**PhD Plus Data Literacy in R Cheatsheet**

**Set working directory**

setwd(“path/to/directory”)

Use tab key to drill into directory tree.

Use .. to go back up one branch.

Or Session…Set Working Directory

**Install/Update/Load Packages**

install.packages(“package”)

Or Tools…Install Packages…

library(package)

Tools…Check for Package Updates…

package::function indicates function in package

Example: readr::read\_csv()

**Assignment**

Use <- or =

Alt + - (Win) Option + - (Mac) to insert <-

**Import data**

d <- read.csv(“path/to/file.csv”)

Or using read\_csv() from readr package:

d <- read\_csv(“path/to/file.csv”)

Use readxl package to import Excel files.

Use haven package to import SAS, SPSS, Stata files.

**Glance at data frame named “d”**

View(d); names(d)

str(d); dplyr::glance(d)

summary(d); head(d); tail(d)

**Comparison and Logical**

== (equality)

!= (not equal)

>,>= (greater than, greater than or equal to)

<,<= (less than, less than or equal to)

& (and)

| (or)

! (not)

%in% (matching operator)

**Missing values**

Missing values indicated with NA

NA = not available

is.na() returns TRUE if value missing, FALSE otherwise

**Create/combine vectors**

x <- c(2, 4, 8)

y <- c(x, 10) # append 10 to 2,4,8

**TRUE/FALSE**

TRUE = 1, FALSE = 0

x <- c(2, 4, 8)

x > 3

[1] FALSE TRUE TRUE

sum(x > 3) # how many TRUE?

[1] 2

**Basic statistical functions**

mean(); median(); sd(); var()

quantile() # percentiles

length() # number of values (n)

sqrt() # square root

log() # natural log

log10() # log base 10

min(); max()

range() # min and max

**Counts and proportions**

Count of males/females in column “sex” of data frame “d”:

table(d$sex)

Proportion of females

mean(d$sex == “female”)

If any missing values, set na.rm = TRUE

mean(d$sex == “female”,

na.rm = TRUE)

**2 and 3-way tables**

Crosstab of column “sex” (m/f) with column “married” (y/n) in data frame “d”:

xtabs(~ sex + married, data = d)

Crosstab of column “sex” (m/f) with column “married” (y/n) stratified by “religious” (y/n) in   
data frame “d”:

xtabs(~ sex + married + religious, data = d)

**marginal proportions**

Given following table saved as “tab”:

married

sex n y

f 20 26

m 30 24

Proportion married by sex with base R pipe (|>):

tab |> proportions(margin = 1)

married

sex n y

f 0.43 0.57

m 0.56 0.44

Proportion sex by married with base R pipe (|>):

tab |> proportions(margin = 2)

married

sex n y

f 0.40 0.52

m 0.60 0.48

**extract table values**

tab[1,] # row 1

tab[,2] # column 2

tab[1,2] # cell in row 1, col 2

tab[,1,drop=FALSE] # col 2 as table

**Plotting with ggplot2**

<https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-visualization.pdf>

Example data frame: d

|  |  |  |
| --- | --- | --- |
| x | y | g |
| 1300 | 3.8 | “a” |
| 1400 | 3.2 | “b” |
| 1280 | 2.9 | “a” |

library(ggplot2)

distribution of y

ggplot(d) + aes(x = y) +

geom\_histogram()

ggplot(d) + aes(x = y) +

geom\_density()

scatterplot of x and y

ggplot(d) + aes(x, y) +

geom\_point()

scatterplot of x and y conditional on g

ggplot(d) + aes(x, y) +

geom\_point() +

facet\_wrap(~g)

scatterplot of x and y points colored by g

ggplot(d) + aes(x, y, color=g) +

geom\_point()

scatterplot of x and y, semi-transparent points

ggplot(d) + aes(x, y, color=g) +

geom\_point(alpha = 1/5)

# alpha ranges from 0

# (invisible) to 1 (solid)

scatterplot with x and y and smooth trend line

ggplot(d) + aes(x, y) +

geom\_point() +

geom\_smooth()

# method=“lm” for straight line

distribution of y for each level of g

ggplot(d) + aes(x = g, y = y) +

geom\_boxplot()

ggplot(d) + aes(x = g, y = y) +

geom\_violin()

ggplot(d) + aes(x = g, y = y) +

geom\_jitter(width = 0.2,

height = 0)

Add title, axis labels, etc

ggplot(d) +

aes(x, y, color=g) +

geom\_point() +

labs(x = “SAT”, y = “GPA”,

title = “SAT vs GPA”)

**Basic data wrangling with dplyr**

<https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-transformation.pdf>

Example data frame: d

|  |  |  |
| --- | --- | --- |
| x | y | g |
| 1300 | 3.8 | “a” |
| 1400 | 3.2 | “b” |
| 1280 | 2.9 | “a” |

dplyr functions work with pipes.

Insert pipe operator:

Ctrl+Shift+M (Win); Cmd+Shift+M (Mac)

dplyr always returns a tibble (data frame)

NOTE: Assign result to save transformation!

Extract rows that meet a condition

d %>% filter(x > 1300)

Arrange data by columns in ascending order

d %>% arrange(y)

Arrange data by columns in descending order

d %>% arrange(desc(y))

Select specific columns

d %>% select(y, g)

d %>% select(-x) # all but x

Two useful select helpers

d %>% select(starts\_with("p"))

d %>% select(-starts\_with("p"))

d %>% select(ends\_with("ing"))

Add a column and save result

d <- d %>%

mutate(z = x – mean(x))

Summaries for each group (eg, mean)

d %>%

group\_by(g) %>%

summarize(m = mean(y))

Count membership in group

d %>% count(g)

Rename columns and save result

d <- d %>% rename(SAT = x)

# new\_name = old\_name

Drop obs missing on a given variable

d %>% drop\_na(y)

**Basic data wrangling with dplyr (cont’d)**

Create an indicator variable using if\_else

# j = 1 if y = 4, else 0

d <- d %>% mutate(j =

if\_else(y==4,1,0))

Random sample of 20 observations

d %>% sample\_n(20)

Combining dplyr functions and saving result

nd <- d %>%

filter(x > 1000) %>%

group\_by(g) %>%

summarize(m = mean(y))

**Working with dates**

Use lubridate to format dates. Use m, d, y to create function. Dates stored as number of days since 1/1/70. Eg, to format dates of form May 2, 2021 in column “date” of data frame “d”

library(lubridate)

d <- d %>%

mutate(date = mdy(date))

Append hms to format date-times, dates with a time component. Date-times stored as number of seconds since 1/1/70. Eg, to format date-time of form May 2, 2021 2:34:23 in column “date” of data frame “d”

d <- d %>%

mutate(date = mdy\_hms(date))

Extract day of week, month, year from formatted “date” column in data frame “d”:

wday(d$date, label = TRUE)

month(d$date, label = TRUE)

year(d$date)

**Confidence intervals**

95% confidence intervals for means

Eg, data frame “d” with column “weight”

t.test(d$weight)$conf.int

# or

Hmisc::smean.cl.normal(d$weight)

95% confidence intervals for proportions

Eg, data frame “d” with binary column “married” where 1 = married, 0 = not married

# proportion married

prop.test(x = sum(d$married),   
 n = length(d$married))

**Linear Models**

Model expected value of a variable based on other variables. Eg, data frame “d” with columns “value”, “size”, “acres”, and “zone”. Model expected value of value as a function of other variables.

m <- lm(value ~ size + acres +   
 zone, data = d)

View model summary:

summary(m)

View model coefficients:

coef(m)

View model diagnostic plots:

plot(m)

95% confidence intervals for coefficients

confint(m)

F-test for all coefficients (except intercept)

anova(m)

**lm model summary**

* Residuals section: quick assessment of residuals. Ideally 1Q/3Q and Min/Max will be roughly equivalent in absolute value.
* Coefficients: lists the estimated coefficients along with hypothesis tests for the null hypothesis that each coefficient is 0.   
  Est/SE = t-value.
* Residual standard error: estimate of the constant standard deviation of the normal distribution of the errors
* degrees of freedom:   
  sample size - number of coefficients
* R-squared: proportion of variance explained
* F-statistic: overall test that all coefficients (except intercept) are 0.

**Visualize interactions in lm model**

When two variables interact, their effects depend on each other. Interactions can be visualized with the ggeffects package. Eg, assume model with interaction:

m <- lm(value ~ size + acres +   
 size:acres, data = d)

Visualize interaction with ggffects:

# size on x-axis

plot(ggpredict(m, terms =   
 c(“size”, “acres”))

# acres on x-axis

plot(ggpredict(m, terms =   
 c(“acres”, “size”))

**Generate citation for R or R package**

citation() # for R

citation(“dplyr”) # for dplyr