

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

It pulls only first 10 rows

* Tidy data standards

- > Variables are organized into columns
- > Observations are organized into rows
- > Each value must have its own cell

— x — x — x —

install.packages("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.

glimpse() & str()

Summary

library(tidyverse)

from dplyr

Data Frames Data Analysis default way of interacting with data.

Page No.	
Date	

2/3
Cleaning up with the basics →

3 Packages Here, skimr, janitor

Here → makes Referencing easier.
Skimr → Summarizing 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 %>%
 select(-species) # Everything but species

* Rename Column

penguins %>%
rename(istand_new = istand)

* rename_with (penguins, tolower)
↓

For consistency,

* Clean Names: Consistent, no duplicate, etc

clean_names(penguins)

Operators ↪

Arithmetic

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

Relational

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

Logical

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

$x \in C(3, 5, 7)$
 $y \in C(2, 4, 6)$

$x < 5 \ \& \ y < 5$

④ T F F

$x < 5 \ \&\& \ y < 5$

True

Assignment Operator

← leftward } Assignment → Rightward
←← leftward }
= leftward }
Page No. _____
Date _____

* 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 %>% group_by(species, island) %>%
dropna() %>% summarize(max_bill = max(bill_length_mm),
mean_b = min(bill_length_mm))

Filter

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

V/3

28/03/23

④ R lets you move back and forth betn analysis & visualization quickly.

> Plotly → wide range; general purpose

> RGL → Specific, 3D

> Lattice

> Diagraphs

> Leaflet

> Highcharter

> Patchwork

> gganimate

> ggeffects

> ggplot2

• ggplot2 is most popular visualization package in R.

• Use it singularly or with others

• less code more viz

Creators

statistician, developer : Hadley Wickham 2005

Inspiration

The Grammar of Graphics

Scholarly study of Data Viz by Jerry 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.

Rule of ggplot2:

- > 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

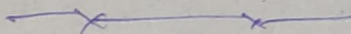
```
ggplot(data = penguins) +  
  geom_point(mapping = aes(x = flipper-length-mm,  
                           y = body-mass-g))
```



```
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 = "bill", y = "depth", 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 variable
outside for all!

Page No.	
Date	

• 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~~), color = "purple")

— x — x —

* ggplot (data = df) +

geom_smooth (mapping = aes (x, y)) →

geom_point (mapping = aes (x, y, ?))

Q1

linetype = species
species

→ different line for different

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.

— x — x —

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 → bar chart

line → Line chart

* Facet → let you display smaller groups, or subsets, of your data
ex. separate plot for all variables

* Label & Annotations → lets you customize your plots.

* Hands on →

```
install.packages ("____")
library ("H")
```

```
pacman :: pload (pacman, ggplot2, partner penguins)
```

```
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 is always paired with aes function which takes x & y variable to map

Same code → ggplot (data = penguins, mapping = aes (x = flipper_l1, y = body_l1))
Different syntax
+ geom_point ()


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



by default calculates row & numbers



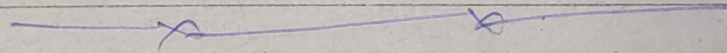
how many time row element is present

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



color: outlines the bar
fill & fill the bar

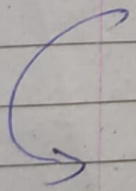
Damn



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() \Rightarrow

ggplot(data = penguins)
 geom_point(mapping = aes(x = flimm, 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)

~~~~~

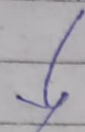
week 4

3/3

### 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

Caption: Data source

```
p <- ggplot() + geom_fcn() + labs()
```

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

— x — x

### \* Saving your Visualization

> ggsave()

> Export option (plots pane)

> save as img or pdf

(total → 6)

Image format

ggsave() → saves last plot & current graph's device size

```
ggsave("Name.png")
```

— x —