Emergency Room Visits

In this project, you will perform exploratory data analysis (EDA) on a subset of the Pediatric Clinically Important Traumatic Brain Injury (ciTBI) dataset. This dataset is from The Pediatric Emergency Care Applied Research Network (PECARN), a well-known multi-center research organization focused on the prevention and management of acute illnesses and injuries in children across the continuum of emergency medicine health care.

The dataset is available in the `data` folder of the repository as a CSV file named `citbi.csv`. Additionally, a data dictionary is provided in the same folder as a spreadsheet file named `citbi_data_dictionary.xlsx`. This data dictionary contains detailed descriptions of each variable in the dataset.

The dataset contains information on children who visited the emergency department with head trauma. Your task is three fold:

- Clean the dataset so that it can be analyzed effectively.
- Explore and explain interesting patterns in the dataset. This process is called exploratory data analysis (EDA), a critical step in any data science project.
- Report your findings to a hypothetical client, **Dr. Bayes**, a pediatric emergency physician and researcher who works with PECARN.

While you are completing each part of the project, you will have a few files to work in and submit via Github Classroom:

- **preparation.qmd**: The document containing your answers to each question.
- **report.qmd**: A document which will show selected data products to the client.
- **report.pdf** A rendered PDF of the `report.qmd` file.
  - Ensure your report has `format: typst` in the yaml header for it to render properly. Typst is a new markup based typesetting system similar to LaTeX. Although you won't be writing any typst code in your report, putting it in the yaml header will give Quarto the ability to render a nicely formatted PDF.

Part I: Data Cleaning

Hello, Stat 133 student! I'm Dr. Bayes, a pediatric emergency physician and researcher. I work with a large network of hospitals called PECARN studying children with traumatic brain injuries (TBI). I am currently dealing with a large and messy dataset and I need your help to make sense of it.

I have given you access to the dataset in the data folder of this repository.

The dataset contains information on children who visited the emergency department with head trauma. I want to understand the characteristics of these patients and their injuries. I also want to identify any patterns or trends that may help us better understand pediatric ciTBI.

To start, you will need to **clone** the github repository that contains the starter code and data. You can find this in the invite link in the project 3 assignment on **Ed**. To do so, you will need to run the navigate to `File > New Folder from Git`, and paste your GH Classroom project repository link.

Your first step will be to clean the dataset in the `preparation.qmd` file. This includes (but is not limited to) the following tasks:

1. Address Missing Data:
   - How is missing data being represented? *Hint: Look at the data dictionary.*
   - Notice that some columns have a special 91 value that indicates a pre-verbal patients (children who haven't started speaking yet), some columns have a 92 that could be specific to that variable.

Sample Solution:

```
# Handling Missing Values

# packages
library(tidyverse)
```

```
Warning: package 'ggplot2' was built under R version 4.3.3
```

```
Warning: package 'purrr' was built under R version 4.3.3
```

```
Warning: package 'lubridate' was built under R version 4.3.3
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
```

```
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.2     ✔ tibble    3.2.1
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.0.4
── 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
```

```
citbi <- read_csv("./data/citbi.csv")
```

```
Rows: 30379 Columns: 26
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
dbl (26): PatNum, Amnesia_verb, LocLen, Seiz, SeizLen, ActNorm, HA_verb, Vom...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# See columns with `92` na values:
na_92 <- citbi |>
    select(LocLen, SeizLen)

# Here we see that there are 25586 `92` rows, and 1812 `NA` rows.
na_92 |>
    group_by(LocLen) |>
    summarize(n = n())
```

```
# A tibble: 6 × 2
  LocLen     n
   <dbl> <int>
1      1   660
2      2  1308
3      3   754
4      4   259
5     92 25586
6     NA  1812
```

```
# Here we see that there are 29972 `92` rows, and 74 `NA` rows.
na_92 |>
    group_by(SeizLen) |>
    summarize(n = n())
```

```
```

```
# A tibble: 6 × 2
  SeizLen     n
    <dbl> <int>
1       1   179
2       2   119
3       3    21
4       4    14
5      92 29972
6      NA    74
```

```
# Replace these with NAs. May choose different method to do this, may decide to fill
NAs with 92. Etc.
citbi <- citbi |>
    mutate(
        LocLen = ifelse(LocLen == 92, NA, LocLen),
        SeizLen = ifelse(SeizLen == 92, NA, SeizLen)
    )

# Investigate `91` values:
na_91 <- citbi |>
    select(Amnesia_verb, HA_verb)

# Here we see that there are 10352 `91` rows, and 1581 `NA` rows.
na_91 |>
    group_by(Amnesia_verb) |>
    summarize(n = n())
```

```
# A tibble: 4 × 2
  Amnesia_verb     n
         <dbl> <int>
1            0 15279
2            1  3167
3           91 10352
4           NA  1581
```

```
# 9802 `91` rows, and 450 `NA` rows.
na_91 |>
    group_by(HA_verb) |>
    summarize(n = n())
```

```
# A tibble: 4 × 2
  HA_verb     n
    <dbl> <int>
1       0 11121
2       1  9006
```

```
3      91  9802
4      NA   450
```

```
# Again this method may vary.
citbi <- citbi |>
    mutate(
        Amnesia_verb = ifelse(Amnesia_verb == 91, NA, Amnesia_verb),
        HA_verb = ifelse(HA_verb == 91, NA, HA_verb)
    )
```

2. Change Variable Names:

- Consider changing variable names. For example, names like "Clav" are unclear. A better name could be "clavicle_trauma".

Sample Solution:

```
# View column names:

column_names_citbi <- colnames(citbi)
print(column_names_citbi)
```

```
 [1] "PatNum"       "Amnesia_verb" "LocLen"       "Seiz"         "SeizLen"
 [6] "ActNorm"      "HA_verb"      "Vomit"        "Dizzy"        "GCSEye"
[11] "GCSVerbal"    "GCSMotor"     "GCSTotal"     "AMS"          "SFxPalp"
[16] "FontBulg"     "Hema"         "Clav"         "NeuroD"       "OSI"
[21] "CTForm1"      "AgeInMonth"   "Gender"       "CTDone"       "DeathTBI"
[26] "PosIntFinal"
```

```
# Current names: PatNum, Amnesia_verb, LocLen, Seiz, SeizLen, ActNorm, HA_verb, Vomit,
Dizzy, GCSEye, GCSVerbal, GCSMotor, GCSTotal, AMS, SFxPalp, FontBulg, Hema, Clav,
NeuroD, OSI, CTForm1, AgeInMonth, Gender, CTDone, DeathTBI, PosIntFinal

# Potentially Better names: patient_number, amnesia, loss_of_consciousness_length,
seizure, seizure_length, acting_normal, headache, vomiting, dizziness,
gcs_eye, gcs_verbal, gcs_motor, gcs_total, altered_mental_status,
skull_fracture, fontanelle_bulging, hematoma, clavicle_trauma, neurological_deficit,
other_significant_injury, ct_form, age_in_month, gender, ct_done, death_tbi,
citbi_outcome

# To rename columns, we can use functions from dplyr. Again, note that there are many
column names to choose, student can theoretically rename just one column if they want.
citbi <- citbi |>
    rename( # dplyr function
```

```
        patient_number = PatNum, amnesia = Amnesia_verb, loss_of_consciousness_length
= LocLen,
        seizure = Seiz, seizure_length = SeizLen, acting_normal = ActNorm,
        headache = HA_verb, vomiting = Vomit, dizziness = Dizzy,
        gcs_eye = GCSEye, gcs_verbal = GCSVerbal, gcs_motor = GCSMotor,
        gcs_total = GCSTotal, altered_mental_status = AMS, skull_fracture = SFxPalp,
        fontanelle_bulging = FontBulg, hematoma = Hema, clavicle_trauma = Clav,
      neurological_deficit = NeuroD, other_significant_injury = OSI, ct_form = CTForm1,
        age_in_month = AgeInMonth, gender = Gender, ct_done = CTDone,
        death_tbi = DeathTBI, citbi_outcome = PosIntFinal
    )

# can also be done using mutate and dropping old columns, e.g.:
# citbi <- citbi |>
#     mutate(
#        patient_number = PatNum, amnesia = Amnesia_verb, loss_of_consciousness_length
= LocLen,
#         seizure = Seiz, seizure_length = SeizLen, acting_normal = ActNorm,
#         headache = HA_verb, vomiting = Vomit, dizziness = Dizzy, ...
#     ) |>
#     select(-PatNum, -Amnesia_verb, -LocLen, -Seiz, -SeizLen, -ActNorm,
#            -HA_verb, -Vomit, -Dizzy, -GCSEye, -GCSVerbal, -GCSMotor,
#            -GCSTotal, -AMS, -SFxPalp, -FontBulg, -Hema, -Clav, ...
#     )

# View new column names:
print(colnames(citbi))
```

```
 [1] "patient_number"               "amnesia"
 [3] "loss_of_consciousness_length" "seizure"
 [5] "seizure_length"               "acting_normal"
 [7] "headache"                     "vomiting"
 [9] "dizziness"                    "gcs_eye"
[11] "gcs_verbal"                   "gcs_motor"
[13] "gcs_total"                    "altered_mental_status"
[15] "skull_fracture"               "fontanelle_bulging"
[17] "hematoma"                     "clavicle_trauma"
[19] "neurological_deficit"         "other_significant_injury"
[21] "ct_form"                      "age_in_month"
[23] "gender"                       "ct_done"
[25] "death_tbi"                    "citbi_outcome"
```

3. Specify Variable Types:
   - Identify character, logical, and numeric vectors. Convert data types where appropriate.
   - Consider converting character vectors to factors.

Sample Solution:

```
# Helpful to see which columns have no NA values:
citbi |>
    summarise(across(everything(), ~sum(is.na(.))))
```

```
# A tibble: 1 × 26
  patient_number amnesia loss_of_consciousness_length seizure seizure_length
           <int>   <int>                        <int>   <int>          <int>
1              0   11933                        27398     630          30046
# ℹ 21 more variables: acting_normal <int>, headache <int>, vomiting <int>,
#   dizziness <int>, gcs_eye <int>, gcs_verbal <int>, gcs_motor <int>,
#   gcs_total <int>, altered_mental_status <int>, skull_fracture <int>,
#   fontanelle_bulging <int>, hematoma <int>, clavicle_trauma <int>,
#   neurological_deficit <int>, other_significant_injury <int>, ct_form <int>,
#   age_in_month <int>, gender <int>, ct_done <int>, death_tbi <int>,
#   citbi_outcome <int>
```

```
# Equivalent (but slower):
# for (column in colnames(citbi)) {
#     if (sum(is.na(citbi[[column]])) == 0) {
#         print(column)
#     }
# }

# "patient_number", "gcs_total", "age_in_month", "ct_done"

# Convert yes/no columns to logicals
citbi <- citbi |>
    mutate(
        seizure = as.logical(seizure),
        acting_normal = as.logical(acting_normal),
        vomiting = as.logical(vomiting),
        dizziness = as.logical(dizziness),
        altered_mental_status = as.logical(altered_mental_status),
        skull_fracture = as.logical(skull_fracture),
        fontanelle_bulging = as.logical(fontanelle_bulging),
        hematoma = as.logical(hematoma),
        clavicle_trauma = as.logical(clavicle_trauma),
        neurological_deficit = as.logical(neurological_deficit),
        other_significant_injury = as.logical(other_significant_injury),
        ct_form = as.logical(ct_form),
        ct_done = as.logical(ct_done),
        death_tbi = as.logical(death_tbi),
        citbi_outcome = as.logical(citbi_outcome)
    )
```

```r
# view changes
citbi |>
    select(
        seizure, acting_normal, vomiting, dizziness, altered_mental_status,
        skull_fracture, fontanelle_bulging, hematoma, clavicle_trauma,
        neurological_deficit, other_significant_injury, ct_form, ct_done,
        death_tbi, citbi_outcome
    ) |>
    head()
```

```
# A tibble: 6 × 15
  seizure acting_normal vomiting dizziness altered_mental_status skull_fracture
  <lgl>   <lgl>         <lgl>    <lgl>     <lgl>                 <lgl>
1 FALSE   TRUE          FALSE    NA        FALSE                 FALSE
2 FALSE   TRUE          FALSE    NA        FALSE                 FALSE
3 NA      NA            FALSE    TRUE      FALSE                 FALSE
4 FALSE   TRUE          FALSE    FALSE     FALSE                 FALSE
5 FALSE   TRUE          FALSE    FALSE     FALSE                 FALSE
6 FALSE   TRUE          FALSE    FALSE     FALSE                 FALSE
# ℹ 9 more variables: fontanelle_bulging <lgl>, hematoma <lgl>,
#   clavicle_trauma <lgl>, neurological_deficit <lgl>,
#   other_significant_injury <lgl>, ct_form <lgl>, ct_done <lgl>,
#   death_tbi <lgl>, citbi_outcome <lgl>
```

```r
# Length of loss of consciousness:
citbi <- citbi |>
    mutate(
        loss_of_consciousness_length = factor(loss_of_consciousness_length,
            levels = c(1, 2, 3, 4),
            labels = c("<5 sec", "5 sec to <1 min", "1-5 min", ">5 min"))
    )

# Length of seizure:
citbi <- citbi |>
    mutate(
        seizure_length = factor(seizure_length,
            levels = c(1, 2, 3, 4),
            labels = c("<1 min", "1-5 min", "5-15 min", ">15 min"))
    )

# Lastly, for gender:
citbi <- citbi |>
    mutate(
        gender = factor(gender,
            levels = c(1, 2),
            labels = c("Male", "Female")
        )
```

```
    )

# View non-yes/no factor columns:
citbi |>
    select(loss_of_consciousness_length, seizure, gender) |>
    head()
```

```
# A tibble: 6 × 3
  loss_of_consciousness_length seizure gender
  <fct>                        <lgl>   <fct>
1 <NA>                         FALSE   Male
2 <NA>                         FALSE   Female
3 <NA>                         NA      Male
4 <NA>                         FALSE   Male
5 <5 sec                       FALSE   Male
6 <NA>                         FALSE   Female
```

Part II: Exploratory Data Analysis (EDA)

Great work cleaning the dataset! Next, I would like for you to perform exploratory data analysis (EDA). This is a critical step for most projects in data science. The goal of EDA is to better understand the data, identify patterns, and generate hypotheses. Additionally, EDA presents an opportunity to create data products that can be shared with others. (Still working withing the `preparation.qmd` file)

4. Create a Missing Data Summary:

- In Positron, navigate to the `session` tab in the top right where you can view a list of objects saved in your environment. Click the spreadsheet icon to the right of the data frame name you created in part I. You should be launched into a new tab with a split pane view. On one side, you will see mini-histograms of each variable along with the proportion of missing values. If you click onto the variables you can see some useful summary statistics. On the other pane, you can see the data frame itself.
- Now, **create a table** that summarizes the number and the proportion of missing values for each variable in the dataset. Does it match up with what you saw in the positron summary tab?

Sample Solution:

```
# Missing Data Summary
missing_summary <- citbi |>
  summarize(across(everything(),
      list(
          "nmissing" = ~sum(is.na(.)),
          "propmissing" = ~mean(is.na(.))
      ),
      .names = "{.col}---{.fn}"  # Use a unique separator
  )) |>
  pivot_longer(everything(),
      names_to = c("variable", ".value"),
      names_sep = "---"  # Match the unique separator
  ) |>
  arrange(desc(nmissing))

# Equivalent (but slower):
# missing_summary <- tibble(
#     variable = character(length = ncol(citbi)),
#     n_missing = integer(length = ncol(citbi)),
#     prop_missing = double(length = ncol(citbi))
# )

# for (i in 1:length(citbi)) {
#     column <- colnames(citbi)[i]
#     missing_summary$variable[i] <- column
#     missing_summary$n_missing[i] <- sum(is.na(citbi[[column]]))
```

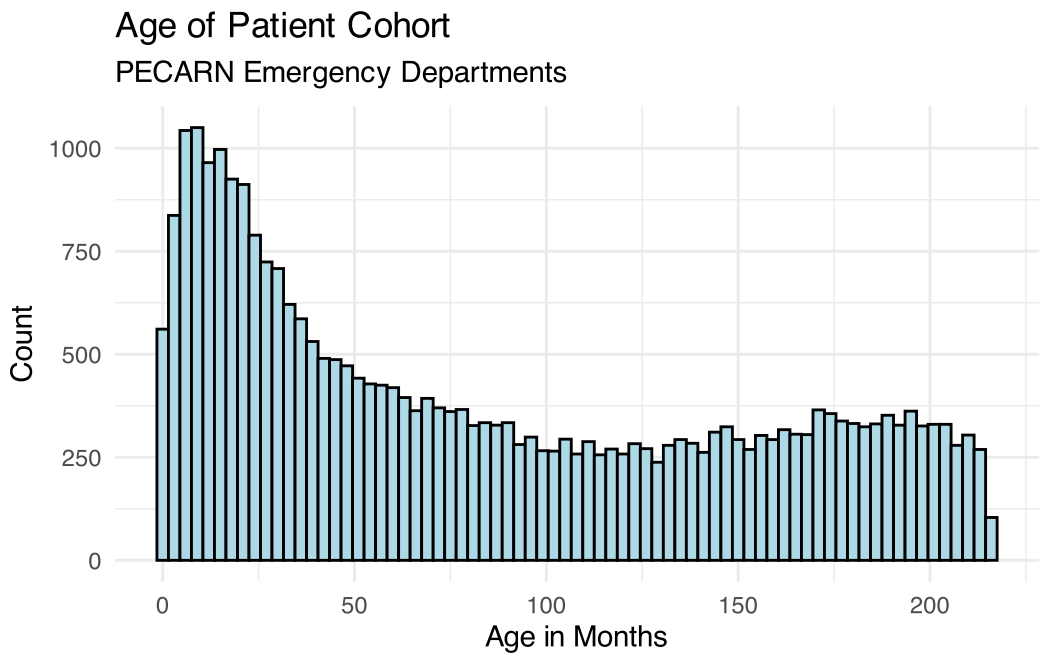
```
#     missing_summary$prop_missing[i] <- mean(is.na(citbi[[column]]))
# }
# missing_summary <- missing_summary |>
#     arrange(desc(n_missing))

missing_summary
```

```
# A tibble: 26 × 3
   variable                     nmissing propmissing
   <chr>                           <int>       <dbl>
 1 seizure_length                  30046      0.989
 2 loss_of_consciousness_length    27398      0.902
 3 amnesia                         11933      0.393
 4 dizziness                       11143      0.367
 5 headache                        10252      0.337
 6 acting_normal                    2324      0.0765
 7 gcs_motor                         915      0.0301
 8 gcs_verbal                        907      0.0299
 9 gcs_eye                           900      0.0296
10 seizure                           630      0.0207
# ℹ 16 more rows
```

5. Create a histogram of the patient ages in months. Describe any interesting patterns you see.

Sample Solution:

```
citbi |>
    ggplot(aes(x = age_in_month)) +
    geom_histogram(binwidth = 3, color = "black", fill = "lightblue") +
    labs(
        title = "Age of Patient Cohort",
        subtitle = "PECARN Emergency Departments",
        x = "Age in Months",
        y = "Count"
    ) +
    theme_minimal()
```



6. Create a grouped summary table that shows the total count of patients for every combination of the loss of consciousness length and ciTBI outcome columns.

Sample Solution:

```
loc_len_citbi <- citbi |>
    group_by(loss_of_consciousness_length, citbi_outcome) |>
    summarize(n = n()) |>
    ungroup()
```

```
`summarise()` has grouped output by 'loss_of_consciousness_length'. You can
override using the `.groups` argument.
```

```
# May decide to pivot wider, remove NA rows, etc.
loc_len_citbi
```

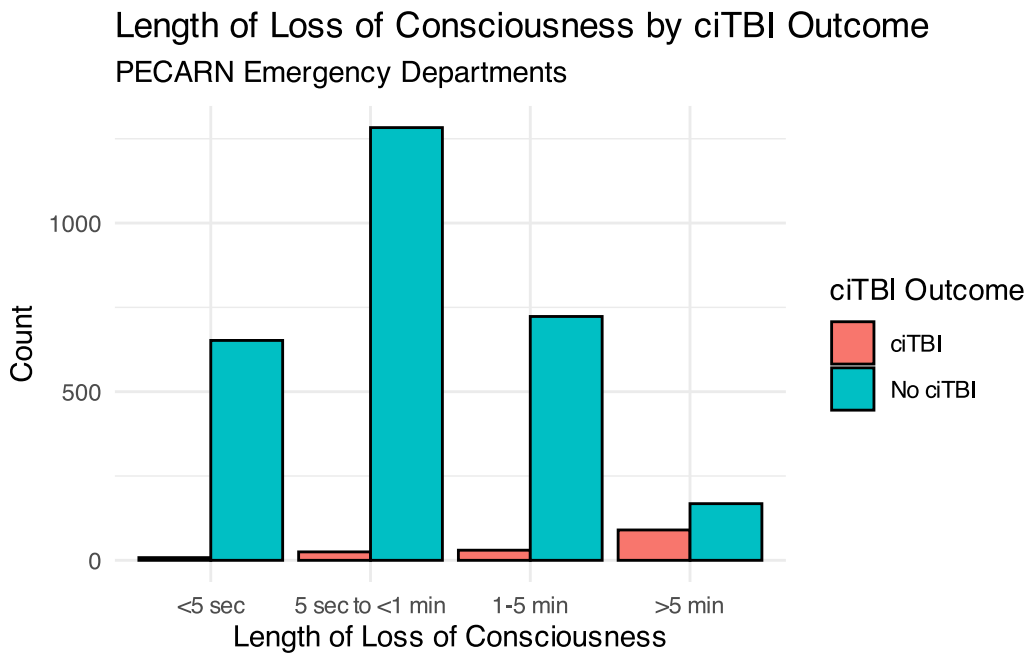
```
# A tibble: 13 × 3
   loss_of_consciousness_length citbi_outcome     n
   <fct>                        <lgl>         <int>
 1 <5 sec                       FALSE           652
 2 <5 sec                       TRUE              8
 3 5 sec to <1 min              FALSE          1283
 4 5 sec to <1 min              TRUE             25
 5 1-5 min                      FALSE           723
```

```
 6 1-5 min                    TRUE            30
 7 1-5 min                    NA               1
 8 >5 min                     FALSE          168
 9 >5 min                     TRUE            90
10 >5 min                     NA               1
11 <NA>                       FALSE        26993
12 <NA>                       TRUE           394
13 <NA>                       NA              11
```

7. Create a bar chart to visualize the count for each category side-by-side in length of loss of consciousness by ciTBI outcome.

Sample Solution:

```
# using citbi dataset
citbi |>
    drop_na(citbi_outcome, loss_of_consciousness_length) |>
    mutate(citbi_outcome = ifelse(citbi_outcome, "ciTBI", "No ciTBI")) |>
    ggplot(aes(x = loss_of_consciousness_length, fill = citbi_outcome)) +
    geom_bar(position = "dodge", color = "black") +
    labs(
        title = "Length of Loss of Consciousness by ciTBI Outcome",
        subtitle = "PECARN Emergency Departments",
        x = "Length of Loss of Consciousness",
        y = "Count",
        fill = "ciTBI Outcome"
    ) +
    theme_minimal()
```

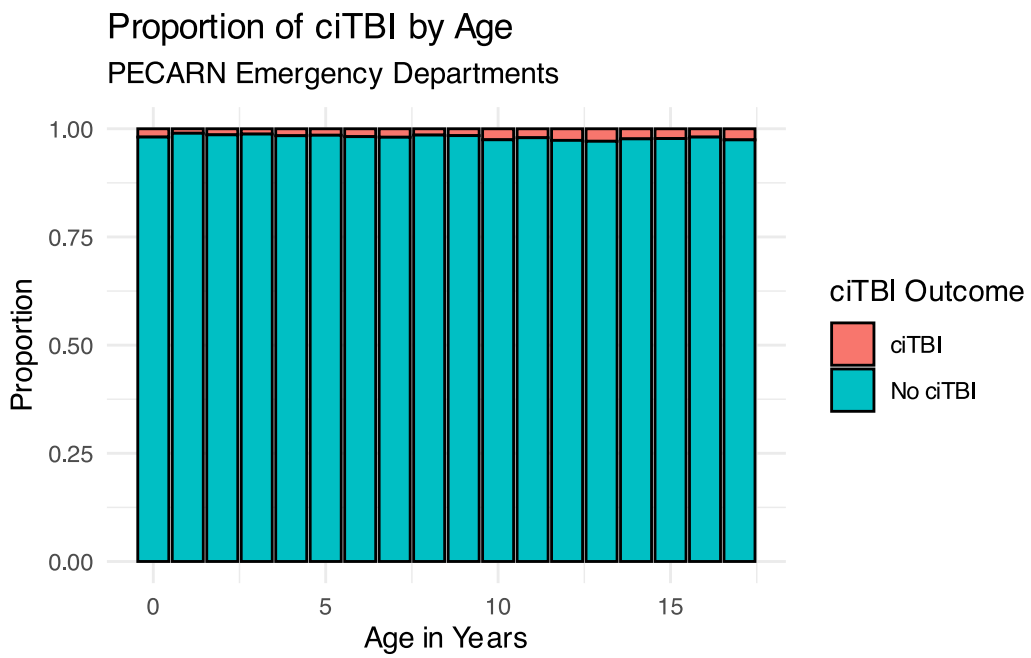



```
# using table from previous problem
# loc_len_citbi |>
#     drop_na(citbi_outcome, loss_of_consciousness_length) |>
#     mutate(citbi_outcome = ifelse(citbi_outcome, "ciTBI", "No ciTBI")) |>
#     ggplot(aes(x = loss_of_consciousness_length, y = n, fill = citbi_outcome)) +
#     geom_bar(position = "dodge", stat = "identity", color = "black") +
#     labs(
#         title = "Length of Loss of Consciousness by ciTBI Outcome",
#         subtitle = "PECARN Emergency Departments",
#         x = "Length of Loss of Consciousness",
#         y = "Count",
#         fill = "ciTBI Outcome"
#     ) +
#     theme_minimal()
```

8. The variable (originally) named "GCSTotal" refers to the Glasgow Coma Scale (GCS) score, a neurological assessment used to evaluate the patient's level of consciousness. Create three visualizations for the relationship between total GCS score, Age in years, and ciTBI outcome.
   a. Create a stacked normalized bar chart to visualize how the proportion of patients with ciTBI varies across different ages.
   b. Explore whether the relationship between age and ciTBI differs across Glasgow Coma Scale scores. Create a stacked normalized bar chart that shows the proportion of patients with ciTBI across age, now faceted by total GCS score.
   c. Create a stacked bar chart that shows the number of patients with ciTBI across age faceted by total GCS score.

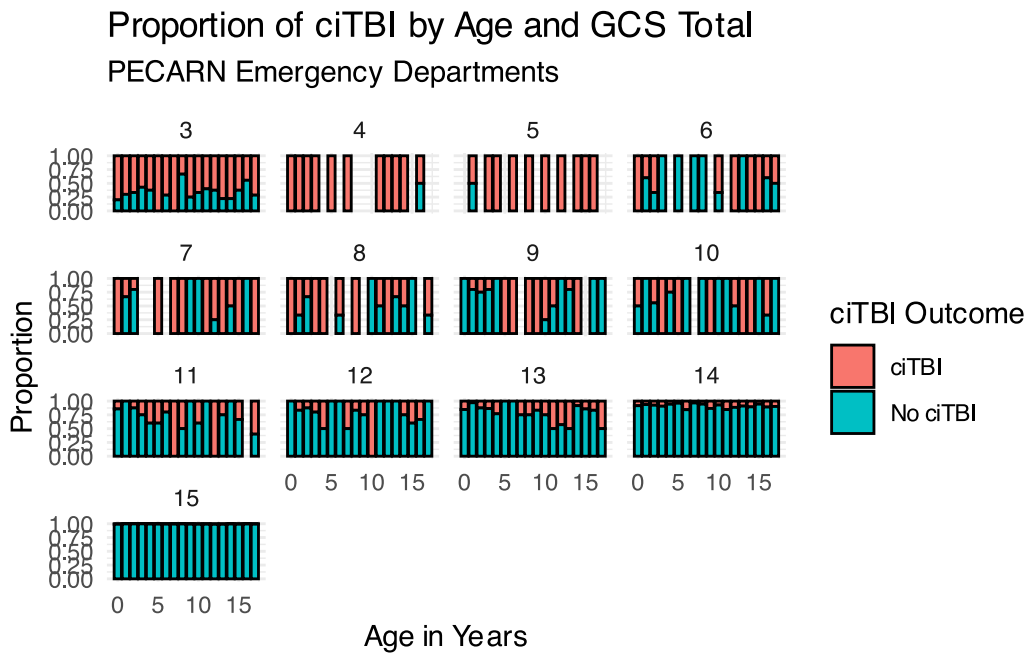
Sample Solution:

```
citbi <- citbi |>
    mutate(age_in_years = as.integer(age_in_month / 12))

# a: segmented bar chart no facet
citbi |>
    drop_na(citbi_outcome, gcs_total) |>
    mutate(citbi_outcome = ifelse(citbi_outcome, "ciTBI", "No ciTBI")) |>
    ggplot(aes(x = age_in_years, fill = citbi_outcome)) +
    geom_bar(position = "fill", color = "black") +
    labs(
        title = "Proportion of ciTBI by Age",
        subtitle = "PECARN Emergency Departments",
        x = "Age in Years",
        y = "Proportion",
        fill = "ciTBI Outcome"
    ) +
    theme_minimal()
```

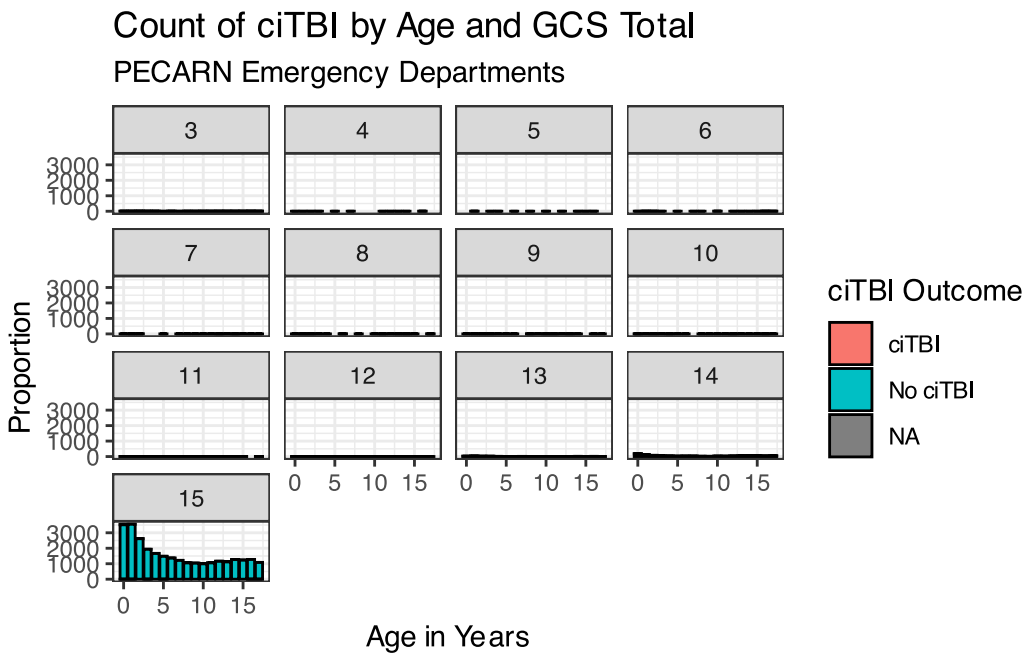



```
# b: segmented bar chart facet by gcs_total
citbi |>
    drop_na(citbi_outcome, gcs_total) |>
    mutate(citbi_outcome = ifelse(citbi_outcome, "ciTBI", "No ciTBI")) |>
    ggplot(aes(x = age_in_years, fill = citbi_outcome)) +
    geom_bar(position = "fill", color = "black") +
    labs(
```

```
        title = "Proportion of ciTBI by Age and GCS Total",
        subtitle = "PECARN Emergency Departments",
        x = "Age in Years",
        y = "Proportion",
        fill = "ciTBI Outcome"
    ) +
    facet_wrap(~gcs_total) +
    theme_minimal()
```



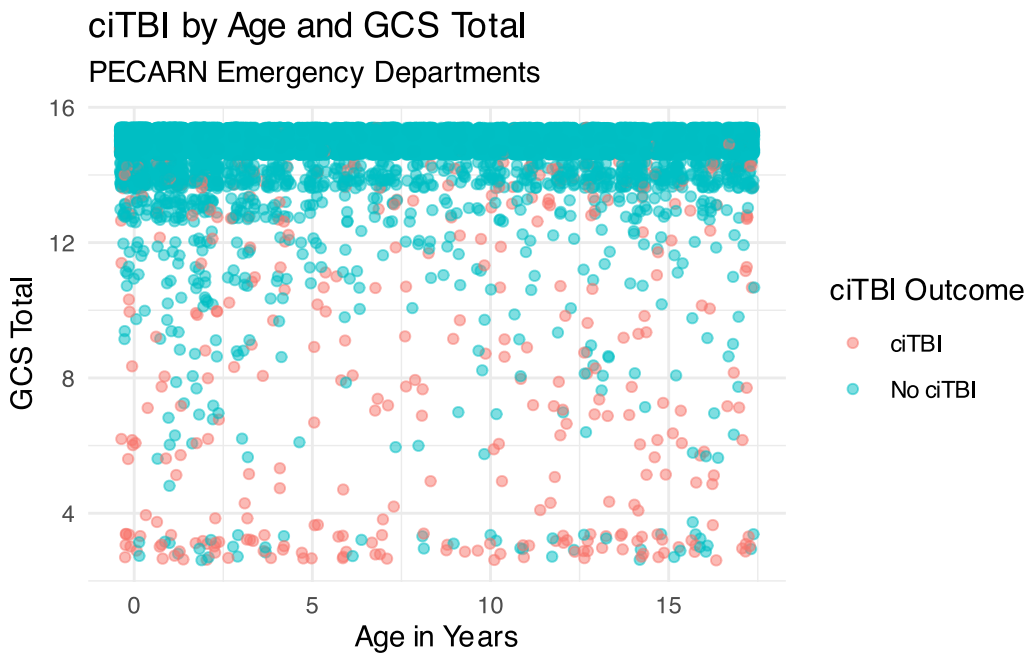

```
# c:
citbi |>
    mutate(citbi_outcome = ifelse(citbi_outcome, "ciTBI", "No ciTBI")) |>
    ggplot(aes(x = age_in_years, fill = factor(citbi_outcome))) +
    geom_bar(color = "black") +
    facet_wrap(~gcs_total) +
    labs(
        title = "Count of ciTBI by Age and GCS Total",
        subtitle = "PECARN Emergency Departments",
        x = "Age in Years",
        y = "Proportion",
        fill = "ciTBI Outcome"
    ) +
    theme_bw()
```



9. Create a scatter plot to visualize any two numeric variables in the dataset. Describe any interesting patterns you see.

Sample Solution:

```
citbi |>
    drop_na(citbi_outcome, gcs_total) |>
    mutate(citbi_outcome = ifelse(citbi_outcome, "ciTBI", "No ciTBI")) |>
    ggplot(aes(x = age_in_years, y = gcs_total, color = citbi_outcome)) +
    geom_point(position = "jitter", alpha = 0.5) +
    labs(
        title = "ciTBI by Age and GCS Total",
        subtitle = "PECARN Emergency Departments",
        x = "Age in Years",
        y = "GCS Total",
        color = "ciTBI Outcome"
    ) +
    theme_minimal()
```



10. Create a table that summarizes at least two statistics grouped by two categorical variables. Describe any interesting patterns you see.

Sample Solution:

```
# visualize relationship between seizure and length of loss of consciousness
citbi |>
    group_by(
        seizure, loss_of_consciousness_length
    ) |>
    summarize(
        count = n(),
        average_age = mean(age_in_month, na.rm = TRUE),
        prop_citbi = mean(citbi_outcome, na.rm = TRUE)
    ) |>
    ungroup() |>
    arrange(seizure) |>
    drop_na(seizure)
```

```
`summarise()` has grouped output by 'seizure'. You can override using the
`.groups` argument.
```

```
# A tibble: 10 × 5
   seizure loss_of_consciousness_length count average_age prop_citbi
```

```
   <lgl>   <fct>                      <int>       <dbl>      <dbl>
 1 FALSE   <5 sec                       636       131.      0.0126
 2 FALSE   5 sec to <1 min             1189       126.      0.0177
 3 FALSE   1-5 min                      668       130.      0.0360
 4 FALSE   >5 min                       189       104.      0.249
 5 FALSE   <NA>                       26660        80.1     0.0114
 6 TRUE    <5 sec                        15        69.7     0
 7 TRUE    5 sec to <1 min               84        80.7     0.0357
 8 TRUE    1-5 min                       65       115.      0.0615
 9 TRUE    >5 min                        23        72.3     0.455
10 TRUE    <NA>                         220        76.4     0.114
```

Part III: Explanatory Data Analysis

Now that you've cleaned and explored the data, your last task is to create a short report to communicate your findings to Dr. Bayes. You will need to highlight 3-5 insights from part II.

Your task

Select 3-5 of the most insightful data products from part II. For each data product, you will need to:

- Polish it so that it highlights a key message and is presentable.
- Provide a description below the data product that explains what it shows and why it is important.
- Consider creating new and related data products that can further highlight this insight. What do these new products contribute to the story?

Tips for an effective report

- Consider your audience. Dr. Bayes is a busy physician and medical researcher, not a statistician. He isn't expecting you to be an expert in the field, but he does want to understand the key findings.
- Be concise. Your report should be no more than 2 pages long.