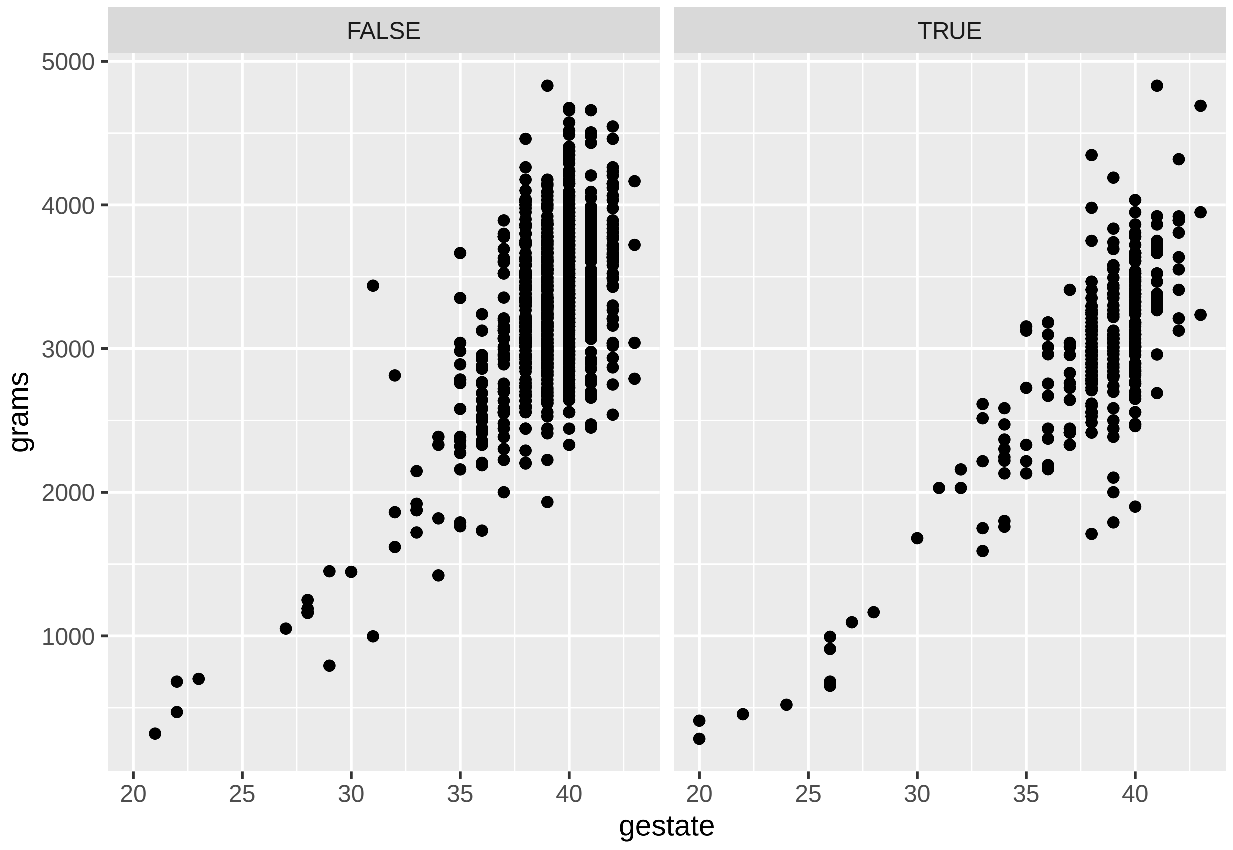
Assignment5

2023-09-21

2. The data set `phbirths` from the `faraway` package contains information on birth weight, gestational length, and smoking status of mother. We'll fit a quadratic model to predict infant birth weight using the gestational time.

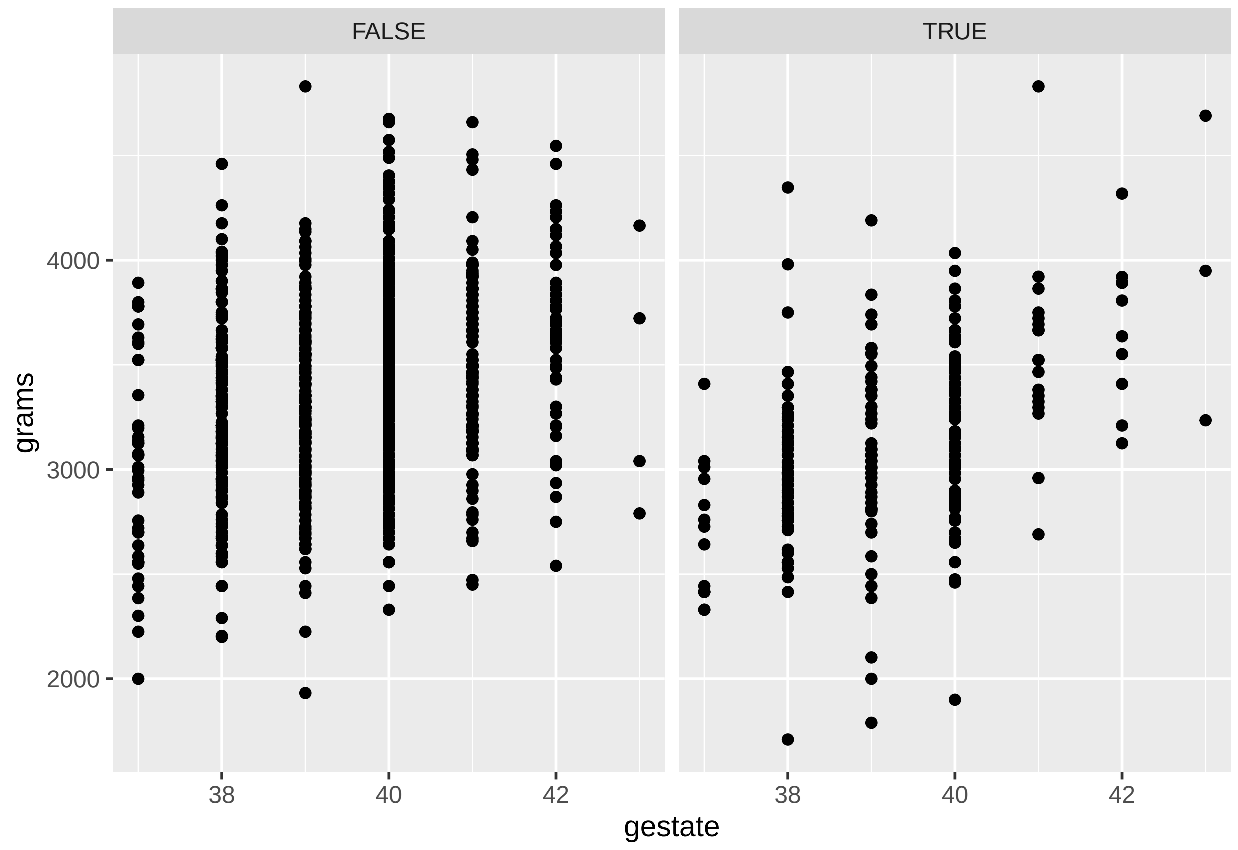
   a) Create two scatter plots of gestational length and birth weight, one for each smoking status.

```
ggplot(phbirths, aes(x=gestate, y=grams)) + geom_point() + # gestate vs grams
  facet_grid(cols = vars(smoke)) # graph for each smoke status
```



```
b)  Remove all the observations that are premature (less than 36 weeks).
For the remainder of the problem, only use these full-term babies.
```

```
phbirths <- phbirths %>%
  filter(gestate > 36) #  mature observations only
ggplot(phbirths, aes(x=gestate, y=grams)) + geom_point() +
  facet_grid(cols = vars(smoke))
```



c) Fit the quadratic model

```
model <- lm(grams ~ poly(gestate,2) * smoke, data=phbirths)
#summary(model)
```

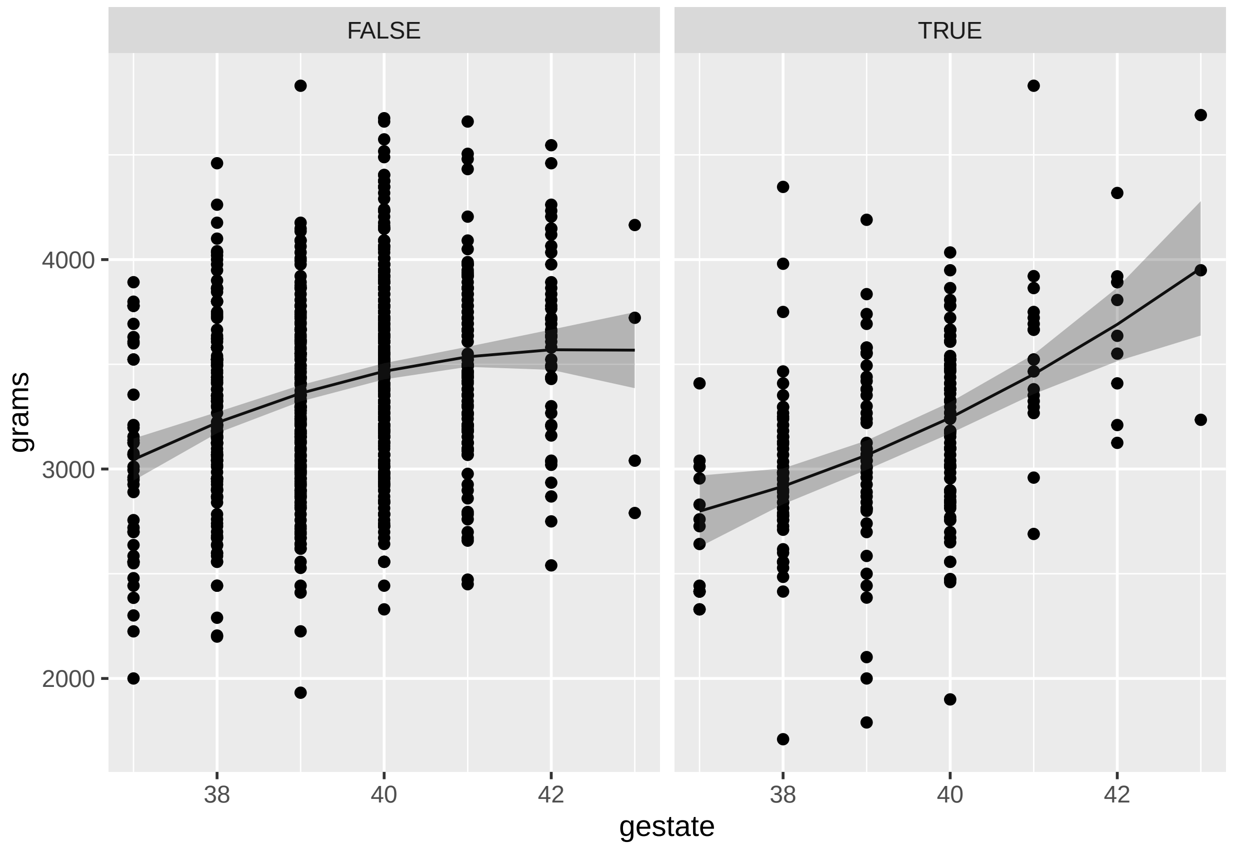
d) Add the model fitted values to the ‘phbirths‘ data frame along with the regression model confidence intervals.

```
# remove any previous model prediction values that I've added,
# and then add the model predictions
phbirths <- phbirths %>%
  dplyr::select( -matches('fit'), -matches('lwr'), -matches('upr') ) %>%
  cbind(predict(model, interval='confidence'))
head(phbirths)
```

```
##   black educ smoke gestate grams      fit      lwr      upr
## 1 FALSE    0  TRUE      40  2898 3244.460 3169.988 3318.932
## 3 FALSE    2 FALSE      38  3977 3221.288 3171.053 3271.524
## 4 FALSE    2  TRUE      37  3040 2798.493 2628.203 2968.783
## 5 FALSE    2 FALSE      38  3523 3221.288 3171.053 3271.524
## 6 FALSE    5  TRUE      40  3100 3244.460 3169.988 3318.932
## 7  TRUE    6 FALSE      40  3670 3466.630 3427.923 3505.337
```

```
e)  On your two scatter plots from part (a), add layers for the model fits
and ribbon of uncertainty for the model fits.
```

```
ggplot(phbirths, aes(x=gestate, y=grams)) + geom_point() +
  facet_grid(cols = vars(smoke)) +
  geom_line( aes(y=fit) ) + # model fit
  geom_ribbon( aes( ymin=lwr, ymax=upr), alpha=.3 ) # ribbon of uncertainty
```



```
f)  Create a column for the residuals in the 'phbirths' data set using any
of the following:
```

```
phbirths <- phbirths %>% mutate( residuals = resid(model) ) # residuals column
head(phbirths)
```

```
##   black educ smoke gestate grams      fit      lwr      upr residuals
## 1 FALSE    0  TRUE      40  2898 3244.460 3169.988 3318.932 -346.4599
## 3 FALSE    2 FALSE      38  3977 3221.288 3171.053 3271.524  755.7117
## 4 FALSE    2  TRUE      37  3040 2798.493 2628.203 2968.783  241.5072
## 5 FALSE    2 FALSE      38  3523 3221.288 3171.053 3271.524  301.7117
## 6 FALSE    5  TRUE      40  3100 3244.460 3169.988 3318.932 -144.4599
## 7  TRUE    6 FALSE      40  3670 3466.630 3427.923 3505.337  203.3700
```

2. The $Uniform\,(a, b)$ distribution is defined on x $\in [a, b]$ and represents a random variable that takes on any value of between `a` and `b` with equal probability. Technically since there are an infinite number

of values between `a` and `b`, each value has a probability of 0 of being selected and I should say each interval of width d has equal probability. It has the density function

$$f(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

The R function `dunif()` evaluates this density function for the above defined values of x, a, and b. Somewhere in that function, there is a chunk of code that evaluates the density for arbitrary values of x. Run this code a few times and notice sometimes the result is 0 and sometimes it is $1/(10-4) = 0.16666667$.

```
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] 0.1666667
```

```
We will write a sequence of statements that utilizes if statements to
appropriately calculate the density of ‘x‘, assuming that ‘a‘, ‘b‘ , and
‘x‘ are given to you, but your code won’t know if ‘x‘ is between ‘a‘ and
‘b‘. That is, your code needs to figure out if it is and give either
‘1/(b-a)‘ or ‘0‘.

a. We could write a set of ‘if else‘ statements.
```

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

if( x < a ){
  result <- 0 # value = 0 when less than 'a'
}else if( x <= b ){
  result <- 1/(b-a) # value = 1/(b-a) when greater than a & less than or equal
}else{                # to b
  result <- 0. # value = 0 when greater than 'a' & 'b'
}
print(paste('x=',round(x,digits=3), '  result=', round(result,digits=3)))
```

```
    Replace the ‘????‘ with the appropriate value, either 0 or
    $1/\left(b-a\right)$. Run the code repeatedly until you are certain
    that it is calculating the correct density value.

b.  We could perform the logical comparison all in one comparison. Recall
    that we can use ‘&‘ to mean "and" and ‘|‘ to mean "or". In the
    following two code chunks, replace the ‘???‘ with either ‘&‘ or ‘|‘ to
    make the appropriate result.
```

i.

```r
x <- runif(n=1, 0,10)  # one random value between 0 and 10
if( (a<=x) & (x<=b) ){ # if a<=x<=b, value = 1/(b-a)
  result <- 1/(b-a)
}else{
  result <- 0 # if not a<=x<=b, value = 0
}
print(paste('x=',round(x,digits=3), '  result=', round(result,digits=3)))
```

ii.

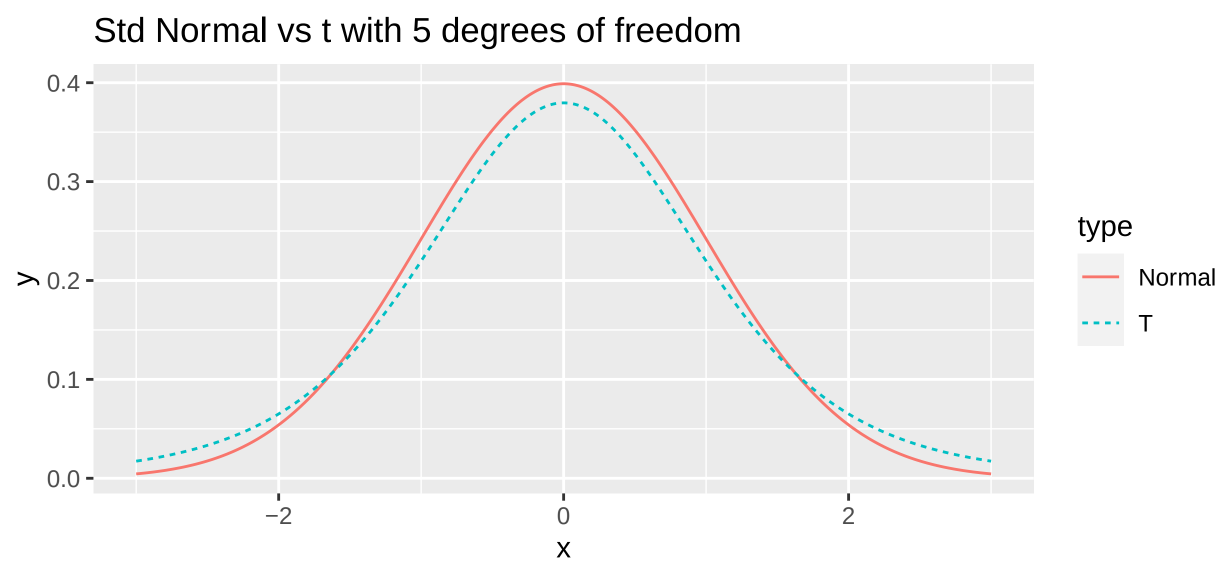
```r
x <- runif(n=1, 0,10)  # one random value between 0 and 10
if( (x<a) | (b<x) ){ # if x = less than a OR greater than b, value = 1/(b-a)
  result <- 0
}else{
  result <- 1/(b-a)
}
print(paste('x=',round(x,digits=3), '  result=', round(result,digits=3)))
```

iii.

```r
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)))
```

3. I often want to repeat some section of code some number of times. For example, I might want to create a bunch plots that compare the density of a t-distribution with specified degrees of freedom to a standard normal distribution.

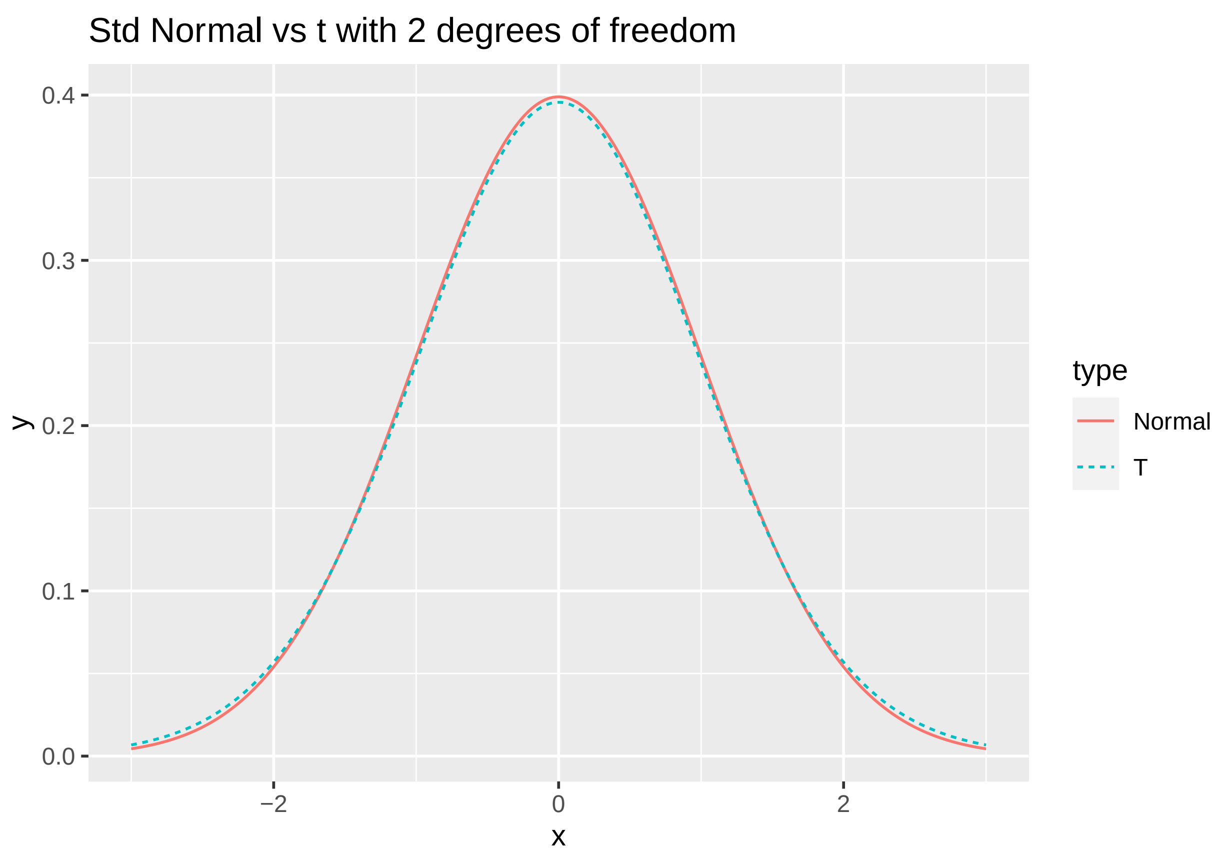
```r
library(ggplot2)
df <- 5
N <- 1000
x.grid <- seq(-3, 3, length=N)
data <- data.frame(
  x = c(x.grid, x.grid),
  y = c(dnorm(x.grid), dt(x.grid, df)),
  type = c( rep('Normal',N), rep('T',N) ) )

# 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)
```



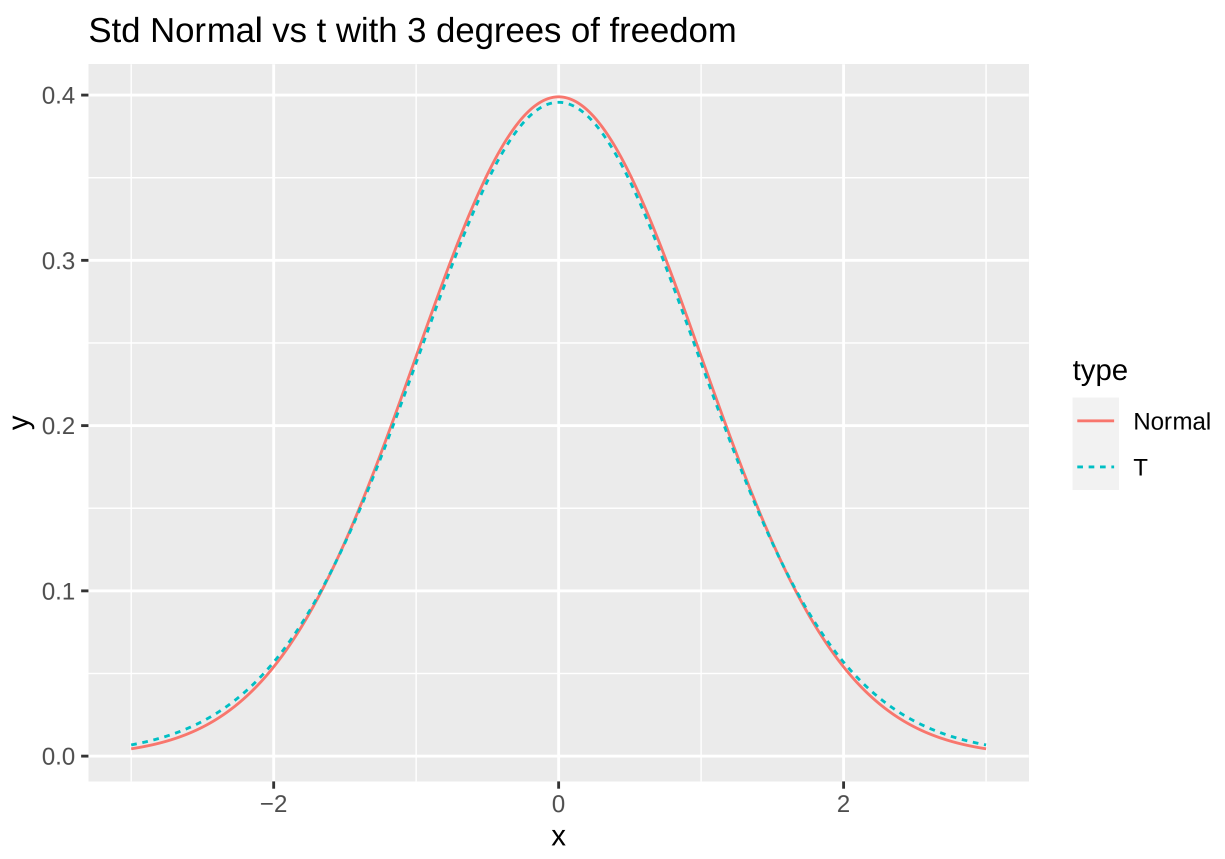
```
a)  Use a ‘for‘ loop to create similar graphs for degrees of freedom
    $2,3,4,\dots,29,30$.
```

```
library(ggplot2)
df <- 30
N <- 1000
result <- 0

result <- result
for( i in 2:df ){ # loop for df 2:30
  x.grid <- seq(-3, 3, length=N)
data <- data.frame(
  x = c(x.grid, x.grid),
  y = c(dnorm(x.grid), dt(x.grid, df)),
  type = c( rep('Normal',N), rep('T',N) ) )
 # plot data
myplot <- ggplot(data, aes(x=x, y=y, color=type, linetype=type)) +
  geom_line() +
  labs(title = paste('Std Normal vs t with', i, 'degrees of freedom'))
  result[i] <- result
print(myplot) # print graphs 2:30
}
```

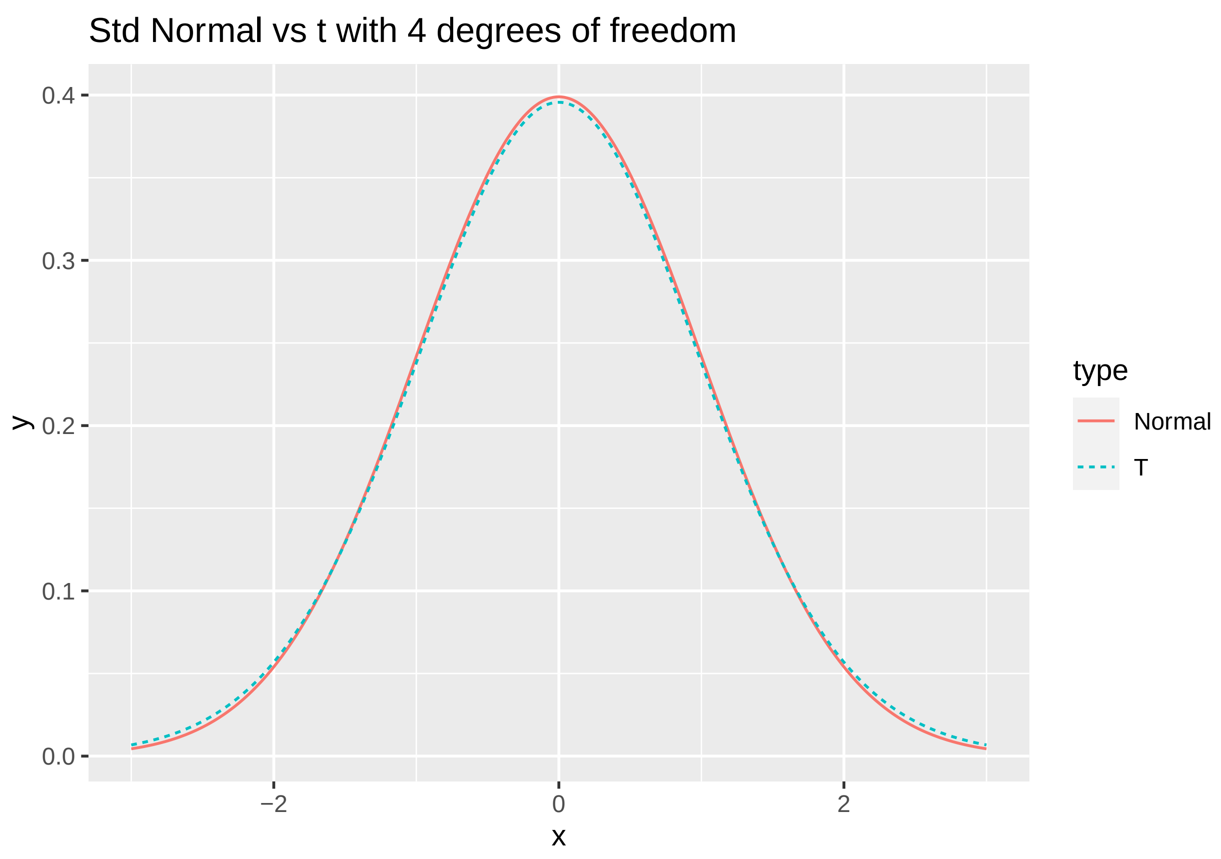
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



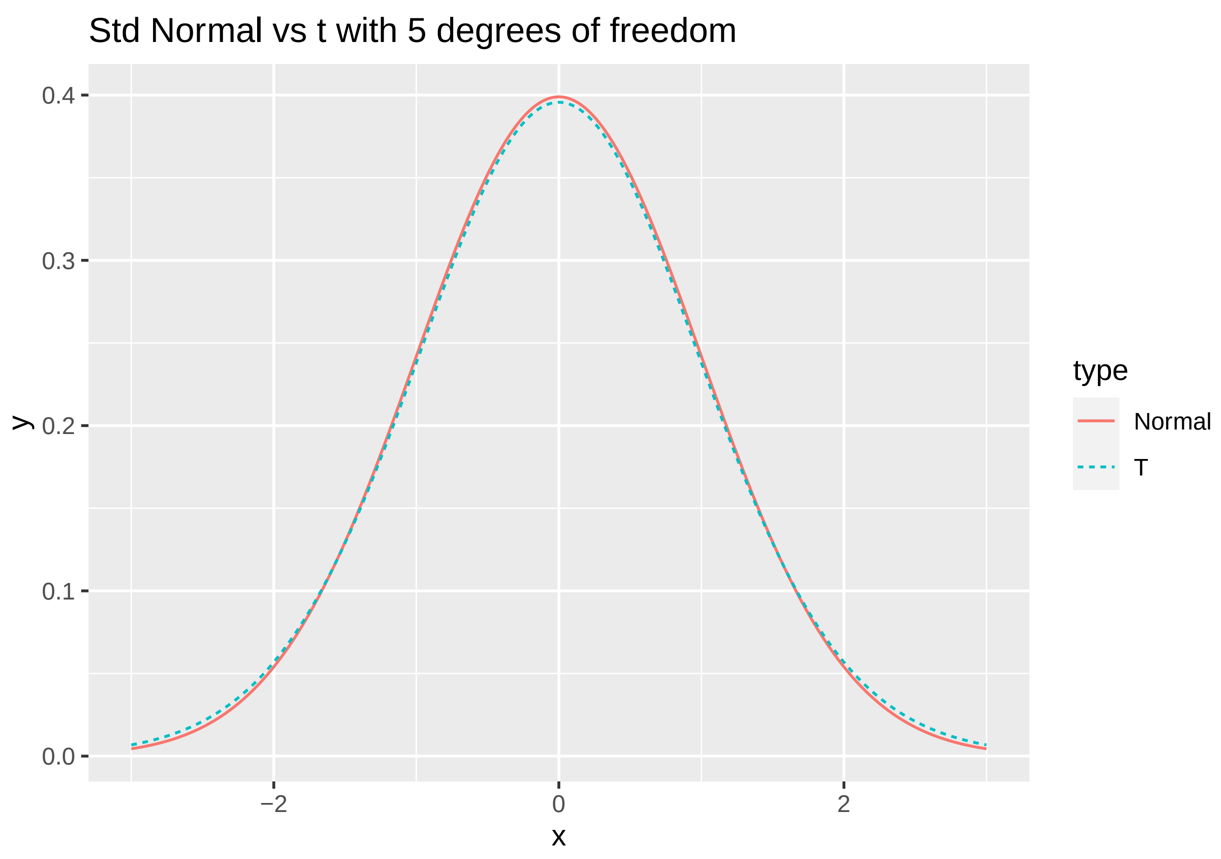
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



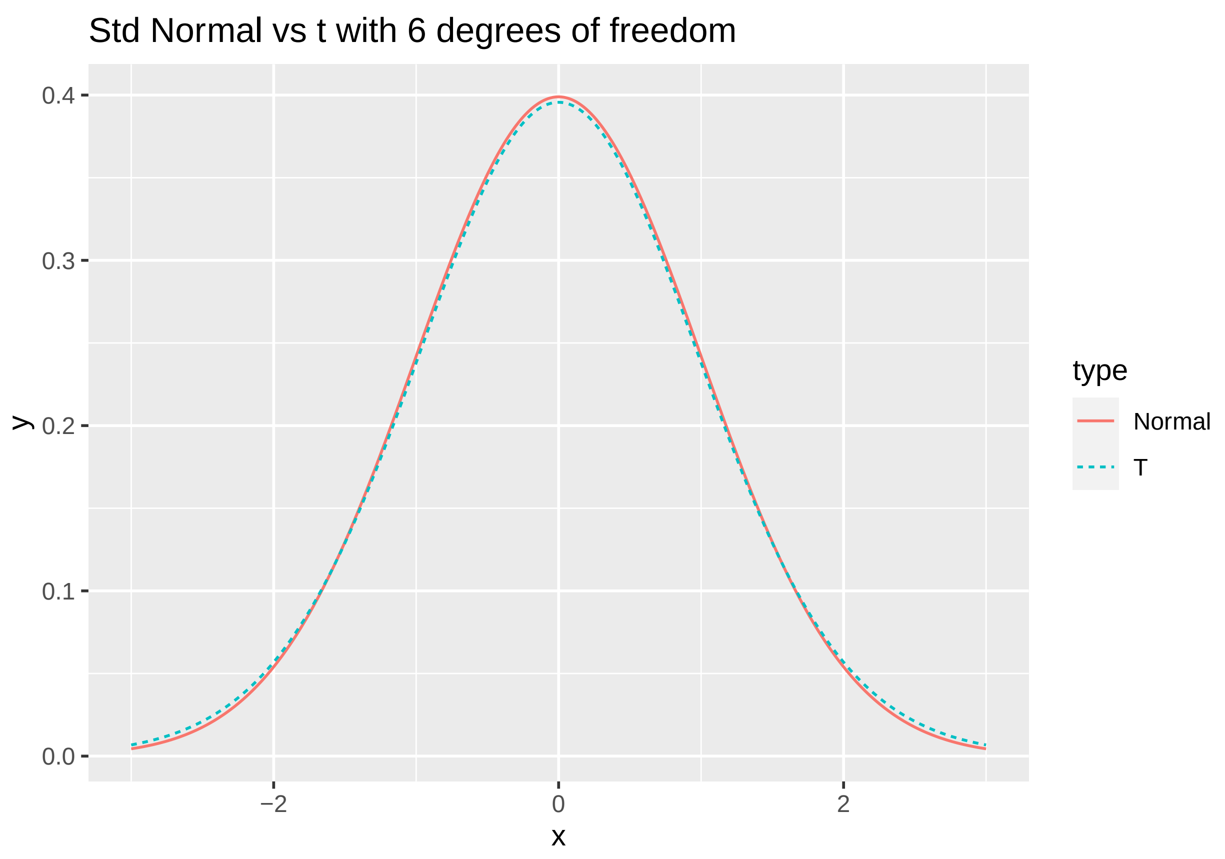
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



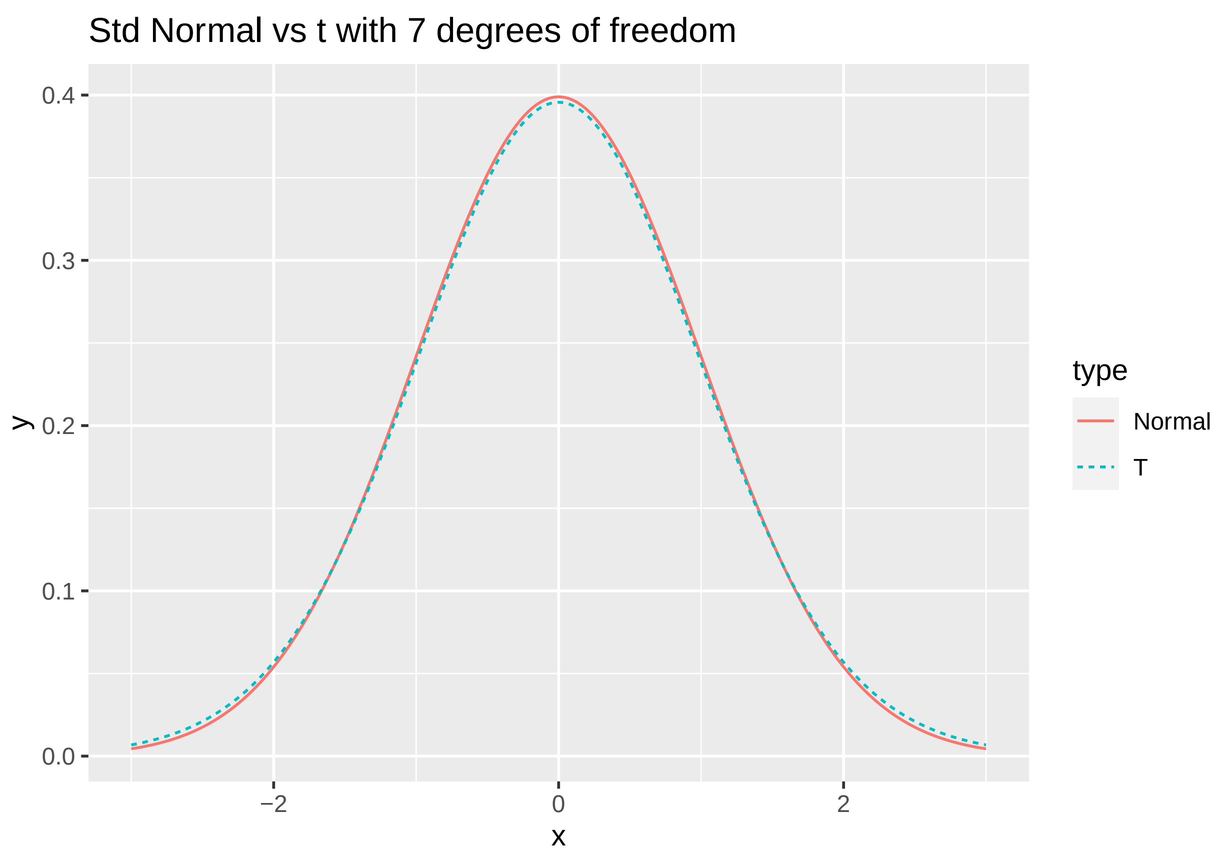
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



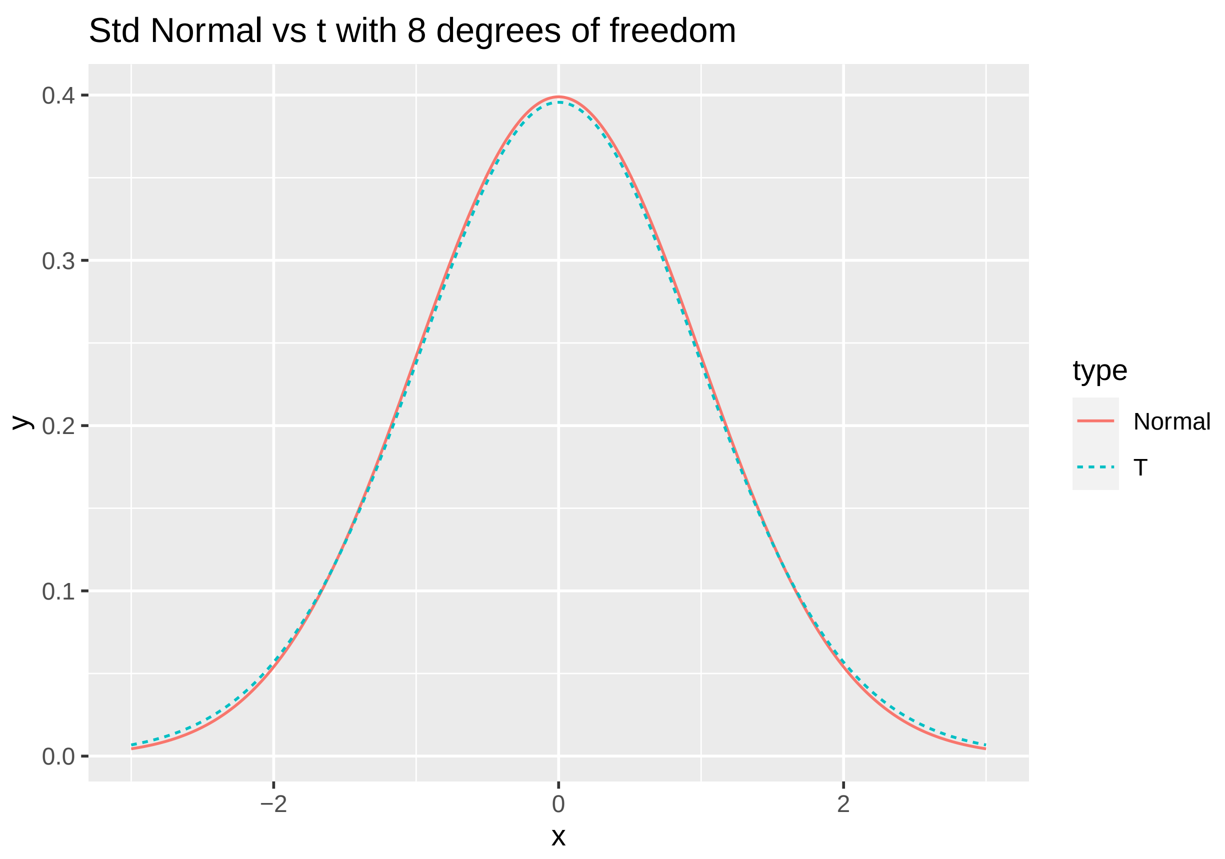
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



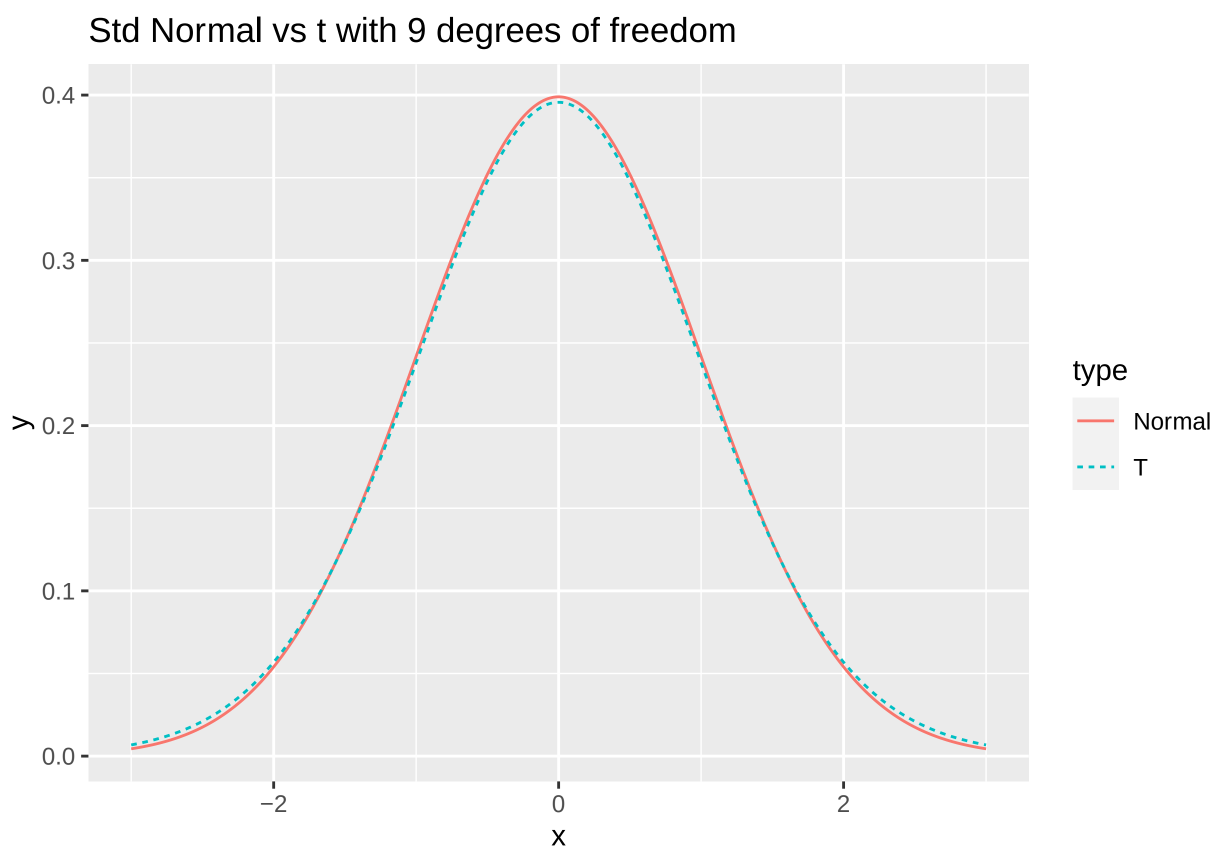
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



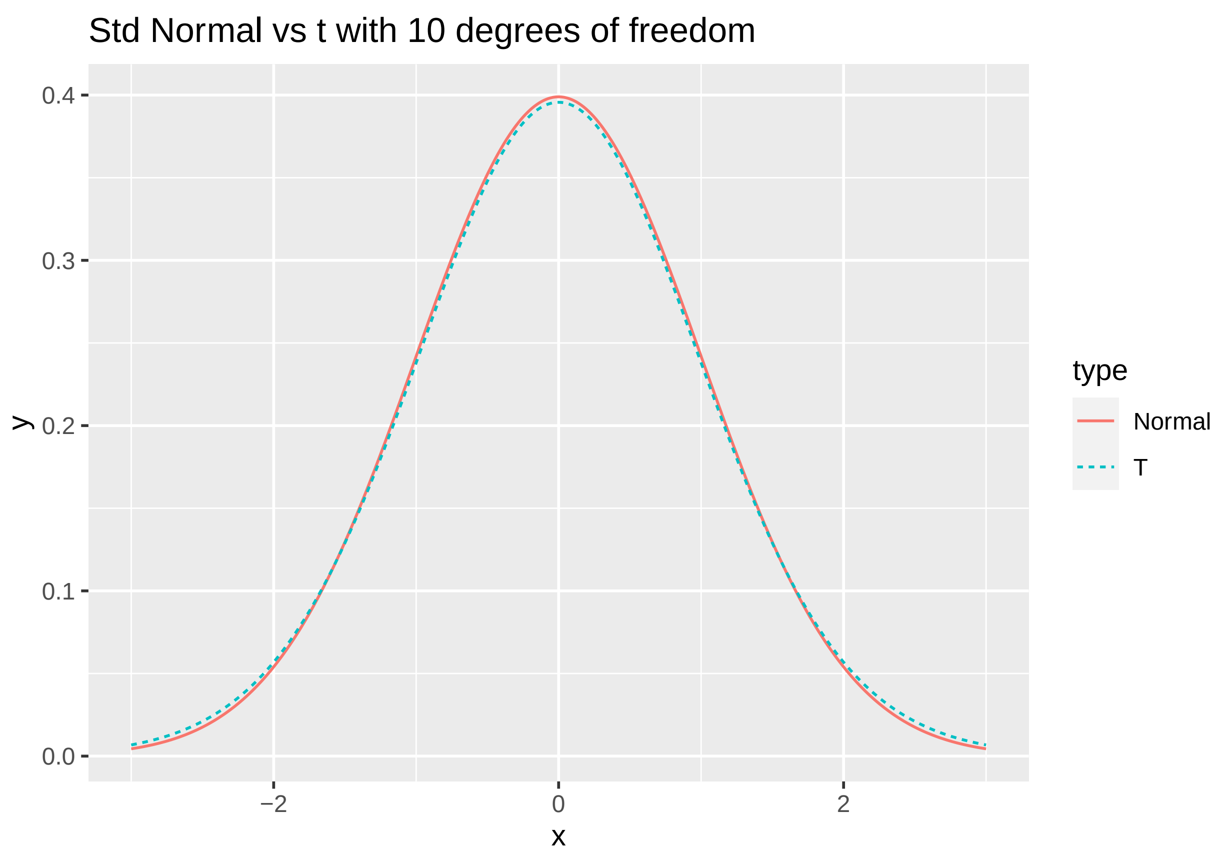
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



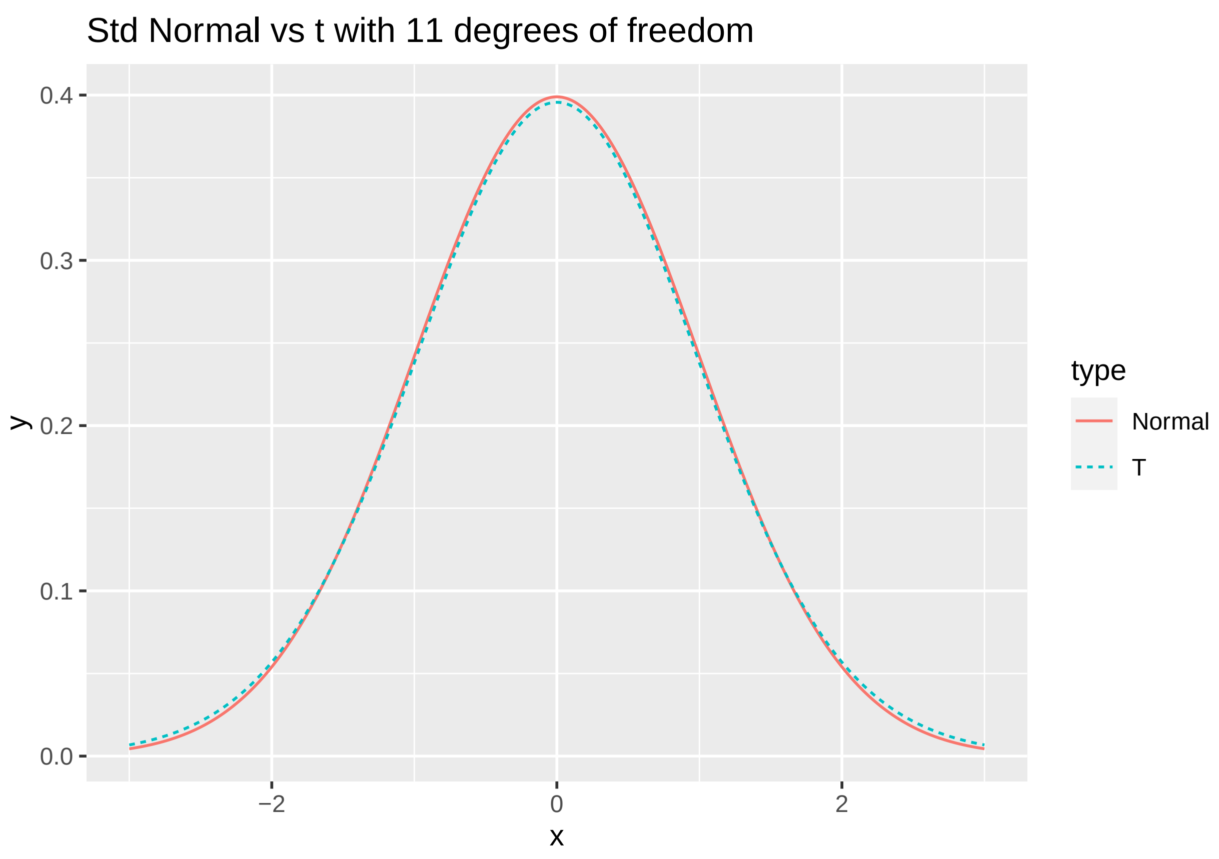
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



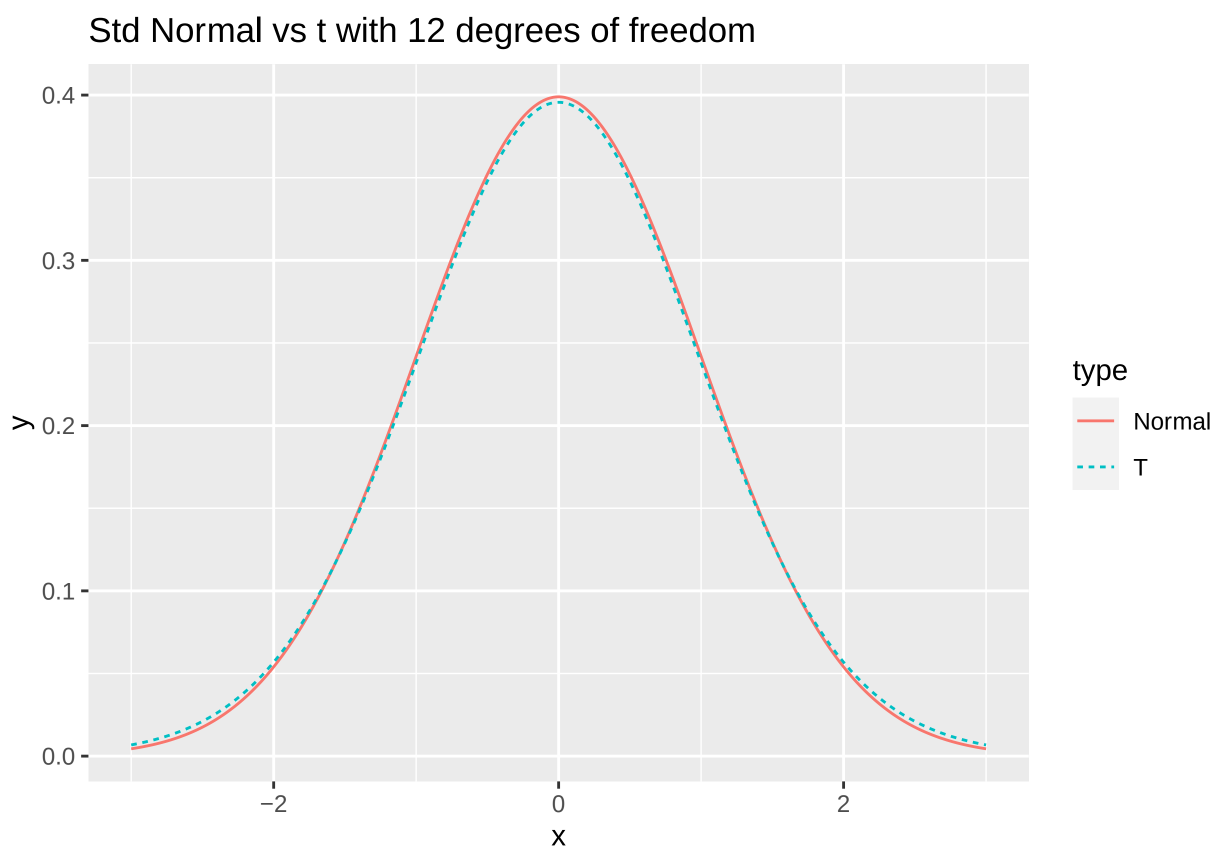
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



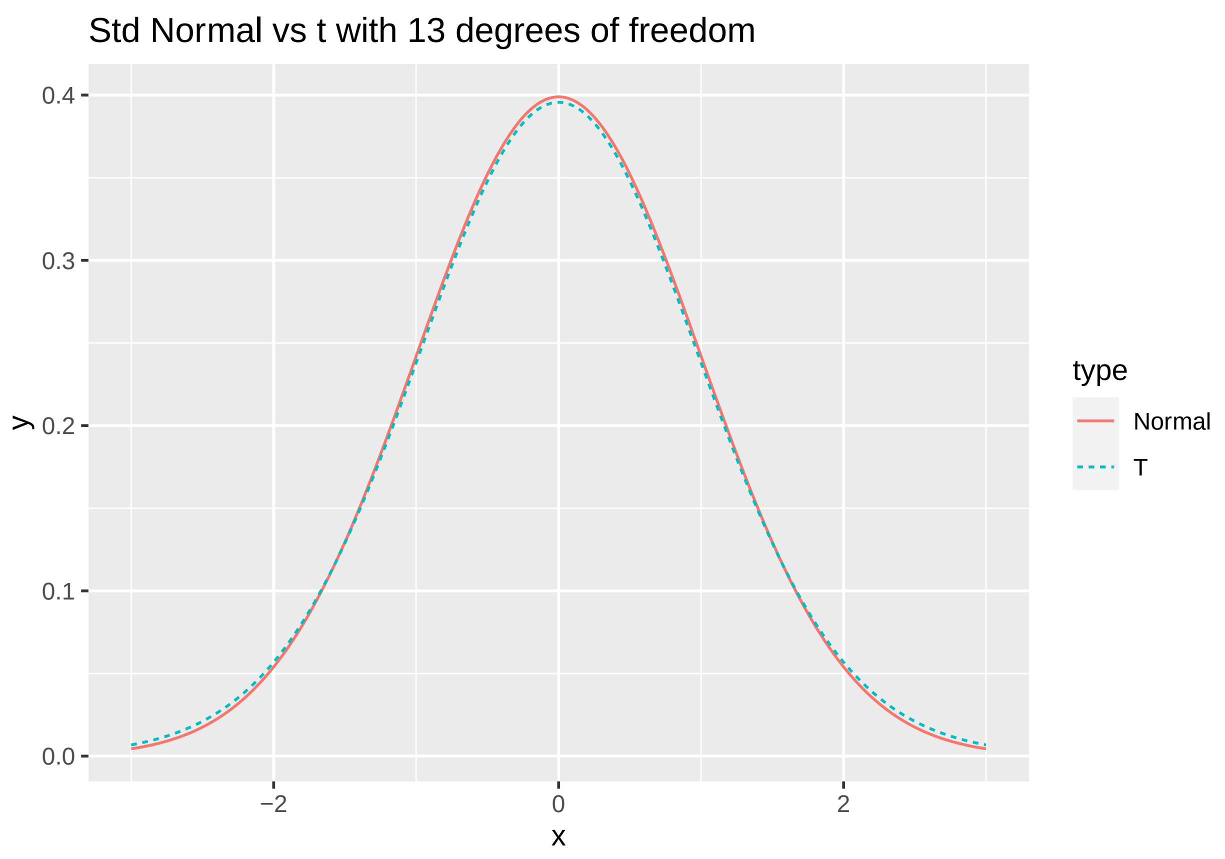
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



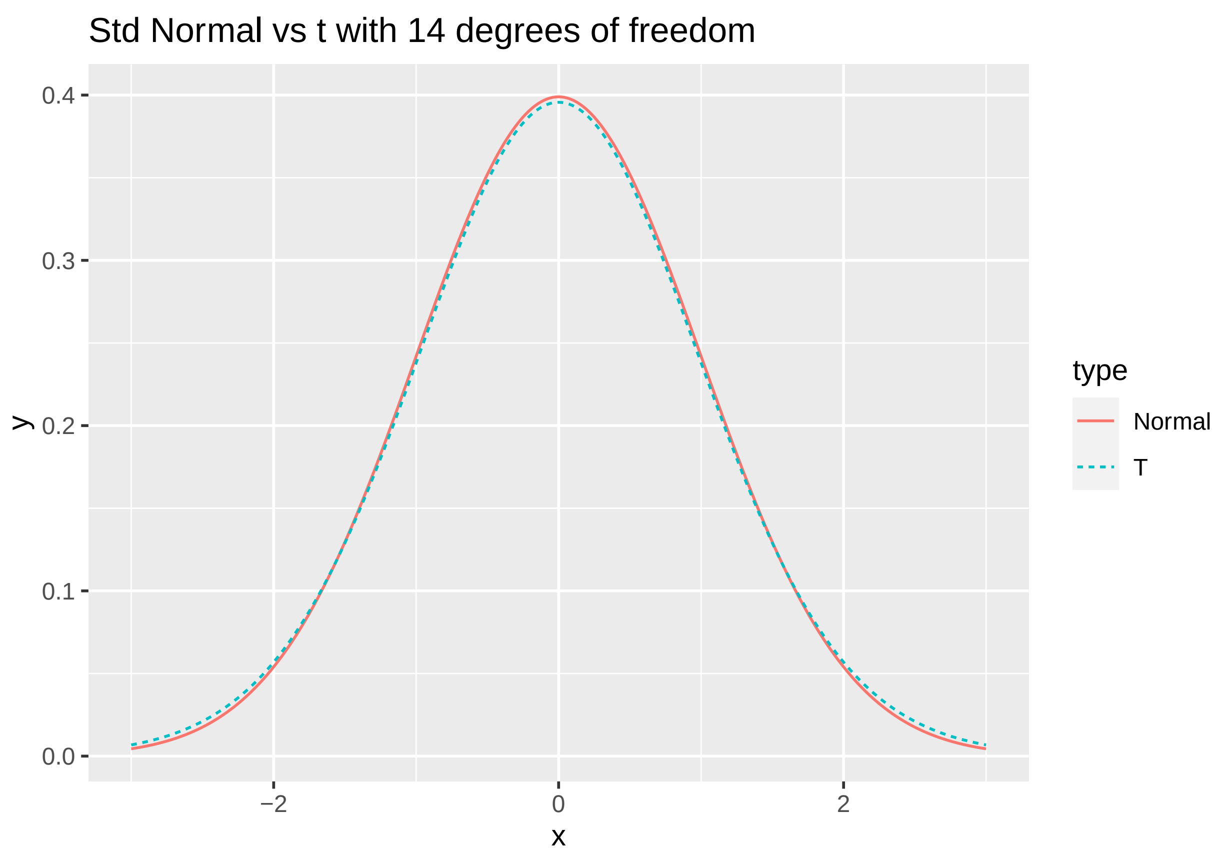
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



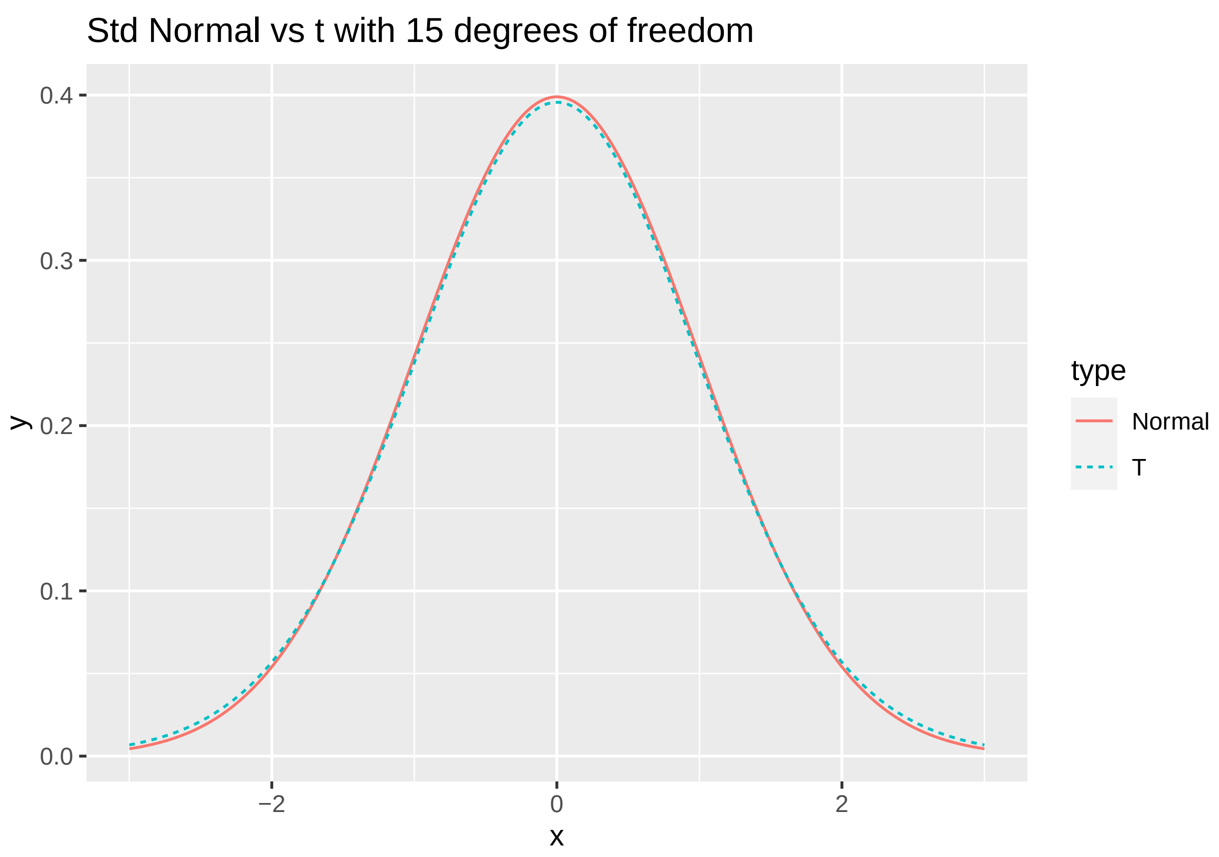
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



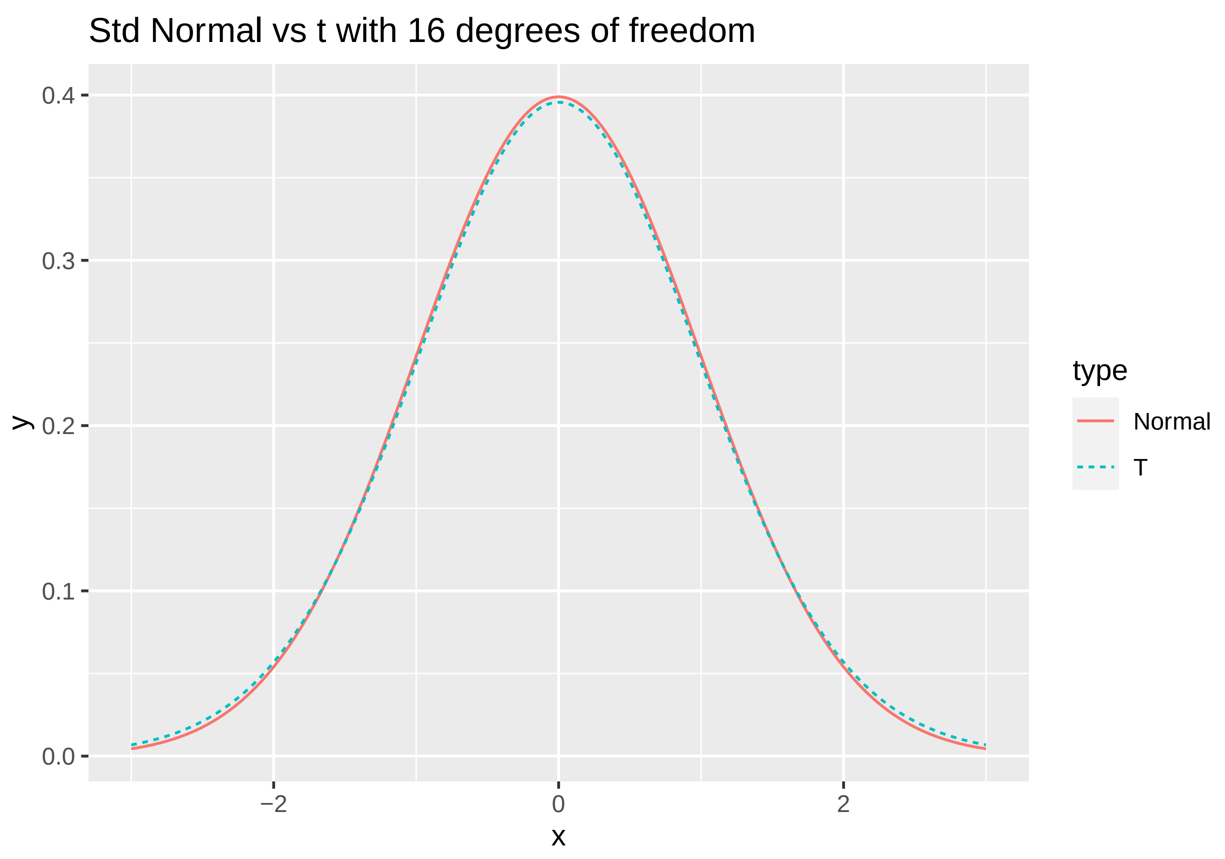
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



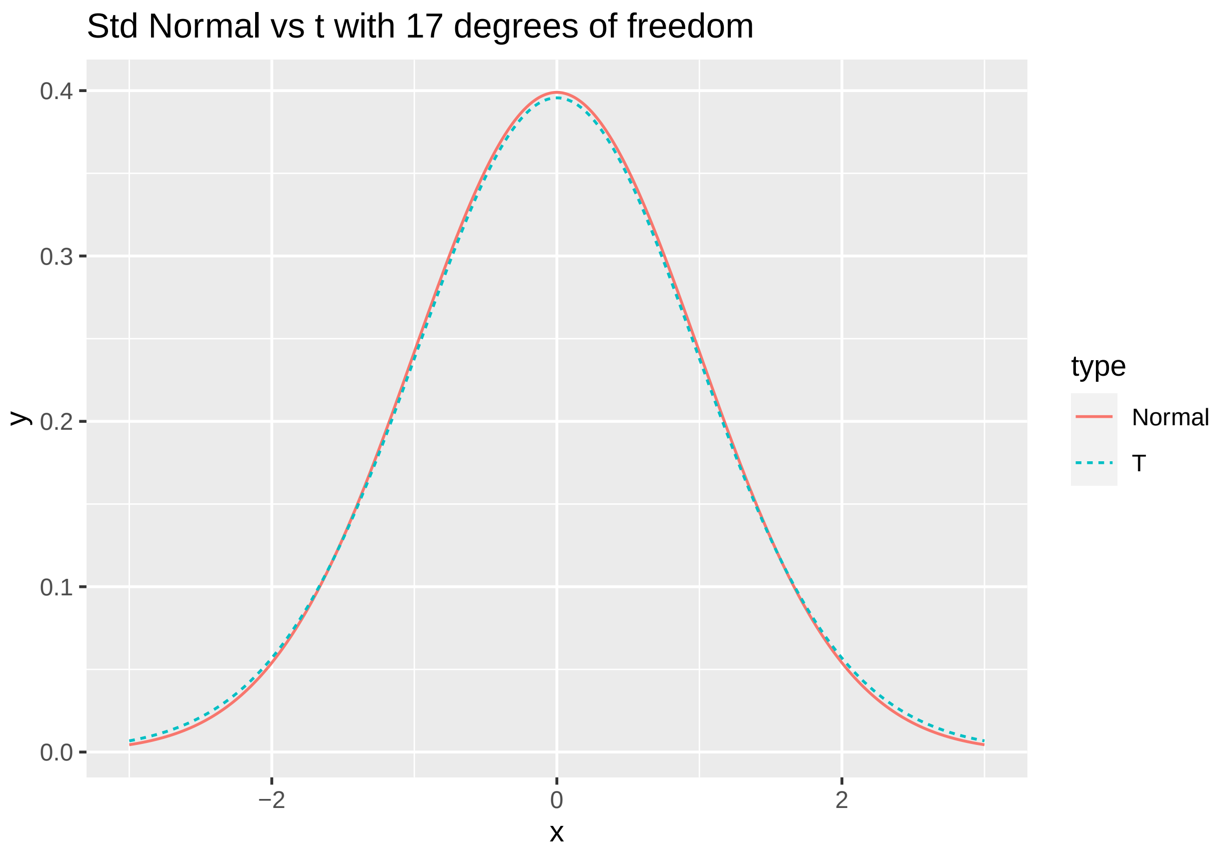
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



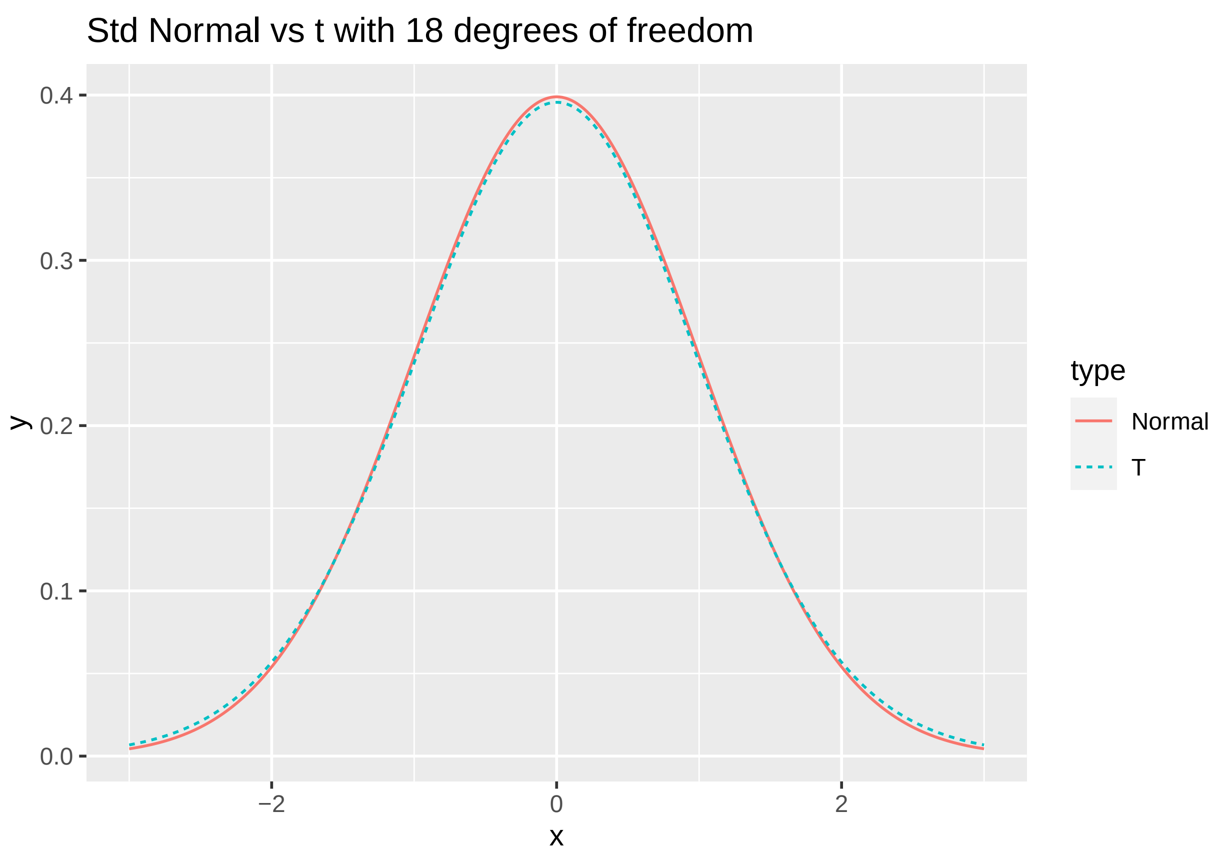
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



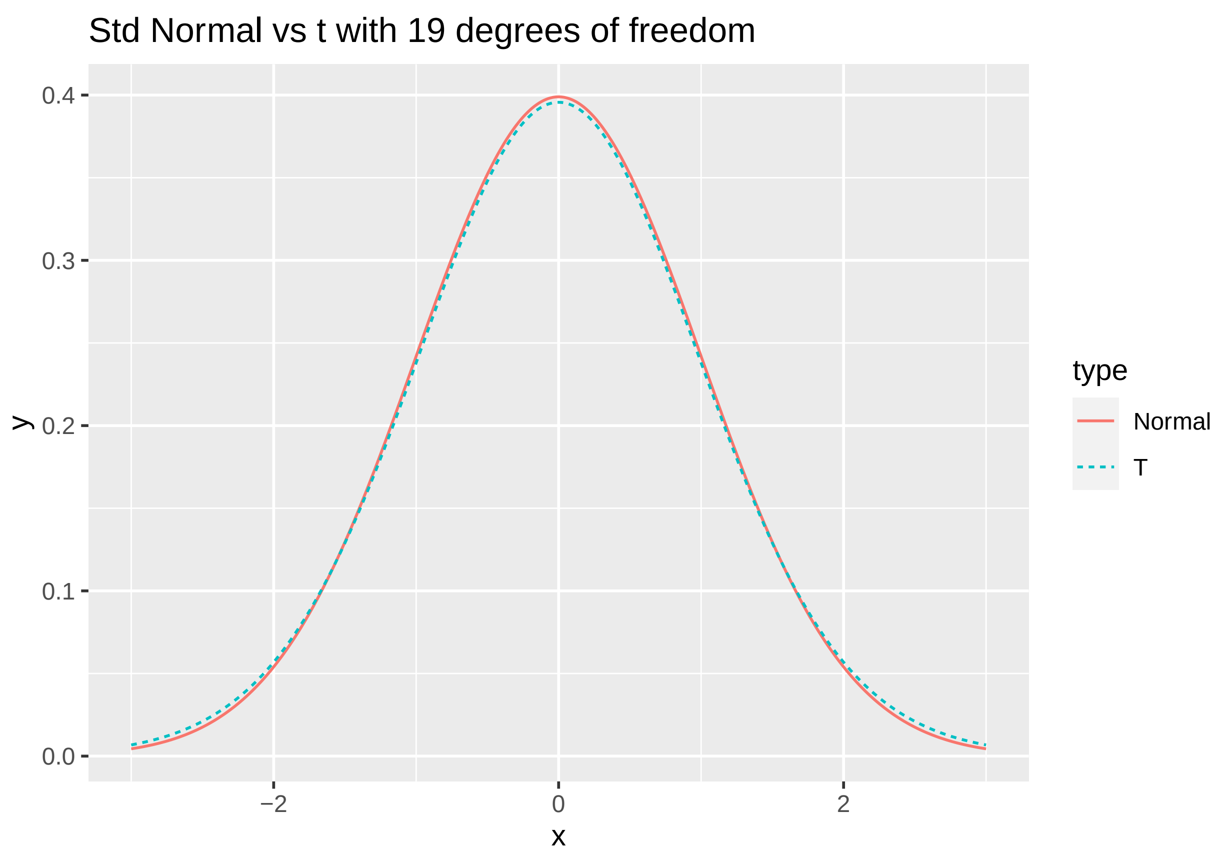
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



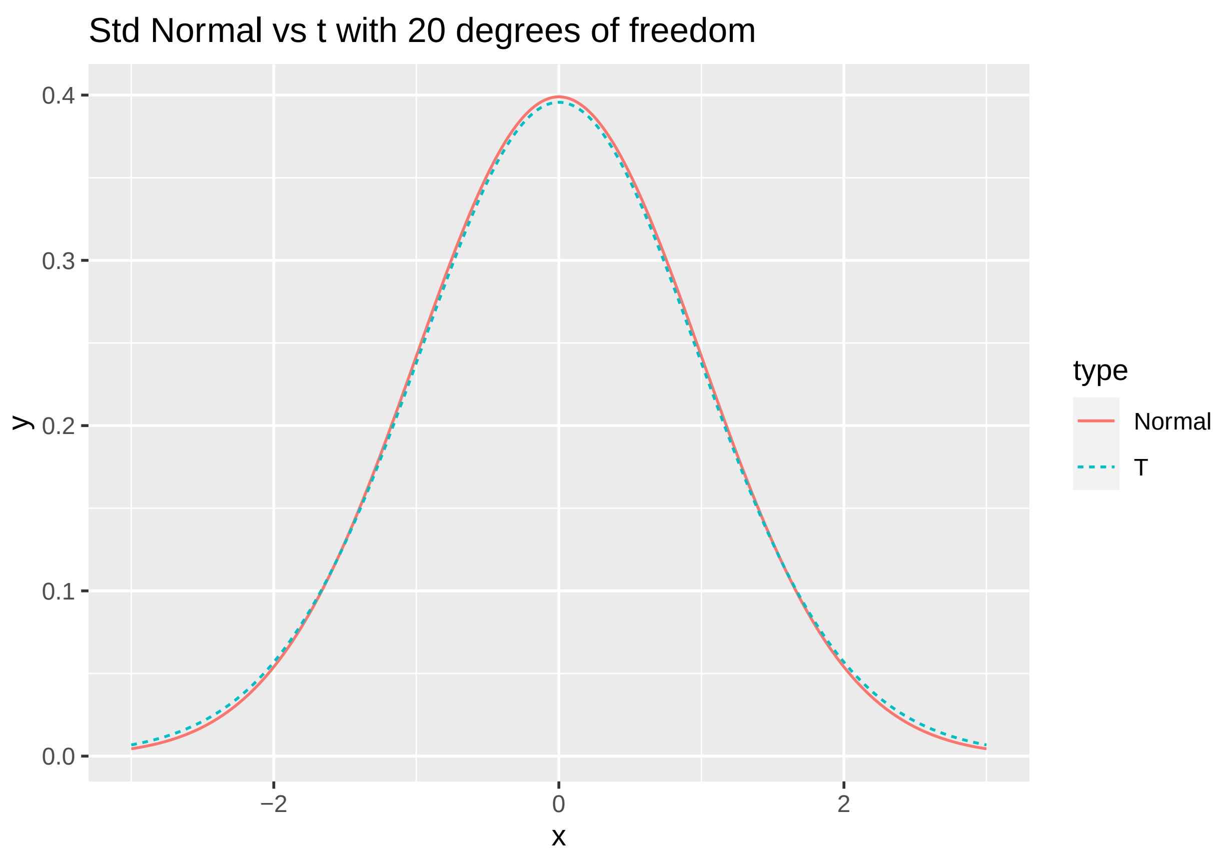
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



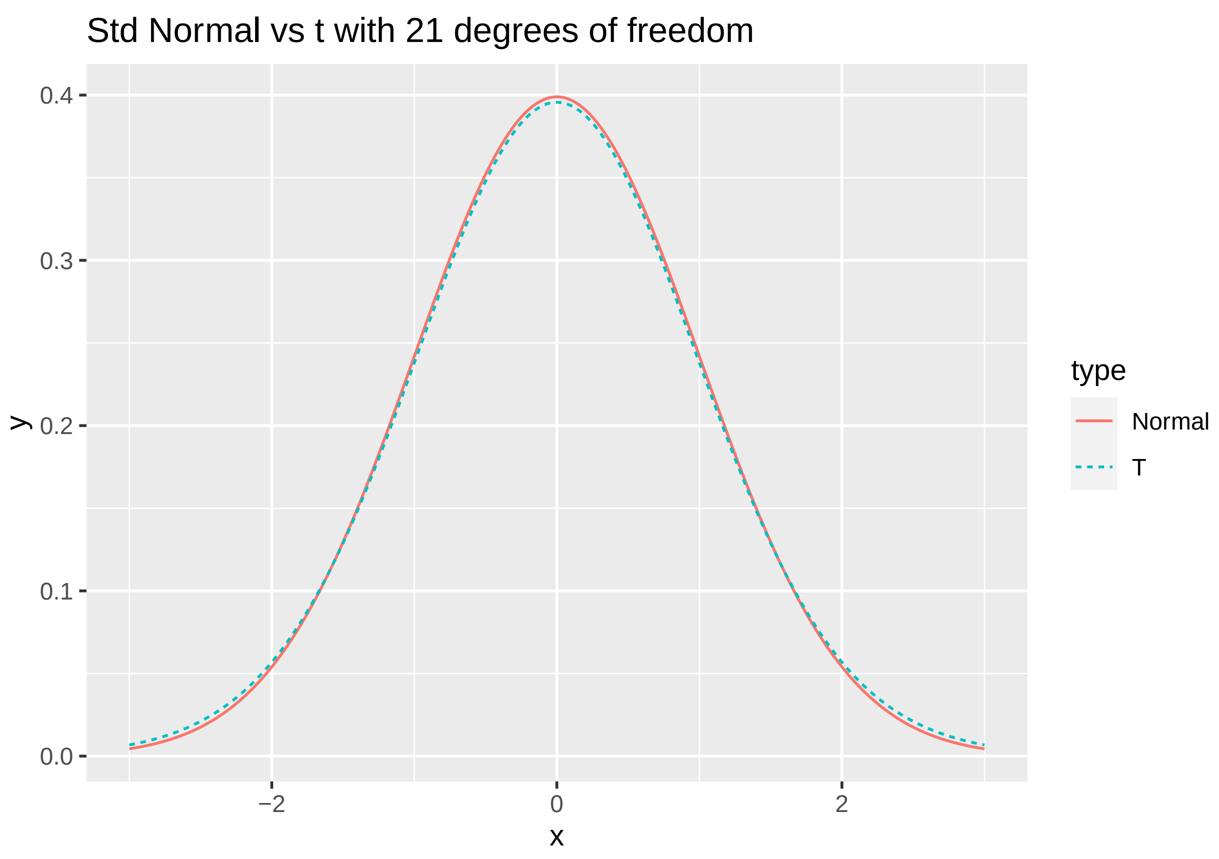
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



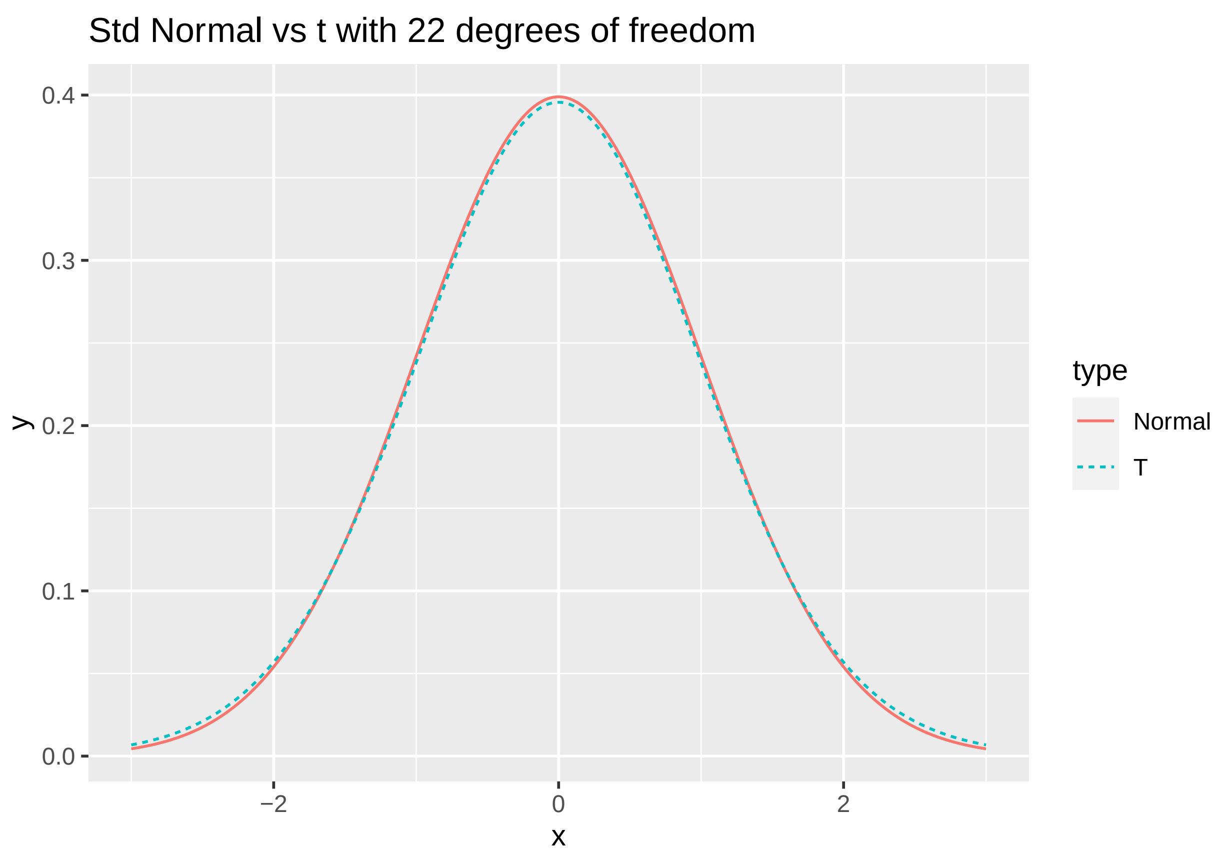
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



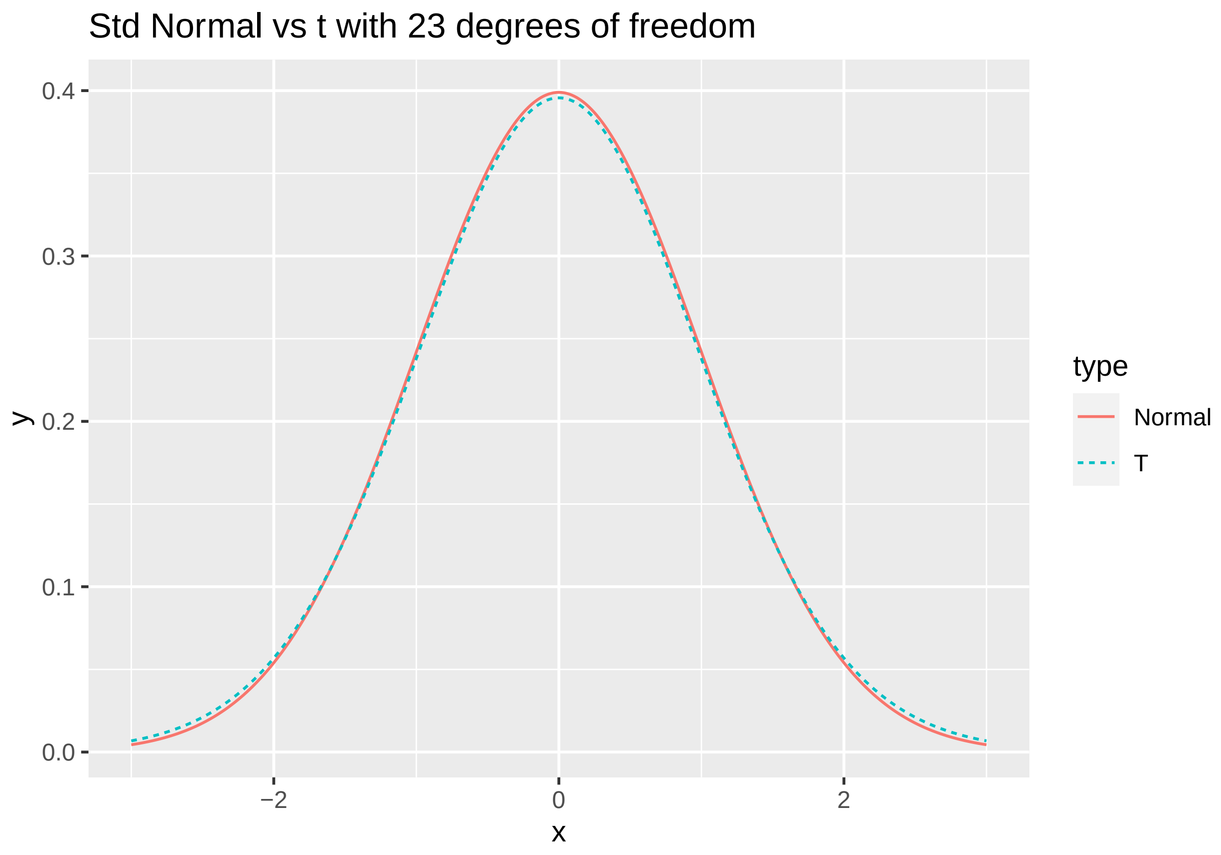
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



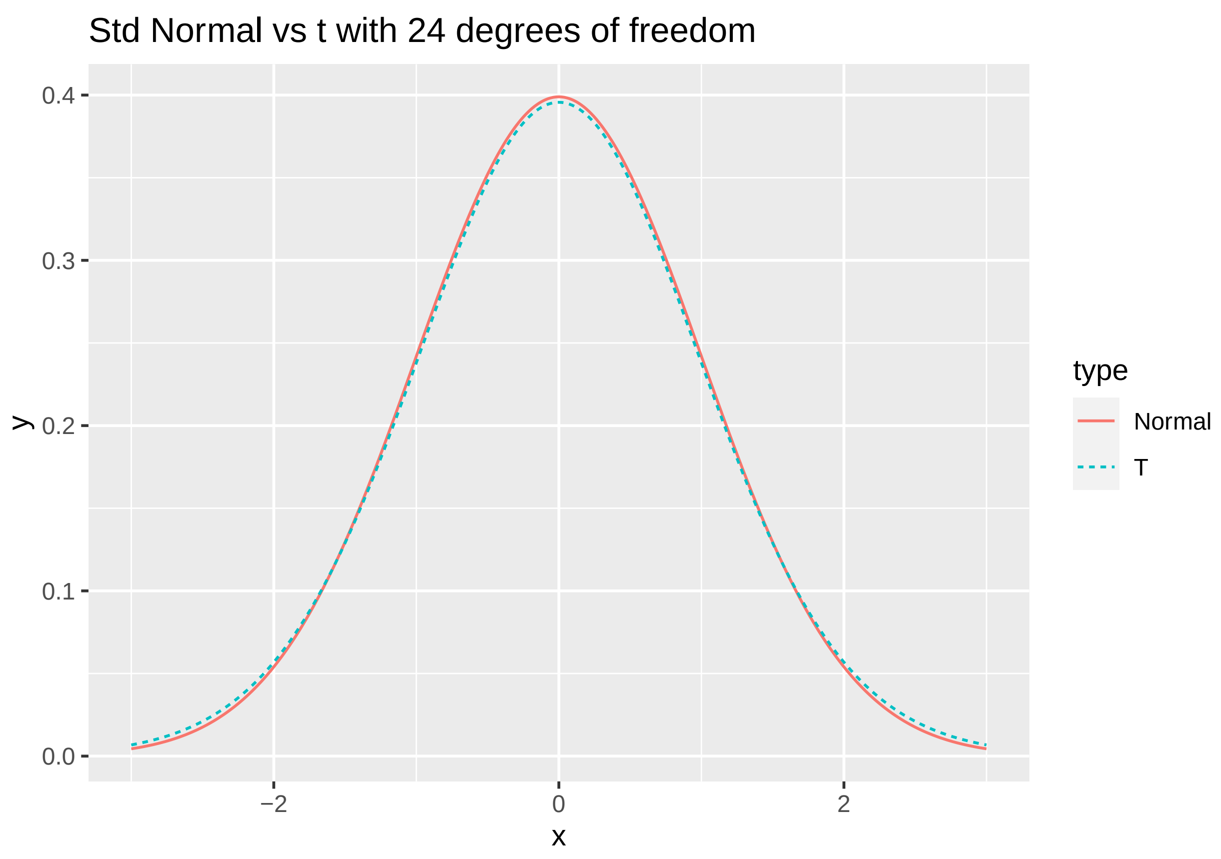
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



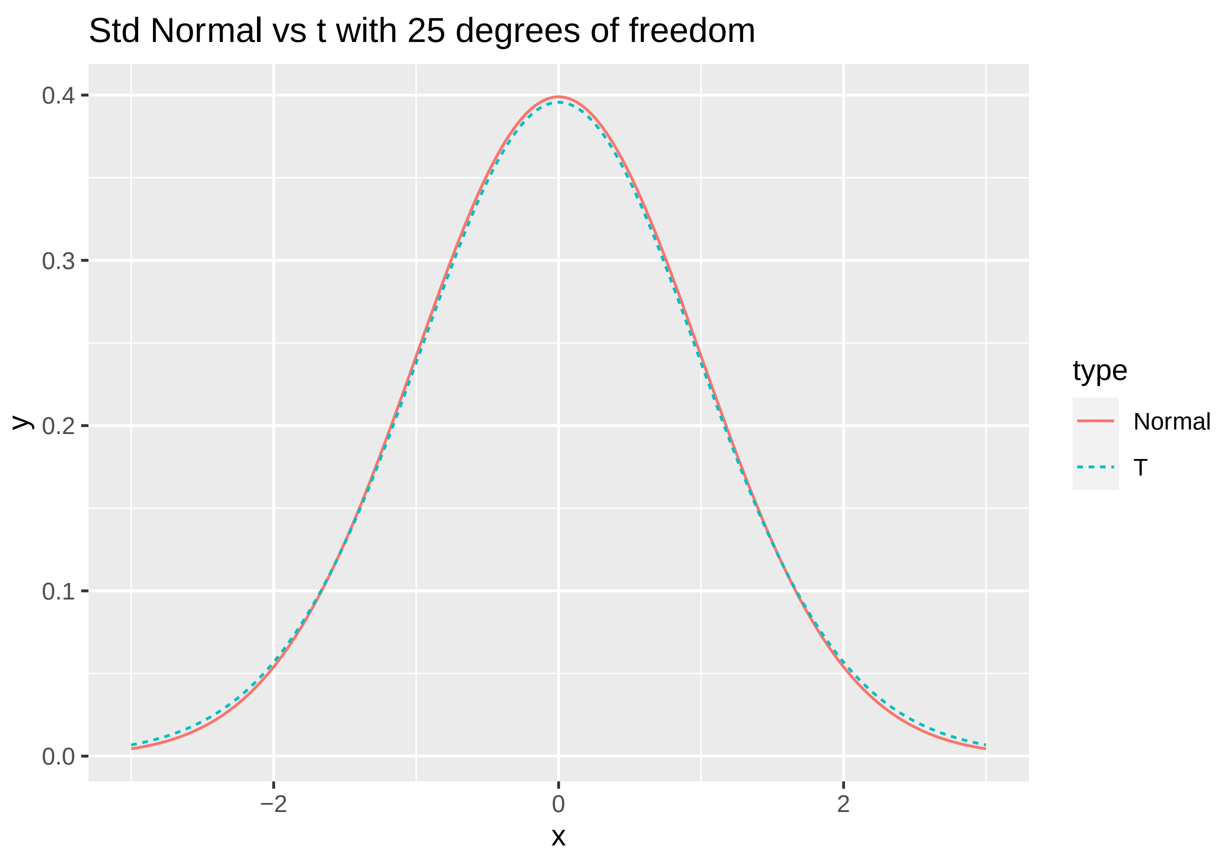
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



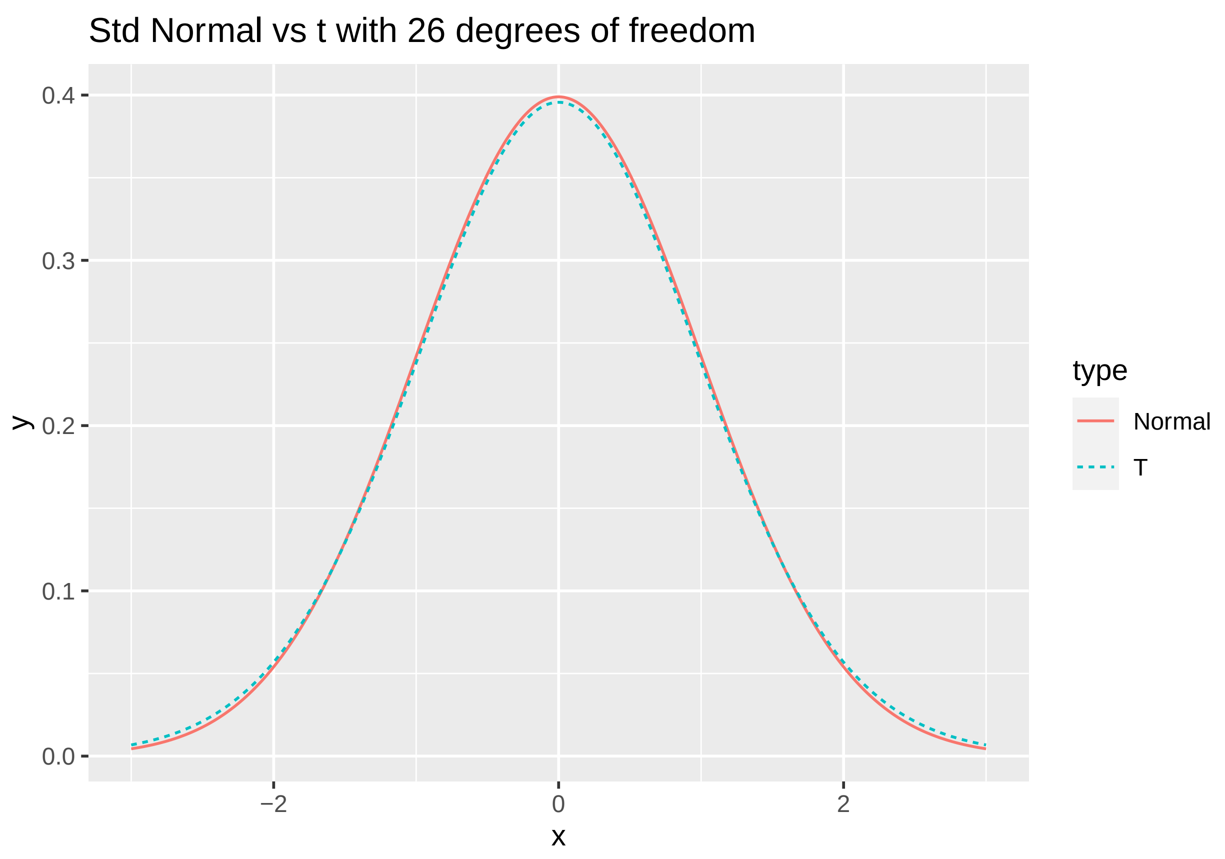
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



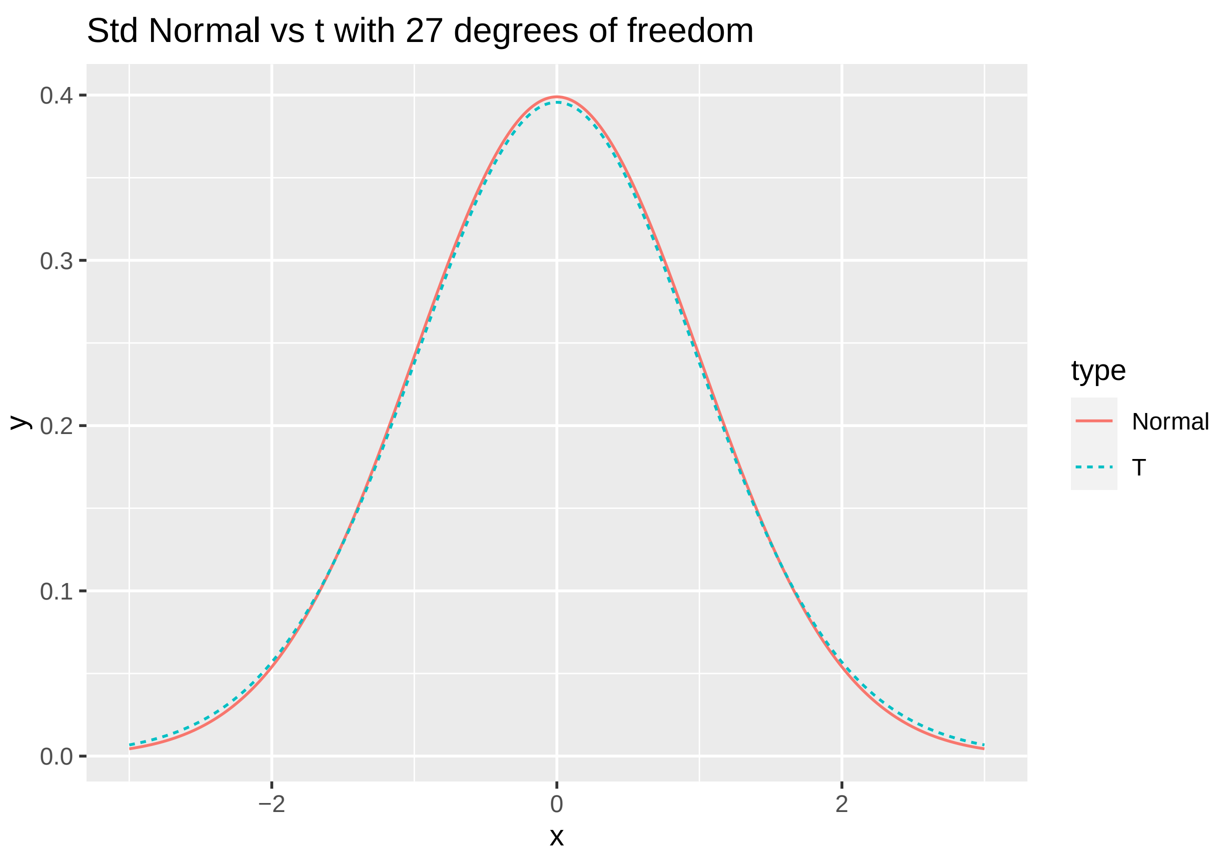
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



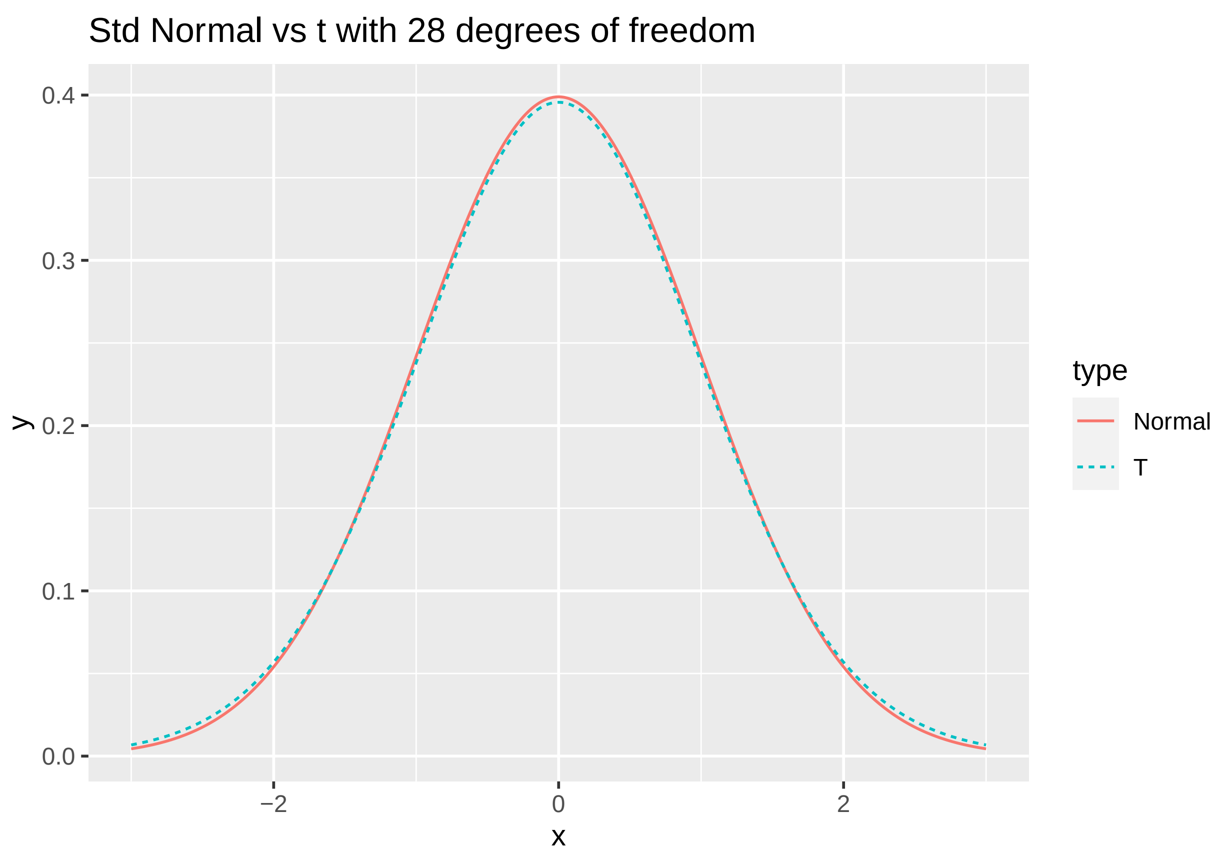
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



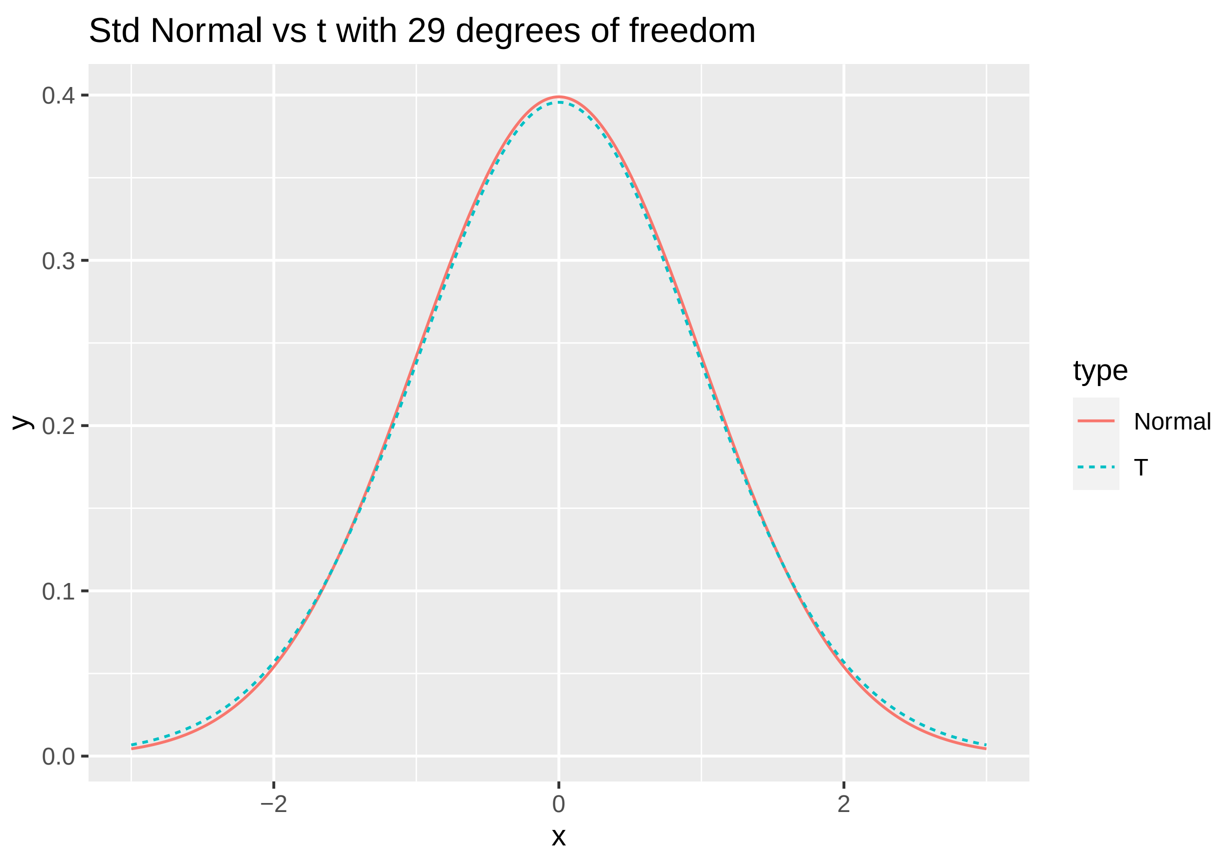
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```

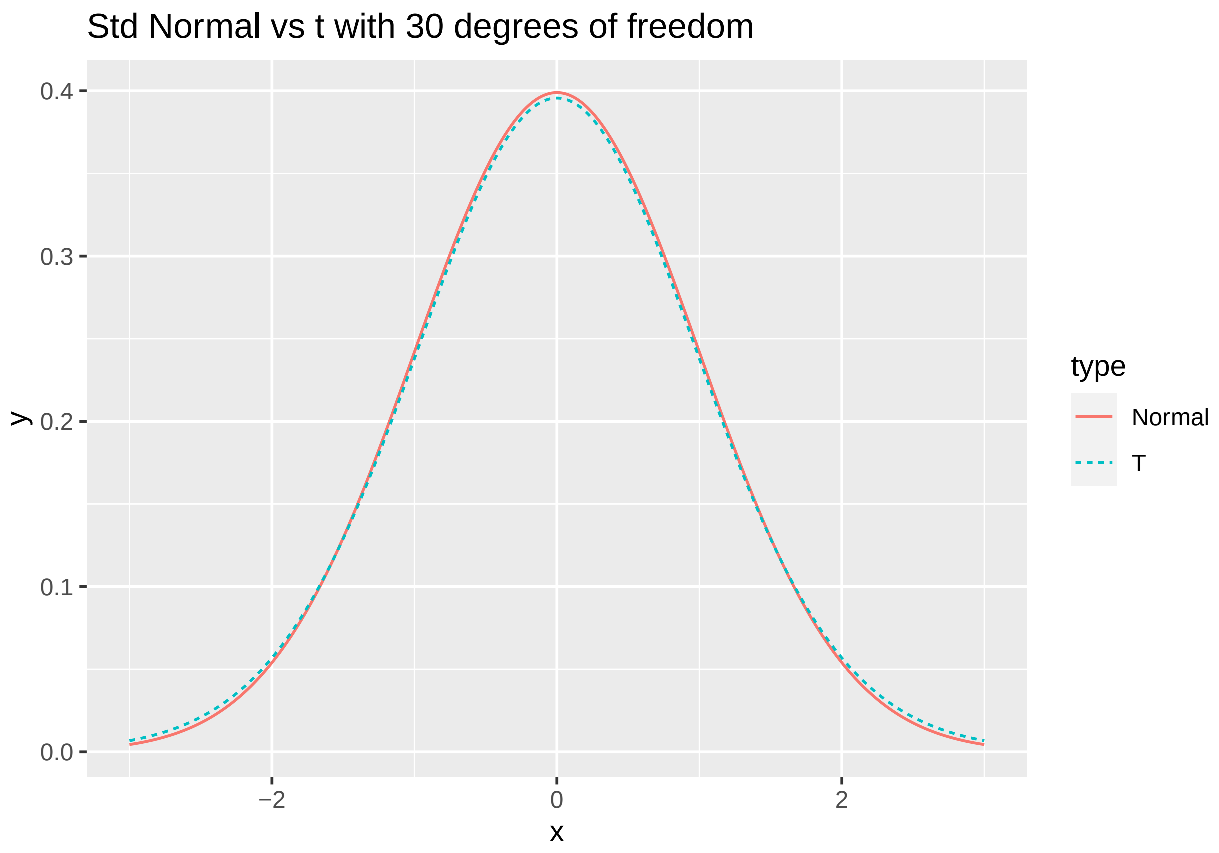


```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```

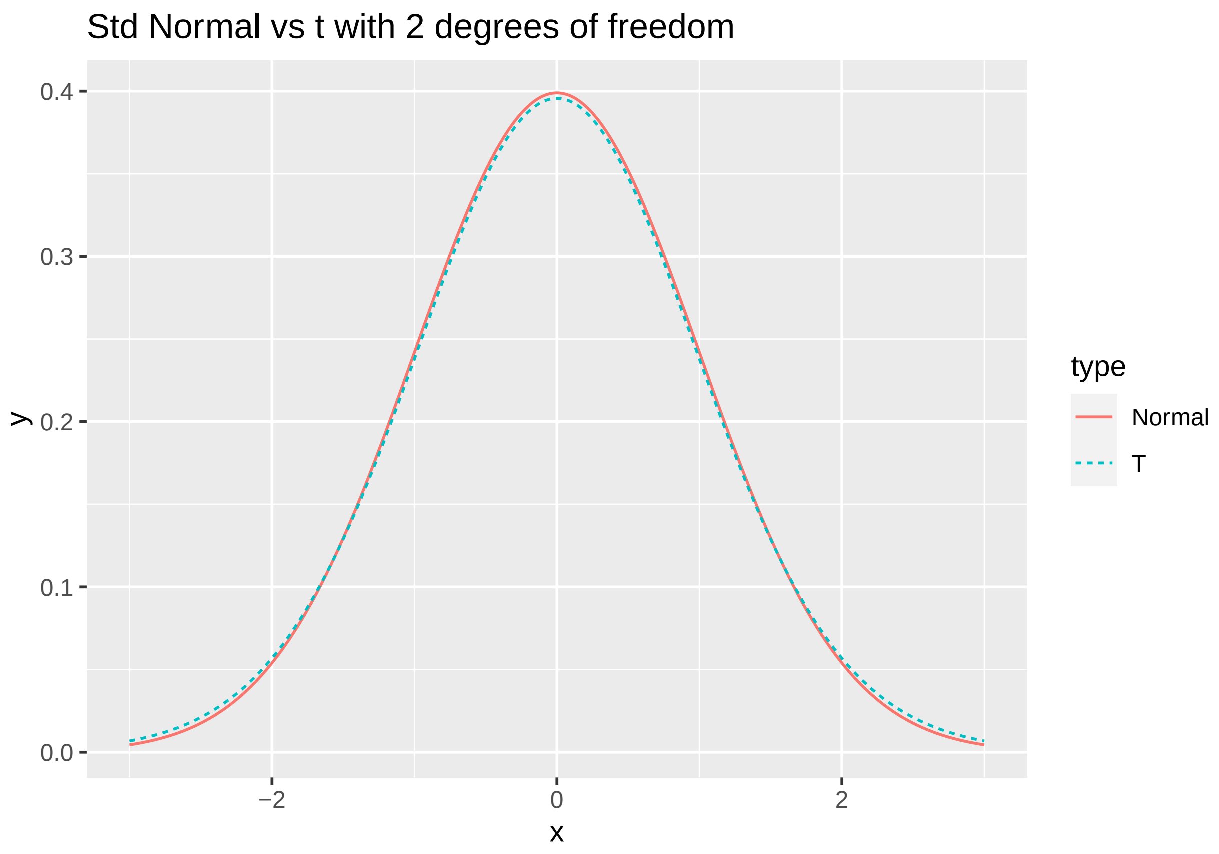

Std Normal vs t with 29 degrees of freedom
0.4
0.3
0.2
0.1
0.0
y
−2
0
2
x
type
Normal
T



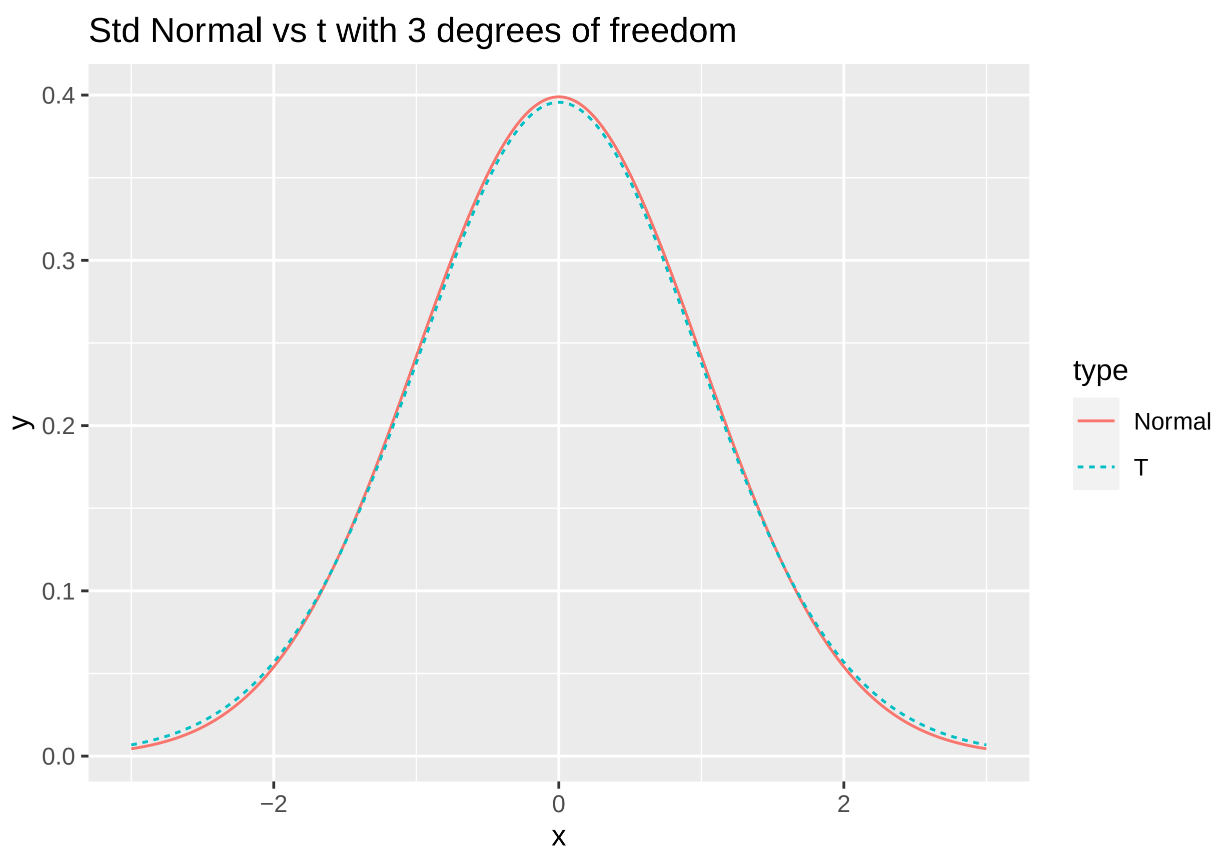
```
b)  In retrospect, perhaps we didn’t need to produce all of those. Rewrite
    your loop so that we only produce graphs for
    $\left\{ 2,3,4,5,10,15,20,25,30\right\}$ degrees of freedom.
    *Hint: you can just modify the vector in the ‘for‘ statement to include*
    *the desired degrees of freedom.*
```

```r
library(ggplot2)
N <- 1000
vector <- c(2,3,4,5,10,15,20,25,30)

result <- result
for( i in vector ){ # loop for df 2:30
  x.grid <- seq(-3, 3, length=N)
data <- data.frame(
  x = c(x.grid, x.grid),
  y = c(dnorm(x.grid), dt(x.grid, df)),
  type = c( rep('Normal',N), rep('T',N) ) )
 # plot data
myplot <- ggplot(data, aes(x=x, y=y, color=type, linetype=type)) +
  geom_line() +
  labs(title = paste('Std Normal vs t with', i, 'degrees of freedom'))
  result[i] <- result
print(myplot) # print graphs 2:30
}
```

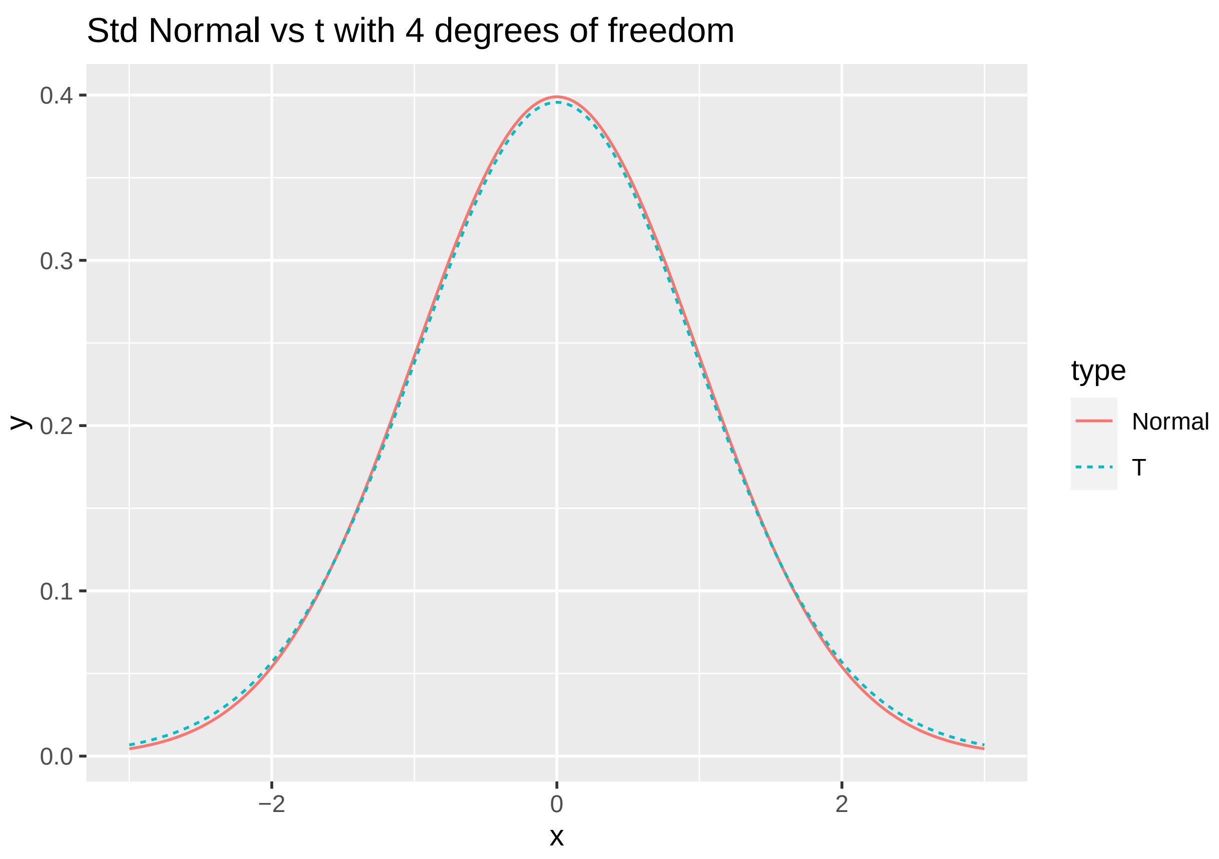
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length

## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



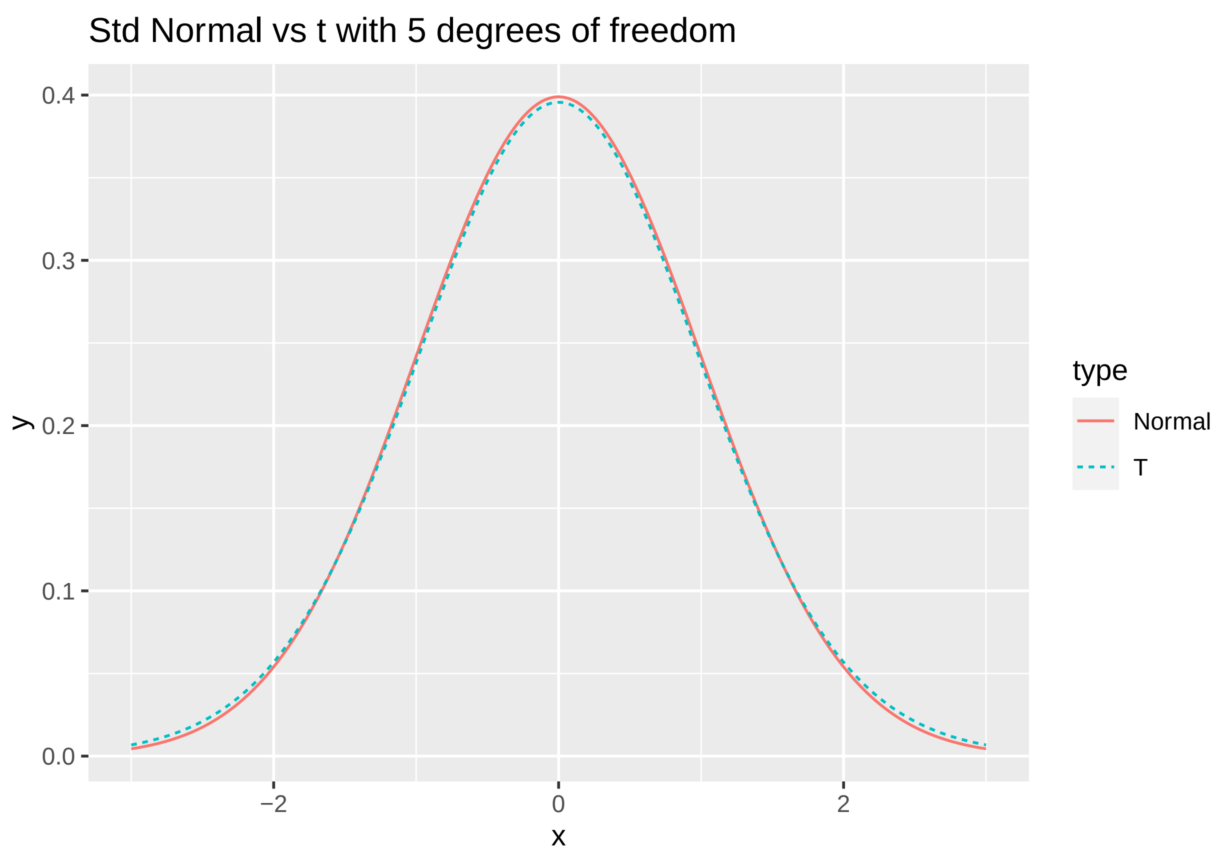
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



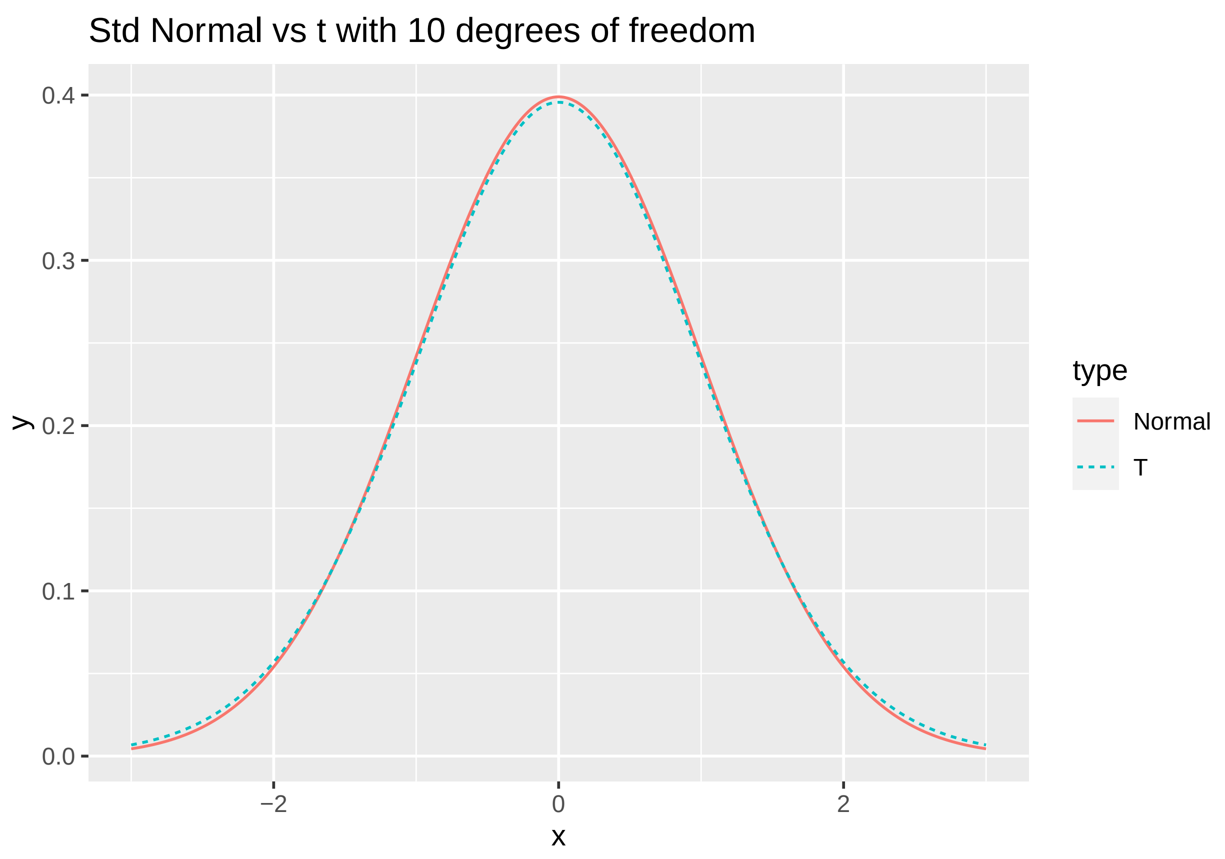
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



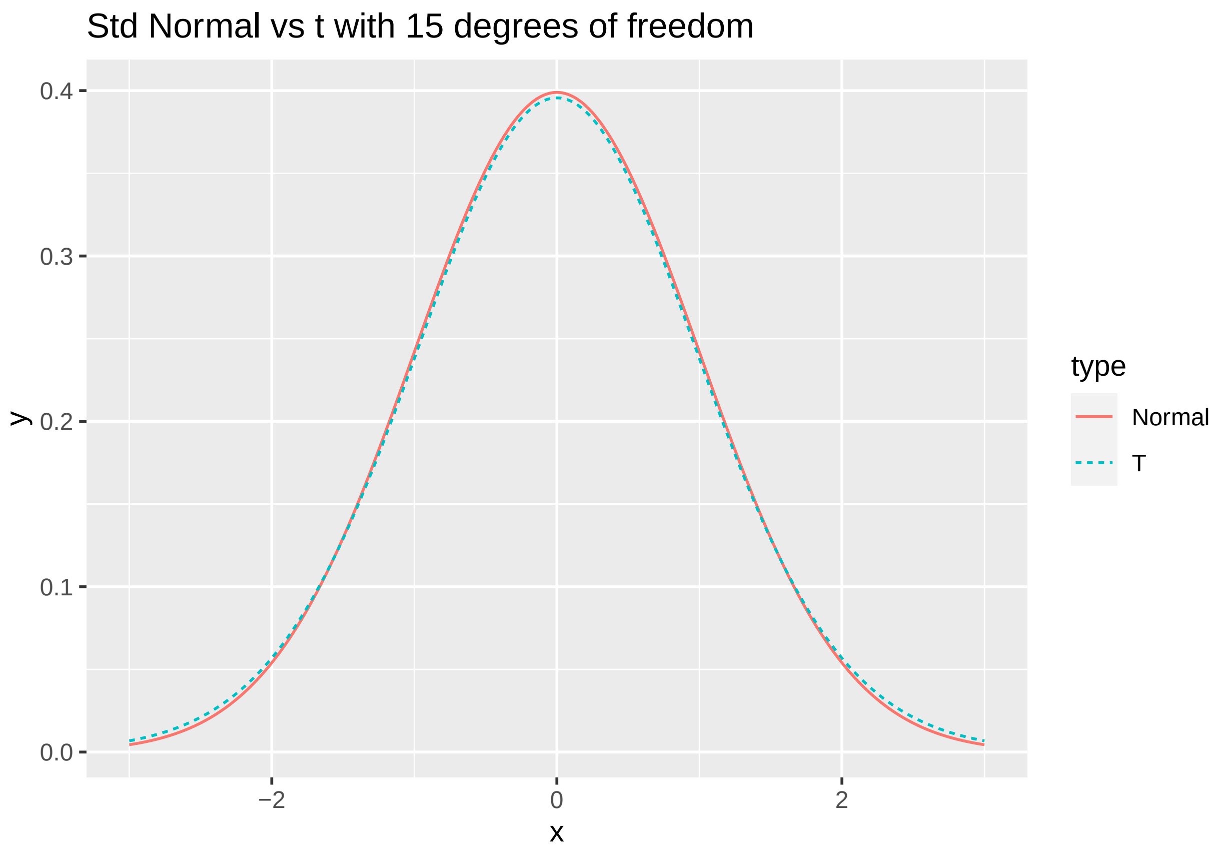
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



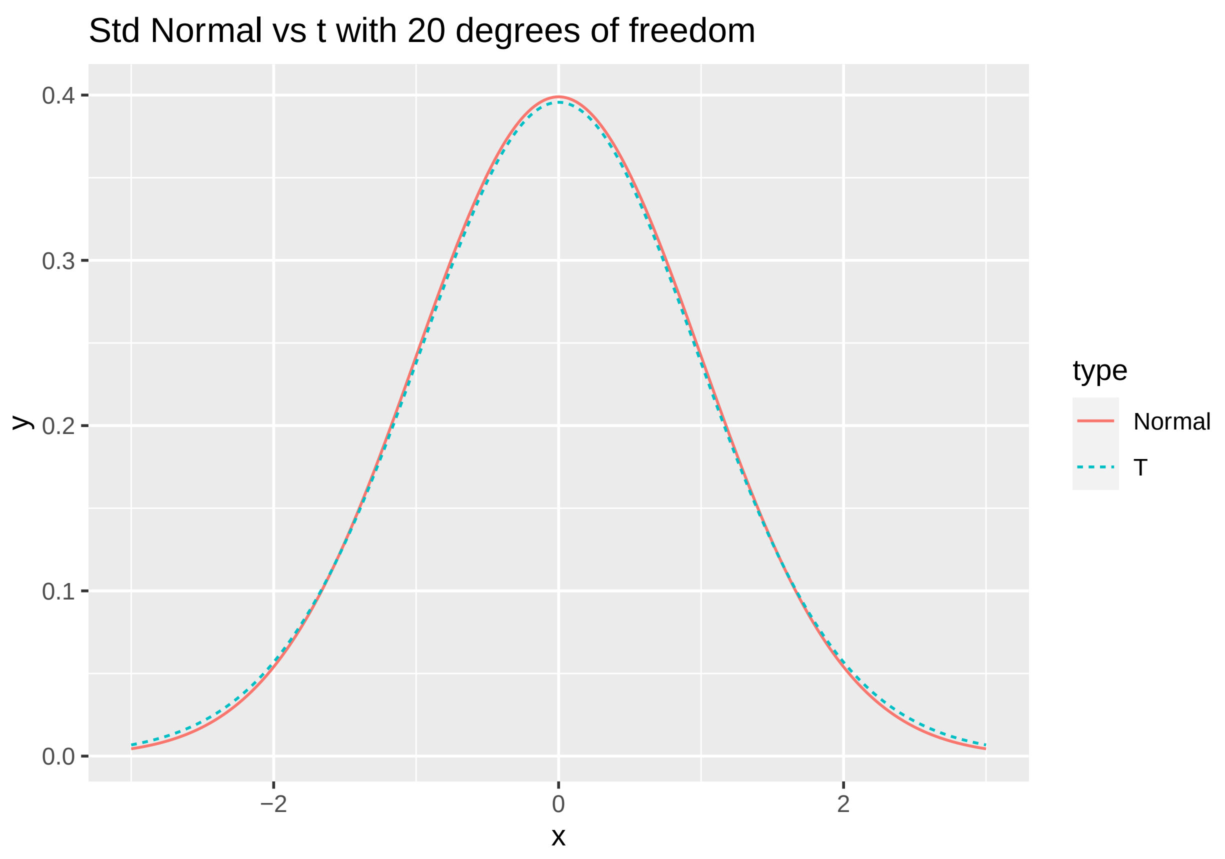
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



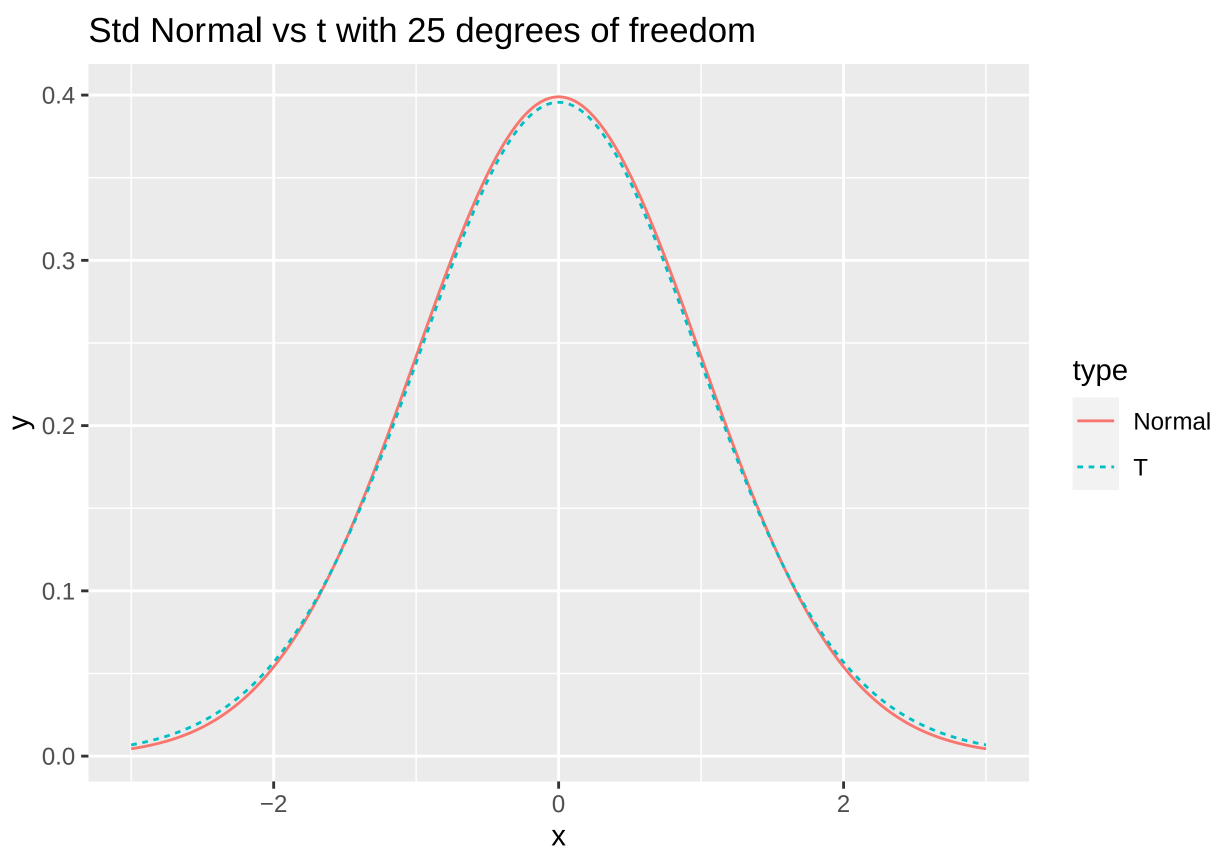
```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```

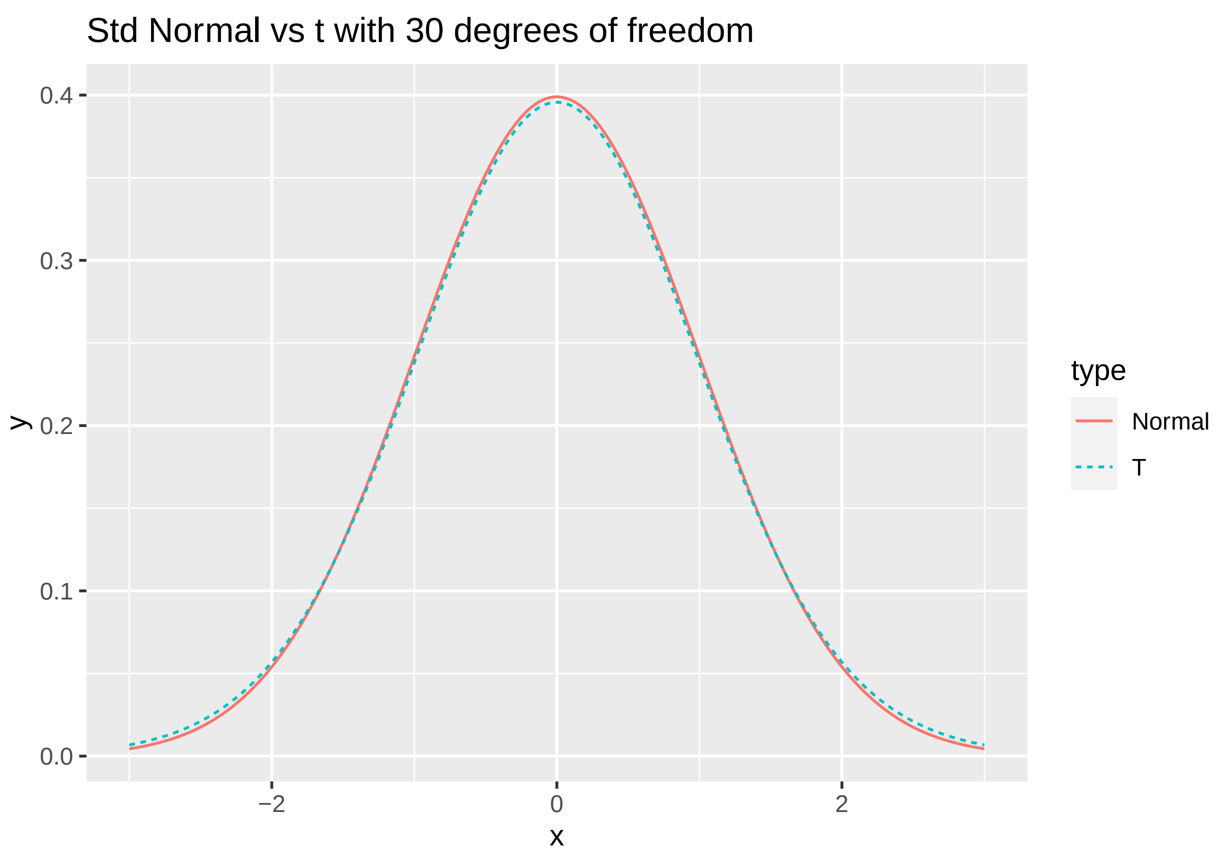


```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```



```
## Warning in result[i] <- result: number of items to replace is not a multiple of
## replacement length
```


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


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