The quick brown fox jumps over the lazy dog.
Generic Steps
Example 1
The quick brown fox jumps over the lazy dog.
Tokenization

# text to be tokenized
"The quick brown fox jumps over the lazy dog."

# split into tokens
"The"    "quick" "brown" "fox"   "jumps" "over"
"the"  "lazy"  "dog."
Tokenization

# text to be tokenized
"The quick brown fox jumps over the lazy dog."

# split into tokens
"The" "quick" "brown" "fox" "jumps" "over" "the" "lazy" "dog."
Tokenization

# text to be tokenized
"The quick brown fox jumps over the lazy dog."

# split into tokens
"The"  "quick" "brown" "fox"   "jumps" "over"
"the"  "lazy" "dog."

# further processing
"the"  "quick" "brown" "fox"   "jumps" "over"
"the"  "lazy" "dog"
Example 2
The quick brown FOX jumps over the lazy dog!
# text to be tokenized
"The quick brown FOX jumps over the lazy dog!"

# split into tokens
"The" "quick" "brown" "FOX" "jumps" "over"
"the" "lazy" "dog!"
# text to be tokenized
"The quick brown FOX jumps over the lazy dog!"

# split into tokens
"The" "quick" "brown" "FOX" "jumps" "over"
"the" "lazy" "dog!"
# text to be tokenized
"The quick brown FOX jumps over the lazy dog!"

# split into tokens
"The" "quick" "brown" "FOX" "jumps" "over" "the" "lazy" "dog!"

# further processing
"the" "quick" "brown" "fox" "jumps" "over" "the" "lazy" "dog"
Example 3
At 9am the quick brown FOX jumps over the lazy dog!
# text to be tokenized
"At 9am the quick brown FOX jumps over the lazy dog!"

# split into tokens
"At" "9am" "the" "quick" "brown" "FOX" "jumps" "over" "the" "lazy" "dog!"

# further processing
"at" "9am" "the" "quick" "brown" "fox" "jumps" "over" "the" "lazy" "dog"
Example 4
Hey, guess what? Every Monday @ 9am the quick brown FOX jumps 10 times over the lazy basset-hound dog. This IS Amazing!!!
"Hey, guess what? Every Monday @ 9am the quick brown FOX jumps 10 times over the lazy basset-hound dog. This IS Amazing!!!
"Hey, guess what? Every Monday @ 9am the quick brown FOX jumps 10 times over the lazy basset-hound dog. This IS Amazing!!!"
"Hey, guess what? Every Monday @ 9am the quick brown FOX jumps 10 times over the lazy basset-hound dog. This IS Amazing!!!"
"Hey, guess what? Every Monday @ 9am the quick brown FOX jumps 10 times over the lazy basset-hound dog. This IS Amazing!!!"
Some common text transformations

Convert to lower case
Remove punctuation symbols
Remove extra spaces
Remove digits
Split into tokens
"Hey, guess what? Every Monday @ 9am the quick brown FOX jumps 10 times over the lazy basset-hound dog. This IS Amazing!!"

"hey guess what every monday am the quick brown fox jumps times over the lazy basset hound dog this is amazing "
"Hey, guess what? Every Monday @ 9am the quick brown FOX jumps 10 times over the lazy basset-hound dog. This IS Amazing!!!"

"hey guess what every monday am the quick brown fox jumps times over the lazy basset hound dog this is amazing"

"hey" "guess" "what" "every" "monday" "am" "the" "quick" "brown" "fox" "jumps" "times" "over" "the" "lazy" "basset" "hound" "dog" "this" "is" "amazing"
Generic Steps
In R
The quick brown fox jumps over the lazy dog.
Some toy text

How to analyze text?

Let’s start by turn it into a character vector in R. Later we’ll turn it into a tabular object.
Let's begin with toy vectors

```r
# vector 1: one element
vec1 = "The quick brown fox jumps over the lazy dog."
```
Once you have text in an R object (e.g. character vector), what’s next?
Tokenization
Tokenization

To analyze text we need to break it apart into single pieces or “words” called tokens.

The process to obtain tokens is known as tokenization.

By the way, the term “word” in text analysis is a more generic term than its linguistic meaning.
Tokenization with regex (and “stringr” functions)

library(tidyverse)

# vector 1
txt1 = str_split(vec1, pattern = " ")
txt1
[[1]]
[1] "The"  "quick"  "brown"  "fox"   "jumps"
[5] "over"  "the"  "lazy"   "dog."
Common Transformations
Some common text transformations

Convert to lower case
Remove punctuation symbols
Remove extra spaces
Remove digits
Remove stop-words
Common transformations

# to lowercase, remove punctuation

tok1 = str_to_lower(txt1[[1]])
tok1 = str_remove(tok1, "[[[:punct:]]]")
tok1

[1] "the" "quick" "brown" "fox" "jumps"
[5] "over" "the" "lazy" "dog"
Common transformations

# we could potentially be interested
# in removing digits
tok1 = str_remove(tok1, "\[[[:digit:]]\])

# and removing extra white spaces
tok1 = str_trim(tok1)
Same previous example but using a 2 element vector
Same text, in a 2-element vector

# vector 2: two elements
vec2 = c(
    "The quick brown fox",
    "jumps over the lazy dog."
)

Gaston Sanchez
Tokenization with regex

# vector 2
txt2 = str_split(vec2, pattern = " ")
txt2
[[1]]
[1] "The" "quick" "brown" "fox"

[[2]]
[1] "jumps" "over" "the" "lazy" "dog."
Some transformations

tok2 = lapply(txt2, str_to_lower)
tok2
[[1]]
[1] "the" "quick" "brown" "fox"

[[2]]
[1] "jumps" "over" "the" "lazy" "dog."
Some transformations

# remove punctuation
tok2 = lapply(
    tok2,
    function(x) str_remove(x, "[[[:punct:]]]"
  )
)
Some transformations

# remove numbers
tok2 = lapply(
  tok2,
  function(x) str_remove(x, "[[[:digit:]]]"
)

# remove extra white spaces
tok2 = lapply(tok2, str_trim)
Many of the previous operations can be easily performed with the R package “tidytext”
“tidytext”
package
Let’s bring back our toy vectors

# vector 1: one element
vec1 = "The quick brown fox jumps over the lazy dog."

# vector 2: two elements
vec2 = c(
    "The quick brown fox",
    "jumps over the lazy dog."
)
Text into a table

# data frame 1
dat1 = data.frame(
    line = seq_along(vec1),
    text = vec1)

# data frame 2
dat2 = data.frame(
    line = seq_along(vec2),
    text = vec2)
### Tokenization with `unnest_tokens()`

```r
tok1 = unnest_tokens(
    tbl = dat1,
    output = word,
    input = text)
```

<table>
<thead>
<tr>
<th>line</th>
<th>word</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
</tr>
<tr>
<td>2</td>
<td>quick</td>
</tr>
<tr>
<td>3</td>
<td>brown</td>
</tr>
<tr>
<td>4</td>
<td>fox</td>
</tr>
<tr>
<td>5</td>
<td>jumps</td>
</tr>
<tr>
<td>6</td>
<td>over</td>
</tr>
<tr>
<td>7</td>
<td>the</td>
</tr>
<tr>
<td>8</td>
<td>lazy</td>
</tr>
<tr>
<td>9</td>
<td>dog</td>
</tr>
</tbody>
</table>
Tokenization with `unnest_tokens()`

```r
tok2 = unnest_tokens(
    tbl = dat2,
    output = word,
    input = text)
```

<table>
<thead>
<tr>
<th>line</th>
<th>word</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
</tr>
<tr>
<td>2</td>
<td>quick</td>
</tr>
<tr>
<td>3</td>
<td>brown</td>
</tr>
<tr>
<td>4</td>
<td>fox</td>
</tr>
<tr>
<td>5</td>
<td>jumps</td>
</tr>
<tr>
<td>6</td>
<td>over</td>
</tr>
<tr>
<td>7</td>
<td>the</td>
</tr>
<tr>
<td>8</td>
<td>lazy</td>
</tr>
<tr>
<td>9</td>
<td>dog</td>
</tr>
</tbody>
</table>
Default behavior of unnest_tokens()

- Each row is split so that there is one token in the output data frame (or tibble).
- Other columns, such as the line number each word came from, are retained.
- Punctuation has been stripped.
- Converts the tokens to lowercase.
- See ?unnest_tokens for more info.