Final Check-In

2025-08-10

```
library(ggplot2)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ lubridate 1.9.4     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ tidyr     1.3.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

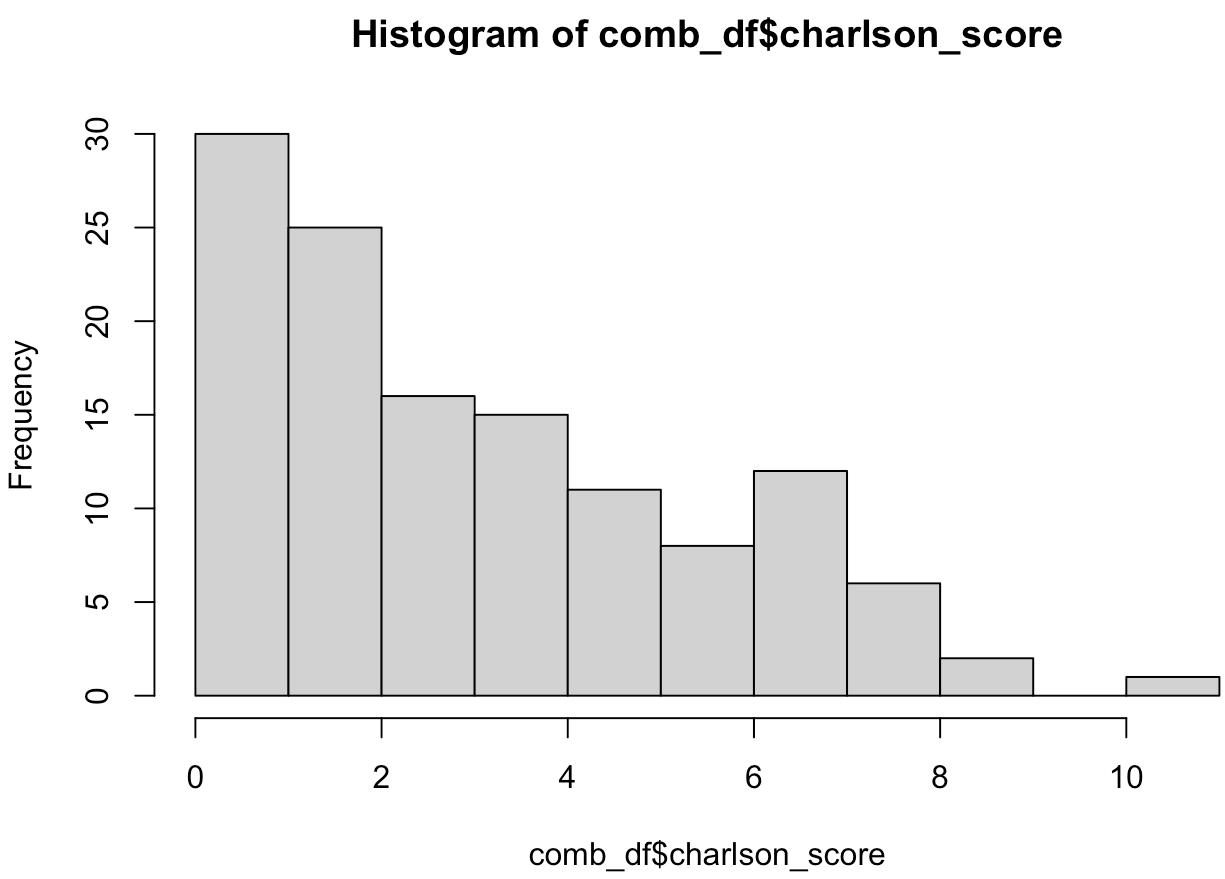
```
library(glue)
library(knitr)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

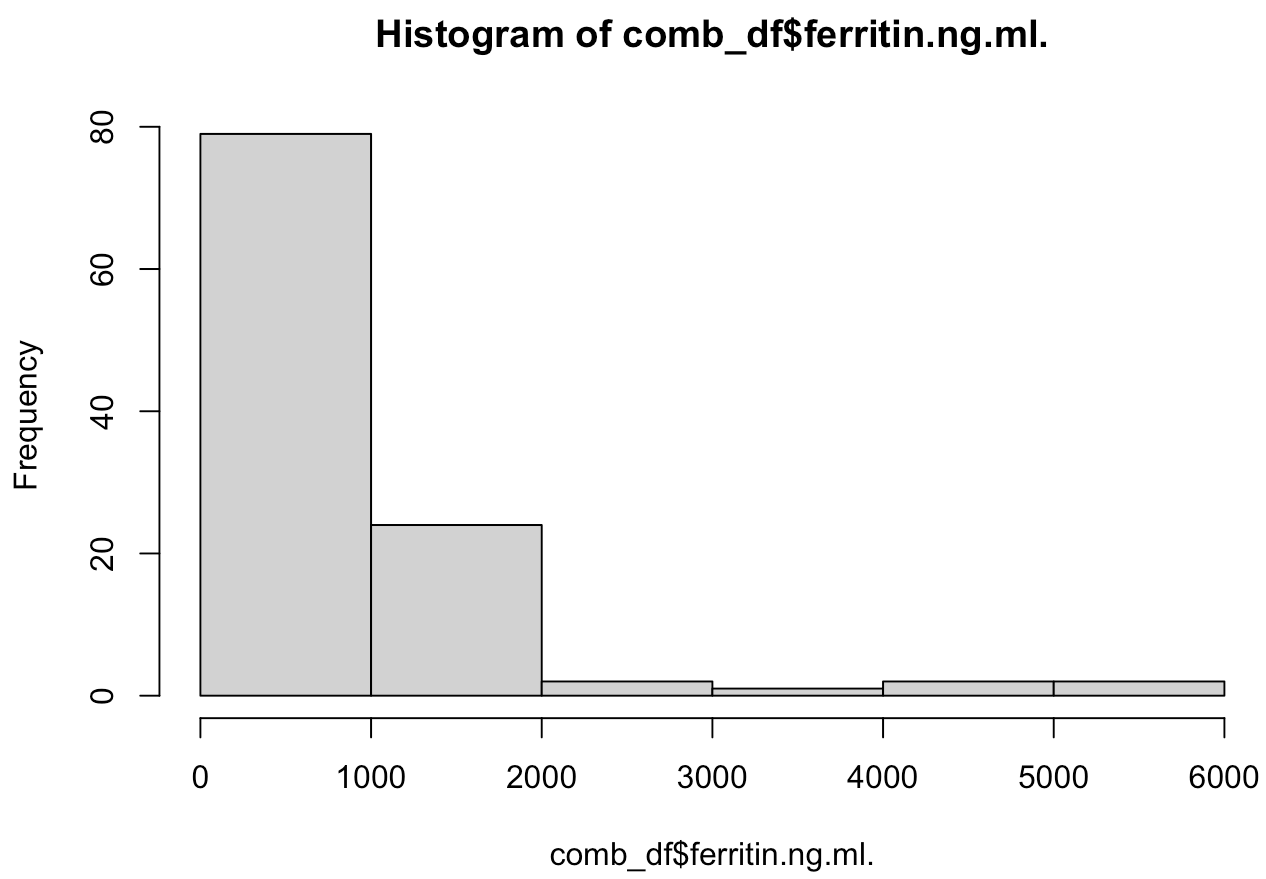
```
filename <- "/Users/nupoormarwah/Downloads/QBS103_GSE157103_series_matrix-1.csv"
meta_data <- fread(filename)

filename <- "/Users/nupoormarwah/Downloads/QBS103_GSE157103_genes.csv"

#Makes so that V1 col that contains the gene names are now the row names
gene_expr <- fread(filename) %>%
  column_to_rownames("V1") %>%
 #Transposing with make the table be 126 observations by 100 observations
  t() %>%
  data.frame()

#Create combined data frame. Now we have one single data frame with all the data. I did
n't initially know how to do this, so I googled "how to combine objects into rows and co
lumns in R" and used the following website for reference: https://www.rdocumentation.or
g/packages/base/versions/3.6.2/topics/cbind
comb_df <- cbind(meta_data, gene_expr)

comb_df <- comb_df %>%
  mutate(
    #age = as.numeric(age),
    `ferritin(ng/ml)` = as.numeric(`ferritin(ng/ml)`),
  ) %>%
  data.frame()
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `ferritin(ng/ml) = as.numeric(`ferritin(ng/ml)`)`.
## Caused by warning:
## ! NAs introduced by coercion
```

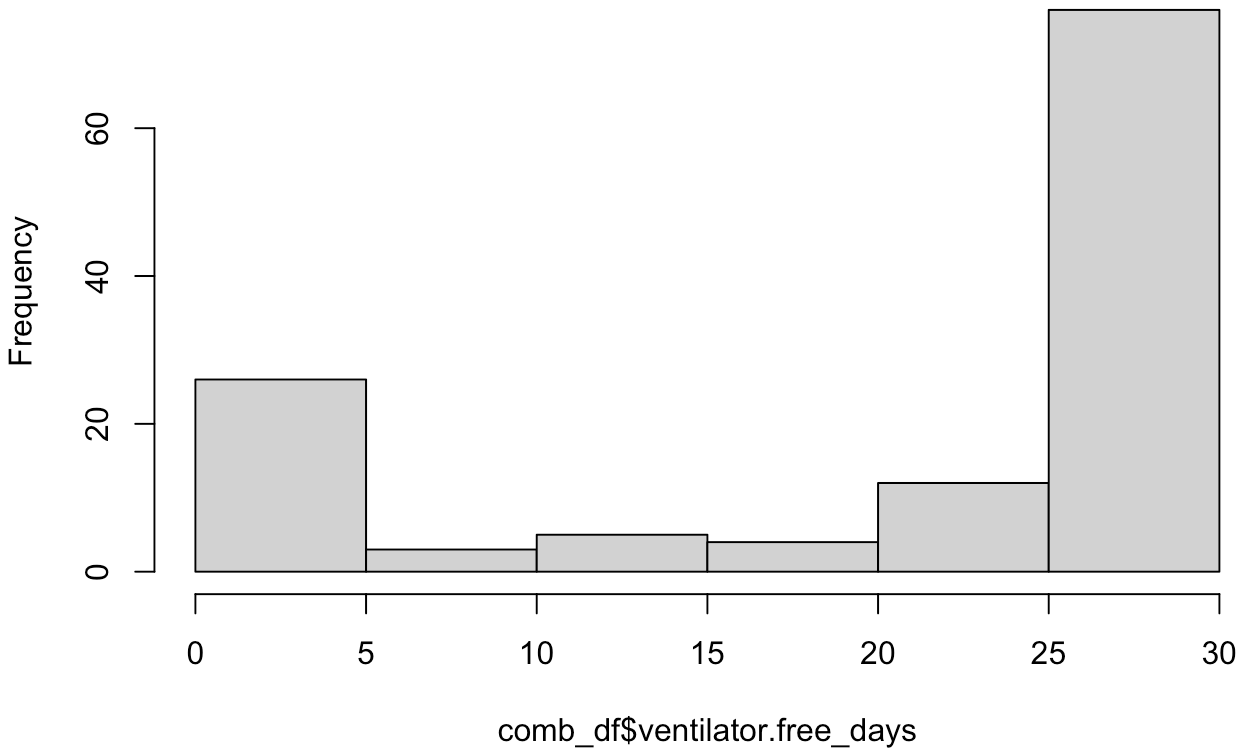
```
#Checking for normality via histograms. They are all skewed so we want median and IQR fo
r summary stats. Note: These were only for my reference, so that I knew which metrics to
report.
hist(comb_df$charlson_score)
```




```
hist(comb_df$`ferritin.ng.ml.`)
```




```
hist(comb_df$ventilator.free_days)
```




```
num_vector <- c("charlson_score", "ferritin.ng.ml.", "ventilator.free_days")
cat_vector <- c("disease_status", "icu_status")

#Categorical summary stats
cat_sum <- comb_df %>%
  select(sex, all_of(cat_vector)) %>%
  pivot_longer(-sex, names_to = "variable", values_to = "category") %>%
  group_by(sex, variable) %>%
  mutate(group_total = n()) %>%
  group_by(sex, variable, category, group_total) %>%
  summarize(n = n(), ) %>%

  mutate(
    perc = round(100 * (n/group_total), 2),
    format_n = glue("{n} ({perc})")
    ) %>%
  #Exclude those columns
  select(-n, -perc) %>%
  pivot_wider(
    names_from = c(variable, category),
    values_from = c(format_n),
    #Allows for string interpolation
    names_glue = "{.value}_{variable}_{category}"
  )
```

```
## `summarise()` has grouped output by 'sex', 'variable', 'category'. You can
## override using the `.groups` argument.
```

```
colnames(cat_sum) <- c("Sex", "Count", "COVID-19: n (%)", "Non-COVID-19: n (%)", "ICU: n
(%)", "Non-ICU: n (%)")

#Continuous summary stats
num_sum <- comb_df %>%
  select(sex, all_of(num_vector)) %>%
  group_by(sex) %>%
  summarize(
    across(
      all_of(num_vector),
      #.x is placeholder for columns. For example, median and IQR for Charlson Score and
then assign it to the next variable, etc.
      list(median = ~ median(.x, na.rm = TRUE), IQR = ~ IQR(.x, na.rm = TRUE)),
      #"fn" stands for function (median or IQR). It will give the name of that function
to that column. ".col" is the column name you are calculating the value for.
      .names = "{.fn} {.col}"
    )
  )
colnames(num_sum) <- c("Sex", "Charlson Score: Median", "Charlson Score: IQR",
                       "Ferritin (ng/mL): Median", "Ferritin (ng/mL): IQR",
                       "Ventilator-Free Days: Median", "Ventilator-Free Days: IQR")

#Final table (both tables merged)
final <- cat_sum %>%
  left_join(num_sum, by = "Sex") %>%
  mutate(
    Sex = factor(
      Sex,
      levels = c("female", "male", "unknown"),
      labels = c("Female", "Male", "Unknown")
    )
  )
```

```
#Final table formatting
kable(final, caption = "Table 1: Multiple Covariates Stratified by Sex") %>%
  kable_classic(full_width = TRUE, position = "center") %>%
  column_spec(1:ncol(final), width = "20em")
```

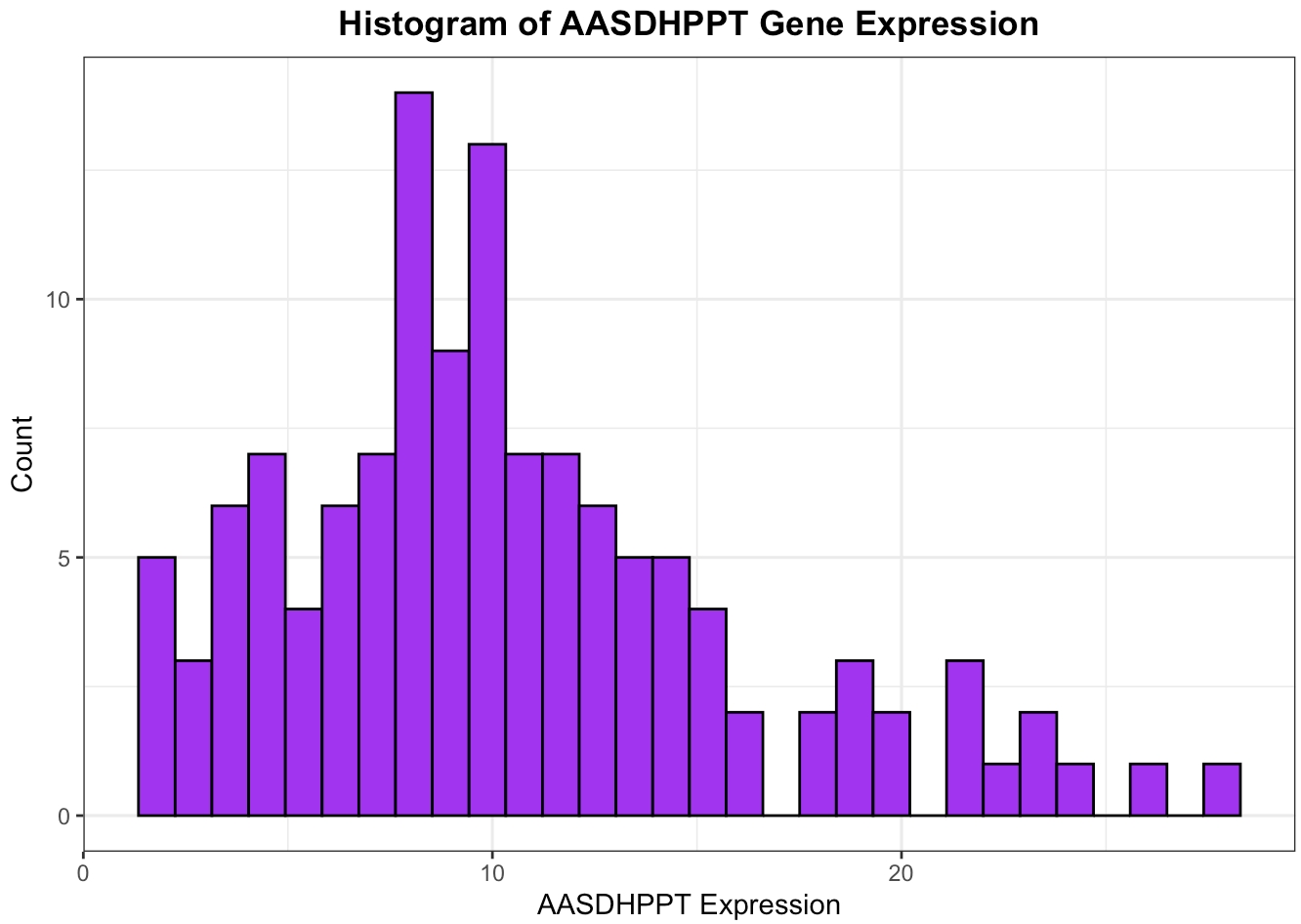
Table 1: Multiple Covariates Stratified by Sex

Sex	Count	COVID-19: n (%)	Non-COVID-19: n (%)	ICU: n (%)	Non-ICU: n (%)	Charlson Score: Median	Charlson Score: IQR	Ferritin (ng/mL): Median	Ferritin (ng/mL): IQR	Ventilator-Free Days: Median	Ventilator-Free Days: IQR
Female	51	38 (74.51)	13 (25.49)	27 (52.94)	24 (47.06)	3	4	318	547	28	10.0
Male	74	62 (83.78)	12 (16.22)	33 (44.59)	41 (55.41)	3	4	755	849	28	18.5

Sex	Count	COVID-19: n (%)	Non-COVID-19: n (%)	ICU: n (%)	Non-ICU: n (%)	Charlson Score: Median	Charlson Score: IQR	Ferritin (ng/mL): Median	Ferritin (ng/mL): IQR	Ventilator-Free Days: Median	Ventilator-Free Days: IQR
Unknown	1	NA	1 (100)	NA	1 (100)	8	0	NA	NA	28	0.0

```
#First layer in ggplot function is data. Next is mapping to tell R where to put the cols and rows on the plot. "aes" is part of the mapping layer and tells R what to make the x and y-axis. Since this is a histogram, we only need to assign x-axis
p1 <- ggplot(data = comb_df, mapping = aes(x = AASDHPPT)) +
  #After mapping, the next step is to add a geom layer to tell R what type of plot you want to make. You can use "fill" to change the color of the entire shape or "color" to change the outline
  geom_histogram(fill = "purple", col = "black") +
  #"labs" gives labels to x and y-axis and you can include a title
  labs(x = "AASDHPPT Expression", y = "Count", title = "Histogram of AASDHPPT Gene Expression") +
  #We can choose any theme. BW gives black axis and text and white background
  theme_bw() +
  #Use theme again to center title and bold it. "Element text" function says that the element in the plot title is a text element. "hjust" is horizontal adjustment (0 = all the way to the left, 1 = all the way to the right, so center = 0.50)
  theme(
     plot.title = element_text(face = "bold", hjust = 0.50)
  )

p1
```

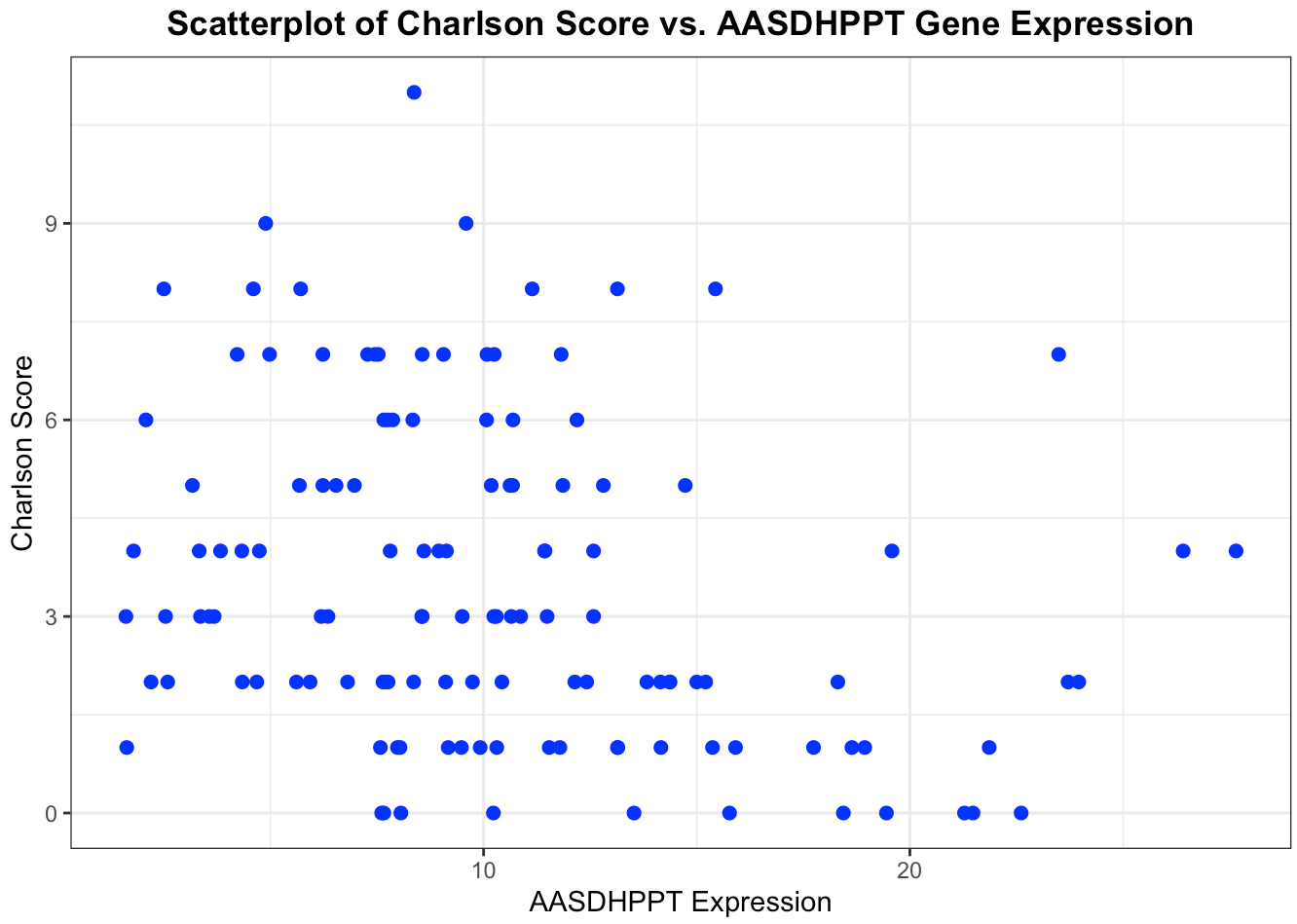
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggsave("Gene Histogram.png", plot = p1)
```

```
## Saving 7 x 5 in image
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

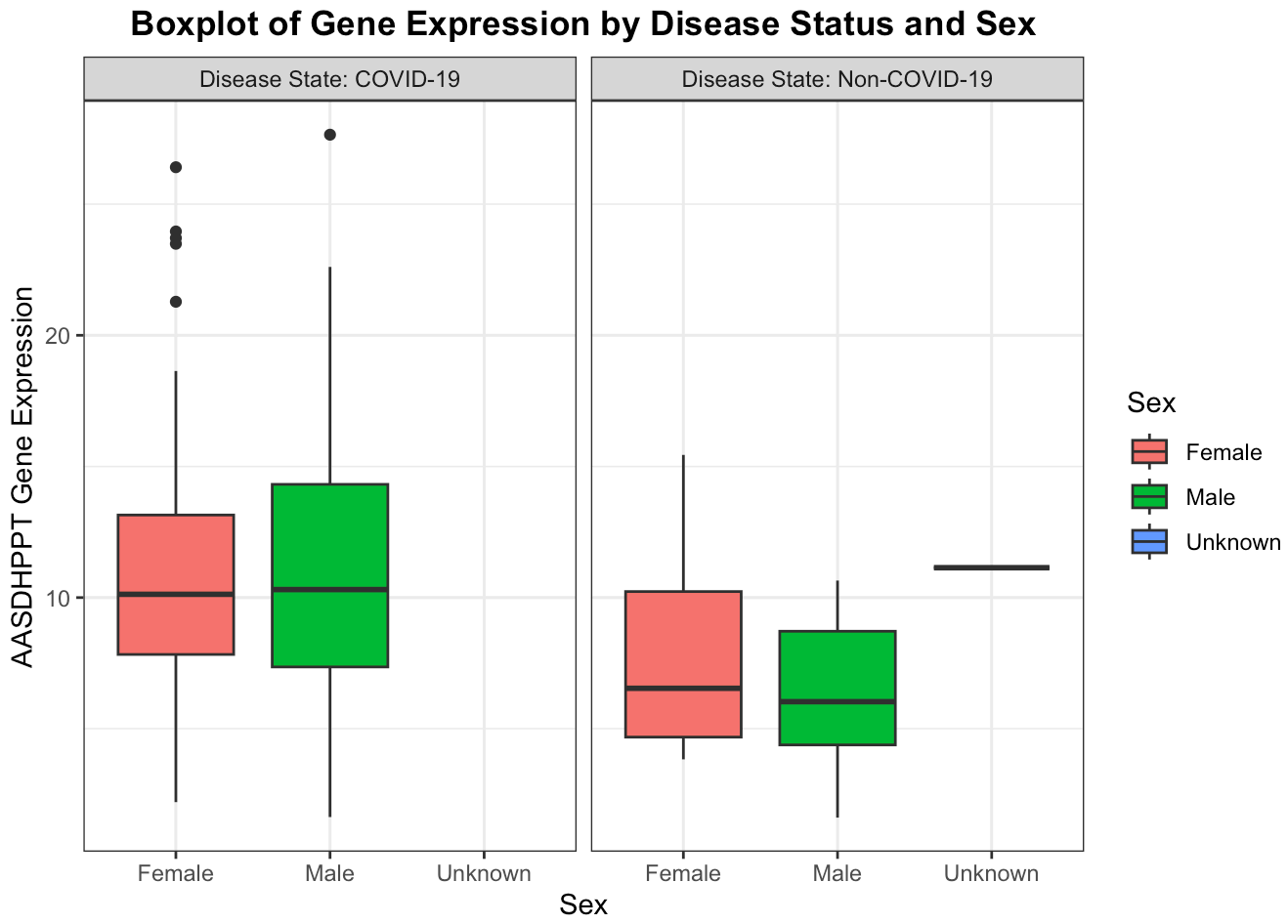
```
#First layer in ggplot function is data. Next is mapping to tell R where to put the cols and rows on the plot. "aes" is part of the mapping layer and tells R what to make the x- and y-axis. Since this is a scatterplot, we need both x- and y-axis. I chose Charlson Score as the continuous variable here
p2 <- ggplot(data = comb_df, mapping = aes(x = AASDHPPT, y = charlson_score)) +
  #After mapping, the next step is to add a geom layer to tell R what type of plot you want to make. You can use "color" to change the color of the dots in the scatterplot
  geom_point(color = "blue", size = 2) +
  #"labs" gives labels to x and y-axis and you can include a title
  labs(x = "AASDHPPT Expression", y = "Charlson Score", title = "Scatterplot of Charlson Score vs. AASDHPPT Gene Expression") +
  #We can choose any theme. BW gives black axis and text and white background
  theme_bw() +
  #Use theme again to center title and bold it. "Element text" function says that the element in the plot title is a text element. "hjust" is horizontal adjustment (0 = all the way to the left, 1 = all the way to the right, so center = 0.50)
  theme(
     plot.title = element_text(face = "bold", hjust = 0.50)
  )

p2
```




```
ggsave("Gene Histogram.png", plot = p2)
```

```
## Saving 7 x 5 in image
```

```
comb_df %>%
  mutate(
    sex = factor(
      sex,
      levels = c("female", "male", "unknown"),
      labels = c("Female", "Male", "Unknown")
    ),
    disease_status = factor(
      disease_status,
      levels = c("disease state: COVID-19", "disease state: non-COVID-19"),
      labels = c("Disease State: COVID-19", "Disease State: Non-COVID-19")
    )
  ) %>%
#First layer in ggplot function is data. Next is mapping to tell R where to put the cols and rows on the plot. "aes" is part of the mapping layer and tells R what to make the x and y-axis. Since this is a boxplot, we need both x- and y-axis. I chose sex and disease status as my categorical variables
ggplot(mapping = aes(x = sex, y = AASDHPPT, fill = sex)) +
  #After mapping, the next step is to add a geom layer to tell R what type of plot you want to make
  geom_boxplot() +
  #Add a facet. A facet lets us separate based on the values of a variable. Here, it would be the "disease_status" var
  facet_wrap(~disease_status) +
  #"labs" gives labels to x and y-axis and you can include a title. To change the legend, use "fill" since we used that up top
  labs(x = "Sex", y = "AASDHPPT Gene Expression", title = "Boxplot of Gene Expression by Disease Status and Sex",
       fill = "Sex") +
  #We can choose any theme. BW gives black axis and text and white background
  theme_bw() +
  #Use theme again to center title and bold it. "Element text" function says that the element in the plot title is a text element. "hjust" is horizontal adjustment (0 = all the way to the left, 1 = all the way to the right, so center = 0.50)
  theme(
     plot.title = element_text(face = "bold", hjust = 0.50)
  )
```

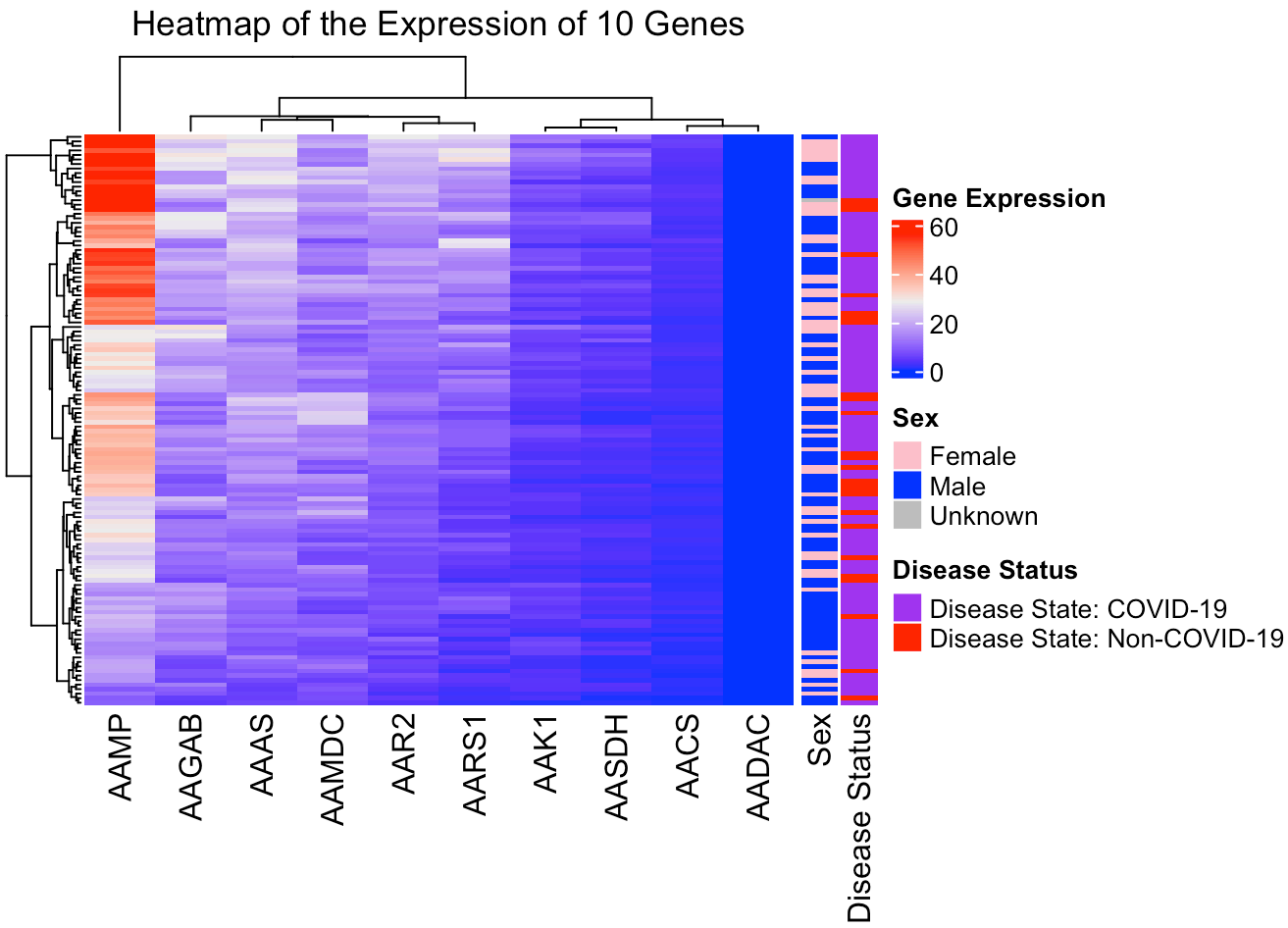



```
library(ComplexHeatmap)
```

```
## Loading required package: grid
```

```
## ========================================
## ComplexHeatmap version 2.22.0
## Bioconductor page: http://bioconductor.org/packages/ComplexHeatmap/
## Github page: https://github.com/jokergoo/ComplexHeatmap
## Documentation: http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite either one:
## - Gu, Z. Complex Heatmap Visualization. iMeta 2022.
## - Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
##     genomic data. Bioinformatics 2016.
##
##
## The new InteractiveComplexHeatmap package can directly export static
## complex heatmaps into an interactive Shiny app with zero effort. Have a try!
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(ComplexHeatmap))
## ========================================
```

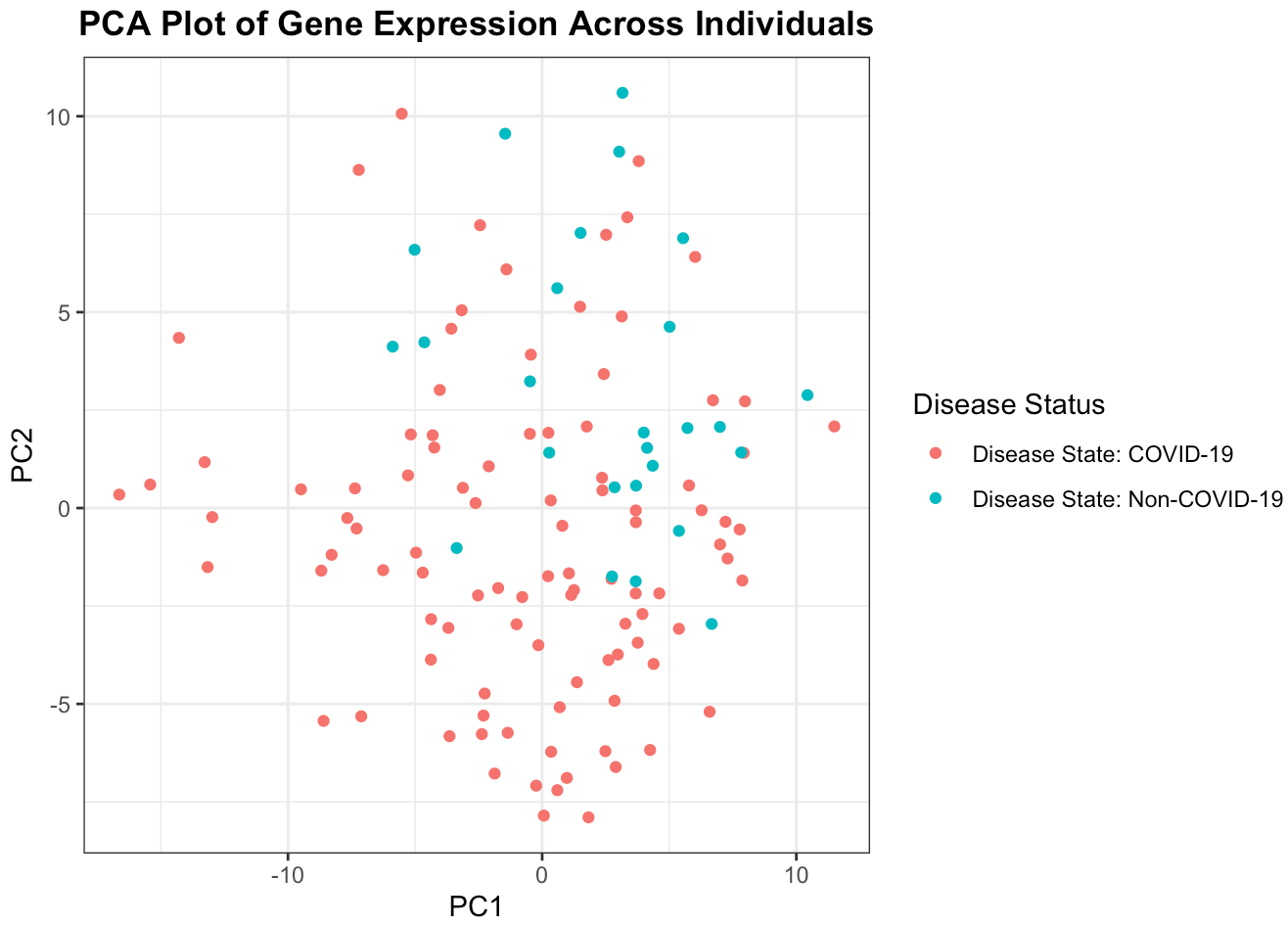
```
genes <- c("AAGAB", "AAK1", "AAMDC", "AAMP", "AAR2", "AARS1", "AASDH", "AAAS", "AACS",
"AADAC")
expression_mat <- as.matrix(comb_df[,genes])


annotations <- data.frame(
  Sex = factor(
    comb_df$sex,
    levels = c("female", "male", "unknown"),
    labels = c("Female", "Male", "Unknown")
    ),
  Disease_Status = factor(
    comb_df$disease_status,
    levels = c("disease state: COVID-19", "disease state: non-COVID-19"),
    labels = c("Disease State: COVID-19", "Disease State: Non-COVID-19")
    )

)

rownames(annotations) <- comb_df$participant_id

color_sex <- setNames(c("pink", "blue", "gray"), levels(annotations$Sex))
color_ds <- setNames(c("purple", "red"), levels(annotations$Disease_Status))
hm_annot <- rowAnnotation(
  Sex = annotations$Sex,
  `Disease Status` = annotations$Disease_Status,
  col = list(Sex = color_sex, `Disease Status` = color_ds)
)

Heatmap(
  expression_mat,
  name = "Gene Expression",
  column_title = "Heatmap of the Expression of 10 Genes",
  right_annotation = hm_annot,
  cluster_rows = TRUE,
  cluster_columns = TRUE
)
```



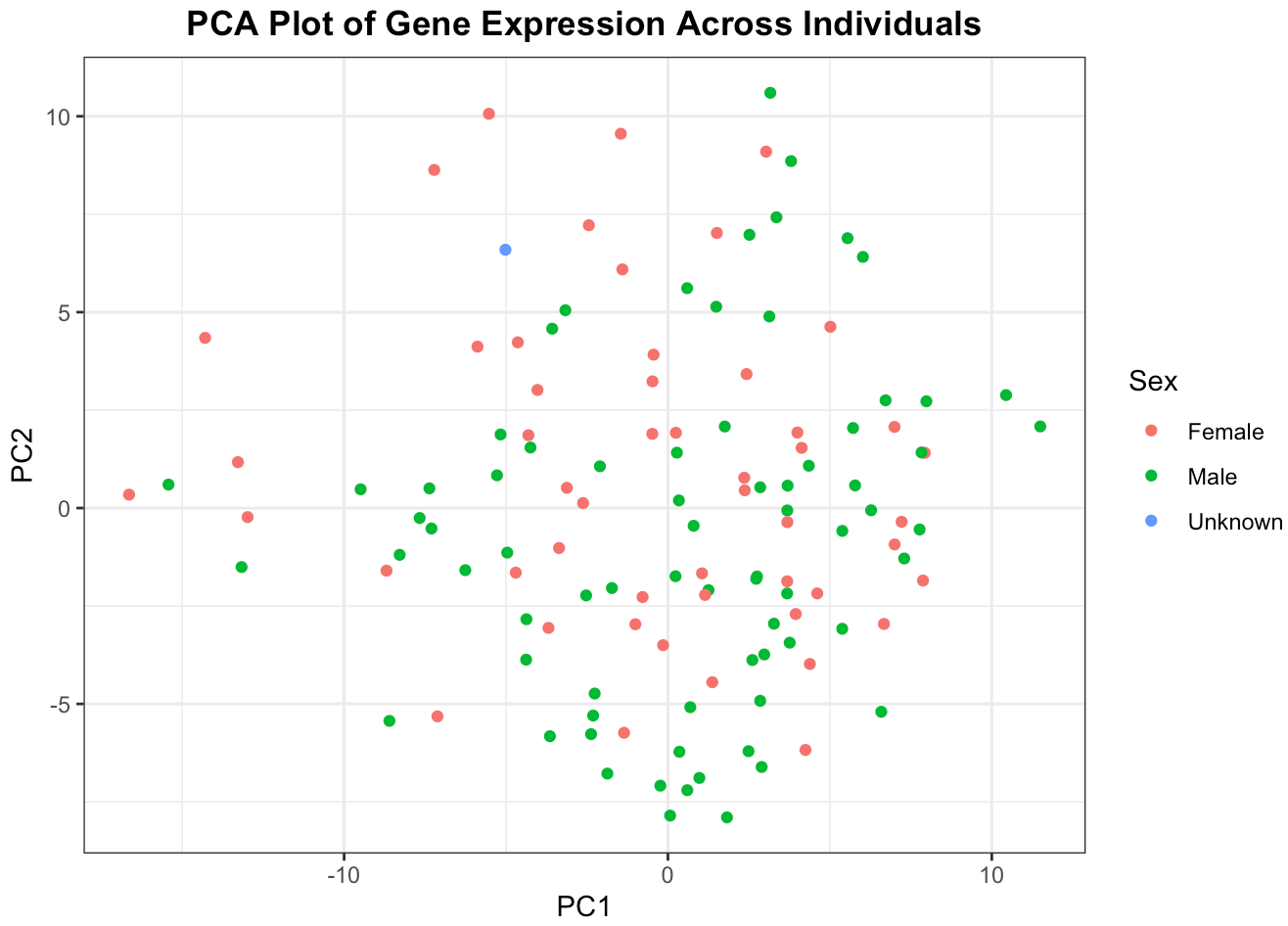
#Principal Component Analysis Plot (PCA Plot)

```
pc_genes <- colnames(gene_expr)
#Apply does variance function on every column of gene expression data frame. After calcu
lating variance, will check if it is equal to 0. If not 0, it will return True. If 0, it
will return False. Indexing so that we only get columns where variance isn't equal to 0
filt_gene_expr <- gene_expr[, apply(gene_expr, 2, var) != 0]
#Based off variation in your data. Columns will often have different levels of variatio
n. We want to scale our coumns to have similar levels of variation
pca_res <- prcomp(filt_gene_expr, scale = TRUE)
```

```
pca_df <- as.data.frame(pca_res$x)
pca_df <- cbind(comb_df[, c("disease_status", "sex")], pca_df)

pca_df %>%
  mutate(
    sex = factor(
      sex,
      levels = c("female", "male", "unknown"),
      labels = c("Female", "Male", "Unknown")
    ),
    disease_status = factor(
      disease_status,
      levels = c("disease state: COVID-19", "disease state: non-COVID-19"),
      labels = c("Disease State: COVID-19", "Disease State: Non-COVID-19")
    )
  ) %>%
ggplot(mapping = aes(x = PC1, y = PC2, color = disease_status)) +
  #After mapping, the next step is to add a geom layer to tell R what type of plot you w
ant to make
  geom_point() +
  #"labs" gives labels to x and y-axis and you can include a title.
  labs(x = "PC1", y = "PC2", title = "PCA Plot of Gene Expression Across Individuals",
       color = "Disease Status") +
  #We can choose any theme. BW gives black axis and text and white background
  theme_bw() +
  #Use theme again to center title and bold it. "Element text" function says that the el
ement in the plot title is a text element. "hjust" is horizontal adjustment (0 = all the
way to the left, 1 = all the way to the right, so center = 0.50)
  theme(
     plot.title = element_text(face = "bold", hjust = 0.50)
  )
```


PCA Plot of Gene Expression Across Individuals
10
5
0
-5
PC2
-10
0
10
PC1
Disease Status
Disease State: COVID-19
Disease State: Non-COVID-19

```
pca_df %>%
  mutate(
    sex = factor(
      sex,
      levels = c("female", "male", "unknown"),
      labels = c("Female", "Male", "Unknown")
    ),
    disease_status = factor(
      disease_status,
      levels = c("disease state: COVID-19", "disease state: non-COVID-19"),
      labels = c("Disease State: COVID-19", "Disease State: Non-COVID-19")
    )
  ) %>%
ggplot(mapping = aes(x = PC1, y = PC2, color = sex)) +
  #After mapping, the next step is to add a geom layer to tell R what type of plot you w
ant to make
  geom_point() +
  #"labs" gives labels to x and y-axis and you can include a title.
  labs(x = "PC1", y = "PC2", title = "PCA Plot of Gene Expression Across Individuals",
       color = "Sex") +
  #We can choose any theme. BW gives black axis and text and white background
  theme_bw() +
  #Use theme again to center title and bold it. "Element text" function says that the el
ement in the plot title is a text element. "hjust" is horizontal adjustment (0 = all the
way to the left, 1 = all the way to the right, so center = 0.50)
  theme(
     plot.title = element_text(face = "bold", hjust = 0.50)
  )
```




```
#Citing R packages
citation("ggplot2")
```

```
## To cite ggplot2 in publications, please use
##
##   H. Wickham. ggplot2: Elegant Graphics for Data Analysis.
##   Springer-Verlag New York, 2016.
##
## A BibTeX entry for LaTeX users is
##
##   @Book{,
##     author = {Hadley Wickham},
##     title = {ggplot2: Elegant Graphics for Data Analysis},
##     publisher = {Springer-Verlag New York},
##     year = {2016},
##     isbn = {978-3-319-24277-4},
##     url = {https://ggplot2.tidyverse.org},
##   }
```

```
citation("tidyverse")
```

```
## To cite package 'tidyverse' in publications use:
##
##   Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R,
##   Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller
##   E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V,
##   Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to
##   the tidyverse." _Journal of Open Source Software_, *4*(43), 1686.
##   doi:10.21105/joss.01686 <https://doi.org/10.21105/joss.01686>.
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {Welcome to the {tidyverse}},
##     author = {Hadley Wickham and Mara Averick and Jennifer Bryan and Winston Chang an
d Lucy D'Agostino McGowan and Romain François and Garrett Grolemund and Alex Hayes and L
ionel Henry and Jim Hester and Max Kuhn and Thomas Lin Pedersen and Evan Miller and Step
han Milton Bache and Kirill Müller and Jeroen Ooms and David Robinson and Dana Paige Sei
del and Vitalie Spinu and Kohske Takahashi and Davis Vaughan and Claus Wilke and Kara Wo
o and Hiroaki Yutani},
##     year = {2019},
##     journal = {Journal of Open Source Software},
##     volume = {4},
##     number = {43},
##     pages = {1686},
##     doi = {10.21105/joss.01686},
##   }
```

```
citation("data.table")
```

```
## To cite package 'data.table' in publications use:
##
##   Barrett T, Dowle M, Srinivasan A, Gorecki J, Chirico M, Hocking T,
##   Schwendinger B (2024). _data.table: Extension of `data.frame`_. R
##   package version 1.16.4,
##   <https://CRAN.R-project.org/package=data.table>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {data.table: Extension of `data.frame`},
##     author = {Tyson Barrett and Matt Dowle and Arun Srinivasan and Jan Gorecki and Mi
chael Chirico and Toby Hocking and Benjamin Schwendinger},
##     year = {2024},
##     note = {R package version 1.16.4},
##     url = {https://CRAN.R-project.org/package=data.table},
##   }
```

```
citation("glue")
```

```
## To cite package 'glue' in publications use:
##
##   Hester J, Bryan J (2024). _glue: Interpreted String Literals_. R
##   package version 1.8.0, <https://CRAN.R-project.org/package=glue>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {glue: Interpreted String Literals},
##     author = {Jim Hester and Jennifer Bryan},
##     year = {2024},
##     note = {R package version 1.8.0},
##     url = {https://CRAN.R-project.org/package=glue},
##   }
```

```
citation("knitr")
```

```
## To cite package 'knitr' in publications use:
##
##   Xie Y (2024). _knitr: A General-Purpose Package for Dynamic Report
##   Generation in R_. R package version 1.49, <https://yihui.org/knitr/>.
##
##   Yihui Xie (2015) Dynamic Documents with R and knitr. 2nd edition.
##   Chapman and Hall/CRC. ISBN 978-1498716963
##
##   Yihui Xie (2014) knitr: A Comprehensive Tool for Reproducible
##   Research in R. In Victoria Stodden, Friedrich Leisch and Roger D.
##   Peng, editors, Implementing Reproducible Computational Research.
##   Chapman and Hall/CRC. ISBN 978-1466561595
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation("kableExtra")
```

```
## To cite package 'kableExtra' in publications use:
##
##   Zhu H (2024). _kableExtra: Construct Complex Table with 'kable' and
##   Pipe Syntax_. R package version 1.4.0,
##   <https://CRAN.R-project.org/package=kableExtra>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {kableExtra: Construct Complex Table with 'kable' and Pipe Syntax},
##     author = {Hao Zhu},
##     year = {2024},
##     note = {R package version 1.4.0},
##     url = {https://CRAN.R-project.org/package=kableExtra},
##   }
```

```
citation("ComplexHeatmap")
```

```
## The methods within the code package can be cited as:
##
##   Gu, Z. (2016) Complex heatmaps reveal patterns and correlations in
##   multidimensional genomic data. Bioinformatics.
##
##   Gu, Z. (2022) Complex Heatmap Visualization. iMeta.
##
## This free open-source software implements academic research by the
## authors and co-workers. If you use it, please support the project by
## citing the appropriate journal articles.
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```