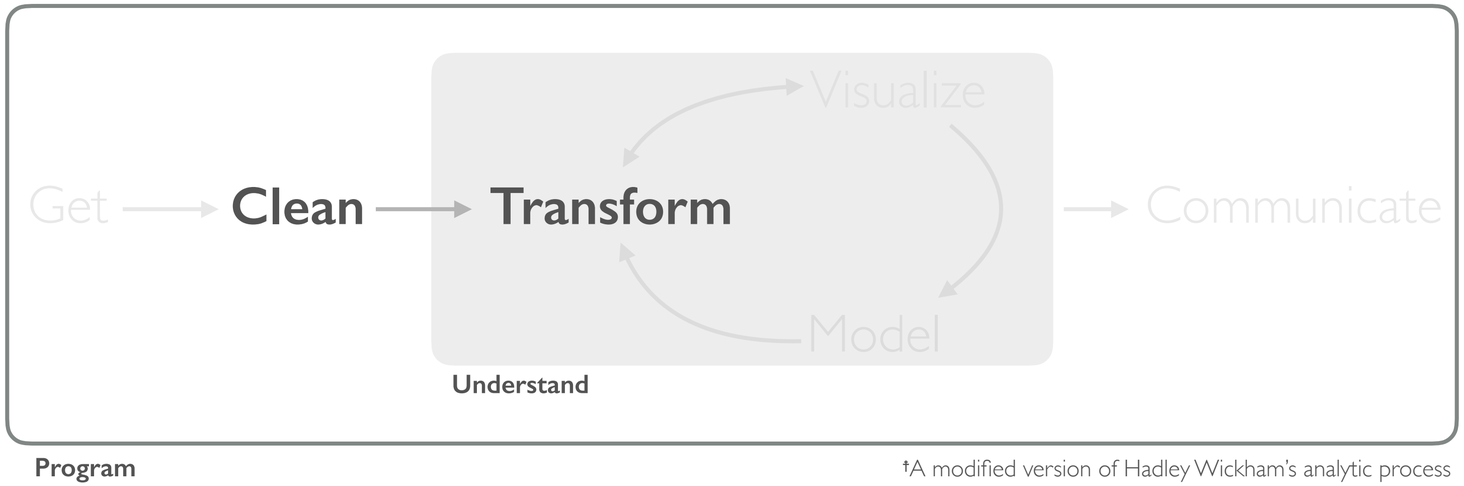
CHARACTER STRINGS



†A modified version of Hadley Wickham's analytic process

HELLO WORLD

- Sooooo… much to cover
- We're just going to focus on dealing with basic text fields in our data frames / tibbles
- Two areas of focus:
    i. basic string manipulation
    ii. regular expressions

PREREQUISITES

PREREQUISITES

- Re-start your R session
  - **Windows:** Ctrl+Shift+F10
  - **Mac:** Command+Shift+F10
- Make sure your working directory is set to the course folder

PACKAGE PREREQUISITE

```
library(stringr)
library(tidyverse)
```

Note: all relevant **`stringr`** *functions start with* **`str_`**

DATA PREREQUISITE

```
airbnb <- read_rds("data/airbnb.rds")

airbnb
# A tibble: 3,585 × 95
         id                              listing_url    scrape_id last_scraped
      <int>                                    <chr>        <dbl>       <date>
1  12147973 https://www.airbnb.com/rooms/12147973 2.016091e+13   2016-09-07
2   3075044  https://www.airbnb.com/rooms/3075044 2.016091e+13   2016-09-07
3      6976     https://www.airbnb.com/rooms/6976 2.016091e+13   2016-09-07
4   1436513  https://www.airbnb.com/rooms/1436513 2.016091e+13   2016-09-07
5   7651065  https://www.airbnb.com/rooms/7651065 2.016091e+13   2016-09-07
6  12386020 https://www.airbnb.com/rooms/12386020 2.016091e+13   2016-09-07
7   5706985  https://www.airbnb.com/rooms/5706985 2.016091e+13   2016-09-07
8   2843445  https://www.airbnb.com/rooms/2843445 2.016091e+13   2016-09-07
9    753446   https://www.airbnb.com/rooms/753446 2.016091e+13   2016-09-07
10   849408   https://www.airbnb.com/rooms/849408 2.016091e+13   2016-09-07
```

STRING BASICS

BASICS

```
airbnb %>%
  select(name) %>%
  mutate(character_count = str_count(name))
# A tibble: 3,585 × 2
                                          name character_count
                                         <chr>           <int>
1                     Sunny Bungalow in the City              26
2              Charming room in pet friendly apt              33
3               Mexican Folk Art Haven in Boston              32
4  Spacious Sunny Bedroom Suite in Historic Home              45
5                            Come Home to Boston              19
6                Private Bedroom + Great Coffee              30
7            New Lrg Studio apt 15 min to Boston              35
8             "Tranquility" on "Top of the Hill"              34
9             6 miles away from downtown Boston!              34
10             Perfect & Practical Boston Rental              33
# ... with 3,575 more rows
```

- We can use `str_count` to count the number of characters in a character field

BASICS

```
airbnb %>%
  select(name) %>%
  mutate(first_five = str_sub(name, start = 1, end = 5),
         last_five = str_sub(name, start = -5))
# A tibble: 3,585 × 3
                                          name first_five last_five
                                         <chr>      <chr>     <chr>
1                     Sunny Bungalow in the City      Sunny      City
2              Charming room in pet friendly apt      Charm     y apt
3               Mexican Folk Art Haven in Boston      Mexic     oston
4  Spacious Sunny Bedroom Suite in Historic Home      Spaci      Home
5                           Come Home to Boston      Come      oston
6                 Private Bedroom + Great Coffee      Priva     offee
7            New Lrg Studio apt 15 min to Boston      New L     oston
8             "Tranquility" on "Top of the Hill"      "Tran     Hill"
9             6 miles away from downtown Boston!      6 mil     ston!
10             Perfect & Practical Boston Rental      Perfe     ental
# ... with 3,575 more rows
```

- We can use `str_sub` with start and end arguments to take out a substring

BASICS

```
airbnb %>%
  select(host_name) %>%
  mutate(lower_case = str_to_lower(host_name),
         upper_case = str_to_upper(host_name))
# A tibble: 3,585 × 3
   host_name lower_case upper_case
       <chr>      <chr>      <chr>
1   Virginia   virginia   VIRGINIA
2     Andrea     andrea     ANDREA
3       Phil       phil       PHIL
4     Meghna     meghna     MEGHNA
5      Linda      linda      LINDA
6    Deborah    deborah    DEBORAH
7     Juliet     juliet     JULIET
8    Marilyn    marilyn    MARILYN
9       Sami       sami       SAMI
10     Damon      damon      DAMON
# ... with 3,575 more rows
```

- We can use `str_to_lower` and `str_to_upper` to normalize text case

YOUR TURN!

1. *What is the average number of characters used in the* **`name`** *column? What about the* **`description`** *column?*

2. *What is the most common name in the* **`host_name`** *column?*

SOLUTION

```
# problem 1
airbnb %>%
  select(name, description) %>%
  mutate(
    name_char = str_count(name),
    desc_char = str_count(description)
    ) %>%
  summarise(
    name_char = mean(name_char, na.rm = TRUE),
    desc_char = mean(desc_char, na.rm = TRUE)
    )
# A tibble: 1 × 2
  name_char desc_char
      <dbl>     <dbl>
1  32.34728  768.5431
```

SOLUTION

```
# problem 2
airbnb %>%
  select(host_name) %>%
  mutate(host_name = str_to_lower(host_name)) %>%
  count(host_name, sort = TRUE)
# A tibble: 1,334 × 2
   host_name     n
       <chr> <int>
1       kara   138
2   seamless    79
3       mike    71
4   flatbook    58
5     alicia    50
6      marie    42
7      jason    35
8      sarah    26
```

MATCHING BASIC PATTERNS

REGULAR EXPRESSIONS

- Regular expressions (or regexp for short) are useful because strings usually contain unstructured or semi-structured data
- Regexp provide a concise way to describe patterns in strings and
- stringr provides several functions to work with regexp

We'll start with looking at simple word matches

BASIC MATCHES

```
airbnb %>%
  select(name) %>%
  mutate(charming = str_detect(name, "charming"))
# A tibble: 3,585 × 2
                                          name charming
                                         <chr>    <lgl>
1                     Sunny Bungalow in the City    FALSE
2              Charming room in pet friendly apt    FALSE
3               Mexican Folk Art Haven in Boston    FALSE
4  Spacious Sunny Bedroom Suite in Historic Home    FALSE
5                            Come Home to Boston    FALSE
6                Private Bedroom + Great Coffee    FALSE
7            New Lrg Studio apt 15 min to Boston    FALSE
8             "Tranquility" on "Top of the Hill"    FALSE
9             6 miles away from downtown Boston!    FALSE
10             Perfect & Practical Boston Rental    FALSE
# ... with 3,575 more rows
```

- Simplest example is to identify certain specific words within text
- We can use str_detect to see if the word "charming" exists in the name

BASIC MATCHES

```
airbnb %>%
  select(name) %>%
  mutate(charming = str_detect(name, "charming"))
# A tibble: 3,585 × 2
                                        name charming
                                       <chr>   <lgl>
1                    Sunny Bungalow in the City   FALSE
2              Charming room in pet friendly apt   FALSE
3               Mexican Folk Art Haven in Boston   FALSE
4  Spacious Sunny Bedroom Suite in Historic Home   FALSE
5                           Come Home to Boston   FALSE
6                Private Bedroom + Great Coffee   FALSE
7            New Lrg Studio apt 15 min to Boston   FALSE
8             "Tranquility" on "Top of the Hill"   FALSE
9             6 miles away from downtown Boston!   FALSE
10             Perfect & Practical Boston Rental   FALSE
# ... with 3,575 more rows
```

- Simplest example is to identify certain specific words within text
- We can use str_detect to see if the word "charming" exists in the name

What happened?

BASIC MATCHES

```
airbnb %>%
  select(name) %>%
  mutate(charming = str_detect(name, ignore.case("charming")))
# A tibble: 3,585 × 2
                                          name charming
                                         <chr>    <lgl>
1                     Sunny Bungalow in the City    FALSE
2              Charming room in pet friendly apt     TRUE
3               Mexican Folk Art Haven in Boston    FALSE
4  Spacious Sunny Bedroom Suite in Historic Home    FALSE
5                            Come Home to Boston    FALSE
6                Private Bedroom + Great Coffee    FALSE
7            New Lrg Studio apt 15 min to Boston    FALSE
8             "Tranquility" on "Top of the Hill"    FALSE
9             6 miles away from downtown Boston!    FALSE
10             Perfect & Practical Boston Rental    FALSE
# ... with 3,575 more rows
```

- Simplest example is to identify certain specific words within text
- We can use str_detect to see if the word "charming" exists in the name
- Wrap with ignore.case

BASIC MATCHES

```
airbnb %>%
  select(name) %>%
  mutate(charming = str_detect(name, ignore.case("charming"))) %>%
  summarise(charming = sum(charming))
# A tibble: 1 × 1
  charming
     <int>
1       92
```

- And since logical values are numeric (TRUE = 1, FALSE = 0) we can easily count how many names have the word "charming" in it.

BASIC MATCHES

```
airbnb %>%
  select(name) %>%
  mutate(charming = str_count(name, ignore.case("charming")))
# A tibble: 3,585 × 2
                                          name charming
                                         <chr>    <int>
1                     Sunny Bungalow in the City        0
2              Charming room in pet friendly apt        1
3               Mexican Folk Art Haven in Boston        0
4  Spacious Sunny Bedroom Suite in Historic Home        0
5                            Come Home to Boston        0
6                Private Bedroom + Great Coffee        0
7            New Lrg Studio apt 15 min to Boston        0
8             "Tranquility" on "Top of the Hill"        0
9             6 miles away from downtown Boston!        0
10             Perfect & Practical Boston Rental        0
# ... with 3,575 more rows
```

- In addition to `str_detect`, you can use
  - `str_count`

BASIC MATCHES

```
airbnb %>%
  select(name) %>%
  mutate(charming = str_extract(name, ignore.case("charming")))
# A tibble: 3,585 × 2
                                          name charming
                                         <chr>    <chr>
1                     Sunny Bungalow in the City     <NA>
2              Charming room in pet friendly apt Charming
3               Mexican Folk Art Haven in Boston     <NA>
4  Spacious Sunny Bedroom Suite in Historic Home     <NA>
5                            Come Home to Boston     <NA>
6                Private Bedroom + Great Coffee     <NA>
7            New Lrg Studio apt 15 min to Boston     <NA>
8             "Tranquility" on "Top of the Hill"     <NA>
9             6 miles away from downtown Boston!     <NA>
10             Perfect & Practical Boston Rental     <NA>
# ... with 3,575 more rows
```

- In addition to `str_detect`, you can use
  - `str_count`
  - `str_extract`

BASIC MATCHES

```
airbnb %>%
  select(name) %>%
  mutate(name = str_replace(name, ignore.case("charming"), "HUGE"))
# A tibble: 3,585 × 1
                                          name
                                         <chr>
1                     Sunny Bungalow in the City
2                  HUGE room in pet friendly apt
3               Mexican Folk Art Haven in Boston
4  Spacious Sunny Bedroom Suite in Historic Home
5                            Come Home to Boston
6                 Private Bedroom + Great Coffee
7            New Lrg Studio apt 15 min to Boston
8             "Tranquility" on "Top of the Hill"
9             6 miles away from downtown Boston!
10             Perfect & Practical Boston Rental
# ... with 3,575 more rows
```

- In addition to `str_detect`, you can use
  - `str_count`
  - `str_extract`
  - `str_replace`

BASIC MATCHES

- Also, note that many `str_` functions have an `_all` companion to execute the function for **all** matching patterns in the string

```
airbnb %>%
  select(name) %>%
  mutate(name = str_replace(name, "u", "uuu"))
# A tibble: 3,585 × 1
                                                  name
                                                 <chr>
1                        Suuunny Bungalow in the City
2                  Charming room in pet friendly apt
3                   Mexican Folk Art Haven in Boston
4  Spaciouuus Sunny Bedroom Suite in Historic Home
5                                Come Home to Boston
6                     Private Bedroom + Great Coffee
7              New Lrg Stuuudio apt 15 min to Boston
```

```
airbnb %>%
  select(name) %>%
  mutate(name = str_replace_all(name, "u", "uuu"))
# A tibble: 3,585 × 1
                                                  name
                                                 <chr>
1                        Suuunny Buuungalow in the City
2                  Charming room in pet friendly apt
3                   Mexican Folk Art Haven in Boston
4  Spaciouuus Suuunny Bedroom Suuuite in Historic Home
5                                Come Home to Boston
6                     Private Bedroom + Great Coffee
7              New Lrg Stuuudio apt 15 min to Boston
```

YOUR TURN!

1. *Using the* **`house_rules`** *column, how many observations (aka hosts) advocate for "no shoes"?*

2. *How would you filter out these observations?*

SOLUTION

```
# problem 1
airbnb %>%
  select(house_rules) %>%
  mutate(no_shoes = str_detect(house_rules, ignore.case("no shoes"))) %>%
  summarise(no_shoes = sum(no_shoes, na.rm = TRUE))
# A tibble: 1 × 1
  no_shoes
     <int>
1       67

# problem 2
airbnb %>%
  filter(!str_detect(house_rules, ignore.case("no shoes”)))
# A tibble: 2,326 × 95
         id                              listing_url    scrape_id last_scraped
      <int>                                    <chr>        <dbl>       <date>
1  12147973 https://www.airbnb.com/rooms/12147973 2.016091e+13   2016-09-07
2   3075044  https://www.airbnb.com/rooms/3075044 2.016091e+13   2016-09-07
3      6976     https://www.airbnb.com/rooms/6976 2.016091e+13   2016-09-07
```

USING ANCHORS

ANCHORS

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "^A"))
# A tibble: 373 × 1
         host_name
             <chr>
1            Andrea
2           Anthony
3            Ashley
4            Alexis
5               Anú
6  Alison (& Shawn)
7            Ashley
8            Andrew
9            Andrea
10             Anya
# ... with 363 more rows
```

- We can use anchors to:
  - ^: match the start of a string

ANCHORS

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "z$"))
# A tibble: 9 × 1
   host_name
       <chr>
1 Hasan Oguz
2        Liz
3        Liz
4        Luz
5        Liz
6 Hasan Oguz
7   Sergiusz
8   Sergiusz
9        Rez
```

- We can use anchors to:
  - ^: match the start of a string
  - $: match the end of a string

YOUR TURN!

What is the most common host_name that starts with a "B" and ends with a "y"?

SOLUTION

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "^B") & str_detect(host_name, "y$")) %>%
  count(host_name, sort = TRUE)
# A tibble: 7 × 2
  host_name     n
      <chr> <int>
1     Billy     3
2     Becky     2
3     Bobby     2
4  Brittany     2
5     Betsy     1
6   Beverly     1
7     Brady     1
```

CHARACTER CLASSES & ALTERNATIVES

SPECIAL PATTERNS

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "a.a"))
# A tibble: 251 × 1
   host_name
       <chr>
1    Mariana
2    Mattaya
3      Sarah
4      Sarah
5       Cara
6      Sarah
7      Sarah
8    Mariana
9      Edana
10   Vinayak
# ... with 241 more rows
```

- We can use several regexp to match special patterns. This includes:
  - `.`: match any character

SPECIAL PATTERNS

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "\\d"))
# A tibble: 1 × 1
    host_name
        <chr>
1 40 Berkeley
```

- We can use several regexp to match special patterns. This includes:
  - `.`: match any character
  - `\\d`: match any digit

SPECIAL PATTERNS

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "\\s"))
# A tibble: 266 × 1
                      host_name
                          <chr>
1                   Carl & Judy
2             Nelson And Marlene
3                   Gena & Leon
4                 Maria Cecilia
5                   Ella & Will
6               Alison (& Shawn)
7                 Katie And Joe
8  L&B (Len & Becky Or Blenky*)
9                 Caitlin & Dan
10                  Maria Elena
# ... with 256 more rows
```

- We can use several regexp to match special patterns. This includes:
  - `.`: match any character
  - `\\d`: match any digit
  - `\\s`: match any white space

SPECIAL PATTERNS

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "[aeiou]"))
# A tibble: 3,509 × 1
   host_name
       <chr>
1   Virginia
2     Andrea
3       Phil
4     Meghna
5      Linda
6    Deborah
7     Juliet
8    Marilyn
9       Sami
10     Damon
# ... with 3,499 more rows
```

- We can use several regexp to match special patterns. This includes:
  - `.`: match any character
  - `\\d`: match any digit
  - `\\s`: match any white space
  - `[aeiou]`: match any of these

Can also include numbers `[0-9]`

SPECIAL PATTERNS

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "[^aeiou]$"))
# A tibble: 2,158 × 1
     host_name
         <chr>
1         Phil
2      Deborah
3       Juliet
4      Marilyn
5        Damon
6        Megan
7      Anthony
8        Megan
9      Mohamed
10 Carl & Judy
# ... with 2,148 more rows
```

adding $ at the end looks for all names that do not end with a vowel

- We can use several regexp to match special patterns. This includes:
  - `.`: match any character
  - `\\d`: match any digit
  - `\\s`: match any white space
  - `[aeiou]`: match any of these
  - `[^aeiou]`: does **not** match any of these

SPECIAL PATTERNS

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "^(Ch|K)ris"))
# A tibble: 62 × 1
          host_name
              <chr>
1             Chris
2           Kristen
3             Chris
4         Christine
5  Chris & Kristina
6         Christine
7  Chris & Kristina
8   Christina Marie
9         Christina
10          Kristin
# ... with 52 more rows
```

Starts with either "Chris" or "Kris"

- We can use several regexp to match special patterns. This includes:
  - `.`: match any character
  - `\\d`: match any digit
  - `\\s`: match any white space
  - `[aeiou]`: match any of these
  - `[^aeiou]`: does **not** match any of these
  - `(Ch|K)`: match either or

YOUR TURN!

1. *What is the most common host_name that:*
    1. *neither starts nor ends with a vowel?*
    2. *ends with either "ie" or "ey"*
    3. *has no vowels in it*

SOLUTION

```
# Problem 1
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "^[^AEIOU]") &
         str_detect(host_name, "[^aeiou]$")) %>%
  count(host_name, sort = TRUE)
# A tibble: 623 × 2
     host_name     n
         <chr> <int>
1     Seamless    79
2     Flatbook    58
3        Jason    35
4        Sarah    26
5         Will    26
6  Stay Alfred    25
7         Todd    25
8     Jonathan    23
9        David    21
```

SOLUTION

```
# Problem 2
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "(ie|ey)$")) %>%
  count(host_name, sort = TRUE)
# A tibble: 61 × 2
   host_name     n
       <chr> <int>
1      Marie    42
2     Ashley    11
3      Julie     8
4     Bernie     6
5  Stephanie     6
6    Jeffrey     5
7     Bonnie     4
8      Katie     4
9     Barrie     3
```

SOLUTION

```
# Problem 3
airbnb %>%
  select(host_name) %>%
  filter(!str_detect(host_name, "[AEIOUaeiou]")) %>%
  count(host_name, sort = TRUE)
# A tibble: 19 × 2
   host_name     n
       <chr> <int>
1       S.P.    11
2         Jp     3
3         Fj     2
4          J     2
5         J.     2
6         Jj     2
7        Lyn     2
8       Lynn     2
9         Vj     2
10      公主     2
```

REPETITION

SPECIAL PATTERNS

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "[AEIOUaeiou]{3}"))
# A tibble: 11 × 1
   host_name
       <chr>
1      Louis
2      Louis
3       Caio
4      Louis
5     Louise
6       Iain
7      Louie
8      Maiah
9      Maiah
10   Feibiao
11      Euan
```

names that have 3 vowels in a row

- There are special characters to control matching repeated patterns. This includes:
  - `{n}`: exactly n

SPECIAL PATTERNS

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "[A-Za-z]{10,}"))
# A tibble: 46 × 1
       host_name
           <chr>
1  JonandMargrit
2  JonandMargrit
3    SleepAfloat
4   Konstantinos
5    SleepAfloat
6    SleepAfloat
7    SleepAfloat
8    SleepAfloat
9    Christopher
10   Christopher
# ... with 36 more rows
```

names that have 10 or more letters

- There are special characters to control matching repeated patterns. This includes:
  - `{n}`: exactly n
  - `{n,}`: n or more

Note that `+` is shorthand for 1 or more

SPECIAL PATTERNS

```
airbnb %>%
  select(host_name) %>%
  filter(str_detect(host_name, "[aeiou]{3,4}$"))
# A tibble: 3 × 1
  host_name
      <chr>
1      Caio
2     Louie
3   Feibiao
```

names that end with 3 or 4 vowels

- There are special characters to control matching repeated patterns. This includes:
  - `{n}`: exactly n
  - `{n,}`: n or more
  - `{n,m}`: between n and m

CHALLENGE

CHALLENGE

*What is the most commonly used first word for names (**name** column)*

SOLUTION

```
airbnb %>%
  select(name) %>%
  mutate(first_word = str_extract(name, "^[A-Za-z0-9]+")) %>%
  count(first_word, sort = TRUE)
# A tibble: 656 × 2
   first_word     n
        <chr> <int>
1        Cozy   179
2     Private   165
3   Beautiful   117
4        <NA>   117
5    Spacious   110
6         Lux   101
7       Sunny    87
8      Boston    84
9    Charming    79
```

WHAT TO REMEMBER

FUNCTIONS TO REMEMBER

Operator/Function	Description
`str_count, str_detect, str_extract, str_replace, str_sub`	parsing functions to count, identify, extract, and replace regular expressions
`str_to_lower, str_to_upper`	normalizing text case
`^, $`	anchors to match regexp at start and end of strings
`., \\d, \\s, [a-z0-9], [^a-z0-9]`	special character classes and alternatives to identify regexp
`{n}, {n,}, +, {n,m},`	arguments to identify repetitions of regexp