phuse·share
connect·share·advance

APAC 2026

# aRe You Ready!! Fun, Fast & Functional Coding Tips

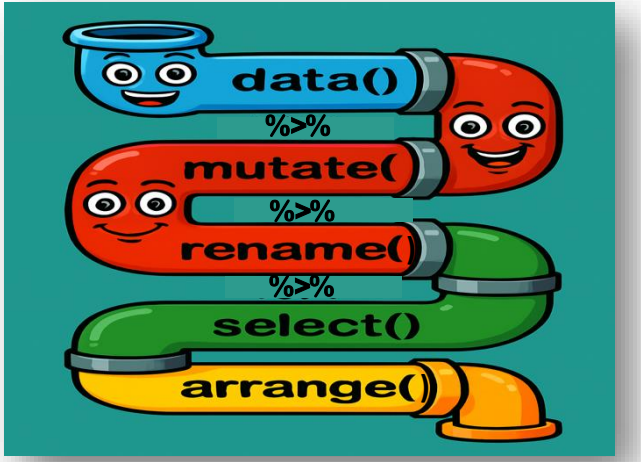
**Xeviers Koner**
**Merck Specialities Pvt. Ltd.**

# Disclaimer:

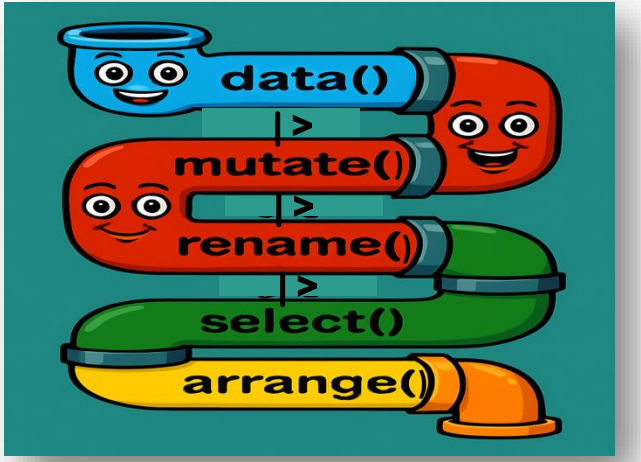
- The views and opinions expressed in this presentation are solely those of the presenter and do not necessarily reflect the official policy or position of Merck Specialities Private Limited.
- These slides are intended for educational purposes only and are not intended for wider distribution outside the intended purpose without presenters' approval.

# %>% (magrittr pipe)

# |> (native pipe)

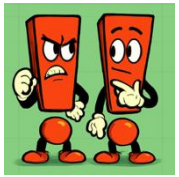







```r
library(dplyr)
ADSL<- adsl %>%
  mutate(AGEGRP = ifelse(AGE > 65, "Elderly", "Adult")) %>%
  rename(SUBJECTID = SUBJID) %>%
  select(SUBJECTID, AGE, AGEGRP, SEX) %>%
  arrange(AGE)
```

```r
ADSL<- adsl |>
  mutate(AGEGRP = ifelse(AGE > 65, "Elderly", "Adult")) |>
  rename(SUBJECTID = SUBJID) |>
  select(SUBJECTID, AGE, AGEGRP, SEX) |>
  arrange(AGE)
```

!  !!  !!! 

## Logical Not

```
ADSL %>% filter(!is.na(AGE))
```

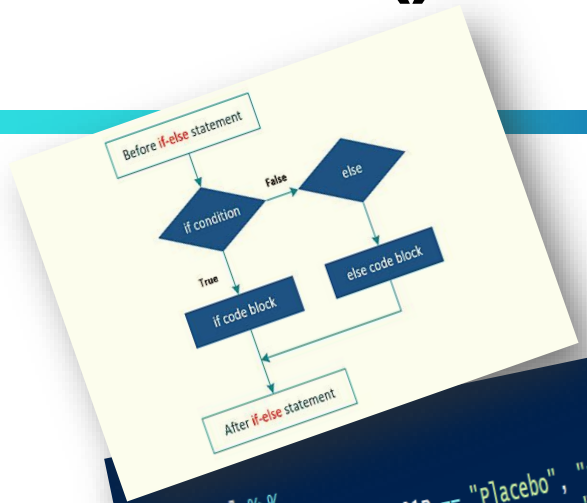
## Unquote a single name/variable

```
varname <- quo(AGE)
ADSL %>% select(!!varname)
```

## Unquote multiple names/variables

```
vars <- quos(USUBJID, AGE, SEX)
ADSL %>% select(!!!vars)
```

# Switch() VS ifelse()

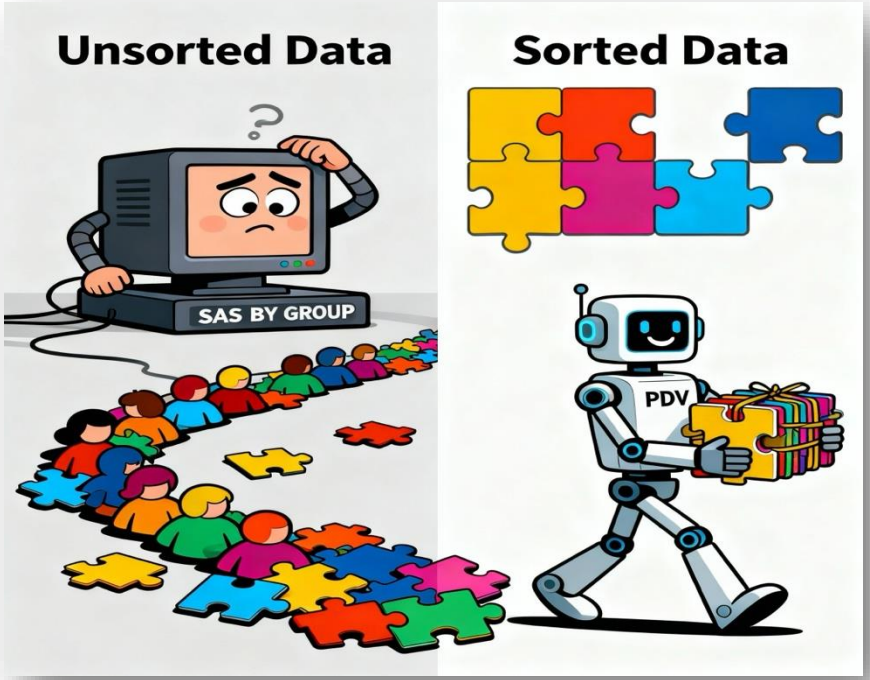



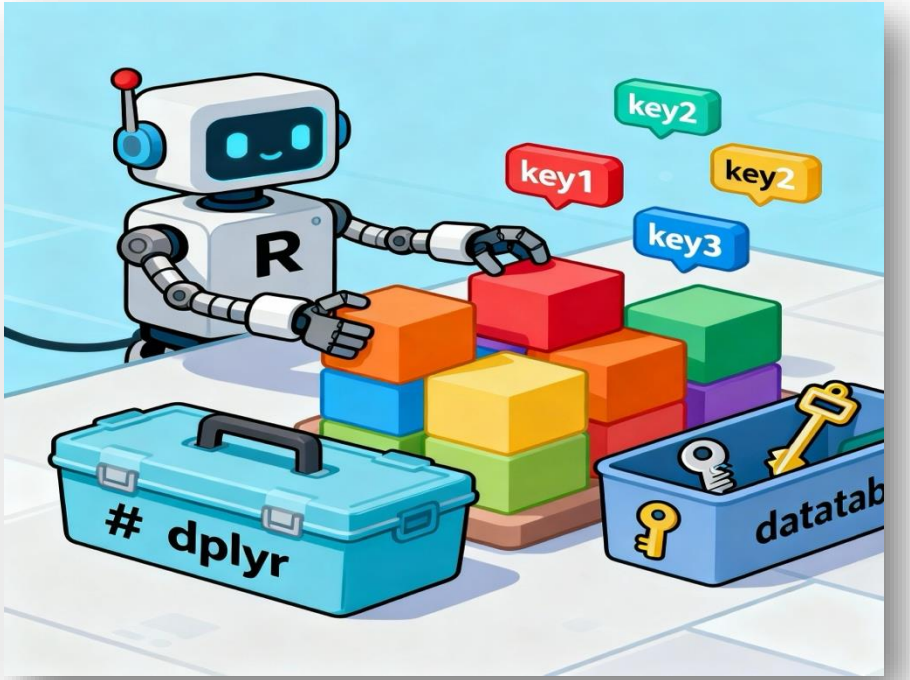

```
ADSL <- adsl %>%
  rowwise() %>%
  mutate(TRTGRP = switch(TRT01P,
                         "Placebo" = "Placebo Group",
                         "DrugA"   = "Drug A Group",
                         "DrugB"   = "Drug B Group",
                         "Other"))
```

```
ADSL <- adsl %>%
  mutate(TRTGRP = ifelse(TRT01P == "Placebo", "Placebo Group",
                  ifelse(TRT01P == "DrugA",   "Drug A Group",
                  ifelse(TRT01P == "DrugB",   "Drug B Group",
                  "Other"))))
```

**Proc Sort in SAS**

**No Sorting required in R**

# Glue() & Paste() – brothers from another mother !


**paste()** - base R function

- paste(..., sep = " ", collapse = NULL)

**glue()** - from the **glue** package

- glue(..., .sep = "", .envir = parent.frame())

**Paste()**

```
name <- "Xeviers"
age <- 25
paste("My name is", name, "and I am", age, "years old.")

# Output: "My name is Xeviers and I am 25 years old."
```

```
ADAE1 $subj_info <- paste(
  "Subject", adsl_ae$ USUBJID, "is a", adsl_ae$ AGE, "years old",
  adsl_ae$ SEX, "with", adsl_ae$ AEDECODE, "adverse event(s)."
)
```

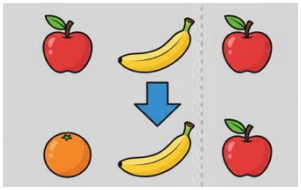
**Glue()**

```
library(glue)

name <- "Xeviers"
age <- 25

glue("My name is {name} and I am {age} years old.")
```

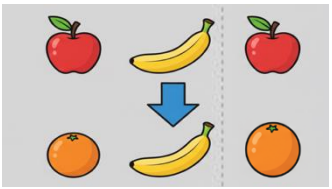
```
ADAE1 $subj_info <- glue(
  "Subject {USUBJID} is a {AGE} year old {SEX} with {AEDECODE} adverse event(s)."
)
```



# Sub() & Gsub()

substitute first

global substitute



**Sub() : sub(pattern, replacement, x, ignore.case=FALSE, perl=FALSE, fixed=FALSE)**

- sub(pattern, replacement, x, ...) finds the first occurrence of pattern in each element of x and replaces it with replacement.

```
ADSL <- data.frame(
  TRT01A = c("  Placebo", "Drug_20  ", "Drug  _40", "  Drug_60   ")
)

ADSL$TRT01A1 <- sub(" ", "", ADSL$TRT01A)
# [1] "Placebo" "Drug_20  " "Drug  _40" "  Drug_60   "
```

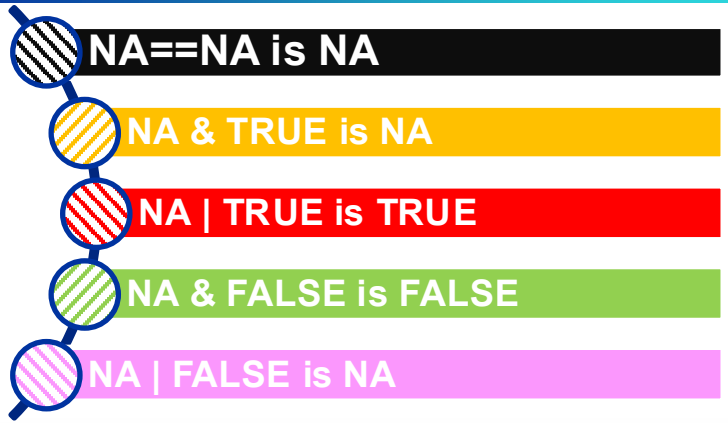
**Gsub() : gsub(pattern, replacement, x, ignore.case = FALSE, perl = FALSE, fixed = FALSE)**

- gsub(pattern, replacement, x, ...) replaces every (global) occurrence of pattern in each element of x.

```
ADSL <- data.frame(
  TRT01A = c("  Placebo", "Drug_20  ", "Drug  _40", "  Drug_60   ")
)

ADSL$TRT01A1 <- gsub(" ", "", ADSL$TRT01A)
# [1] "Placebo" "Drug_20"   "Drug_40"   "Drug_60"
```

# Moody “NA”

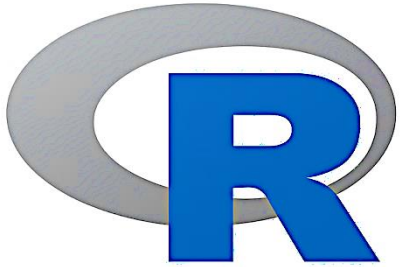



| Operand 1 | Operand 2 | & Result | \| Result | Explanation (Layman) |
|---|---|---|---|---|
| TRUE | TRUE | TRUE | TRUE | Both true, so AND and OR are true |
| TRUE | FALSE | FALSE | TRUE | AND false because one false; OR true because one true |
| FALSE | FALSE | FALSE | FALSE | Both false, so AND and OR are false |
| TRUE | NA | NA | TRUE | AND unknown because unknown & true unknown; OR true because one true |
| FALSE | NA | FALSE | NA | AND false because false & unknown is false; OR unknown because unknown or false unknown |
| NA | NA | NA | NA | Both unknown, result unknown |

```
# AND examples
TRUE & TRUE      # TRUE: both true
TRUE & FALSE     # FALSE: one false makes whole false
FALSE & NA       # FALSE: false dominates
TRUE & NA        # NA: unknown because one side unknown
NA & NA          # NA: both unknown
```

```
# OR examples
TRUE | FALSE     # TRUE: one true makes whole true
FALSE | FALSE    # FALSE: both false
FALSE | NA       # NA: unknown because one side unknown
TRUE | NA        # TRUE: true dominates
NA | NA          # NA: both unknown
```

# R 4.5.1: Great Square Root



4.5.1

Great Square Root

- factanal()
- pbeta()
- prettynum()
- signif()
- dbinom(), dnbinom()
- Tcl/Tk 9 support
- unzip()

# Genzplyr: only for fun purpose


## Installation

```
pak::pak("hadley/genzplyr")
```

### The translation guide

| dplyr verb | genzplyr verb | Meaning |
|---|---|---|
| `filter()` | `yeet()` | Remove rows that don't pass the vibe check |
| `select()` | `vibe_check()` | Keep only columns that matter |
| `mutate()` | `glow_up()` | Transform your data into its best self |
| `summarise()` | `no_cap()` | Get real summary stats, no lies |
| `arrange()` | `slay()` | Sort by slay factor |
| `group_by()` | `squad_up()` | Group data into squads |
| `ungroup()` | `disband()` | Break up the squad |
| `rename()` | `lowkey()` | Change names on the down-low |
| `distinct()` | `periodt()` | Remove duplicates, and that's on periodt |
| `pull()` | `main_character()` | Extract a column (give it main character energy) |
| `slice_head()` | `send_it()` | Take the top rows and send it |
| `count()` | `its_giving()` | Count occurrences (it's giving statistics) |
| `left_join()` | `link_up()` | Merge datasets but keep all your day-ones (left table stays main squad) |
| `right_join()` | `clout_chase()` | Merge, but the other chums call the shots on who stays (right table is main squad) |
| `inner_join()` | `mutuals_only()` | Merge, but only keep rows where both tables are mutually following each other |
| `full_join()` | `everyone_in_the_groupchat()` | Bring everyone, even if messy (keeps all rows) |
| `anti_join()` | `ghost()` | Forget about those who don't vibe with you (remove non-matching rows) |
| `semi_join()` | `only_the_reals()` | Just keep the rows that vibe with both tables (matching rows only) |

**Old way (Cringe):**

```
mtcars |>
  filter(mpg > 20) |>
  select(mpg, cyl, hp) |>
  mutate(kpg = mpg * 1.6) |>
  arrange(desc(mpg))
```

**New way (Bussin):**

```
mtcars |>
  yeet(mpg > 20) |>
  vibe_check(mpg, cyl, hp) |>
  glow_up(kpg = mpg * 1.6) |>
  slay(desc(mpg))
```

# Reference:


1. https://tidyverse.org/blog/2023/04/base-vs-magrittr-pipe/
2. https://www.geeksforgeeks.org/cpp/switch-vs-else/
3. https://stackoverflow.com/questions/53049986/r-functions-glue-vs-paste
4. https://communities.sas.com/t5/SAS-Programming/Is-it-a-must-to-do-proc-sort-right-before-merging-dataset-using/td-p/709497
5. https://stackoverflow.com/questions/44045220/data-need-to-be-sorted-before-merging-in-r
6. https://github.com/hadley/genzplyr/?tab=readme-ov-file


phuse

**The Global Healthcare Data Science Community**

**Contact Channels**
UK +44 1843 609600
office@phuse.global
phuse.global

**Social Media**
@phusetwitta
/phusebook
/phusetube
/company/phuse