Data Programming with R: Final Project

Conor Heffron (23211267)

Part 1: Analysis

Summary

- The dataset I have selected for this assignment is from the *CBio Portal for Cancer Genomics* website.
- The full data set is in tar / gzip format that can be downloaded directly from https://www.cbioportal.org/study/summary?id=mel_mskimpact_2020.
- I will focus on the clinical data files (data_clinical_patient.txt, data_clinical_sample.txt, and data_mutations.txt), primarily the clinical patient file. These are tab delimited plain text files that can be accessed after the tar is decompressed.
- There is a mix of categorical (factors), continuous and numerical features (some to be inferred by R code).
- This data represents the targeted sequencing (MSK-IMPACT) of 696 melanoma tumour / normal pairs.
- There is a corresponding published medical journal entitled "Therapeutic Implications of Detecting MAPK-Activating Alterations in Cutaneous and Unknown Primary Melanomas".

Reference:

- Shoushtari AN, Chatila WK, Arora A, Sanchez-Vega F, Kantheti HS, Rojas Zamalloa JA, et al. . Therapeutic implications of detecting MAPK-activating alterations in cutaneous and unknown primary melanomas. Clin Cancer Res. (2021) 27:2226–35. doi: 10.1158/1078-0432.CCR-20-4189, PMID: - DOI - PMC - PubMed: https://pubmed.ncbi.nlm.nih.gov/33509808/

- See PMCID link for journal and `Table 1` referenced throughout `Part 1` and `Part 2` of this report, the direct link to this is https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8046739/#S7title.
- Skin related & unclassified primary melanomas often conceal changes that can activate the MAPK pathway. The disease pathway describes the mechanisms by which it evolves & either persists or is resolved. The MAPK pathway activation is a prominent step in melanoma pathogenesis.
- The targeted capture of multi-gene sequencing can detect oncogenic RTK-RAS-MAPK pathway alterations (can cause development of a tumour or tumours) in almost all cutaneous and unknown primary melanomas.

Load packages & re-usable source code / functions.

```r
source('final_proj_imports.R')
```

```
Loading required package: ggplot2


Attaching package: 'plotly'

The following object is masked from 'package:ggplot2':

    last_plot

The following object is masked from 'package:stats':

    filter

The following object is masked from 'package:graphics':

    layout


Attaching package: 'rio'

The following object is masked from 'package:plotly':

    export
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.3     v stringr   1.5.0
v forcats   1.0.0     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.2
[conflicted] Will prefer dplyr::filter over any other package.
[conflicted] Will prefer dplyr::lag over any other package.
```

```r
source('final_proj_utils.R')# Import and load source code from local R script
```

```
[conflicted] Removing existing preference.
[conflicted] Will prefer dplyr::filter over any other package.
[conflicted] Removing existing preference.
[conflicted] Will prefer dplyr::lag over any other package.
```

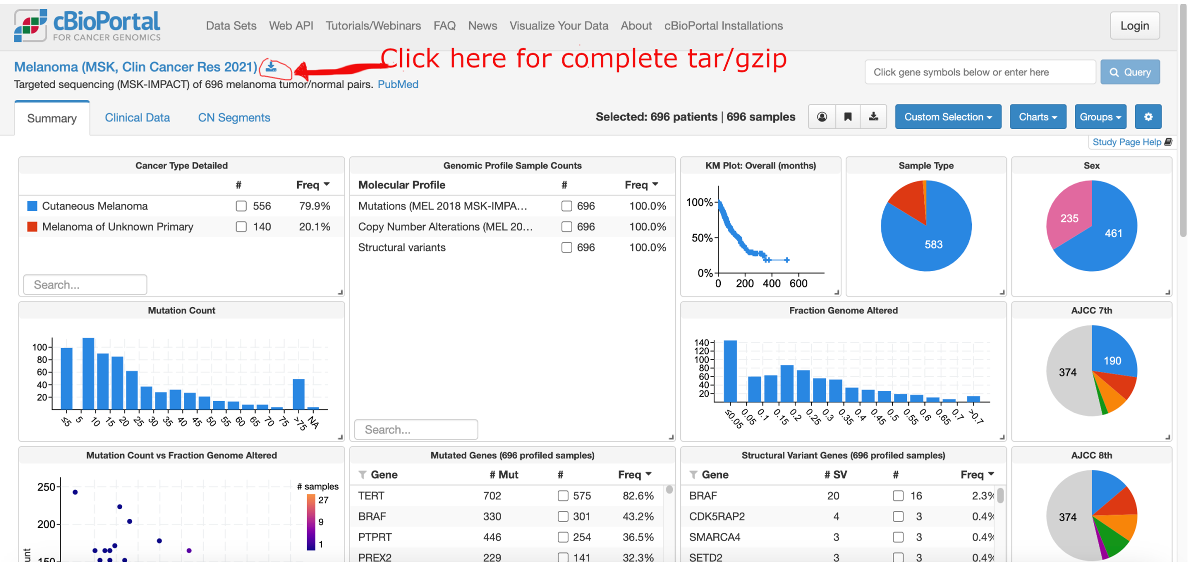
Prepare working directory and data for import

- Set working directory to `path_wd`
- The working directory should contain .qmd, *.R and tar/gzip files before executing `final-project-23211267.qmd` file.

```r
path_wd <- "~/Library/CloudStorage/GoogleDrive-conor.heffron@ucdconnect.ie/My Drive/UCD/MS

# setwd('Path/To/Your/Folder')
setwd(path_wd)
```

Decompress Data Directory

- Untar data downloaded directly from https://www.cbioportal.org/study/summary?id=mel_mskimpact_2020



Click here for complete tar/gzip

Melanoma (MSK, Clin Cancer Res 2021)

Targeted sequencing (MSK-IMPACT) of 696 melanoma tumor/normal pairs. PubMed

Selected: 696 patients | 696 samples

Cancer Type Detailed

	#	Freq
🟦 Cutaneous Melanoma	□ 556	79.9%
🟥 Melanoma of Unknown Primary	□ 140	20.1%

Genomic Profile Sample Counts

Molecular Profile	#	Freq
Mutations (MEL 2018 MSK-IMPA…	□ 696	100.0%
Copy Number Alterations (MEL 20…	□ 696	100.0%
Structural variants	□ 696	100.0%

Mutated Genes (696 profiled samples)

Gene	# Mut	#	Freq
TERT	702	□ 575	82.6%
BRAF	330	□ 301	43.2%
PTPRT	446	□ 254	36.5%
PREX2	229	□ 141	32.3%

Structural Variant Genes (696 profiled samples)

Gene	# SV	#	Freq
BRAF	20	□ 16	2.3%
CDK5RAP2	4	□ 3	0.4%
SMARCA4	3	□ 3	0.4%
SETD2	3	□ 3	0.4%

Untar / decompress tarball / gzip file

```
dir_name <- "mel_mskimpact_2020"
extension <- ".tar.gz"
untar(paste(dir_name, extension, sep=""), files = NULL, list = FALSE, exdir = ".",
      extras = NULL, verbose = FALSE,
      restore_times =  TRUE,
      support_old_tars = Sys.getenv("R_SUPPORT_OLD_TARS", FALSE),
      tar = Sys.getenv("TAR"))
```

Import Relevant Data with genric function from `final_proj_utils.R`

```
data_files <- import_skin_cancer_data(dir_name, "^data_")
```

```
[1] "data_clinical_patient.txt - importing clinical data"
[1] "data_clinical_sample.txt - importing clinical data"
[1] "data_cna_hg19.seg is not needed for import..."
[1] "data_cna.txt - importing other data"
[1] "data_gene_panel_matrix.txt - importing other data"
[1] "data_mutations.txt - importing other data"
[1] "data_sv.txt - importing other data"
```

List data sets that were impoprted

```
names(data_files)
```

```
[1] "data_clinical_patient"  "data_clinical_sample"   "data_cna"
[4] "data_gene_panel_matrix" "data_mutations"         "data_sv"
```

Merge data sets (clinical patient, clinical sample & data gene panel matrix as they have the same number of observations at 696 each)

```
# Merge by patient ID
df_clin <- merge(x = data_files$data_clinical_patient, y = data_files$data_clinical_sample

# Merge / Join by Sample ID
df_clin <- merge(x = df_clin, y = data_files$data_gene_panel_matrix, by = "SAMPLE_ID", all
```

Data manipulation (add columns, add proportion percentages, omit NAs etc.)

```
# Add column to represent proportion as a percentage for the male:female samples
df_clin <- df_clin %>%
    group_by(SEX) %>%
    mutate(gender_prop =  100 * round(sum(n() / length(df_clin$SEX)), 2)) %>%
    ungroup

# Add column to represent proportion as a percentage  for the primary cancer site
# samples
df_clin <- df_clin %>%
    group_by(DMT_PRIMARY_SITE) %>%
    mutate(site_prop =  100 * round(sum(n() / length(df_clin$DMT_PRIMARY_SITE)), 2)) %>%
    ungroup

# Add column to represent proportion as a percentage the primary cancer site
# classified (generic site) samples
df_clin <- df_clin %>%
    group_by(DMT_PRIMARY_SITE_CLASSIFIED) %>%
    mutate(site_classified_prop =  100 * round(sum(n() / length(df_clin$DMT_PRIMARY_SITE_C
    ungroup
```

```
# extract OS years from OS months column
df_clin$OS_YEARS <- round(df_clin$OS_MONTHS / 12)

# estimate patient current age by adding age at diagnosis + OS years
df_clin$EST_CURR_AGE <- df_clin$OS_YEARS + df_clin$AGE_AT_INITIAL_DIAGNOSIS

# Extract 2 columns from OS_STATUS where OS_STATUS_INT is 1 or 0
# and OS_STATUS_GRP is Living or Deceased
df_clin <- df_clin %>% separate_wider_delim(OS_STATUS, ":", names = c("OS_STATUS_INT", "OS

# Add column to represent age group - this column is an ordered factor based on estimated
df_clin$age_group = cut(df_clin$EST_CURR_AGE, c(0, 5, 15, 25, 35, 45, 55, 65, 75, 85, 95,

# There NA values but they do not hamper any calculation that I have replicated so comment
# na.omit(df_clin)
```

Journal Statistics (Part 1): Tables / Numeric Summaries of Data

- See link to article `Table 1` statistics that I have replicated in R code. This is the 1st section of data analytics https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8046739/#S7title.

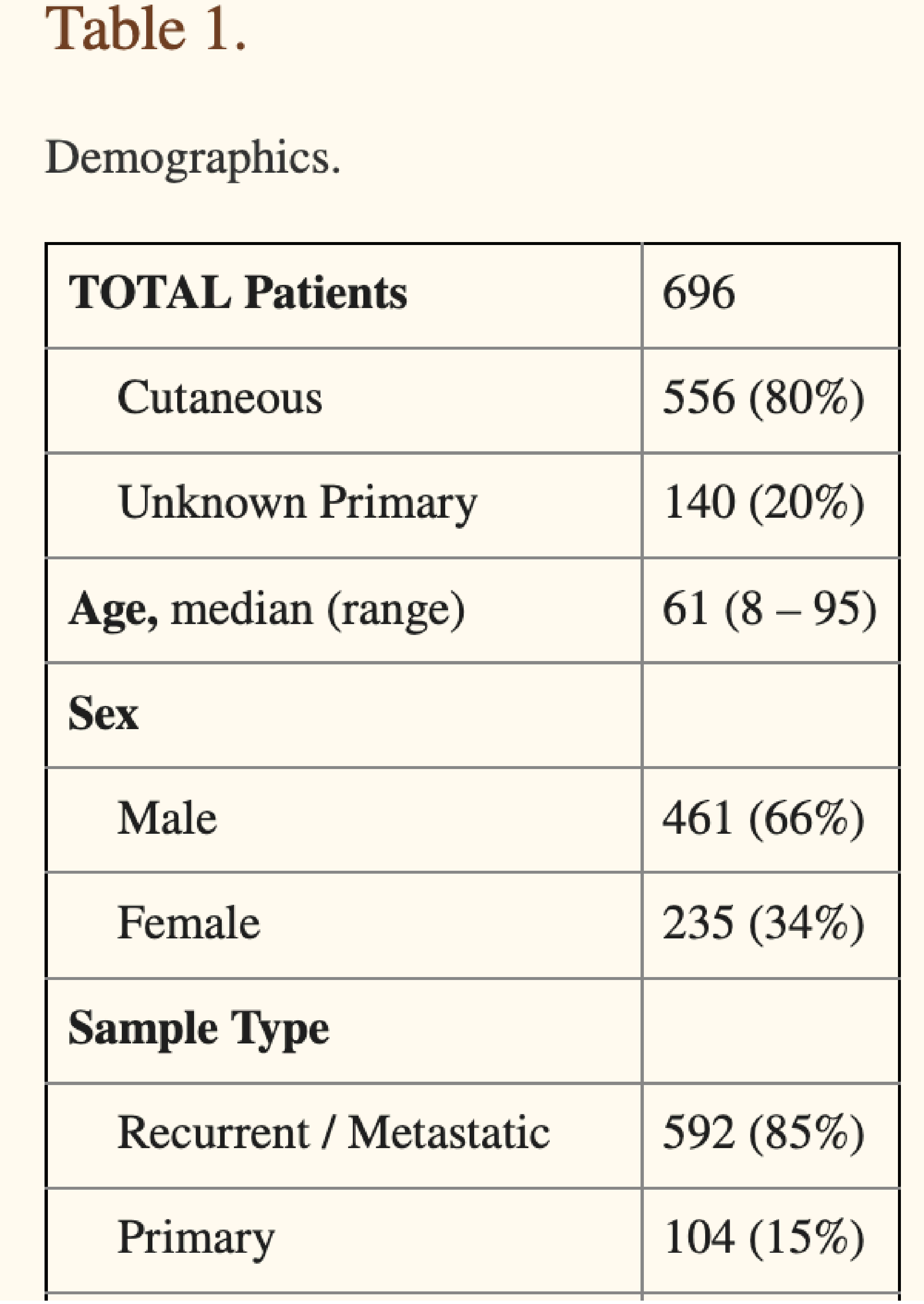


Table 1.

Demographics.

TOTAL Patients	696
Cutaneous	556 (80%)
Unknown Primary	140 (20%)
Age, median (range)	61 (8 – 95)
Sex	
Male	461 (66%)
Female	235 (34%)
Sample Type	
Recurrent / Metastatic	592 (85%)
Primary	104 (15%)

Get main data frame `df_clin` dimensions (696 observations x 39 variables)

- This equates to 696 patients with sample data and gene matrix values mapped.

```
dim(df_clin)
```

```
[1] 696  39
```

Create summary table with count by group aggregation that adds count variable `n` and count proportion as a percentage (variable `Freq`).

- This is done by calling a genric function from `final_proj_utils.R`
- The grouping field here is `CANCER_TYPE_DETAILED`
- The default settings will be used for optional `count_agg` parameters

```
count_agg(df_clin, "CANCER_TYPE_DETAILED")
```

CANCER_TYPE_DETAILED	n	Freq
Cutaneous Melanoma	556	80
Melanoma of Unknown Primary	140	20

Get rounded median value for `AGE_AT_INITIAL_DIAGNOSIS`

- The value of 61 matches `Table 1` from referenced journal.

```
round(median(df_clin$AGE_AT_INITIAL_DIAGNOSIS))
```

```
[1] 61
```

Create summary table with count by group aggregation that adds count variable `n` and count proportion as a percentage (variable `Freq`).

- This is done by calling a genric function from `final_proj_utils.R`
- The grouping field here is `SEX`
- Male samples make up two thirds of the data for analysis

```
count_agg(df_clin, "SEX")
```

SEX	n	Freq
Male	461	66
Female	235	34

Create summary table with count by group aggregation that adds count variable `n` and count proportion as a percentage (variable `Freq`).

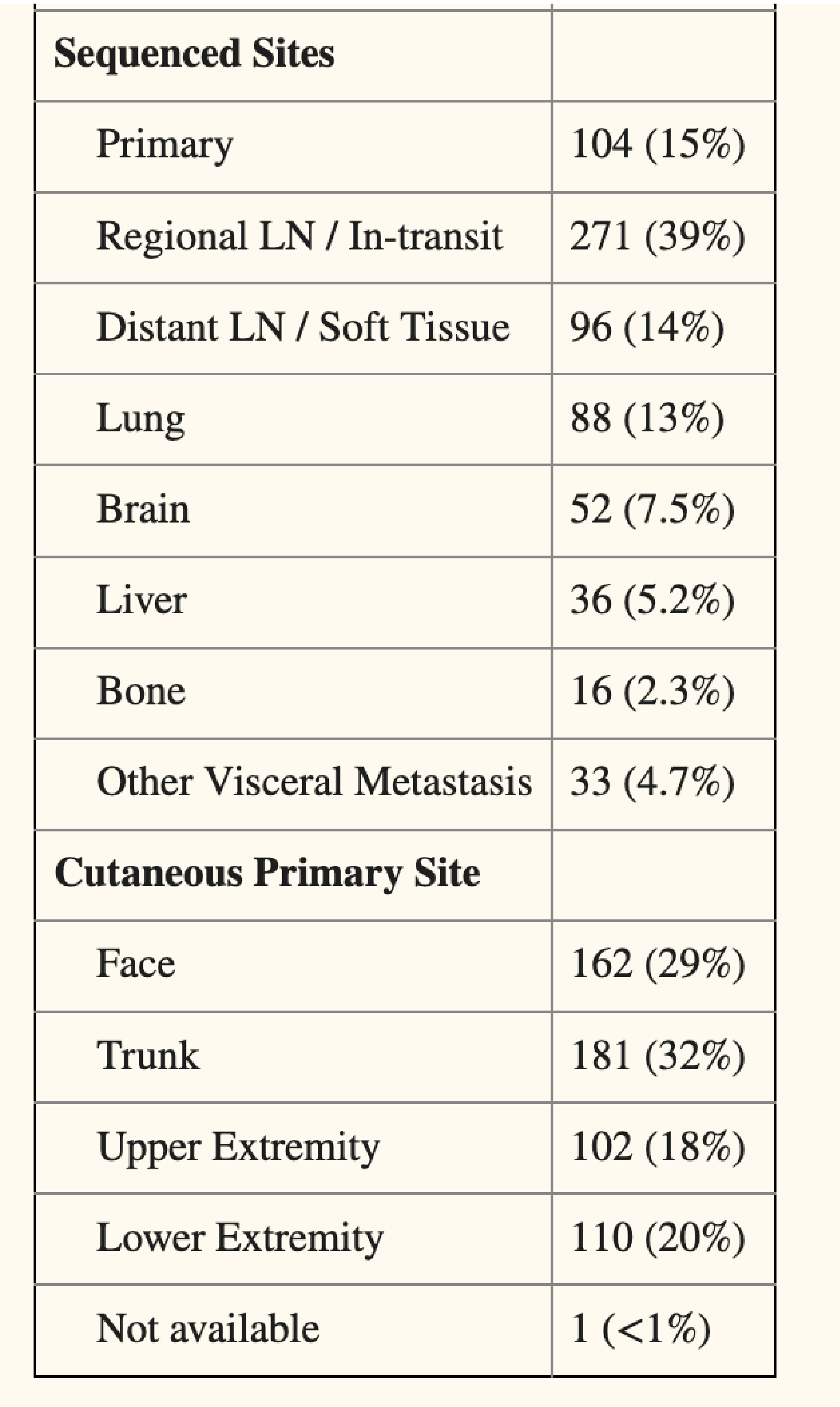
- This is done by calling a genric function from `final_proj_utils.R`
- The grouping field here is `SAMPLE_TYPE`
- 84% of samples are not from the primary or neighbouring / local recurrence site of melanoma

```
count_agg(df_clin, "SAMPLE_TYPE")
```

SAMPLE_TYPE	n	Freq
Metastasis	583	84
Primary	104	15
Local Recurrence	9	1

Journal Statistics (Part 2)

- See link to article `Table 1` statistics that I have replicated in R code. This is the first section of data analytics in this report - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8046739/#S7title.



Sequenced Sites	
Primary	104 (15%)
Regional LN / In-transit	271 (39%)
Distant LN / Soft Tissue	96 (14%)
Lung	88 (13%)
Brain	52 (7.5%)
Liver	36 (5.2%)
Bone	16 (2.3%)
Other Visceral Metastasis	33 (4.7%)
Cutaneous Primary Site	
Face	162 (29%)
Trunk	181 (32%)
Upper Extremity	102 (18%)
Lower Extremity	110 (20%)
Not available	1 (<1%)

.

Create summary table with count by group aggregation that adds count variable `n` and count proportion as a percentage (variable `Freq`).

- This is done by calling a genric function from `final_proj_utils.R`
- The grouping field here is `SIMPLIFIED_METASTATIC_SITE`
- The most common sites for cancers to metastasize include the lungs, liver, bones and brain.
- From this sample set, `Regional Local Recurrences (LR) / In-Transit / Lymph Node (LN)` is by far the most common at 39% followed by `Soft Tissue / Distant LN` and `Lung` at 14% and 13% respectively.
- `NA` here corresponds to `Primary` site in `Table 1`

```
count_agg(df_clin, "SIMPLIFIED_METASTATIC_SITE")
```

SIMPLIFIED_METASTATIC_SITE	n	Freq
LR / In-Transit / Regional LN	271	39
NA	104	15
Soft Tissue / Distant LN	96	14
Lung	88	13
Brain	52	7
Liver	36	5
Other Visceral	33	5
Bone	16	2

Create summary table with count by group aggregation that adds count variable `n` and count proportion as a percentage (variable `Freq`).

- This is done by calling a genric function from `final_proj_utils.R`
- The grouping field here is `DMT_PRIMARY_SITE_CLASSIFIED`
- The author applies a filter to this data where `CANCER_TYPE_DETAILED == "Cutaneous Melanoma"` before creating summary statistics. This makes sense because we want to rule out the patients with unknown primary cancer site/area as this will skew the result.
- There are 140 patients (20%) with unknown DMT primary cancer site which we want to rule out before reporting the proportion of patients per generic primary cancer site.
- The `count_agg` function has an optional parameter for digits (number of decimals) which has default value of `0` but here we explicitly set `digits = 2` for a more accurate `Freq` (group size proportion as a percentage based on count variable `n`) to match the counts reported.
- There is a relatively equaly split here where each site has a share between 18% and 32%.
- `Trunk` and `Head` are the top 2 results from this summary.

```
count_agg(df_clin |> filter(CANCER_TYPE_DETAILED == "Cutaneous Melanoma"), "DMT_PRIMARY_SI
```

DMT_PRIMARY_SITE_CLASSIFIED	n	Freq
Trunk	181	32.55
Head	162	29.14
Lower Extremity	110	19.78
Upper Extremity	102	18.35
NA	1	0.18

Further Exploratory Analysis (*NOT* based on `Table 1`)

Create summary table with count by group aggregation that adds count variable `n` and count proportion as a percentage (variable `Freq`).

- This is done by calling a genric function from `final_proj_utils.R`
- The grouping field here is `DMT_PRIMARY_SITE`
- The `count_agg` has an optional parameter for digits (number of decimals) which has default value of `0` but here we explicitly set `digits = 2` for a more accurate `Freq` (group size proportion as a percentage based on count variable `n`)
- This summary table goes into more detail on where the site is, giving further detail on the lower and upper extremities in particular.
- Unfortunately, `Unknown Primary` observations account for 20% of the data set.
- However there are more than 20 records for neck, face, scalp, arm/shoulder, leg/hip and trunk. Trunk again is the top result at 26% which may be expected to a certain extent as it is a large component of the human body.

```
count_agg(df_clin, "DMT_PRIMARY_SITE", digits = 2)
```

DMT_PRIMARY_SITE	n	Freq
Skin, trunk	180	25.86
Unknown Primary	140	20.11
Skin, leg/hip	104	14.94
Skin, arm/shoulder	99	14.22
Skin, scalp	60	8.62
Skin, face	52	7.47
Skin, neck	28	4.02
Skin, external ear	15	2.16
Skin, eyelid	6	0.86
Skin, foot	6	0.86

DMT_PRIMARY_SITE	n	Freq
Skin, hand	3	0.43
Skin, Site Unspecified	1	0.14
Skin, lip	1	0.14
NA	1	0.14

Create summary table with count by group aggregation that adds count variable `n` and count proportion as a percentage (variable `Freq`).

- This is done by calling a genric function from `final_proj_utils.R`
- This is the first summary of mutations data
- The grouping field here is `Hugo_Symbol`
- The `count_agg` has an optional parameter for n_results (number of results returned after sorting results set by `n` in descending order) which by default will return first 20 results (includes each group). There are many `Hugo_Symbol` values and I am only interested in the top 15 occurences so we set `n_results = 15`
- The `count_agg` has an optional parameter for digits (number of decimals) which has default value of `0` but here we explicitly set `digits = 2` for a more accurate `Freq` (group size proportion as a percentage based on count variable `n`)
- One of the most common changes in melanoma cells is a mutation in the BRAF oncogene, which is found in approx. half of diagnosed melanomas.
- Other genes that can be affected in melanoma include NRAS, CDKN2A & NF1 (Usually only 1 of these genes is affected).
- We can see BRAF (# 5), NRAS (# 15), CDKN2A (# 4) and NF1 (# 7)
- The top 3 in this data set consist of: TERT, PTPRT and GRIN2A
- It appears the BRAF mutation is associated with aggressive tumor features & can increase the risk of persistent and recurrent disease
- I think this is a promising subset of data to examine further for discerning data features or even prinicpal component analysis in the mutations data frame which has a column of values per patient.

```r
count_agg(data_files$data_mutations, "Hugo_Symbol", 15, 2, F)
```

rank H	ugo_Symbol	n	Freq
1	TERT	702	3.60
2	PTPRT	446	2.29
3	GRIN2A	351	1.80
4	CDKN2A	349	1.79
5	BRAF	330	1.69
6	PTPRD	324	1.66

rank H	ugo_Symbol	n	Freq
7	NF1	302	1.55
8	ROS1	297	1.52
9	PAK7	261	1.34
10	ERBB4	230	1.18
11	PIK3C2G	229	1.18
12	PREX2	229	1.18
13	KMT2D	223	1.14
14	TP53	214	1.10
15	NRAS	211	1.08

Get dimensions of mutations data set

- We can see there are 19,479 observations or records
- We can see there is now 122 variables or columns
- The top 5 symbols in order of count were: TERT, PTPRT, GRIN2A, *CDKN2A*, *BRAF*

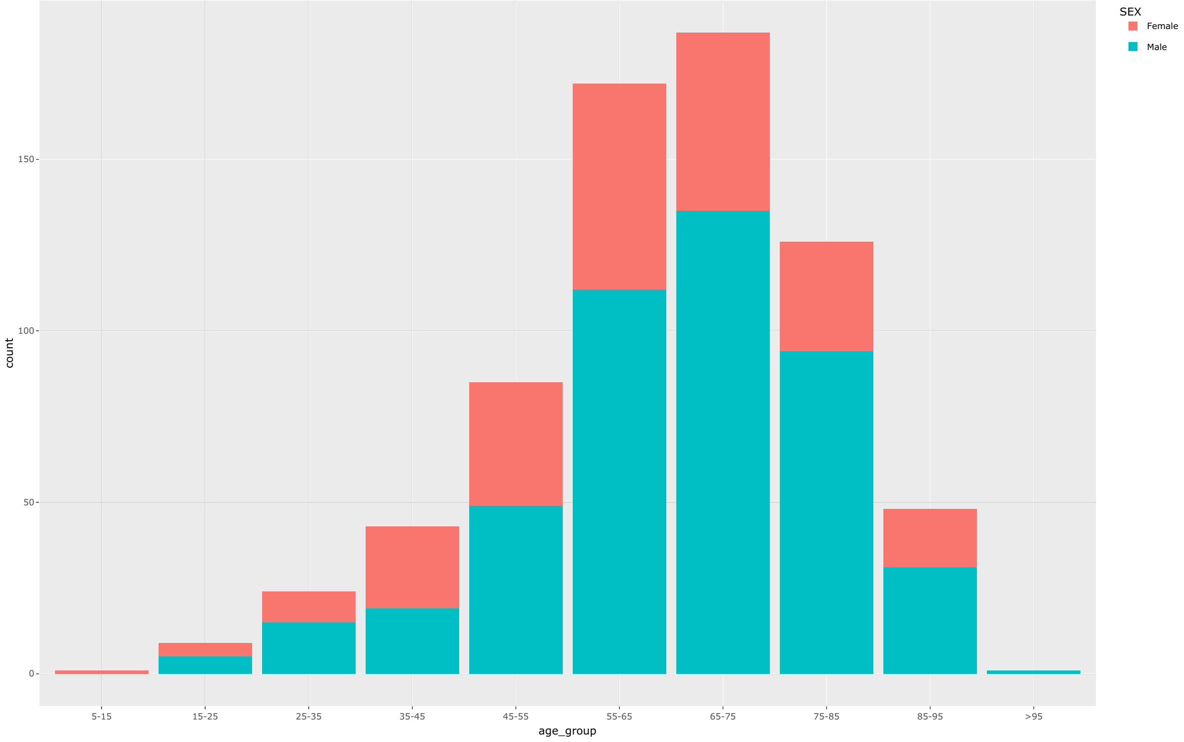
```
dim(data_files$data_mutations)
```

```
[1] 19479   122
```

Graphical Summaries / Data Visualisation

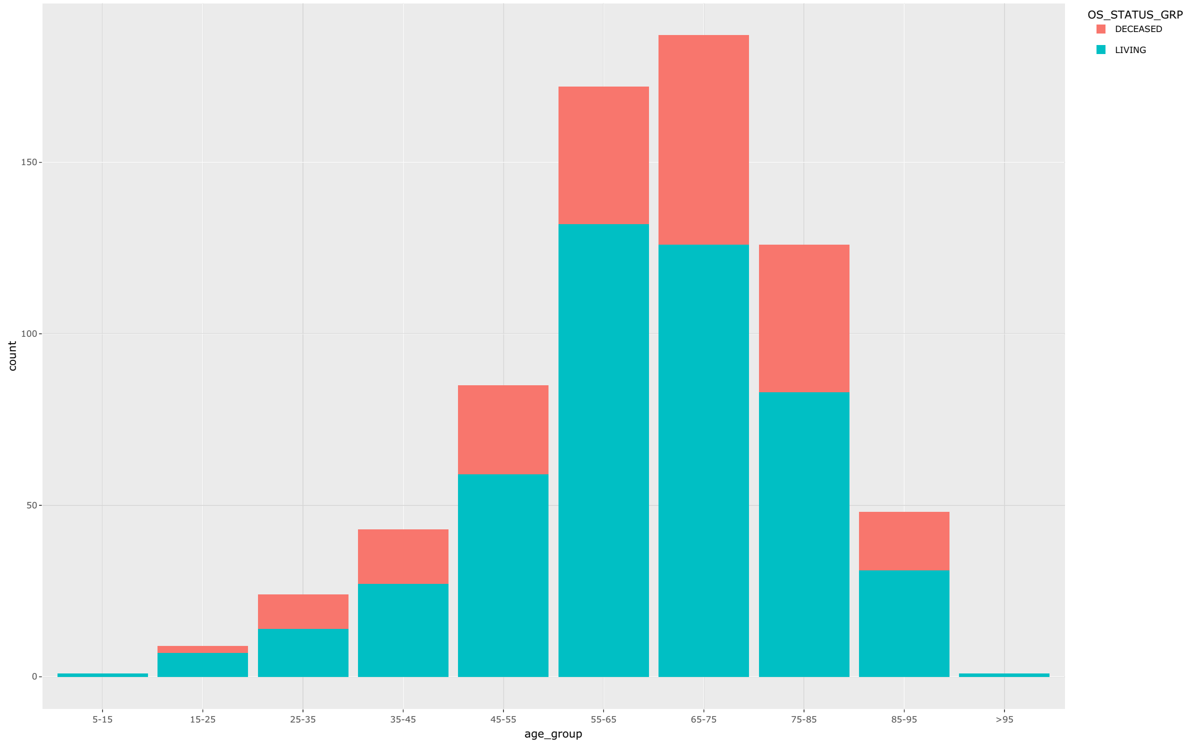
- I would like to create some data visualisations to further explore the summary information from `Table 1`.
- I am interested in the split for `age_group` (Ordinal factor) and `SEX` (Male or Female - categporical).
- The majority of samples are `Male` and lie between age groups `45-55` -> `75-85`

```
ggplotly(ggplot(df_clin, aes(x=age_group, fill = SEX)) +
  geom_bar())
```



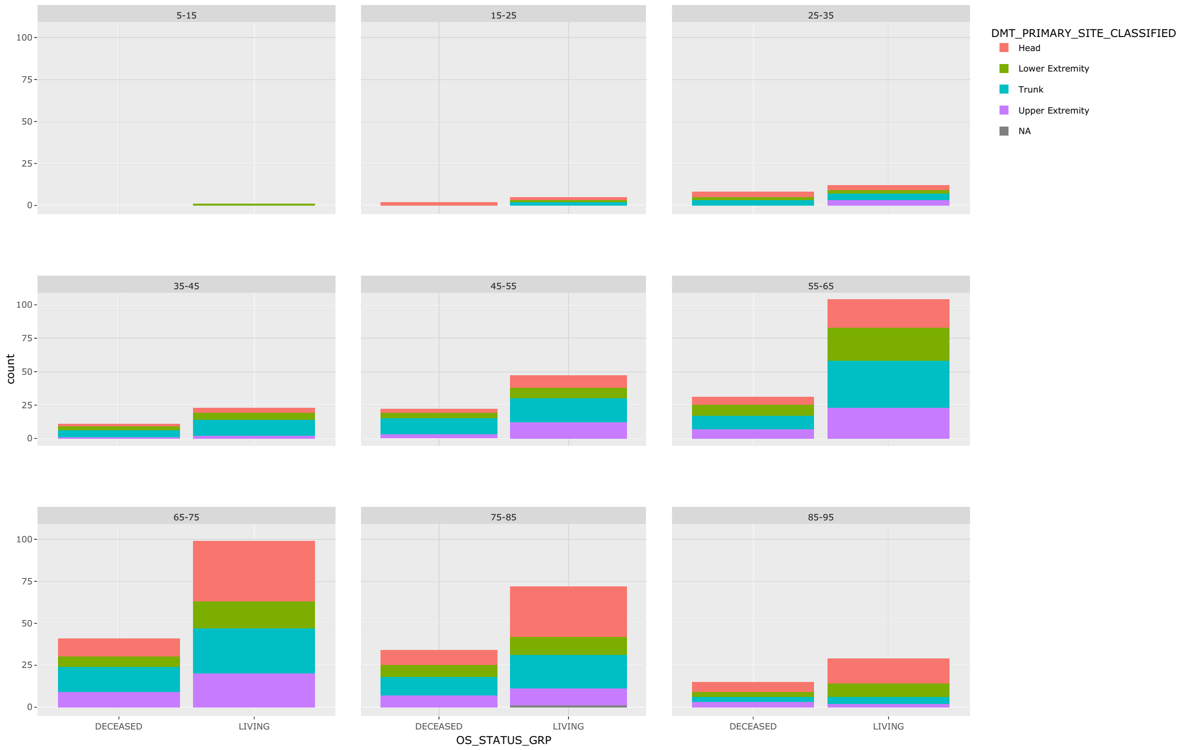
- I am interested in the split for `OS_STATUS` (Living or Deceased) and `age_group` (Ordinal factor)
- The largest group of deceased patient samples are in the age bracket of `65-75`.

```
ggplotly(ggplot(df_clin, aes(x=age_group, fill = OS_STATUS_GRP)) +
  geom_bar())
```



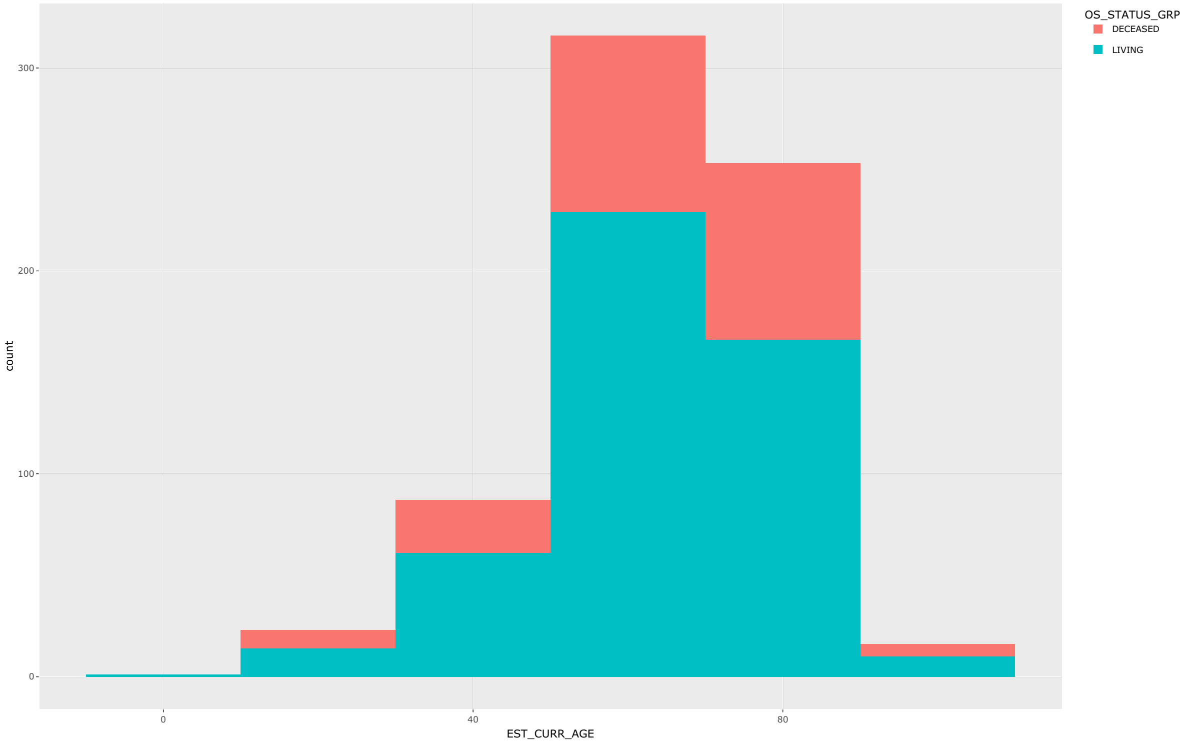
- I am interested in the split for `OS_STATUS_GRP` (Living or Deceased) and `DMT_PRIMARY_SITE_CLASSIFIED` (categorical feature)
- This shows a lot of samples for living patients that were diagnosed with skin cancer in the head area, especially 65-85.
- The area with a high proportion of deceased patients were diagnosed with skin cancer in trunk or torso area across all age groups followed by the Head area. In the 15-25 age group, the deceased appear to all have Head diagnosis as primary site of melanoma.

```
ggplotly(ggplot(df_clin |> filter(CANCER_TYPE_DETAILED == "Cutaneous Melanoma"), aes(x=OS_
  geom_bar() + facet_wrap(~age_group))
```



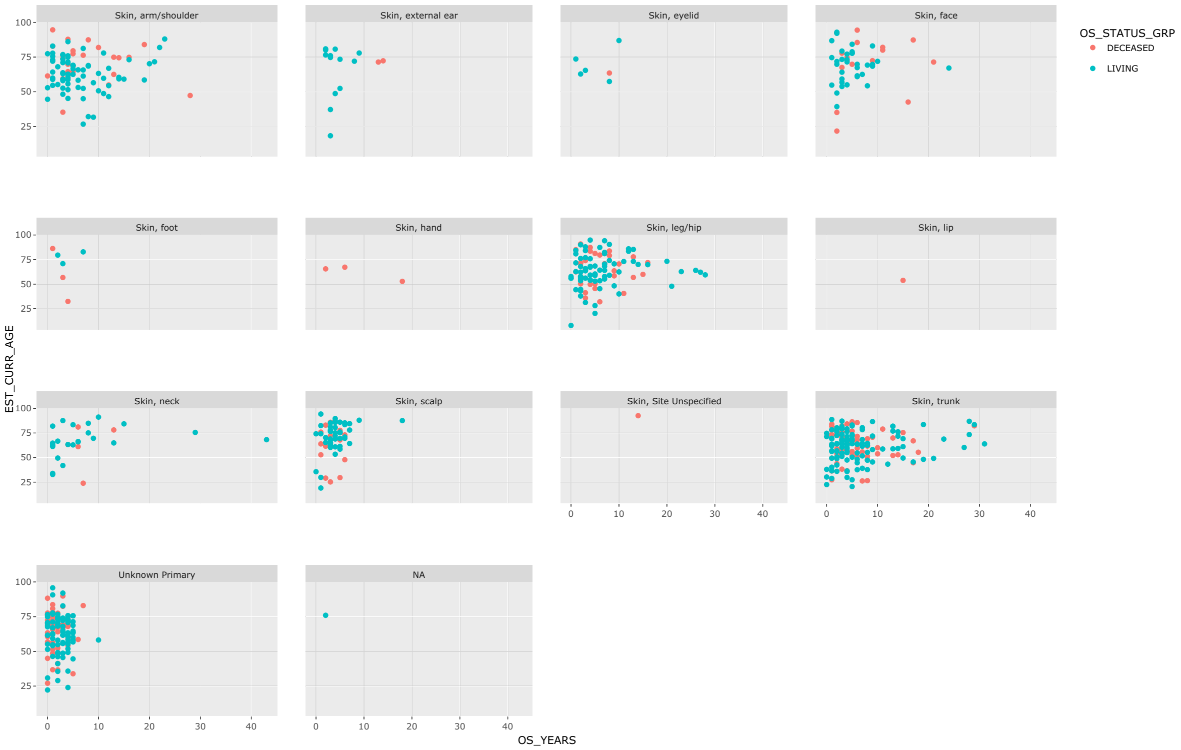
- I am interested in the split for `OS_STATUS_GRP` (Living or Deceased) and `EST_CURR_AGE` (continuous variable that represents the estimated current age of patient - age at diagnosis + OS_MONTHS)
- There are a lot more patients living with skin cancer diagnosis than there are deceased in this data set.
- However, there is a relatively high count of deceased with estimated current age between 40 and 80 years of age (count 174).

```
ggplotly(ggplot(df_clin, aes(x=EST_CURR_AGE, fill = OS_STATUS_GRP)) +
  geom_histogram(binwidth = 20))
```



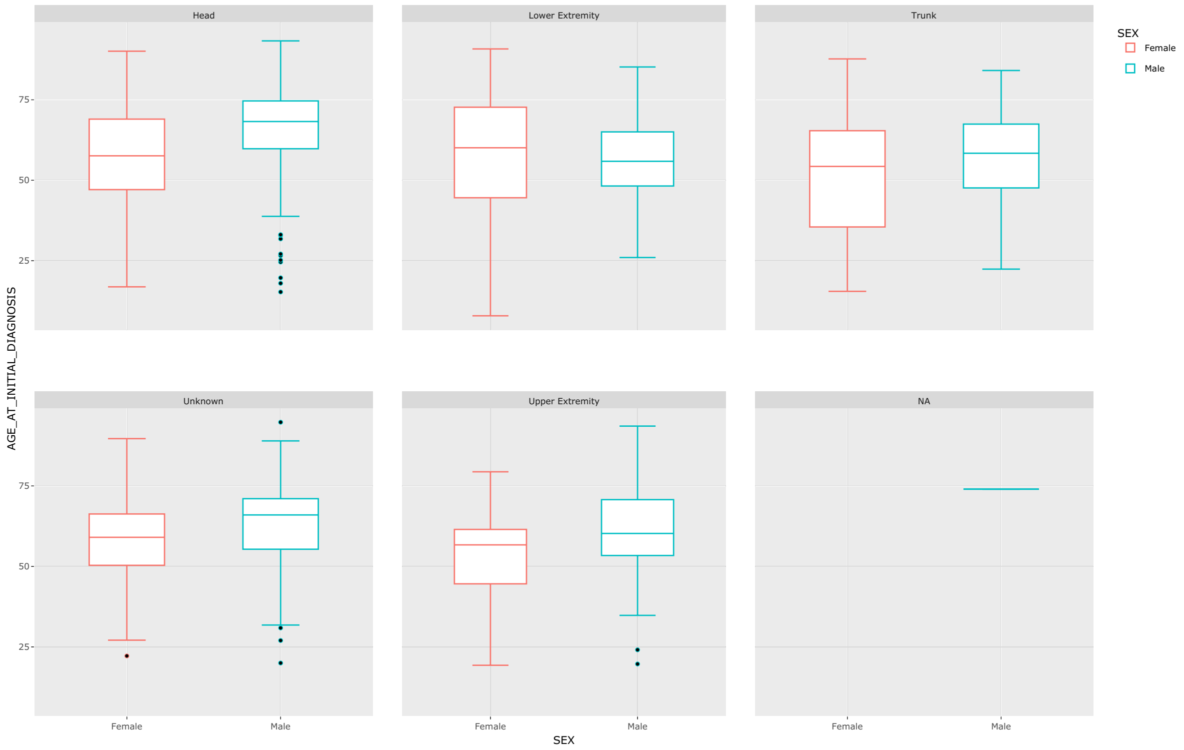
- I am interested in the split for `OS_STATUS_GRP` (Living or Deceased) and `OS_YEARS` (continuous variable that represents the number of years after initial diagnosis)
- This chart is replicated for each primary cancer site (`DMT_PRIMARY_SITE`) as a sub plot or graph.
- All patients diagnosed with skin cancer in hand/lip and site unspecified as primary site of melanoma are deceased.
- Patients diagnosed with skin cancer where foot was primary site has a 50:50 split for Living:Deceased (3:3).
- There are patients that had diagnosis in the leg/hip and trunk and primary site living long after the initial diagnosis from this data set (up to and just over 30 years).
- There is an outlier for one patient that is living after initial diagnosis of neck skin cancer 43 years after initial diagnosis and counting with estimated current age of 68yrs old.

```
ggplotly(ggplot(df_clin, aes(x=OS_YEARS, y = EST_CURR_AGE, color = OS_STATUS_GRP )) +
  geom_point() +
  facet_wrap(~DMT_PRIMARY_SITE))
```



- I would like to see if there are any outliers based on primary cancer site and gender so I will use a box plot this time.
- I am interested in the split between `SEX` (Male or Female) and `AGE_AT_INITIAL_DIAGNOSIS` (continuous variable that represents the patients age at initial diagnosis)
- This chart is replicated for each generic primary cancer site (`DMT_PRIMARY_SITE_CLASSIFIED`) as a sub plot or graph.
- `Head` and `Upper Extremity` graphs show outliers for males that were diagnosed earlier than normal for Head/Upper Extremity skin cancer.
- This is odd because males on average are diagnosed later with skin cancer except for `Lower Extremity`. This may be a sign that the data is not great. In addition, There are lot more male samples than female. Conversely, it is possible that the samples for female patients may be lacking.

```r
ggplotly(ggplot(df_clin, aes(x=SEX, y = AGE_AT_INITIAL_DIAGNOSIS, color = SEX)) +
  geom_boxplot() + facet_wrap(~DMT_PRIMARY_SITE_CLASSIFIED))
```



Part 2: R Packages

- I have selected the `testthat` and `covr` packages for this section.

testthat: Unit Testing for R

- The testthat package is for writing unit tests against R code (packages / functions).
- 'testthat' is a testing framework that is easy to learn and use, and integrates with RStudio.

covr: Test Coverage for Packages

- Track and report code coverage for your package and there is an option to upload the results to a coverage service. The covr package is used to generate reports that highlight the level of unit test coverage by percenatge and number of lines, logic branches (if else etc.) covered. It helps to identify missed lines of code or code that does not have any unit test coverage. It provides information on the code and its structure.

- Clean code has unit tests and unit tests also encourages code improvements and refactoring with piece of mind that the code is still functioning to a good level especially if unit test coverage is at a high percentage such as 85%+.

Please see test__final__proj__utils.R (uni tests / test suite) and final__proj__utils.R (source code) files attached.

See `test_final_proj_utils.R` for a test suite containing 5 tests that roll out to 12 test cases. The structure of the tests is the de-facto "Given, When, Then" formula for test driven development (TDD). `- Given` a set of test variables/constants/parameters for a given test scenario `- When` we call a method or unit of code with these configurations / arguments `- Then` we assert or check the result is what we expect. This is the most important step as we need to test for something tangible. Running a test and not verifying the result would be pointless.

- I have demonstrated some of the methods available from the `testthat` package for asserting the result of the unit test is as expected with functions: `expect_null`, `expect_false`, `expect_equal`, and `expect_failure`. There are other notable functions where you can expect s3 or s4 objects amongst others. This is very easy to use and from previous experience in unit testing with other languages, `testthat` package in R matches up quite well with JUnit / PyTest etc.
- I have used the `covr::file_coverage` method to run the unit tests with coverage to allow for code coverage report generation but you also run the tests as a stand alone directly from the test script in RStudio as long as `testthat` library is loaded.See screenshots for further information on this.

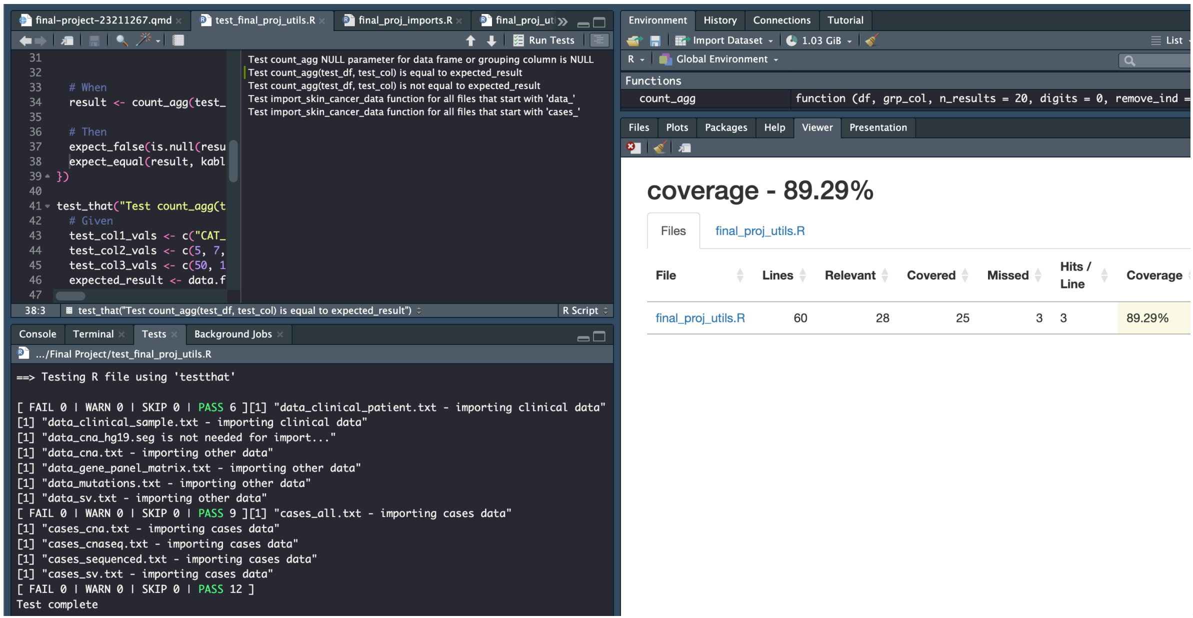
```r
# Run tests with coverage
covr <- file_coverage("final_proj_utils.R", "test_final_proj_utils.R")
```

```
[conflicted] Removing existing preference.
[conflicted] Will prefer dplyr::filter over any other package.
[conflicted] Removing existing preference.
[conflicted] Will prefer dplyr::lag over any other package.
[conflicted] Removing existing preference.
[conflicted] Will prefer dplyr::filter over any other package.
[conflicted] Removing existing preference.
[conflicted] Will prefer dplyr::lag over any other package.
[conflicted] Removing existing preference.
[conflicted] Will prefer dplyr::filter over any other package.
[conflicted] Removing existing preference.
[conflicted] Will prefer dplyr::lag over any other package.
```

```
Test passed
Test passed
Test passed
[1] "data_clinical_patient.txt - importing clinical data"
[1] "data_clinical_sample.txt - importing clinical data"
[1] "data_cna_hg19.seg is not needed for import..."
[1] "data_cna.txt - importing other data"
[1] "data_gene_panel_matrix.txt - importing other data"
[1] "data_mutations.txt - importing other data"
[1] "data_sv.txt - importing other data"
Test passed
[1] "cases_all.txt - importing cases data"
[1] "cases_cna.txt - importing cases data"
[1] "cases_cnaseq.txt - importing cases data"
[1] "cases_sequenced.txt - importing cases data"
[1] "cases_sv.txt - importing cases data"
Test passed
```

```
# Generate HTML coverage report
report(covr)
```

- The report function runs the coverage for the source and test code files passed as arguments. This can also be done for a whole package via `package_coverage`. There are methods to convert the report to cobertura code coverage or sonarqube reports in XML format which often plug-in to build servers like Jenkins for code coverage and static code scanning analysis. More information on the functions available at https://cran.r-project.org/web/packages/covr/covr.pdf.
- The report generated is in HTML format and can be viewed directly in RStudio or via internet browser such as chrome.
- The test script with the `testthat` framework installed and loaded allows 'Run Tests' option on top RHS of the code editor. The bottom left pain shows the results of running test suite (pass/fail/issues).



```r
  # When
  result <- count_agg(test_

  # Then
  expect_false(is.null(resu
  expect_equal(result, kabl
})

test_that("Test count_agg(t
  # Given
  test_col1_vals <- c("CAT_
  test_col2_vals <- c(5, 7,
  test_col3_vals <- c(50, 1
  expected_result <- data.f
```

Test count_agg NULL parameter for data frame or grouping column is NULL
Test count_agg(test_df, test_col) is equal to expected_result
Test count_agg(test_df, test_col) is not equal to expected_result
Test import_skin_cancer_data function for all files that start with 'data_'
Test import_skin_cancer_data function for all files that start with 'cases_'

38:3 test_that("Test count_agg(test_df, test_col) is equal to expected_result")

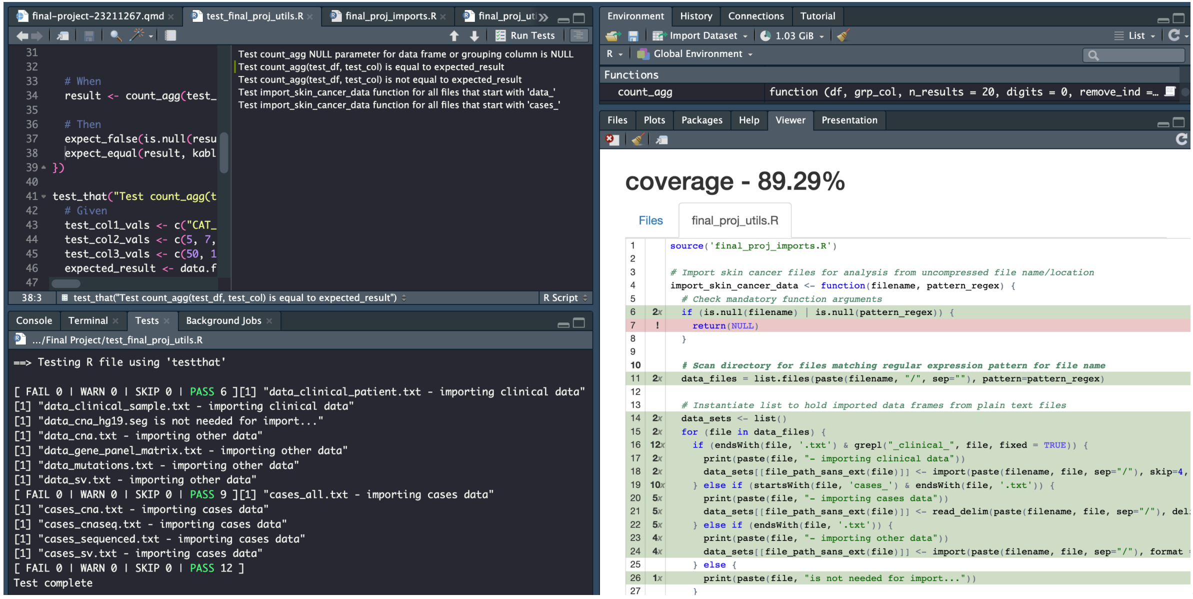
.../Final Project/test_final_proj_utils.R

```
==> Testing R file using 'testthat'

[ FAIL 0 | WARN 0 | SKIP 0 | PASS 6 ][1] "data_clinical_patient.txt - importing clinical data"
[1] "data_clinical_sample.txt - importing clinical data"
[1] "data_cna_hg19.seg is not needed for import..."
[1] "data_cna.txt - importing other data"
[1] "data_gene_panel_matrix.txt - importing other data"
[1] "data_mutations.txt - importing other data"
[1] "data_sv.txt - importing other data"
[ FAIL 0 | WARN 0 | SKIP 0 | PASS 9 ][1] "cases_all.txt - importing cases data"
[1] "cases_cna.txt - importing cases data"
[1] "cases_cnaseq.txt - importing cases data"
[1] "cases_sequenced.txt - importing cases data"
[1] "cases_sv.txt - importing cases data"
[ FAIL 0 | WARN 0 | SKIP 0 | PASS 12 ]
Test complete
```

Functions
count_agg function (df, grp_col, n_results = 20, digits = 0, remove_ind =

coverage - 89.29%

File	Lines	Relevant	Covered	Missed	Hits / Line	Coverage
final_proj_utils.R	60	28	25	3	3	89.29%



```r
  # When
  result <- count_agg(test_

  # Then
  expect_false(is.null(resu
  expect_equal(result, kabl
})

test_that("Test count_agg(t
  # Given
  test_col1_vals <- c("CAT_
  test_col2_vals <- c(5, 7,
  test_col3_vals <- c(50, 1
  expected_result <- data.f
```

Test count_agg NULL parameter for data frame or grouping column is NULL
Test count_agg(test_df, test_col) is equal to expected_result
Test count_agg(test_df, test_col) is not equal to expected_result
Test import_skin_cancer_data function for all files that start with 'data_'
Test import_skin_cancer_data function for all files that start with 'cases_'

38:3 test_that("Test count_agg(test_df, test_col) is equal to expected_result")

.../Final Project/test_final_proj_utils.R

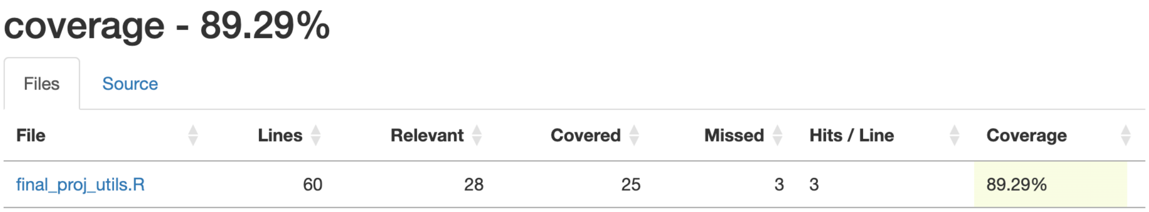
```
==> Testing R file using 'testthat'

[ FAIL 0 | WARN 0 | SKIP 0 | PASS 6 ][1] "data_clinical_patient.txt - importing clinical data"
[1] "data_clinical_sample.txt - importing clinical data"
[1] "data_cna_hg19.seg is not needed for import..."
[1] "data_cna.txt - importing other data"
[1] "data_gene_panel_matrix.txt - importing other data"
[1] "data_mutations.txt - importing other data"
[1] "data_sv.txt - importing other data"
[ FAIL 0 | WARN 0 | SKIP 0 | PASS 9 ][1] "cases_all.txt - importing cases data"
[1] "cases_cna.txt - importing cases data"
[1] "cases_cnaseq.txt - importing cases data"
[1] "cases_sequenced.txt - importing cases data"
[1] "cases_sv.txt - importing cases data"
[ FAIL 0 | WARN 0 | SKIP 0 | PASS 12 ]
Test complete
```

Functions
count_agg function (df, grp_col, n_results = 20, digits = 0, remove_ind =…

coverage - 89.29%

```
1       source('final_proj_imports.R')
2
3       # Import skin cancer files for analysis from uncompressed file name/location
4       import_skin_cancer_data <- function(filename, pattern_regex) {
5         # Check mandatory function arguments
6   2x    if (is.null(filename) | is.null(pattern_regex)) {
7   !       return(NULL)
8         }
9
10        # Scan directory for files matching regular expression pattern for file name
11  2x    data_files = list.files(paste(filename, "/", sep=""), pattern=pattern_regex)
12
13        # Instantiate list to hold imported data frames from plain text files
14  2x    data_sets <- list()
15  2x    for (file in data_files) {
16  12x     if (endsWith(file, '.txt') & grepl("_clinical_", file, fixed = TRUE)) {
17  2x        print(paste(file, "- importing clinical data"))
18  2x        data_sets[[file_path_sans_ext(file)]] <- import(paste(filename, file, sep="/"), skip=4,
19  10x     } else if (startsWith(file, 'cases_') & endsWith(file, '.txt')) {
20  5x        print(paste(file, "- importing cases data"))
21  5x        data_sets[[file_path_sans_ext(file)]] <- read_delim(paste(filename, file, sep="/"), del
22  5x      } else if (endsWith(file, '.txt')) {
23  4x        print(paste(file, "- importing other data"))
24  4x        data_sets[[file_path_sans_ext(file)]] <- import(paste(filename, file, sep="/"), format
25          } else {
26  1x        print(paste(file, "is not needed for import..."))
27          }
```

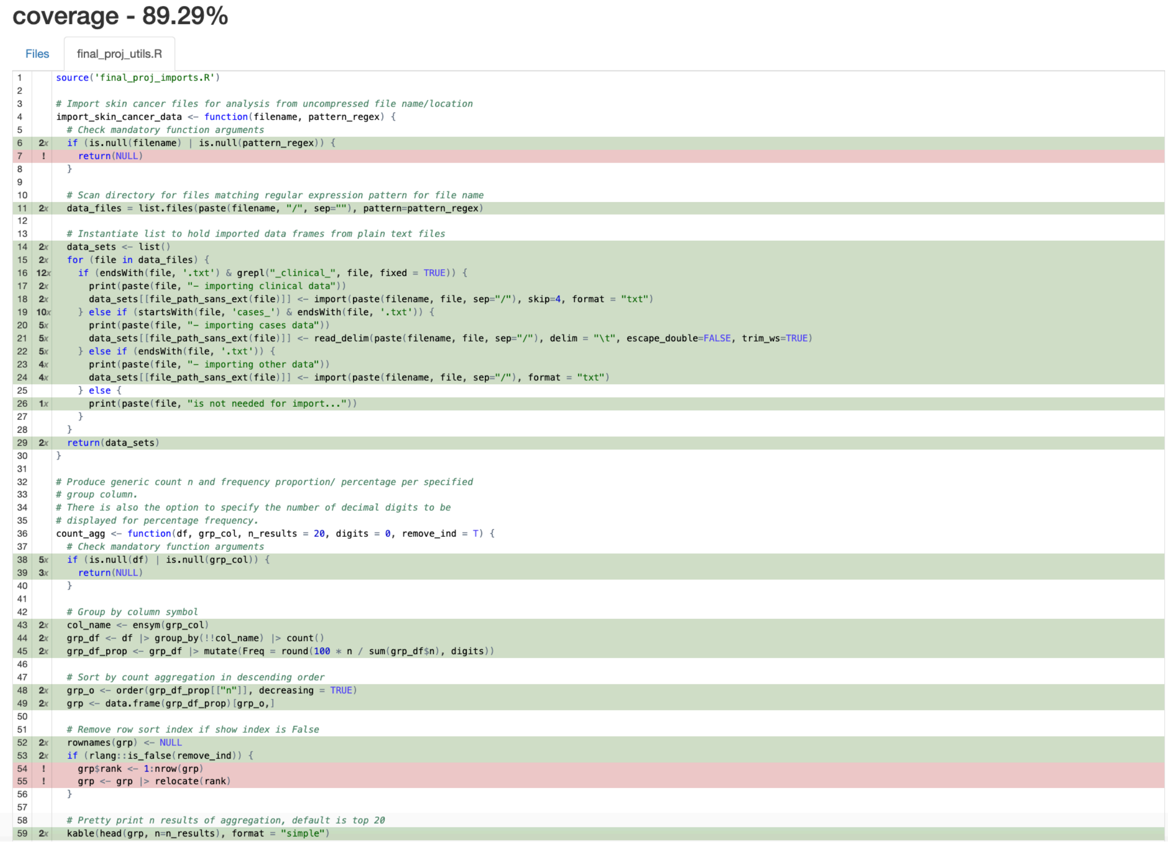
- The bottom right 'viewer' displays report generated by 'report' function.



coverage - 89.29%

Files | Source

File	Lines	Relevant	Covered	Missed	Hits / Line	Coverage
final_proj_utils.R	60	28	25	3	3	89.29%



coverage - 89.29%

Files | final_proj_utils.R

```
     source('final_proj_imports.R')

     # Import skin cancer files for analysis from uncompressed file name/location
     import_skin_cancer_data <- function(filename, pattern_regex) {
       # Check mandatory function arguments
2x     if (is.null(filename) | is.null(pattern_regex)) {
!        return(NULL)
       }

       # Scan directory for files matching regular expression pattern for file name
2x     data_files = list.files(paste(filename, "/", sep=""), pattern=pattern_regex)

       # Instantiate list to hold imported data frames from plain text files
2x     data_sets <- list()
2x     for (file in data_files) {
12x      if (endsWith(file, '.txt') & grepl("_clinical_", file, fixed = TRUE)) {
2x         print(paste(file, "- importing clinical data"))
2x         data_sets[[file_path_sans_ext(file)]] <- import(paste(filename, file, sep="/"), skip=4, format = "txt")
10x      } else if (startsWith(file, 'cases_') & endsWith(file, '.txt')) {
5x         print(paste(file, "- importing cases data"))
5x         data_sets[[file_path_sans_ext(file)]] <- read_delim(paste(filename, file, sep="/"), delim = "\t", escape_double=FALSE, trim_ws=TRUE)
5x       } else if (endsWith(file, '.txt')) {
4x         print(paste(file, "- importing other data"))
4x         data_sets[[file_path_sans_ext(file)]] <- import(paste(filename, file, sep="/"), format = "txt")
         } else {
1x         print(paste(file, "is not needed for import..."))
         }
       }
2x     return(data_sets)
     }

     # Produce generic count n and frequency proportion/ percentage per specified
     # group column.
     # There is also the option to specify the number of decimal digits to be
     # displayed for percentage frequency.
     count_agg <- function(df, grp_col, n_results = 20, digits = 0, remove_ind = T) {
       # Check mandatory function arguments
5x     if (is.null(df) | is.null(grp_col)) {
3x       return(NULL)
       }

       # Group by column symbol
2x     col_name <- ensym(grp_col)
2x     grp_df <- df |> group_by(!!col_name) |> count()
2x     grp_df_prop <- grp_df |> mutate(Freq = round(100 * n / sum(grp_df$n), digits))

       # Sort by count aggregation in descending order
2x     grp_o <- order(grp_df_prop[["n"]], decreasing = TRUE)
2x     grp <- data.frame(grp_df_prop)[grp_o,]

       # Remove row sort index if show index is False
2x     rownames(grp) <- NULL
2x     if (rlang::is_false(remove_ind)) {
!        grp$rank <- 1:nrow(grp)
!        grp <- grp |> relocate(rank)
       }

       # Pretty print n results of aggregation, default is top 20
2x     kable(head(grp, n=n_results), format = "simple")
```

Part 3: Functions / Programming

S3 Class / Objects / Methods

- Create `AggregationArgs` class from object instance `aggregation_drv_class`
- The code creates a list called `aggregation_drv_class` with 4 elements: df, col, n_results and digits.
- The values of these elements are `df_clin`, "DRIVER_CLASS", 3, and 1 respectively.
- Then, the class() function is used to assign a class to the aggregation_drv_class list.

- The class is set to “AggregationArgs”.

```r
aggregation_drv_class <- list(df = df_clin, col = "DRIVER_CLASS", n_results=3, digits = 1)
class(aggregation_drv_class) <- "AggregationArgs"
```

- Define `countAgg` function / method.
- The function takes one argument, “object”.
- The function then uses the “UseMethod” function to dispatch the method that matches the class of the “object” argument.
- This is an example of OOP in R, where different methods can be defined for different classes of objects.
- This code defines a function called `countAgg` that takes an object as its argument.
- Within the function, it uses the `count_agg` function that I previously defined in `final_proj_itls.R` to print a summary table with count and Frequency info for the data frame and `group_by` column passed as arguments, it also takes 2 parameters (`n_results` and `digits`) that are normally optional (more strict version).

```r
countAgg <- function(object) {
  UseMethod("countAgg")
}

countAgg.AggregationArgs <- function(object) {
  cat("Entering countAgg.AggregationArgs method with params", object$col, object$n_results
  count_agg(object$df, object$col, n_results = object$n_results, digits = object$digits)
}
```

- Initially countAgg prints the object components except for the data frame as it is a large object (too much information for debugging).
- Then it prints to console the summary table as expected in `knitr::kable format`

```r
countAgg(aggregation_drv_class)
```

```
Entering countAgg.AggregationArgs method with params DRIVER_CLASS 3 1
```

DRIVER_CLASS	n	Freq
BRAF_CLASS_1_V600E	172	24.7
NRAS_Q61	172	24.7
NF1	155	22.3

- This time I call the same S3 function with different arguments which is straight forward as the OOP part is already defined now and reusable.

```
aggregation_tmb <- list(df = df_clin, col = "TMB_NONSYNONYMOUS", n_results=5, digits = 2)
class(aggregation_tmb) <- "AggregationArgs"
countAgg(aggregation_tmb)
```

```
Entering countAgg.AggregationArgs method with params TMB_NONSYNONYMOUS 5 2
```

TMB_NONSYNONYMOUS	n	Freq
4.323491	18	2.59
5.188189	17	2.44
2.594094	15	2.16
6.052887	15	2.16
12.970471	14	2.01

Finally, I want to verify that the method is S3.

```
isS3method("countAgg.AggregationArgs")
```

```
[1] TRUE
```

```
isS3stdGeneric(countAgg)
```

```
countAgg
    TRUE
```

S4 Class / Objects / Methods

- The setup for S4 is a bit different.
- First I want to create the equivalent S4 class with defined slots (object or list components by type instead of value)

```
setClass("countAggregation", slots=list(df='data.frame', col="character", n_results="numer
```

- Then I create an instance of my class object
- I want to verify the object is a valid object and that it is a valid instance of an S4 class

```r
obj1 <- new("countAggregation",df=df_clin, col="SIMPLIFIED_METASTATIC_SITE", n_results=6,
is.object(obj1)
```

```
[1] TRUE
```

```r
isS4(obj1)
```

```
[1] TRUE
```

- Instead of the `$` / dollar sign symbol in S3, I can use the `@` symbol to access components of an S4 object

```r
obj1@col
```

```
[1] "SIMPLIFIED_METASTATIC_SITE"
```

```r
obj1@n_results
```

```
[1] 6
```

```r
obj1@digits
```

```
[1] 4
```

- Now, I would like to set the method to be called to show the output of this aggregation.

```r
setMethod("show", "countAggregation", function(object) {
  cat("Entering show.countAggregation method with params", object@col, object@n_results, o
  print(count_agg(object@df, object@col, n_results = object@n_results, digits = object@dig
})
```

Now I would like to test the method is called when I show/print the object to console

```r
obj1
```

```
Entering show.countAggregation method with params SIMPLIFIED_METASTATIC_SITE 6 4

SIMPLIFIED_METASTATIC_SITE          n      Freq
-----------------------------  ----  --------
LR / In-Transit / Regional LN    271   38.9368
NA                               104   14.9425
Soft Tissue / Distant LN          96   13.7931
Lung                              88   12.6437
Brain                             52    7.4713
Liver                             36    5.1724
```

- Instead of creating another object, I am just going to modify the existing S4 object to generate another summary table for `DRIVER_CLASS`

```
obj1@col <- "DRIVER_CLASS"
```

Finally, I would like to test the method is called when I show/print the *UPDATED* object to console

```
obj1
```

```
Entering show.countAggregation method with params DRIVER_CLASS 6 4

DRIVER_CLASS            n      Freq
-------------------  ----  --------
BRAF_CLASS_1_V600E    172   24.7126
NRAS_Q61              172   24.7126
NF1                   155   22.2701
BRAF_CLASS_2_3         44    6.3218
BRAF_CLASS_1_Other     43    6.1782
Other_Driver           43    6.1782
```

- This is an example of how useful OOP programming is for re-usable code whether using S3/S4. S4 is newer and cleaner so I think I prefer S4 (less code after setup).