HST.953x Workshop 2.07: Exploratory Data Analysis

H. David Shea

21 Jul 2021

Contents

(Note: Updated and modified from the hst953-edx github version.)

Principles of Exploratory Data Analysis (EDA)

EDA's goal is to better understand the data and the process by which it was generated.

Within statistics, it is largely considered separate from inferential/confirmatory statistics (e.g., hypothesis testing, point and interval estimates, etc), where EDA has a very diverse and important set of goals:

- Provide an opportunity to do additional data cleaning.
- Understand how the data is generated, and what the relationships between variables may be.
- Suggest questions and hypotheses that can be subsequently answered and tested.
- Identify what statistical methods may be most appropriate for the data to follow up with these questions and hypotheses.

EDA was coined, developed and advocated for by John Tukey. His book, entitled *"Exploratory Data Analysis"* was published in 1977, and is still in use today. It may seem like an oddity, but it was a fundamental change in how data science / statistics was done. Fundamentally he sums up EDA with this quote:

"It is important to understand what you CAN DO, before you learn to measure how WELL you seem to have DONE it." – J. W. Tukey (1977)

If you don't understand the data, it becomes difficult to know how to analyze it. Confirmatory and exploratory analyses are not superior or inferior to one another, rather they are complementary. With all the tools available to do both, ignoring one of them is inexcusable.

"Today, exploratory and confirmatory (analysis) – can – and should – proceed side by side." – J. W. Tukey (1977)

Cognitive Disfluency – make it work for you?

There is often an urge due to productivity, laziness, or other factors to plow through with an analysis, using sophisticated analysis techniques to find the results you are seeking. With the proliferation of large datasets, this can be quite ineffective, as it largely separates the analyst from the data, resulting in misunderstanding or not understanding the data at all.

There is some evidence that *cognitive disfluency* (making it harder to learn) can lead to deeper learning. For analysts and data scientists this means slowing things down, often using basic (and sometimes tedious) methods to integrate the primary structure and relationships contained in the data, before pulling out the heavy machinery of modern data analysis.

See: *Alter, A.L., 2013. The benefits of cognitive disfluency. Current Directions in Psychological Science, 22(6), pp.437-442.*

All too often the success/failure of an analysis is determined from a single number, when in reality, understanding the data should be the goal.

Prerequisites

For this workshop we will use data from a study that examined the effect of indwelling arterial catheters (aka IAC or aline) on 28 day mortality in intensive care unit (ICU) patients on mechanical ventilation during the first day of ICU admission. The data originates from MIMIC II v2.6. The data is ready for exploratory data analysis (the data extraction and cleaning have already been completed), and contained in a comma separated value (.csv) file generated after this process and stored on Physionet. Start by loading the data file from Physionet into a data frame called `dat`:

```r
fnm <- fs::path(base_dir, "course_exercises/exploratory_data_analysis/aline_full_cohort_data.csv")
dat <- tibble(read.csv(fnm))
rm(fnm)

# Public dataset has NA values for variables required to complete workshop
# Replace NA values with defaults since dataset only intended for teaching
dat <- dat %>%
  mutate(
    gender_num = ifelse(is.na(gender_num), 0L, gender_num),
    sofa_first = ifelse(is.na(sofa_first), 0L, sofa_first)
  )
```

Numerical Forms of EDA

One form of EDA is to provide numerical summaries of the dataset. This can have many purposes:

- To verify the dataset you loaded is the one you think you did.
- To quantify characteristics of the dataset which need to be reported numerically.

The `summary` function in `R`

`R` has a very handy function, which performs differently depending on the type of data structures to whcih you apply it. This is the `summary` function, and it provides a very useful data summary of data frames. This comes in the form of five-number summaries (plus the mean) (min-Q1-mean-median-Q3-max) for numeric data and counts for categorical (factor) data. Summary also works on many types of objects in `R`, and when you don't know what to do with an `R` object (`obj`), it is often good to try `summary(obj)`.

```
summary(dat)
#>    aline_flg        icu_los_day      hospital_los_day       age       
#>  Min.   :0.0000   Min.   : 0.500   Min.   :  1.000   Min.   :15.18  
#>  1st Qu.:0.0000   1st Qu.: 1.370   1st Qu.:  3.000   1st Qu.:38.25  
#>  Median :1.0000   Median : 2.185   Median :  6.000   Median :53.68  
#>  Mean   :0.5541   Mean   : 3.346   Mean   :  8.111   Mean   :54.38  
#>  3rd Qu.:1.0000   3rd Qu.: 4.003   3rd Qu.: 10.000   3rd Qu.:72.76  
#>  Max.   :1.0000   Max.   :28.240   Max.   :112.000   Max.   :99.11  
#>                                                                      
#>    gender_num      weight_first          bmi          sapsi_first   
#>  Min.   :0.0000   Min.   : 30.00   Min.   :12.78   Min.   : 3.00  
#>  1st Qu.:0.0000   1st Qu.: 65.40   1st Qu.:22.62   1st Qu.:11.00  
#>  Median :1.0000   Median : 77.00   Median :26.32   Median :14.00  
#>  Mean   :0.5771   Mean   : 80.08   Mean   :27.83   Mean   :14.14  
#>  3rd Qu.:1.0000   3rd Qu.: 90.00   3rd Qu.:30.80   3rd Qu.:17.00  
#>  Max.   :1.0000   Max.   :257.60   Max.   :98.80   Max.   :32.00  
#>                   NA's   :110      NA's   :466     NA's   :85     
#>    sofa_first     service_unit        service_num     day_icu_intime    
#>  Min.   : 0.000   Length:1776        Min.   :0.0000   Length:1776       
#>  1st Qu.: 4.000   Class :character   1st Qu.:0.0000   Class :character  
#>  Median : 6.000   Mode  :character   Median :1.0000   Mode  :character  
#>  Mean   : 5.801                      Mean   :0.5529                     
#>  3rd Qu.: 7.000                      3rd Qu.:1.0000                     
#>  Max.   :17.000                      Max.   :1.0000                     
#>                                                                          
#>  day_icu_intime_num hour_icu_intime  hosp_exp_flg     icu_exp_flg     
#>  Min.   :1.000      Min.   : 0.00   Min.   :0.0000   Min.   :0.00000  
#>  1st Qu.:2.000      1st Qu.: 3.00   1st Qu.:0.0000   1st Qu.:0.00000  
#>  Median :4.000      Median : 9.00   Median :0.0000   Median :0.00000  
#>  Mean   :4.054      Mean   :10.59   Mean   :0.1374   Mean   :0.09572  
#>  3rd Qu.:6.000      3rd Qu.:19.00   3rd Qu.:0.0000   3rd Qu.:0.00000  
#>  Max.   :7.000      Max.   :23.00   Max.   :1.0000   Max.   :1.00000  
#>                                                                          
#>    day_28_flg     mort_day_censored   censor_flg       sepsis_flg
#>  Min.   :0.0000   Min.   :   0.0    Min.   :0.0000   Min.   :0   
#>  1st Qu.:0.0000   1st Qu.: 434.3    1st Qu.:0.0000   1st Qu.:0   
#>  Median :0.0000   Median : 731.0    Median :1.0000   Median :0   
#>  Mean   :0.1593   Mean   : 614.3    Mean   :0.7202   Mean   :0   
#>  3rd Qu.:0.0000   3rd Qu.: 731.0    3rd Qu.:1.0000   3rd Qu.:0   
#>  Max.   :1.0000   Max.   :3094.1    Max.   :1.0000   Max.   :0   
#>                                                                   
#>     chf_flg          afib_flg         renal_flg          liver_flg      
#>  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000  
#>  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000  
#>  Median :0.0000   Median :0.0000   Median :0.00000   Median :0.00000  
#>  Mean   :0.1199   Mean   :0.1166   Mean   :0.03378   Mean   :0.05574  
#>  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000
#>  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
#>
#>     copd_flg         cad_flg          stroke_flg        mal_flg
#>  Min.   :0.0000   Min.   :0.00000   Min.   :0.000   Min.   :0.0000
#>  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.000   1st Qu.:0.0000
#>  Median :0.0000   Median :0.00000   Median :0.000   Median :0.0000
#>  Mean   :0.0884   Mean   :0.06926   Mean   :0.125   Mean   :0.1441
#>  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.000   3rd Qu.:0.0000
#>  Max.   :1.0000   Max.   :1.00000   Max.   :1.000   Max.   :1.0000
#>
#>     resp_flg         map_1st           hr_1st          temp_1st
#>  Min.   :0.0000   Min.   :  5.00   Min.   : 30.00   Min.   : 32.00
#>  1st Qu.:0.0000   1st Qu.: 76.67   1st Qu.: 74.75   1st Qu.: 96.90
#>  Median :0.0000   Median : 87.00   Median : 87.00   Median : 98.10
#>  Mean   :0.3181   Mean   : 88.25   Mean   : 87.91   Mean   : 97.79
#>  3rd Qu.:1.0000   3rd Qu.: 99.00   3rd Qu.:100.00   3rd Qu.: 99.30
#>  Max.   :1.0000   Max.   :195.00   Max.   :158.00   Max.   :104.80
#>                                                     NA's   :3
#>     spo2_1st        abg_count         wbc_first        hgb_first
#>  Min.   :  4.00   Min.   :  0.000   Min.   :  0.17   Min.   : 2.00
#>  1st Qu.: 98.00   1st Qu.:  1.000   1st Qu.:  8.20   1st Qu.:11.10
#>  Median :100.00   Median :  3.000   Median : 11.30   Median :12.70
#>  Mean   : 98.43   Mean   :  5.985   Mean   : 12.32   Mean   :12.55
#>  3rd Qu.:100.00   3rd Qu.:  7.000   3rd Qu.: 15.00   3rd Qu.:14.12
#>  Max.   :100.00   Max.   :115.000   Max.   :109.80   Max.   :19.00
#>                                     NA's   :8        NA's   :8
#>  platelet_first   sodium_first   potassium_first   tco2_first
#>  Min.   :  7.0   Min.   :105.0   Min.   :1.900   Min.   : 2.00
#>  1st Qu.:182.0   1st Qu.:137.0   1st Qu.:3.600   1st Qu.:22.00
#>  Median :239.0   Median :140.0   Median :4.000   Median :24.00
#>  Mean   :246.1   Mean   :139.6   Mean   :4.108   Mean   :24.42
#>  3rd Qu.:297.0   3rd Qu.:142.0   3rd Qu.:4.400   3rd Qu.:27.00
#>  Max.   :988.0   Max.   :165.0   Max.   :9.800   Max.   :62.00
#>  NA's   :8       NA's   :5       NA's   :5       NA's   :5
#>  chloride_first    bun_first      creatinine_first   po2_first
#>  Min.   : 78.0   Min.   :  2.00   Min.   : 0.000   Min.   : 22.0
#>  1st Qu.:101.0   1st Qu.: 11.00   1st Qu.: 0.700   1st Qu.:108.0
#>  Median :104.0   Median : 15.00   Median : 0.900   Median :195.0
#>  Mean   :103.8   Mean   : 19.28   Mean   : 1.096   Mean   :227.6
#>  3rd Qu.:107.0   3rd Qu.: 22.00   3rd Qu.: 1.100   3rd Qu.:323.0
#>  Max.   :133.0   Max.   :139.00   Max.   :18.300   Max.   :634.0
#>  NA's   :5       NA's   :5        NA's   :6        NA's   :186
#>    pco2_first        iv_day_1
#>  Min.   :  8.00   Min.   :    0.0
#>  1st Qu.: 36.00   1st Qu.:  329.8
#>  Median : 41.00   Median : 1081.5
#>  Mean   : 43.41   Mean   : 1622.9
#>  3rd Qu.: 47.00   3rd Qu.: 2493.9
#>  Max.   :158.00   Max.   :13910.0
#>  NA's   :186      NA's   :143
```

As you can see, this function is very verbose, but produces some useful output. At this point, it's also a good idea to verify the number of rows and columns are correct:

```
str(dat)
#> tibble [1,776 x 46] (S3: tbl_df/tbl/data.frame)
#>  $ aline_flg         : int [1:1776] 1 0 0 1 1 0 1 1 1 1 ...
#>  $ icu_los_day       : num [1:1776] 7.63 1.14 2.86 0.58 1.75 1.38 7.06 15.3 3.79 7.14 ...
#>  $ hospital_los_day  : int [1:1776] 13 1 5 3 5 9 27 33 4 7 ...
#>  $ age               : num [1:1776] 72.4 64.9 36.5 44.5 23.7 ...
#>  $ gender_num        : int [1:1776] 1 0 0 0 1 1 1 0 0 0 ...
#>  $ weight_first      : num [1:1776] 75 55 70 NA 95.2 72 90 69.7 52.6 61.5 ...
#>  $ bmi               : num [1:1776] 29.9 20.1 27.1 NA 28.5 ...
#>  $ sapsi_first       : int [1:1776] 15 NA 16 21 18 14 15 16 9 13 ...
#>  $ sofa_first        : int [1:1776] 9 5 5 7 7 5 6 8 5 9 ...
#>  $ service_unit      : chr [1:1776] "SICU" "MICU" "MICU" "SICU" ...
#>  $ service_num       : int [1:1776] 1 0 0 1 1 1 1 0 0 0 ...
#>  $ day_icu_intime    : chr [1:1776] "Friday   " "Saturday " "Friday   " "Saturday " ...
#>  $ day_icu_intime_num: int [1:1776] 6 7 6 7 7 1 7 5 6 6 ...
#>  $ hour_icu_intime   : int [1:1776] 6 17 3 4 7 12 22 14 21 10 ...
#>  $ hosp_exp_flg      : int [1:1776] 1 0 0 1 0 0 0 0 0 1 ...
#>  $ icu_exp_flg       : int [1:1776] 0 0 0 1 0 0 0 0 0 1 ...
#>  $ day_28_flg        : int [1:1776] 1 0 0 1 0 0 0 0 0 1 ...
#>  $ mort_day_censored : num [1:1776] 11.9 731 731 0 731 ...
#>  $ censor_flg        : int [1:1776] 0 1 1 0 1 1 1 0 1 0 ...
#>  $ sepsis_flg        : int [1:1776] 0 0 0 0 0 0 0 0 0 0 ...
#>  $ chf_flg           : int [1:1776] 0 0 0 0 0 0 0 1 0 0 ...
#>  $ afib_flg          : int [1:1776] 0 0 0 0 0 0 0 1 0 0 ...
#>  $ renal_flg         : int [1:1776] 0 0 0 0 0 0 0 0 0 0 ...
#>  $ liver_flg         : int [1:1776] 0 0 0 0 0 0 0 0 0 0 ...
#>  $ copd_flg          : int [1:1776] 0 0 0 0 0 0 0 0 0 0 ...
#>  $ cad_flg           : int [1:1776] 0 0 0 0 0 0 0 0 0 0 ...
#>  $ stroke_flg        : int [1:1776] 0 0 0 0 0 0 0 0 0 0 ...
#>  $ mal_flg           : int [1:1776] 1 0 0 1 0 0 0 1 0 0 ...
#>  $ resp_flg          : int [1:1776] 0 0 0 0 0 0 1 1 1 0 ...
#>  $ map_1st           : num [1:1776] 92 86.7 69.7 101 105 ...
#>  $ hr_1st            : int [1:1776] 86 85 135 125 107 90 94 105 85 114 ...
#>  $ temp_1st          : num [1:1776] 95.9 97.6 96.3 100.1 96.3 ...
#>  $ spo2_1st          : int [1:1776] 100 100 99 100 100 100 100 100 100 93 ...
#>  $ abg_count         : int [1:1776] 22 1 3 4 9 0 18 40 3 11 ...
#>  $ wbc_first         : num [1:1776] 8.1 NA 27 7.1 4.8 12.1 21.6 19.9 11.1 7.7 ...
#>  $ hgb_first         : num [1:1776] 14.1 NA 13.1 12.6 10.7 14.4 13.4 7.8 11.4 13.9 ...
#>  $ platelet_first    : int [1:1776] 354 NA 295 262 22 182 130 20 238 137 ...
#>  $ sodium_first      : int [1:1776] 138 NA 144 139 146 145 143 140 143 143 ...
#>  $ potassium_first   : num [1:1776] 4.6 NA 3.9 4.2 3.4 3.6 3.8 3.7 4 3.7 ...
#>  $ tco2_first        : num [1:1776] 15 NA 17 31 19 26 32 20 25 28 ...
#>  $ chloride_first    : int [1:1776] 109 NA 101 100 110 110 104 105 107 104 ...
#>  $ bun_first         : int [1:1776] 41 NA 16 16 10 10 17 30 15 2 ...
#>  $ creatinine_first  : num [1:1776] 1.6 NA 0.8 0.5 1 0.7 1.3 1.2 0.7 0.3 ...
#>  $ po2_first         : int [1:1776] 196 NA 298 146 134 NA 38 57 212 284 ...
#>  $ pco2_first        : int [1:1776] 39 NA 30 23 30 NA 62 28 41 33 ...
#>  $ iv_day_1          : num [1:1776] 2231 600 2087 NA 2358 ...
```

Expecting (`1776` and `46`).

As you will note, many of the `flg` variables listed in the summary output above, are constrained by 0 and 1. This is because they have a binary encoding (usually 1 if present, and 0 if not). Although not necessary in this particular instance, it is sometimes useful to encode these types of variables as factors. In the original version, they use the function `convert.bin.fac` from the `MIMICbook` package. I have opted to change to `tidyverse` and a function defined to test where the column is only 0s and 1s.

The `MIMICbook` package provides some useful functions written for the textbook that we will use throughout some of the workshops. It installs via GitHub - see `setup` chunk above.

```
is01_factor_column <- function(x) {
  v <- unique(x[!is.na(x)])
  ((length(v) == 1) & (v[1] %in% c(0,1))) | ((length(v) == 2) & (v[1] %in% c(0,1)) & (v[2] %in% c(0,1))
}

dat2 <- dat %>%
  mutate(across(where(is01_factor_column), as.factor))

summary(dat2)
#>  aline_flg  icu_los_day     hospital_los_day       age         gender_num
#>  0:792     Min.   : 0.500   Min.   :  1.000   Min.   :15.18   0: 751
#>  1:984     1st Qu.: 1.370   1st Qu.:  3.000   1st Qu.:38.25   1:1025
#>            Median : 2.185   Median :  6.000   Median :53.68
#>            Mean   : 3.346   Mean   :  8.111   Mean   :54.38
#>            3rd Qu.: 4.003   3rd Qu.: 10.000   3rd Qu.:72.76
#>            Max.   :28.240   Max.   :112.000   Max.   :99.11
#>
#>   weight_first          bmi          sapsi_first      sofa_first
#>  Min.   : 30.00   Min.   :12.78   Min.   : 3.00   Min.   : 0.000
#>  1st Qu.: 65.40   1st Qu.:22.62   1st Qu.:11.00   1st Qu.: 4.000
#>  Median : 77.00   Median :26.32   Median :14.00   Median : 6.000
#>  Mean   : 80.08   Mean   :27.83   Mean   :14.14   Mean   : 5.801
#>  3rd Qu.: 90.00   3rd Qu.:30.80   3rd Qu.:17.00   3rd Qu.: 7.000
#>  Max.   :257.60   Max.   :98.80   Max.   :32.00   Max.   :17.000
#>  NA's   :110      NA's   :466     NA's   :85
#>  service_unit       service_num day_icu_intime     day_icu_intime_num
#>  Length:1776        0:794       Length:1776        Min.   :1.000
#>  Class :character   1:982       Class :character   1st Qu.:2.000
#>  Mode  :character               Mode  :character   Median :4.000
#>                                                    Mean   :4.054
#>                                                    3rd Qu.:6.000
#>                                                    Max.   :7.000
#>
#>  hour_icu_intime hosp_exp_flg icu_exp_flg day_28_flg mort_day_censored
#>  Min.   : 0.00   0:1532       0:1606      0:1493     Min.   :   0.0
#>  1st Qu.: 3.00   1: 244       1: 170      1: 283     1st Qu.: 434.3
#>  Median : 9.00                                       Median : 731.0
#>  Mean   :10.59                                       Mean   : 614.3
#>  3rd Qu.:19.00                                       3rd Qu.: 731.0
#>  Max.   :23.00                                       Max.   :3094.1
#>
#>  censor_flg sepsis_flg chf_flg  afib_flg renal_flg liver_flg copd_flg cad_flg
#>  0: 497     0:1776     0:1563   0:1569   0:1716    0:1677    0:1619   0:1653
#>  1:1279                1: 213   1: 207   1:  60    1:  99    1: 157   1: 123
#>
#>
#>
```

```
#>
#>
#>  stroke_flg mal_flg  resp_flg    map_1st           hr_1st
#>  0:1554     0:1520   0:1211   Min.   :  5.00   Min.   : 30.00
#>  1: 222     1: 256   1: 565   1st Qu.: 76.67   1st Qu.: 74.75
#>                               Median : 87.00   Median : 87.00
#>                               Mean   : 88.25   Mean   : 87.91
#>                               3rd Qu.: 99.00   3rd Qu.:100.00
#>                               Max.   :195.00   Max.   :158.00
#>
#>     temp_1st         spo2_1st        abg_count         wbc_first
#>  Min.   : 32.00   Min.   :  4.00   Min.   :  0.000   Min.   :  0.17
#>  1st Qu.: 96.90   1st Qu.: 98.00   1st Qu.:  1.000   1st Qu.:  8.20
#>  Median : 98.10   Median :100.00   Median :  3.000   Median : 11.30
#>  Mean   : 97.79   Mean   : 98.43   Mean   :  5.985   Mean   : 12.32
#>  3rd Qu.: 99.30   3rd Qu.:100.00   3rd Qu.:  7.000   3rd Qu.: 15.00
#>  Max.   :104.80   Max.   :100.00   Max.   :115.000   Max.   :109.80
#>  NA's   :3                                           NA's   :8
#>    hgb_first     platelet_first   sodium_first   potassium_first
#>  Min.   : 2.00   Min.   :  7.0   Min.   :105.0   Min.   :1.900
#>  1st Qu.:11.10   1st Qu.:182.0   1st Qu.:137.0   1st Qu.:3.600
#>  Median :12.70   Median :239.0   Median :140.0   Median :4.000
#>  Mean   :12.55   Mean   :246.1   Mean   :139.6   Mean   :4.108
#>  3rd Qu.:14.12   3rd Qu.:297.0   3rd Qu.:142.0   3rd Qu.:4.400
#>  Max.   :19.00   Max.   :988.0   Max.   :165.0   Max.   :9.800
#>  NA's   :8       NA's   :8       NA's   :5       NA's   :5
#>    tco2_first    chloride_first    bun_first      creatinine_first
#>  Min.   : 2.00   Min.   : 78.0   Min.   :  2.00   Min.   : 0.000
#>  1st Qu.:22.00   1st Qu.:101.0   1st Qu.: 11.00   1st Qu.: 0.700
#>  Median :24.00   Median :104.0   Median : 15.00   Median : 0.900
#>  Mean   :24.42   Mean   :103.8   Mean   : 19.28   Mean   : 1.096
#>  3rd Qu.:27.00   3rd Qu.:107.0   3rd Qu.: 22.00   3rd Qu.: 1.100
#>  Max.   :62.00   Max.   :133.0   Max.   :139.00   Max.   :18.300
#>  NA's   :5       NA's   :5       NA's   :5        NA's   :6
#>    po2_first       pco2_first        iv_day_1
#>  Min.   : 22.0   Min.   :  8.00   Min.   :    0.0
#>  1st Qu.:108.0   1st Qu.: 36.00   1st Qu.:  329.8
#>  Median :195.0   Median : 41.00   Median : 1081.5
#>  Mean   :227.6   Mean   : 43.41   Mean   : 1622.9
#>  3rd Qu.:323.0   3rd Qu.: 47.00   3rd Qu.: 2493.9
#>  Max.   :634.0   Max.   :158.00   Max.   :13910.0
#>  NA's   :186     NA's   :186      NA's   :143
```

As you can now see, instead of means (which under the old encoding equate to proportions of patients where the variable == 1), now we have counts of patients with each *level* of the variable. This is because `R`'s summary function treats factors and numerical values differently.

Often, you will want to report these summaries separately for different groups. For instance, is the mean or median age the same for those who received an IAC, and those who didn't? A multi-purpose function called `tapply` can help us with this.

```
tapply(dat2$age, dat2$aline_flg, summary)
#> $`0`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   15.18   34.80   50.85   53.02   72.11   97.46
#> 
#> $`1`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
#>   15.19   40.36   56.02   55.48   73.21   99.11
```

This function stratifies the first argument (`age`) by the second argument (`aline_flg`) and run the third argument (`summary`) on it. So, in our case, run the `summary` function on `age` for those who received an IAC (`aline_flg` = 1) and those who didn't (`aline_flg` = 0).

Student Question 1:

a) Using the `dat2` data frame, run the summary function for `sofa_first` and `service_unit` separately for those with an IAC, and those without.
b) Run the summary function for `age`, `sofa_first`, and `service_unit` separately for those who died within 28 days, and those who survived.

```
# a)
tapply(dat2$sofa_first, dat2$aline_flg, summary)
#> $`0`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
#>   0.000   4.000   5.000   4.816   6.000  14.000 
#> 
#> $`1`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
#>   0.000   5.000   6.000   6.595   8.000  17.000
tapply(dat2$service_unit, dat2$aline_flg, summary)
#> $`0`
#>    Length     Class      Mode 
#>       792 character character 
#> 
#> $`1`
#>    Length     Class      Mode 
#>       984 character character

# b)
tapply(dat2$age, dat2$day_28_flg, summary)
#> $`0`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
#>   15.18   34.83   49.46   50.78   66.62   99.11 
#> 
#> $`1`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
#>   22.06   65.32   77.83   73.35   83.83   97.46
tapply(dat2$sofa_first, dat2$day_28_flg, summary)
#> $`0`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
#>   0.000   4.000   5.000   5.661   7.000  16.000 
#> 
#> $`1`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
#>   0.000   5.000   6.000   6.541   8.000  17.000
```

```
tapply(dat2$service_unit, dat2$day_28_flg, summary)
#> $`0`
#>    Length     Class      Mode 
#>      1493 character character 
#> 
#> $`1`
#>    Length     Class      Mode 
#>       283 character character
```

Producing a Table One and Other Tables

The output from summary is very useful, but is generally not acceptable for formal research reports, let alone a published paper. There are several ways to produce a publication which has a better layout. One way is described in the textbook (Chapter 15). Another, which we will cover here, is through an `R` package called `tableone`.

As some of you may know, "Table 1" often refers to the table presented in most medical manuscripts which contains information used to describe the cohort. This typically includes information such as average patient age, gender distribution, and other important demographic, clinical, and socioeconomic characteristics. We will cover briefly how to use the `CreateTableOne` function in this package to generate a table which is closer to being publication worthy.

Here is an example functional call to `CreateTableOne`, which computes either the mean and standard deviation for numeric variables, or count and percentage for factors. You specify which variables you want to include in the table

```
CreateTableOne(vars=c("age","service_unit","aline_flg","day_28_flg"),data=dat2)
#>                      
#>                       Overall      
#>   n                    1776        
#>   age (mean (SD))     54.38 (21.06)
#>   service_unit (%)                 
#>      FICU                62 ( 3.5) 
#>      MICU               732 (41.2) 
#>      SICU               982 (55.3) 
#>   aline_flg = 1 (%)     984 (55.4) 
#>   day_28_flg = 1 (%)    283 (15.9)
```

We may want to breakdown these summaries further, like we did above with `tapply`, but we can do it with one function with the `CreateTableOne` function by passing the `strata` parameter. `strata` specifies which variable to stratify (breakdown) the others by. For example, here is the same table in the previous chunk, broken down by whether a patient received an IAC or not.

```
CreateTableOne(
  vars =c ("age", "service_unit", "aline_flg", "day_28_flg"),
  strata = "aline_flg",
  data = dat2,
  test = FALSE
  )
#>                      Stratified by aline_flg
#>                       0             1            
#>   n                     792           984        
#>   age (mean (SD))     53.02 (21.67) 55.48 (20.51)
#>   service_unit (%)
#>      FICU               24 ( 3.0)     38 (  3.9)
#>      MICU              480 (60.6)    252 ( 25.6)
#>      SICU              288 (36.4)    694 ( 70.5)
#>   aline_flg = 1 (%)      0 ( 0.0)    984 (100.0)
#>   day_28_flg = 1 (%)   113 (14.3)    170 ( 17.3)
```

Student Question 2:

a) Compute a Table to summarize those variable considered before (`age`, `service_unit`, `aline_flg` and `day_28_flg`) in addition to `gender_num` and `chf_flg`, but now stratify by survival at 28 days (`day_28_flg`).
b) Repeat part a), but now use the `dat` data frame instead of `dat2`. Note the differences in how variables that were previously recast as factors are summarized.

```
# a)
CreateTableOne(
  vars =c ("age", "service_unit", "aline_flg", "day_28_flg", "gender_num", "chf_flg"),
  strata = "day_28_flg",
  data = dat2,
  test = FALSE
  )
#>                       Stratified by day_28_flg
#>                        0             1
#>   n                     1493           283
#>   age (mean (SD))      50.78 (20.06) 73.35 (15.32)
#>   service_unit (%)
#>      FICU               59 ( 4.0)      3 (  1.1)
#>      MICU              605 (40.5)    127 ( 44.9)
#>      SICU              829 (55.5)    153 ( 54.1)
#>   aline_flg = 1 (%)    814 (54.5)    170 ( 60.1)
#>   day_28_flg = 1 (%)     0 ( 0.0)    283 (100.0)
#>   gender_num = 1 (%)   886 (59.3)    139 ( 49.1)
#>   chf_flg = 1 (%)      145 ( 9.7)     68 ( 24.0)

# b)
CreateTableOne(
  vars =c ("age", "service_unit", "aline_flg", "day_28_flg", "gender_num", "chf_flg"),
  strata = "day_28_flg",
  data = dat,
  test = FALSE
  )
#>                           Stratified by day_28_flg
#>                            0             1
#>   n                         1493           283
#>   age (mean (SD))          50.78 (20.06) 73.35 (15.32)
#>   service_unit (%)
#>      FICU                   59 ( 4.0)      3 ( 1.1)
#>      MICU                  605 (40.5)    127 (44.9)
#>      SICU                  829 (55.5)    153 (54.1)
#>   aline_flg (mean (SD))   0.55 (0.50)   0.60 (0.49)
#>   day_28_flg (mean (SD))  0.00 (0.00)   1.00 (0.00)
#>   gender_num (mean (SD))  0.59 (0.49)   0.49 (0.50)
#>   chf_flg (mean (SD))     0.10 (0.30)   0.24 (0.43)
```

Optional:

> As an aside, the following code may help for your projects, as it improves the presentation of the tables above. You will still need to update the column and row names manually, but this should paste nicely into Word or LateX!

```
CreateTableOne(
  vars = c ("age", "service_unit", "aline_flg", "day_28_flg"),
  strata = "aline_flg",
  data = dat2,
  test = FALSE
) %>%
  print(printToggle      = FALSE,
        showAllLevels    = TRUE,
        cramVars         = "kon") %>%
  {
    data.frame(
      variable_name    = gsub(" ", "&nbsp;", rownames(.), fixed = TRUE),
      .,
      row.names        = NULL,
      check.names      = FALSE,
      stringsAsFactors = FALSE
    )
  } %>%
  knitr::kable()
```

variable_name	level	0	1
n		792	984
age (mean (SD))		53.02 (21.67)	55.48 (20.51)
service_unit (%)	FICU	24 (3.0)	38 (3.9)
	MICU	480 (60.6)	252 (25.6)
	SICU	288 (36.4)	694 (70.5)
aline_flg (%)	0	792 (100.0)	0 (0.0)
	1	0 (0.0)	984 (100.0)
day_28_flg (%)	0	679 (85.7)	814 (82.7)
	1	113 (14.3)	170 (17.3)

Other Bivariate Numerical Summaries

Sometimes you may wish to display the relationships between two or more variables directly. For categorical variables this can be tricky. One common way to explore relationships between categorical variables is by producing the cross tabulated tables (“crosstabs” for short). This is mainly done via the `table` function, which can take several categorical variables, and produce the number of patients which meet criteria for those variables. For instance, looking at how an IAC was used in men and women:

```
table(dat2$gender_num, dat2$aline_flg, dnn = c("Gender", "IAC"))
#>       IAC
#> Gender   0   1
#>      0 345 406
#>      1 447 578
```

We can see that an IAC was used 578 times in men (`gender_num == 1`) and 447 times in women (`gender_num == 0`). The raw numbers are often difficult to compare, so often the proportions are more useful. Applying `prop.table` to our existing table, and adding the argument 1 (for by row, use 2 for columns), we get the proportion of men and women who had an IAC (56% vs. 54%).

```
prop.table(table(dat2$gender_num, dat2$aline_flg, dnn = c("Gender", "IAC")), 1)
#>       IAC
#> Gender         0         1
#>      0 0.4593875 0.5406125
#>      1 0.4360976 0.5639024
```

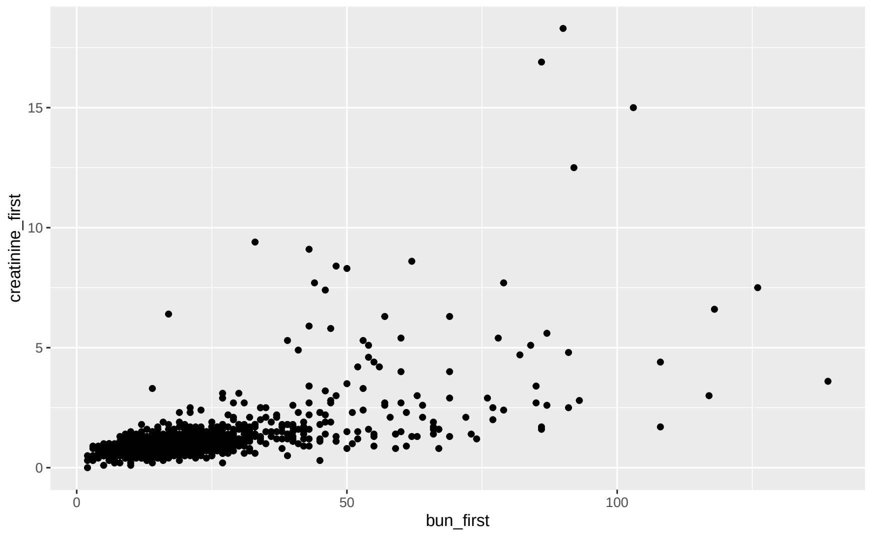
A different summary for bivariate numeric data exists to present the strength of the relationship between two variables, called the correlation coefficient. There is a `cor` function in `R`, but when dealing with only two variables, it's easiest to use the `cor.test` function. Under the defaults, it computes the Pearson product-moment correlation and computes a hypothesis test to assess if there's evidence that the correlation is not zero. Other forms of correlation are computed below, including Spearman's rho and Kendall's tau. These latter methods are useful when dealing with data which is not necessarily numeric but ordered (e.g., likert based rankings on a 1-5 scale) or has outliers. Spearman's rho and Kendall's tau are rank based methods, and also have a certain degree of robustness to outliers in the data. None of these methods are robust to non-linear relationships, and it's very easy to miss a strong relationship between two variables if you rely on these methods in isolation.

```
cor.test(dat2$bun_first,dat2$creatinine_first)
#> 
#>  Pearson's product-moment correlation
#> 
#> data:  dat2$bun_first and dat2$creatinine_first
#> t = 33.233, df = 1768, p-value < 2.2e-16
#> alternative hypothesis: true correlation is not equal to 0
#> 95 percent confidence interval:
#>  0.5905441 0.6479479
#> sample estimates:
#>       cor
#> 0.6200752
cor.test(dat2$bun_first,dat2$creatinine_first,method="spearman")
#> 
#>  Spearman's rank correlation rho
#> 
#> data:  dat2$bun_first and dat2$creatinine_first
#> S = 415893613, p-value < 2.2e-16
#> alternative hypothesis: true rho is not equal to 0
#> sample estimates:
#>       rho
#> 0.5499986
cor.test(dat2$bun_first,dat2$creatinine_first,method="kendall")
#> 
#>  Kendall's rank correlation tau
```

```
#>
#> data:  dat2$bun_first and dat2$creatinine_first
#> z = 24.807, p-value < 2.2e-16
#> alternative hypothesis: true tau is not equal to 0
#> sample estimates:
#>      tau
#> 0.418814
```

We can produce a scatterplot of the same two variables:

```
dat2 %>%
  ggplot(aes(x = bun_first, y = creatinine_first)) +
  geom_point()
```

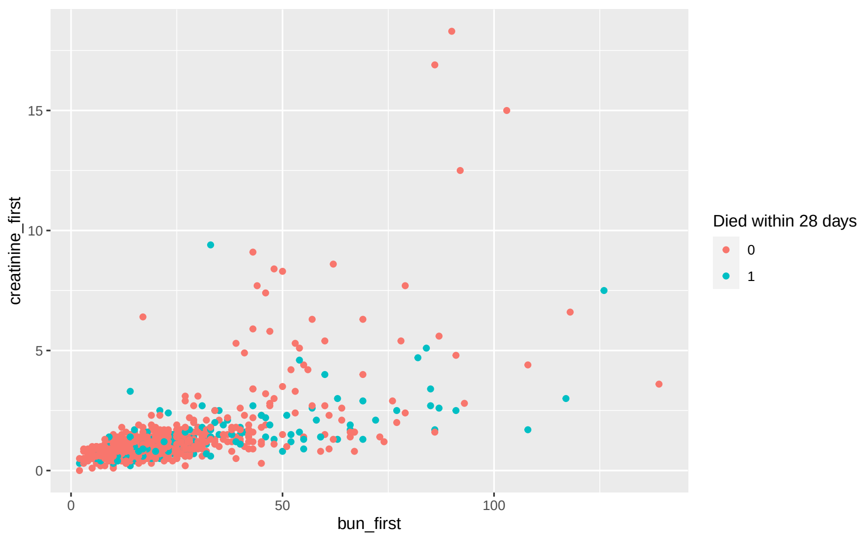


We can see that there is indeed a positive correlation between the two variables, but the data has more variability for higher values of `bun_first` and `creatinine_first`. It's advisable to consider transformations of these two variables and be wary about using Pearson's correlation.

Going beyond two dimensions can be a little tricky. Plotting on a three dimensional axis, while possible, is not ideal, and very few people can see in four dimensions.

What is possible, is to use other aspects of the plot (e.g., size, color, shape, hue, transparency, location) to identify features you would like to see. For instance, in the above plot, we can add color to identify those who died:

```
dat2 %>%
  ggplot(aes(x = bun_first, y = creatinine_first)) +
  geom_point(aes(color = day_28_flg)) +
  labs(color = "Died within 28 days")
```



Creating Categorical Variables from Continuous/Numeric Variables

Sometimes numeric variables need to be broken down into categorical variables or factors. This can be done for a variety of reasons. There is a useful function called `cut2` in the `Hmisc` package. W

```
dat2$age.cat <- cut2(dat2$age,g=5)
table(dat2$age.cat)
#>
#> [15.2,33.6) [33.6,47.6) [47.6,60.3) [60.3,76.4) [76.4,99.1]
#>         356         355         355         355         355

dat2$age.cat2 <- cut2(dat2$age,c(25,40,55,70,85))
table(dat2$age.cat2)
#>
#> [15.2,25.0) [25.0,40.0) [40.0,55.0) [55.0,70.0) [70.0,85.0) [85.0,99.1]
#>         208         287         428         341         382         130
```

`cut2` typically needs two arguments. The first is a numeric variable to convert into a factor, and the second is how to do the splitting. Specifying `g=5` (as above for `age.cat`) breaks the numeric variable into 5 groups, with the cut points determined by attempting to make the groups as equally sized as possible. As you can see in this example, due to the odd number of patients, they are not perfectly even. The second approach requires passing the cut points. In the second example, we tell `R` to cut the data at 25, 40,. . . . This results in 6 groups for five cut points.

Student Question 3:

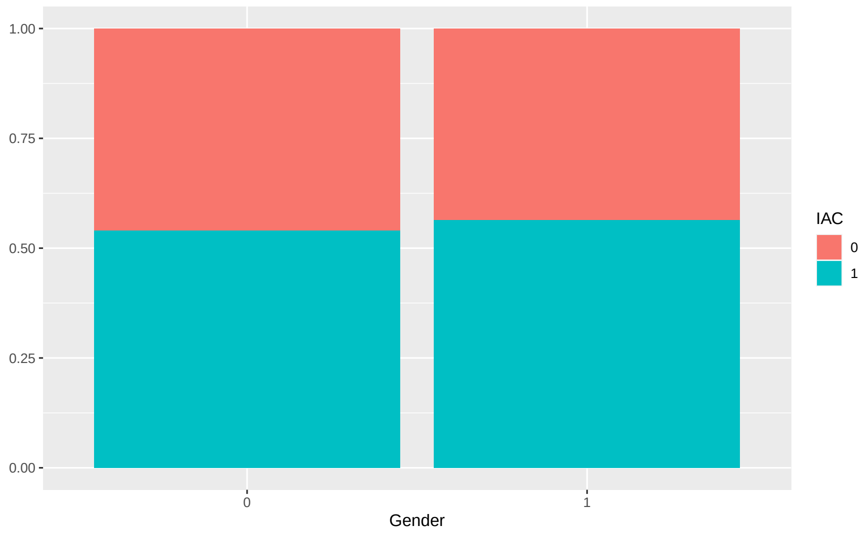
a) Create a new variable in the `dat2` data frame called `sofa.cat` made up of four (approximately) equally sized groups for SOFA. Print the sample size in each group. Consider why the group sizes may differ significantly from each other.
b) For each SOFA group calculate the number of people who survived and died in the hospital and at 28 days (use the `hosp_exp_flg` and `day_28_flg` variable). Does the mortality increase or decrease as SOFA increases?

```
# a)
dat2$sofa.cat <- cut2(dat2$sofa_first,g=4)
table(dat2$sofa.cat)
#>
#> [0, 5) [5, 7)      7 [8,17]
#>    529    640    243    364
dat2 %>%
  group_by(sofa_first) %>%
  count()
#> # A tibble: 17 x 2
#> # Groups:   sofa_first [17]
#>    sofa_first     n
#>         <int> <int>
#>  1          0    17
#>  2          1    31
#>  3          2    35
#>  4          3   158
#>  5          4   288
#>  6          5   346
#>  7          6   294
#>  8          7   243
#>  9          8   148
#> 10          9    94
#> 11         10    56
#> 12         11    33
#> 13         12    13
#> 14         13    13
#> 15         14     5
#> 16         16     1
#> 17         17     1
# Groupings have to be contiguous and several neighboring numbers have very high frequencies

# b)
prop.table(table(dat2$sofa.cat, dat2$hosp_exp_flg, dnn = c("SOFA Grp", "Died in Hosp")),1)
#>         Died in Hosp
#> SOFA Grp          0          1
#>   [0, 5) 0.93950851 0.06049149
#>   [5, 7) 0.84218750 0.15781250
#>   7      0.83127572 0.16872428
#>   [8,17] 0.80769231 0.19230769
prop.table(table(dat2$sofa.cat, dat2$day_28_flg, dnn = c("SOFA Grp", "Died w/i 28 days")),1)
#>         Died w/i 28 days
#> SOFA Grp          0          1
#>   [0, 5) 0.92060491 0.07939509
#>   [5, 7) 0.82187500 0.17812500
#>   7      0.80246914 0.19753086
#>   [8,17] 0.78296703 0.21703297
```

Plotting relationships with discrete variables

Plotting discrete data can be a little tricky, but if done right can be very effective. For an example of why it's difficult, let's plot two discrete variables: `gender_num` and `aline_flg`.

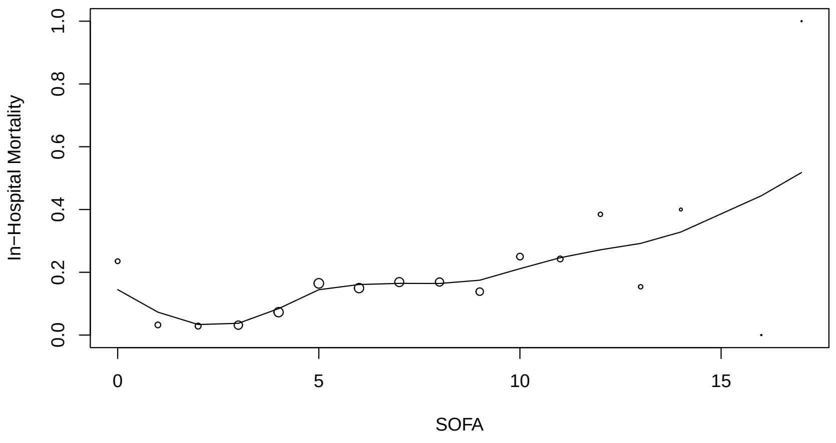
```
#plot(dat2$gender_num,dat2$aline_flg,xlab="Gender",ylab="IAC")

dat2 %>%
  ggplot(aes(x = gender_num, fill = aline_flg)) +
  geom_bar(position = "fill") +
  labs(x = "Gender", y = NULL, fill = "IAC")
```



Because we have converted `gender_num` and `aline_flg` to a factor, `R` gives us what is called a "Factor Plot". The area of the light grey region is proportional to the proportion of each gender who received an IAC. In this case, there is not that big of a difference between the genders.

Sometimes the covariate may take on more than two levels. Here, we plot the in-hospital mortality rate by the different SOFA values, and put a smooth curve through the points. This covers a more *advanced* topic, and we don't expect you to understand the technical details of the code below.

```
plot(names(table(dat2$sofa_first)),sapply(split(dat2,dat2$sofa_first),function(x) { mean(x$hosp_exp_flg
lines(smooth.spline(dat2$sofa_first,dat2$hosp_exp_flg==1),type="l")
```

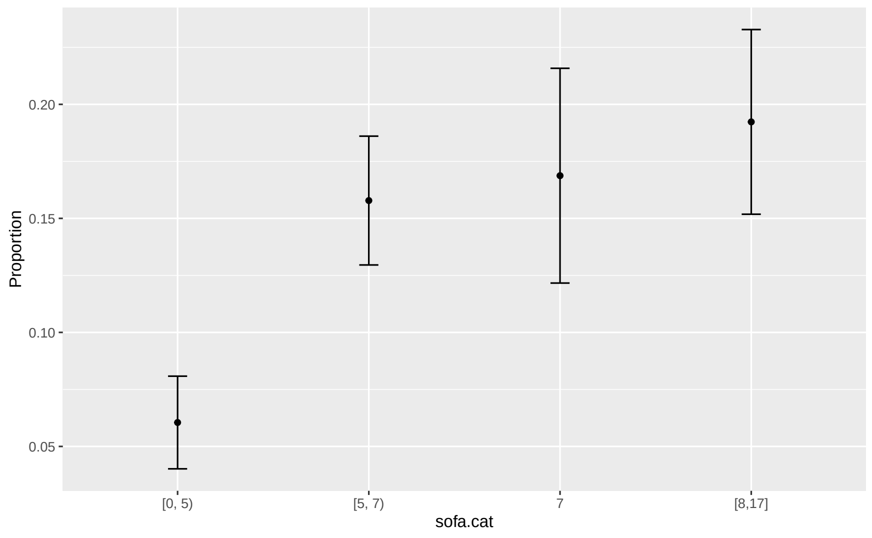


SOFA is a validated disease severity scale for the ICU, and generally correlates strongly with mortality. Here, while the mortality rate generally increases as SOFA increases, the smooth fit isn't necessarily non-decreasing as SOFA values increase. We have added points roughly proportional to the sample size of each SOFA level, and you'll see towards the high levels of SOFA, very few patients are observed, with the second highest score (16) having a 100% *survival* rate (but with only *one* patient).

For binary outcomes, it is often useful to plot the proportion of patients with the outcome (e.g., mortality rate) by the different levels of a covariate of interest. Because sample size plays such an important role in the uncertainty associated with these estimate proportions, it seems appropriate to include an estimate of our uncertainty via a confidence interval.

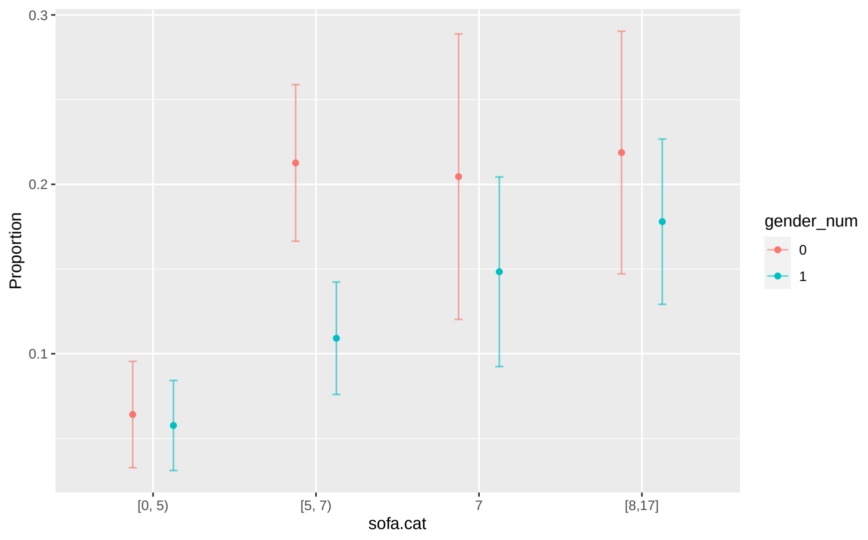
In the `MIMICbook` package you installed above, there is a `plot_prop_by_level` which can plot the proportion of patients with an outcome by one or two factor variables. For instance, if we wished to plot the in hospital mortality rate by the SOFA categories (`sofa.cat`) we defined above, we can using:

```
plot_prop_by_level(dat2,"sofa.cat","hosp_exp_flg")
```



Often it's useful to consider more than one covariate at a time to assess confounding and effect modification. Here, if we wished to examine `sofa.cat` and `gender_num` at the same time, we add `factor.var2="gender_num"` to our previous use of `plot_prop_by_level`.

```
plot_prop_by_level(dat2,"sofa.cat","hosp_exp_flg",factor.var2="gender_num")
```

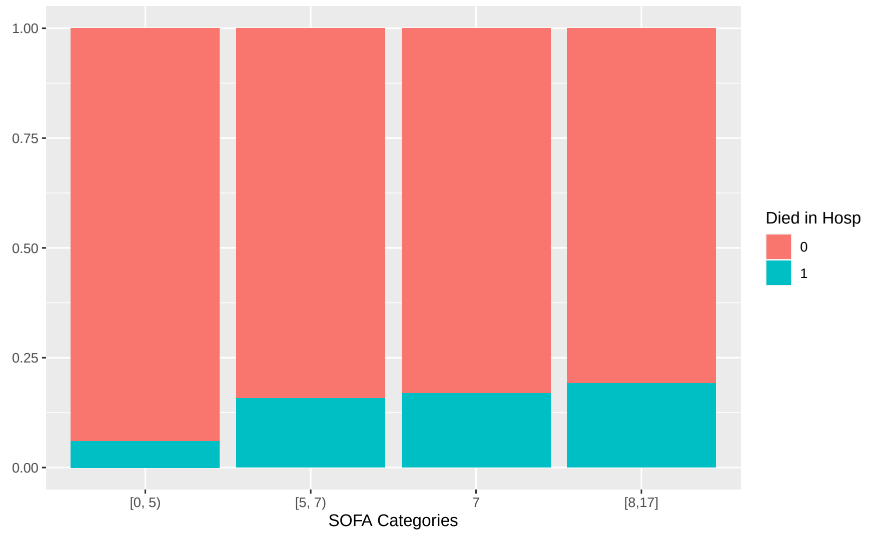


Here we see that in hospital mortality is higher in women for the SOFA groups we have considered, suggesting that it might be an important confounder for this outcome and variable.

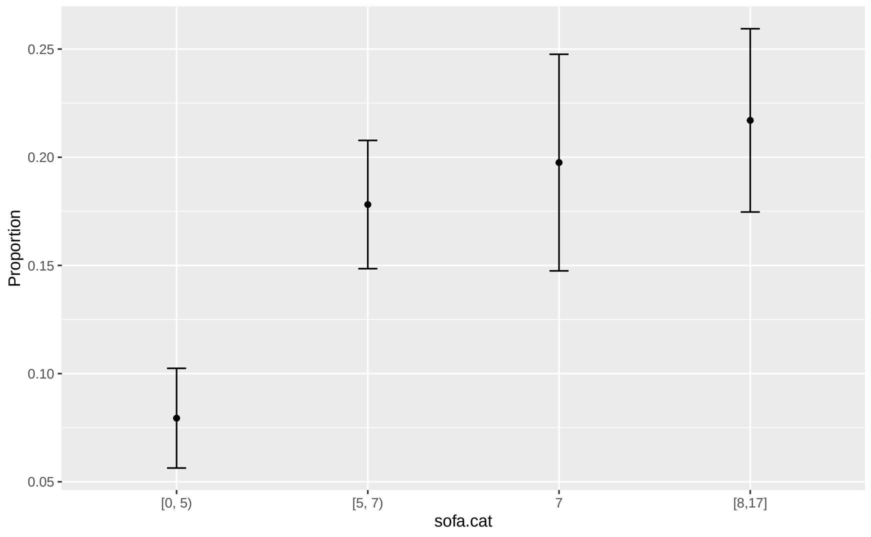
Student Question 4:

a) Make a factor plot of the categories of SOFA we created (`sofa.cat`) and hospital mortality (`hosp_exp_flg`). Does the trend align with your expectations based on the non-graphical EDA performer earlier?
b) Use `plot_prop_by_level` using `sofa.cat` as the covariate of interest and 28 day mortality (`day_28_flg`) as the outcome.
c) Include the main covariate of interest for this study `aline_flg` as the second factor variable and extend part b).
d) Repeat part c), but swap the IAC and SOFA arguments. Consider how the different depictions of the underlying data could better support different objectives.
e) Create a new variable, `sofa.cat2`, with cut points at 3, 6, 9, 12. Repeat parts b) and c).
f) Make a plot of the 28 day mortality outcome, `aline_flg` and `chf_flg`. Ignoring the statistical significance (i.e., do not perform any formal testing), consider why this plot may suggest the complexity of any potential effect of an IAC on mortality.

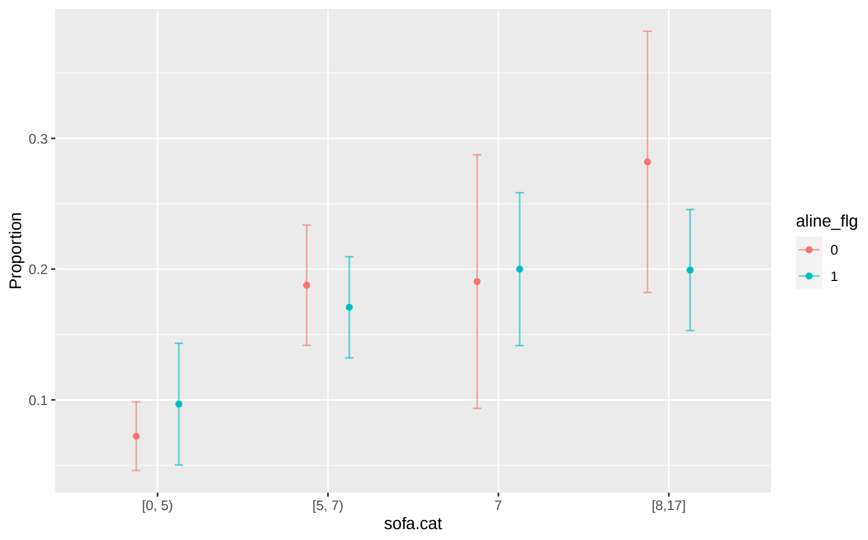
```
# a)
dat2 %>%
  ggplot(aes(x = sofa.cat, fill = hosp_exp_flg)) +
  geom_bar(position = "fill") +
  labs(x = "SOFA Categories", y = NULL, fill = "Died in Hosp")
```



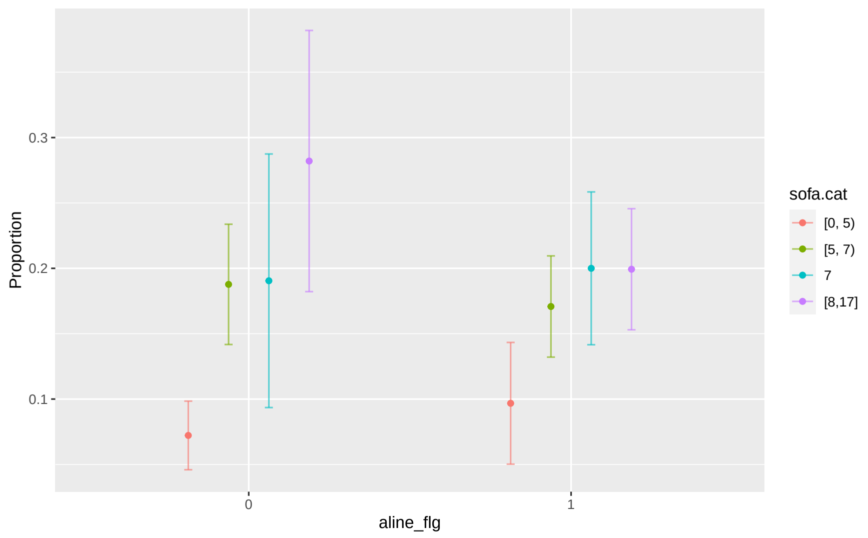
```
# Yes, percentage of patients in groups as they get higher are increasing

# b)
plot_prop_by_level(dat2,"sofa.cat","day_28_flg")
```



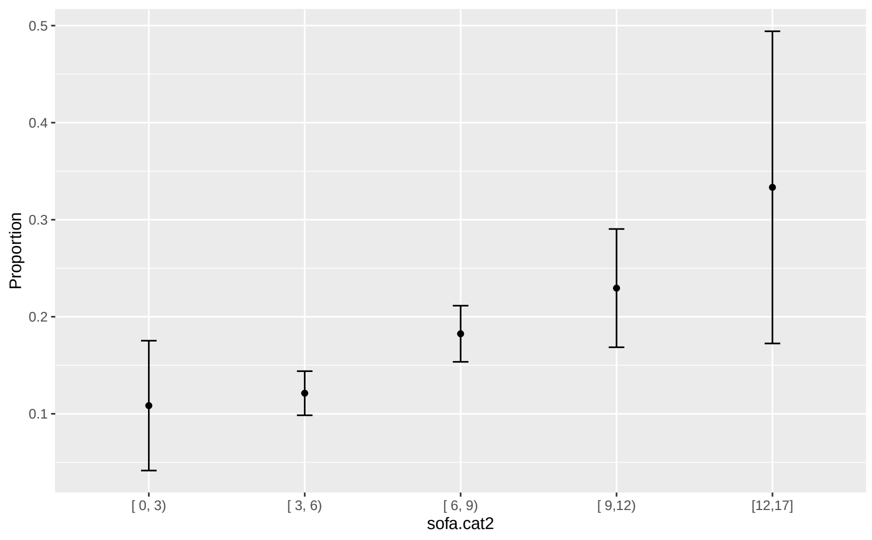
```
# c)
plot_prop_by_level(dat2,"sofa.cat","day_28_flg",factor.var2="aline_flg")
```



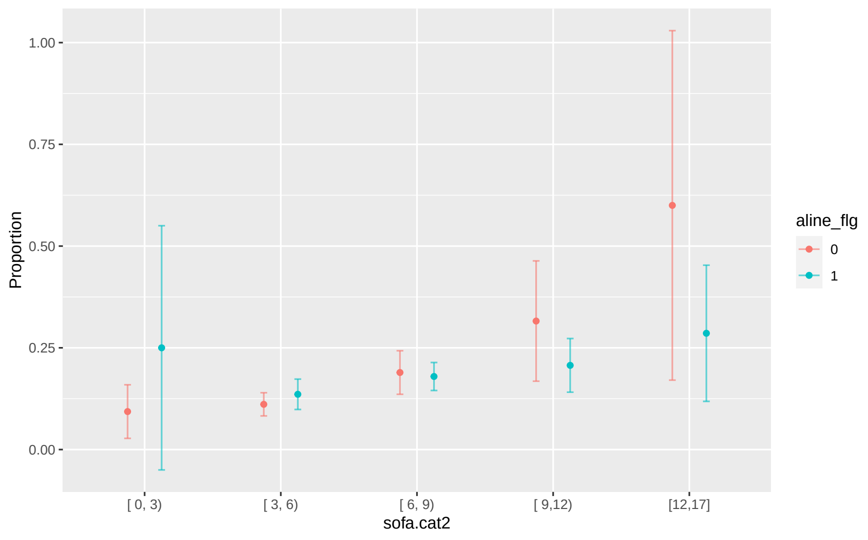
```
# d)
plot_prop_by_level(dat2,"aline_flg","day_28_flg",factor.var2="sofa.cat")
```



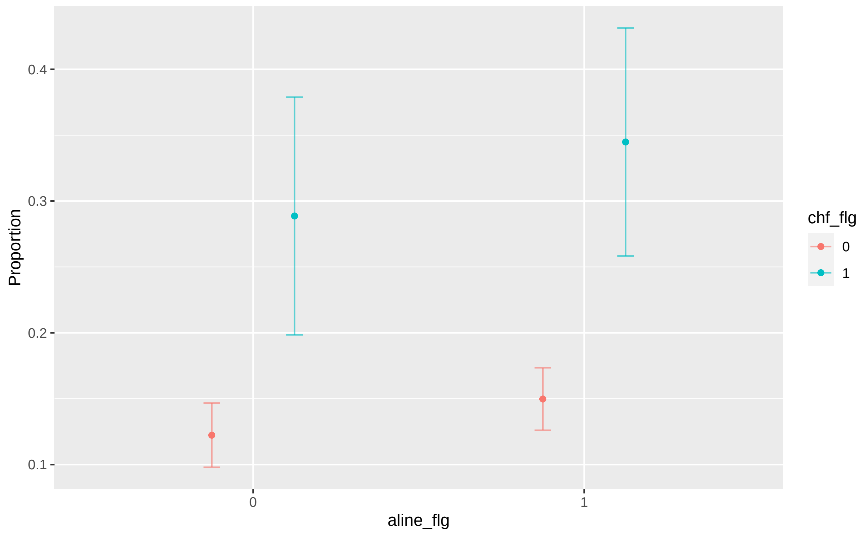
```
# e)
dat2$sofa.cat2 <- cut2(dat2$sofa_first,c(3, 6, 9, 12))
plot_prop_by_level(dat2,"sofa.cat2","day_28_flg")
```



```r
plot_prop_by_level(dat2,"sofa.cat2","day_28_flg",factor.var2="aline_flg")
```



```r
# f)
plot_prop_by_level(dat2,"aline_flg","day_28_flg",factor.var2="chf_flg")
```



Odds ratios

Note: For those with a programming background, `R` *indexes vectors starting from 1.*

As previously discussed, odds ratios are very commonly used to communicate relative effect sizes for binary outcomes, particularly in observational data. Calculation is straightforward, but often misunderstood. We start with a 2 x 2 table. Below is the 2 x 2 table for in hospital mortality and having an arterial line. I've assigned it to a new variable called `egtab`.

```
egtab <- table(dat2$aline_flg,dat2$hosp_exp_flg,dnn=c("IAC","Hosp. Mort"))
egtab
#>    Hosp. Mort
#> IAC   0   1
#>   0 702  90
#>   1 830 154
```

It's hard to interpret the raw counts, so we'll use `prop.table` to compute the proportions who died and lived by row (margin 1, IAC).

```
pegtab <- prop.table(egtab,1)
pegtab
#>    Hosp. Mort
#> IAC         0         1
#>   0 0.8863636 0.1136364
#>   1 0.8434959 0.1565041
```

Odds are $\frac{p}{1-p}$ where p is the proportion with the outcome (death) in a group of patients, which is in the second column. We can index the above table by column (`tab[,idx]` will retrieve column `idx` from the table [or matrix] `tab`) to compute the odds in each group.

```
Oddsegtab <-pegtab[,2]/pegtab[,1]
Oddsegtab
#>         0         1
#> 0.1282051 0.1855422
```

Now we have the odds of the outcome in those who got an IAC `1` and those who didn't `0`. We need to pick a reference group. We'll calculate it both ways, but let's assume we want those without an IAC to be the reference:

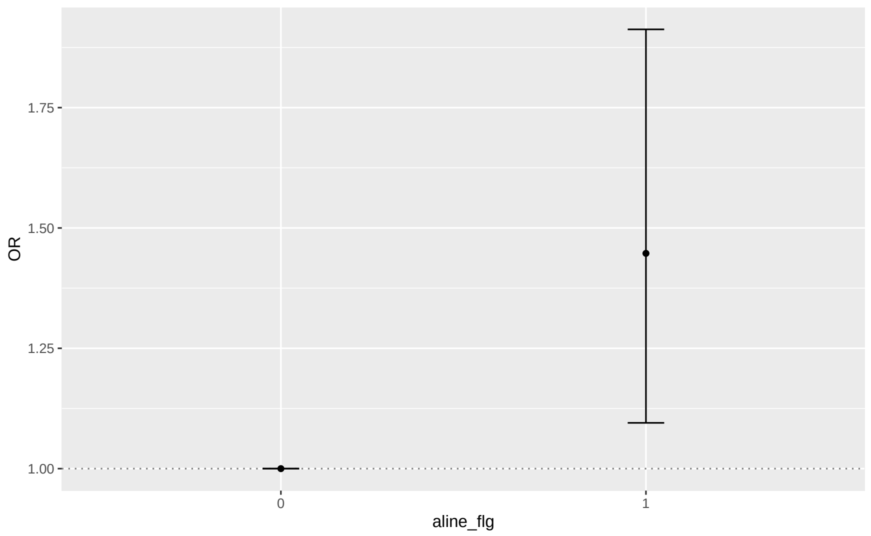
```
Oddsegtab[2]/Oddsegtab[1]
#>        1
#> 1.447229
```

If we wanted those with an IAC to be the reference group:

```
Oddsegtab[1]/Oddsegtab[2]
#>         0
#> 0.6909757
```

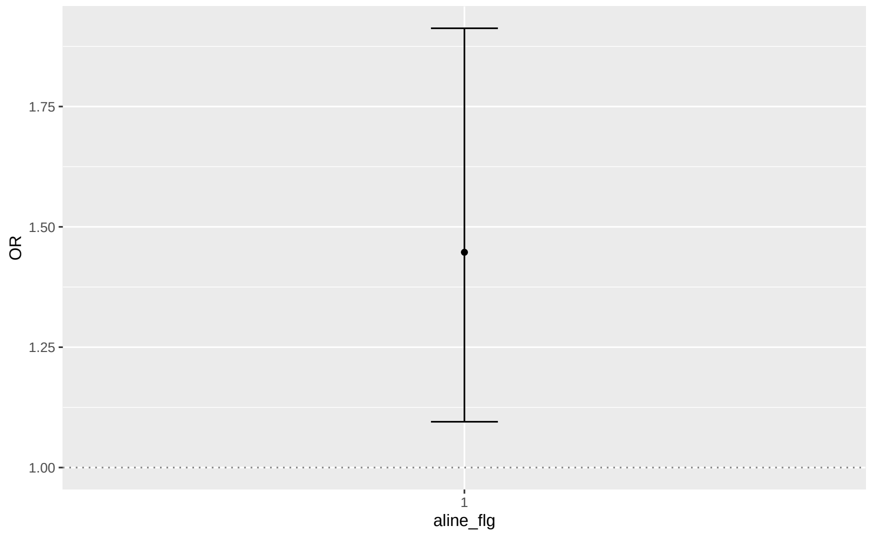
If we wanted to plot this information, and include a confidence interval, we can use the `plot_OR_by_level` from the `MIMICbook` package:

```
plot_OR_by_level(dat2,"aline_flg","hosp_exp_flg")
```



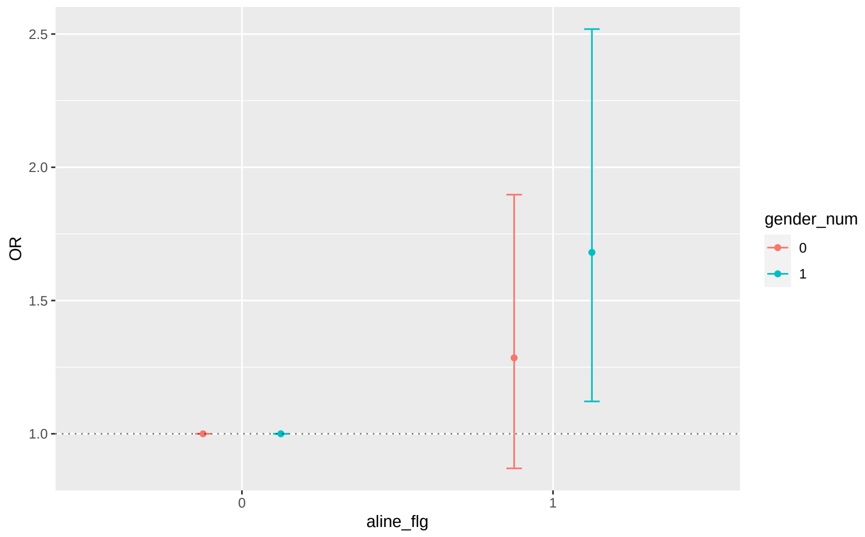
This by default includes an odds ratio of 1 indicating the reference group. To remove this point use the `include.ref.group.effect` argument:

```
plot_OR_by_level(dat2,"aline_flg","hosp_exp_flg",include.ref.group.effect = FALSE)
```



You can also look at more than one covariate at a time. For instance, looking at `aline_flg` and the `gender_num` variable:

```
plot_OR_by_level(dat2,"gender_num","hosp_exp_flg",factor.var2="aline_flg",include.ref.group.effect = TRU
```



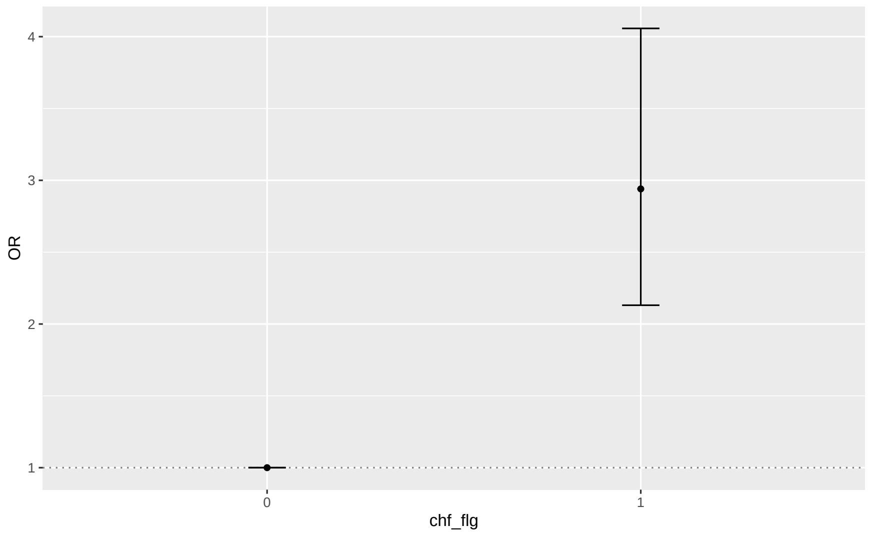
Here we have computed the odds ratio for an IAC (vs no IAC) separately for men and women.

Student Question 5:

a) Create a 2 x 2 table with `chf_flg` and the variable `day_28_flg` outcome and assign it to a variable called `tab22`.
b) Compute the odds ratio for having CHF vs. not having CHF using this table.
c) Construct a plot of the odds ratios and 95% confidence intervals using the `plot_OR_by_level` function for CHF and 28 day mortality.
d) Create a 4 x 2 table with the `sofa.cat` variable and the `day_28_flg` outcome and assign it to a variable called `tab42`.

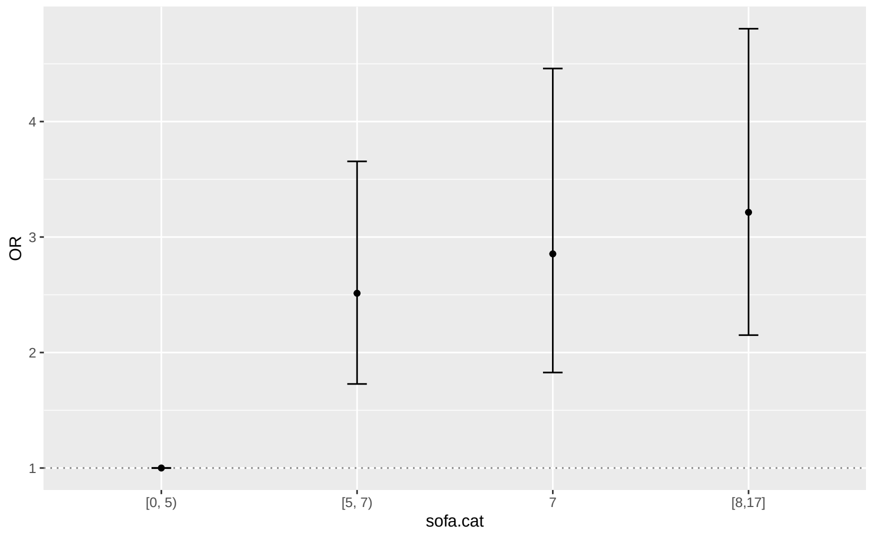
e) Pick and define a reference group for `sofa.cat`, and compute the odds ratio(s) for the other levels of `sofa.cat` using `day_28_flg` as your outcome.
f) Construct a plot of the odds ratios and 95% confidence intervals using the `plot_OR_by_level` function for the SOFA categories and 28 day mortality. Make sure the reference groups are the same as parts d) and e). Look into the `relevel` function in `R` or the `ref.group` argument in the `plot_OR_by_level` function.
g) Construct a plot looking at the 28 day mortality outcome, and the two variables we considered here, `sofa.cat` and `chf_flg`. Exchange the variables assigned to the factor.var1 and factor.var2 arguments, and consider briefly two reasons why you might prefer one plot over the other, and what you would conclude from your chosen plot.

```r
# a)
tab22 <- table(dat2$chf_flg,dat2$day_28_flg,dnn=c("CHF","28 Day Mort"))

# b)
ptab22 <- prop.table(tab22,1)
Oddstab22 <-ptab22[,2]/ptab22[,1]
Oddstab22[2]/Oddstab22[1]
#>        1
#> 2.940305

# c)
plot_OR_by_level(dat2,"chf_flg","day_28_flg")
```

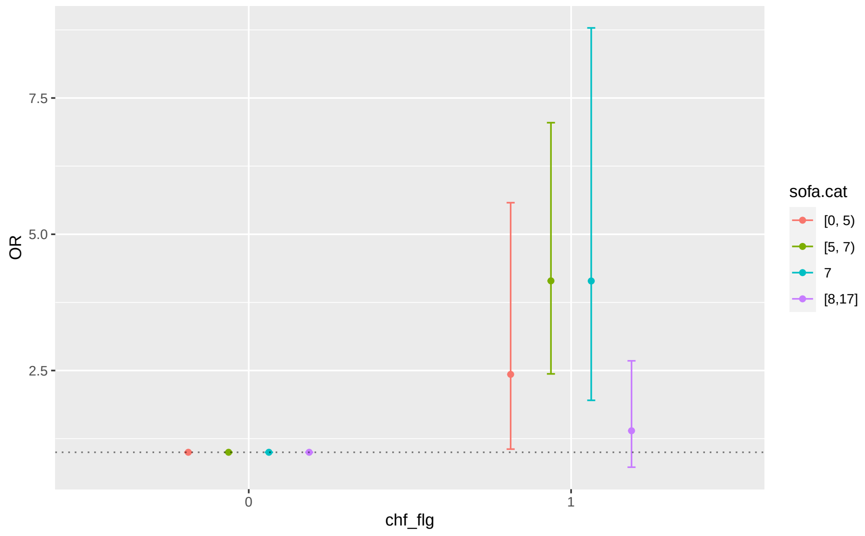


```r
# d)
tab42 <- table(dat2$sofa.cat,dat2$day_28_flg,dnn=c("SOFA Cat.","28 Day Mort"))

# e)
ptab42 <- prop.table(tab42,1)
Oddstab42 <-ptab42[,2]/ptab42[,1]
Oddstab42[2:4]/Oddstab42[1]
#>   [5, 7)        7   [8,17]
#> 2.513036 2.854212 3.214119
```

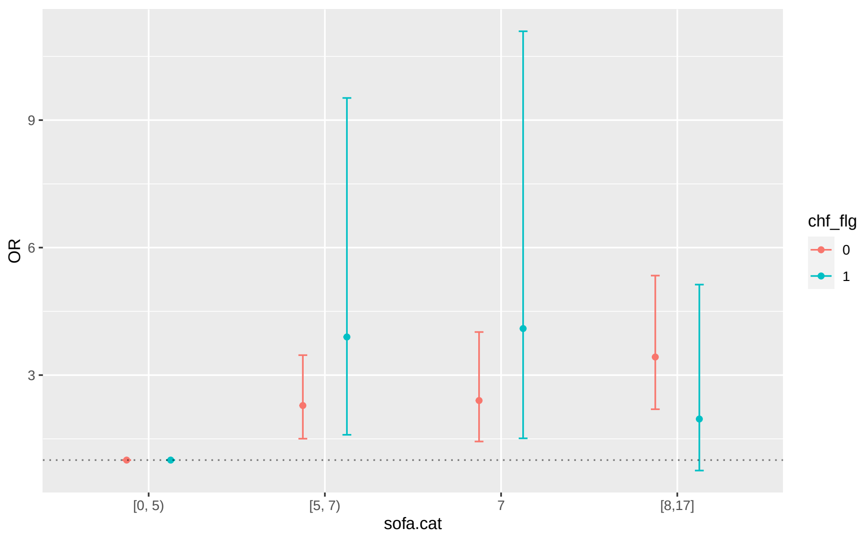
```r
#f )
plot_OR_by_level(dat2,"sofa.cat","day_28_flg")
```



```r
# g)
plot_OR_by_level(dat2,"sofa.cat","day_28_flg",factor.var2="chf_flg")
```



```r
plot_OR_by_level(dat2,"chf_flg","day_28_flg",factor.var2="sofa.cat")
```



OR
9
6
3
[0, 5)
[5, 7)
7
[8,17]
sofa.cat
chf_flg
0
1