Wellbeing and Lifestyle

Xinran Du

1. Data

The `Wellbeing_and_lifestyle_data.csv` dataset (https://www.kaggle.com/ydalat/lifestyle-and-wellbeing-data (https://www.kaggle.com/ydalat/lifestyle-and-wellbeing-data)) contains 12,757 responses to global Work-Life Balance survey. This online survey includes 23 questions about the way we design our lifestyle and achieve work-life balance.

It evaluates how we thrive in both professional and personal lives from five dimensions: healthy body, healthy mind, expertise, connection, meaning. The two main aspects I'm going to focus on are healthy body and healthy mind. For healthy body, the goal is to determine whether `DAILY_STEPS` is the predictor of `BMI RANGE`. For healthy mind, the purpose is to determine whether `DAILY_STRESS` is influenced by `GENDER` and `SUFFICIENT_INCOME`.

The values in this dataset are mainly discrete variables (e.g. count, rating and condition). Therefore, cumulative-link mixed model and ordinal logistic regression model will be adapted.

2. Packages

First, we load `tidyverse` package, a collection of essential packages in R. Here, we use it for data import(`readr`), wrangling, transformation (`tidyr` , `dplyr`) and visualisation (`ggplot2`).

Second, we load `likert` , `HH` , `ggcorrplot` , `gganimate` and `streamgraph` for visualising data.

`likert` is an approach to analyse Likert response items, with an emphasis on visualizations. The stacked bar plot is the preferred method for presenting Likert results.

`HH` is created by Richard M. Heiberger and Burt Holland to explore the extensive use of graphical displays, and show how accompanying traditional tabular results are used to confirm the visual impressions derived directly from the graphs.

In my work, both `likert` and `HH` are used for stacked bar chart.

`ggcorrplot` package can be used to visualise a correlation matrix using `ggplot2` . It includes functions for reordering the correlation matrix, displaying the significance level on the plot, and computing a matrix of correlation p-values.

`gganimate` package provides a set of grammar, fully compatible with `ggplot2` for specifying transitions and animations in graphics.

The `streamgraph` pacakge is an `htmlwidget1` that is based on the `D3.js2` JavaScript library. Byron & Wattenberg describes streamgraphs as “a generalization of stacked area graphs where the baseline is free, thereby making it easier to perceive the thickness of any given layer across the data”. This package allows us to build a streamgraph involving an interactive component that enables filtering each “flow”.

Third, we load `ordinal` , `MASS` and `emmeans` for building models.

`ordinal` provides an approach for implementation of cumulative link (mixed) models. Estimation is via maximum likelihood and mixed models are fitted with the Laplace approximation and adaptive Gauss-Hermite quadrature. Multiple random effect terms are allowed. Here, we use `clmm` function to build cumulative-link mixed model.

`MASS` includes functions supporting computer-intensive methods (e.g. GLMMs) mentioned in Venables and Ripley's book "Modern Applied Statistics with S" (4th edition, 2002). Here, we use `polr` function to build ordinal logistic regression model.

`emmeans` helps to obtain estimated marginal means (EMMs) and compute contrasts or linear functions of EMMs. It can also estimate and contrast slopes of trend lines. We use this for pairwise comparisons.

```
library(tidyverse)
library(HH)
library(likert)
library(ggcorrplot)
library(gganimate)
library(streamgraph)
library(ordinal)
library(MASS)
library(emmeans)
```

3. Data wrangling

Read `Wellbeing_and_lifestyle_data.csv` into R and call it `mydata` .

```
mydata <- read_csv("Wellbeing_and_lifestyle_data.csv")
```

View the dataset. We can see there are 12,756 responses with 23 attributes.

```
structure(mydata)
```

```
## # A tibble: 12,756 x 23
##    Timestamp FRUITS_VEGGIES DAILY_STRESS PLACES_VISITED CORE_CIRCLE
##    <chr>              <dbl>        <dbl>          <dbl>       <dbl>
##  1 07/07/20~              3            2              2           5
##  2 07/07/20~              2            3              4           3
##  3 07/07/20~              2            3              3           4
##  4 07/07/20~              3            3             10           3
##  5 07/07/20~              5            1              3           3
##  6 07/08/20~              3            2              3           9
##  7 07/08/20~              4            2             10           6
##  8 07/09/20~              3            4              5           3
##  9 07/09/20~              5            3              6           4
## 10 07/10/20~              4            4              2           6
## # ... with 12,746 more rows, and 18 more variables: SUPPORTING_OTHERS <dbl>,
## #   SOCIAL_NETWORK <dbl>, ACHIEVEMENT <dbl>, DONATION <dbl>, BMI_RANGE <dbl>,
## #   TODO_COMPLETED <dbl>, FLOW <dbl>, DAILY_STEPS <dbl>, LIVE_VISION <dbl>,
## #   SLEEP_HOURS <dbl>, LOST_VACATION <dbl>, DAILY_SHOUTING <dbl>,
## #   SUFFICIENT_INCOME <dbl>, PERSONAL_AWARDS <dbl>, TIME_FOR_PASSION <dbl>,
## #   DAILY_MEDITATION <dbl>, AGE <chr>, GENDER <chr>
```

Break down `Timestamp` , as we need `year` to be an independent column for analysis. `Timestamp` from 2015 to 2017 has three components: day, month, year (e.g. 07/07/2015); data from 2018 to 2020 includes four components: day, month, year, time (e.g. 01/01/2019 10:53). Firstly, we separate `col = "Timestamp"` into three values `c("day", "month", "year")` by the separator `/` . Then, we separate `year` containing time (e.g. 2019 10:53) into two values `year` and `time` .

```
mydata <- separate(mydata, col = "Timestamp", into = c("day", "month", "year"), sep = "/")
mydata <- separate(mydata, col = "year", into = c("year", "time"), sep = " ")
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 10004 rows [1, 2,
## 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

Recode `year` . Since both `dplyr` and `likert` package have `recode` function and they work differently, we need to specify the function we use is from `dpltr` . `year` has two formats (e.g. 2015 and 15) and we only want to recode `15` as `2015` . Set `.default=NULL` so unmatched values will not be changed.

```
mydata$year <- dplyr::recode(mydata$year, "15" = "2015", "16" = "2016", "17" = "2017", "18" = "2
018", .default = NULL)
```

View the data. We can check `Timestamp` have been separated into four columns and `year` has been separated from it. `NA` is due to `Timestamp` from 2015 to 2017 does not contain `time` component.

```
str(mydata)
```

```
## tibble [12,756 x 26] (S3: tbl_df/tbl/data.frame)
##  $ day              : chr [1:12756] "07" "07" "07" "07" ...
##  $ month            : chr [1:12756] "07" "07" "07" "07" ...
##  $ year             : chr [1:12756] "2015" "2015" "2015" "2015" ...
##  $ time             : chr [1:12756] NA NA NA NA ...
##  $ FRUITS_VEGGIES   : num [1:12756] 3 2 2 3 5 3 4 3 5 4 ...
##  $ DAILY_STRESS     : num [1:12756] 2 3 3 3 1 2 2 4 3 4 ...
##  $ PLACES_VISITED   : num [1:12756] 2 4 3 10 3 3 10 5 6 2 ...
##  $ CORE_CIRCLE      : num [1:12756] 5 3 4 3 3 9 6 3 4 6 ...
##  $ SUPPORTING_OTHERS: num [1:12756] 0 8 4 10 10 10 10 5 3 10 ...
##  $ SOCIAL_NETWORK   : num [1:12756] 5 10 10 7 4 10 10 7 3 10 ...
##  $ ACHIEVEMENT      : num [1:12756] 2 5 3 2 2 2 3 4 5 0 ...
##  $ DONATION         : num [1:12756] 0 2 2 5 4 3 5 0 4 4 ...
##  $ BMI_RANGE        : num [1:12756] 1 2 2 2 2 1 2 1 1 2 ...
##  $ TODO_COMPLETED   : num [1:12756] 6 5 2 3 5 6 8 8 10 3 ...
##  $ FLOW             : num [1:12756] 4 2 2 5 0 1 8 2 2 2 ...
##  $ DAILY_STEPS      : num [1:12756] 5 5 4 5 5 7 7 8 1 3 ...
##  $ LIVE_VISION      : num [1:12756] 0 5 5 0 0 10 5 10 5 0 ...
##  $ SLEEP_HOURS      : num [1:12756] 7 8 8 5 7 8 7 6 10 6 ...
##  $ LOST_VACATION    : num [1:12756] 5 2 10 7 0 0 10 0 0 0 ...
##  $ DAILY_SHOUTING   : num [1:12756] 5 2 2 5 0 2 0 2 2 0 ...
##  $ SUFFICIENT_INCOME: num [1:12756] 1 2 2 1 2 2 2 2 2 1 ...
##  $ PERSONAL_AWARDS  : num [1:12756] 4 3 4 5 8 10 10 8 10 3 ...
##  $ TIME_FOR_PASSION : num [1:12756] 0 2 8 2 1 8 8 2 3 8 ...
##  $ DAILY_MEDITATION : num [1:12756] 5 6 3 0 5 3 10 2 10 1 ...
##  $ AGE              : chr [1:12756] "36 to 50" "36 to 50" "36 to 50" "51 or more" ...
##  $ GENDER           : chr [1:12756] "Female" "Female" "Female" "Female" ...
```

4. Healthy Body

Healthy body in this dataset includes four aspects: `BMI_RANGE` , `FRUITS_VEGGIES` , `DAILY_STEPS` , `SLEEP_HOURS` .

First, we filter these four values out to create a new dataset called `mydata_HB` .

```
mydata_HB <- mydata[c(5, 13, 16, 18)]
```

Correlation

Check which factor is most relevant to `BMI_RANGE` . We can use functions from `ggcorrplot` package to check the correlations.

`cor` computes the correlation and `cor_pmat` compute a matrix of correlation p-values.

`method="kendall"` indicates Kendall's is used to estimate a rank-based measure of association. We use "kendall" here because "spearman" cannot compute exact p-value with ties. Kendall tau rank correlation is also a non-parametric test for statistical dependence between two ordinal variables and can handle ties.

```
corr <- cor(mydata_HB, method = "kendall")
p.mat <- cor_pmat(mydata_HB, method="kendall")
head(corr)
```

```
##                FRUITS_VEGGIES   BMI_RANGE  DAILY_STEPS  SLEEP_HOURS
## FRUITS_VEGGIES     1.00000000 -0.08066264  0.188711942  0.086699345
## BMI_RANGE         -0.08066264  1.00000000 -0.110963165 -0.091674932
## DAILY_STEPS        0.18871194 -0.11096317  1.000000000  0.005557699
## SLEEP_HOURS        0.08669934 -0.09167493  0.005557699  1.000000000
```

```
head(p.mat)
```

```
##                FRUITS_VEGGIES    BMI_RANGE   DAILY_STEPS  SLEEP_HOURS
## FRUITS_VEGGIES   0.000000e+00 1.478855e-24 1.689171e-170 8.199942e-34
## BMI_RANGE        1.478855e-24 0.000000e+00  3.951308e-48 3.525617e-30
## DAILY_STEPS     1.689171e-170 3.951308e-48  0.000000e+00 4.208667e-01
## SLEEP_HOURS      8.199942e-34 3.525617e-30  4.208667e-01 0.000000e+00
```

The charts shows `DAILY_STEPS` has the biggest correlation coefficient (-0.11096317) compared to `FRUITS_VEGGIES` and `SLEEP_HOURS` , and it is significantly negatively related to `BMI_RANGE` (p<0.001).

We can visualise the above results using `ggcorrplot` function.

`hc.order = TRUE` reorders the correlation matrix using hierarchical clustering.

`type = "lower"` helps to get the lower triangle.

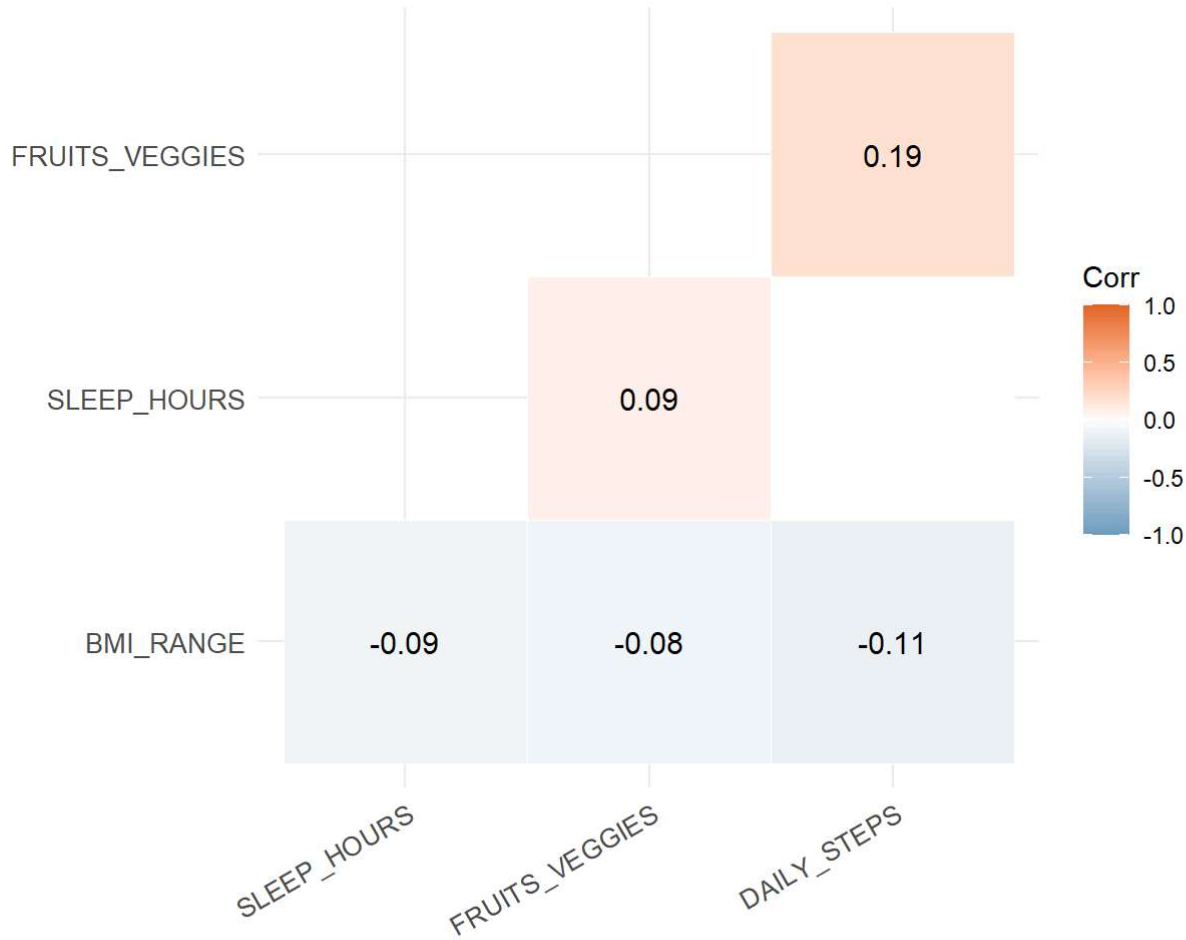
`lab = TRUE` adds correlation coefficients to the plot.

`outline.col = "white"` changes the outline of each block to white.

`p.mat = p.mat1` adds correlation significance level and `insig = "blank"` leaves blank on no significant coefficient.

`theme` adjust the font size and angle of text on axes.

```
ggcorrplot(corr,
           hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           outline.col = "white",
           colors = c("#6D9EC1", "white", "#E46726"),
           p.mat = p.mat,
           insig = "blank") +
  theme(axis.text.x = element_text(size=10, angle=30),
        axis.text.y = element_text(size=10))
```



The plot shows `DAILY_STEPS` is most related to `BMI_RANGE` . Next, we will explore the relationship between these two factors.

Stacked Bar Chart

The stacked bar chart uses bars to show comparisons between categories and segments of data. Here, it is used to show the distribution of each level of `DAILY_STRESS` in different `BMI_RANGE` group.

Recode the values in `BMI_RANGE` column. Set `.default = NULL` so unmatched values won't be changed. `BMI_RANGE` has two categories, "1" means "BMI below 25" and "2" means "BMI above 25". BMI over 25 is used to categorize a person as overweight.

```
mydata_HB$BMI_RANGE <- dplyr::recode(mydata_HB$BMI_RANGE, "1"="BMI Below 25", "2"="BMI Above 2
5", .default = NULL)
```

We can check that the values have been recoded.

```
str(mydata_HB)
```

```
## tibble [12,756 x 4] (S3: tbl_df/tbl/data.frame)
##  $ FRUITS_VEGGIES: num [1:12756] 3 2 2 3 5 3 4 3 5 4 ...
##  $ BMI_RANGE     : chr [1:12756] "BMI Below 25" "BMI Above 25" "BMI Above 25" "BMI Above 25"
...
##  $ DAILY_STEPS   : num [1:12756] 5 5 4 5 5 7 7 8 1 3 ...
##  $ SLEEP_HOURS   : num [1:12756] 7 8 8 5 7 8 7 6 10 6 ...
```

Save the counts as tabular format `mydata_BD` , because tabular results are used to confirm the visual impressions derived directly from the graphs in `HH` package.

`group_by` is used for grouping two variables `BMI_RANGE` and `DAILY_STEPS` .

`tally` is for counting the number of responses.

`mutate` is for creating a new column `count` in which the values equal the obtained numbers in `n` .

`select` is for dropping the `n` variable. Since both `dplyr` and `MASS` package have `select` function and they work differently, we need to specify the function we use is from `dplyr` .

`spread` is for changing the dataset from long format to wide (tabular) format. `DAILY_STEPS` is the key for header and `count` is the value to be filled in data frame.

```
mydata_BD <- mydata_HB %>%
  group_by(BMI_RANGE, DAILY_STEPS) %>%
  tally() %>%
  mutate(count = n) %>%
  dplyr::select(-n) %>%
  spread(DAILY_STEPS, count)
```

View `mydata_BD` .

```
print(mydata_BD)
```

```
## # A tibble: 2 x 11
## # Groups:   BMI_RANGE [2]
##   BMI_RANGE      `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`
##   <chr>        <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1 BMI Above 25   497   576   583   533   723   495   442   399   203   661
## 2 BMI Below 25   473   656   657   692  1007   751   734   829   352  1493
```

As both `HH` package and `likert` package have `likert` function, we need to define `likert` here is from `HH` package.

Plot the data as divergent stacked bar chart using `likert` function from `HH` package.

`BMI_RANGE~` indicates the comparison is based on the two categories in `BMI_RANGE` .

`as.percent=TRUE` shows the percentage of each segment. As the total number of “BMI above 25” and “BMI below 25” are different, it’s hard to compare these two groups by the number, but better with percentage.

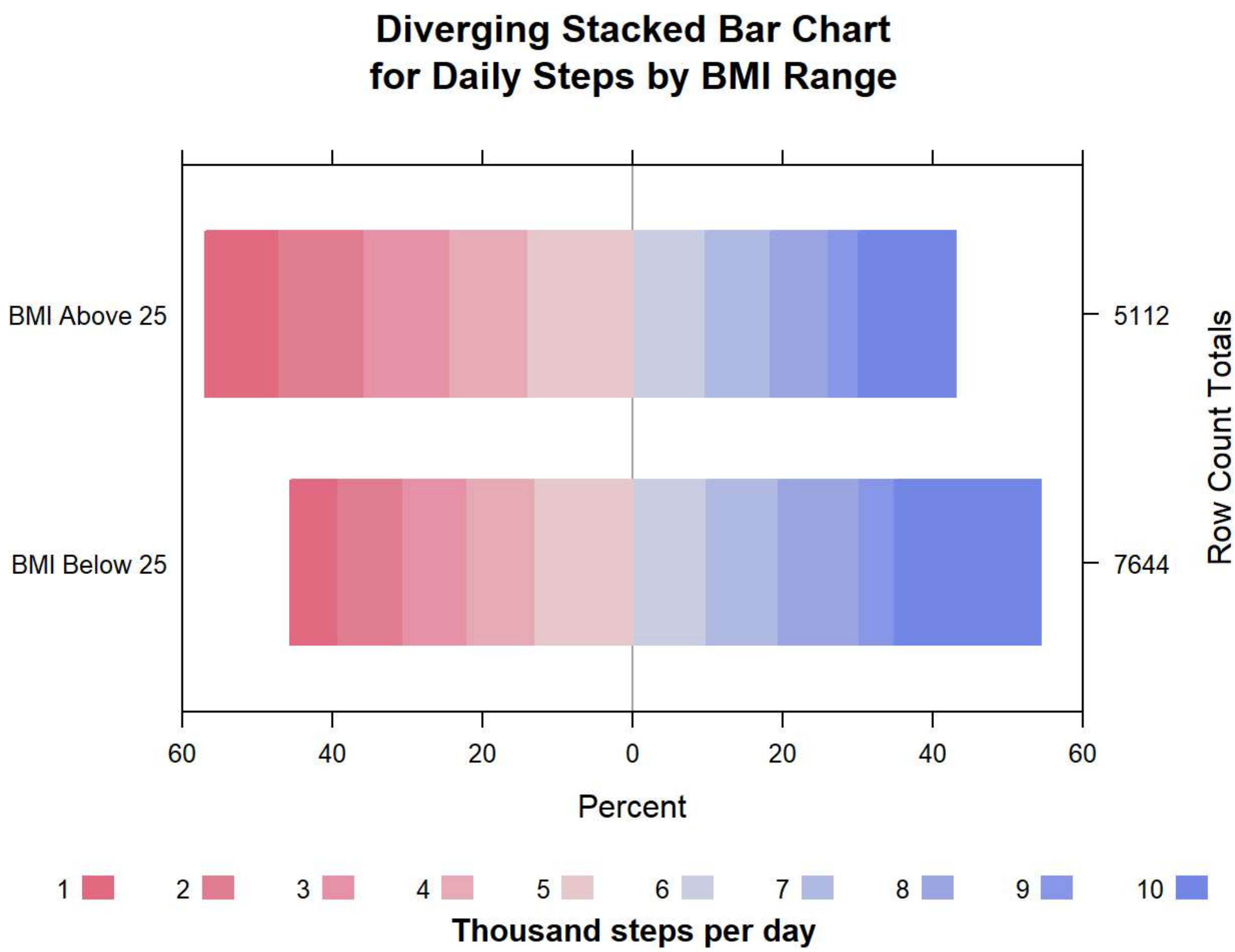
`ylab = NULL` hides the label of y-axis.

`xlim = c(-60,-40,-20,0,20,40,60)` adjust the range of x-axis.

`auto.key=list(cex = 0.6)` adjusts the font size of legend.

`main=` and `sub=` add titles to the plot.

```
HH::likert(BMI_RANGE ~ ., mydata_BD, as.percent = TRUE, ylab = NULL, xlim = c(-60,-40,-20,0,20,4
0,60), auto.key = list(cex = 0.8),
       main = "Diverging Stacked Bar Chart\nfor Daily Steps by BMI Range",
       sub = "Thousand steps per day")
```



The plot shows the proportion of people whose daily steps are over 5,000 in “BMI below 25” group is larger than that in “BMI above 25”. The the proportion of people whose daily steps are less than 5,000 in “BMI below 25” group is smaller than that in “BMI above 25”. That is to say, people who are overweight could have less steps each day. Hence, we will explore whether `DAILY_STEPS` can be predicted by `BMI_RANGE` .

Cumulative Link Mixed Models (CLMM)

We can build a cumulative link mixed model to determine whether `DAILY_STEPS` is influenced by `BMI_RANGE` . So, we have `DAILY_STEPS` as DV, `BMI_RANGE` as IV and `AGE` as random effect. `AGE` includes four groups: less than 20, 21 to 35, 36 to 50, 51 or more.

Before we build models, we need to code DV `DAILY_STEPS` as an ordinal variable, and convert `BMI_RANGE` and `AGE` to factor so they can be treated correctly in the model.

```
mydata$DAILY_STEPS <- as.ordered(mydata$DAILY_STEPS)
mydata$BMI_RANGE <- as.factor(mydata$BMI_RANGE)
mydata$AGE <- as.factor(mydata$AGE)
```

Build null model and experimental model using `clmm` function from `ordinal` package.

```
model.clm.null <- clmm(DAILY_STEPS ~ 1 + (1 + BMI_RANGE | AGE), data = mydata)
model.clm <- clmm(DAILY_STEPS ~ BMI_RANGE  + (1 + BMI_RANGE | AGE), data = mydata)
```

Test whether the experimental model and null model differ.

```
anova(model.clm.null, model.clm)
```

```
## Likelihood ratio tests of cumulative link models:
##
##                formula:                                        link: threshold:
## model.clm.null DAILY_STEPS ~ 1 + (1 + BMI_RANGE | AGE)         logit flexible
## model.clm      DAILY_STEPS ~ BMI_RANGE + (1 + BMI_RANGE | AGE) logit flexible
##
##                no.par   AIC logLik LR.stat df Pr(>Chisq)
## model.clm.null     12 57283 -28629
## model.clm          13 57277 -28625  8.3819  1    0.00379 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Two models are significantly diferent (*P*=0.00379), and the experimental model is better fit the data as it has the lower AIC value (57277<57283).

Show details of `model.clm`.

```
summary(model.clm)
```

```
## Cumulative Link Mixed Model fitted with the Laplace approximation
##
## formula: DAILY_STEPS ~ BMI_RANGE + (1 + BMI_RANGE | AGE)
## data:    mydata
##
##  link  threshold nobs  logLik    AIC      niter      max.grad cond.H
##  logit flexible  12756 -28625.29 57276.57 2529(5060) 4.58e-02 6.3e+02
##
## Random effects:
##  Groups Name        Variance Std.Dev. Corr
##  AGE    (Intercept) 0.007819 0.08843
##         BMI_RANGE2  0.020099 0.14177  -0.603
## Number of groups:  AGE 4
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## BMI_RANGE2 -0.45414    0.08052   -5.64  1.7e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##      Estimate Std. Error z value
## 1|2  -2.73503    0.05799 -47.166
## 2|3  -1.79904    0.05272 -34.123
## 3|4  -1.22181    0.05112 -23.899
## 4|5  -0.77074    0.05039 -15.295
## 5|6  -0.20687    0.04996  -4.141
## 6|7   0.19493    0.04995   3.903
## 7|8   0.60522    0.05023  12.049
## 8|9   1.11524    0.05111  21.821
## 9|10  1.40069    0.05194  26.970
```

The result indicates that `DAILY_STEPS` is significantly influenced by `BMI_RANGE` (z=-5.64, *P*<0.001).

Explore the effect of `BMI_RANGE` using `emmeans` function.

```
emmeans(model.clm, pairwise ~ BMI_RANGE, adjust = "none")
```

```
## $emmeans
##  BMI_RANGE  emmean     SE  df asymp.LCL asymp.UCL
##  1          0.3797 0.0493 Inf     0.283    0.4763
##  2         -0.0744 0.0648 Inf    -0.201    0.0525
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast estimate     SE  df z.ratio p.value
##  1 - 2       0.454 0.0805 Inf 5.640   <.0001
```

The pairwise comparisons report that `DAILY_STEPS` of people with BMI below 25 is significantly higher than that of people with BMI above 25 (z=5.640, *P*<0.001).

To conclude, people’s daily steps is influenced by their BMI range. When people’s BMI is below 25, they have more daily steps than people with BMI above 25.

5. Healthy Mind

We will choose `DAILY_STRESS` from healthy mind as the object. In the survey, `DAILY_STRESS` has six levels from 0 to 6. “0” means not much stress and “5” means a lot of stress.

We need to filter out the missing values in `DAILY_STRESS` column and save the new dataset as `mydata_F`.

```
mydata_F <- mydata %>%
  filter(!is.na(DAILY_STRESS))
```

General Visualisation of Daily Stress by Year

First, let’s create a bubble plot to show the number of people on each level of daily stress from 2015 to 2020.

We need to build a new dataset containing the counts of each group.

`group_by` helps to group the responses by `DAILY_STRESS` and `year`.

`tally` is for counting the number of responses.

```
mydata_count <- mydata_F %>%
  group_by(DAILY_STRESS, year) %>%
  tally()
```

View `mydata_count`.

```
head(mydata_count)
```

```
## # A tibble: 6 x 3
## # Groups:   DAILY_STRESS [1]
##   DAILY_STRESS year      n
##          <dbl> <chr> <int>
## 1            0 2015     74
## 2            0 2016    151
## 3            0 2017    135
## 4            0 2018     87
## 5            0 2019    100
## 6            0 2020     15
```

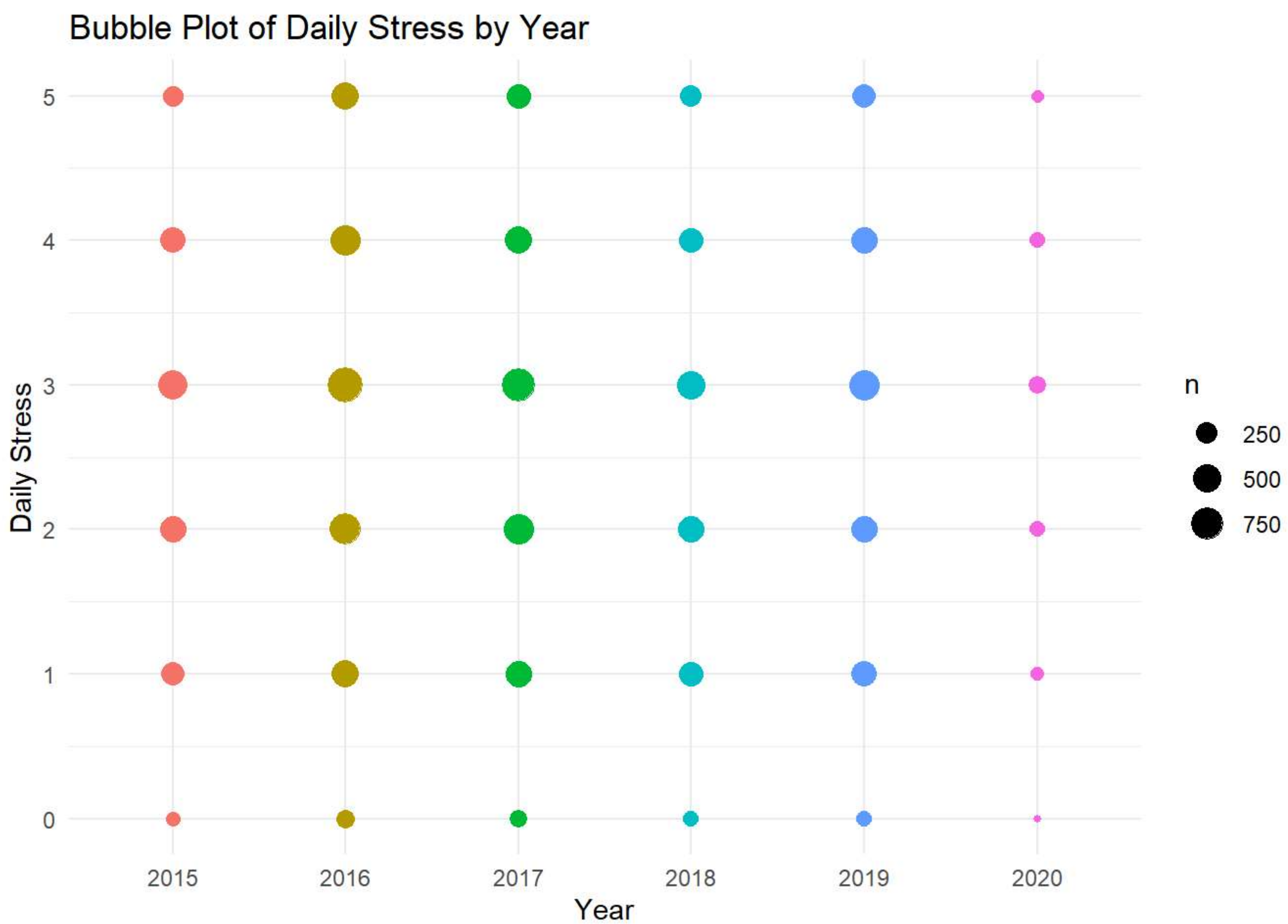
Let’s build the plot. Set `year` on x-axis and `DAILY_STRESS` on y-axis.

`geom_point` add the points which represent counts by size.

`theme_minimal` sets minimalistic theme with no background annotations.

`guides(colour = FALSE)` hide the legend and `labs` adds title and labels of axes.

```
ggplot(mydata_count, aes(x = year, y = DAILY_STRESS, colour = year)) +
  geom_point(aes(size = n)) +
  theme_minimal() +
  guides(colour = FALSE) +
  labs(title = "Bubble Plot of Daily Stress by Year",
       x = "Year",
       y = "Daily Stress")
```



The plot shows that people on level 3 account for the largest proportion in every year.

Also, we can build a streamgraph which allows us to interact with data using `streamgraph` package.

`DAILY_STRESS` is key, `n` (count) is value and `year` refers to the argument “data”.

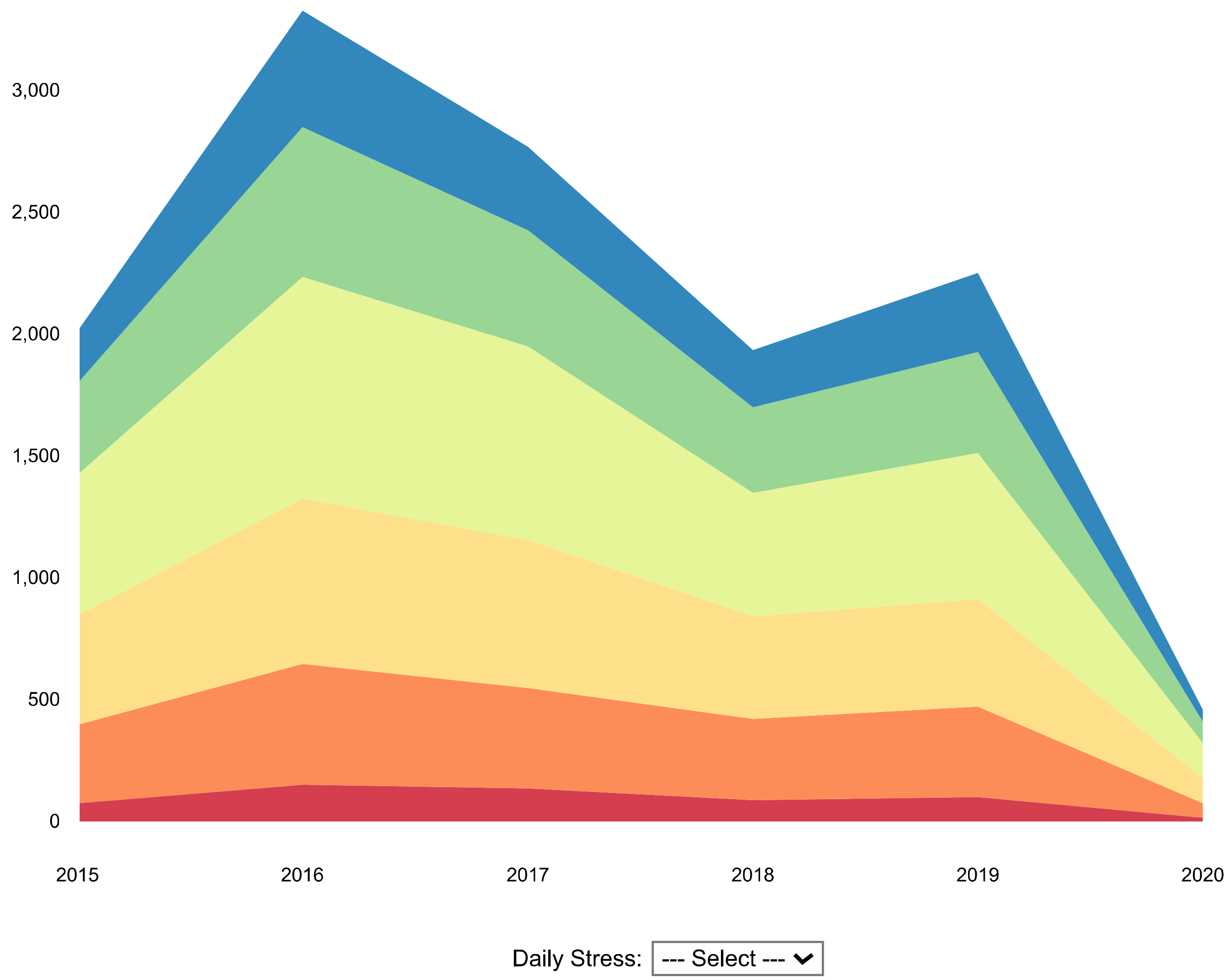
`offset="zero"` changes the baseline for the streamgraph to 0.

`interpolate="linear"` uses a linear interpolation (making the graph more “pointy”) .

`sg_axis_x(1)` changes the aesthetics by using year ticks every year.

`sg_legend` adds a select menu with all the categories of the `DAILY_STRESS` . Selecting a category will highlight that stream on the streamgraph.

```
streamgraph(mydata_count,
            "DAILY_STRESS", "n", "year",
            offset="zero",
            interpolate="linear") %>%
  sg_axis_x(1) %>%
  sg_legend(show=TRUE, label="Daily Stress: ")
```



The graph shows the number of responses in 2016 is the largest. People on level 3 accounts for the largest proportion, while people on level 0 accounts for the smallest proportion. The number of participants increased sharply from 2015 to 2016, dropped for the next two years, and grew slightly in 2019. The number of 2020 is small because the data is only up to February.

Stacked Bar Chart

The above two graphs show the counts of each group. Now we want to check the percentage of each stress level in these 6 years. Except for `HH` package, we can also use `likert` package to create a stacked bar chart.

First, we filter out two columns `DAILY_STRESS` and `year` as a new dataset. Then, we need to convert the dataset from tibble to data frame and encode two vectors as ordered factor so it can work with `likert` package.

```
mydata_DY <- as.data.frame(mydata_F[c(3,6)])
mydata_DY$year <- as.ordered(mydata_DY$year)
mydata_DY$DAILY_STRESS <- as.ordered(mydata_DY$DAILY_STRESS)
```

Check the structure of `mydata_DY` .

```
str(mydata_DY)
```

```
## 'data.frame':    12755 obs. of  2 variables:
##  $ year        : Ord.factor w/ 6 levels "2015"<"2016"<..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ DAILY_STRESS: Ord.factor w/ 6 levels "0"<"1"<"2"<"3"<..: 3 4 4 4 2 3 3 5 4 5 ...
```

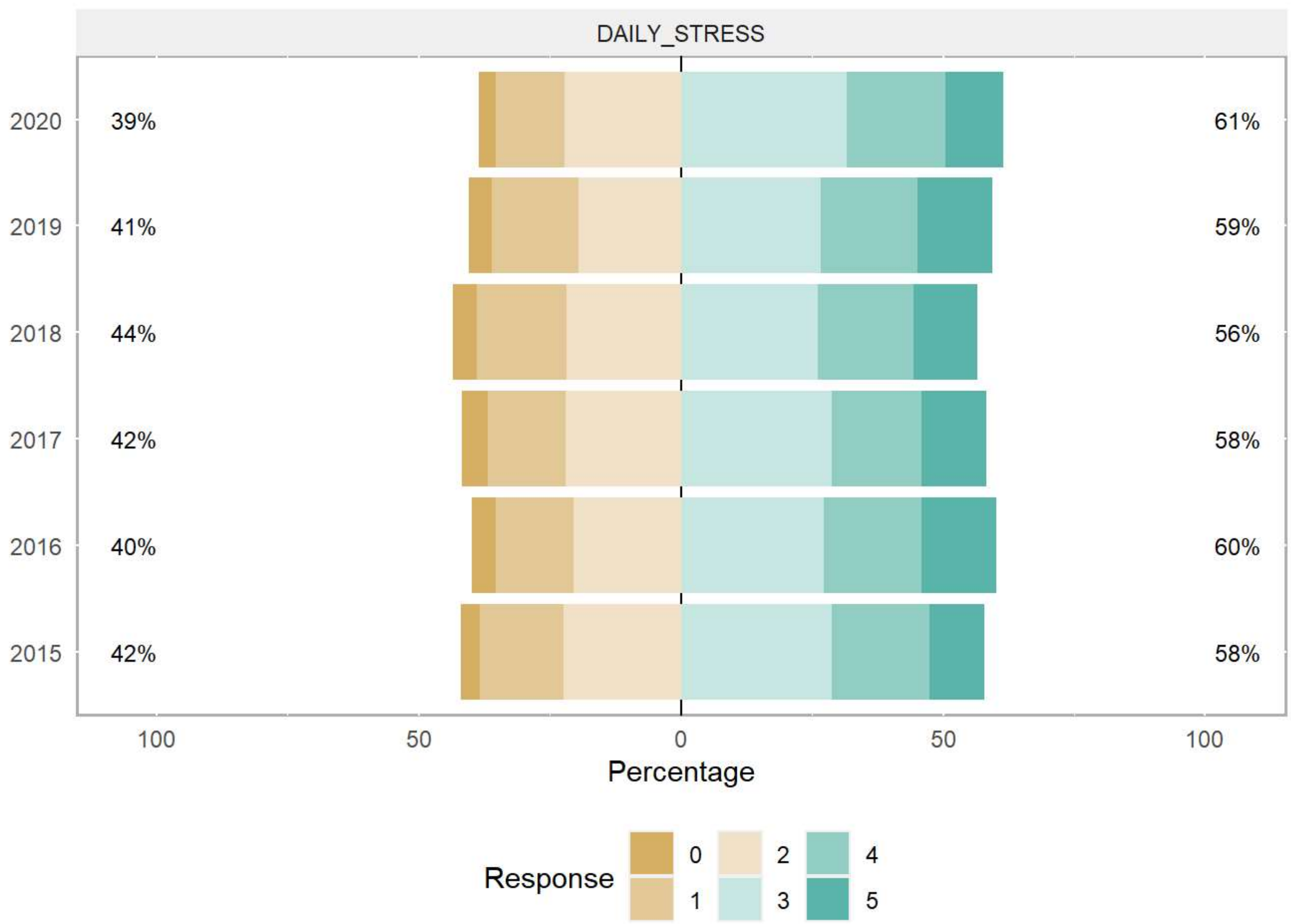
As both `HH` package and `likert` package have `likert` function, we need to define `likert` here is from `likert` package.

`likert` function provides various statistics about likert items. As we try to subset the data frame to analyse only the second column `DAILY_STRESS` , we need to add `drop=FALSE` . `grouping` means the results should be summarized by `year` .

```
likt <- likert::likert(items = mydata_DY[,2, drop=FALSE], grouping = mydata_DY[,1])
```

Plot based on the statistics in `likt` .

```
plot(likt)
```



It is consistent with our previous result that people on level 3 account for the largest proportion. Moreover, the percentages of people on higher level (3,4,5) of daily stress are higher than that of people on lower level (0,1,2) every year.

Animation

We can also visualise the change of daily stress in each age group from 2015 to 2020 using animation.

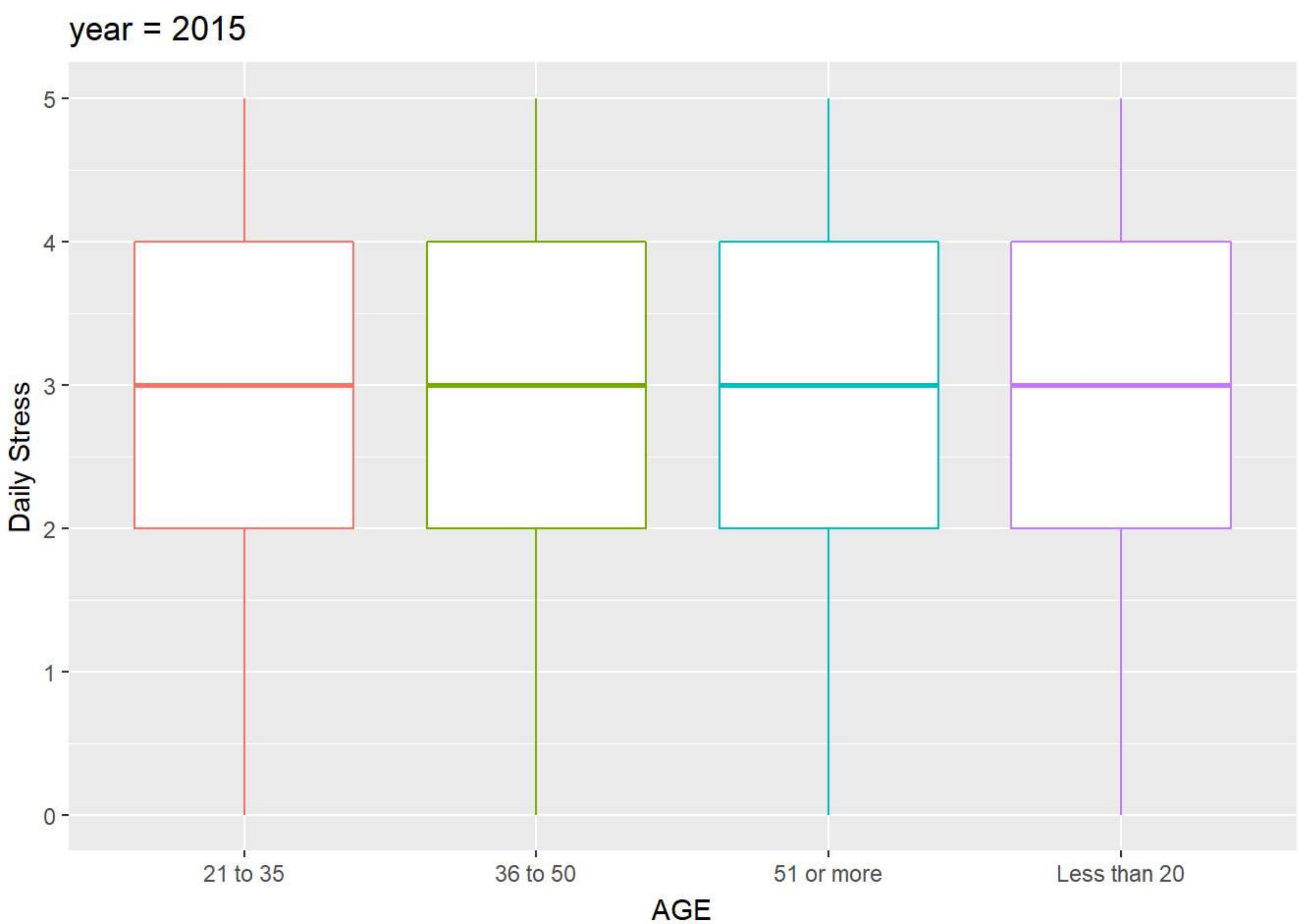
`ggplot` puts `AGE` on x-axis and `DAILY_STRESS` on y-axis and `geom_boxplot` is used to build a boxplot.

`transition` defines how the data should be spread out and how it relates to itself across time. `transition_length` represents the relative length of the transition which will be recycled to match the number of states in the data. `state_length` represents the relative length of the pause at the states.

`guides(colour = FALSE)` hides the legend.

`title="year={closest_state}"` returns the name of the state closest to this frame. Here, the year related to each state is added to the title.

```
ggplot(mydata_F, aes(x = AGE, y = DAILY_STRESS, colour = AGE)) +
  geom_boxplot() +
  transition_states(year,
    transition_length = 2,
    state_length = 2) +
    guides(colour = FALSE) +
    labs(x = "AGE", y = "Daily Stress", title = "year = {closest_state}")
```



The plot shows that the median, first quartile and third quartile of “21 to 35” and “35 to 50” did not change across six years. While, the summary of “51 or more” changed every year. The data of “51 or more” group had the lowest first quartile in 2016 and lowest third quartile in 2020.

Ordinal Logistic Regression Model

Ordinal logistic regression is an extension of simple logistic regression model. In simple logistic regression, dependent variable is categorical and the modeling ignores its ordering. Ordinal logistic regression model overcomes this limitation by using cumulative events for the log of the odds computation. In this analysis, we want to know whether `GENDER` (1=Male, 2=Female) and `SUFFICIENT_INCOME` (1=Not or hardly sufficient to cover basic expenses, 2=Sufficient to cover basic expenses) will influence `DAILY_STRESS`.

First, let's examine whether `GENDER` and `SUFFICIENT_INCOME` are relevant to `DAILY_STRESS` by `cor.test` function from `ggcorrplot` package.

As `cor.test` requires numeric vectos, we need to recode `GENDER` and convert it into number.

```
mydata_F$GENDER <- dplyr::recode(mydata_F$GENDER, "Male" = "1", "Female" = "2")
mydata_F$GENDER <- as.numeric(mydata_F$GENDER)
```

In `cor.test`, `method="kendall"` indicates Kendall's is used to estimate a rank-based measure of association. We use "kendall" here because "spearman" cannot compute exact p-value with ties. Kendall tau rank correlation is also a non-parametric test for statistical dependence between two ordinal variables and can handle ties.

```
cor.test(mydata_F$GENDER, mydata_F$DAILY_STRESS, method = "kendall")
```

```
##
##  Kendall's rank correlation tau
##
## data:  mydata_F$GENDER and mydata_F$DAILY_STRESS
## z = 14.091, p-value < 2.2e-16
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##       tau
## 0.1112779
```

```
cor.test(mydata_F$SUFFICIENT_INCOME, mydata_F$DAILY_STRESS, method = "kendall")
```

```
##
##  Kendall's rank correlation tau
##
## data:  mydata_F$SUFFICIENT_INCOME and mydata_F$DAILY_STRESS
## z = -16.336, p-value < 2.2e-16
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##        tau
## -0.1290017
```

The results suggest that `DAILY_STRESS` is significantly correlated with `GENDER` (z=14.091, P<0.001, tau=0.1112779) and `SUFFICIENT_INCOME` (z=-16.336, P<0.001, tau=-0.1290017).

Build the ordinal logistic regression model using `polr` function from `MASS` package.

In the formula, `DAILY_STRESS` is DV, `GENDER` and `SUFFICIENT_INCOME` are IVs.

Set `Hess = TRUE` to return hessian matrix.

```
mydata_F$DAILY_STRESS <- as.factor(mydata_F$DAILY_MEDITATION)
model.olr<- polr(formula = DAILY_STRESS ~ GENDER + SUFFICIENT_INCOME, data = mydata_F, Hess = TR
UE)
summary(model.olr)
```

```
## Call:
## polr(formula = DAILY_STRESS ~ GENDER + SUFFICIENT_INCOME, data = mydata_F,
##     Hess = TRUE)
##
## Coefficients:
##                     Value Std. Error t value
## GENDER            -0.3457    0.03210  -10.77
## SUFFICIENT_INCOME  0.2699    0.03533    7.64
##
## Intercepts:
##      Value    Std. Error t value
## 0|1   -4.0754   0.1038   -39.2577
## 1|2   -2.7946   0.0884   -31.6051
## 2|3   -1.9452   0.0848   -22.9472
## 3|4   -1.3118   0.0835   -15.7093
## 4|5   -0.8977   0.0830   -10.8108
## 5|6   -0.3622   0.0827    -4.3789
## 6|7   -0.1010   0.0827    -1.2215
## 7|8    0.4822   0.0827     5.8339
## 8|9    0.7552   0.0827     9.1284
## 9|10   0.8885   0.0828    10.7280
##
## Residual Deviance: 54911.54
## AIC: 54935.54
```

The results suggest that an increase in value of `GENDER` by one unit decreases the expected value of `DAILY_STRESS` in log odds by 0.3457, and an increase in value of `SUFFICIENT_INCOME` by one unit increases the expected value of `DAILY_STRESS` in log odds by 0.2699. Give the first category (0|1) as an example, the estimated model can be written as:

logit(P(Y≤1)) = -4.0754 - (-0.3457) * GENDER - 0.2699 * SUFFICIENT_INCOME

To conclude, `DAILY_STRESS` is positively influenced by `GENDER` and negatively influenced by `SUFFICIENT_INCOME`. Female's stress level is higher than male. The stress level of people who have sufficient income to cover life basic expenses is lower than that of people whose income is not or hardly sufficient.