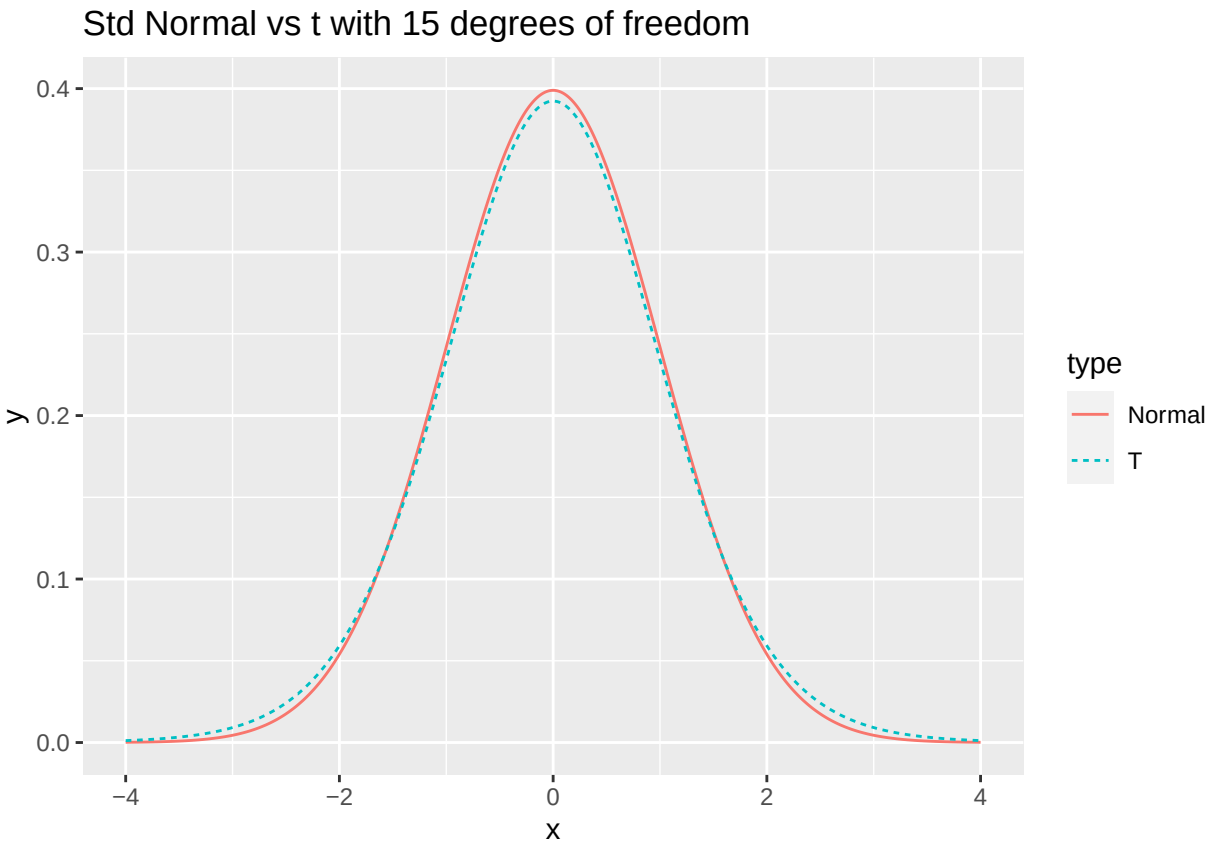

Std Normal vs t with 10 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

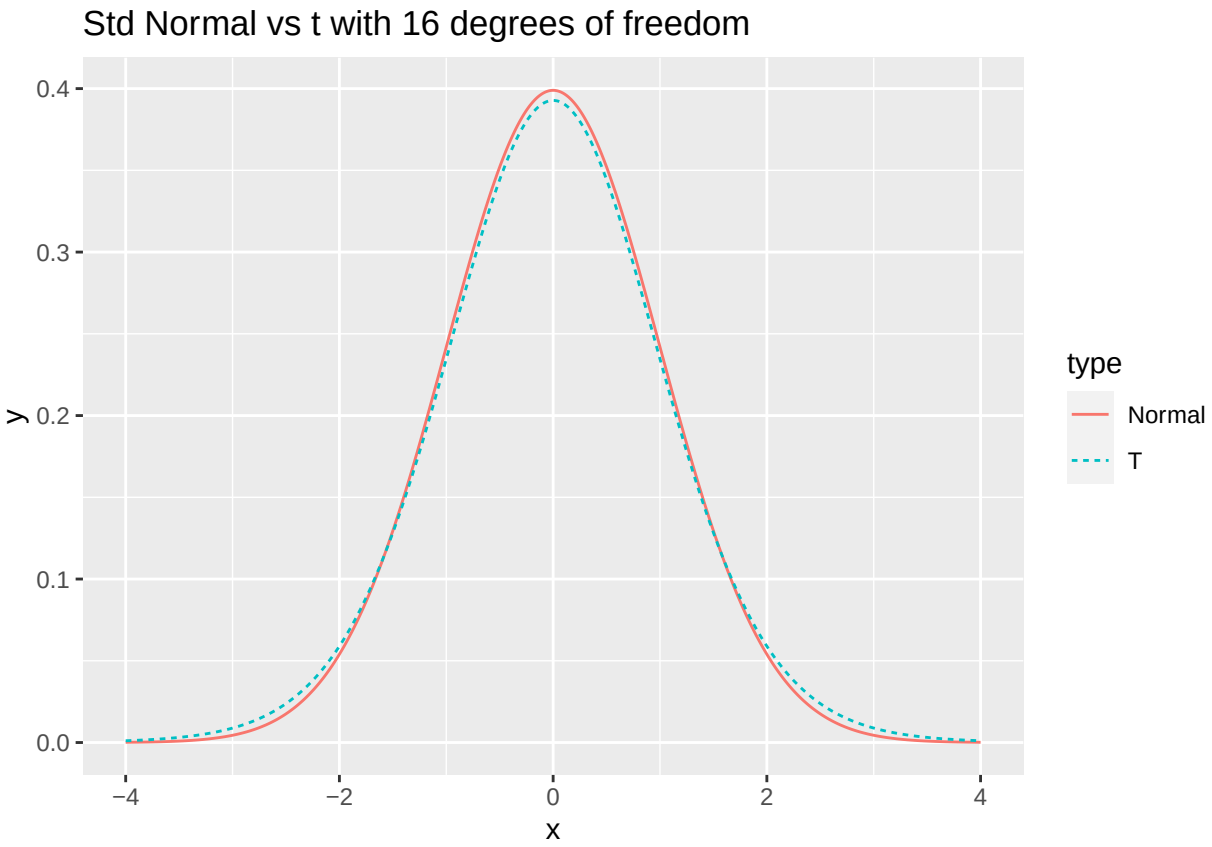

Std Normal vs t with 11 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

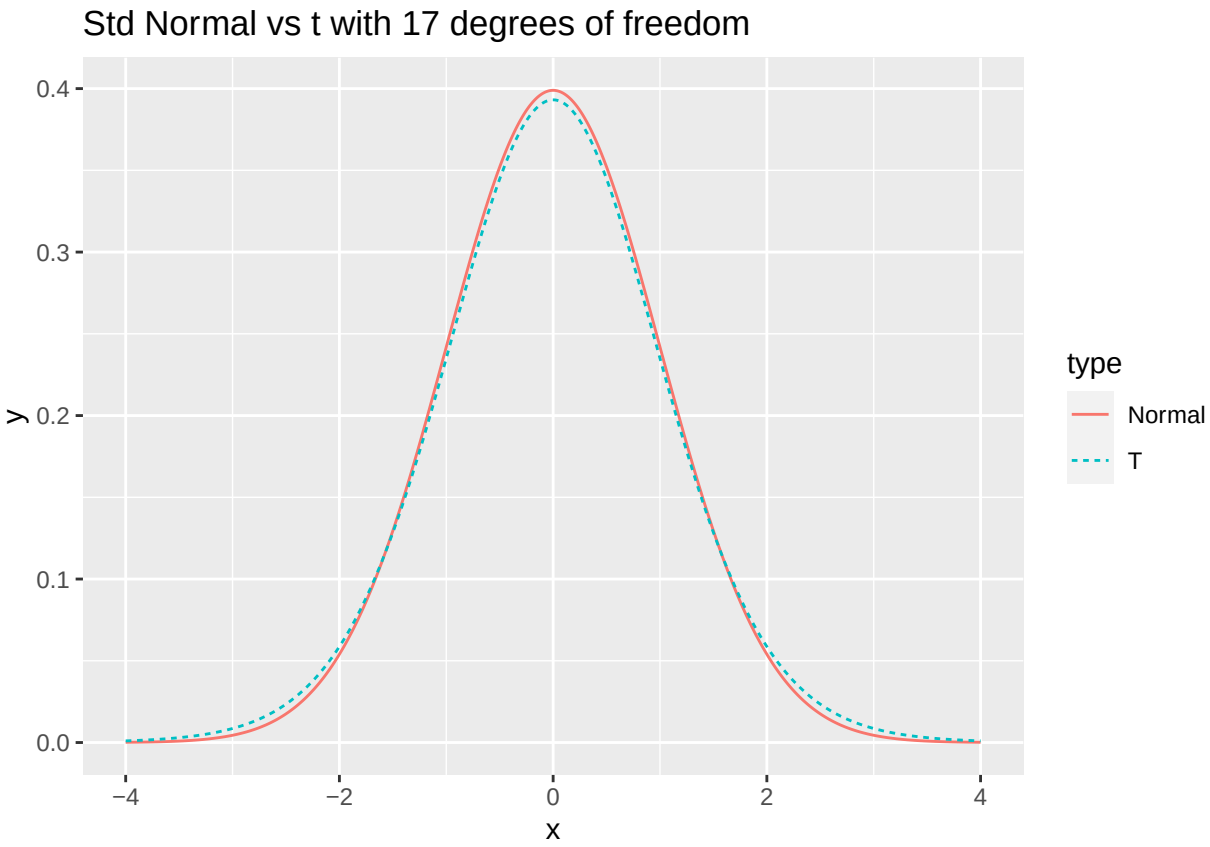

Std Normal vs t with 12 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

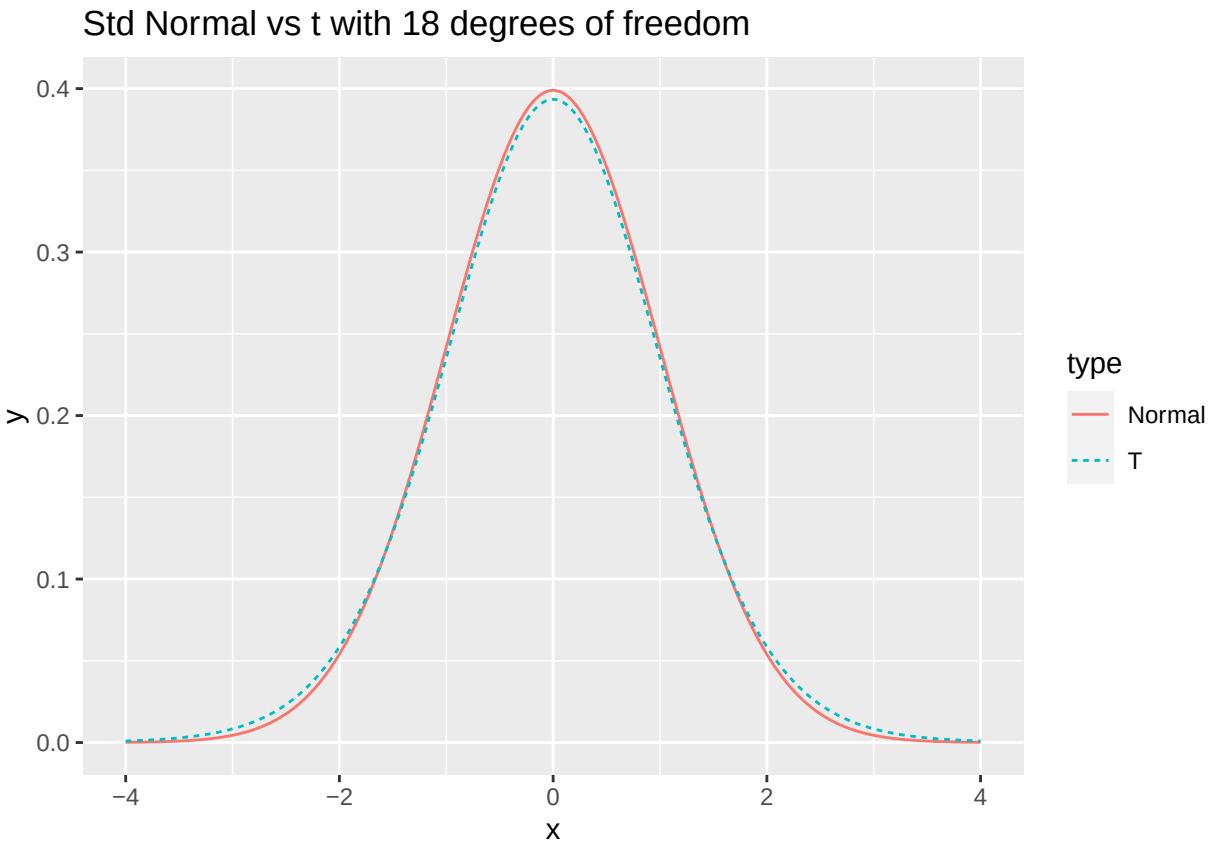

Std Normal vs t with 13 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

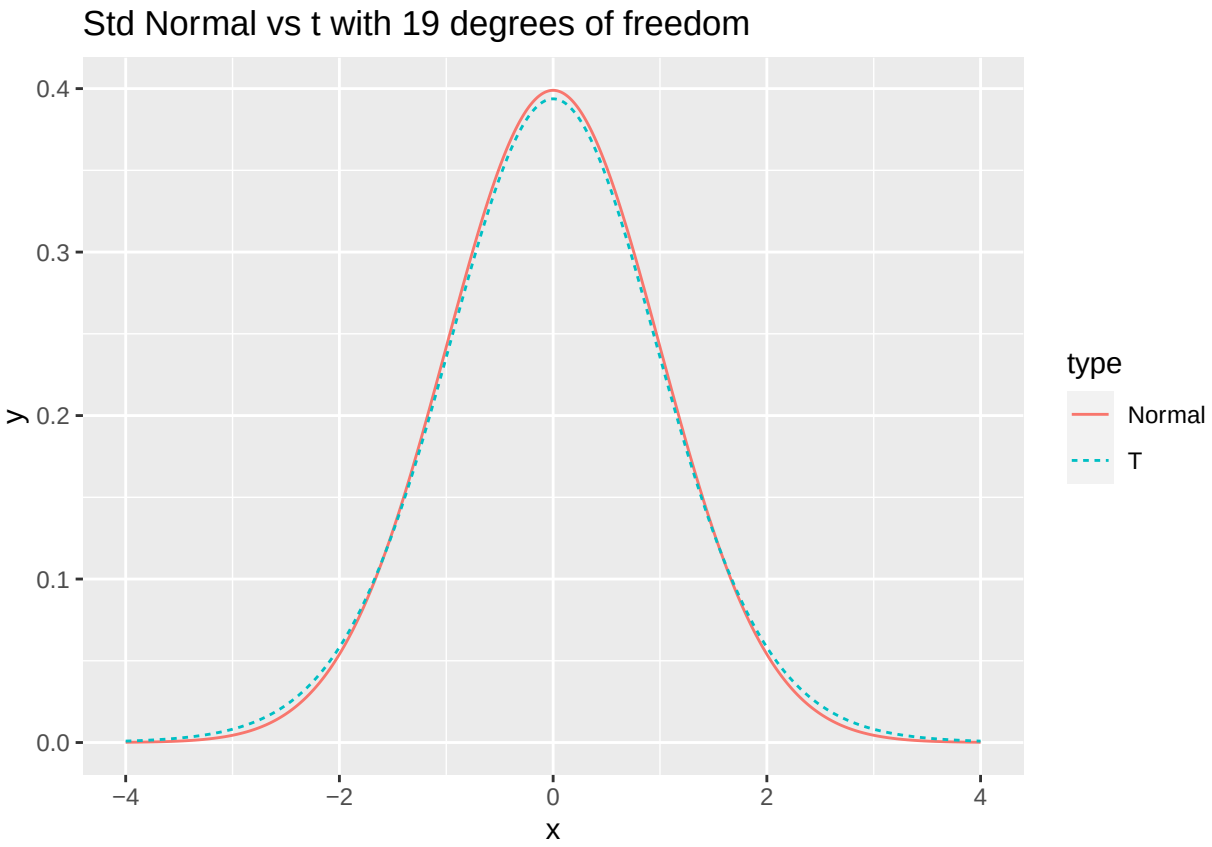

Std Normal vs t with 14 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

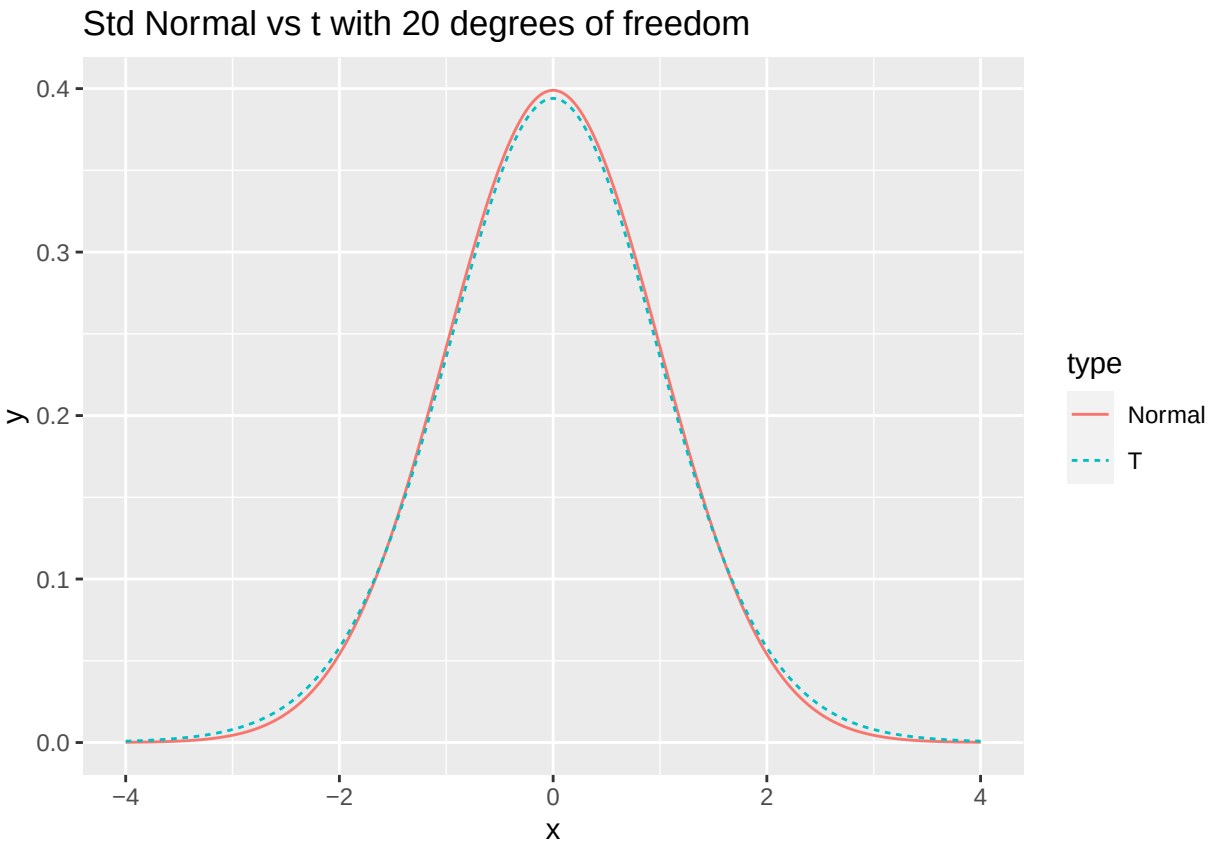

Std Normal vs t with 15 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

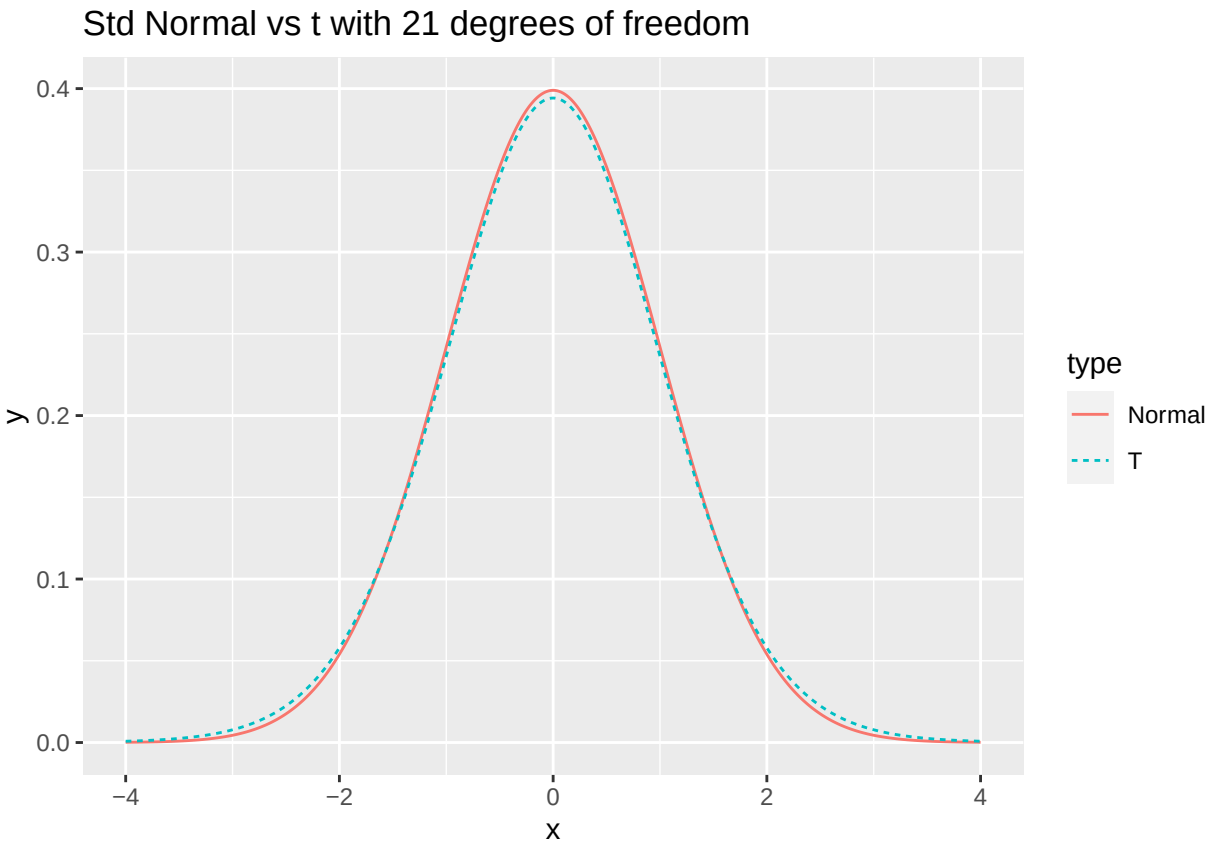

Std Normal vs t with 16 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

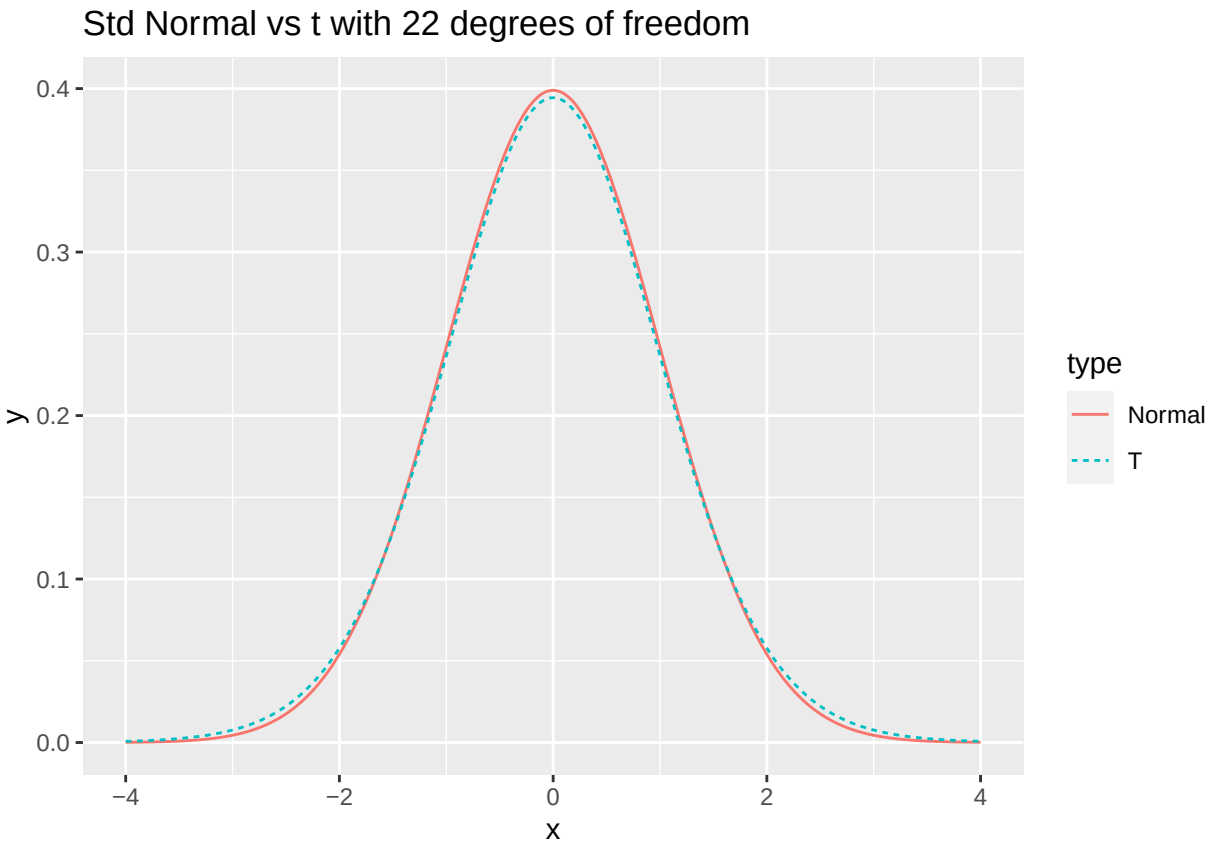

Std Normal vs t with 17 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

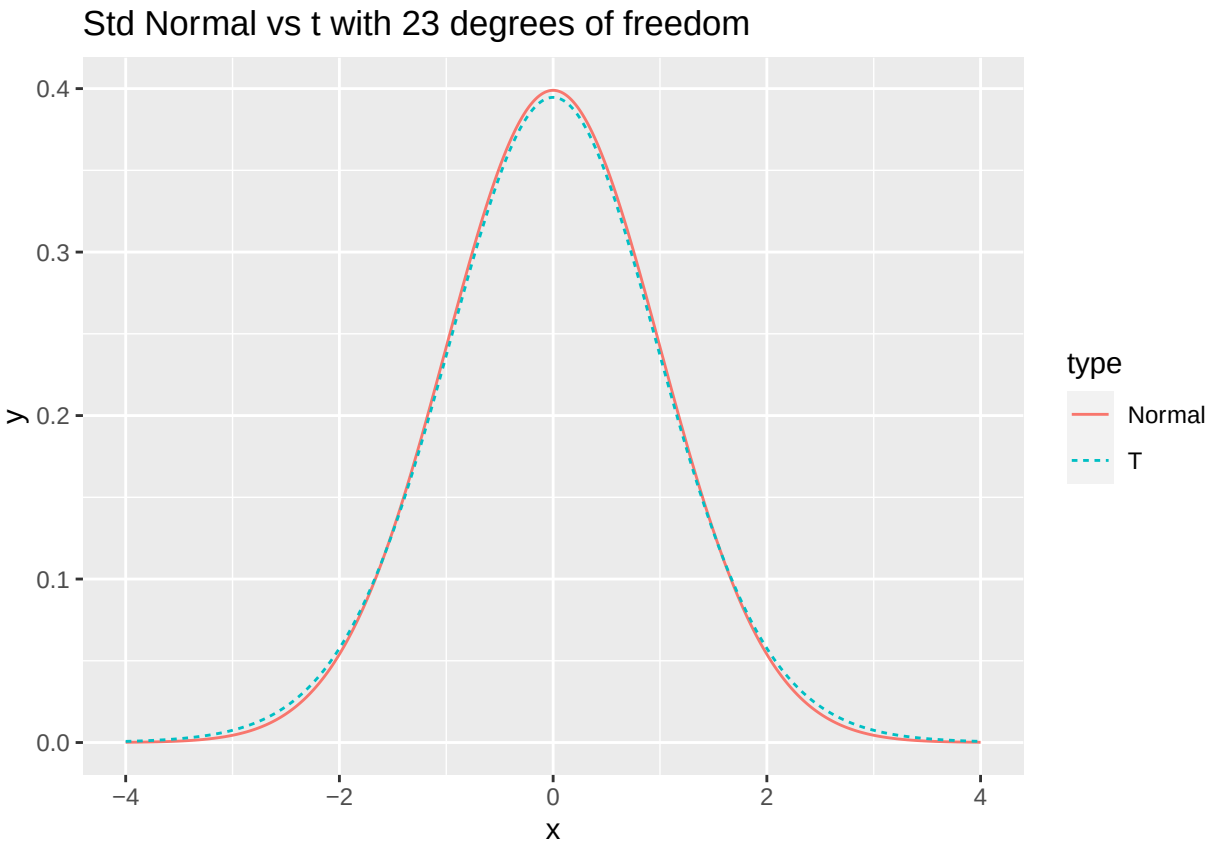

Std Normal vs t with 18 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

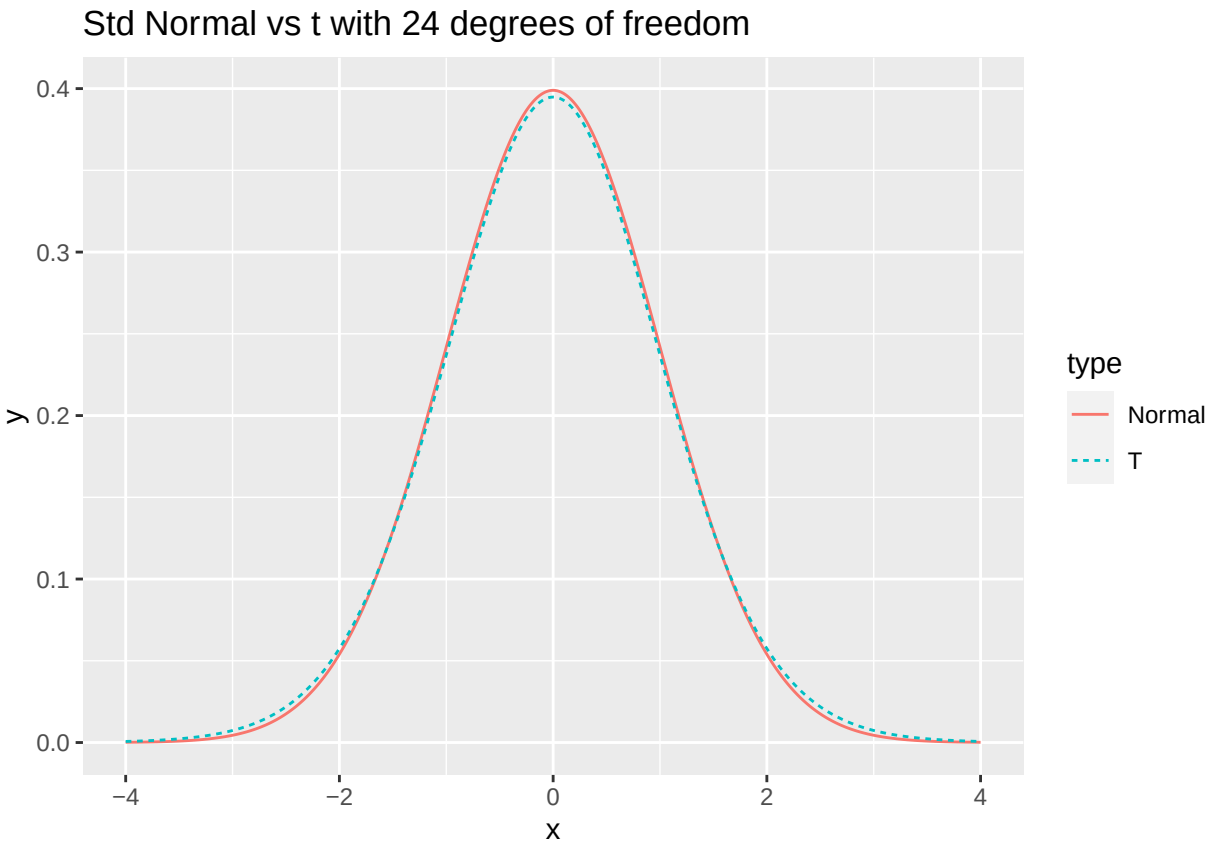

Std Normal vs t with 19 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

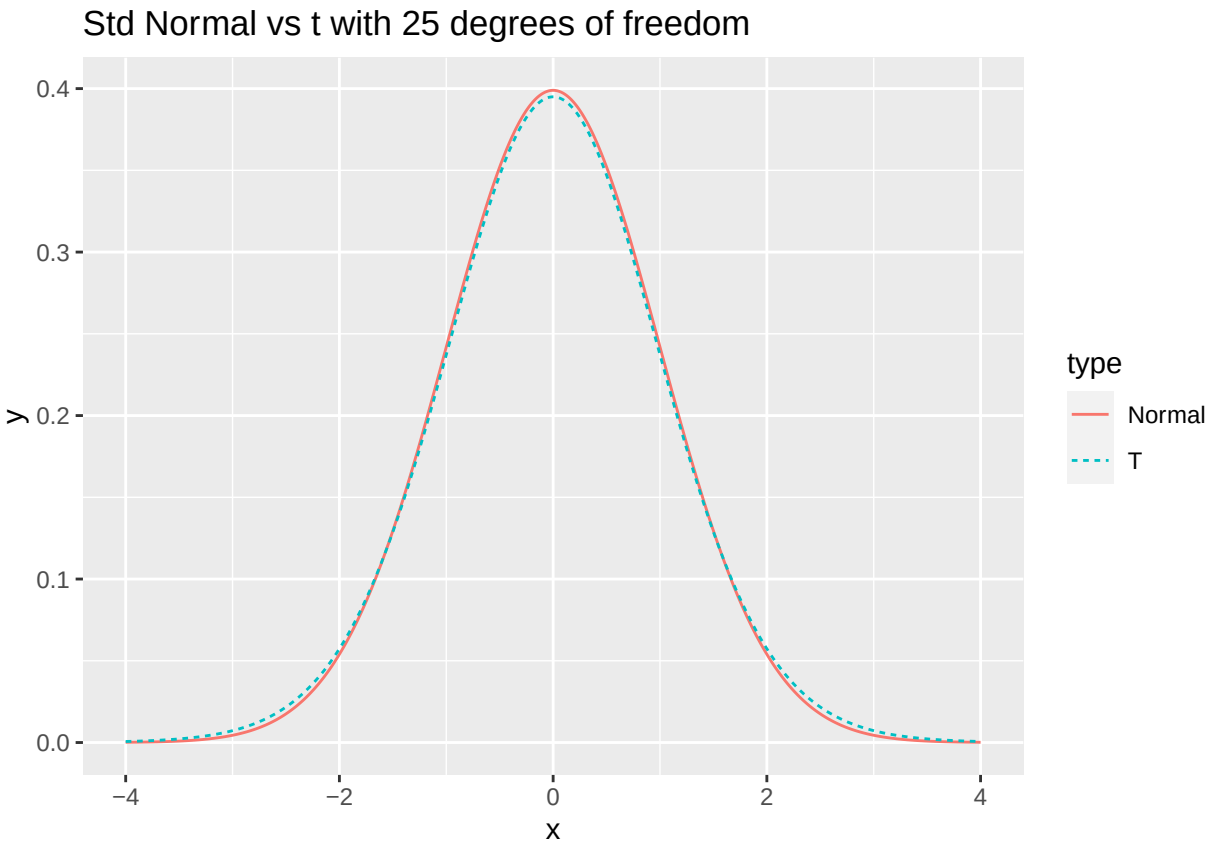

Std Normal vs t with 20 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

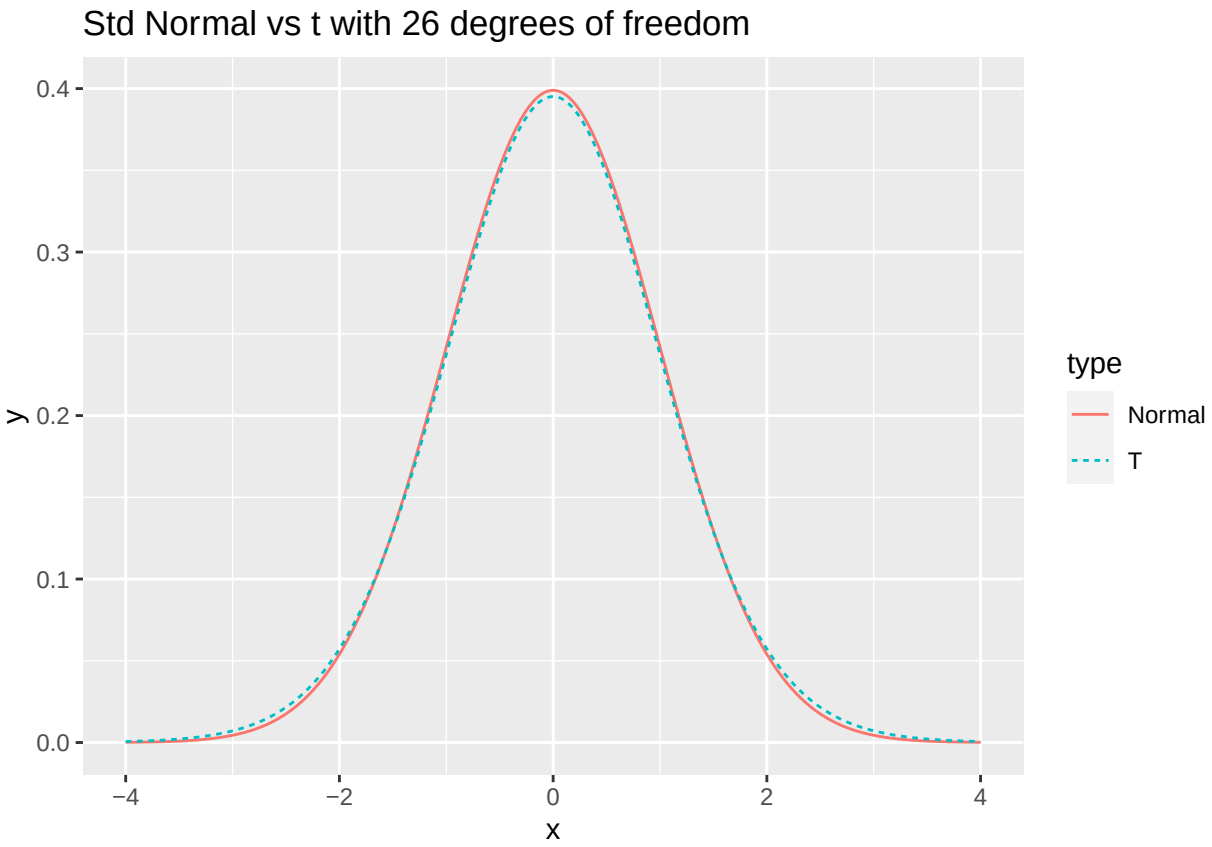

Std Normal vs t with 21 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

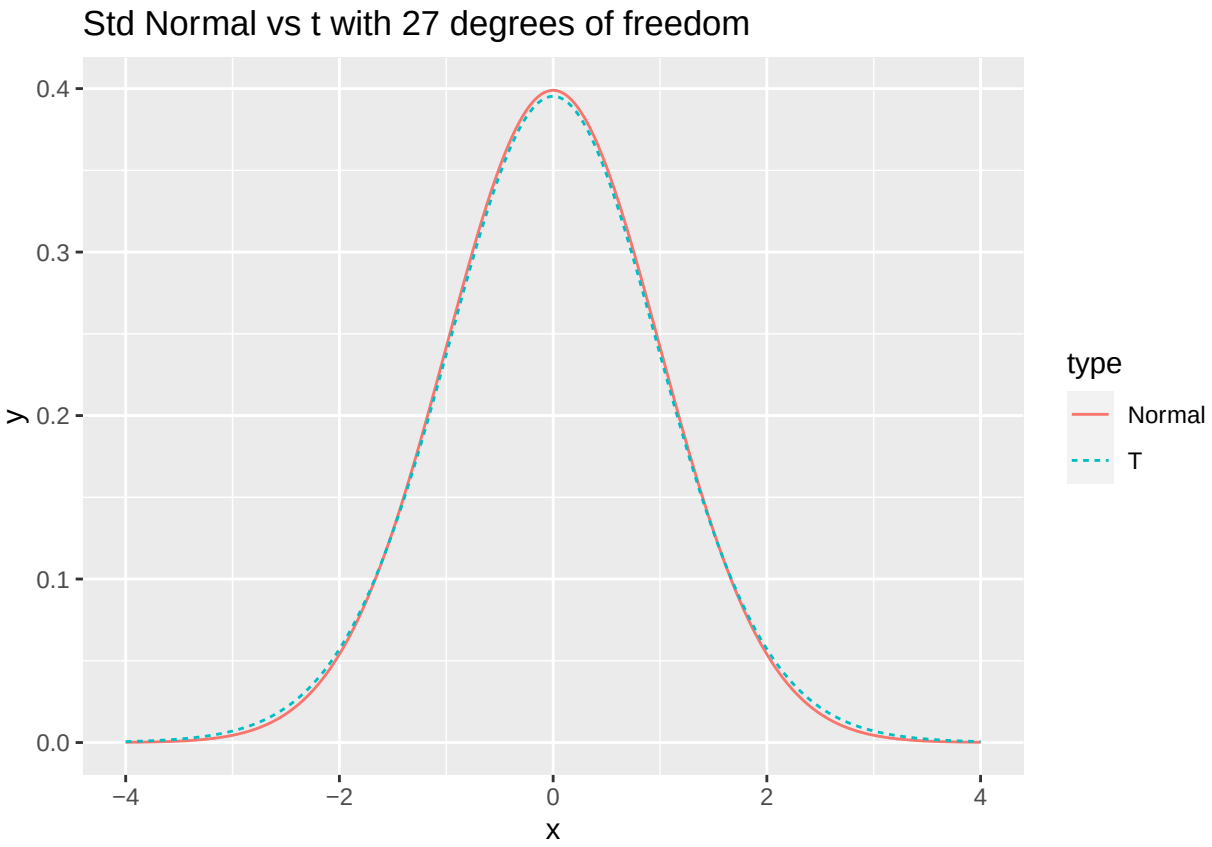

Std Normal vs t with 22 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

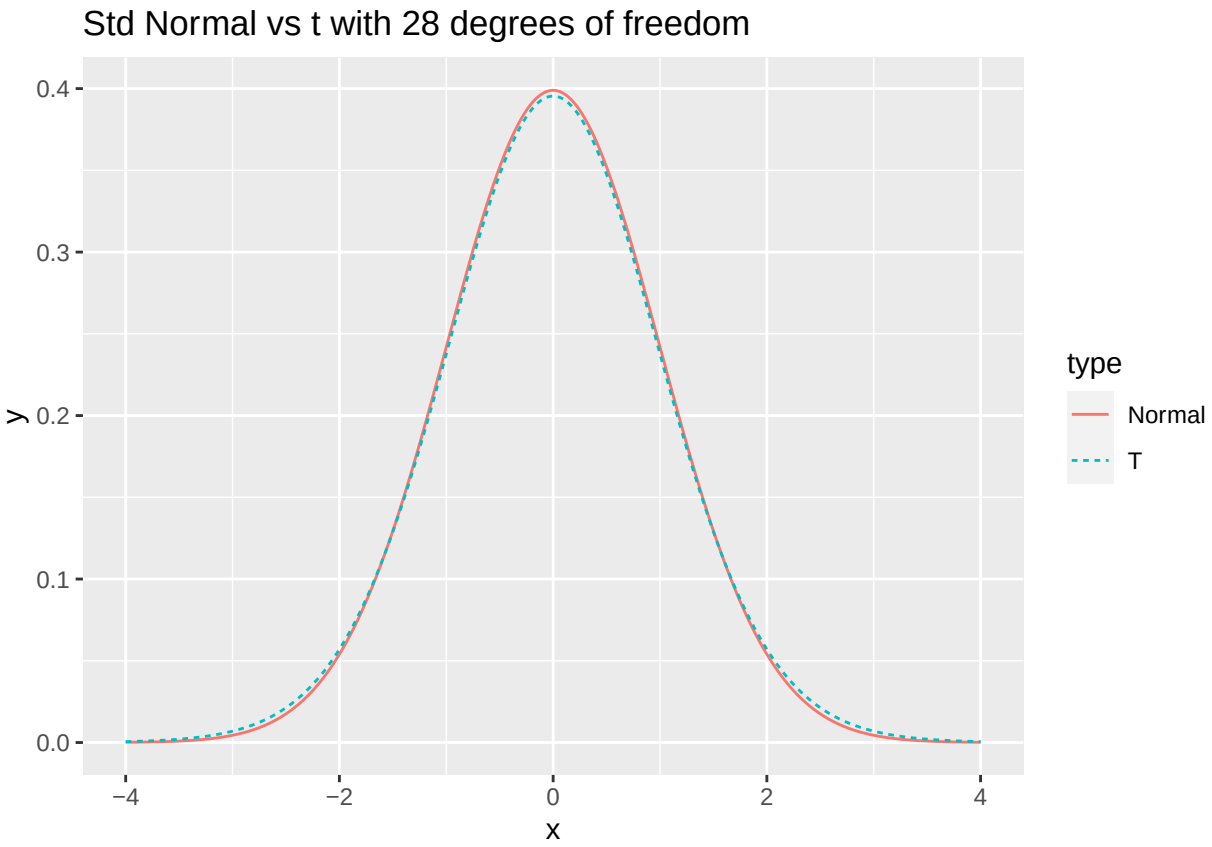

Std Normal vs t with 23 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

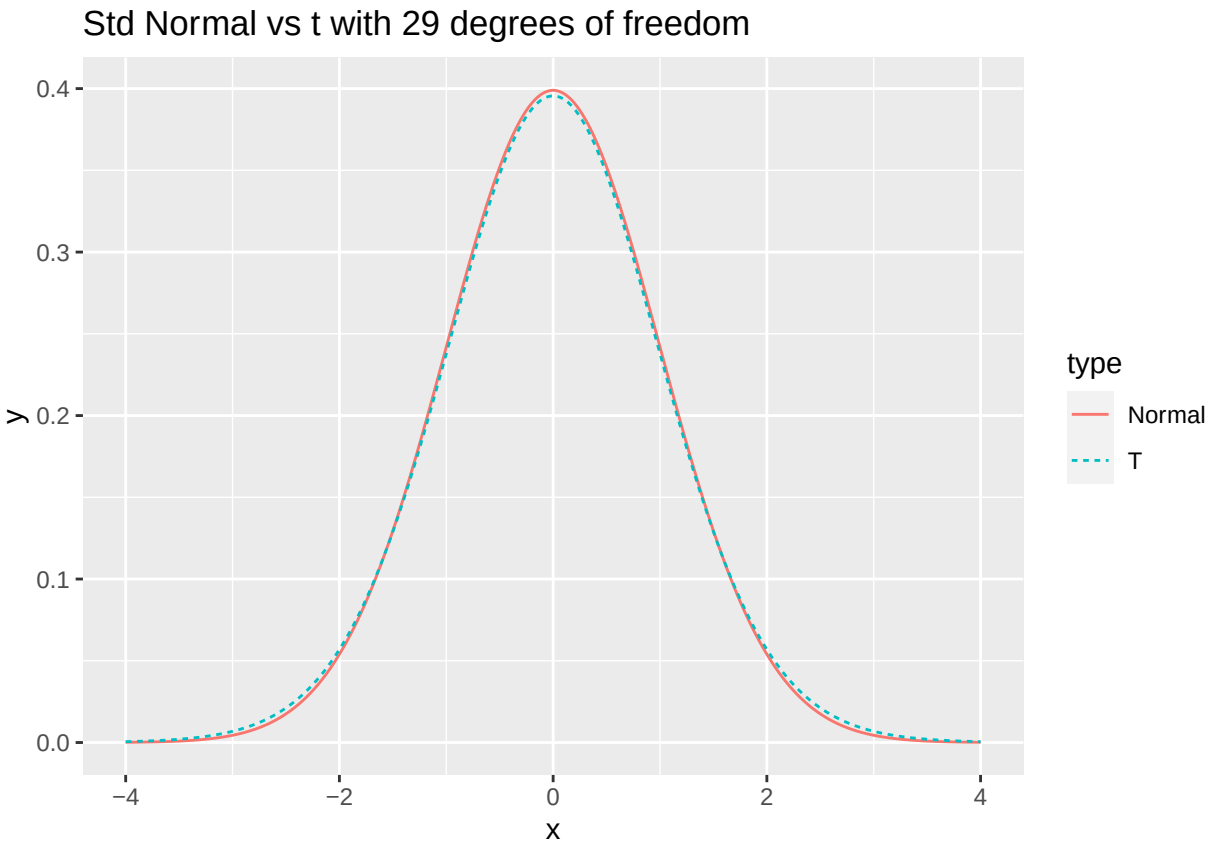

Std Normal vs t with 24 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

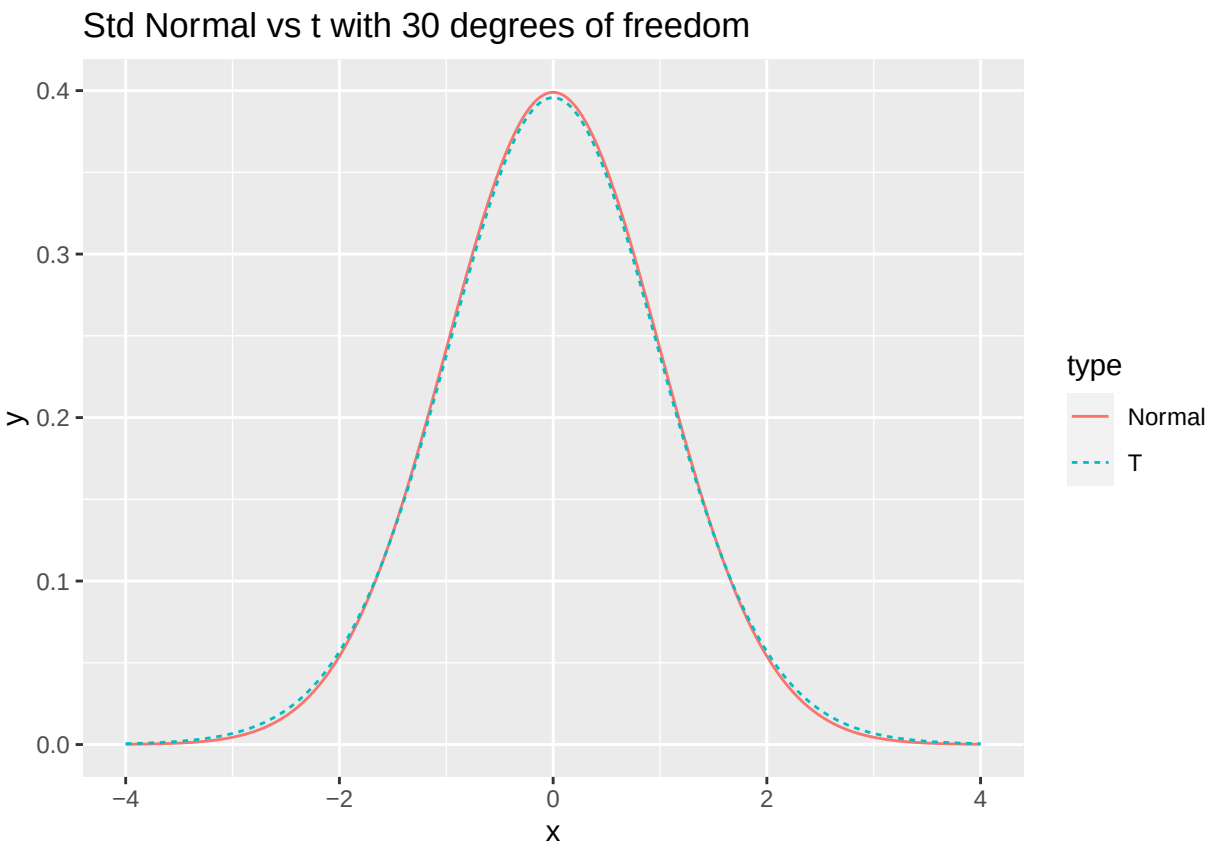

Std Normal vs t with 25 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T


Std Normal vs t with 26 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T


Std Normal vs t with 27 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

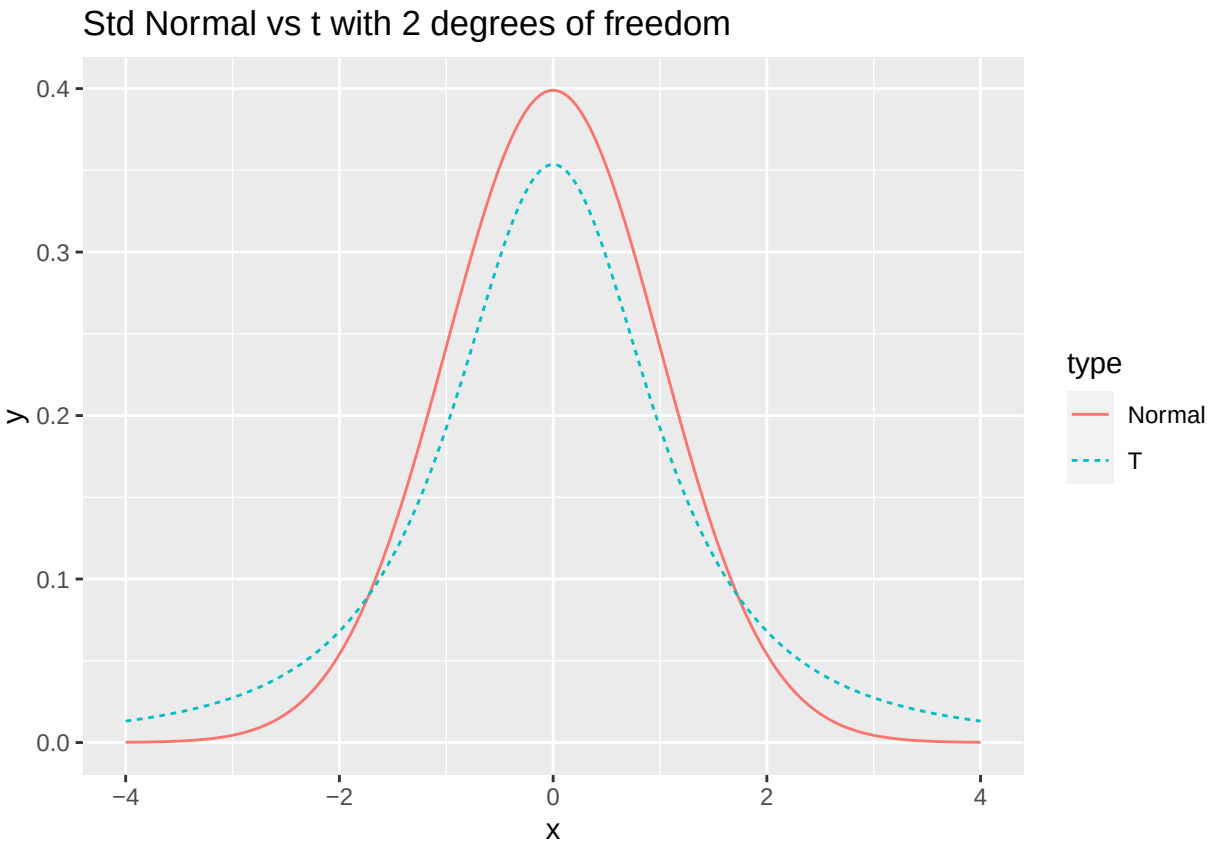

Std Normal vs t with 28 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

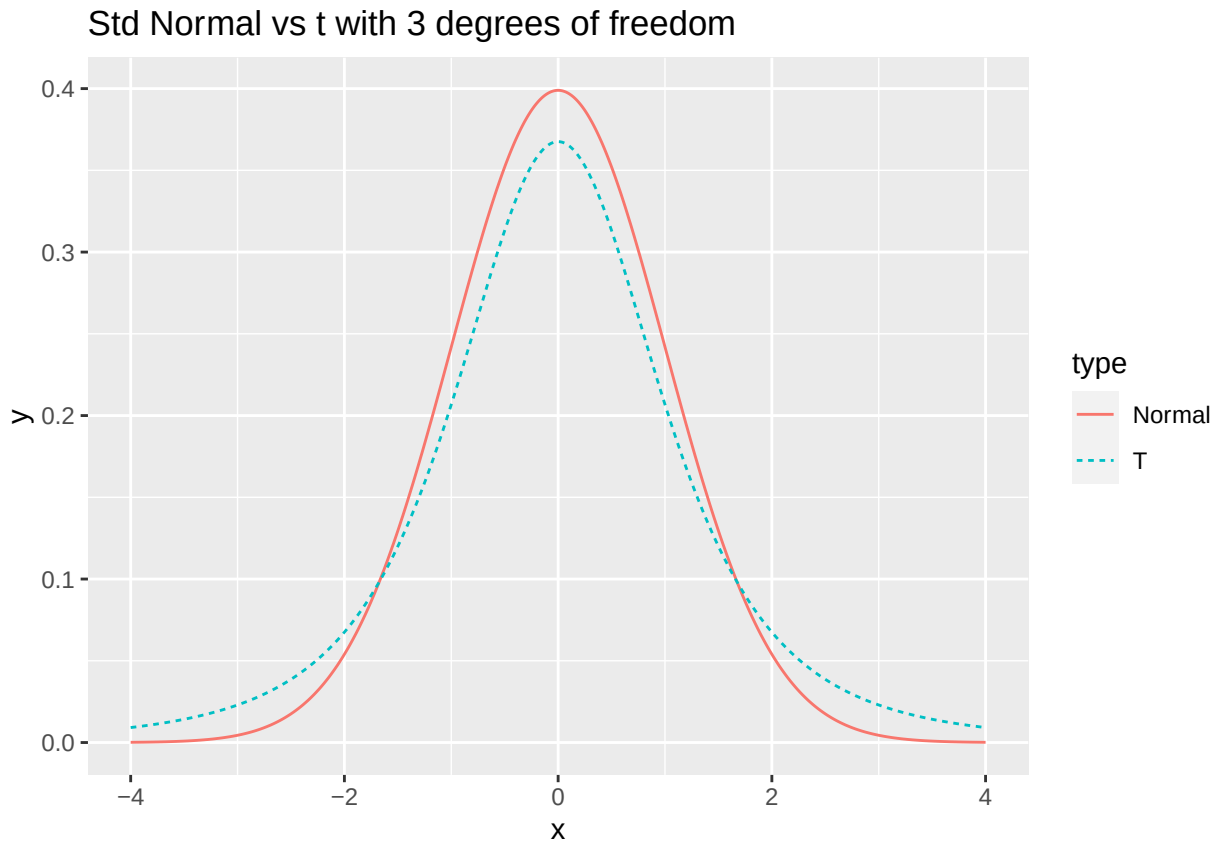

Std Normal vs t with 29 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

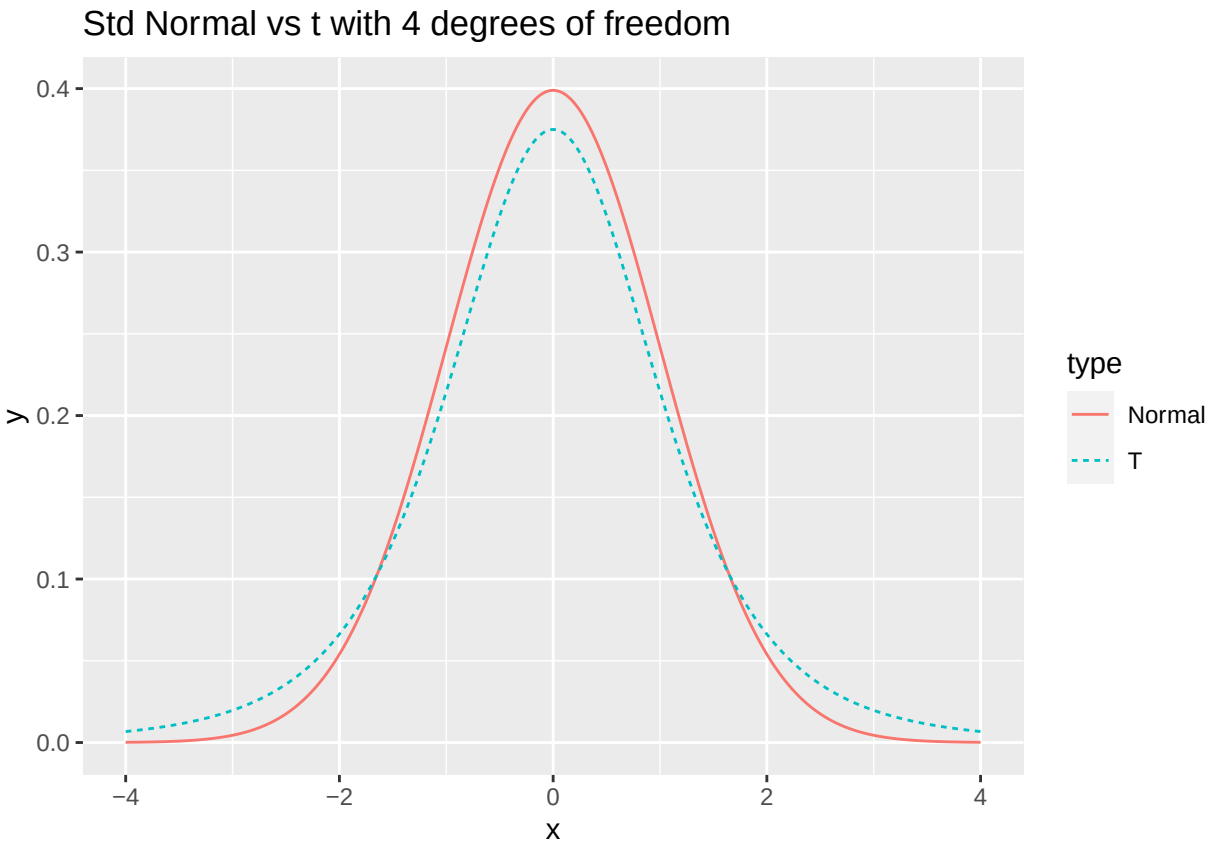


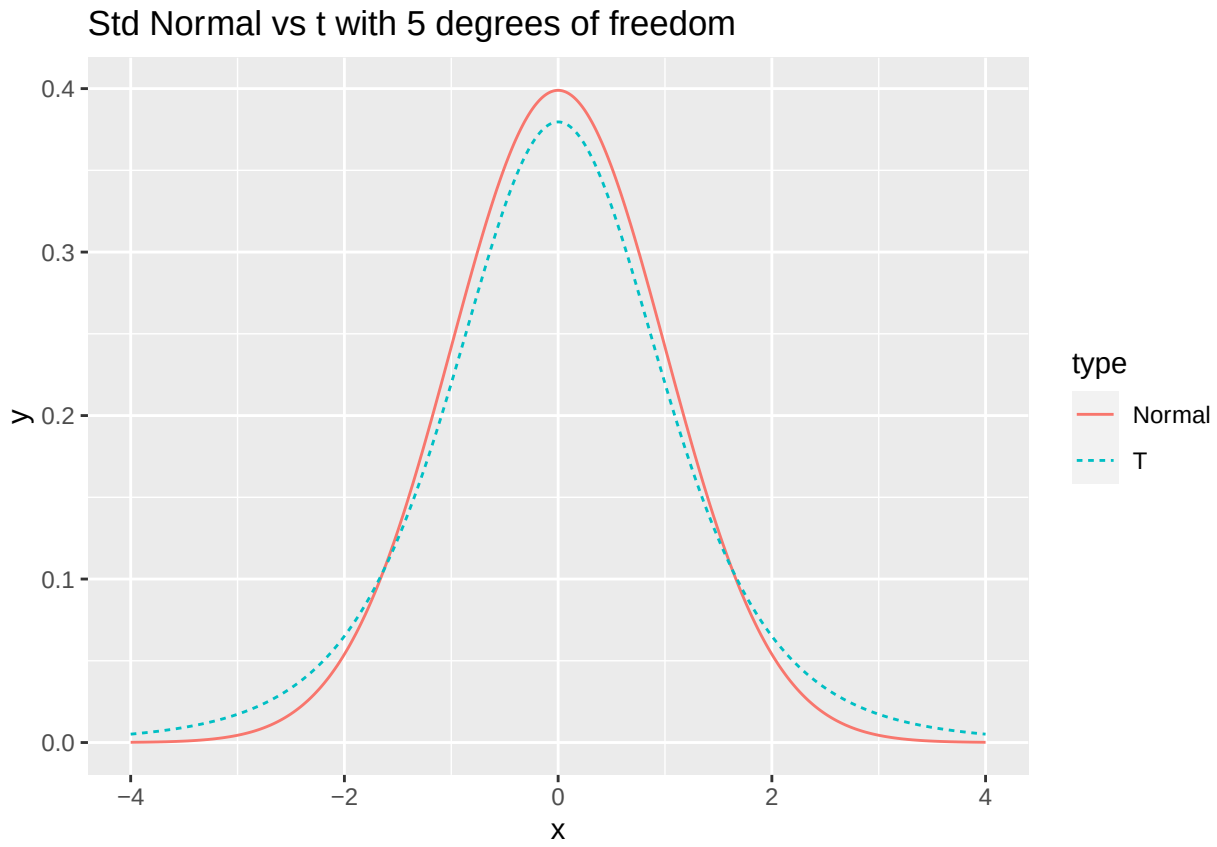
b. Use a for loop to create similar graphs for degrees of freedom 2,3,4,5,10,15,20 25,30

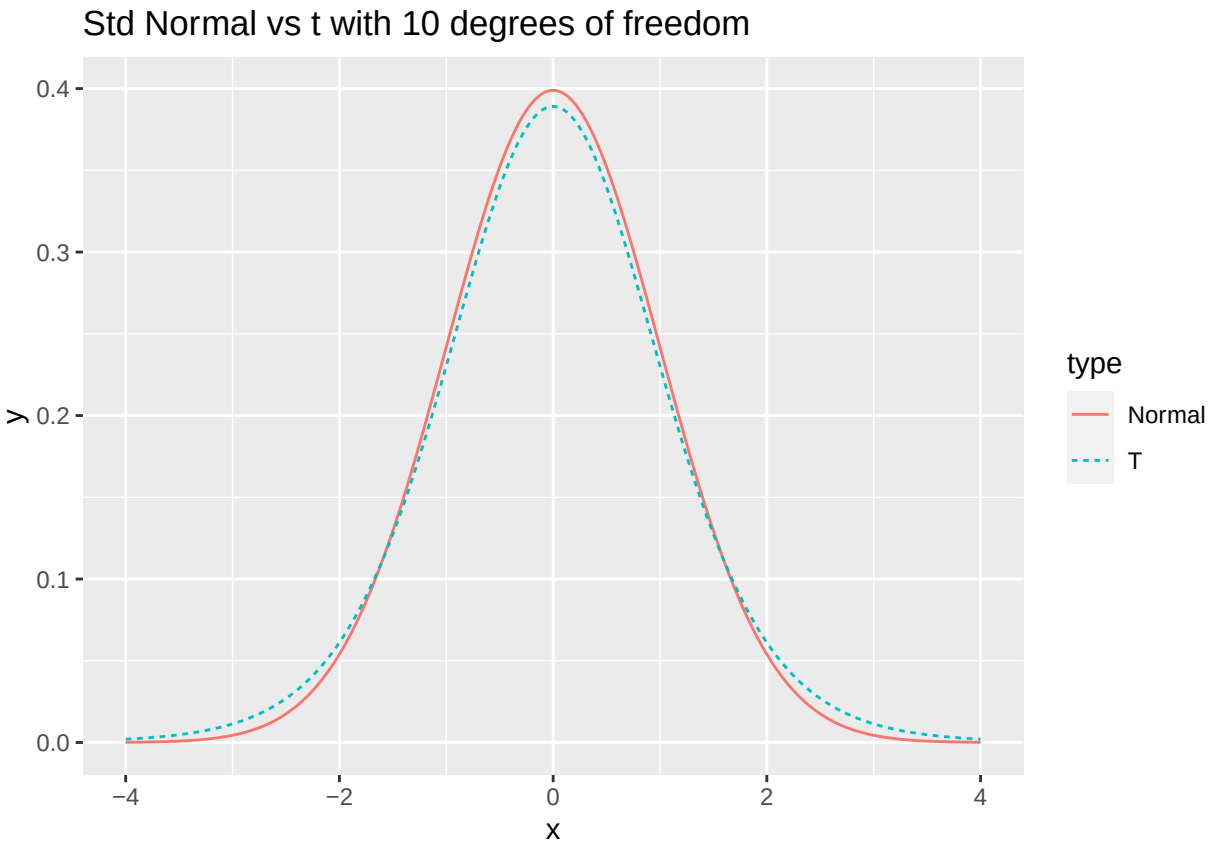
```
dfVec = c(2,3,4,5,10,15,20,25,30)
for(df in dfVec){
  # Update our y column for each df value
  data$y <- c(dnorm(x), dt(x, df))

  # Make a graph to reflect each new df value
  myplot <- ggplot(data, aes(x=x, y=y, color=type, linetype=type)) +
    geom_line() +
    labs(title = paste('Std Normal vs t with', df, 'degrees of freedom'))

  print(myplot)
}
```

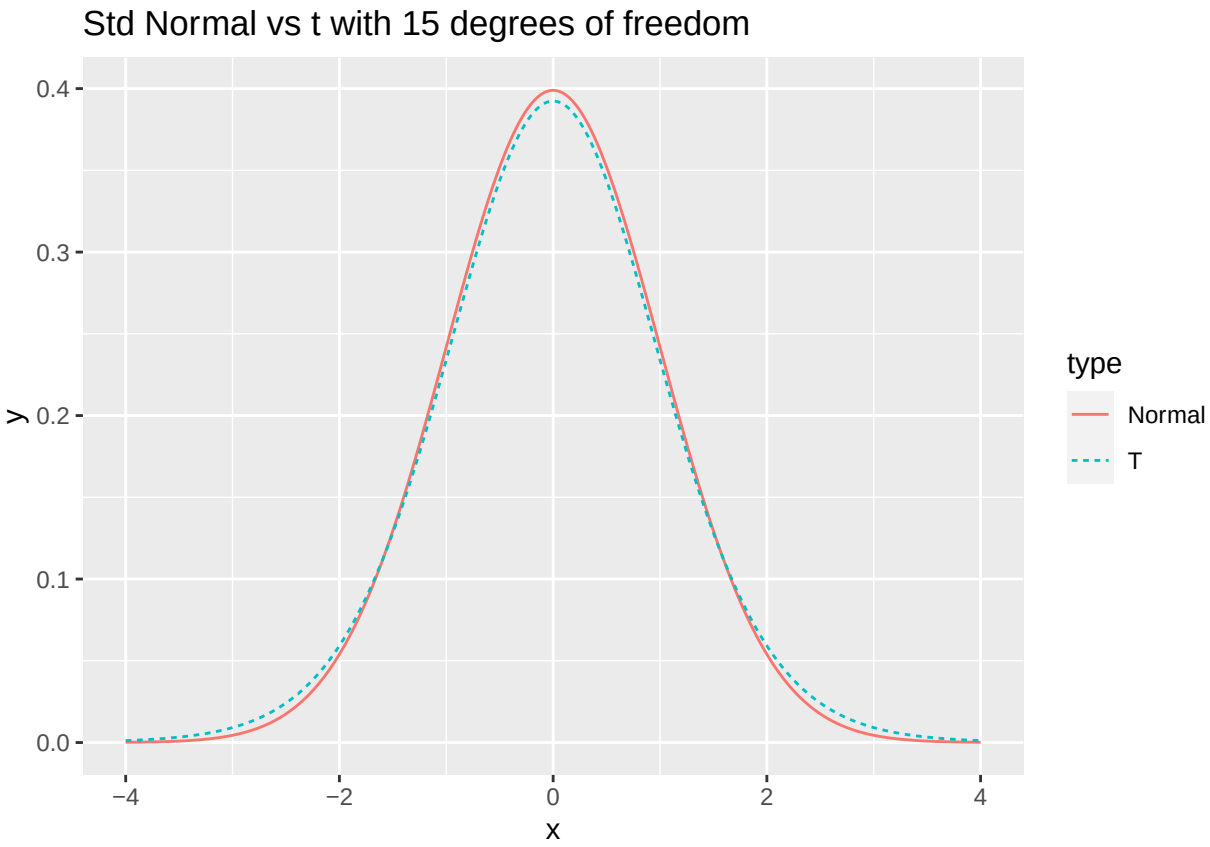

Std Normal vs t with 2 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

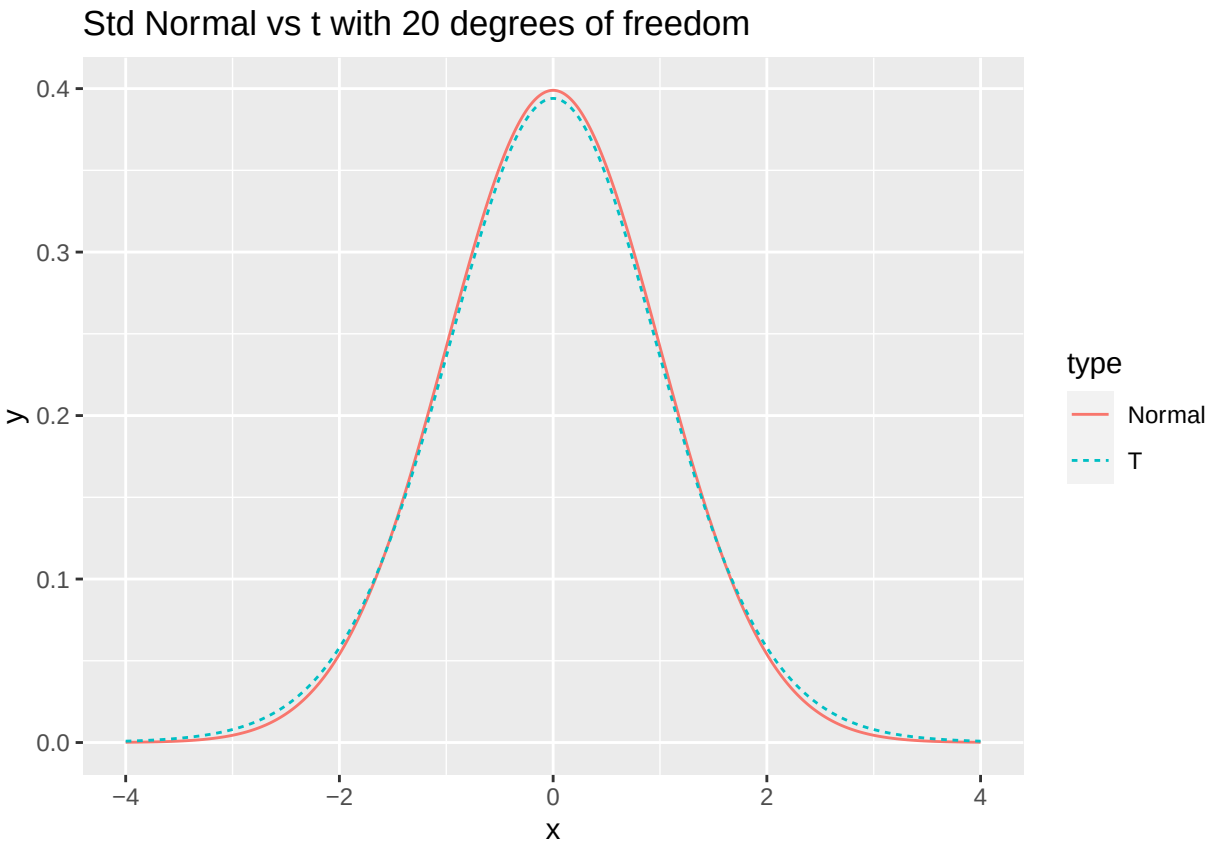

Std Normal vs t with 3 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

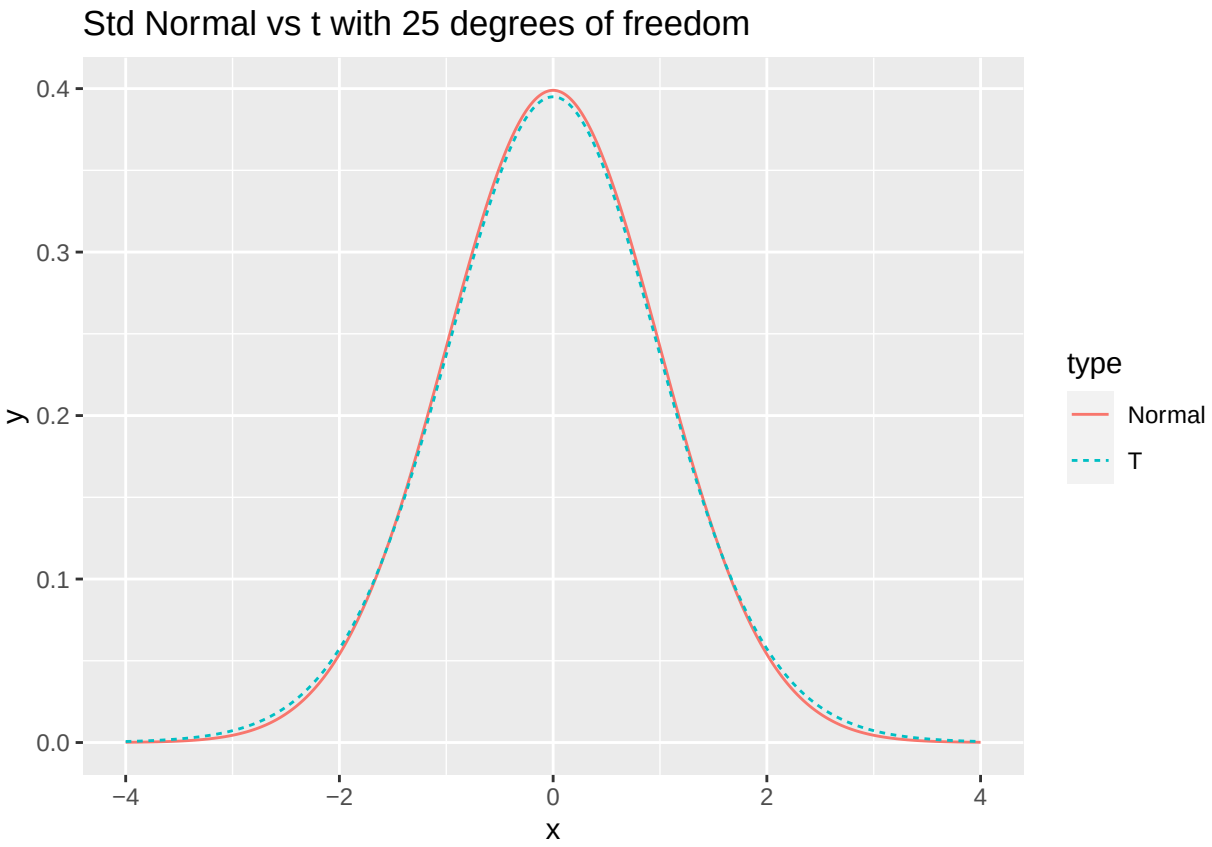

Std Normal vs t with 4 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T

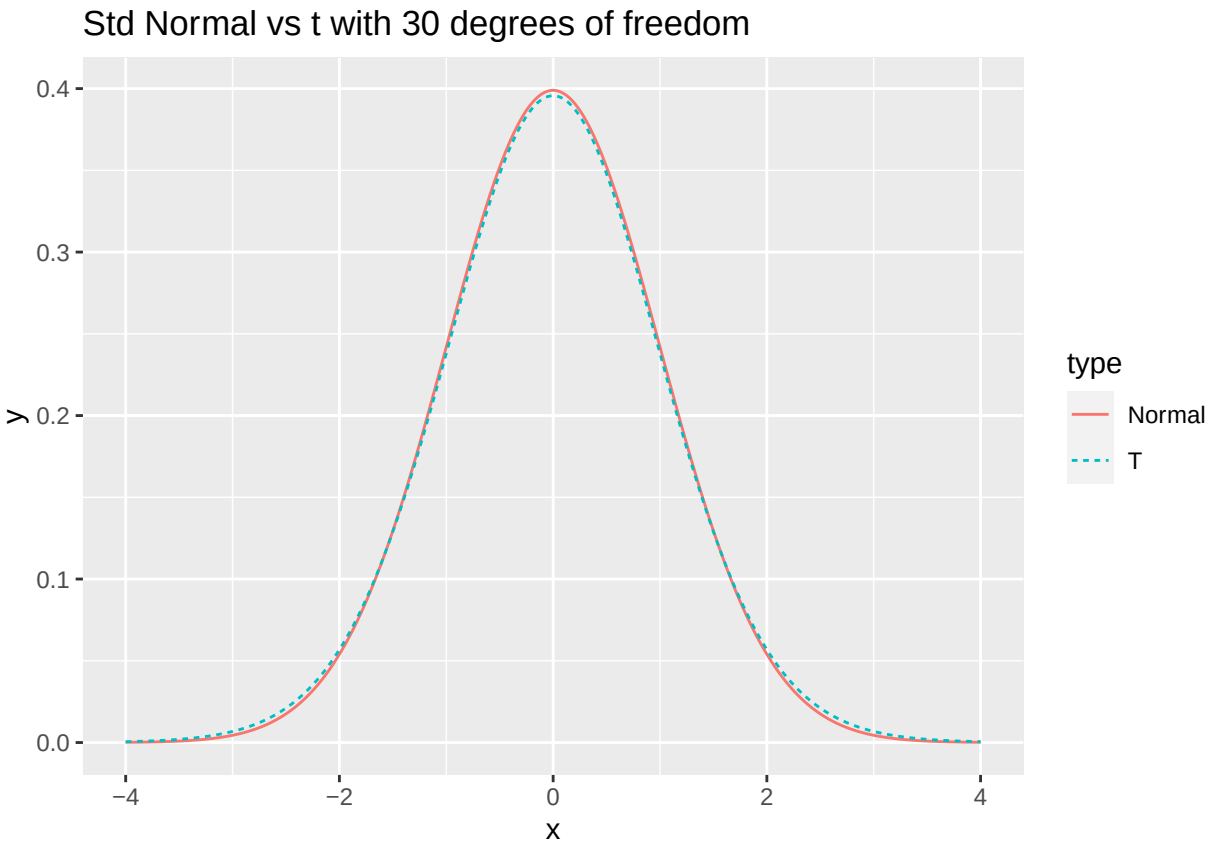

Std Normal vs t with 5 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T


Std Normal vs t with 10 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T


Std Normal vs t with 15 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T


Std Normal vs t with 20 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T


Std Normal vs t with 25 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T


Std Normal vs t with 30 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−4
−2
0
2
4
x
type
Normal
T