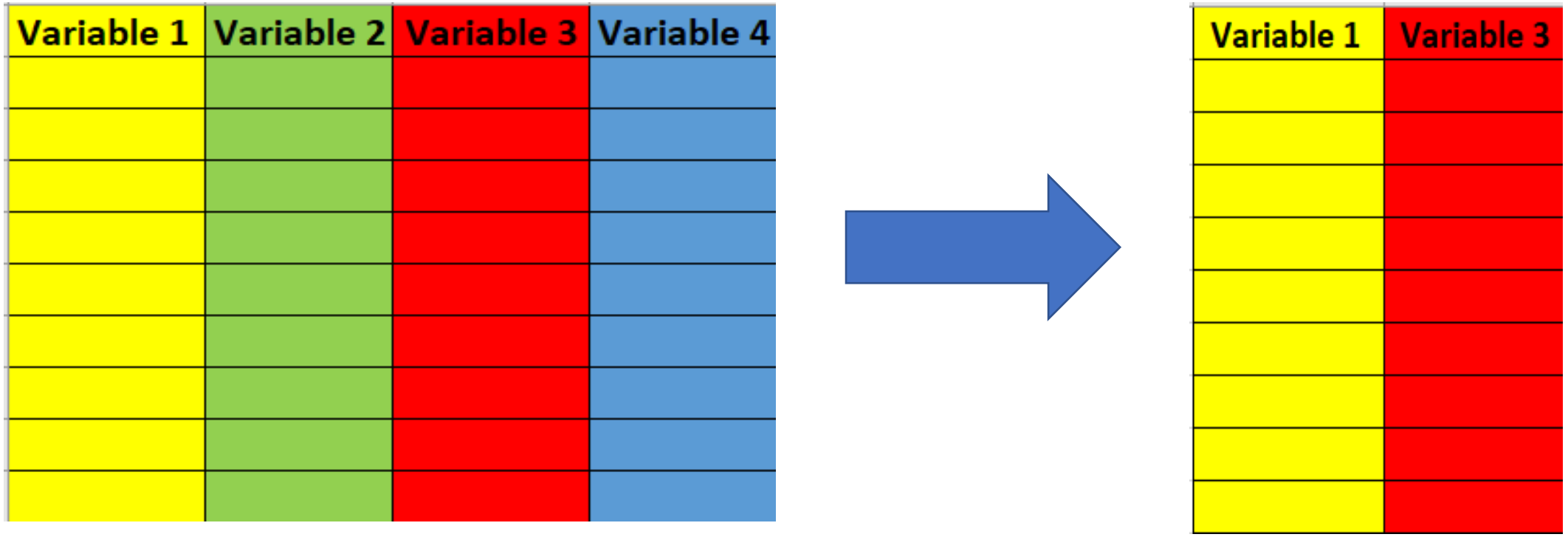
DSCI 101 – Midterm Review

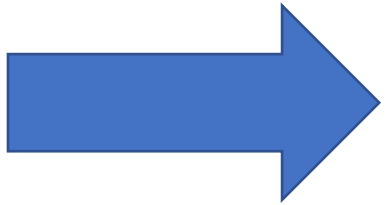
Data Wrangling – select()



Descriptions	Codes
Choose specific column name from the data frame	`new_data <- data %>%` `  select(Variable1, Variable3)`
Select Columns by Name Patterns: use special helper functions like `starts_with()`, `ends_with()`, `contains()`, `matches()`, and `everything()` to select columns based on their names.	
Exclude Columns: To exclude specific columns, you can use the "-" (minus) sign before the column name.	`new_data <- data %>%` `  select(-Variable2,-Variable4)`
Select Columns by Index and Range:	`new_data <- data %>%` `  select(c(1,3))`

Data Wrangling – filter()

Variable 1	Variable 2	Variable 3	Variable 4



Variable 1	Variable 2	Variable 3	Variable 4

Descriptions	Codes

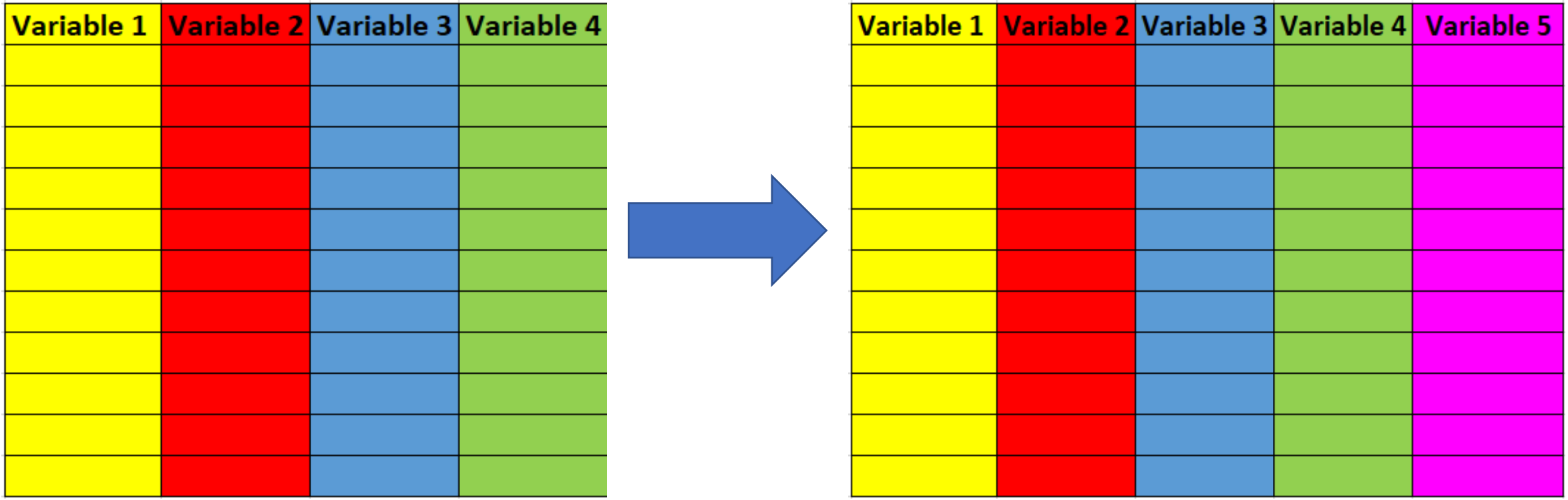
Simple Conditions - greater than (`>`), less than (`<`), or equal to (`==`)	`new_data <- data %>%`
	`  filter(Variable1 > 30)`
	
	`new_data <- data %>%`
	`  filter(Variable2 == "yes")`

Multiple Conditions - You can combine conditions using logical operators like `&` (AND) and `|` (OR).	`new_data <- data %>%`
	`  filter(Variable1 > 30 &`
	`Variable2 < 50000)`
	
	`new_data <- data %>%`
	`  filter(Variable1 > 30 |`
	`Variable2 < 50000)`

Exclusion - to exclude certain rows, you can use the `!=` operator (not equal to).	`new_data <- data %>%` `  filter(Variable4 != 30)`

Filter rows based on vector of conditions - The `%in%` operator is useful for filtering rows with values in a specified vector.	`new_data <- data %>%`
	`  filter(Variable3 %in%`
	`c("bananas", "grapes"))`

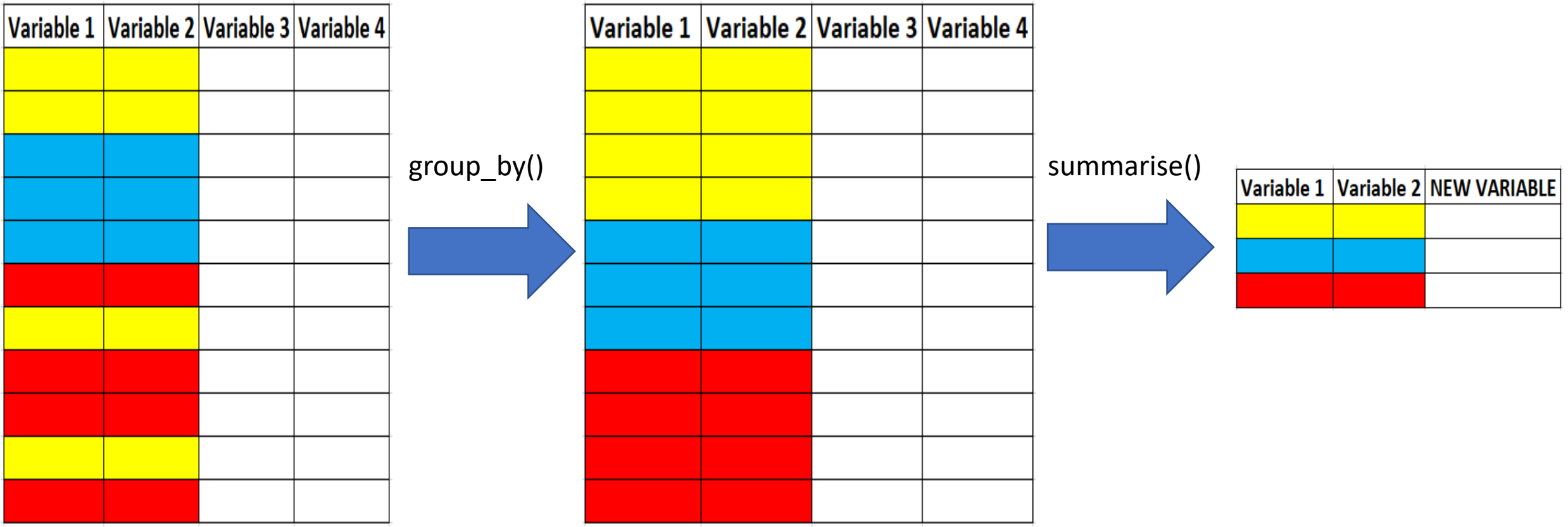
Data Wrangling – mutate()



Descriptions	Codes
Create a new variable calculated from another variable	`new_data <- data %>%` `  mutate(Variable5 = Variable2 / 12)`
We can use other functions within mutate to help us make the new variable like `ifelse()`.	`new_data <- data %>%` `  mutate(Variable5 = ifelse(Variable1 > 30, "Old", "Young"))`

If you have multiple condition you can use the `case_when()` function and list out your possible options.	`new_data <- data %>%`
	`  mutate(Variable5 = case_when(`
	`  Variable1 < 30 ~ "Young",`
	`  Variable1 >= 30 & Variable1 <= 50 ~ "Middle-aged",`
	`  Variable1 > 50 ~ "Old",`
	`  TRUE ~ "No Category"`
	`  ))`

Data Wrangling – summarise() & group_by



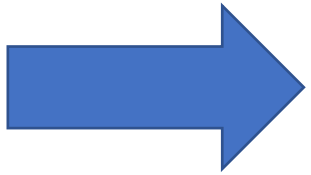
Descriptions	Codes

Compute different statistics (mean, median..etc) grouped by different variables.	`new_data <- data %>%`
	`group_by(Variable1, Variable2) %>%`
	`summarise(NEW_VARIABLE1 = mean(Variable1),`
	`NEW_VARIABLE2 = median(Variable2))`

If you do not group_by, it will make a computation on the entire column:	`new_data <- data %>%`
	`summarise(NEW_VARIABLE1 = mean(Variable1),`
	`NEW_VARIABLE2 = median(Variable2))`

Data Wrangling – arrange()

Variable 1	Variable 2	Variable 3	Variable 4

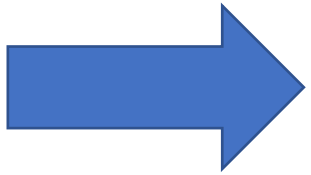


Variable 1	Variable 2	Variable 3	Variable 4

Descriptions	Codes
Arrange in ascending order (smallest to largest)	`new_data <- data %>%` `arrange(Variable1)`
Arrange in descending order (largest to smallest)	`new_data <- data %>%` `arrange(desc(Variable1))`

Data Wrangling – count()

Variable 1	Variable 2	Variable 3	Variable 4

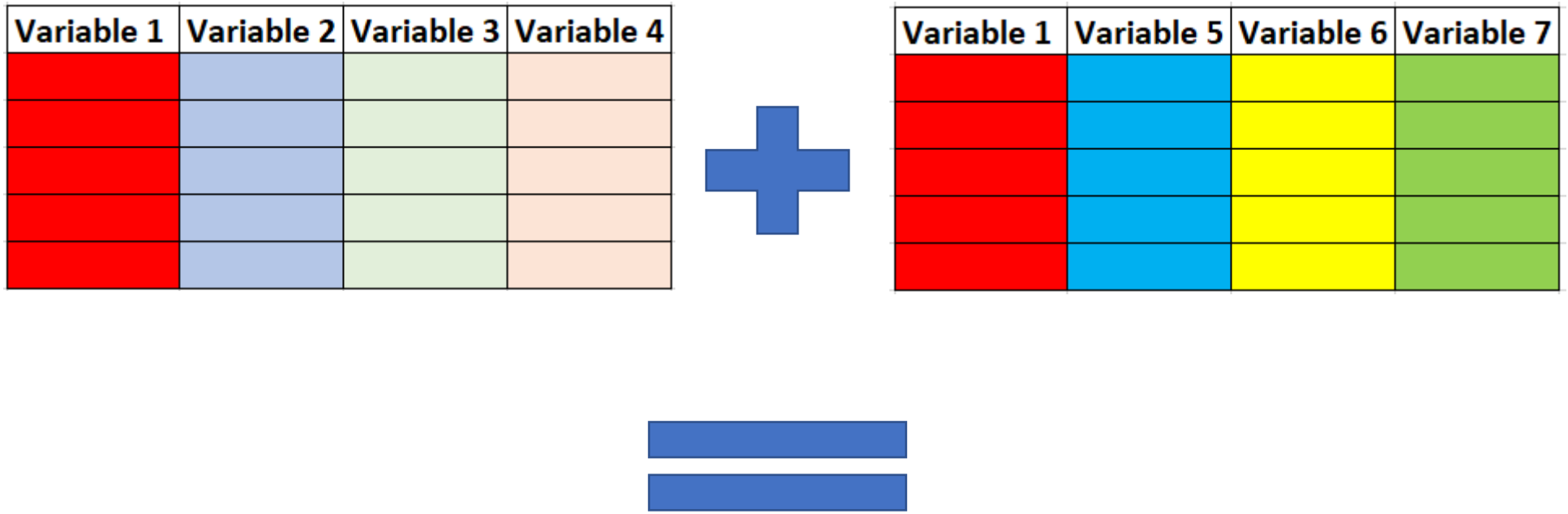


Variable 1	n
	2
	3

Descriptions	Codes
Count the occurrences of each "category"	`new_data <- data %>%` `count(Variable1)`

Can achieve the same thing as count using summarize. With summarize you can name the new variable whatever you want. With count, it automatically calls it “n”.	`new_data <- data %>%`
	`group_by(Variable1) %>%`
	`summarise(NEW_VARIABLE = n())`

Joining Datasets



Joining Datasets – inner_join()

Variable 1	Variable 2	Variable 3	Variable 4
#			
*			
&			
@			
%			

Variable 1	Variable 5	Variable 6	Variable 7
#			
*			
&			
^			
!			

Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable 6	Variable 7
#						
*						
&						

```
new_data <- left_data %>%
    inner_join(right_data, by = c("Variable1" = "Variable1"))
```

Joining Datasets – left_join()

Variable 1	Variable 2	Variable 3	Variable 4
#			
*			
&			
@			
%			

Variable 1	Variable 5	Variable 6	Variable 7
#			
*			
&			
^			
!			

Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable 6	Variable 7
#						
*						
&						
@				NA	NA	NA
%				NA	NA	NA

```
new_data <- left_data %>%
    left_join(right_data, by = c("Variable1" = "Variable1"))
```

Joining Datasets – right_join()

Variable 1	Variable 2	Variable 3	Variable 4
#			
*			
&			
@			
%			

Variable 1	Variable 5	Variable 6	Variable 7
#			
*			
&			
^			
!			

Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable 6	Variable 7
#						
*						
&						
^	NA	NA	NA			
!	NA	NA	NA			

```
new_data <- left_data %>%
      right_join(right_data, by = c("Variable1" = "Variable1"))
```

Joining Datasets – full_join()

Variable 1	Variable 2	Variable 3	Variable 4
#			
*			
&			
@			
%			

+

Variable 1	Variable 5	Variable 6	Variable 7
#			
*			
&			
^			
!			

=

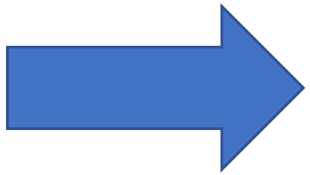
Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable 6	Variable 7
#						
*						
&						
@				NA	NA	NA
%				NA	NA	NA
^	NA	NA	NA			
!	NA	NA	NA			

```
new_data <- left_data %>%
        full_join(right_data, by = c("Variable1" = "Variable1"))
```

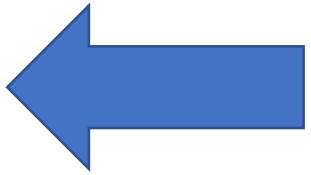
Pivot – pivot_longer() & pivot_wider

Pivot_Variable	Variable1	Variable2	Variable3

pivot_longer()



pivot_wider()



Pivot_Variable	NEW_VARIABLE	NEW_VARIABLE2
	Variable1	
	Variable1	
	Variable1	
	Variable2	
	Variable2	
	Variable2	
	Variable3	
	Variable3	
	Variable3	

```
new_data <- data %>%
pivot_longer(cols = -Pivot_Variable,
names_to = “NEW_VARIABLE",
values_to = “NEW_VARIABLE2")
```

```
new_data <- data %>%
pivot_wider(names_from = NEW_VARIABLE,
values_from = NEW_VARIABLE2)
```

Data Visualization – ggplot()

- **Data:** The dataset you're working with.
- **Aesthetics Mapping (aes):** How data variables map to plot aesthetics like position, color, shape, etc.

$$x = variable1$$
$$y = variable2$$
$$color = variable3$$
$$fill = variable4$$
$$shape = variable5$$

Data Visualization – ggplot()

- **Geometric Objects (geom):** The visual elements to represent the data (points, lines, bars, etc.).
    - geom_histogram()
    - geom_density()
    - geom_boxplot()
    - geom_line()
    - geom_bar()
    - geom_point()

Data Visualization – ggplot()

- **Facets (facet_wrap or facet_grid):** Splitting data into subplots based on a variable.

```
facet_wrap(~ Variable1)
facet_grid(Variable1 ~ Variable2)
```

Data Visualization – ggplot()

- **Theme:** Controlling the overall appearance of the plot.

  theme_gray()
  theme_bw()
  theme_minimal()
  theme_void()

Data Visualization – ggplot()

- **Add ons**

```
coord_flip()
geom_smooth(method = "lm", se = FALSE)
geom_smooth(method = "loess", se = TRUE)
scale_fill_manual(values = c("red", "green", "blue"))
scale_color_manual(values = c("red", "green", "blue"))
labs()
```

For loops & If/Else Statements

```
x<-NA
y<-NA

for (i in 1:5) {
      y[i] <- 1+i

      if (y[i]>3) {
            x[i] <- "Yes"
      }else if (y[i]<3){
            x[i] <- "No"
      }else{
            x[i]<-"Maybe"
      }
}
y
x
```

Functions

```
circumference <- function(r){
  c = 2*pi*r
  return(c)
}


circumference(5)
circumference(r = 5)
```