BIS 15L R Markdown Cheatsheet

Lab2

2_1

Is integer?:

is.integer(my_numeric)

Create new object as integer:

my_integer <- as.integer(my_numeric)

Check for NA:

is.na(my_missing)

anyNA(my_missing)

Calculate without NA:

**```{r}**

mean(herbivores$mean.hra.m2, na.rm = T)

**```**

2_2

Generate a sequence of number:

my_vector_sequence <- c(1:100)

Pull out a vector:

Days_of_the_week[3]

Lab 3

3_1:

Build data frame:

Combine vectors

```{r}
hbirds <- data.frame(Sex, Length, Weight)
hbirds
```

Column names:

names(hbirds)

Dimension of the frame:

dim(hbirds)

Structure of the data frame:

str(hbirds)

Rename:

**```{r}**
hbirds <- data.frame(sex = Sex, length_in = Length, weight_oz = Weight) #renaming will become more helpful in later labs
names(hbirds)
**```**

OR

**```{r}**
superhero_info <- rename(superhero_info, gender = "Gender", eye_color = "Eye color", race = "Race", hair_color = "Hair color", height = "Height", pulisher = "Publisher", skin_color = "Skin color", alignment = "Alignment", weight = "Weight")
superhero_info
**```**

Select data:

First row:

hbirds[1,]

Third column:

```{r}
**hbirds[,3]**
```

Select value using $ sign:

```{r}
**w <- hbirds$weight_oz**
**mean(w)**
```

Adding a new column:

~~~
```{r}
hbirds<- rbind(hbirds, new_bird)
hbirds
```
~~~

Writing a csv file:

~~~
```{r}
write.csv(hbirds, "hbirds_data.csv", row.names = FALSE)
```
~~~

3_2:

Change a column to factor and show the level:

~~~
```{r}
hot_springs$scientist <- as.factor(hot_springs$scientist)
levels(hot_springs$scientist)
```
~~~

Summary function:

~~~
```{r}
summary(fish)
```
~~~

Glimpse function:

~~~
```{r}
glimpse(fish)
```
~~~

Number of rows:

~~~
```{r}
~~~

nrow(fish) #the number of rows or observations

Number of columns:

```{r}
**ncol(fish) #the number of columns or variables**
```

Head function:

Give the first n row of the data frame
```{r}
**head(fish, n = 10)**
```

Tail function:

```{r}
**tail(fish, n = 10)**
```

Table function:

Produces fast counts of the number of observations in a variable
```{r}
**table(fish$lakeid)**
```

Filter function:

Pulling out observations that meet specific criteria in a variable
little_fish <- filter(fish, length<=100)

Lab 4

4_1:

Data structure:

```{r}
glimpse(fish)
```

```{r}
str(fish)
```

```{r}
summary(fish)
```

```{r}
names(fish) (Column names)
```

dplyr:

The first package that we will use that is part of the tidyverse is `dplyr`. `dplyr` is used to transform data frames by extracting, rearranging, and summarizing data such that they are focused on a question of interest. This is very helpful, especially when wrangling large data, and makes dplyr one of most frequently used packages in the tidyverse. The two functions we will use most are `select()` and `filter()`.

Select:

select(fish, "lakeid", "scalelength")

To select a range of column:

select(fish, fish_id:length)

To select everything except:

select(fish, -fish_id, -annnumber, -length, -radii_length_mm)

To contain certain characters:

select(fish, contains("length"))

To start with certain characters:

select(fish, starts_with("radii"))

More of them:

matches() = Select columns that match a regular expression
one_of() = Select columns names that are from a group of names
ends_with() = Select columns that end with a character string

Regex:

column contains a letter, followed by a subsequent string
select(fish, matches("a.+er")) # names start with a end with er

Select based on class of data:

select_if(fish, is.numeric)

"Not" a class of data:

select_if(fish, ~!is.numeric(.))

HW:

Change the class of the variables `taxon` and `order` to factors and display their levels.

```{r}
**homerange$taxon <- as.factor(homerange$taxon)**
```

~~~
homerange$taxon
```
~~~

Lab 5

5_1:

Pipes:

shortcut: shift + command + M

```{r}
fish %>%
  select(lakeid, scalelength) %>%
  filter(lakeid == "AL")
```

List all of the superheros that are not human:

```{r}
superhero_info %>%
  filter(race != "Human")
```

Arrange:

arrange(scalelength) (Ascending)

arrange(desc(scalelength)) (Descending)

Mutate:

Mutate allows us to create a new column from existing columns in a data frame

mutate(length_mm = length*10)

mutate_all():

mutate_all(tolower) (Mutate all observations to lowercase)

Specify specific columns:

mutate(across(c("order", "family"), tolower))

Ifelse:

With `ifelse()`, you first specify a logical statement, afterwards what needs to happen if the statement returns `TRUE`, and lastly what needs to happen if it's `FALSE`.

mutate(newborn_new = ifelse(newborn == -999.00, NA, newborn))%>%

Within the parentheses, first comes the condition, next comes what to replace when the condition is met, and the last comes what happens if the condition doesn't meet.

5_2:

Know the data, how do we take out NA, spaces...:

superhero_info <- readr::read_csv("data/heroes_information.csv", *na = c("", "-99", "-")*)

Janitor: help cleans the data, especially renaming columns.

library("janitor")
superhero_powers <- janitor::clean_names(superhero_powers)

Tabyl:

tabyl(superhero_info, alignment) # show both counts and percentages
Within the parentheses, the first argument is the data frame, second is the column.

HW5:

Filter certain row(s) with all TRUE variables.

**```{r}**
superhero_powers %>%
filter(hero_names == "Anti-Spawn") %>%
select_if(all_vars(.=="TRUE"))
**```**

Lab 6

Warmup

Skip 2 rows:

ecosphere <- read_csv("data/ecs21351-sup-0003-SupplementS1.csv", skip=2)

6_1

Skimr:

library("skimr")

Get rid of NA:

filter(!is.na(vore))

Skim:

skim(msleep24)

Histograms:

hist(msleep24$sleep_total_24)

Tabyl use for multi variable:

tabyl(vore, order)

Summarize:

```{r}
msleep %>%
  filter(bodywt > 200) %>%
  summarize(mean_sleep_lg = mean(sleep_total),
            min_sleep_lg = min(sleep_total),
            max_sleep_lg = max(sleep_total),
            total = n()) (Total number of observations)
```

n_distinct():

Presenting the number of distinct observations
summarize(**n_genera=n_distinct(genus)**)

Ex. number of distinct genera over 100 in body weight.

```{r}
msleep %>%
 filter(bodywt > 100) %>%
 summarize(n_genera=n_distinct(genus))
```

Ex. number of genera are represented in the msleep data frame.

```{r}
msleep %>%
 summarize(n_genera=n_distinct(genus))
```

OR

```{r}
n_distinct(msleep$genus)
```

First:

first() (returns first value in a column)

Last:

last() (returns last value in a column)

Group by:

group_by(vore)

6_2

Count:

An easy way of determining how many observations you have within a column.

```{r}
penguins %>%
  count(island, sort = T) #sort=T sorts the column in descending order
```

Count with combination of columns.

```{r}
superhero_powers %>%
  count(accelerated_healing & durability & super_strength)
```

Across multiple variables:

penguins %>%
  count(island, species, sort = T) # sort=T will arrange in descending order

```
```

Use of tabyl for two variables:

~~~
```{r}
penguins %>%
  tabyl(species, island) %>%
  adorn_percentages() %>%
  adorn_pct_formatting(digits = 2) # 2 decimal places
```
~~~