Data Visualization CHIS Exercise

John Campi

November 14, 2018

Contents

1 Case Study

In this case study, we'll explore the large, publicly available California Health Interview Survey dataset from 2009. We'll go step-by-step through the development of a new plotting method - a mosaic plot - combining statistics and flexible visuals. At the end, we'll generalize our new plotting method to use on a variety of datasets we've seen throughout the first two courses.

1.1 CHIS - Descriptive Statistics

See video. Original dataset located at (http://healthpolicy.ucla.edu) Modified dataset (adult.csv) located in Data directory.

1.2 Exploring Data

In this chapter we're going to continuously build on our plotting functions and understanding to produce a mosaic plot (aka Marimekko plot). This is a visual representation of a contingency table, comparing two categorical variables. Essentially, our question is which groups are over or under represented in our dataset. To visualize this we'll color groups according to their Pearson residuals from a chi-squared test. At the end of it all we'll wrap up our script into a flexible function so that we can look at other variables.

We'll familiarize ourselves with a small number of variables from the 2009 CHIS adult-response dataset (as opposed to children). We have selected the following variables to explore:

- RBMI: BMI Category description
- BMI_P: BMI value
- RACEHPR2: race
- SRSEX: sex
- SRAGE_P: age
- MARIT2: Marital status
- AB1: General Health Condition
- ASTCUR: Current Asthma Status

- AB51: Type I or Type II Diabetes
- POVLL: Poverty level

We'll filter our dataset to plot a more reliable subset (we'll still retain over 95% of the data).

Before we get into mosaic plots it's worthwhile exploring the dataset using simple distribution plots - i.e. histograms.

ggplot2 is already loaded and the dataset, named adult, is already available in the workspace.

Instructions

- Use the typical commands for exploring the structure of adult to get familiar with the variables: summary() and str().
- As a first exploration of the data, plot two histograms using ggplot2 syntax: one for age (SRAGE_P) and one for BMI (BMI_P). The goal is to explore the dataset and get familiar with the distributions here. Feel free to explore different bin widths. We'll ask some questions about these in the next exercises.
- Next plot a binned-distribution of age, filling each bar according to the BMI categorization. Inside geom_histogram(), set binwidth = 1. You'll want to use fill = factor(RBMI) since RBMI is a categorical variable.

```
# Read 2009 CHIS adult-response dataset.
path_in <- file.path("Data", "adult.csv")
adult <- read.csv(path_in, stringsAsFactors=FALSE, na.strings = "")

# Explore the dataset with summary and str
summary(adult)
```

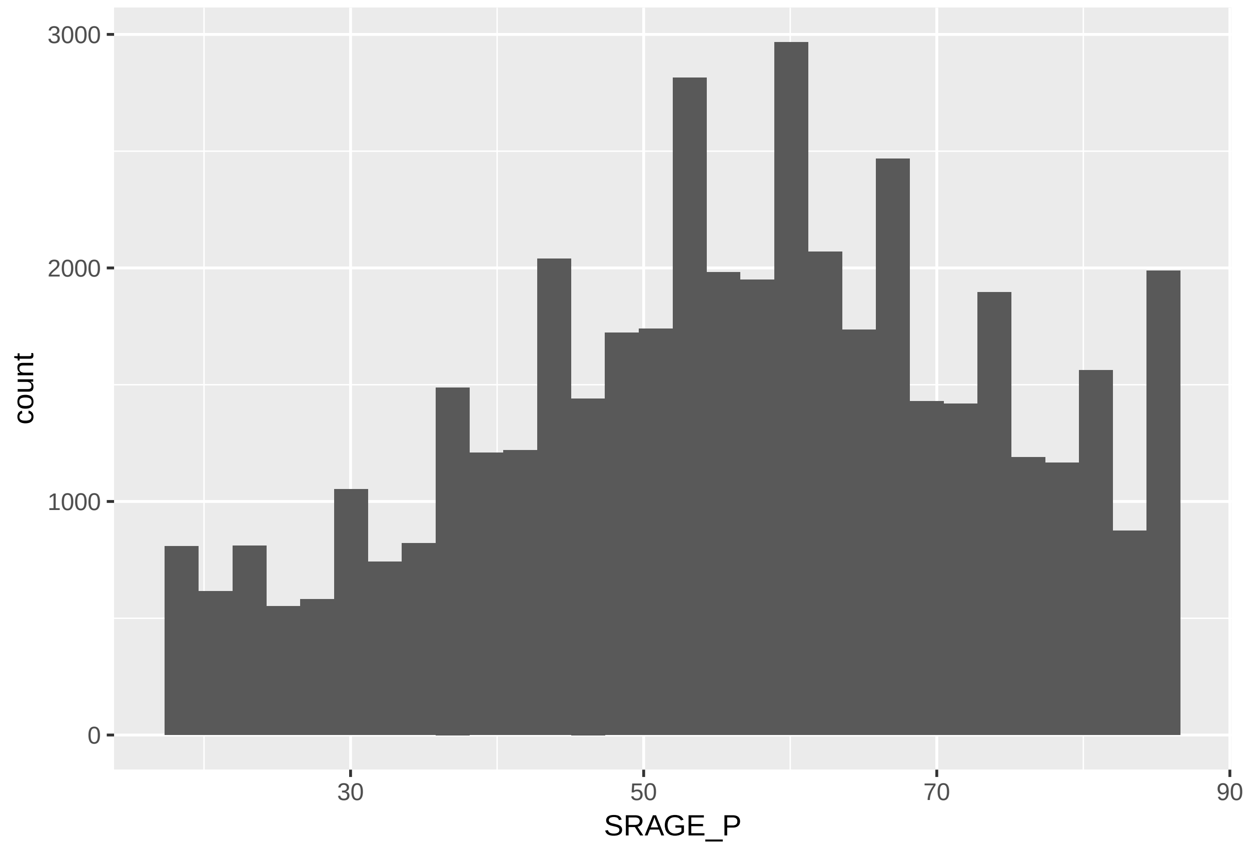
```
##       RBMI           BMI_P          RACEHPR2        SRSEX
##  Min.   :1.000   Min.   :12.65   Min.   :1.000   Min.   :1.000
##  1st Qu.:2.000   1st Qu.:22.77   1st Qu.:5.000   1st Qu.:1.000
##  Median :3.000   Median :25.72   Median :6.000   Median :2.000
##  Mean   :2.748   Mean   :26.64   Mean   :5.088   Mean   :1.591
##  3rd Qu.:3.000   3rd Qu.:29.32   3rd Qu.:6.000   3rd Qu.:2.000
##  Max.   :4.000   Max.   :93.72   Max.   :6.000   Max.   :2.000
##     SRAGE_P         MARIT2           AB1            ASTCUR
##  Min.   :18.00   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:44.00   1st Qu.:1.000   1st Qu.:2.000   1st Qu.:2.000
##  Median :57.00   Median :1.000   Median :2.000   Median :2.000
##  Mean   :56.14   Mean   :2.043   Mean   :2.525   Mean   :1.915
##  3rd Qu.:69.00   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:2.000
##  Max.   :85.00   Max.   :4.000   Max.   :5.000   Max.   :2.000
##       AB51              POVLL
##  Min.   :-1.0000   Min.   :1.000
##  1st Qu.:-1.0000   1st Qu.:2.000
##  Median :-1.0000   Median :4.000
##  Mean   :-0.7108   Mean   :3.196
##  3rd Qu.:-1.0000   3rd Qu.:4.000
##  Max.   : 3.0000   Max.   :4.000
```

```
str(adult)
```

```
## 'data.frame':    44346 obs. of  10 variables:
##  $ RBMI    : int  3 3 3 2 3 4 3 2 3 3 ...
##  $ BMI_P   : num  28.9 26.1 25.1 25 25.1 ...
##  $ RACEHPR2: int  6 6 6 6 6 6 6 6 6 6 ...
##  $ SRSEX   : int  1 2 1 1 1 2 1 2 1 2 ...
##  $ SRAGE_P : int  32 80 71 39 75 53 42 33 67 52 ...
##  $ MARIT2  : int  1 3 1 4 1 1 1 1 3 3 ...
##  $ AB1     : int  1 1 2 1 2 3 2 2 1 5 ...
##  $ ASTCUR  : int  2 2 1 2 2 1 2 2 2 2 ...
##  $ AB51    : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ POVLL   : int  4 4 4 4 4 4 4 3 4 4 ...
```

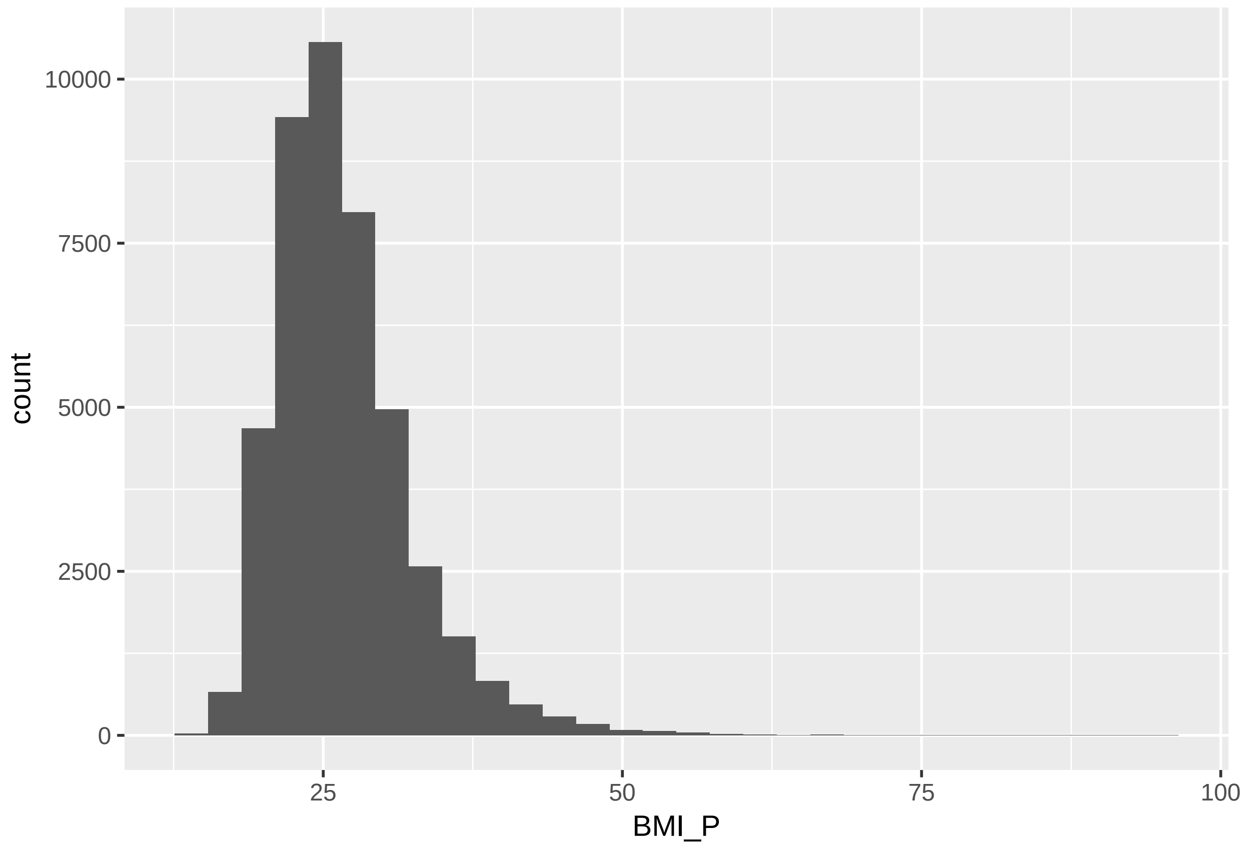
```
# Age histogram
ggplot(adult, aes(x = SRAGE_P)) +
  geom_histogram()
```

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

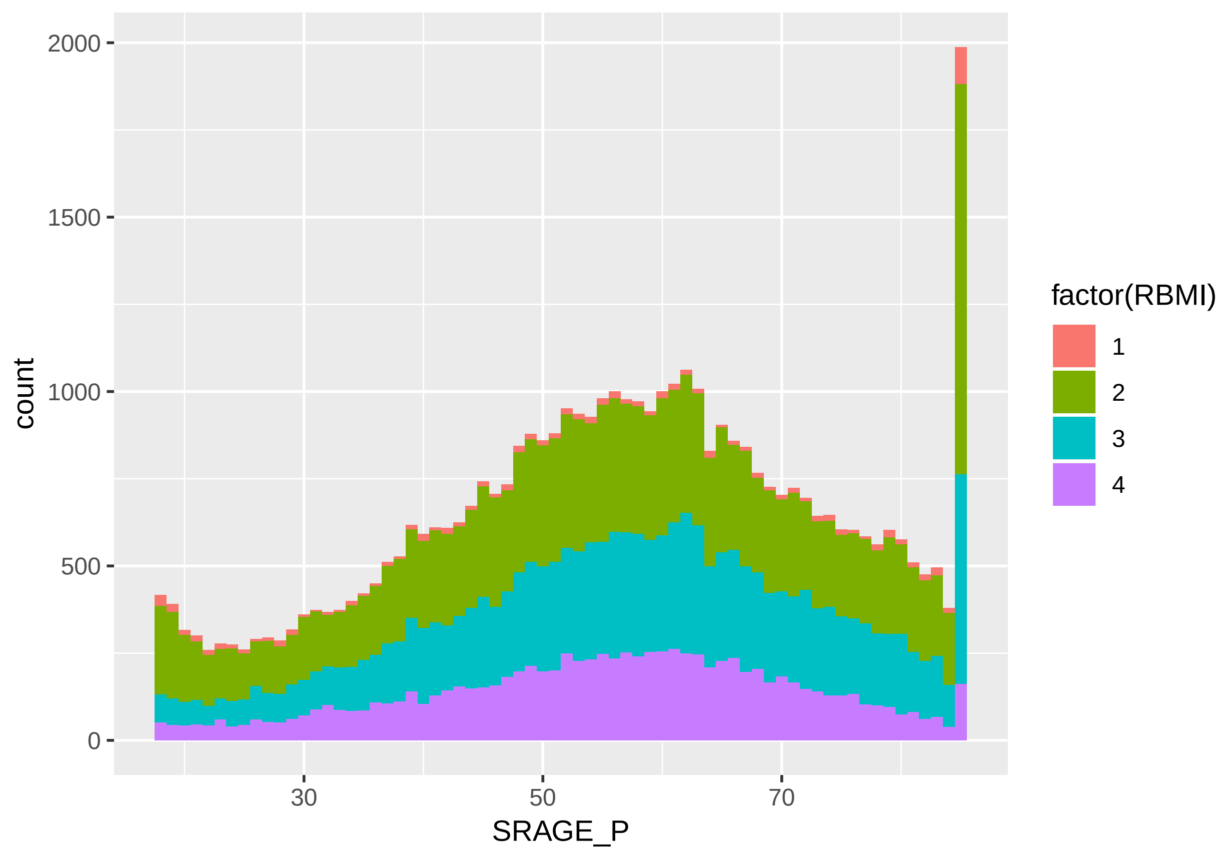


```
# BMI value histogram
ggplot(adult, aes(x = BMI_P)) +
  geom_histogram()
```

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



```
# Age colored by BMI, binwidth = 1
ggplot(adult, aes(x = SRAGE_P, fill=factor(RBMI))) +
  geom_histogram(binwidth=1)
```



1.3 Unusual Values

In the previous exercise you used histograms to explore the age and BMI distributions and their relationships to each other in the CHIS dataset. What unusual phenomenon stood out?

If you want to experiment some more with the data, go ahead - it's available as adult in your workspace!

Instructions

Possible Answers (1) BMI is symmetrical around 25, the border between healthy and overweight. (2) There is an unexpectedly large number of very old people. (3) The proportion of each BMI category is consistent throughout ages. (4) The dataset indicates that California is a very young population.

Answer: (2) There is an unexpectedly large number of very old people.

1.4 Default Binwidths

If you don't specify the binwidth argument inside geom_histogram() you can tell from the message that 30 bins are used by default. This will then specify the binwidth that is used. What is this binwidth for the age variable, SRAGE_P, of the adult dataset?

Instructions

Possible Answers (1) 1.00 (2) 1.66 (3) 2.00 (4) 2.23

Answer: (85 - 18)/30 = 2.23

1.5 Data Cleaning

Now that we have an idea about our data, let's clean it up.

Instructions

- You should have noticed in the age distribution that there is an unusual spike of individuals at 85, which seems like an artifact of data collection and storage. Solve this by only keeping observations for which adult$SRAGE_P is smaller than or equal to 84.
- There is a long positive tail on the BMIs that we'd like to remove. Only keep observations for which adult$BMI_Pislargerthanorequalto16andadult$BMI_P is strictly smaller than 52.
- We'll focus on the relationship between the BMI score (& category), age and race. To make plotting easier later on, we'll change the labels in the dataset. Define adult$RACEHPR2asafactorwithlabelsc(”Latino”,”Asian”,”Afri$
using the labels c("Under-weight", "Normal-weight", "Over-weight", "Obese").

```r
# Keep adults younger than or equal to 84
adult <- adult[adult$SRAGE_P <= 84, ]

# Keep adults with BMI at least 16 and less than 52
adult <- adult[adult$BMI_P >= 16 & adult$BMI_P < 52, ]

# Relabel the race variable
adult$RACEHPR2 <- factor(adult$RACEHPR2, labels = c("Latino", "Asian", "African American", "White"))

# Relabel the BMI categories variable
adult$RBMI <- factor(adult$RBMI, labels = c("Under-weight", "Normal-weight", "Over-weight", "Obese"))
```

1.6 Multiple Histograms

When we introduced histograms we focused on univariate data, which is exactly what we've been doing here. However, when we want to explore distributions further there is much more we can do. For example, there are density plots, which you'll explore in the next course. For now, we'll look deeper at frequency histograms and begin developing our mosaic plots.

The adult dataset, which is cleaned up by now, is available in the workspace for you.

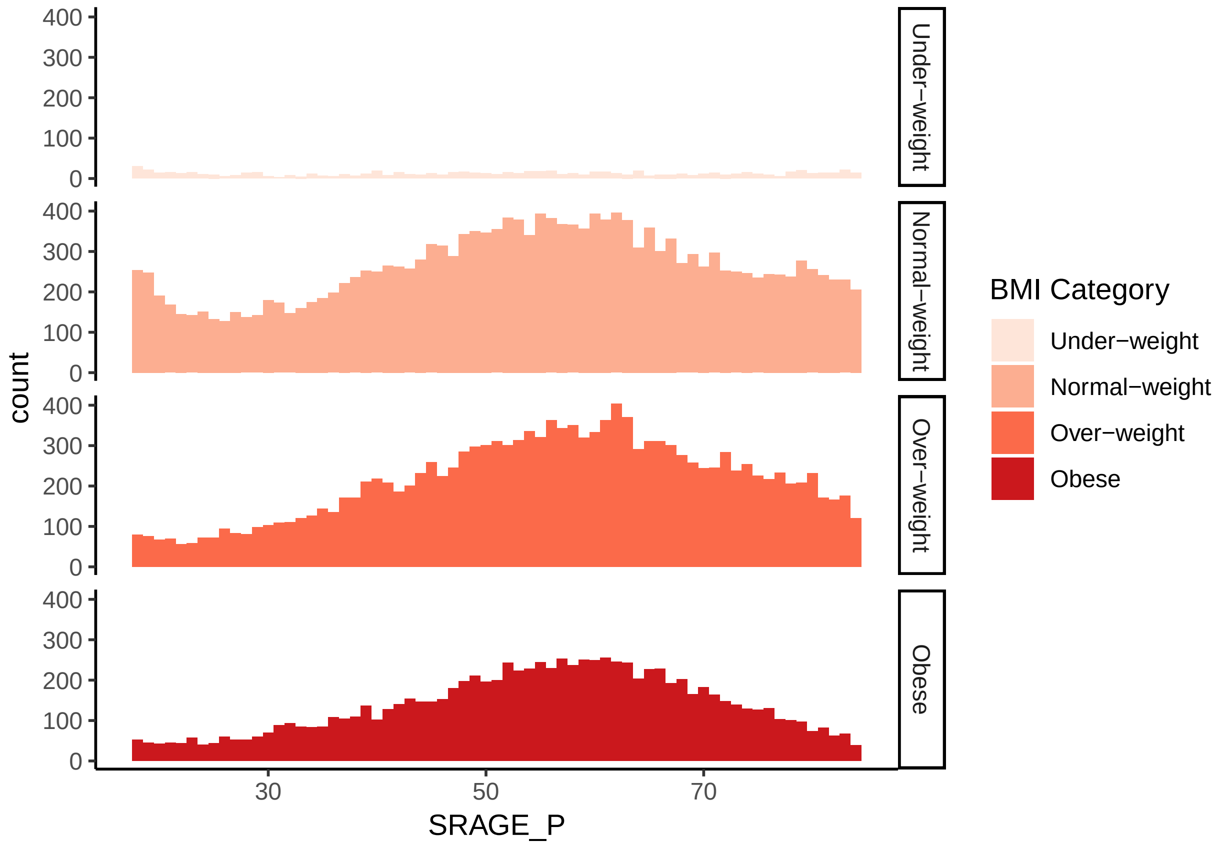
Two layers have been pre-defined for you: BMI_fill is a scale layer which we can add to a ggplot() command using +: ggplot(...) + BMI_fill. fix_strips is a theme() layer to make nice facet titles.

Instructions

- The histogram from the first exercise of age colored by BMI has been provided. The predefined theme(), fix_strips (see above), has been added to the histogram. Add BMI_fill to this plot using the + operator as well.
- In addition, add the following elements to create a pretty & insightful plot:
- Use facet_grid() to facet the rows according to RBMI (Remember formula notation ROWS ~ COL and use . as a place-holder when not facetting in that direction).
- Add the classic theme using theme_classic().

```r
# The dataset adult is available

# The color scale used in the plot
BMI_fill <- scale_fill_brewer("BMI Category", palette = "Reds")

# Theme to fix category display in faceted plot
fix_strips <- theme(strip.text.y = element_text(angle = 0,
                                                hjust = 0,
                                                vjust = 0.1,
                                                size = 14),
                                                strip.background = element_blank(),
                                                legend.position = "none")
```

```r
# Histogram, add BMI_fill and customizations
ggplot(adult, aes (x = SRAGE_P, fill= factor(RBMI))) +
  geom_histogram(binwidth = 1) +
  fix_strips +
  BMI_fill +
  facet_grid(RBMI ~ .) +
  theme_classic()
```



1.7 Alternatives

In the previous exercise we looked at different ways of showing the absolute count of multiple histograms. This is fine, but density would be a more useful measure if we wanted to see how the frequency of one variable changes across another. However, there are some difficulties here, so let's take a closer look at different plots.

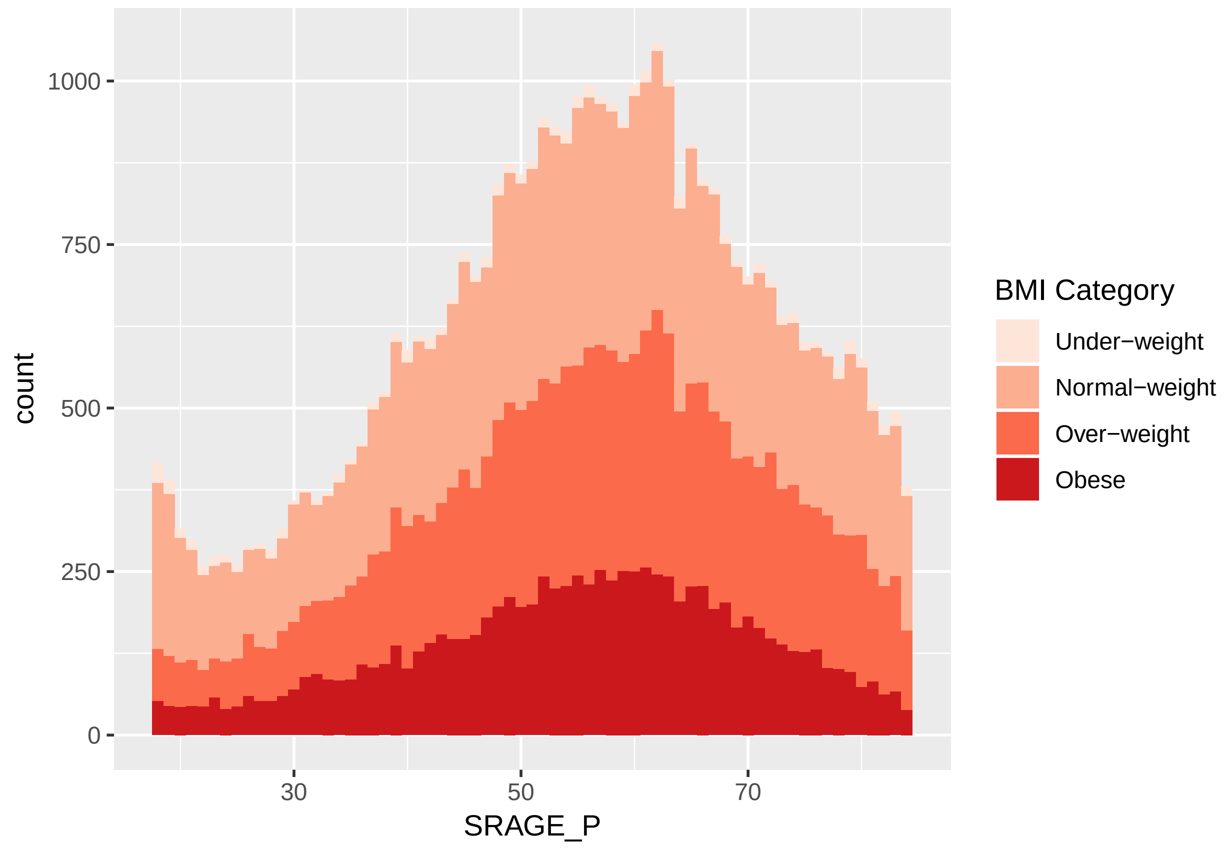
The clean adult dataset is available, as is the BMI_fill color palette. The first plot simply shows a histogram of counts, without facets, without modified themes. It's denoted Plot 1.

Instructions

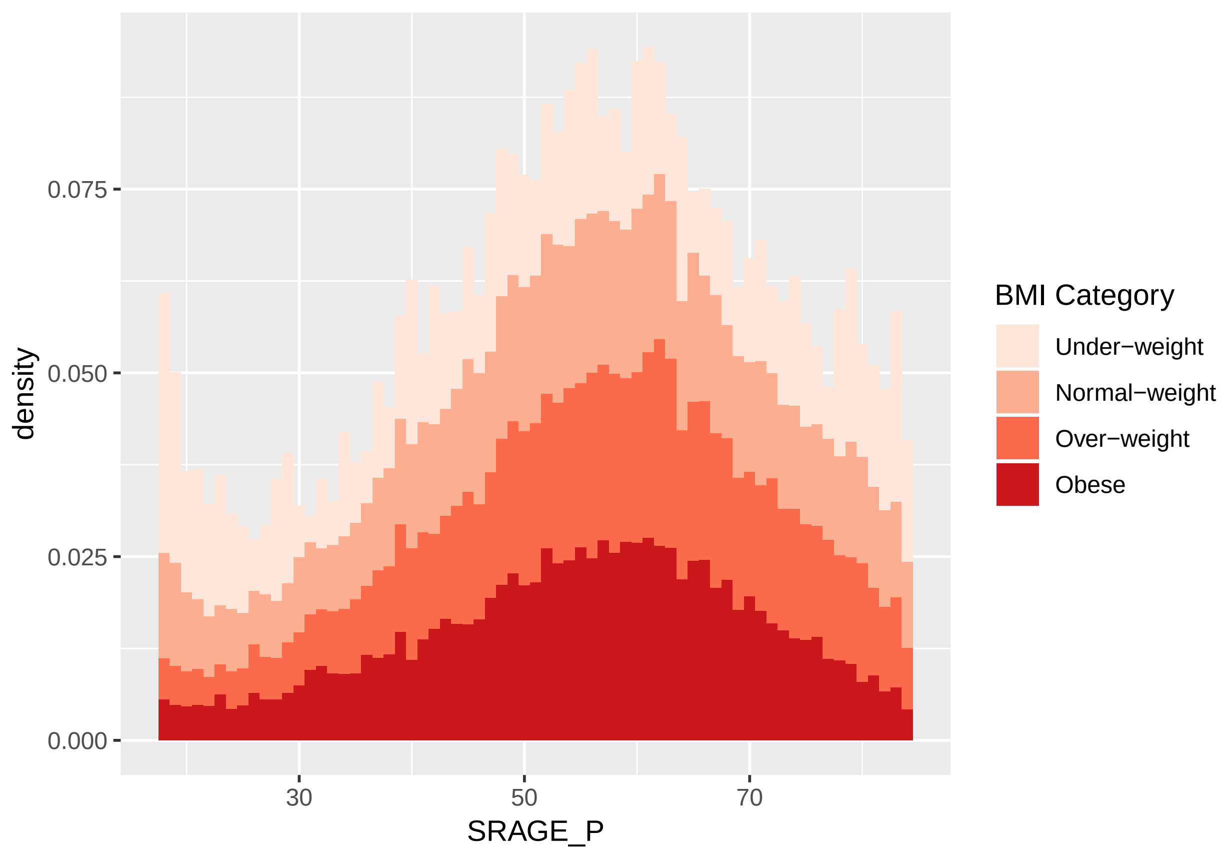
- Plot 2 - Copy, paste and adapt the code for plot 1 so that it shows density. Do this by adding aes(y = ..density..) inside the geom_histogram() function. This plot looks really strange, because we get the density within each BMI category, not within each age group!
- Plot 3 - starting from plot 1, create a faceted histogram. Use facet_grid() with the formula: RBMI ~ ..
- Plot 4 - starting from plot 2, create a faceted histogram showing density. Use facet_grid() with the formula RBMI ~ .. Plots 3 and 4 can be useful if we are interested in the frequency distribution within each BMI category.

- Plot 5 - Change the second plot to have position = "fill". This is not an accurate representation, as density calculates the proportion across category, and not across bin.
- Plot 6 - To get an accurate visualization, change Plot 5, but this time, instead of ..density.., set the y aesthetic to ..count../sum(..count..).

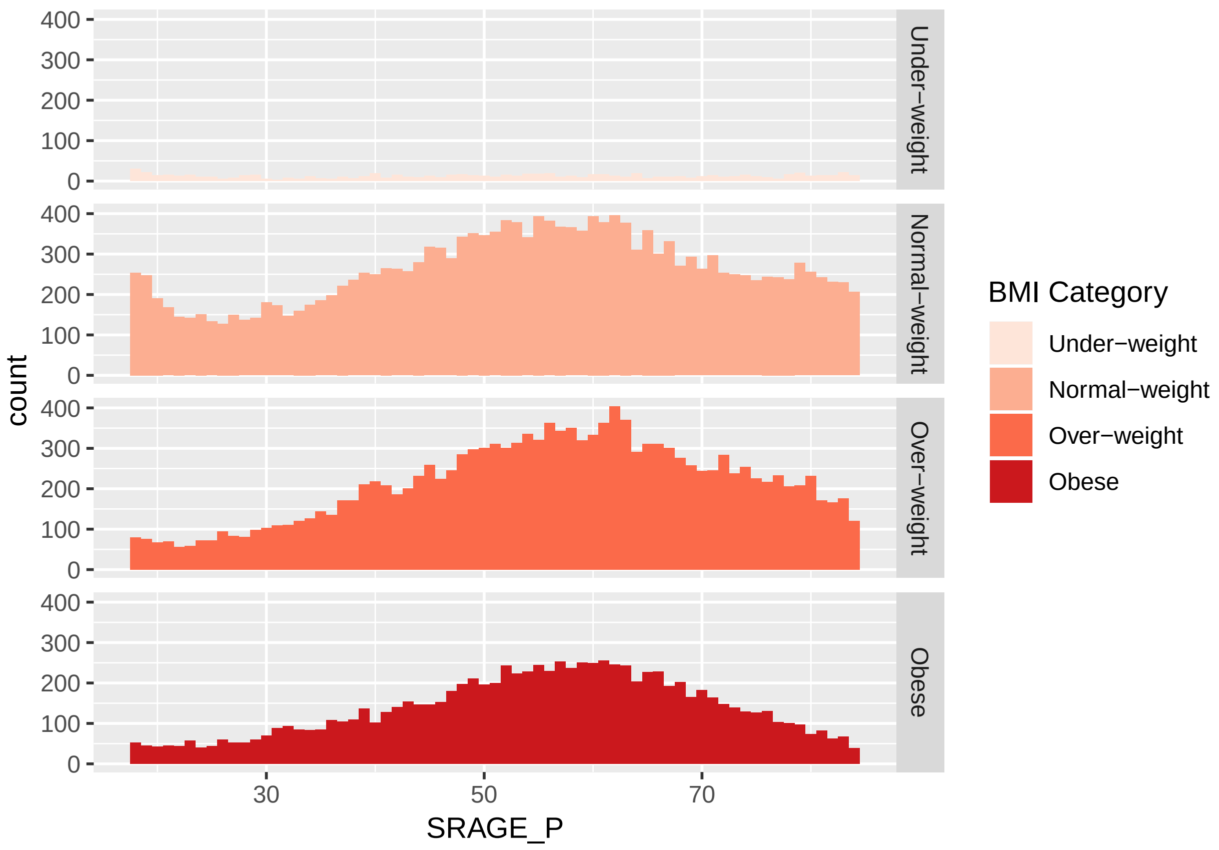
```r
# Plot 1 - Count histogram
ggplot(adult, aes (x=SRAGE_P, fill=factor(RBMI))) +
  geom_histogram(binwidth=1) +
  BMI_fill
```



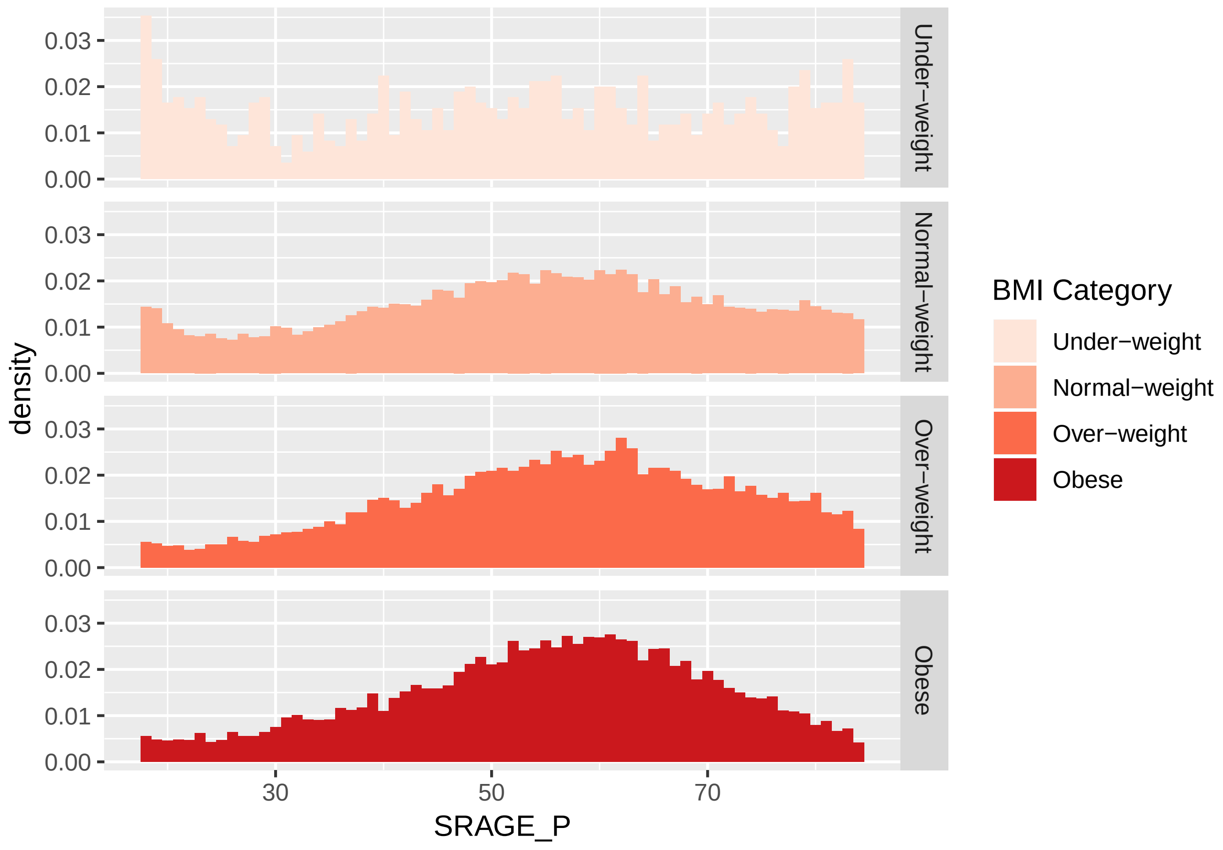
```r
# Plot 2 - Density histogram
ggplot(adult, aes (x=SRAGE_P, fill=factor(RBMI))) +
  geom_histogram(aes(y=..density..), binwidth=1) +
  BMI_fill
```



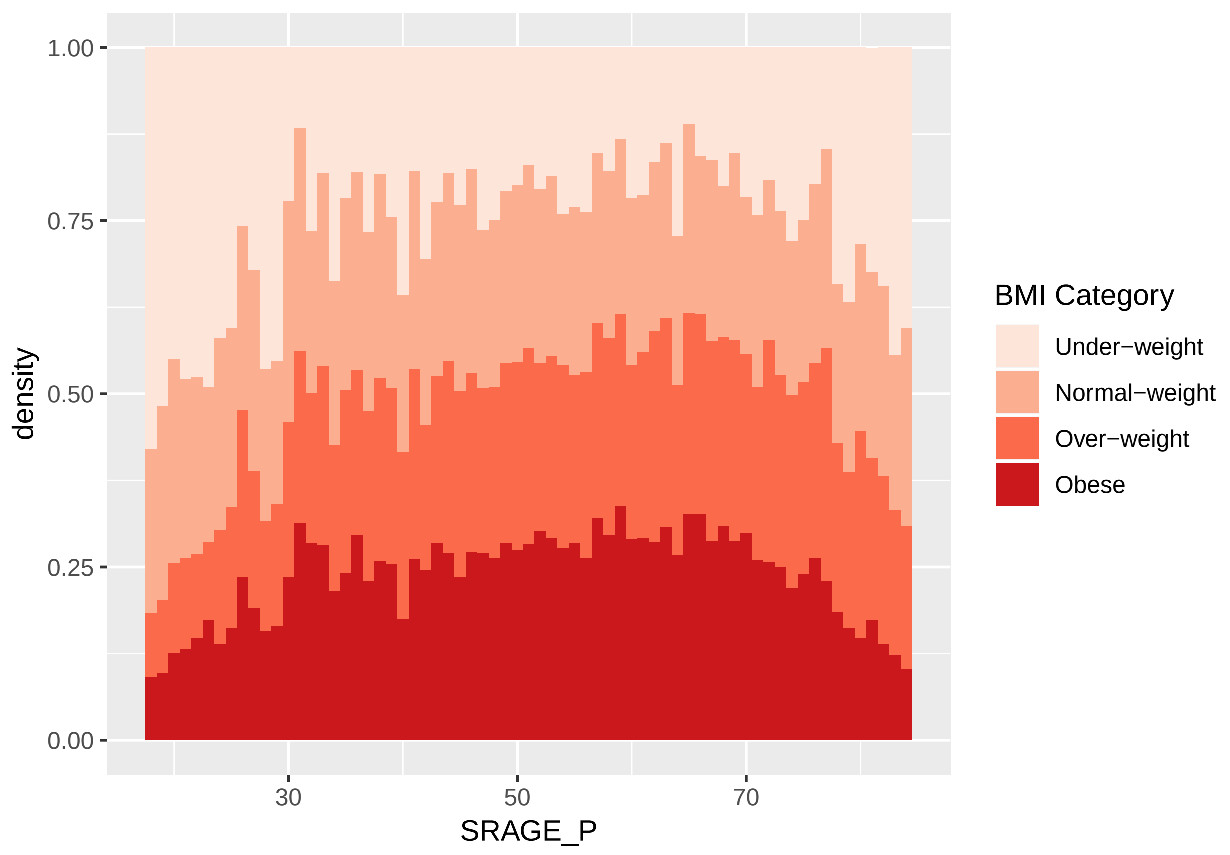
```
# Plot 3 - Faceted count histogram
ggplot(adult, aes (x=SRAGE_P, fill=factor(RBMI))) +
  geom_histogram(binwidth=1) +
  BMI_fill +
  facet_grid(RBMI ~ .)
```



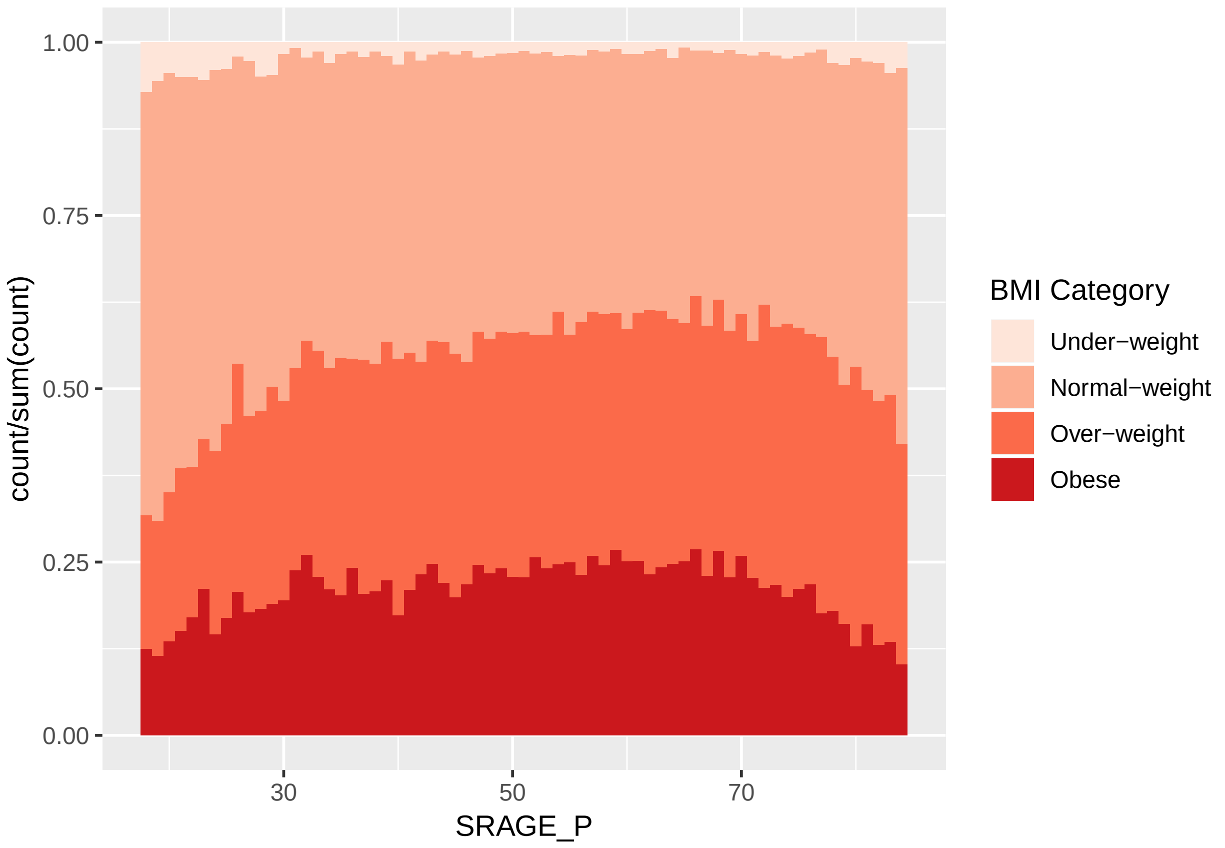
```r
# Plot 4 - Faceted density histogram
ggplot(adult, aes (x=SRAGE_P, fill=factor(RBMI))) +
  geom_histogram(aes(y=..density..), binwidth=1) +
  BMI_fill +
  facet_grid(RBMI ~ .)
```



```
# Plot 5 - Density histogram with position = "fill"
ggplot(adult, aes (x=SRAGE_P, fill=factor(RBMI))) +
  geom_histogram(aes(y=..density..), binwidth=1, position="fill") +
  BMI_fill
```



```
# Plot 6 - The accurate histogram
ggplot(adult, aes (x = SRAGE_P, fill= factor(RBMI))) +
  geom_histogram(aes(y = ..count../sum(..count..)), binwidth = 1, position="fill") +
  BMI_fill
```



1.8 Do Things Manually

In the previous exercise we looked at how to produce a frequency histogram when we have many sub-categories. The problem here is that this can't be facetted because the calculations occur on the fly inside ggplot2.

To overcome this we're going to calculate the proportions outside ggplot2. This is the beginning of our flexible script for a mosaic plot.

The dataset adult and the BMI_fill object from the previous exercise have been carried over for you. Code that tries to make the accurate frequency histogram facetted is available. You should understand these commands by now.

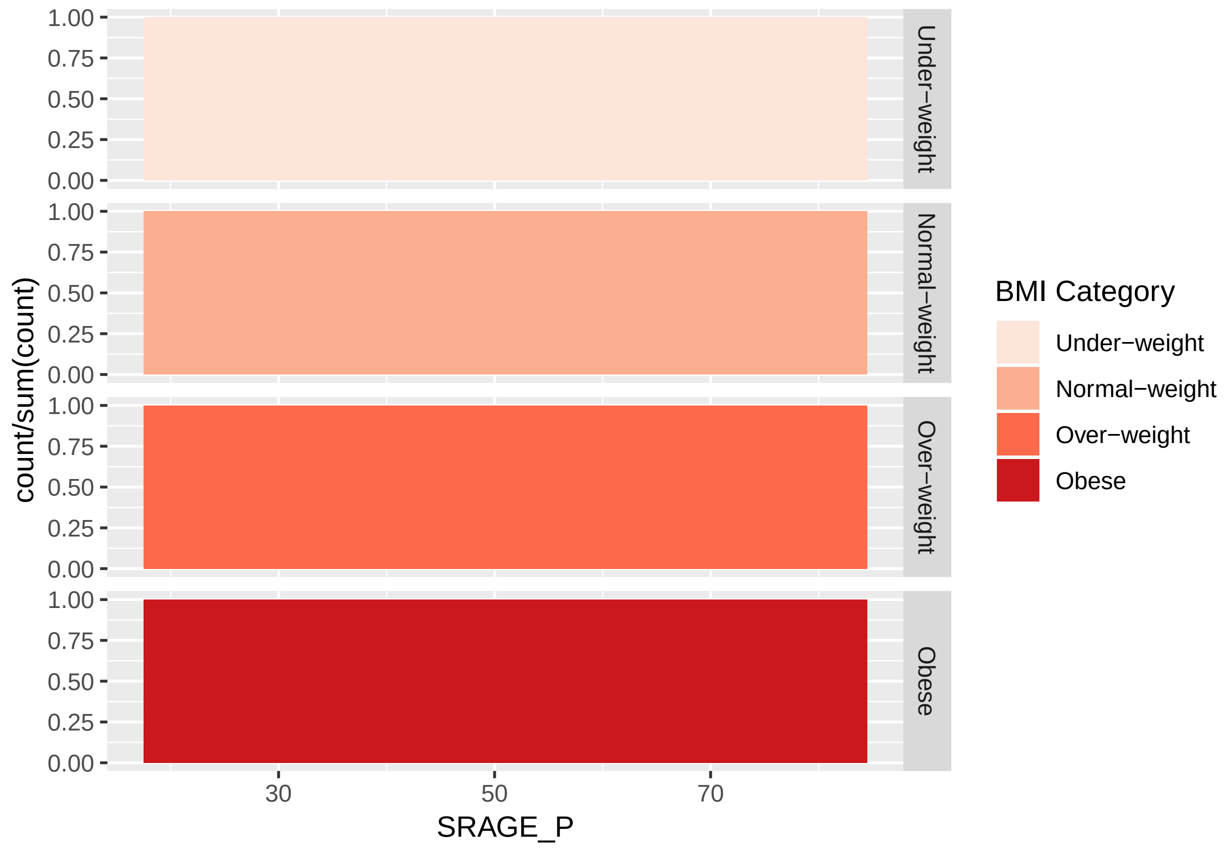
Instructions

- Use adult$RBMI and adult$SRAGE_P as arguments in table() to create a contingency table of the two variables. Save this as DF.
- Use apply() To get the frequency of each group. The first argument is DF, the second argument 2, because you want to do calculations on each column. The third argument should be function(x) x/sum(x). Store the result as DF_freq.
- Load the reshape2 package and use the melt() function on DF_freq. Store the result as DF_melted. Examine the structure of DF_freq and DF_melted if you are not familiar with this operation.

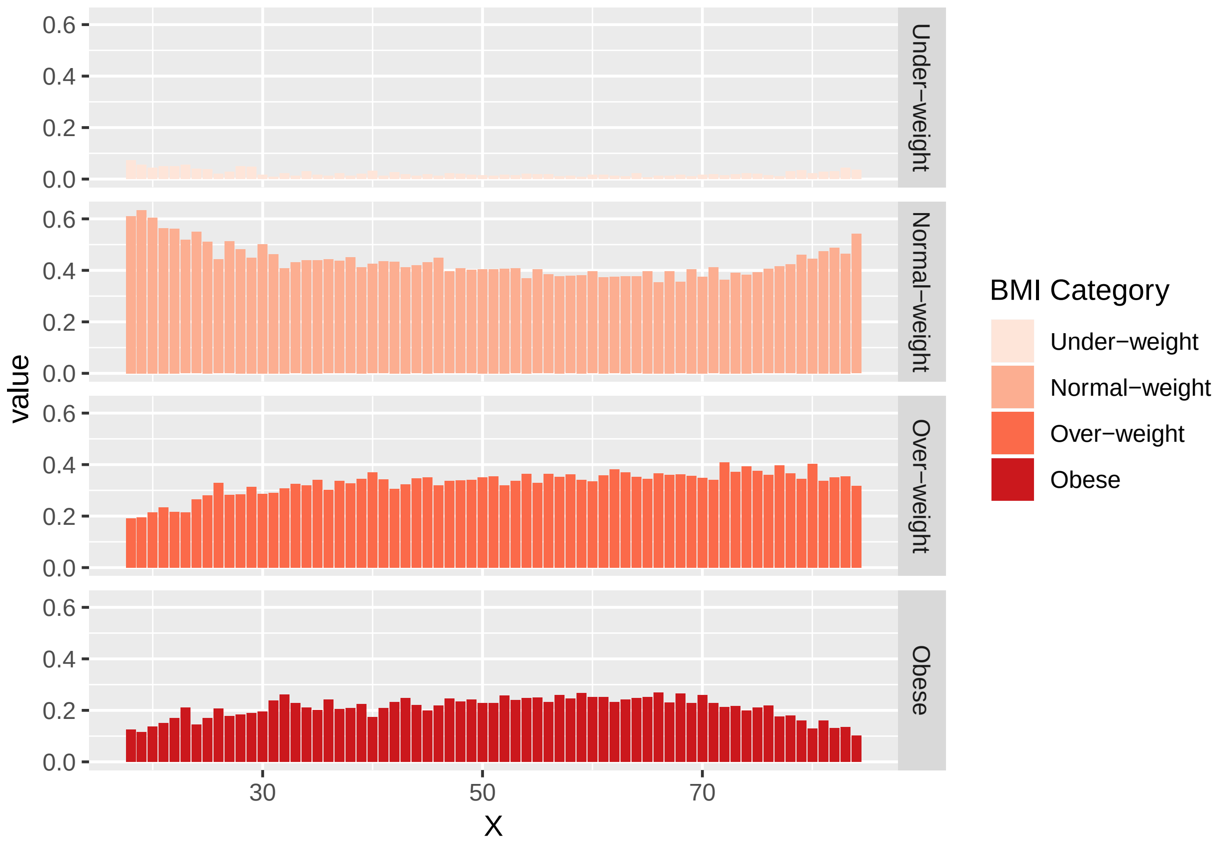
Note: Here we use reshape2 instead of the more current tidyr because reshape2::melt() allows us to work directly on a table. tidyr::gather() requires a data frame.

- Use names() to rename the variables in DF_melted to be c("FILL", "X", "value"), with the prospect of making this a generalized function later on.
- The plotting call at the end uses DF_melted. Add code to make it facetted. Use the formula FILL ~ .. Note that we use geom_col() now, this is just a short-cut to geom_bar(stat = "identity").

```r
# An attempt to facet the accurate frequency histogram from before (failed)
ggplot(adult, aes (x=SRAGE_P, fill=factor(RBMI))) +
  geom_histogram(aes(y = ..count../sum(..count..)), binwidth=1, position="fill") +
  BMI_fill +
  facet_grid(RBMI ~ .)
```



```r
# Create DF with table()
DF <- table(adult$RBMI, adult$SRAGE_P)

# Use apply on DF to get frequency of each group
DF_freq <- apply(DF, 2, function(x) x/sum(x))

# Load reshape2 and use melt on DF to create DF_melted
library(reshape2)
DF_melted <- melt(DF_freq)

# Change names of DF_melted
names(DF_melted) <- c("FILL", "X", "value")

# Add code to make this a faceted plot
ggplot(DF_melted, aes(x=X, y=value, fill=FILL)) +
  geom_col(position="stack") +
  BMI_fill +
  facet_grid(FILL ~ .) # Facets
```



1.9 Marimekko/Mosaic Plot

In the previous exercise we looked at different ways of showing the frequency distribution within each BMI category. This is all well and good, but the absolute number of each age group also has an influence on if we will consider something as over-represented or not. Here, we will proceed to change the widths of the bars to show us something about the n in each group.

This will get a bit more involved, because the aim is not to draw bars, but rather rectangles, for which we can control the widths. You may have already realized that bars are simply rectangles, but we don't have easy access to the xmin and xmax aesthetics, but in geom_rect() we do! Likewise, we also have access to ymin and ymax. So we're going to draw a box for every one of our 268 distinct groups of BMI category and age.

The clean adult dataset, as well as BMI_fill, are already available. Instead of running apply() like in the previous exercise, the contingency table has already been transformed to a data frame using as.data.frame.matrix().

Instructions

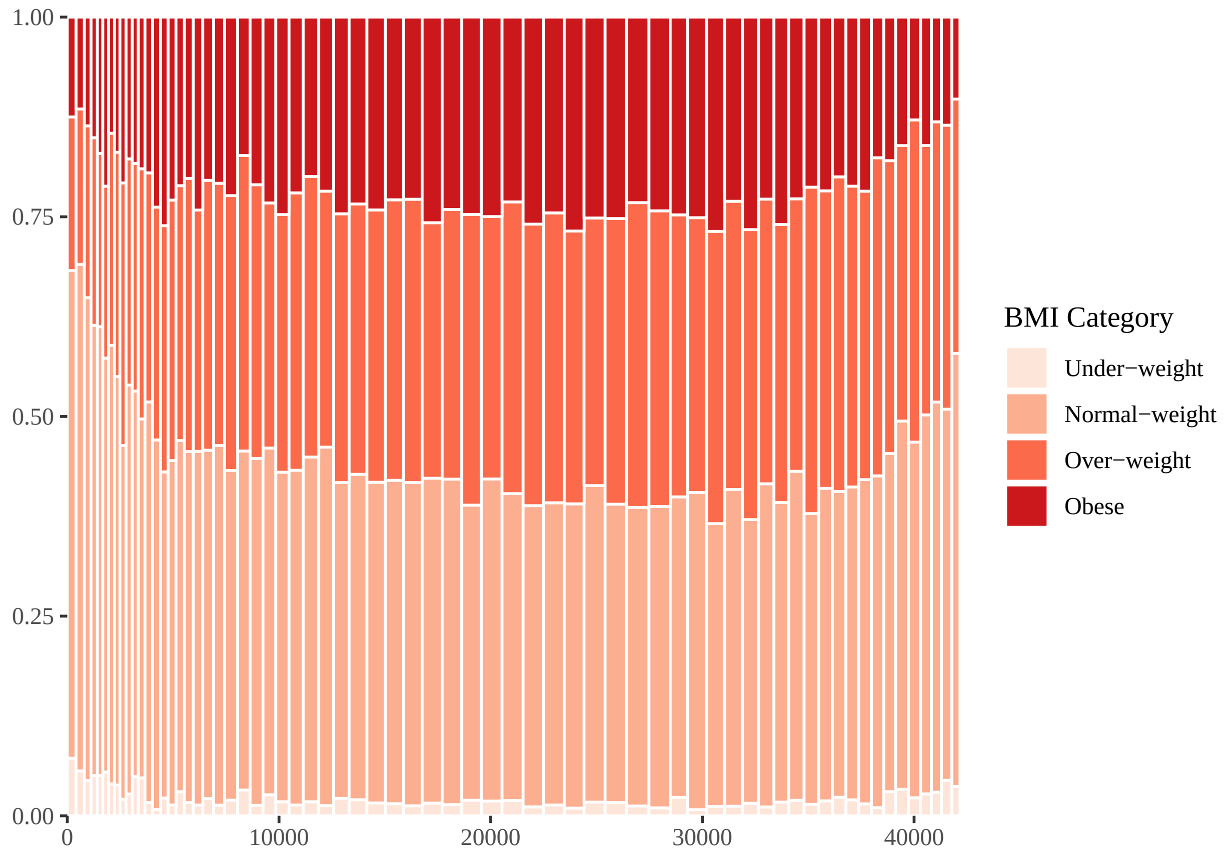
- To build the rectangle plot, we'll add several variables to DF:
- groupSum, containing the sum of each row in the DF. Use rowSums() to calculate this. groupSum represents the total number of individuals in each age group.
- xmax: the xmax value for each rectangle, calculated as cumsum(DF$groupSum)
- xmin: the xmin value for each rectangle, calculated by subtracting the groupSum column from the xmax column.
- The names of the x axis groups are stored in the row names, which is pretty bad style, so make a new variable, X, that stores the values of row.names() for DF.
- Now we are ready to melt the dataset. Load reshape2 and use melt() on DF. Specify the id.vars variables as c("X", "xmin", "xmax") and the variable.name argument as "FILL". Store the result as DF_melted.

- Have a look at the dplyr call that calculates the ymax and ymin columns of DF_melted. It first groups by X and then calculates cumulative proportions. The result is stored as DF_melted again.
- If all goes well you should see the plot in the viewer when you execute the plotting function at the bottom of the script.

```r
# The initial contingency table
DF <- as.data.frame.matrix(table(adult$SRAGE_P, adult$RBMI))

# Create groupSum, xmax and xmin columns
DF$groupSum <- rowSums(DF)
DF$xmax <- cumsum(DF$groupSum)
DF$xmin <- DF$xmax - DF$groupSum

# The groupSum column needs to be removed; don't remove this line
DF$groupSum <- NULL

# Copy row names to variable X
DF$X <- row.names(DF)

# Melt the dataset
library(reshape2)
DF_melted <- melt(DF, id.vars=c("X", "xmin", "xmax"), variable.name="FILL")

# dplyr call to calculate ymin and ymax - don't change
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
DF_melted <- DF_melted %>%
  group_by(X) %>%
  mutate(ymax = cumsum(value/sum(value)),
         ymin = ymax - value/sum(value))

# Plot rectangles - don't change
library(ggthemes)
ggplot(DF_melted, aes(ymin = ymin,
                 ymax = ymax,
                 xmin = xmin,
                 xmax = xmax,
                 fill = FILL)) +
  geom_rect(colour = "white") +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0)) +
  BMI_fill +
  theme_tufte()
```



1.10 Adding statistics

In the previous exercise we generated a plot where each individual bar was plotted separately using rectangles (shown in the viewer). This means we have access to each piece and we can apply different fill parameters.

So let's make some new parameters. To get the Pearson residuals, we'll use the chisq.test() function.

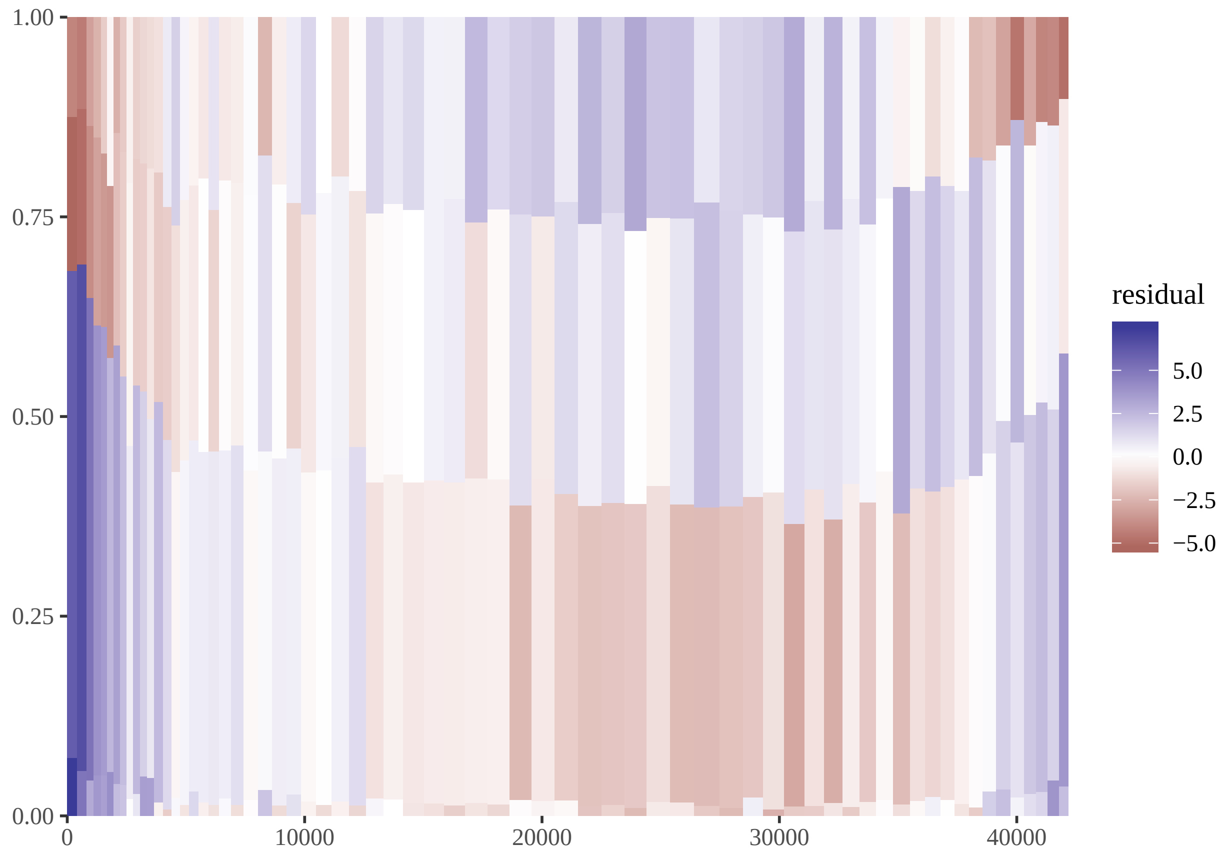
The data frames adult and DF_melted, as well as the object BMI_fill that you created throughout this chapter, are all still available. The reshape2 package is already loaded.

Instructions

- Use the adult$RBMI (corresponding to FILL) and adult$SRAGE_P (corresponding to X) columns inside the table() function that's inside the chisq.test() function. Store the result as results.
- The residuals can be accessed through results$residuals. Apply the melt() function on them with no further arguments. Store the resulting data frame as resid.
- Change the names of resid to c("FILL", "X", "residual"). This is so that we have a consistent naming convention similar to how we called our variables in the previous exercises.
- The data frame from the previous exercise, DF_melted is already available. Use the merge() function to bring the two data frames together. Store the result as DF_all.
- Adapt the code in the ggplot command to use DF_all instead of DF_melted. Also, map residual onto fill instead of FILL.

```r
# Perform chi.sq test (RBMI and SRAGE_P)
results <- chisq.test(table(adult$RBMI, adult$SRAGE_P))

# Melt results$residuals and store as resid
resid <- melt(results$residuals)
```

```r
# Change names of resid
names(resid) <- c("FILL", "X", "residual")

# merge the two datasets:
DF_all <- merge(DF_melted, resid)

# Update plot command
library(ggthemes)
p <- ggplot(DF_all, aes(ymin = ymin,
                        ymax = ymax,
                        xmin = xmin,
                        xmax = xmax,
                        fill = residual)) +
  geom_rect() +
  scale_fill_gradient2() +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0)) +
  theme_tufte()

# Resulting plot.
p
```



1.11 Adding text

Since we're not coloring according to BMI, we have to add group (and x axis) labels manually. Our goal is the plot in the viewer.

For this we'll use the label aesthetic inside geom_text(). The actual labels are found in the FILL (BMI category) and X (age) columns in the DF_all data frame. (Additional attributes have been set inside geom_text() in the exercise for you).

The labels will be added to the right (BMI category) and top (age) inner edges of the plot. (We could have also added margin text, but that is a more advanced topic that we'll encounter in the third course. This will be a suitable solution for the moment.)

The first two commands show how we got the the four positions for the y axis labels. First, we got the position of the maximum xmax values, i.e. at the very right end, stored as index. We want to calculate the half difference between each pair of ymax and ymin (e.g. (ymax - ymin)/2) at these index positions, then add this value to the ymin value. These positions are stored in the variable yposn.

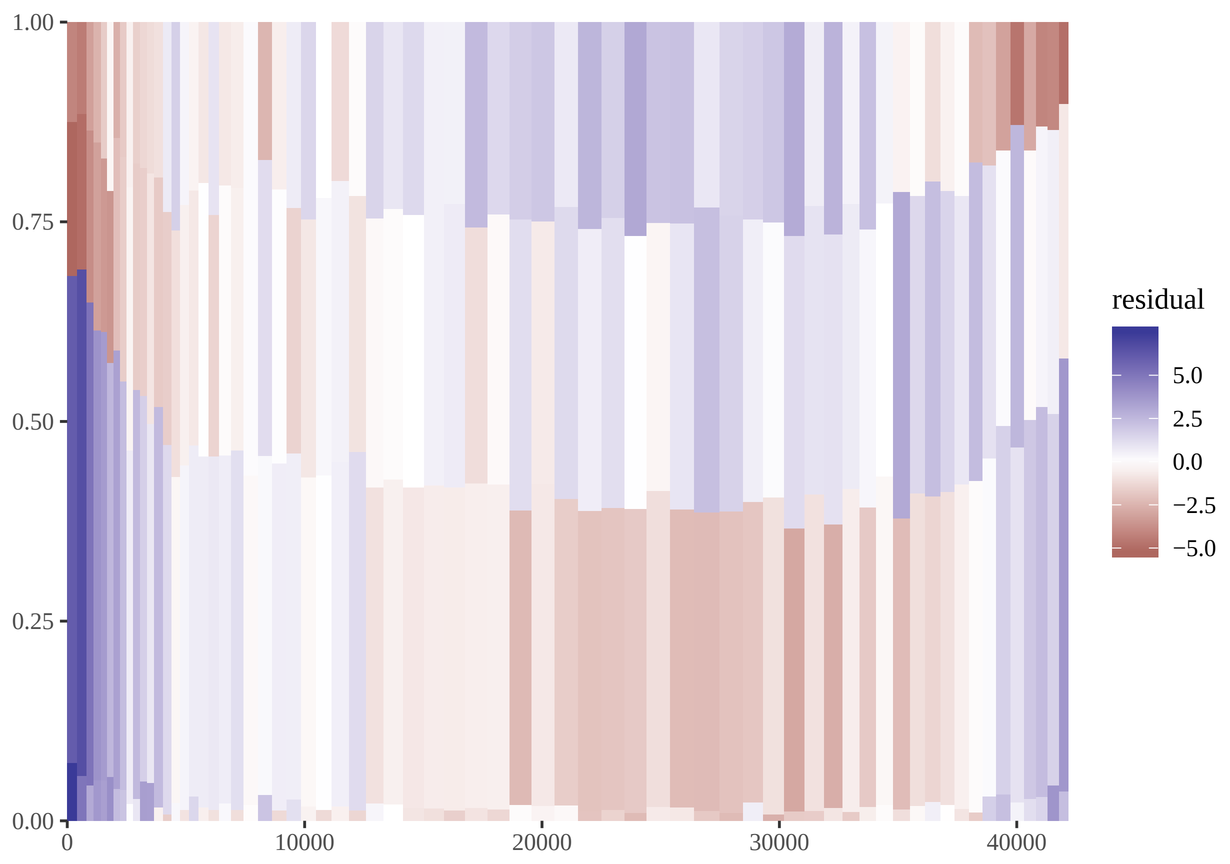
We'll begin with the plot thus far, stored as object p. In the sample code, %+% DF_all refreshes the plot's dataset with the extra columns.

Instructions

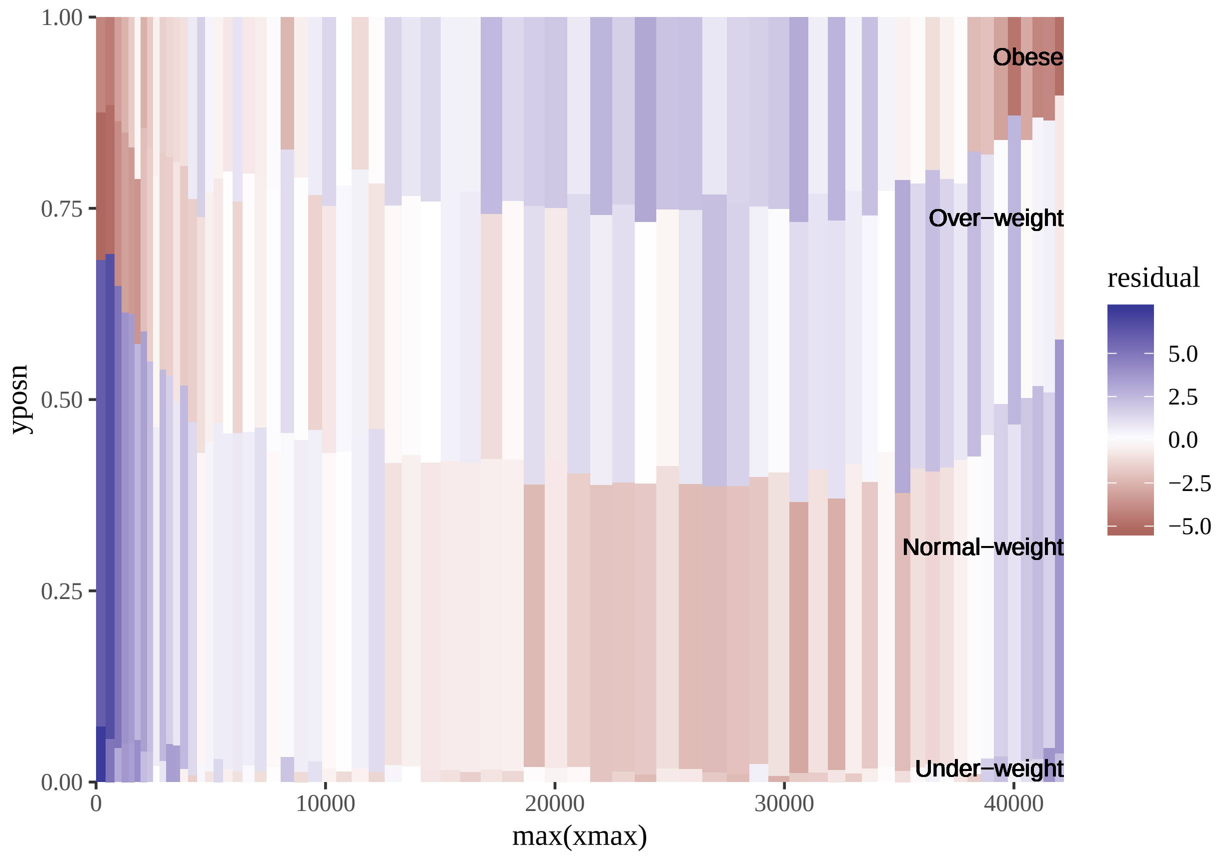
- Plot 1: In the geom_text() function, define the x, y and label aesthetics.
- Set x to max(xmax), so the labels are on the right side of the plot.
- Set the position of y to yposn.
- Set the label text to FILL.
- Plot 2: The same thing for the x axis label positions. You don't need to find an index here, since you can use the same y position for all these labels: 1.
- Calculate the half difference between each pair of xmax and xmin then add this value to xmin.
- Complete the plot command by adding the labels in the xposn to our plot, the label this time will be X, which in this case is the age.

This plot isn't perfect, but it does a pretty good job for an exploratory plot.

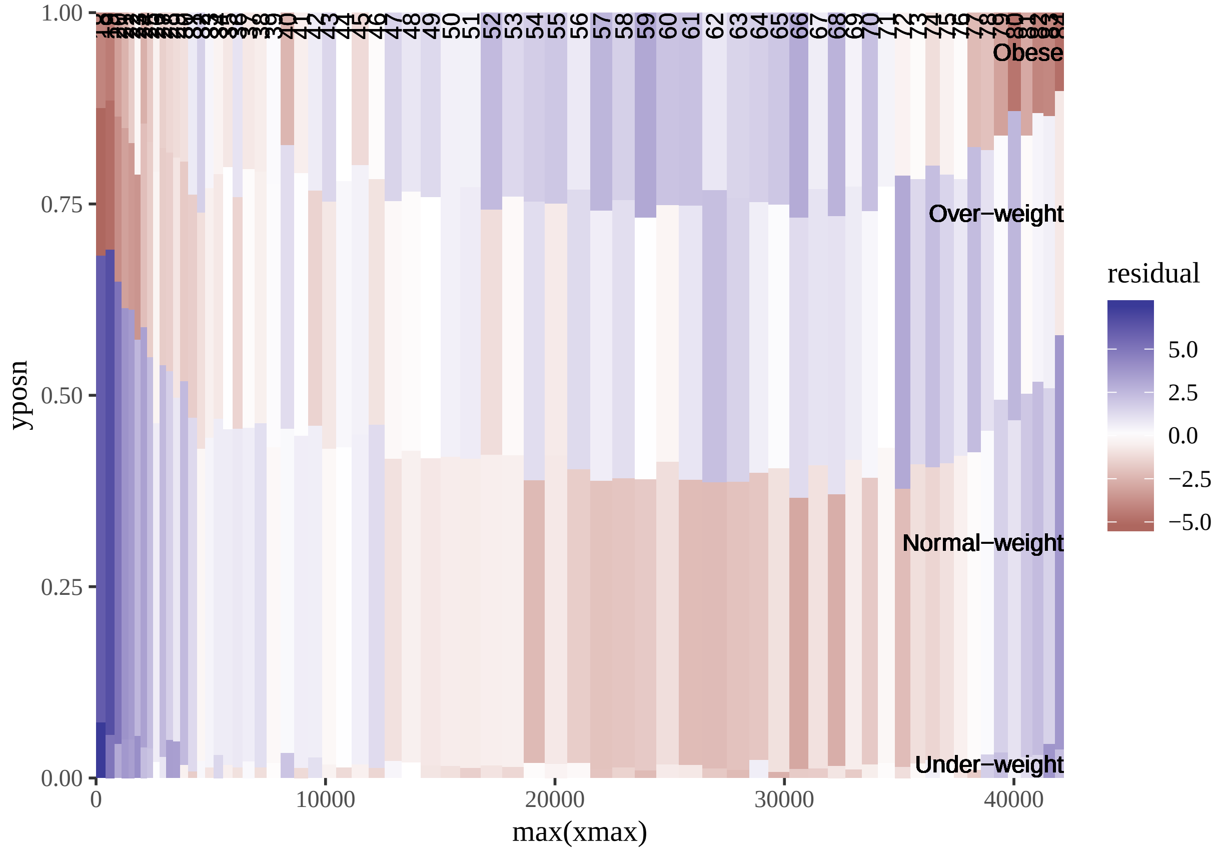
```
# Plot so far
p
```



```
# Position for labels on y axis (don't change)
index <- DF_all$xmax == max(DF_all$xmax)
DF_all$yposn <- DF_all$ymin[index] + (DF_all$ymax[index] - DF_all$ymin[index])/2

# Plot 1: geom_text for BMI (i.e. the fill axis)
p1 <- p %+% DF_all +
  geom_text(aes(x = max(xmax), y = yposn, label = FILL),
            size = 3, hjust = 1,
            show.legend  = FALSE)
p1
```



```
# Plot 2: Position for labels on x axis
DF_all$xposn <- DF_all$xmin + (DF_all$xmax - DF_all$xmin)/2

# geom_text for ages (i.e. the x axis)
p1 %+% DF_all +
  geom_text(aes(x = xposn, label = X),
            y = 1, angle = 90,
            size = 3, hjust = 1,
            show.legend = FALSE)
```



1.12 Generalizations

Now that you've done all the steps necessary to make our mosaic plot, you can wrap all the steps into a single function that we can use to examine any two variables of interest in our data frame (or in any other data frame for that matter). For example, we can use it to examine the Vocab data frame we saw earlier in this course.

You've seen all the code in our function, so there shouldn't be anything surprising there. Notice that the function takes multiple arguments, such as the data frame of interest and the variables that you want to create the mosaic plot for. None of the arguments have default values, so you'll have to specify all three if you want the mosaicGG() function to work.

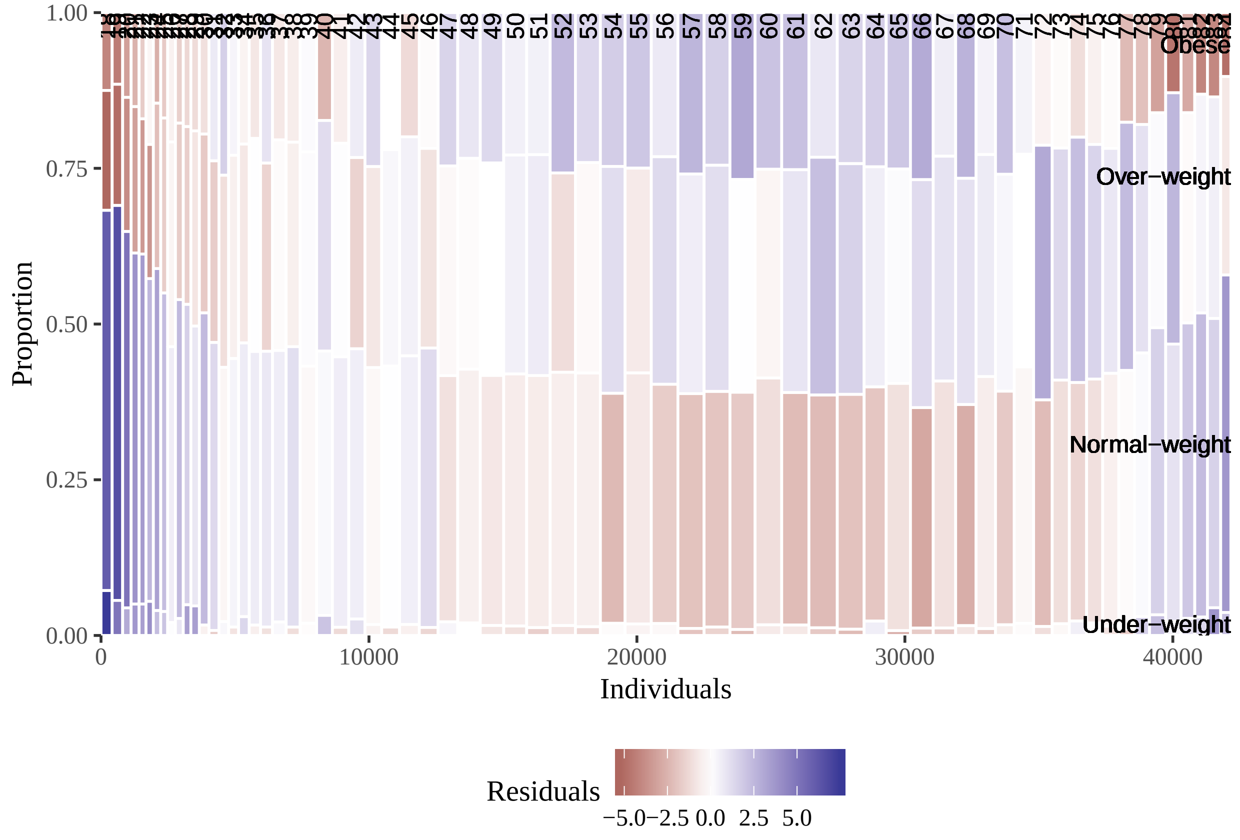
Start by going through the code and see if you understand the function's implementation.

Instructions

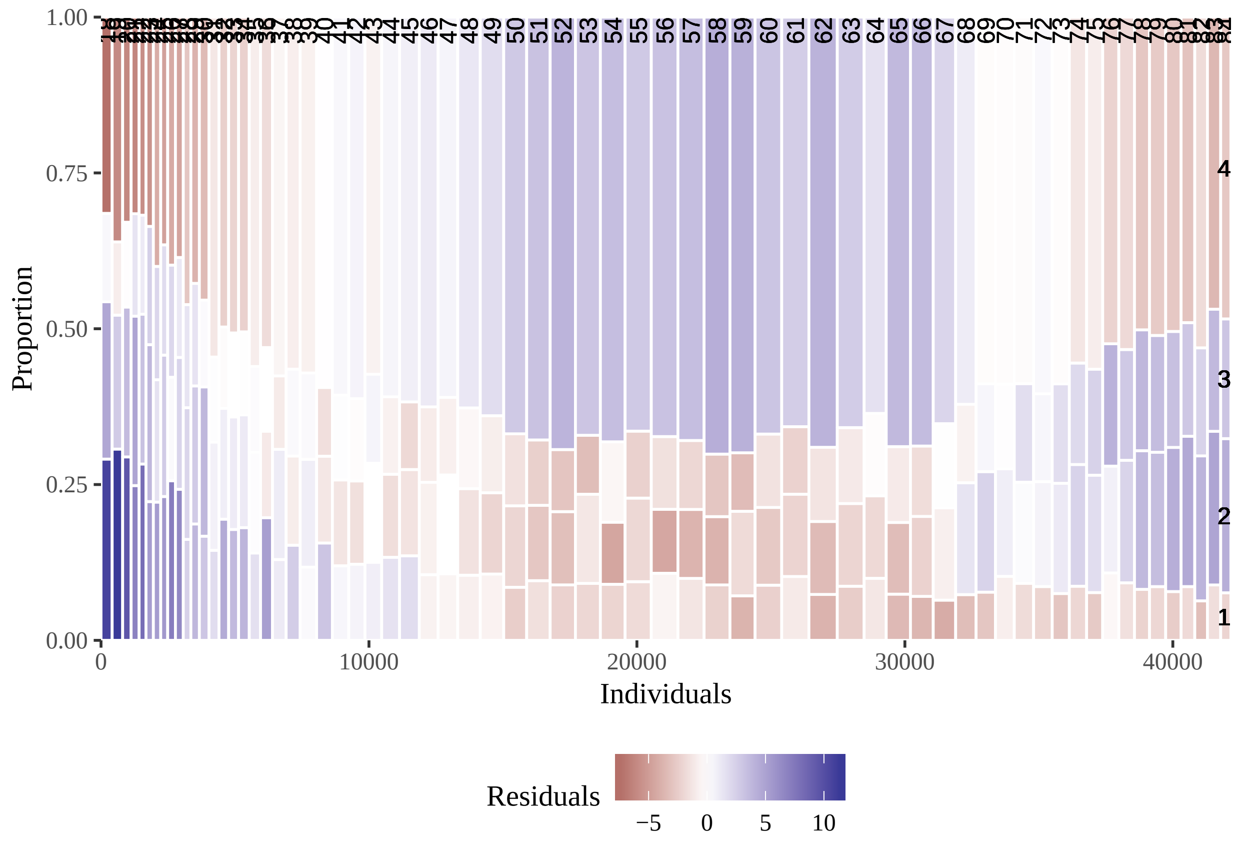
- Print mosaicGG and read its contents.
- Calling mosaicGG(adult, "SRAGE_P","RBMI") will result in the plot you've been working on so far. Try this out. This gives you a mosaic plot where BMI is described by age.
- Test out another combination of variables in the adult data frame: Poverty (POVLL) described by Age (SRAGE_P).
- Try the function on other datasets we've worked with throughout this course:
- mtcars dataset: am described by cyl
- Vocab dataset: vocabulary described by education.

```
# Load all packages
library(ggplot2)
library(reshape2)
```

```r
library(dplyr)
library(ggthemes)

# Script generalized into a function
mosaicGG <- function(data, X, FILL) {
  # Proportions in raw data
  DF <- as.data.frame.matrix(table(data[[X]], data[[FILL]]))
  DF$groupSum <- rowSums(DF)
  DF$xmax <- cumsum(DF$groupSum)
  DF$xmin <- DF$xmax - DF$groupSum
  DF$X <- row.names(DF)
  DF$groupSum <- NULL
  DF_melted <- melt(DF, id = c("X", "xmin", "xmax"), variable.name = "FILL")
  DF_melted <- DF_melted %>%
    group_by(X) %>%
    mutate(ymax = cumsum(value/sum(value)),
           ymin = ymax - value/sum(value))

  # Chi-sq test
  results <- chisq.test(table(data[[FILL]], data[[X]])) # fill and then x
  resid <- melt(results$residuals)
  names(resid) <- c("FILL", "X", "residual")

  # Merge data
  DF_all <- merge(DF_melted, resid)

  # Positions for labels
  DF_all$xposn <- DF_all$xmin + (DF_all$xmax - DF_all$xmin)/2
  index <- DF_all$xmax == max(DF_all$xmax)
  DF_all$yposn <- DF_all$ymin[index] + (DF_all$ymax[index] - DF_all$ymin[index])/2

  # Plot
  g <- ggplot(DF_all, aes(ymin = ymin,  ymax = ymax, xmin = xmin,
                          xmax = xmax, fill = residual)) +
  geom_rect(col = "white") +
  geom_text(aes(x = xposn, label = X),
            y = 1, size = 3, angle = 90, hjust = 1, show.legend = FALSE) +
  geom_text(aes(x = max(xmax),  y = yposn, label = FILL),
            size = 3, hjust = 1, show.legend = FALSE) +
  scale_fill_gradient2("Residuals") +
  scale_x_continuous("Individuals", expand = c(0,0)) +
  scale_y_continuous("Proportion", expand = c(0,0)) +
  theme_tufte() +
  theme(legend.position = "bottom")
  print(g)
}

# BMI described by age (as previously seen)
mosaicGG(adult, X = "SRAGE_P", FILL = "RBMI")
```

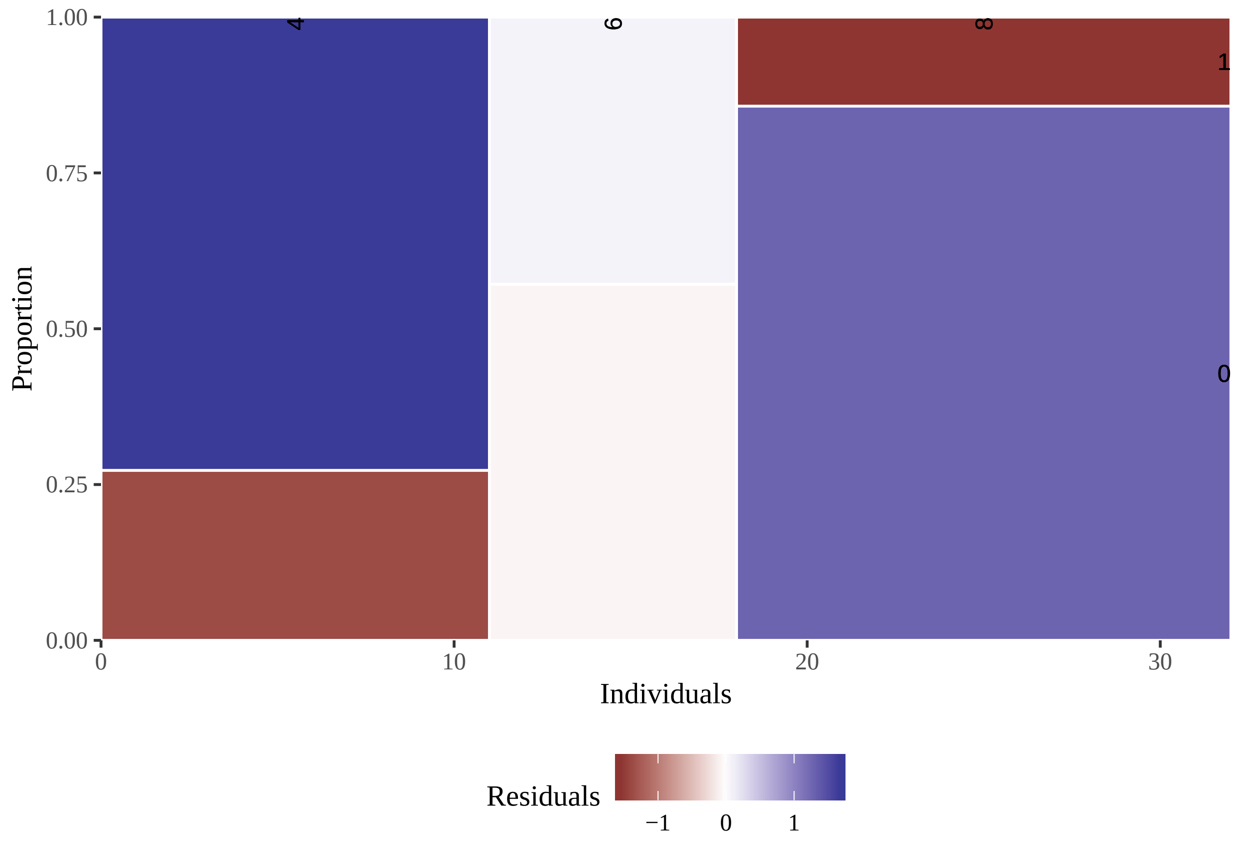


```
# Poverty described by age
mosaicGG(adult, X = "SRAGE_P", FILL = "POVLL")
```



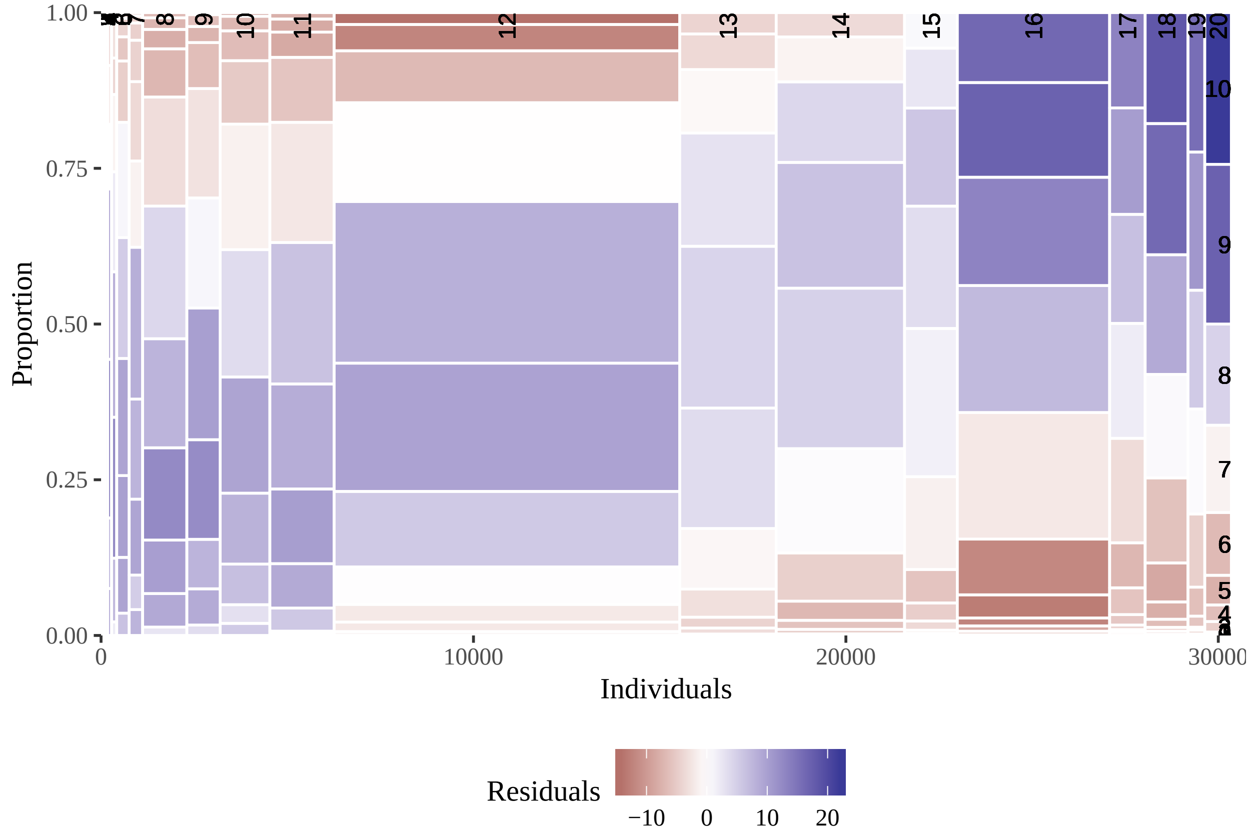
```r
# mtcars: am described by cyl
mosaicGG(mtcars, "cyl", "am")
```

```
## Warning in chisq.test(table(data[[FILL]], data[[X]])): Chi-squared
## approximation may be incorrect
```



```r
# Vocab: vocabulary described by education
library(carData)
mosaicGG(Vocab, "education", "vocabulary")
```

```
## Warning in chisq.test(table(data[[FILL]], data[[X]])): Chi-squared
## approximation may be incorrect
```



Proportion
1.00
0.75
0.50
0.25
0.00
0
10000
20000
30000
Individuals
8
9
10
11
12
13
14
15
16
17
18
19
20
10
9
8
7
6
5
4
Residuals
−10
0
10
20