Practical R Cheat Sheet

By Tony Fraser

dplyr basics

```
library(dplyr)
# Basic SQL Operations
df_select <- df %>% select(column1, column2) # SELECT
df_where <- df %>% filter(column1 == "value") # WHERE
df_orderby <- df %>% arrange(column1) # ORDER BY


 # GROUP BY & HAVING
df_groupby <- df %>%
  group_by(column1) %>%
  summarise(avg_col2 = mean(column2)) %>%
  filter(avg_col2 > value)

# GROUP BY AND COUNT
grouped_data <- accidents %>%
  filter(!is.na(BOROUGH) & `CONTRIBUTING FACTOR VEHICLE 1` != "unspecified") %>%
  group_by(BOROUGH, `CONTRIBUTING FACTOR VEHICLE 1`, `VEHICLE TYPE CODE 1`) %>%
  summarise(count = n()) %>%
  arrange(-count)

## Between
accidents_zoomed <-   accidents %>%
  filter(
    between(LATITUDE, latitude_center - radius_constant_latitude, latitude_center + radius
    between(LONGITUDE, longitude_center - radius_constant_longitude, longitude_center + ra
  ) %>%
  slice(1:record_count)

#JOINS
df_join <- df1 %>%
  left_join(df2, by = c("column" = "column"))
result <- employees %>%
  left_join(dayWorkers, by = c("employeeId" = "emp_id"))
```

```r
df_limit <- df %>% slice_head(n = 10) # LIMIT

# Functions & Operations
df_avg <- df %>% summarise(avg_col1 = mean(column1)) # Aggregate Functions

# String functions in dplyr might need additional packages like stringr.
# Date functions can vary depending on the class of the date in the dataframe.

# Modifying Data Frames
# In dplyr, operations are non-mutative by default. You'd need to overwrite
# the dataframe or assign to a new one.
df <- df %>% add_row(column1 = "value1", column2 = "value2") # INSERT
df <- df %>% mutate(column1 = ifelse(column2 == "value", "new_value", column1)) # UPDATE
df <- df %>% filter(column1 != "value") # DELETE

# Miscellaneous
df_value <- df %>% filter(column1 == value) # Using R Variables in Queries
df_na <- df %>% filter(is.na(column1)) # Handling NA

# if one column has a string add another
df_new <- df %>%
  mutate(newColumn = if_else(str_detect(oldColumn, "appletv"), "appleTV", "notAppleTv"))
```

gt basics

```r
library(dplyr)
library(gt)

by_borough <- accidents %>%
  filter(!is.na(BOROUGH) & nchar(BOROUGH) > 0 & `CONTRIBUTING FACTOR VEHICLE 1` != "unspec
  group_by(`VEHICLE TYPE CODE 1`, BOROUGH) %>%
  summarise(count = n(), .groups = "drop") %>%
  mutate(`VEHICLE TYPE CODE 1` = ifelse(count < 5, "Other", `VEHICLE TYPE CODE 1`),
         `VEHICLE TYPE CODE 1` = ifelse(nchar(`VEHICLE TYPE CODE 1`) < 1, "unspecified", `
  arrange(`VEHICLE TYPE CODE 1`, -count)

by_borough_table <- by_borough %>%
  gt() %>%
  tab_style(
    style = cell_text(align = "left"),
    locations = cells_body(columns = c(`VEHICLE TYPE CODE 1`, BOROUGH, count))
  ) %>%
  cols_label(
    `VEHICLE TYPE CODE 1` = "Vehicle Code",
    BOROUGH = "Borough",
    count = "Accident Count"
  ) %>%
  tab_style(
    style = cell_text(weight = "bold", size = "large"),
    locations = cells_column_labels(columns = c(`VEHICLE TYPE CODE 1`, BOROUGH, count))
  )

by_borough_table
```

Quarto

https://quarto.org/docs/computations/execution-options.html