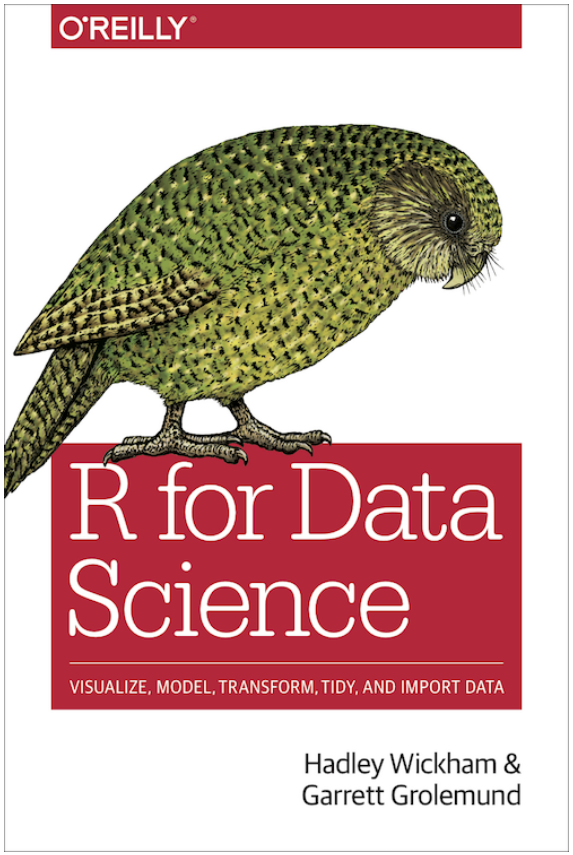
# Socio-Informatics 348

## Data Visualisation
## Strings

Dr Lisa Martin

Department of Information Science
Stellenbosch University

# Today's Reading




*R for Data Science, Chapter 14*

# Strings

- Mainly use the `stringr` package from the `tidyverse`
- `stringr` functions all start with `str_`

# Creating Strings

- Strings are a sequence of characters
- Create strings with "" or ’ ’
- To include quotes in strings, use the opposite quote type to enclose the string, or escape the quote with a backslash (\)

```
double_quote <- "\"" # or '"'
single_quote <- '\'' # or "'"
```

- To include a backslash in strings, use two backslashes.

```
backslash <- "\\"
```

# Viewing Strings

- Printed strings will show escaped characters, use `str_view()` to see the actual string

```
x <- c(single_quote, double_quote, backslash)
x
#> [1] "'"  "\"" "\\"
str_view(x)
#> [1] │ '
#> [2] │ "
#> [3] │ \
```

# Special Characters

- Use `?Quotes` to see a list of special characters

```
x <- c("one\ntwo", "one\ttwo", "\u00b5", "\U0001f604")
x
#> [1] "one\ntwo" "one\ttwo" "µ"        "😄"
str_view(x)
#> [1] │ one
#>     │ two
#> [2] │ one{\t}two
#> [3] │ µ
#> [4] │ 😄
```

# Creating Strings from Data

`str_c()`

- Takes any number of string arguments, and concatenates them together
- Default separator is "", but can be changed with the `sep` argument
- If any argument is `NA`, the result will be `NA`

```
df <- tibble(name = c("Flora", "David", "Terra", NA))
df |> mutate(greeting = str_c("Hi ", name, "!"))
#> # A tibble: 4 × 2
#>   name  greeting
#>   <chr> <chr>
#> 1 Flora Hi Flora!
#> 2 David Hi David!
#> 3 Terra Hi Terra!
#> 4 <NA>  <NA>
```

# Creating Strings from Data

`str_glue()`

- New package: `glue`
- Similar to `str_c()`, but can be 'cleaner' to use

```
df |> mutate(greeting = str_glue("Hi {name}!"))
#> # A tibble: 4 × 2
#>   name  greeting
#>   <chr> <glue>
#> 1 Flora Hi Flora!
#> 2 David Hi David!
#> 3 Terra Hi Terra!
#> 4 <NA>  Hi NA!
```

- Note how NA values are handled

# Creating Strings from Data

`str_flatten()`

- To collapse a vector into a single string, use `str_flatten()`

```
str_flatten(c("x", "y", "z"))
#> [1] "xyz"
str_flatten(c("x", "y", "z"), ", ")
#> [1] "x, y, z"
str_flatten(c("x", "y", "z"), ", ", last = ", and ")
#> [1] "x, y, and z"
```

- Better to use with `summarise()`
- `str_c()` and `str_glue()` work on vectors element-wise – better with `mutate()`

# Creating Strings from Data

## `str_flatten()`

```
df <- tribble(
  ~ name, ~ fruit,
  "Carmen", "banana",
  "Carmen", "apple",
  "Marvin", "nectarine",
  "Terence", "cantaloupe",
  "Terence", "papaya",
  "Terence", "mandarin"
)
df |>
  group_by(name) |>
  summarize(fruits = str_flatten(fruit, ", "))
#> # A tibble: 3 × 2
#>   name    fruits
#>   <chr>   <chr>
#> 1 Carmen  banana, apple
#> 2 Marvin  nectarine
#> 3 Terence cantaloupe, papaya, mandarin
```

# Extracting Data from Strings

**Separating into rows:**

```
df1 <- tibble(x = c("a,b,c", "d,e", "f"))
df1 |>
  separate_longer_delim(x, delim = ",")
#> # A tibble: 6 × 1
#>   x
#>   <chr>
#> 1 a
#> 2 b
#> 3 c
#> 4 d
#> 5 e
#> 6 f
```

# Extracting Data from Strings

**Separating into rows:**

```
df2 <- tibble(x = c("1211", "131", "21"))
df2 |>
  separate_longer_position(x, width = 1)
#> # A tibble: 9 × 1
#>   x
#>   <chr>
#> 1 1
#> 2 2
#> 3 1
#> 4 1
#> 5 1
#> 6 3
#> # ℹ 3 more rows
```

# Extracting Data from Strings

**Separating into columns:**

```
df3 <- tibble(x = c("a10.1.2022", "b10.2.2011", "e15.1.2015"))
df3 |>
  separate_wider_delim(
    x,
    delim = ".",
    names = c("code", "edition", "year")
  )
#> # A tibble: 3 × 3
#>   code  edition year
#>   <chr> <chr>   <chr>
#> 1 a10   1       2022
#> 2 b10   2       2011
#> 3 e15   1       2015
```

# Extracting Data from Strings

**Separating into columns:**

```
df3 |>
  separate_wider_delim(
    x,
    delim = ".",
    names = c("code", NA, "year")
  )
#> # A tibble: 3 × 2
#>   code  year
#>   <chr> <chr>
#> 1 a10   2022
#> 2 b10   2011
#> 3 e15   2015
```

# Extracting Data from Strings

**Separating into columns:**

```
df4 <- tibble(x = c("202215TX", "202122LA", "202325CA"))
df4 |>
  separate_wider_position(
    x,
    widths = c(year = 4, age = 2, state = 2)
  )
#> # A tibble: 3 × 3
#>   year  age   state
#>   <chr> <chr> <chr>
#> 1 2022  15    TX
#> 2 2021  22    LA
#> 3 2023  25    CA
```

# Letters

`str_length()`

- Returns the number of characters in a string

```
str_length(c("a", "R for data science", NA))
#> [1]  1 18 NA
```

# Letters

## `str_sub()` – **Subsetting**

- Provide starting point and number of characters to extract

```
x <- c("Apple", "Banana", "Pear")
str_sub(x, 1, 3)
#> [1] "App" "Ban" "Pea"
```

- Use negative values to count back from the end of the string

```
str_sub(x, -3, -1)
#> [1] "ple" "ana" "ear"
```

- Won't fail if the string is too shortly

```
str_sub("a", 1, 5)
#> [1] "a"
```