

# String manipulation with stringr :: CHEATSHEET



The **stringr** package provides a set of internally consistent tools for working with character strings, i.e. sequences of characters surrounded by quotation marks.

## Detect Matches

	<b>str_detect(string, pattern, negate = FALSE)</b> Detect the presence of a pattern match in a string. Also <b>str_like()</b> . str_detect(fruit, "a")
	<b>str_starts(string, pattern, negate = FALSE)</b> Detect the presence of a pattern match at the beginning of a string. Also <b>str_ends()</b> . str_starts(fruit, "a")
	<b>str_which(string, pattern, negate = FALSE)</b> Find the indexes of strings that contain a pattern match. str_which(fruit, "a")
	<b>str_locate(string, pattern)</b> Locate the positions of pattern matches in a string. Also <b>str_locate_all()</b> . str_locate(fruit, "a")
	<b>str_count(string, pattern)</b> Count the number of matches in a string. str_count(fruit, "a")

## Subset Strings

	<b>str_sub(string, start = 1L, end = -1L)</b> Extract substrings from a character vector. str_sub(fruit, 1, 3); str_sub(fruit, -2)
	<b>str_subset(string, pattern, negate = FALSE)</b> Return only the strings that contain a pattern match. str_subset(fruit, "p")
	<b>str_extract(string, pattern)</b> Return the first pattern match found in each string, as a vector. Also <b>str_extract_all()</b> to return every pattern match. str_extract(fruit, "[aeiou]")
	<b>str_match(string, pattern)</b> Return the first pattern match found in each string, as a matrix with a column for each () group in pattern. Also <b>str_match_all()</b> . str_match(sentences, "(a the) ([^ +])")

## Manage Lengths

	<b>str_length(string)</b> The width of strings (i.e. number of code points, which generally equals the number of characters). str_length(fruit)
	<b>str_pad(string, width, side = c("left", "right", "both"), pad = " ")</b> Pad strings to constant width. str_pad(fruit, 17)
	<b>str_trunc(string, width, side = c("right", "left", "center"), ellipsis = "...")</b> Truncate the width of strings, replacing content with ellipsis. str_trunc(sentences, 6)
	<b>str_trim(string, side = c("both", "left", "right"))</b> Trim whitespace from the start and/or end of a string. str_trim(str_pad(fruit, 17))
	<b>str_squish(string)</b> Trim whitespace from each end and collapse multiple spaces into single spaces. str_squish(str_pad(fruit, 17, "both"))

## Mutate Strings

	<b>str_sub()</b> <- value. Replace substrings by identifying the substrings with str_sub() and assigning into the results. str_sub(fruit, 1, 3) <- "str"
	<b>str_replace(string, pattern, replacement)</b> Replace the first matched pattern in each string. Also <b>str_remove()</b> . str_replace(fruit, "p", "-")
	<b>str_replace_all(string, pattern, replacement)</b> Replace all matched patterns in each string. Also <b>str_remove_all()</b> . str_replace_all(fruit, "p", "-")
	<b>str_to_lower(string, locale = "en")<sup>1</sup></b> Convert strings to lower case. str_to_lower(sentences)
	<b>str_to_upper(string, locale = "en")<sup>1</sup></b> Convert strings to upper case. str_to_upper(sentences)
	<b>str_to_title(string, locale = "en")<sup>1</sup></b> Convert strings to title case. Also <b>str_to_sentence()</b> . str_to_title(sentences)

## Join and Split

	<b>str_c(..., sep = "", collapse = NULL)</b> Join multiple strings into a single string. str_c(letters, LETTERS)
	<b>str_flatten(string, collapse = "")</b> Combines into a single string, separated by collapse. str_flatten(fruit, ",")
	<b>str_dup(string, times)</b> Repeat strings times times. Also <b>str_unique()</b> to remove duplicates. str_dup(fruit, times = 2)
	<b>str_split_fixed(string, pattern, n)</b> Split a vector of strings into a matrix of substrings (splitting at occurrences of a pattern match). Also <b>str_split()</b> to return a list of substrings and <b>str_split_i()</b> to return the ith substring. str_split_fixed(sentences, " ", n=3)
	<b>str_glue(..., .sep = "", .envir = parent.frame())</b> Create a string from strings and {expressions} to evaluate. str_glue("Pi is {pi}")
	<b>str_glue_data(.x, ..., .sep = "", .envir = parent.frame(), .na = "NA")</b> Use a data frame, list, or environment to create a string from strings and {expressions} to evaluate. str_glue_data(mtcars, "{rownames(mtcars)} has {hp} hp")

## Order Strings

	<b>str_order(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)<sup>1</sup></b> Return the vector of indexes that sorts a character vector. fruit[str_order(fruit)]
	<b>str_sort(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)<sup>1</sup></b> Sort a character vector. str_sort(fruit)

## Helpers

	<b>str_conv(string, encoding)</b> Override the encoding of a string. str_conv(fruit, "ISO-8859-1")
	<b>str_view(string, pattern, match = NA)</b> View HTML rendering of all regex matches. str_view(sentences, "[aeiou])")
	<b>str_equal(x, y, locale = "en", ignore_case = FALSE, ...)<sup>1</sup></b> Determine if two strings are equivalent. str_equal(c("a", "b"), c("a", "c"))
	<b>str_wrap(string, width = 80, indent = 0, exdent = 0)</b> Wrap strings into nicely formatted paragraphs. str_wrap(sentences, 20)

<sup>1</sup> See [bit.ly/ISO639-1](https://bit.ly/ISO639-1) for a complete list of locales.

