

Day 3 Cheatsheet

Subsetting Data in R

Functions

Library/Package	Piece of code	Example of usage	What it does
Base R	<code>nrow(x); ncol(x)</code>	<code>nrow(x); ncol(x)</code>	Get the number of rows and the number of columns in an object <code>x</code> , respectively.
Base R	<code>dim(x)</code>	<code>dim(x)</code>	Get the number of rows <i>and</i> number of columns in an object <code>x</code>
dplyr	<code>glimpse(x)</code>	<code>glimpse(mtcars)</code>	Get an overview of data frame <code>x</code>
Base R	<code>data.frame()</code>	<code>df <- data.frame(1:3)</code>	Creates a data frame where the named arguments will be the same length.
Base R	<code>tibble()</code>	<code>tibble(mtcars)</code>	Creates a tibble from a data.frame or matrix.
dplyr	<code>rename()</code>	<code>df <- rename(df, MPG = mpg)</code>	Renames designated columns while keeping all variables of the data.frame
dplyr	<code>pull()</code>	<code>pull(df, 'existing_variable_name')</code>	Extract a column as a vector
dplyr	<code>select()</code>	<code>select(df, 'existing_variable_name')</code>	Selects columns that match the specified argument
dplyr	<code>filter()</code>	<code>filter(df, mpg > 20)</code>	Returns a subset of rows matching the conditions of the specified logical argument
Base R	<code>==, <=, >=, !=</code>	<code>filter(df, mpg > 20)</code>	These are binary operators which allow for the comparison of values in an object. They are handy for use with <code>filter()</code>
Base R	<code>%in%</code>	<code>filter(df, mpg %in% c(20,21,22))</code>	Checks if the given value(s) on the left side of the operator are in the vector or other R object defined on the right side of the operator. It returns a logical <code>TRUE</code> or <code>FALSE</code> statement.
dplyr	<code>%>%</code>	<code>df <- df %>% select('new_variable_name')</code>	Funnel a data.frame through tidyverse operations

Library/Package	Piece of code	Example of usage	What it does
dplyr	<code>mutate()</code>	<code>df <- mutate(df, newcol = wt/2.2)</code>	Adds a new column that is a function of existing columns
dplyr	<code>recode()</code>	<code>df <- mutate(df, mpg = recode(mpg, oldval = newval))</code>	This function allows you to recode based on conditions.
dplyr	<code>case_when()</code>	<code>df <- mutate(df, mpg = case_when(mpg > value ~ newvalue))</code>	This function allows you to recode based on conditions. Anything not specified will become an NA.

- See `tidyselect` helpers for handy things to use with `select()`.

* This format was adapted from the cheatsheet format from AlexsLemonade.