The Skeletons of Guildhall Yard

A statistical analysis of sexual dimorphism

QuantArch Week 7 | 21-03-2022

Universiteit Leiden

Let's start over

RStudio project

Create a new RStudio project

Project: > New Project > New Directory > New Project

Call it **guildhall-yard_analysis**

Now let's organise the project with folders called `data`, `scripts`, and `figures`.

```
sapply(c("data", "scripts", "figures"), dir.create)
```

Project data

The data we'll use for this project are from the Wellcome Osteological Research Database

specifically bone measurements

Download the data to the `data` folder

```
download.file(
  url = "https://www.museumoflondon.org.uk/application/files/7614/6132/6621/MGY_metrics.lst",
  destfile = here("data"),
  mode = "wb"
  )
```

and the metadata (also to the `data` folder)

```
download.file(
  url = "https://www.museumoflondon.org.uk/application/files/6514/7308/7527/RapidMethodRecordingMa
  destfile = here("data"),
  mode = "wb")
```

Data upload

Now that we have the data in our project folder, we can open a new script

File > New File > R Script

Save the script as `01-data_cleaning` in the `scripts` folder ([Ctrl] + [S] or [Cmd] + [S])

At the top of the script we load the necessary packages

```
library(tidyverse)
library(here)
```

Data upload

Now that we have the data in our project folder, we can open a new script

File > New File > R Script

Save the script as `01-data_cleaning` in the `scripts` folder ([Ctrl] + [S] or [Cmd] + [S])

then upload our data

```
library(tidyverse)
library(here)

# Upload data
metric_data_raw <- readr::read_delim(here("data/MGY_metrics.lst"))
```

Data cleaning

Data cleaning

Let's take a look at the data

```
view(metric_data_raw)
```

CEMETERY	SITECODE	PERIOD	LU_INT	E_DATE	L_DATE	CONTEXT	SEX	AGE	TRAIT_T
Medieval-Guildhall Yard	GYE92	11	OA115	1140	1230	10647	UNDETERMINABLE	UNCLASSIFIED ADULT	Bone measure
Medieval-Guildhall Yard	GYE92	11	OA115	1140	1230	10647	UNDETERMINABLE	UNCLASSIFIED ADULT	Bone measure
Medieval-Guildhall Yard	GYE92	11	OA115	1140	1230	10647	UNDETERMINABLE	UNCLASSIFIED ADULT	Bone measure
Medieval-Guildhall Yard	GYE92	11	OA115	1140	1230	10650	FEMALE	ADULT 26-35 YEARS	Tooth measure (Perman

Data cleaning

For our analysis we are interested in the variables `CONTEXT`, `SEX`, `GROUP`, `EXPANSION` and `VALUE`.

Use the metadata PDF to see what these are

```
metric_data_cleaned <- metric_data_raw %>%
  select(CONTEXT, SEX, GROUP, EXPANSION, VALUE)
```

Data cleaning

Now we can get a better overview

```
str(metric_data_cleaned)
```

```
## tibble [1,788 × 5] (S3: tbl_df/tbl/data.frame)
##  $ CONTEXT  : num [1:1788] 10647 10647 10647 10650 10650 ...
##  $ SEX      : chr [1:1788] "UNDETERMINABLE" "UNDETERMINABLE" "UNDETERMINABLE" "FEMALE" ...
##  $ GROUP    : chr [1:1788] "Post-Cranial" "Post-Cranial" "Post-Cranial" "Dental mandibular (permanent
##  $ EXPANSION: chr [1:1788] "HuE1 R" "RaHD R" "ULL1 R" "L M1 buccolingual" ...
##  $ VALUE    : chr [1:1788] "57.7|" "22.3|" "269|" "9.3|" ...
```

Data cleaning

It looks like something is wrong with `VALUE`

It should be numeric, but the `|` at the end of each number is converting it to a string

We can remove all `|`s with the `str_remove` function from the **stringr** package (loaded with tidyverse)

```
args(str_remove)
```

```
## function (string, pattern)
## NULL
```

The argument `string` requires an input vector (i.e. `VALUE`)

and `pattern` requires the pattern to look for (i.e. `|`)

Data cleaning

The default for `pattern` is a regex expression, but that's a separate course in itself...

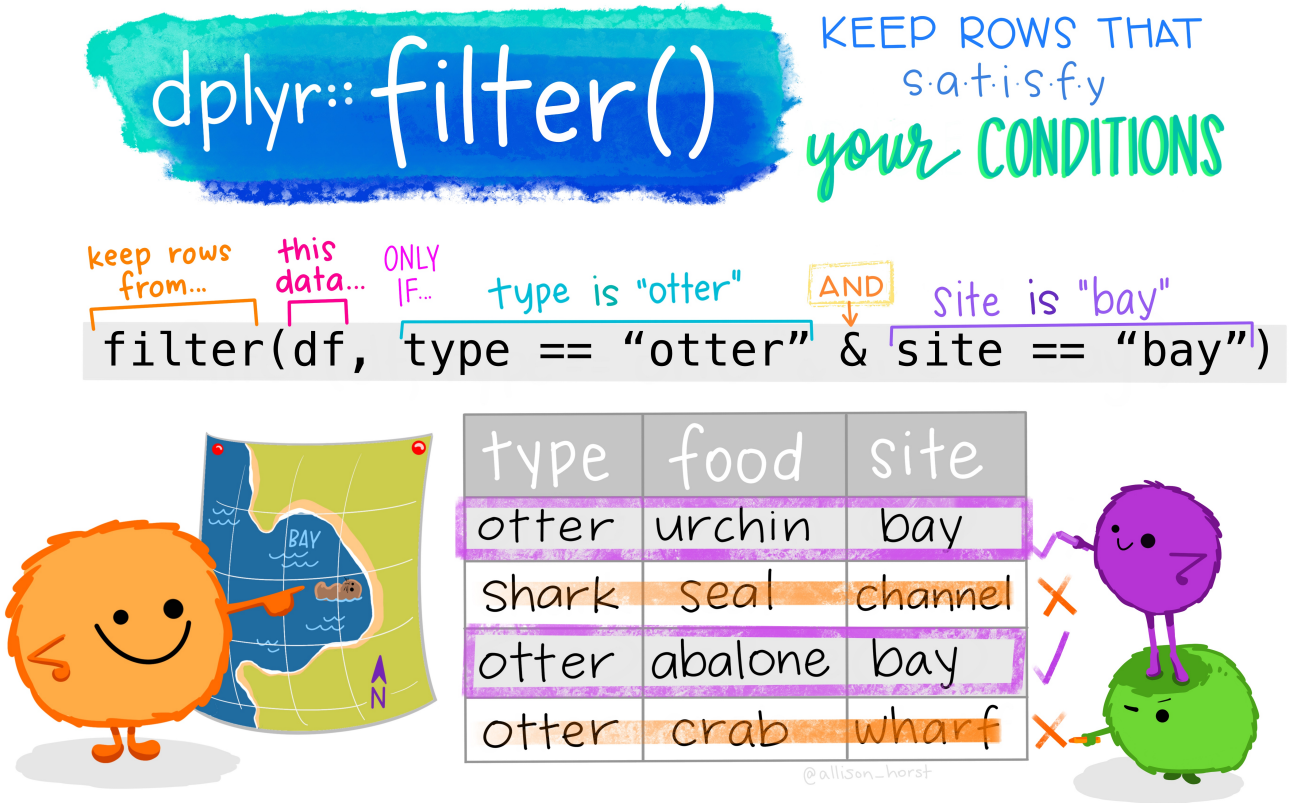
For now we'll use the `coll` function to give it the value we want to remove

```
metric_data_cleaned <- metric_data_raw %>%
  select(CONTEXT, SEX, GROUP, EXPANSION, VALUE) %>%
  mutate(VALUE = str_remove(
    string = VALUE,
    pattern = coll("|")
  ))
```

The regex expression method would be `str_remove(VALUE, "\\|")`

Data cleaning

We are only interested in Post-cranial measurements, and individuals with a determined sex.



Exercise

We are only interested in individuals with a determined sex (💡 take a look at the metadata PDF).

Add the code to our existing code pipe

```
metric_data_cleaned <- metric_data_raw %>%
  select(CONTEXT, SEX, GROUP, EXPANSION, VALUE) %>%
  mutate(VALUE = str_remove(
    string = VALUE,
    pattern = coll("|")
  ))
```

Exercise

Solution

```
metric_data_cleaned <- metric_data_raw %>%
  select(CONTEXT, SEX, GROUP, EXPANSION, VALUE) %>%
  mutate(VALUE = str_remove(
    string = VALUE,
    pattern = coll("|")
  )) %>%
  filter(SEX != "UNDETERMINABLE",
         SEX != "UNSEXED CHILD")
```

Data cleaning

That should do it for now

Let's export the data

```
readr::write_csv(metric_data_cleaned, here("data/MGY_metrics_cleaned.csv"))
```

Data wrangling

Data wrangling

Open a new R Script and save it as `02-data_analysis`

Then we add the required packages and data upload to the script

```
library(tidyverse)
library(here)

# Upload data
metric_data <- readr::read_csv(here("data/MGY_metrics_cleaned.csv"))
```

Data wrangling

We're going to focus on `Post-Cranial` measurements.

```
post_cranial <- metric_data %>%
  filter(GROUP == "Post-Cranial")
```

Let's take a look at the `EXPANSION` variable

```
post_cranial$EXPANSION
```

```
##   [1] "CLL1 L" "GLL R"  "GLB R"  "HuL1 R" "HuHD R" "HuHD L" "HuD1 R" "HuD2 R"
##   [9] "HuE1 R" "RaL L"  "RaHD L" "BW"     "CLL1 R" "GLB R"  "GLB L"  "HuL1 L"
##  [17] "HuHD L" "HuD1 L" "HuD2 L" "HuC L"  "HuE1 R" "HuE1 L" "TaL L"  "FeD1 R"
##  [25] "FeD1 L" "FeD2 R" "FeD2 L" "FeD3 L" "FeD4 L" "FeC L"  "TiL1 L" "TiL2 L"
##  [33] "TiD1 L" "TiD2 L" "TaL R"  "FeHD L" "FeHD R" "SaB"    "BW"     "FeL1 L"
##  [41] "CLL1 R" "GLL R"  "GLB R"  "HuL1 L" "HuHD L" "HuD1 L" "HuD2 L" "HuC L"
##  [49] "HuE1 L" "RaL L"  "RaHD L" "ULL1 L" "FeD1 L" "FeD2 L" "FeD3 L" "FeD4 L"
##  [57] "FeC L"  "FeE L"  "FeHD L" "BW"     "FeL1 R" "RaL L"  "RaHD L" "FeD1 R"
##  [65] "FeD2 R" "FeD3 R" "FeD4 R" "FeC R"  "FeE R"  "FeL2 R" "FeHD R" "GLL L"
##  [73] "GLB L"  "HuL1 R" "HuHD R" "HuD1 R" "HuD2 R" "HuC R"  "FeD1 L" "FeD2 L"
##  [81] "FeHD L" "FeHD R" "SaB"    "BW"     "GLL R"  "GLL L"  "GLB R"  "GLB L"
##  [89] "HuL1 R" "HuL1 L" "HuHD R" "HuHD L" "HuD1 R" "HuD1 L" "HuD2 R" "HuD2 L"
```

Data wrangling

It looks like the side the measurement was taken on is hidden within this variable. Not very convenient.

To make life easier, we will create a new column with side of measurement.

Here, the `case_when` function will come in handy.

Data wrangling

We can make a statement such as

"When the value of EXPANSION contains an 'R', then RIGHT. When the value of EXPANSION contains 'L', then LEFT. When it contains neither, NA"

Data wrangling

"When the value of EXPANSION contains an 'R' at the end of the string, then RIGHT. When the value of EXPANSION contains 'L', then LEFT. When it contains neither, `NA`"

```
case_when(
    str_detect(EXPANSION, "[R]$") ~ "RIGHT",
    str_detect(EXPANSION, "[L]$") ~ "LEFT"))
```

`NA` will be returned if there is no match.

Data wrangling

"When the value of EXPANSION contains an 'R' at the end of the string, then RIGHT. When the value of EXPANSION contains 'L', then LEFT. When it contains neither, `NA`"

💡 regex `[\\sR]$` = match an R preceeded by whitespace (`\\s`) at end of string (`$`).

```
post_cranial %>%
  mutate(
    SIDE = case_when(
      str_detect(EXPANSION, "[\\sR]$"
      str_detect(EXPANSION, "[\\sL]$"
    )
  )
```

CONTEXT	SEX	GROUP	EXPANSION	VALUE	SIDE
10650	FEMALE	Post-Cranial	CLL1 L	133.9	LEFT
10650	FEMALE	Post-Cranial	GLL R	36.8	RIGHT
10650	FEMALE	Post-Cranial	GLB R	21.9	RIGHT
10650	FEMALE	Post-Cranial	HuL1 R	294.0	RIGHT
10650	FEMALE	Post-Cranial	HuHD R	40.0	RIGHT
10650	FEMALE	Post-Cranial	HuHD L	39.9	LEFT
10650	FEMALE	Post-Cranial	HuD1 R	17.7	RIGHT
10650	FEMALE	Post-Cranial	HuD2 R	15.7	RIGHT
10650	FEMALE	Post-Cranial	HuE1 R	54.8	RIGHT
10650	FEMALE	Post-Cranial	RaL L	215.0	LEFT
10650	FEMALE	Post-Cranial	RaHD L	18.1	LEFT
10656	FEMALE	Post-Cranial	BW	52.2	NA
10656	FEMALE	Post-Cranial	CLL1 R	140.7	RIGHT
10656	FEMALE	Post-Cranial	GLB R	25.9	RIGHT
10656	FEMALE	Post-Cranial	GLB L	25.9	LEFT
10656	FEMALE	Post-Cranial	HuL1 L	328.0	LEFT
10656	FEMALE	Post-Cranial	HuHD L	44.5	LEFT

Data wrangling

Now we just need to clean up the `EXPANSION` variable by `str_remove`ing the isolated L's and R's, and `str_trim`ming the whitespace.

```
post_cranial %>%
  mutate(
    SIDE = case_when(
      str_detect(EXPANSION, "[\\sR]$"
      str_detect(EXPANSION, "[\\sL]$"
    ),
    EXPANSION = str_remove(EXPANSION,
    EXPANSION = str_trim(EXPANSION)
  )
```

CONTEXT	SEX	GROUP	EXPANSION	VALUE	SIDE
10650	FEMALE	Post-Cranial	CLL1	133.9	LEFT
10650	FEMALE	Post-Cranial	GLL	36.8	RIGHT
10650	FEMALE	Post-Cranial	GLB	21.9	RIGHT
10650	FEMALE	Post-Cranial	HuL1	294.0	RIGHT
10650	FEMALE	Post-Cranial	HuHD	40.0	RIGHT
10650	FEMALE	Post-Cranial	HuHD	39.9	LEFT
10650	FEMALE	Post-Cranial	HuD1	17.7	RIGHT
10650	FEMALE	Post-Cranial	HuD2	15.7	RIGHT
10650	FEMALE	Post-Cranial	HuE1	54.8	RIGHT
10650	FEMALE	Post-Cranial	RaL	215.0	LEFT
10650	FEMALE	Post-Cranial	RaHD	18.1	LEFT
10656	FEMALE	Post-Cranial	BW	52.2	NA
10656	FEMALE	Post-Cranial	CLL1	140.7	RIGHT
10656	FEMALE	Post-Cranial	GLB	25.9	RIGHT
10656	FEMALE	Post-Cranial	GLB	25.9	LEFT
10656	FEMALE	Post-Cranial	HuL1	328.0	LEFT
10656	FEMALE	Post-Cranial	HuHD	44.5	LEFT

Data wrangling

To increase our sample size, we're going to combine all `FEMALE(?)` and `MALE(?)` estimates.

"When SEX is FEMALE?, then FEMALE.
When SEX is MALE?, then MALE.
Everything else stays as is."

```
post_cranial %>%
  mutate(
    SIDE = case_when(
      str_detect(EXPANSION, "[\\sL]$"
      str_detect(EXPANSION, "[\\sR]$"
    ),
    EXPANSION = str_remove(EXPANSION,
    EXPANSION = str_trim(EXPANSION),
    COMB_SEX = case_when(
      SEX == "FEMALE?" ~ "FEMALE",
      SEX == "MALE?" ~ "MALE",
      TRUE ~ SEX
    )
  )
```

CONTEXT	SEX	GROUP	EXPANSION	VALUE	SIDE	COMB_SEX
10650	FEMALE	Post-Cranial	CLL1	133.9	LEFT	FEMALE
10650	FEMALE	Post-Cranial	GLL	36.8	RIGHT	FEMALE
10650	FEMALE	Post-Cranial	GLB	21.9	RIGHT	FEMALE
10650	FEMALE	Post-Cranial	HuL1	294.0	RIGHT	FEMALE
10650	FEMALE	Post-Cranial	HuHD	40.0	RIGHT	FEMALE
10650	FEMALE	Post-Cranial	HuHD	39.9	LEFT	FEMALE
10650	FEMALE	Post-Cranial	HuD1	17.7	RIGHT	FEMALE
10650	FEMALE	Post-Cranial	HuD2	15.7	RIGHT	FEMALE
10650	FEMALE	Post-Cranial	HuE1	54.8	RIGHT	FEMALE
10650	FEMALE	Post-Cranial	RaL	215.0	LEFT	FEMALE

Data wrangling

We're only interested in using a single measurement from each individual, and we currently have two measurements for some, and one for others.

So, we're going to take all the `RIGHT` measurements, and only use `LEFT` if there is no `RIGHT`, for each skeletal element.

```
post_cranial <- post_cranial %>%
  mutate(SIDE = case_when(
    str_detect(EXPANSION, "[\\sL]$") ~ "LEFT",
    str_detect(EXPANSION, "[\\sR]$") ~ "RIGHT"
    ),
    EXPANSION = str_remove(EXPANSION, "[\\sLR]$"),
    EXPANSION = str_trim(EXPANSION),
    COMB_SEX = case_when(
      SEX == "FEMALE?" ~ "FEMALE",
      SEX == "MALE?" ~ "MALE",
      TRUE ~ SEX)) %>%
  group_by(EXPANSION, CONTEXT) %>%
  slice(which(SIDE == "LEFT" || SIDE == "RIGHT"))
```

💡 The `||` makes sure the condition is satisfied at the first instance, meaning it stops looking after it encounters

Break

Data exploration

Data exploration

Let's `summarise` the data, looking at the `mean` values of males and females,

and throw in `n` and `sd`, too

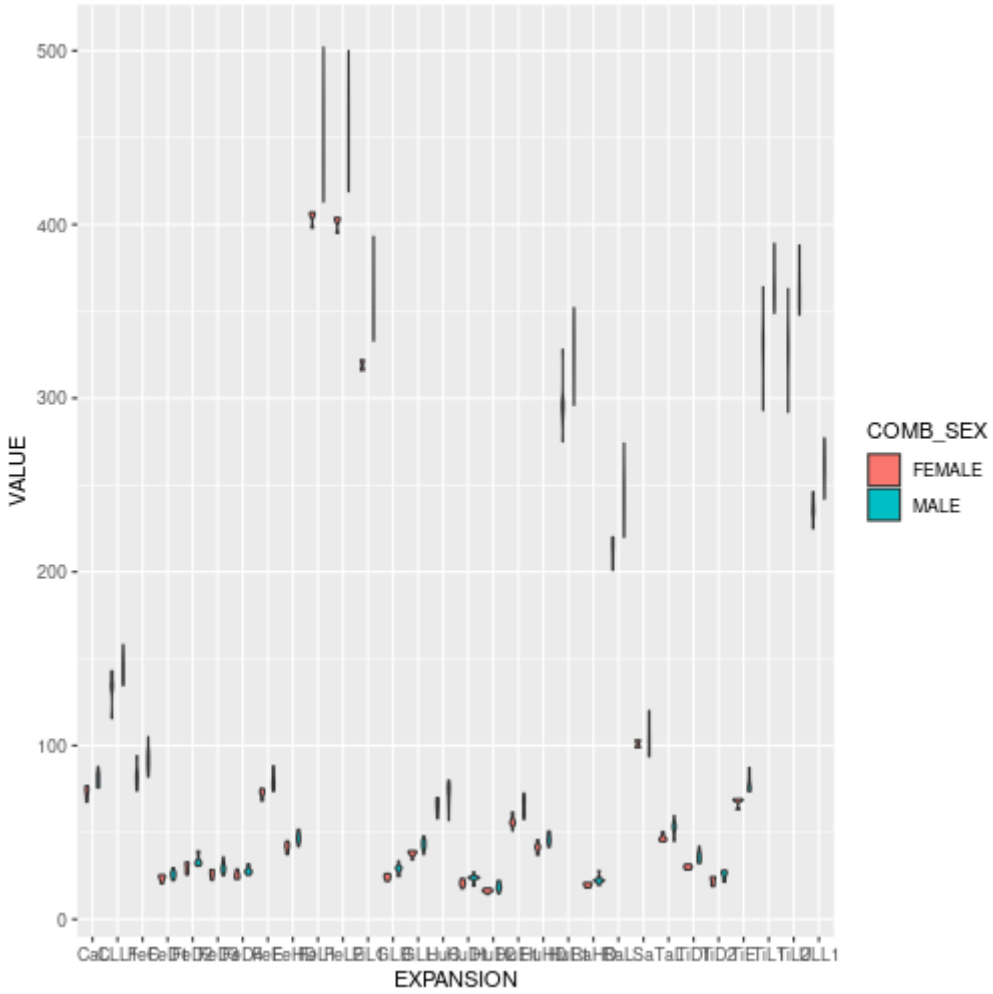
```
post_cranial %>%
  group_by(COMB_SEX) %>%
  summarise(n = n(),
            mean = mean(VALUE),
            sd = sd(VALUE))
```

EXPANSION	COMB_SEX	n	mean	sd
CaL	FEMALE	7	73.10000	3.409790
CaL	MALE	9	80.62222	3.983961
CLL1	FEMALE	6	133.66667	9.594512
CLL1	MALE	10	144.56000	7.728332
FeC	FEMALE	5	82.80000	7.463243
FeC	MALE	13	91.76923	6.260376
FeD1	FEMALE	8	23.61250	1.796375
FeD1	MALE	13	25.78462	2.197668
FeD2	FEMALE	8	29.66250	2.920341
FeD2	MALE	13	33.72308	2.841406
FeD3	FEMALE	5	26.06000	2.522499
FeD3	MALE	13	29.62308	2.998376
FeD4	FEMALE	5	25.48000	2.265392
FeD4	MALE	13	28.02308	1.900944
FeE	FEMALE	4	72.27500	3.015930
FeE	MALE	10	80.14000	4.929097
FeHD	FEMALE	8	41.40000	2.656528

Data exploration

We can also visualise this

```
post_cranial %>%
  ggplot(aes(x = EXPANSION,
             y = VALUE,
             fill = COMB_SEX)) +
    geom_violin()
```



Data exploration

We can also visualise this

```
post_cranial %>%
  ggplot(aes(x = EXPANSION,
             y = VALUE,
             fill = COMB_SEX)) +
    geom_violin()
```

Data exploration

Maybe we should break it into multiple plots...

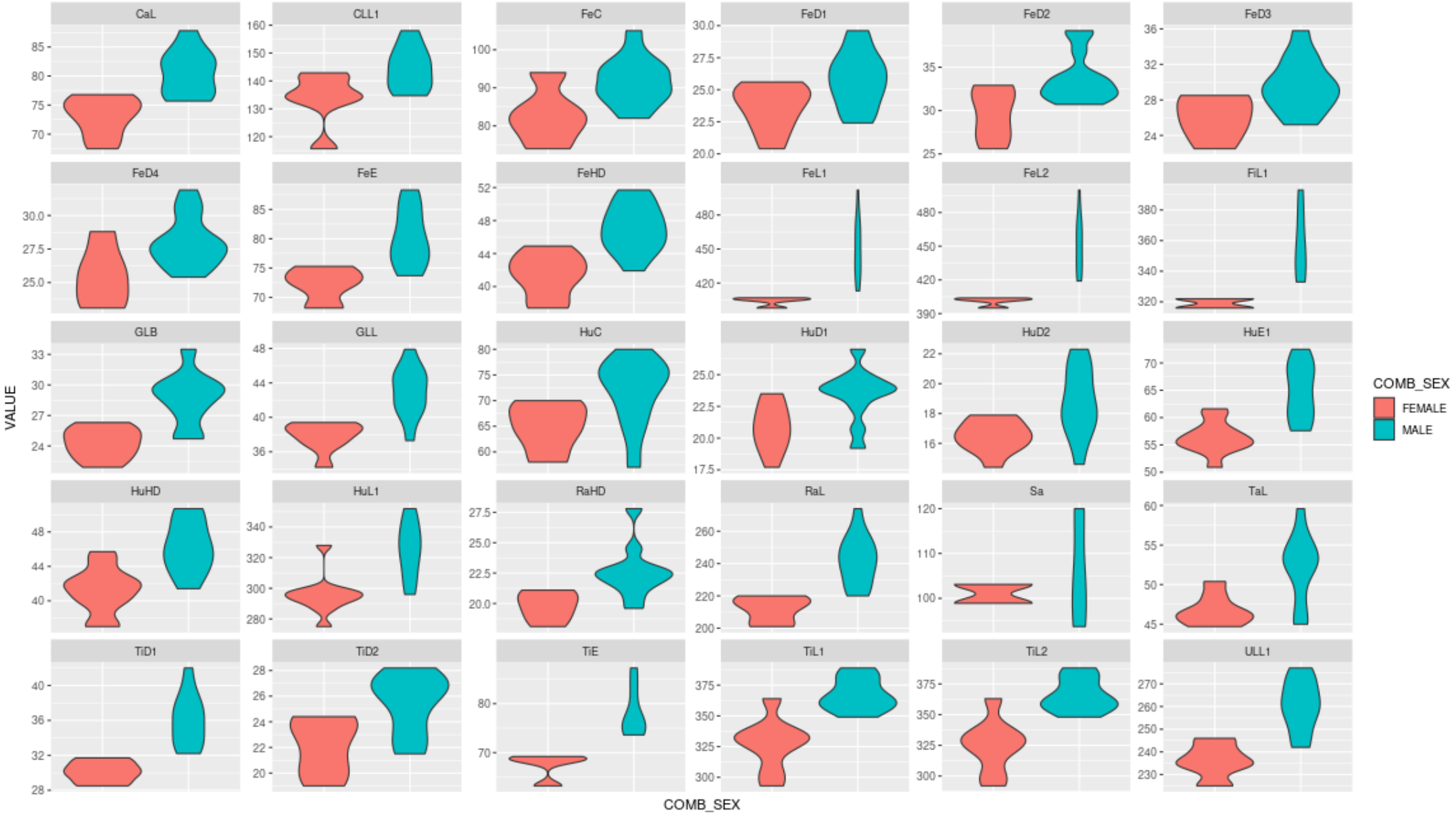
Instead of creating separate plots for each skeletal element,

we can use `facet_wrap` from **ggplot**.

This allows us to specify a variable to separate the plots.

```
post_cranial %>%
  ggplot(aes(x = COMB_SEX,
             y = VALUE,
             fill = COMB_SEX)) +
    geom_violin() +
    facet_wrap(~ EXPANSION, scales = "free_y")
```

💡 `scales = "free_y"` adjusts the scale for each individual plot, instead of all the y-axes being the same.


CaL
CLL1
FeC
FeD1
FeD2
FeD3
FeD4
FeE
FeHD
FeL1
FeL2
FiL1
GLB
GLL
HuC
HuD1
HuD2
HuE1
HuHD
HuL1
RaHD
RaL
Sa
TaL
TiD1
TiD2
TiE
TiL1
TiL2
ULL1
VALUE
COMB_SEX
COMB_SEX
FEMALE
MALE

Data analysis

Data analysis: two-sample t test

Let's look at the difference between males and females within Femoral Head width `FeHD`.

```
femoral_head_split <- post_cranial %>%
  filter(EXPANSION == "FeHD") %>%
  group_by(COMB_SEX) %>%
  group_split()
```

`femoral_head_split` is a list of two data frames. `[[1]]` is the female data frame and `[[2]]` is the male.

Now we can run our `t.test`

```
t.test(femoral_head_split[[1]]$VALUE, femoral_head_split[[2]]$VALUE)
```

Wait… are we violating any of the t test assumptions?

Testing assumptions

T-test assumptions:

- Independence of observations
- Normally(ish) distributed
- Homogeneity of variances
- Random sampling

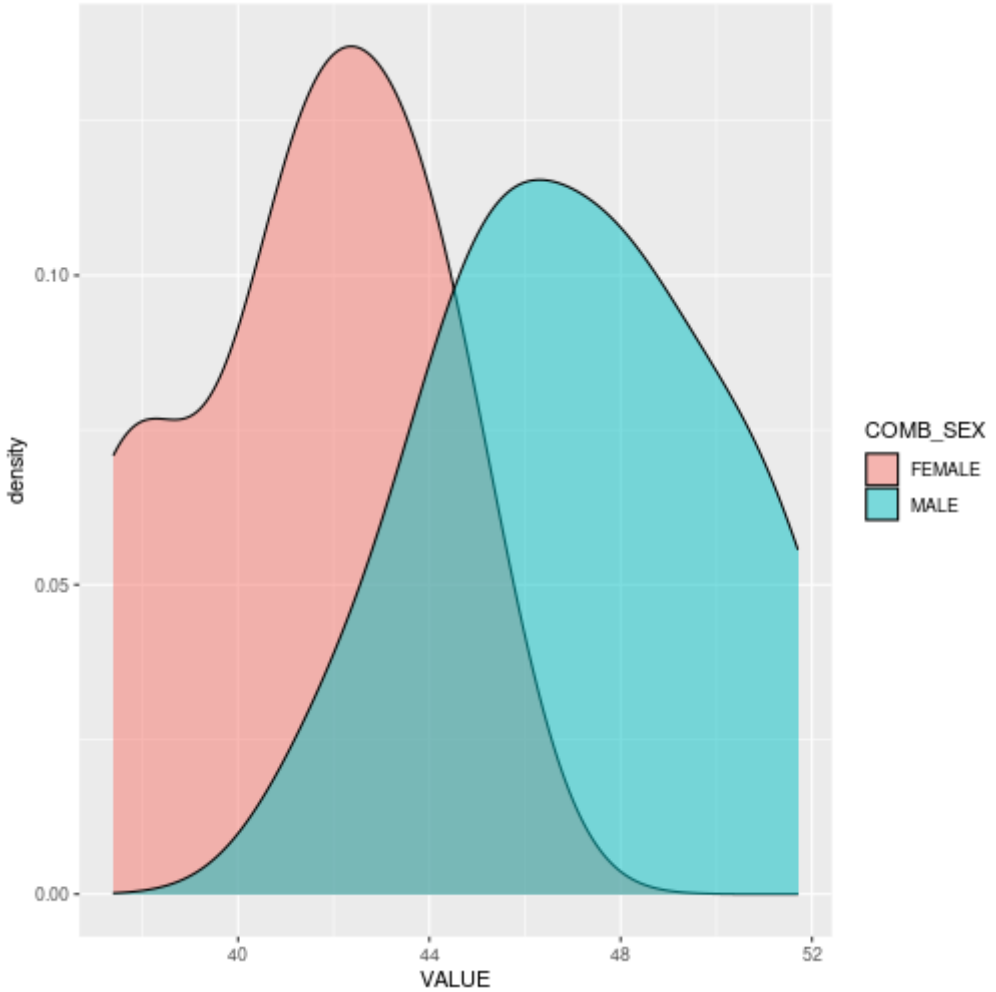
Testing assumptions: Independence of observations

Each measurement is taken from an individual skeleton ✅

Normally distributed

Best option is to visualise

```
post_cranial %>%
  filter(EXPANSION == "FeHD") %>%
  ggplot(aes(x = VALUE, fill = COMB_SEX)) +
    geom_density(alpha = 0.5)
```



Testing assumptions: Normally distributed

Or we can run a `shapiro`-Wilk test, null hypothesis is that the sample is normally distributed

```
shapiro.test(femoral_head_split[[1]]$VALUE)
```

```
##
##     Shapiro-Wilk normality test
##
## data:  femoral_head_split[[1]]$VALUE
## W = 0.94479, p-value = 0.6588
```

```
shapiro.test(femoral_head_split[[2]]$VALUE)
```

```
##
##     Shapiro-Wilk normality test
##
## data:  femoral_head_split[[2]]$VALUE
## W = 0.98011, p-value = 0.98
```

Normally distributed... ✅

Testing assumptions: Homogeneity of variances

We assume homogeneity of sample variances.

In other words, do the two samples have a similar variance?

Again, visualisation can be useful

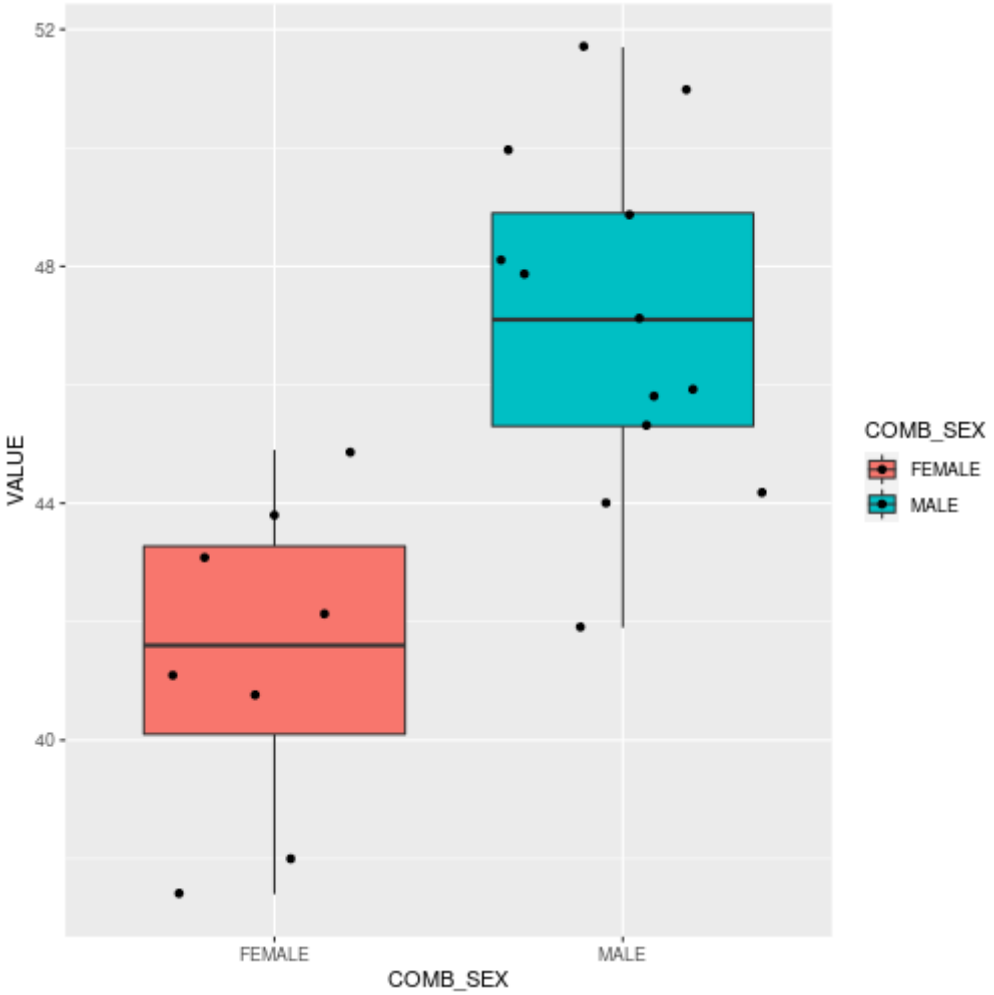
Testing assumptions: Homogeneity of variances

Homogeneity of variances.

In other words, do the two samples have a similar variance?

Again, visualisation can be useful

```
post_cranial %>%
  filter(EXPANSION == "FeHD") %>%
  ggplot(aes(x = COMB_SEX, y = VALUE, fill = C
    geom_boxplot() +
    geom_jitter()
```



Testing assumptions: Homogeneity of variances

Or we can use a `leveneTest` from the **car** package.

The `leveneTest` `y`-argument can take a formula.

Formulas are variables separated by the tilde, `~`. For example `x ~ y`

The `leveneTest` formula is `variable ~ group`, so for us: `VALUE ~ COMB_SEX`

The null hypothesis is that the variances are equal

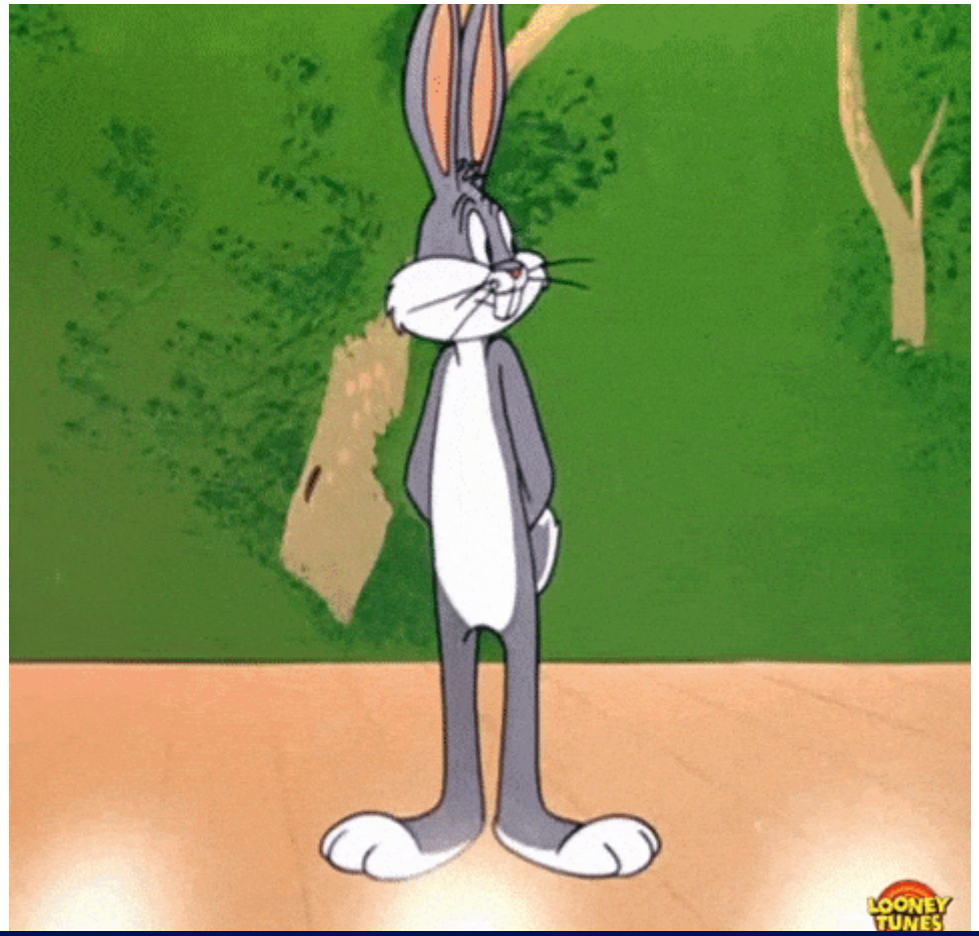
```
post_cranial %>%
  filter(EXPANSION == "FeHD") %>%
  car::leveneTest(y = VALUE ~ COMB_SEX)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  1  0.1544 0.6987
##       19
```

Homogeneity of variances... ✅

Testing assumptions: Random sampling

Were the samples drawn from a random sample?



Testing assumptions

T-test assumptions:

- Independence of observations
- Normally(ish) distributed
- Homogeneity of variances
- Random sampling

Testing assumptions

T-test assumptions:

- Independence of observations ✅
- Normally(ish) distributed ✅
- Homogeneity of variances ✅
- Random sampling

Testing assumptions

T-test assumptions:

- Independence of observations ✅
- Normally(ish) distributed ✅
- Homogeneity of variances ✅
- Random sampling (probably close enough…) 🤷

Two-sample t test

NOW we can run our test.

with `var.equal = TRUE` (equal variances; if not equal => `FALSE`)

```
t.test(femoral_head_split[[2]]$VALUE, femoral_head_split[[1]]$VALUE, var.equal = T)
```

```
##
##     Two Sample t-test
##
## data:  femoral_head_split[[2]]$VALUE and femoral_head_split[[1]]$VALUE
## t = 4.4771, df = 19, p-value = 0.0002583
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  3.014767 8.308310
## sample estimates:
## mean of x mean of y
##  47.06154  41.40000
```

Reporting

The mean femoral head width of males (m = 47.1) is 5.66 mm larger than the mean of females (m = 41.4), $t(19) = 4.48$, $p < 0.001$, 95%CI(3.01, 8.31).