SAS® to R: CHEAT SHEET – A Practical Guide for Clinical Programmers

Author: Lekitha JK, Senior Statistical Programmer, Atorus Research
Co-author: Jalendhar Ettedi, Principal Programmer, Atorus Research

INSTALLING AND LOADING PACKAGES

`install.packages()` installs specified package.
`library()` loads specified package.

READING IN FILES

`haven::read_xpt()` reads in XPT files.
`haven::read_sas()` reads in sas7bdat files.
`readxl::read_excel()` reads in xls/xlsx files.
`readr::read_delim()`, `read_csv()`, `read_tsv()` read in various delimited files.
`getwd()` returns working directory.

FINALIZING AND OUTPUTTING FILES

APPLYING METADATA

`xportr::xportr_df_label()` assigns a data frame label from a data frame containing dataset level metadata.

`xportr::xportr_label()`, `xportr::xportr_length()`, `xportr::xportr_type()`, `xportr::xportr_order()`, `xportr::xportr_format()` assign applicable column attribute from a data frame containing variable level metadata.

WRITING FILES

`xportr::xportr_write()` writes an XPT file.
`openxlsx::write.xlsx()` writes an xlsx file.
`readr::write_delim()`, `write_csv()`, `write_tsv()` write various delimited files.

TIDY SELECTION

SELECTION HELPERS

`tidyselect::starts_with()` selects columns that start with a prefix.
`tidyselect::ends_with()` selects columns that end with a suffix.
`tidyselect::contains()` selects columns that contain a literal string.
`tidyselect::matches()` selects columns that match a regular expression.
`tidyselect::num_range()` selects columns that match a numerical range like x01, x02, x03.

TIDY SELECT OPERATORS

- `:` selects a range of consecutive columns.
- `!` takes the complement of a set of columns.
- `&` and `|` selects the intersection or the union of two sets of columns.
- `c()` combines selections.

MISSINGS AND FACTORS

`NaN` represents "not a number" and can be checked using `is.nan()`.
`NA` represents "not applicable" and can be checked using `is.na()`.
`factor()` encodes a vector as a factor.
`levels()` provides access to the levels attribute of a variable.

OPERATORS

An operator is a symbol that tells the compiler to perform specific operations.

MISCELLANEOUS OPERATORS

Operator	Description
`<-`	assign a value to a name
`%>%`	chain multiple calls into a single statement
`:`	creates a series of numbers in sequence
`%in%`	identifies if an element belongs to a vector

ARITHMETIC OPERATORS

Operator	Description
`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division
`^ or **`	exponentiation
`x %% y`	modulus (x mod y) 7%%2 is 1
`x %/% y`	integer division 7%/%2 is 3

LOGICAL OPERATORS

Operator	Description
`<`	less than
`<=`	less than or equal to
`>`	greater than
`>=`	greater than or equal to
`==`	exactly equal to
`!`	not
`x	y`	x OR y
`x & y`	x AND y

PROGRAMMING BASICS

CREATING A NEW COLUMN

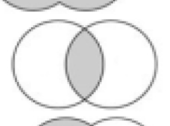
`dplyr::mutate()` adds new columns to a data frame and preserves the existing ones.

DATA STEP OPTIONS & STATEMENTS

SET

Use the **assignment operator** (`<-`) to create a new data frame from an existing data frame.
`dplyr::bind_rows()` stacks rows of two or more data frames.

MERGE

The join functions add columns from the right data frame to the left data frame and match by specified "keys".
`dplyr::full_join()` includes all rows in either data frame.
`dplyr::inner_join()` includes all rows in both data frames. 
`dplyr::left_join()` includes all rows in the left data frame.
`dplyr::right_join()` includes all rows in the right data frame.

DROP/KEEP/RENAME

`dplyr::select()` selects columns in a data frame. To drop a column, precede the column name with a dash (-). To rename a column use `new_name=old_name` syntax.
`dplyr::rename()` changes the names of individual columns using `new_name = old_name` syntax.

IF/ELSE

`dplyr::if_else()` modifies variables by applying a single conditional statement.
`dplyr::case_when()` modifies variables by applying a series of conditional statements.
`dplyr::case_match()` a "vectorized switch" variant of `dplyr::case_when()` that matches on values rather than logical expressions.

WHERE

`dplyr::filter()` subsets a data frame, retaining all rows that satisfy the conditions.

PROCEDURES

PROC CONTENTS

`str()` displays the internal structure of an R object.
`class()` reveals the type of object being inspected.
`attr()` allows access to object attributes to get the value.

PROC FREQ

`dplyr::count()` counts the unique values of one or more columns.
When needed within groups, `dplyr::group_by()` performs operations within a specified group of columns.

PROC MEANS

`dplyr::summarize()` uses summary functions to summarize data into a single row of values.
When needed within groups, `dplyr::group_by()` performs operations within a specified group of columns.
Common Summary Functions:
`dplyr::n(), min(), max(), mean(), median(), var(), sd(), quantile(), IQR(), sum().`

PROC PRINT

R prints results directly to the console. But you can also use `print()`.
`head()` returns the first parts of a vector, matrix, table, data frame or function. `tail()` returns the last parts. And `dplyr::slice()` indexes rows by their integer locations.

PROC SORT

`dplyr::arrange()` orders the rows of a data frame by the values of selected columns.
`dplyr::desc()` switches the order to descending.

PROC TRANSPOSE

`tidyr::pivot_wider()` widens data, increasing the number of columns and decreasing the number of rows.
`tidyr::pivot_longer()` lengthens data, increasing the number of rows and decreasing the number of columns.