Strings manipulation in R

Alexander Matrunich
8 June 2017, R-Ladies Tbilisi

What are we going to do

- String basics (length, combining, sebsetting)
- Regular expressions (specials, anchors, classes, repetition, grouping)
- Strings and factors
- File paths

The content of these slides is heavily based on the chapter Strings from **R for Data Science** by Garrett Grolemund and Hadley Wickham.

Alexander (Alex) Matrunich

Specialist degree in sociology at Pskov Volny Institute

Free open source enthusiast (GNU/Linux, LibreOffice, etc.)

Open data contributor (Wikipedia, OpenStreetMap)

Notable projects with R:

- Opinion poll data processing and report generation
- Exit-poll real-time data processing and report generation
- R training for the United Nations FAO staff
- World trade data analysis for the UN FAO

String basics

Use “ or ' to delimit strings. Double quotes are preferred.

```
s1 <- "Hello - Привет - გამარჯობა"
s2 <- 'ჟენევის მოლაპარაკებების "ახალი
კონტექსტი"'
s2
[1] "ჟენევის მოლაპარაკებების \"ახალი
კონტექსტი\""
s3 <- c("ერთი", "ორი", "სამი")
s3
[1] "ერთი" "ორი"  "სამი"
```

Some special characters

To include special symbol in a string you have to “escape” it with backslash.

Symbol	**Meaning**
\n	newline
\t	tab
\“	quotation mark ”
\'	apostrophe '

Use ? " ' " to list them all.

R package stringr

In base R there are many functions to work with character string.

Package `stringr` provides:

- consistent framework;
- ability to work with many languages.

```
library("stringr")
```

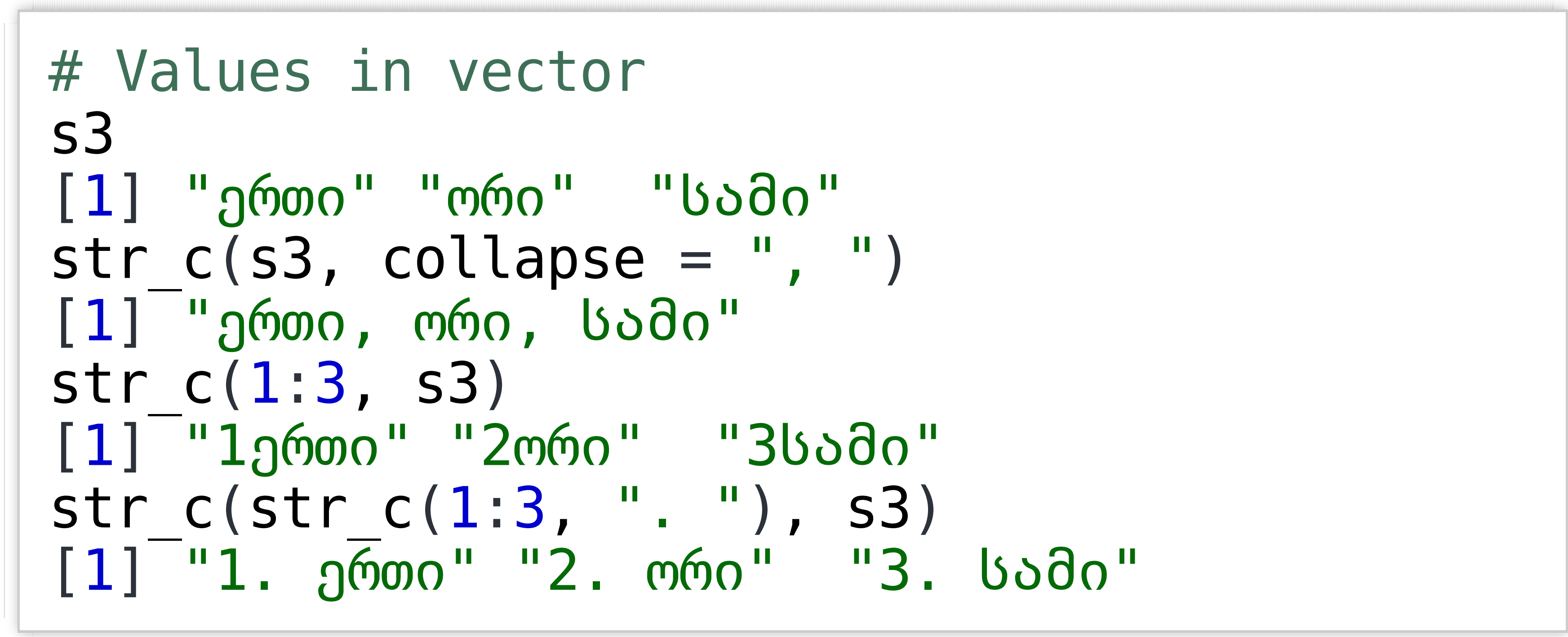
String length

```
s3
[1] "ერთი" "ორი"  "სამი"
str_length(s3)
[1] 4 3 4
str_length(NA) # Beware of using base
nchar() on outdated R
[1] NA
```

Combining strings

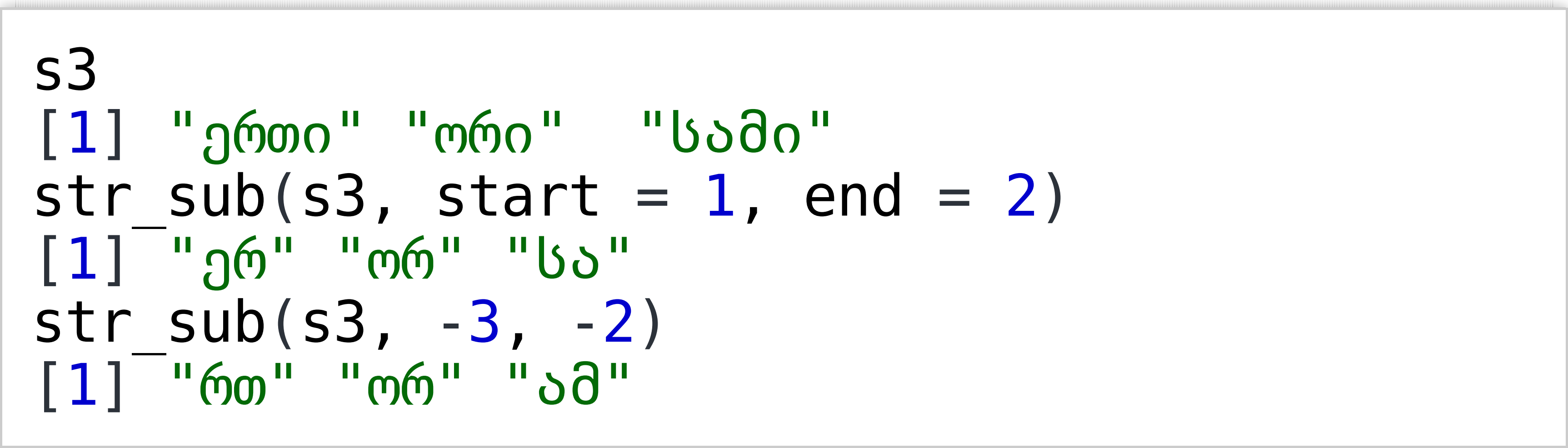
```
# Separate values
str_c("ერთი", "ორი", "სამი")
[1] "ერთიორისამი"
str_c("ერთი", "ორი", "სამი", sep = ", ")
[1] "ერთი, ორი, სამი"
```

Combining strings



```
# Values in vector
s3
[1] "ერთი" "ორი"  "სამი"
str_c(s3, collapse = ", ")
[1] "ერთი, ორი, სამი"
str_c(1:3, s3)
[1] "1ერთი" "2ორი"  "3სამი"
str_c(str_c(1:3, ". "), s3)
[1] "1. ერთი" "2. ორი"  "3. სამი"
```

Subsetting strings



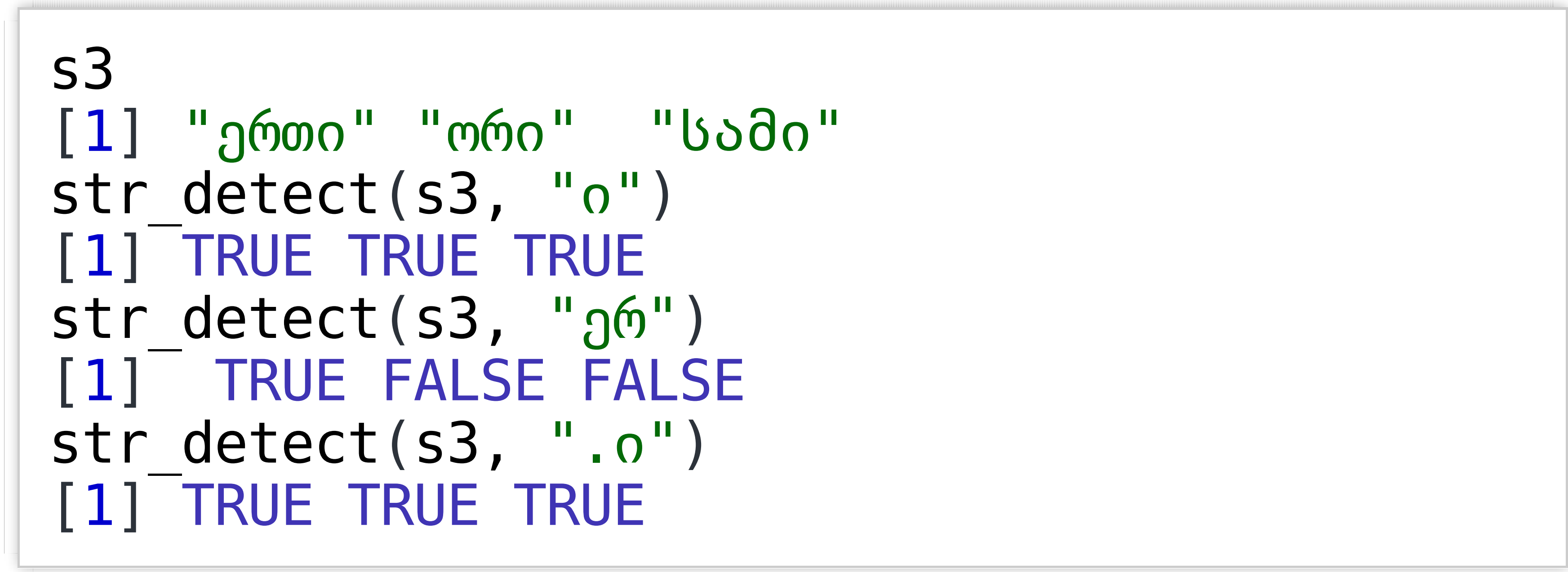
```
s3
[1] "ერთი" "ორი"  "სამი"
str_sub(s3, start = 1, end = 2)
[1] "ერ" "ორ" "სა"
str_sub(s3, -3, -2)
[1] "რთ" "ორ" "ამ"
```

Lower and upper cases

```
str_to_lower(s1)
[1] "hello - привет - გამარჯობა"
str_to_upper(s1)
[1] "HELLO - ПРИВЕТ - გამარჯობა"
str_to_title(s1)
[1] "Hello - Привет - გამარჯობა"
```

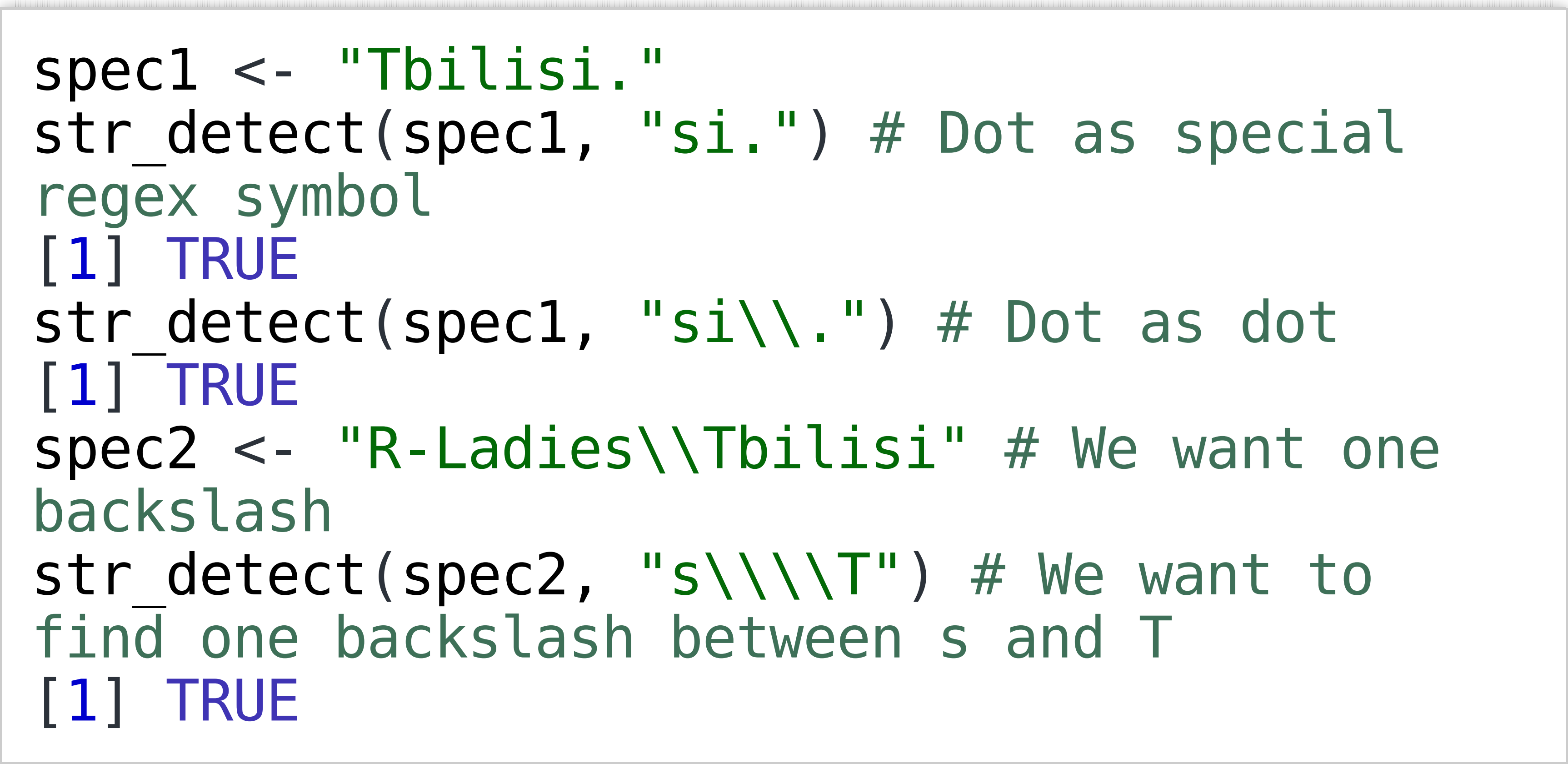
Regular expressions aka regex

Basic matches



```
s3
[1] "ერთი" "ორი"  "სამი"
str_detect(s3, "ი")
[1] TRUE TRUE TRUE
str_detect(s3, "ერ")
[1]  TRUE FALSE FALSE
str_detect(s3, ".ი")
[1] TRUE TRUE TRUE
```

Special symbols in regex



```
spec1 <- "Tbilisi."
str_detect(spec1, "si.") # Dot as special
regex symbol
[1] TRUE
str_detect(spec1, "si\\.") # Dot as dot
[1] TRUE
spec2 <- "R-Ladies\\Tbilisi" # We want one
backslash
str_detect(spec2, "s\\\\T") # We want to
find one backslash between s and T
[1] TRUE
```

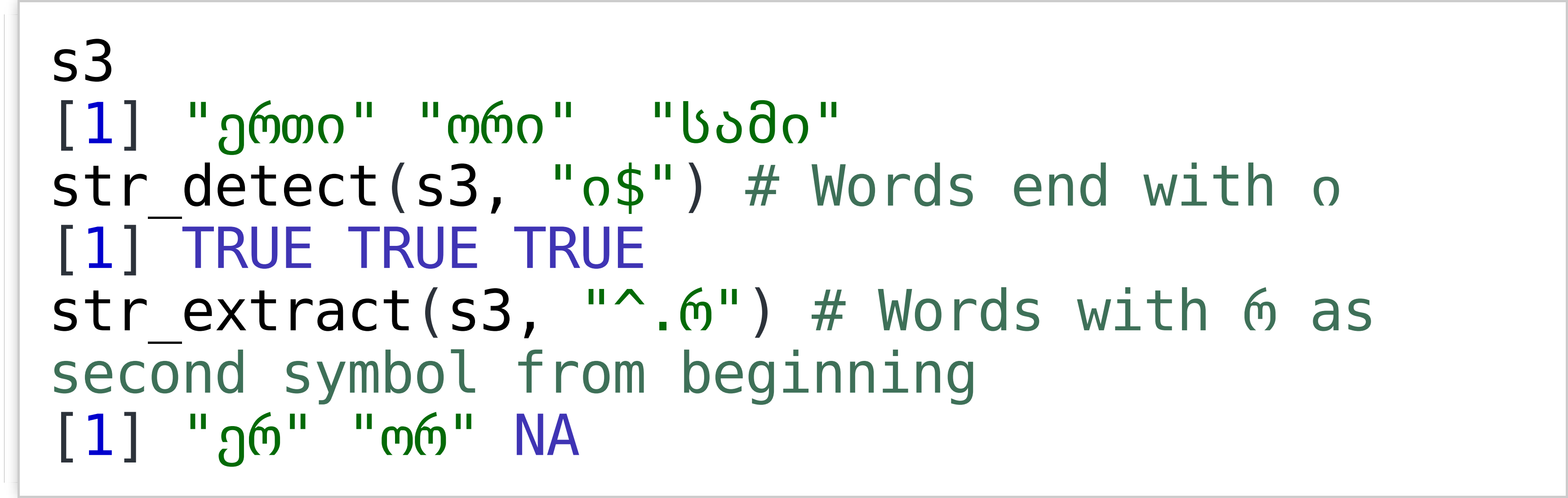
Four backslashes to match one?

```
\\\\
```

1. We are looking for backslash (#4).
2. Escape special symbol in regex pattern (#2).
3. Escape in string for original backslash (#3).
4. Escape in string for regex escape backslash (#1).

Anchors

- ^ - start of string (Start of Elon Mask's rocket)
- $ - end of string (Profits of Mask after successful lunch)



```
s3
[1] "ერთი" "ორი"  "სამი"
str_detect(s3, "ი$") # Words end with ი
[1] TRUE TRUE TRUE
str_extract(s3, "^.რ") # Words with რ as
second symbol from beginning
[1] "ერ" "ორ" NA
```

Character classes and alternatives

- \\d - any digit;
- \\s - any whitespace (space, tab, newline);
- [abc] - a, b or c;
- [^abc] - anything except a, b or c.

```
str_extract(c("7a", "56bc", "a7"),
"\\d[ab]")
[1] "7a" "6b" NA
```

Repetition

- `?` - 0 or 1;
- `+` - 1 or more;
- `*` - 0 or more;
- `{n}` - exactly n;
- `{n,}` - n or more;
- `{,m}` - at most m;
- `{n,m}` - between n and m.

Grouping and backreferences

Each pair of parentheses defines a “group”. Use backreferences like \1, \2 to refer to them.

```r
c("banana", "coconut", "cucumber") %>%
  # R U comfortable with pipes?
  str_extract("(..)\\1")
[1] "anan" "coco" "cucu"
```

Tools to work with strings

Already known

- `str_length()`
- `str_to_lower()`
- `str_to_upper()`
- `str_to_title()`
- `str_detect()`
- `str_extract()`

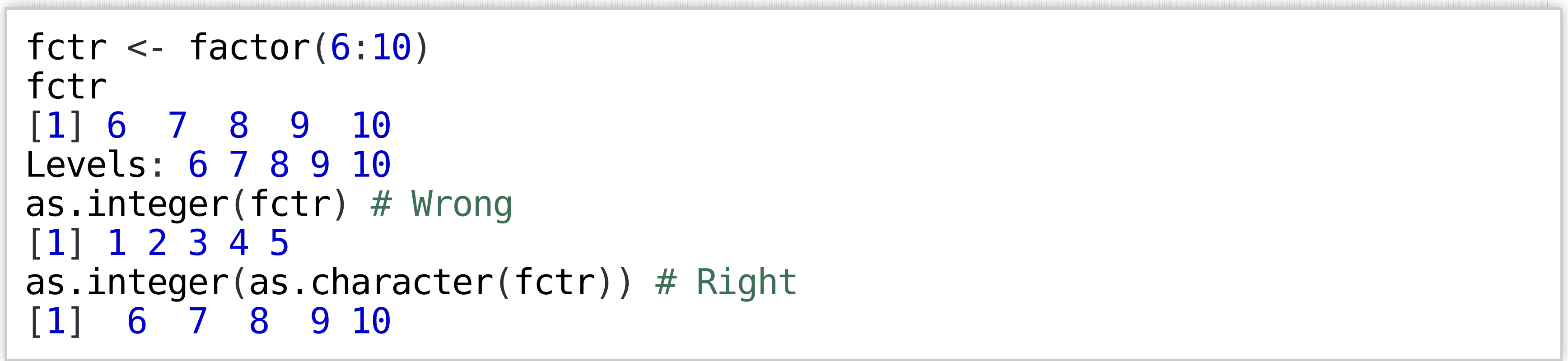
Tools to explore

- `str_count()`
- `str_subset()`
- `str_replace()`
- `str_split()`
- ...

Factor vectors vs Character vectors

- Factor == Categorical
- In ancient times factor vectors were preferable to characters vectors due to speed
- Current defaults in `read.table()`, `data.frame()` is legacy
- Use factors when you want limited set of categories
- Check **`forcats`** package if you often work with factors

Convert numeric factor to number



```
fctr <- factor(6:10)
fctr
[1] 6  7  8  9  10
Levels: 6 7 8 9 10
as.integer(fctr) # Wrong
[1] 1 2 3 4 5
as.integer(as.character(fctr)) # Right
[1]  6  7  8  9 10
```

Working with file paths

Use `file.path()` to construct file paths, it use correct separators.

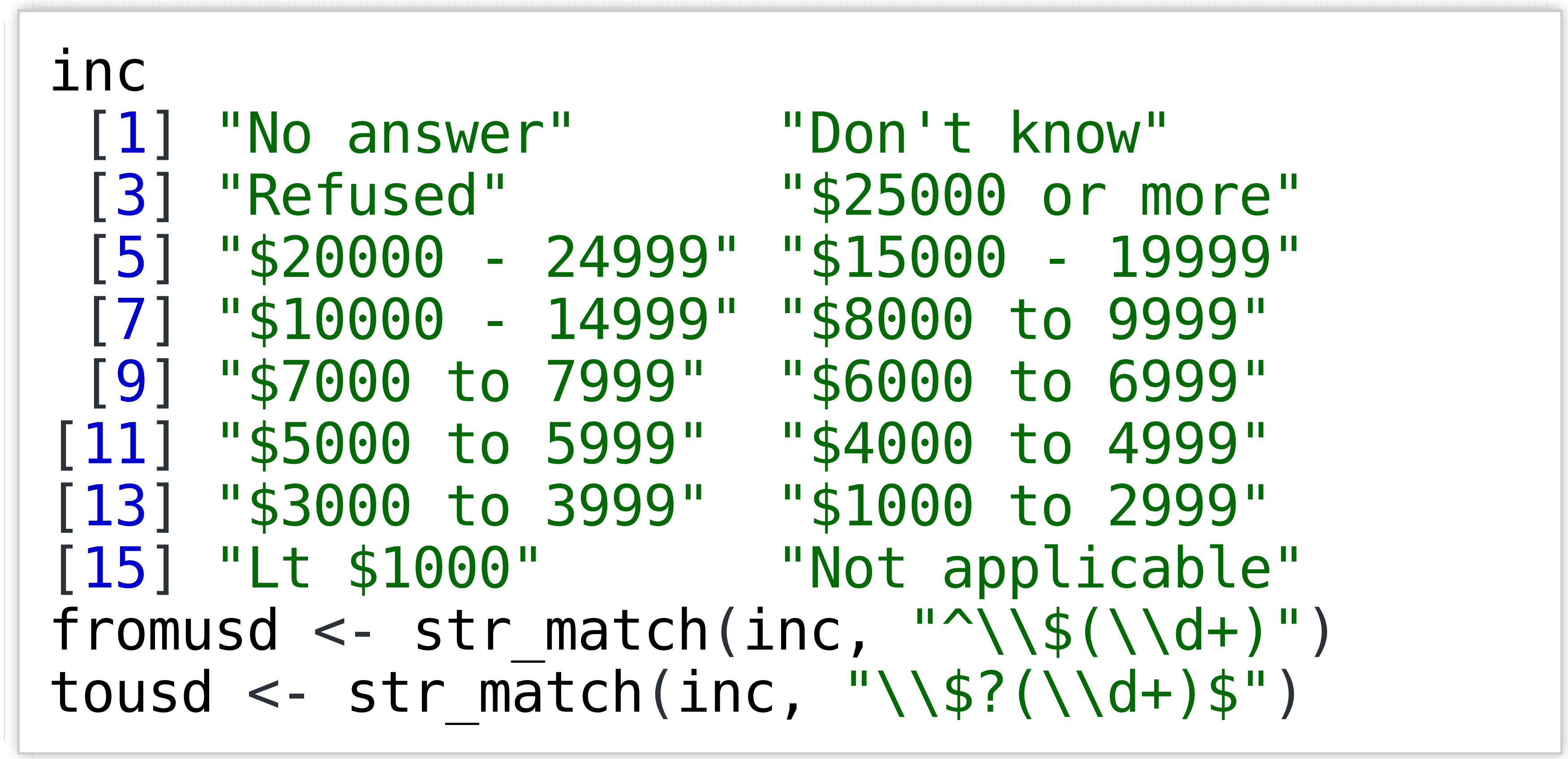
```
disk <- "c:"
docs <- "users"
user <- "alex"
file.path(disk, docs, user, "projects",
"rladies_stringr")
[1] "c:/users/alex/projects/rladies_stringr"
```

Income level 1

Let's use a sample from the General Social survey

```
glimpse(forcats::gss_cat, width = 40)
Observations: 21,483
Variables: 9
$ year    <int> 2000, 2000, 2000, 2...
$ marital <fctr> Never married, Div...
$ age     <int> 26, 48, 67, 39, 25,...
$ race    <fctr> White, White, Whit...
$ rincome <fctr> $8000 to 9999, $80...
$ partyid <fctr> Ind,near rep, Not ...
$ relig   <fctr> Protestant, Protes...
$ denom   <fctr> Southern baptist, ...
$ tvhours <int> 12, NA, 2, 4, 1, NA...
levels(forcats::gss_cat$rincome) -> inc
```

Income level 2



```
inc
 [1] "No answer"        "Don't know"
 [3] "Refused"          "$25000 or more"
 [5] "$20000 - 24999" "$15000 - 19999"
 [7] "$10000 - 14999" "$8000 to 9999"
 [9] "$7000 to 7999"  "$6000 to 6999"
[11] "$5000 to 5999"  "$4000 to 4999"
[13] "$3000 to 3999"  "$1000 to 2999"
[15] "Lt $1000"         "Not applicable"
fromusd <- str_match(inc, "^\\$(\\d+)")
tousd <- str_match(inc, "\\$?(\\d+)$")
```

Income level 3

```
inc_matrix <- c(inc, fromusd)
dim(inc_matrix) <- c(16, 3) # "^\\$(\\d+)"
inc_matrix
      [,1]             [,2]     [,3]   
 [1,] "No answer"      NA       NA     
 [2,] "Don't know"     NA       NA     
 [3,] "Refused"        NA       NA     
 [4,] "$25000 or more" "$25000" "25000"
 [5,] "$20000 - 24999" "$20000" "20000"
 [6,] "$15000 - 19999" "$15000" "15000"
 [7,] "$10000 - 14999" "$10000" "10000"
 [8,] "$8000 to 9999"  "$8000"  "8000" 
 [9,] "$7000 to 7999"  "$7000"  "7000" 
[10,] "$6000 to 6999"  "$6000"  "6000" 
[11,] "$5000 to 5999"  "$5000"  "5000" 
[12,] "$4000 to 4999"  "$4000"  "4000" 
[13,] "$3000 to 3999"  "$3000"  "3000" 
[14,] "$1000 to 2999"  "$1000"  "1000" 
```

```
[15,] "Lt $1000"       NA       NA
[16,] "Not applicable" NA       NA
```

Income level 4

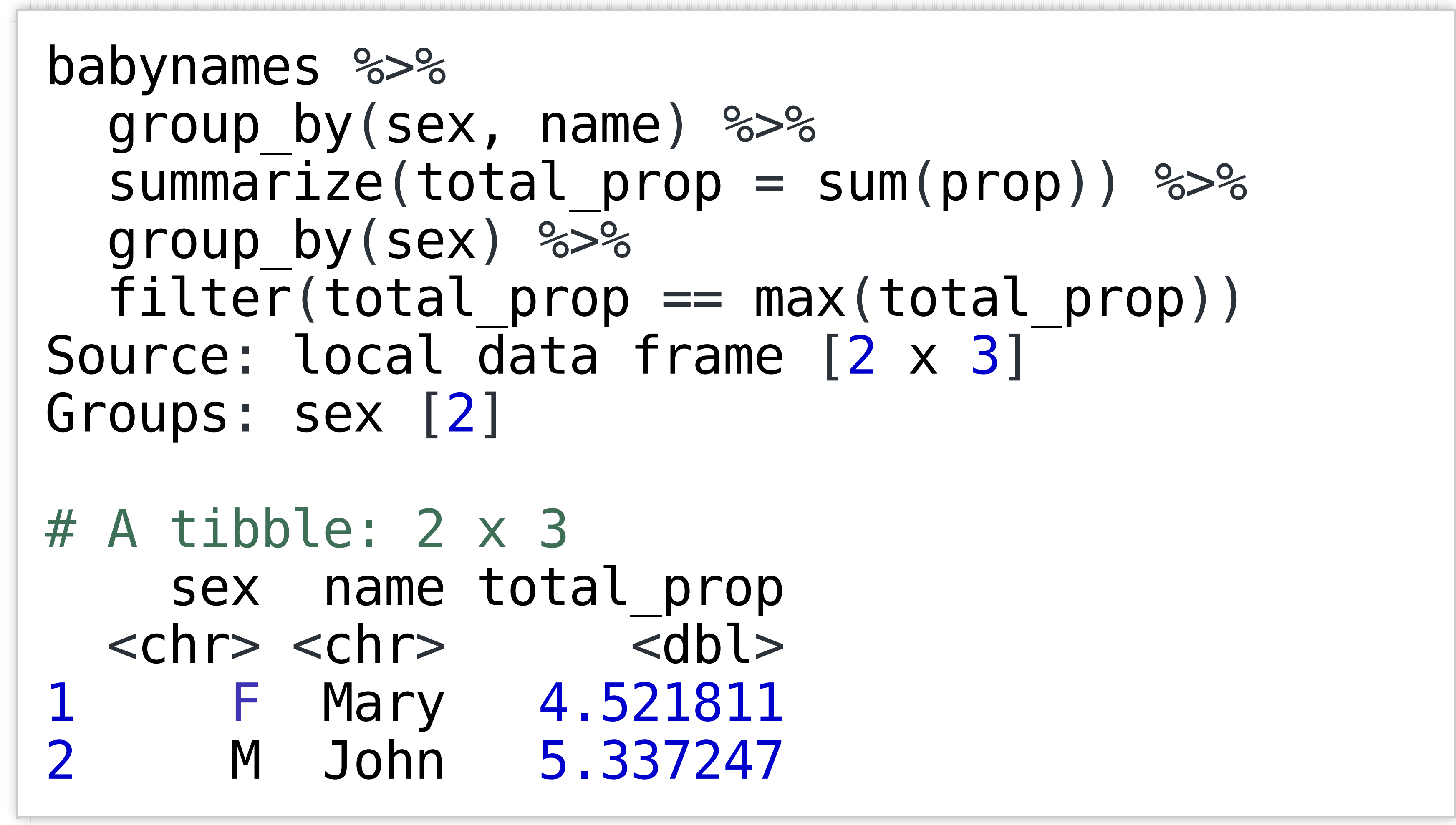
```
tibble(original = inc, from = fromusd[,2],
to = tousd[,2]) %>%
  mutate_at(vars(from, to), as.integer) %>%
  mutate(usd = (to + from) / 2)
# A tibble: 16 x 4
         original  from    to      usd
            <chr> <int> <int>   <dbl>
 1      No answer    NA    NA       NA
 2     Don't know    NA    NA       NA
 3        Refused    NA    NA       NA
 4 $25000 or more 25000    NA       NA
 5 $20000 - 24999 20000 24999 22499.5
 6 $15000 - 19999 15000 19999 17499.5
 7 $10000 - 14999 10000 14999 12499.5
 8  $8000 to 9999  8000  9999  8999.5
 9  $7000 to 7999  7000  7999  7499.5
10  $6000 to 6999  6000  6999  6499.5
11  $5000 to 5999  5000  5999  5499.5
```

```
12  $4000 to 4999  4000  4999  4499.5
13  $3000 to 3999  3000  3999  3499.5
14  $1000 to 2999  1000  2999  1999.5
15       Lt $1000    NA  1000      NA
16 Not applicable    NA    NA      NA
```

Dataset babynames

```
library("tidyverse") # To get dplyr etc.
data("babynames", package = "babynames")
glimpse(babynames)
Observations: 1,858,689
Variables: 5
$ year <dbl> 1880, 1880, 1880, 1880, 1880,
18...
$ sex  <chr> "F", "F", "F", "F", "F", "F",
"F...
$ name <chr> "Mary", "Anna", "Emma",
"Elizabe...
$ n    <int> 7065, 2604, 2003, 1939, 1746,
15...
$ prop <dbl> 0.07238433, 0.02667923,
0.020521...
```

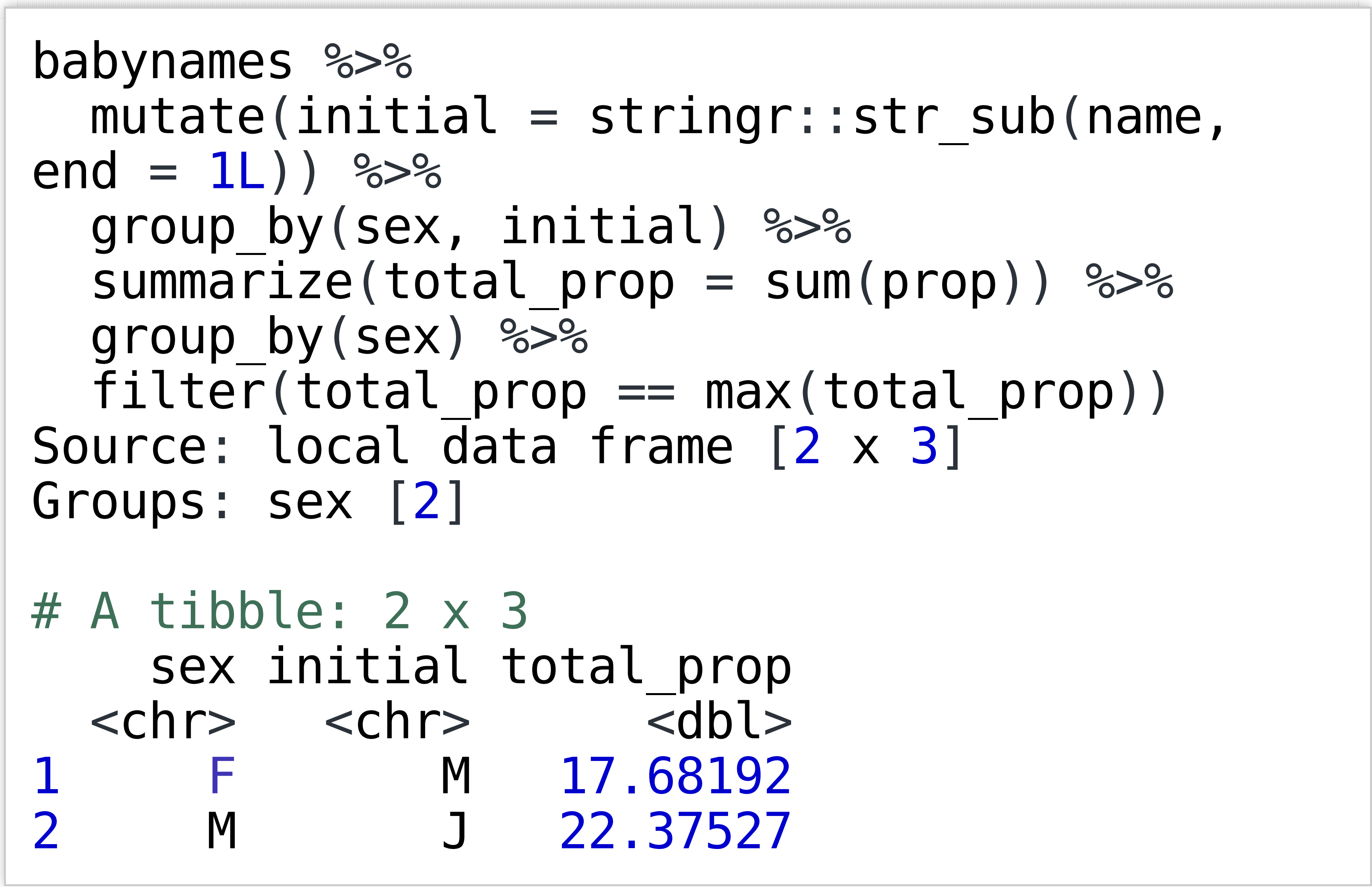
Most popular names



```
babynames %>%
  group_by(sex, name) %>%
  summarize(total_prop = sum(prop)) %>%
  group_by(sex) %>%
  filter(total_prop == max(total_prop))
Source: local data frame [2 x 3]
Groups: sex [2]

# A tibble: 2 x 3
    sex  name total_prop
  <chr> <chr>      <dbl>
1     F  Mary   4.521811
2     M  John   5.337247
```

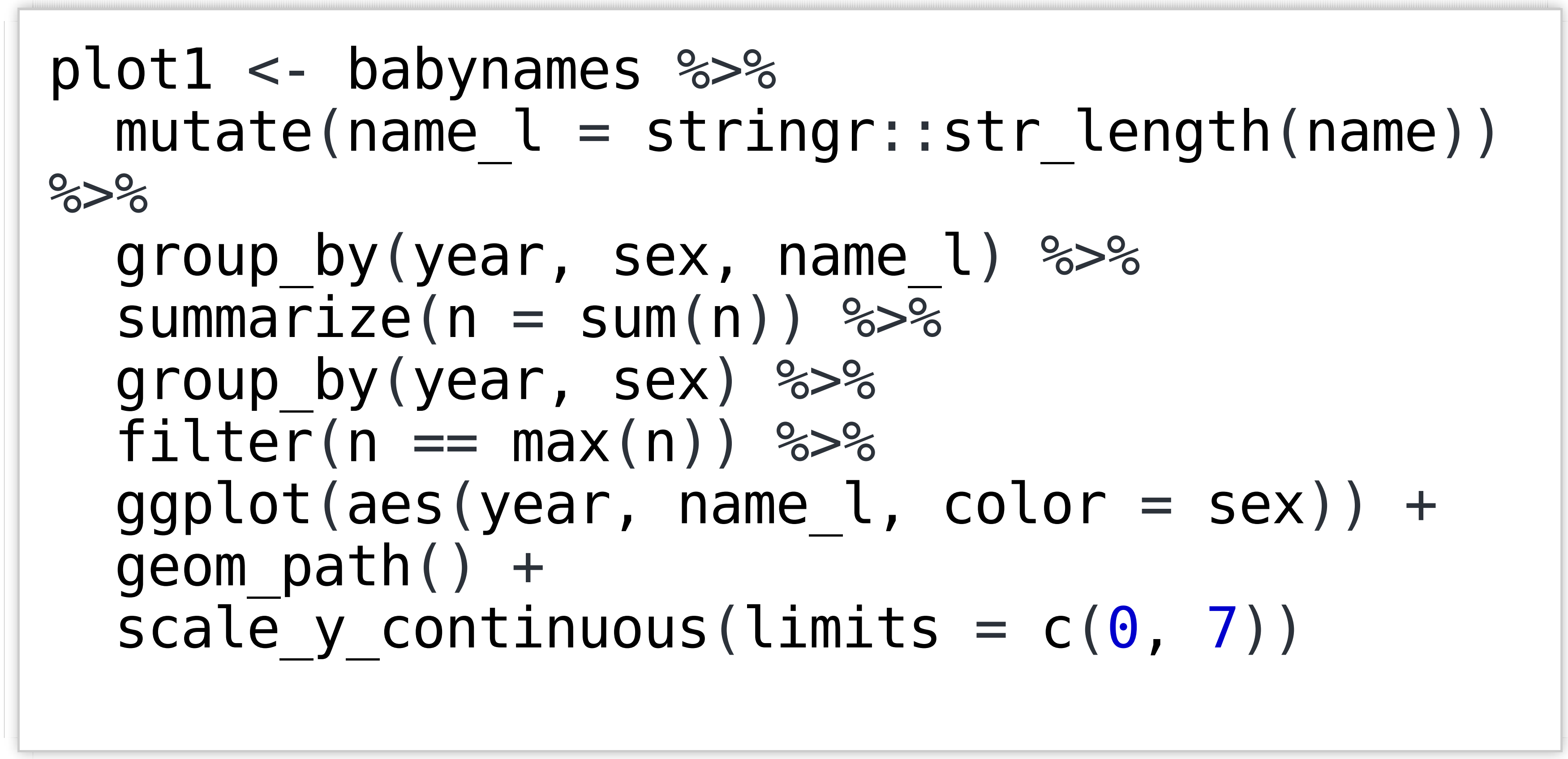
Most popular initial



```
babynames %>%
  mutate(initial = stringr::str_sub(name,
end = 1L)) %>%
  group_by(sex, initial) %>%
  summarize(total_prop = sum(prop)) %>%
  group_by(sex) %>%
  filter(total_prop == max(total_prop))
Source: local data frame [2 x 3]
Groups: sex [2]

# A tibble: 2 x 3
    sex initial total_prop
  <chr>   <chr>      <dbl>
1     F       M   17.68192
2     M       J   22.37527
```

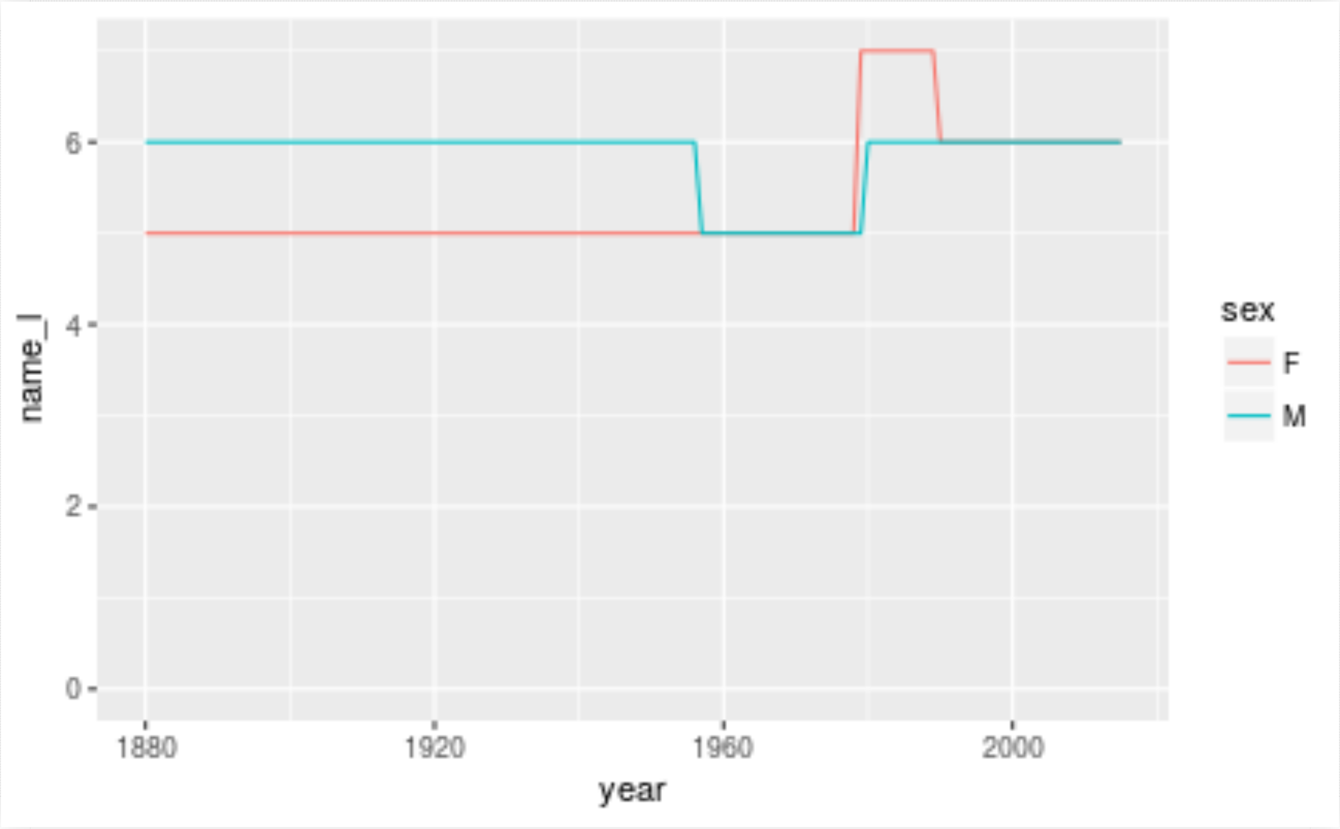
Length of name by years



```
plot1 <- babynames %>%
  mutate(name_l = stringr::str_length(name))
%>%
  group_by(year, sex, name_l) %>%
  summarize(n = sum(n)) %>%
  group_by(year, sex) %>%
  filter(n == max(n)) %>%
  ggplot(aes(year, name_l, color = sex)) +
  geom_path() +
  scale_y_continuous(limits = c(0, 7))
```

Length of name by years

```
plot1
```



Contacts

Alexander (Alex) Matrunich

a@matrunich.com

Twitter: **@matrunich**