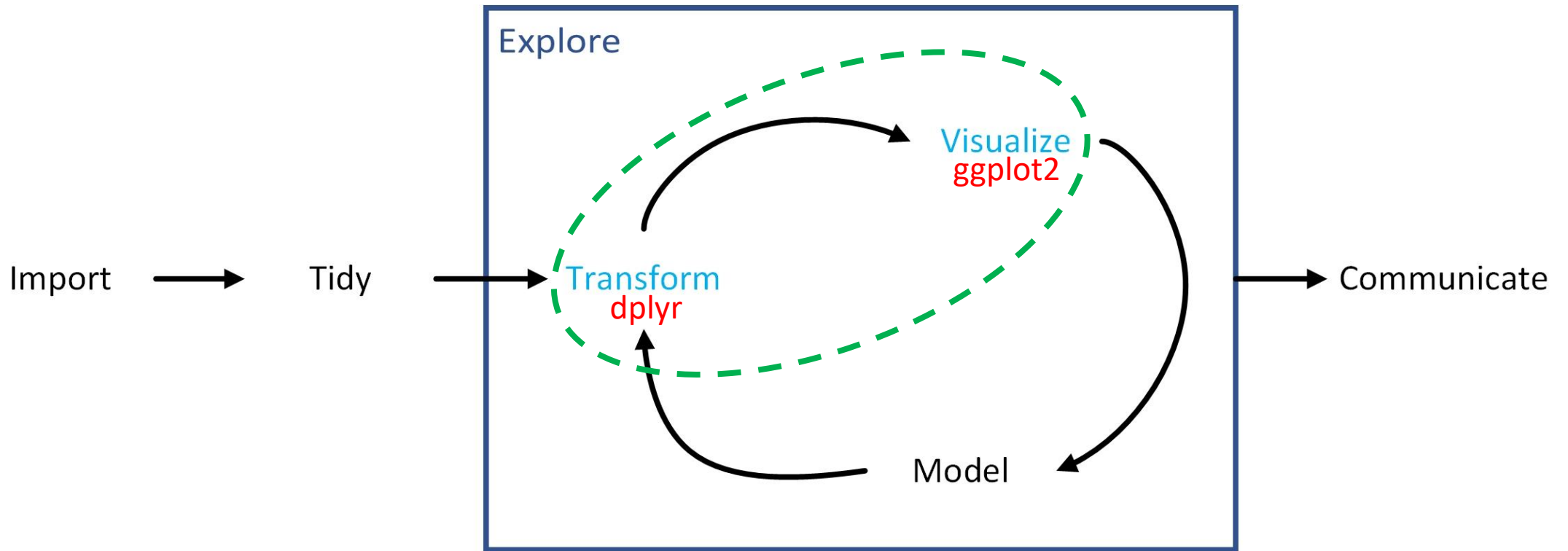


Data Transformation with dplyr & stringr + ggplot2

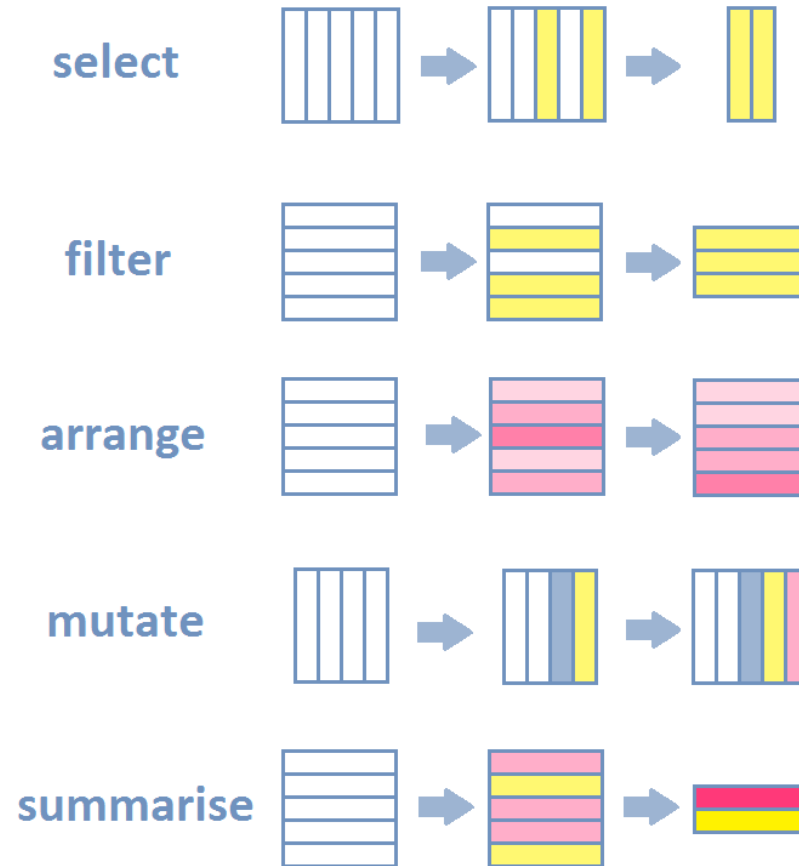
CMB 710 Tidybiology

Junqi Lu

ggplot2 Works with dplyr for Data Communication



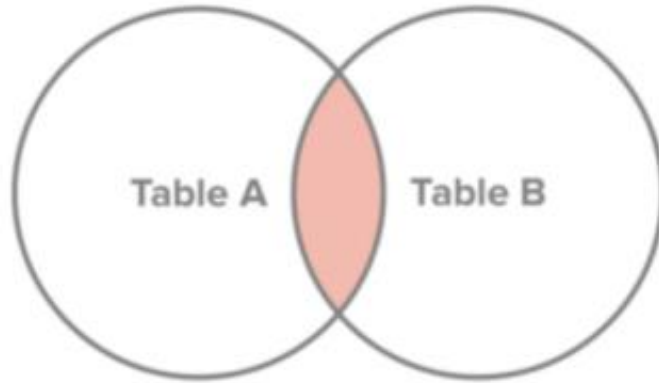
Quick Review of Basic dplyr



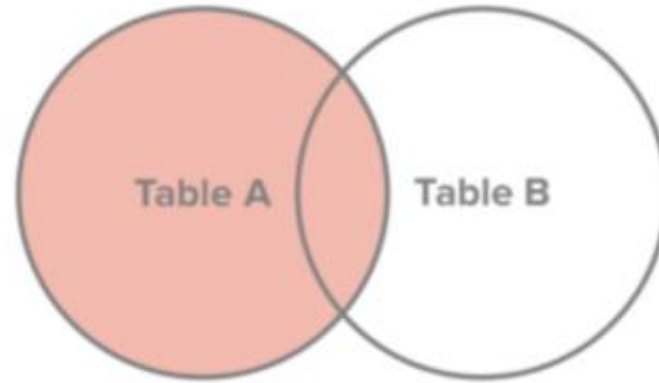
<http://perso.ens-lyon.fr/lise.vaudor/dplyr/>

Graphical Review of 4 join()

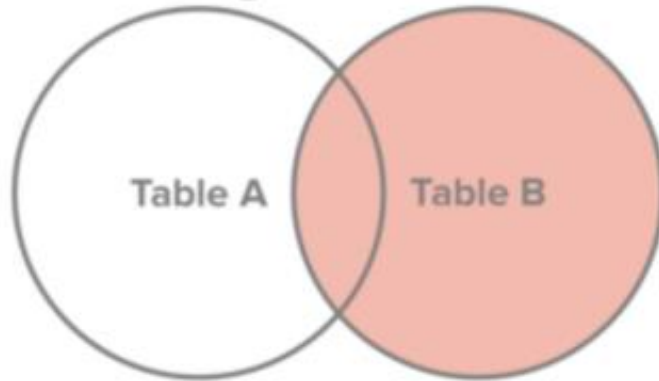
Inner Join



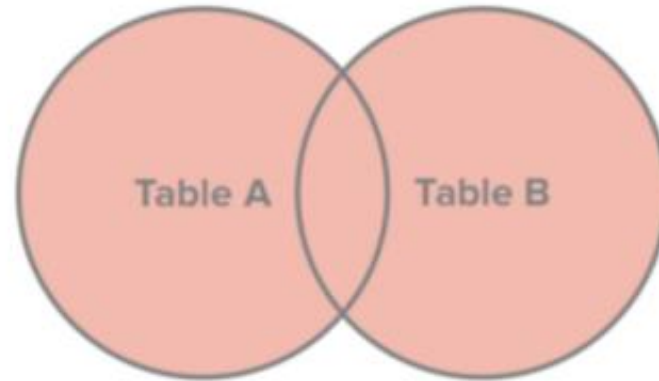
Left Join



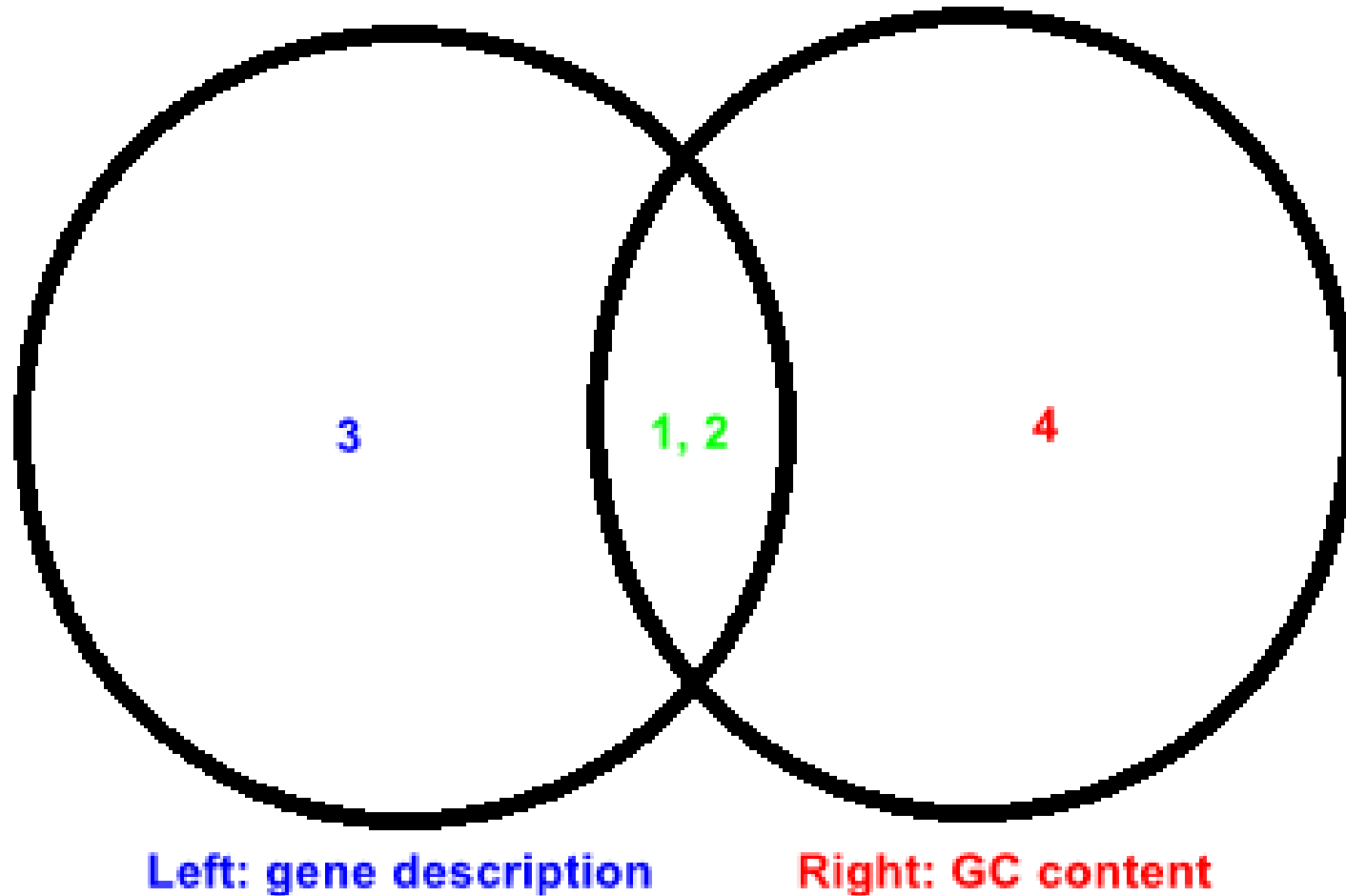
Right Join



Full Join

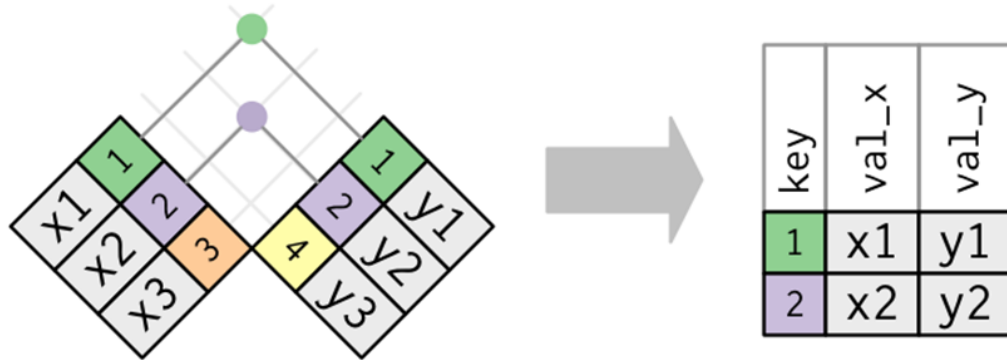


Dummy DataFrames for join() Revisit

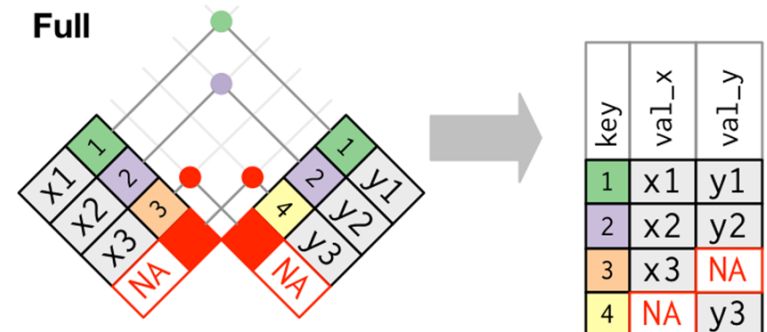
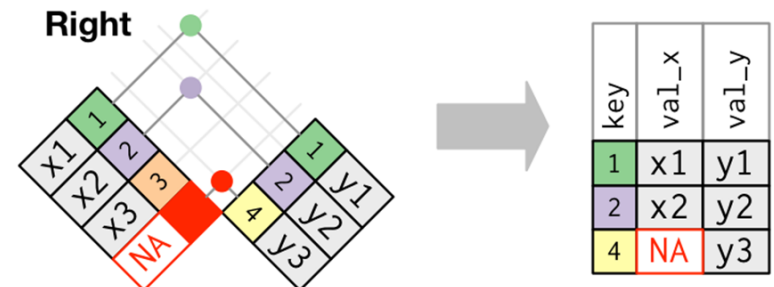
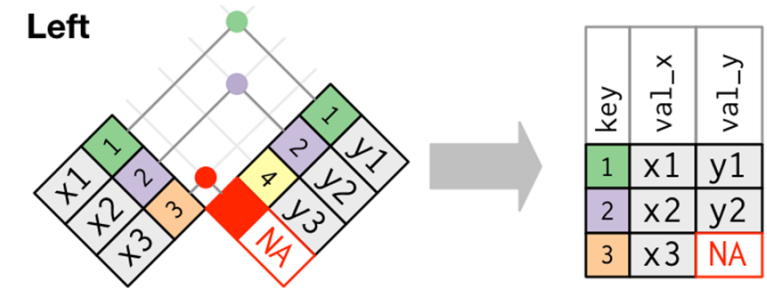


Graphical Review of 4 Mutating join()

Inner join

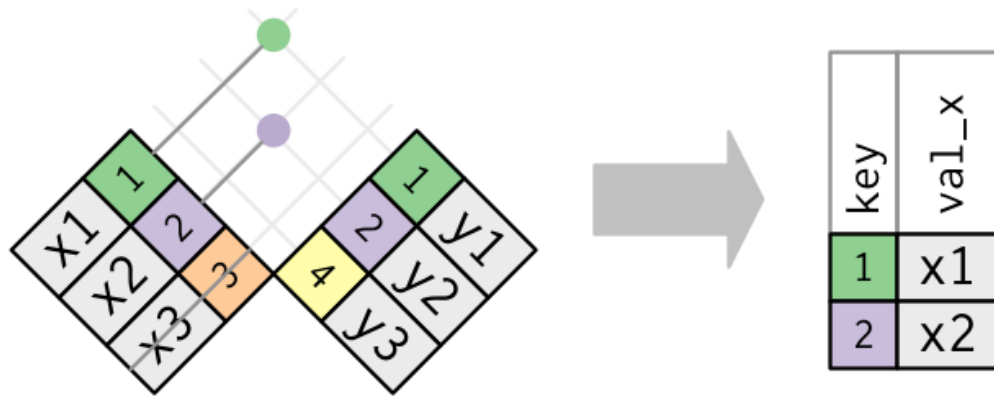


Outer join

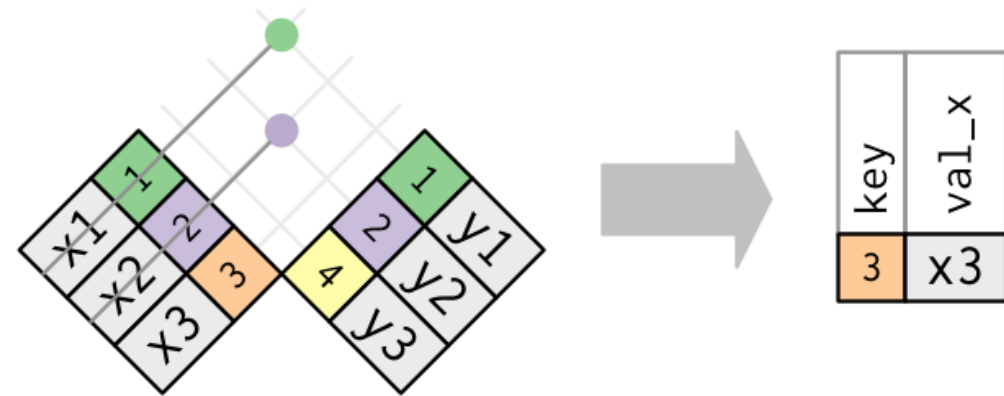


2 More Filtering join()

semi_join(x, y) keeps all observations in x that have a match in y



anti_join(x, y) drops all observations in x that have a match in y.



DataFrames Binding/Assembly

bind_rows() faster than **rbind()**

bind_cols() faster than **cbind()**

X

A	B	C
a	t	1
b	u	2
c	v	3

+

Z

A	B	C
c	v	3
d	w	4

=

DF	A	B	C
x	a	t	1
x	b	u	2
x	c	v	3
z	c	v	3
z	d	w	4

X

A	B	C
a	t	1
b	u	2
c	v	3

+

y

A	B	D
a	t	3
b	u	2
d	w	1

=

A	B	C	A	B	D
a	t	1	a	t	3
b	u	2	b	u	2
c	v	3	d	w	1

Data Transformation on the Run for ggplot2

- Data filtering by df subsetting
 - Syntax: `ggplot(df[JUDGEMENT(df$col_name),])+`
- Conditional styling by `ifelse()`
 - Syntax: `...styling_parameter=ifelse(JUDGEMENT(df$col_name), ValuelfTrue, ValuelfFalse)...`
- If used in combination, need to tell `ifelse()` that df has changed
 - Method 1
 - `NewDf <- df[JUDGEMENT(df$col_name),]`
 - `ggplot(NewDf)+`
 - `...styling_parameter=ifelse(JUDGEMENT(NewDf$col_name), ValuelfTrue, ValuelfFalse)...`
 - Method 2
 - `ggplot(df[JUDGEMENT(df$col_name),])+`
 - `...styling_parameter=ifelse(JUDGEMENT(df[JUDGEMENT(df$col_name),]$col_name), ValuelfTrue, ValuelfFalse)...`

String Processing: Stringr \approx Shape Sorter



Stringr Pattern Design

Meaning	Syntax	Example pattern	Example result
Or	$C_1 C_2$	<code>str_view_all('abcdefg','bc f')</code>	abc bc defg
One of the collection	$[C_1C_2\dots]$	<code>str_view_all('abcdefg','[bdf]')</code>	ab cd efg
Anything but	$[^C_1C_2\dots]$	<code>str_view_all('abcdefg','[^bdf]')</code>	ab cd efg
Range	$[C_1-C_2]$	<code>str_view_all('abcdefg','[b-f]')</code>	abc defg
Start of	C	<code>str_view_all(c('abc','def'),'^a')</code>	abc def
End of	$C\$$	<code>str_view_all(c('abc','def'),'f\$')</code>	abc def
0 or 1	$C?$	<code>str_view_all('loooloolol','o?')</code>	_loool o oolol_
0 or more	C^*	<code>str_view_all('loooloolol','o*')</code>	_loool oolool _
1 or more	C^+	<code>str_view_all('loooloolol','o+')</code>	loool oolool
N or more	$C\{N,\}$	<code>str_view_all('loooloolol','o{2,})')</code>	loool oolool

Stringr Pattern Tester

- Pattern explanation with diagram: <https://extendsclass.com/regex-tester.html#python>

Regular expression to test

`/^lo{2,}[bd]e$/gm` Flags +

String to test Generate a string from RegEx (Beta)

```
1 looolool
2 lool
3 abcde
4 abbe
5 abce
6
```

Explanation

RegExp: `/^lo{2,}[bd]e$/gm`

The diagram illustrates the structure of the regular expression `/^lo{2,}[bd]e$/gm`. It starts with a green dot representing the start of the string, followed by a purple box labeled `LineBegin!`. This is followed by a light blue box with the letter `l`. Then, a red circle containing the letter `o` is shown with a red line looping back to itself, labeled `2 or more times`. Below this, a bracket indicates `One of:` followed by two yellow boxes labeled `bd`. This is followed by a light blue box with the letter `e`, and finally a purple box labeled `LineEnd!`. The diagram ends with a grey dot representing the end of the string.

- Pattern explanation with texts: <https://regex101.com/>

Realize Ideas to Graphs with dplyr & ggplot2

- Interpret the questions → cheat sheets / Stack Overflow / Google → ggplot2 explorer → finalize details
- Filtering a dataset on-the-run in ggplot2 by subset()