Week 3
1/3

Tidy data:-
A way of standardizing the organization of data within R.

Date 18/03/23

Data Frame :- Collection of columns.

- columns should be named
- Data stored can be many different types, like numeric, factor, or character.
- Each column should contain same number of data items.

* Tibbles : streamlined data frames

- Never change data types of the inputs
- Never change names of your variables
- Never create row names
- make printing easier

pulls only first 10 rows

* Tidy data standars :-
- Variables are organized into columns
- Observations are organized into rows
- Each value must have its own cell

——x——x——x——

```
install.packeeges ("tidyverse")
library (ggplot2)
data ("diamonds)
View (diamonds)
head (diamonds)     # first 6 rows.
str (diamonds) # High level info col_name & type
colnames (diamonds) # only names
mutate (diamonds, carat_2 = carat_2 * 100)
                new column created.
```

library(tidyverse)

from dplyr] →

glimpse() & str()
↘ Summary

Data Frames Data Analysts default way of interacting with data.

2/3

Cleaning up with the basics :→

3 Packages Here, Skimr, Janitor

Here :→ makes Referencing easier.
Skimr :→ Summaring data is easy & skim quickly.
Janitor :→ functions for cleaning data.

```
pacman :: p_load (pacman, here, skimr, janitor, dplyr)
```

↓
Saved 10 lines

Palmer Penguin Package : 3 species, size, clutch size, blood isotope ratio.

```
install.packages ("palmerpenguins")
library (palmerpenguins)
```

* skim_without_charts()   glimpse()
* head()   select()

```
> skim_without_charts(penguins)      # Gives summary
> glimpse (penguins)
> head (penguins)
> penguins %>%
      select (species)        # only species column

> penguins %>%              # Everything but species.
      select (-species)
```

* Rename Column

```
penguins %>%
        rename(island_new = island)
```

* rename_with(penguins, tolower)
↓
for consistency.

* Clean Names : Consistents, no duplicate, etc

clean_names(penguins)

—x—x—

Operators ↳

Arithmetic

+ :	Add	%%	Modulus
- :	Sub	%/%	Integer Div
* :	Mul	^	Exponent
/ :	Div		

—x—

Relational

> →	greater than	==	equal to
< →	less than	!=	not equal to
>= →	greater than equal		
<= →	less than equal		

—x—x—

Logical

& → Element-wise logical AND
&& → logical AND
| → Element wise logical OR
|| → logical OR
! → logical Not

```
x <- c(3,5,7)
y <- c(2,4,6)

x<5 & y<5
# T F F
x<5 && y<5
True
```

Assignment Operator

```
<-    Leftward  }
<<-   Leftward  } Assingment
=     Leftward  }

->    } Rightword
->>   } Assignment
```

* Organizing data :- arrange()
group-by()
filter()

sort by column

```
penguins %>% arrange(bill_length_mm)  #Ascending
penguins %>% arrange(-bill_length_mm)  #Descending
```

These results are only in the console to save create a new dataframe

```
penguins2 <- penguins %>% arrange(bill_length_mm)
view(penguins2)
```

Group By

```
penguins %>% group_by(island) %>% drop_na()
         %>% summarize(mean_bill_length_mm =
                       mean(bill_length_mm))
```

```
penguins %>% groupby(species, island) %>%
dropna() %>% summarize(max_bl = max(bill_length_mm),
                       mean_b = mn(bill_length_mm))
```

Filter

```
penguins %>% filter(species == "Adelie")
```

Week 4 : Visualization in R

$\cancel{y}_3$

(*) R let's you move back and forth betn analysis & visualization quickly.

> Plotly → wide Range; General Purpose

> RGL :→ Specific, 3D

> Lattice.

> Diagraphs

> leaflet

> Highcharter

> Patchwork

> gganimate

> gg ridges

> ggplot2

- ggplot2 is most popular visualization package in R.
- Use it singularly or with others
- less code more Viz

Creater: statistician, developer : Hadley Wickham 2005

Inspiration:→ The Grammar of Graphics
Schorarly study of Data Viz by Leland Wilkinson

In the same way English grammar gives us rules to build any kind of sentence

It gives us rules to build any kind of visual.

2/3

Rule of GGplot 2:

> Start with the ggplot function & choose a dataset to work with
> Add a geom-function to display your data.
> Map the variables you want to plot in the arguments of the aes() function.

Template :→

Ex

```
ggplot (data= penguins) +
    geom_point (mapping = aes (x= flipper_length_mm,
                               y= body_mass_g))
        ↓
ggplot ( data= <Data> ) +
    <GEOM_FUNC> (mapping = aes (<AES mappings>))
```

Ex

```
ggplot (data= penguins) +
    geom_point (mapping = aes (x= bill_length_mm,
                               y= bill_depth_mm))
```

—×—×—

(*) aes (x= "-||-", y= "-||-", color = Species)

changes color of the different data points.

(*) Shape= species : assigns different shape to each data point (clustering)

Legend is generated automatically.

(*). Color =species & shape= species :→ can be used simultaneously

Write inside aes for chart in regard to variabl
outside for all!

- alpha : controls transparency

mapping = aes(

ggplot (data = df) + geom_point (x= "" , y= "" ,
color = species, shape = species , size = species))

mapping = aes(
geom_point (x= "" , y= "" ,), color = "purple")

——x——x

* ggplot (data = df)+
geom_smooth (mapping = aes (x, y,)) +
geom_point (mapping = aes (x, y,))

eg,
linetype = species → different line for different species

——x——

geom_jitter (mapping = aes (x, y))
scatter plot with random noise

Jittering helps us deal with overplotting

——x——x

> In simpler words, when you learn the basic steps for creating a plot in ggplot2.

> You can reuse these steps to create lots of different kinds of plots.

+> You can add/remove layers of detail to your plot without changing its basic structure or underlying data.

Upcoming:- Aesthetics, Geoms, Facets, labels & annotations

⊛ Aesthetic :→ A visual property of an object in your plot.

ex. in scatter plots, aesthetics are:→ size, shape, color

⊛ Geom :→ Geometric object used to represent data

ex. points → Scatter Bars → barchart

line :→ Line chart

⊛ Facet → let you display smaller group, or subsets, of your data

ex. separate plot for all variable

⊛ label & Annotations → lets you customize your plots.

Page No:
Date

* Hands on →

```
install.packages ("___")
library ("//")
```

```
pacman :: pload(pacman, ggplot2, palmerpenguins)
```

```
penDF = data.frame (penguins)
```

***
```
ggplot(data = penDF) + geom_point(mapping =
        aes(x=flipper_length_mm, y= body_mass_g))
```

ggplot (data = penDF) → create a plane

Add layers

```
geom_point( ~~aes~~ mapping = aes(x= flipper_length_mm,
                                  y = body_mass_g))
```

geom functions :- different function for different plots.
geom_point → Scatterplot | geom_bar → Bar chart

Each geom takes a mapping argument and its always paired with aes function which takes x & y variable to map

Same code :- Different syntax →
```
ggplot(data = penguins, mapping = aes(x= flipper_//,
                                      y= body_//_))
  + geom_point()
```

```
library("diamonds")   # already loaded
 ggplot (data=diamonds)
geom_bar( mapping = aes (x= cut ))
```

↓

by default calculates row & numbers

↓

how many time row element is present

```
geom_bar (mapping =   aes (x= cut, color = cut))
                                 fill = cut
```

↓

color: outlines the bar

fill : fills the bar

Damn

facets →

Shows different views of your data

> used for comparison |

> facet_wrap() : single variable

> facet_grid()

```
ggplot (data = penguins) +
 geom_point (mapping = aes(x, y, color = species)) +
 facet_wrap (~ species)
```

used when viz is too dense. | 3 diff plots for 3 vars

facet-grid () :->

```
ggplot(data=penguins)
  geom-point(mapping = aes (x= flmm, y= bmg,
                              color = species))+
   facet-grid(sex~species)
```

Splits / divides viz into vertically by first variable
horizontally by second variable

<*> can also use
just one variable

```
facet_grid(~sex)    or  facet-grid(~species)
```

week 4

3/3

Page No.
Date

Annotate →

To add notes to a document or diagram to explain or comment upon it

```
+ labs( title = "palmer Penguins: Body mass vs.
                                Flipper len"))
```

arguments → title, subtitle,
Captron: Data source

```
P <- ggplot() + geom_fun() + labs()

P + annotate("text", x=220, y=3500,
        label= "The Gentoos are the Largest",
        color= "purple", fontface = "bold", size = 4.5,
        angle = 25)
```

* Saving your Visualization

> gg save()
> Expert option (plots pane)
  > save as Img or pdf (total → 6 Image format)

ggsave() → salves last plot & current graphic device size

ggsave("Name.png")