R Graph Templates for Visualizing Living Income

Molly Leavens on behalf of the Sustainable Food Lab

Last updated 6/17/2021

Contents

Instructions for use

- The Sustainable Food Lab created this document to support companies and research organizations who are visualizing income data and its relationship to established Living Income Benchmarks. This document supplements tools developed by KIT to generate these graphics in Stata and Excel.
- The advantages of data cleaning, management, and visualization in R programming language are countless and include: free open source use, reproducibility and documentation of data cleaning, seamless integration with GitHub (a software/platform to track, store, and collaborate on code scripts), and flexible functions.
- While the code presented in this document is intended to be as user friendly as possible, some basic knowledge of R is necessary. The user will need to understand how to navigate the basic R interface to run the script and export desired graphics.
- The example graphs below visualize a publicly-available data set based on previous work of KIT for the Living Income Community of Practice.
- The sample data set assigns different Living Income benchmarks to different household types. No changes to the code are necessary if your data only has one benchmark for all household types or genders.

Troubleshooting

If you are encountering challenges and/or errors when running this code, try these five initial troubleshooting:

1. Check your 'working directory' and ensure it is the same file destination as where you have saved your data file CSV. A working directory error would occur when trying to upload the data file in the setup.
2. Ensure all the required libraries are installed on your computer. If you have never used the tidyverse, knitr, or scales libraries, you will need to individually install them first with the function install.packages("LIBRARY NAME").
3. Ensure that you format all numerical variables without any commas or currency notations and any blank values are assigned NA.
4. Check for extreme data outliers. Very large, small, or negative income calculations will result in skewed visualizations and could even lead to errors when running the code.
5. Copy the exact error message from R into Google as someone has almost always previously asked about the error message on the forum StackOverflow.

If you have further questions about this document, please reach out to the Living Income Community of Practice at livingincome@isealalliance.org.

Setup

```
#### Before continuing any further, load the R libraries that have the
# necessary functions for this analysis.

# If you have not previously used these libraries,
# you will need to instaLl them with the function install.packages("LIBRARY NAME").
# For example: install.packages(tidyverse)
library(tidyverse)
library(knitr)
library(scales)
```

User-entered data

This section allows the user to enter the information specific to their dataset. The user should replace all the wording in ALL CAPS in the following code chunk.

For the graph code to function, it is imperative that you write the variable names below exactly as they appear in your data's CSV file.

When replacing the capitalized sections, retain the quotation marks. In other words, write your replacement within the quotation marks as this signals to R that you are not referencing an object that already exists.

For an example of how to fill in the following code chunk, see the following code chunk for the example dataset.

```
######################## Import data set ########################

# If you have troubles with the import,
# ensure the original data file is stored in your "working directory"
# (most often the same folder you have saved your working code script)

# If your original data file is saved as an Excel, export it (from Excel,
  # 'save as') as a CSV file before importing it here.
# Ensure the top row of the Excel/CSV is only column names.

df <- read_csv("NAME OF DATA FILE.csv")

## Make all column names syntactically valid
# In R, column names must only consist of letters, numbers, periods,
  # and underscores. They cannot start with a number nor include spaces.
# If your excel has any column names that do not comply with this syntax,
  # the below code will alter the names accordingly.

names(df) <- make.names(names(df), unique=TRUE)

######################## Replace variable names ########################
# In the below section, replace the words in all caps with the column
  # name present in your dataframe.
# Note: Ensure that you write the column names exactly as they appear in your
  # data's current dataframe. The make.names() function executed above may have
  # altered some of the column names to enable R compatibility.
  # Run names(df) for a list of your dataset's current column names.
```

```
# Income from main crop
df$income_main_crop <- df$"VARIABLE NAME"
# Income from all other sources
df$other_income <- df$"VARIABLE NAME"
# Total HH income
  # If you do not already have a named variable for total income, you can
  # instead assign the following line of code to total_hh_income:
  # df$income_main_crop + df$other_income
total_hh_income <- "TOTAL HH INCOME"
# Groups to compare incomes between
  # e.g: gender, household size, farm type, etc.
df$grouping <- df$"VARIABLE NAME"

######################## Replace graph labels ######################
main_crop <- "NAME OF MAIN CROP" # e.g: "cocoa"
currency <- "CURRENCY OF INCOME DATA" # e.g: "USD"
```

Example dataset

```
## Import data set
df <- read_csv("KIT_Sample_Data.csv")

## Replace variable names
# Main crop
df$income_main_crop <- df$total_cocoa_income_2018
# Other income
# df$other_income <- df$
# Total income
df$total_hh_income <- df$total_hh_income_2018

# Replace graph labels
main_crop <-"cocoa"
currency <- "USD"
```

Graph aesthetics changes

The following three code chunks allow the user to define graph colors, fonts and bar placement, respectively. These functionalities are helpful in matching the graphs to company/organization specific reports and presentations.

Colors

If you would like to leave the graph colors as they are in the example graphs, simply run the following code chunk without making any alterations. Note: As the code chunk for each graph template references the color names assigned in the below color code chunk, **the graph code will result in an error if you do not run the color chunk first**. You only need to run the color chunk once per session as R will save the assigned colors in the working environment.

```
# You can communicate colors with R in two ways: through one of the 657 color
# names build into R, or through a hex color code.

# To see the names of all 657 colors built into R, run the colors() formula in the command line.
# The below links to a PDF file with a color sample for all 657 built in colors
```

```r
# http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

# You can replace any of the below colors with a new color name or hex color code.

## Bar graph sections
# Gap to the mean
gap_color <- "#ed3833"
# Other income
other_color <- "#b3dceb"
# Income from main crop
main_color <- "#b3b3fa"
# Value of crops consumed at home
home_color <- "#fad8b3"

## Share of observations below the Living Income Benchmark
# Note: these bars all appear the same color
share_color <- "#ed3833"

## Distributional plots

## By household type
# Group 1
color_1 <- "#b6dce8"
# Group 2
color_2 <- "#c2c1fa"
# Group 3
color_3 <- "#c1dfc1"

color_4 <- "magenta"

test_pallet <- c(color_1, color_2, color_3, color_4)

scale_color_drsimonj <- function(palette = test_pallet, discrete = TRUE, reverse = FALSE, ...) {
  pal <- drsimonj_pal(palette = palette, reverse = reverse)

  if (discrete) {
    discrete_scale("colour", paste0("drsimonj_", palette), palette = pal, ...)
  } else {
    scale_color_gradientn(colours = pal(256), ...)
  }
}

## With mean and median
# Distribution curve
curve_color <- "#b6dce8"
# Income Mean
mean_color <- "blue"
# Income Median
median_color <-  "green"
# Living Income Benchmark
benchmark_color <- "red"

################## Font color
```

```
# To change the color of the font for all graph axis and labels, replace the color
# in the quotation marks, then run the following two lines of code.
font_color <- "black"
theme_set(theme_get() + theme(text = element_text(color = font_color)))
```

Fonts

The following font code chunk is **optional** and allows you to update the font for all ensuing graphs and data tables.

If you would like the graph and data table font to remain as Arial, the R default font, do not run the below code. If you would like to update the font, follow the steps outlined below and ensure to replace the single line of all caps.

```
# If you have previously imported fonts in R, you can skip steps 1 and 2.

# Step 1: Load extrafont library
library(extrafont)

# Step 2: Import fonts already on your system
# For guidance on uploading a custom font, view [this article] (https://r-coder.com/custom-fonts-r/)
font_import()
# After running the font_import() function, you will likely be prompted with
  # the following message: "Importing fonts may take a few minutes, depending
  # on the number of fonts and the speed of the system. Continue? [y/n]"
  # Type Y, then enter and the function should run.

# Step 3: Output table with all available fonts.
fonttable()

# Step 4: Select your desired font
# Ensure you paste the 'FontName' as listed in the front table printed in step 3
  # There should not be any spaces in the same pasted below
font <- "PASTE THE NAME OF DESIRED FONT HERE." # e.g.: "CourierNewPSMT"

# Step 5: Set default to your selected font.
theme_set(theme_get() + theme(text = element_text(family = font)))

# When running your graph code, if you get the following error: "polygon edge not found"
```

Bar Placement (variable releveling)

The below code chunk is **optional**. It allows you to define the order in which the groups - gender, household size, or other grouping variable - appear from left to right on the bar graphs.

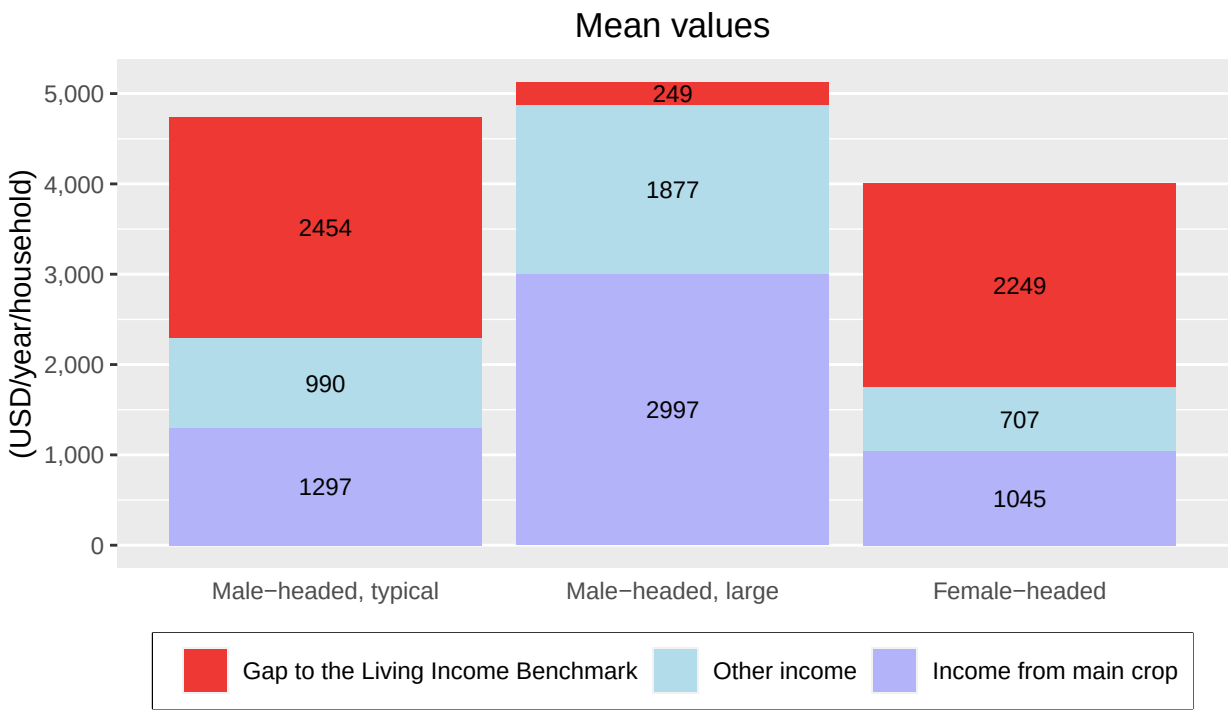
```
# In the example dataset, there are three groups that income is assessed across:
  # Male-headed, typical; Male-headed, large; and Female-headed.
# The below code assigns Male-headed, typical as the far left bar on graphs,
  # Male-headed, large in the middle, and Female-headed on the far right.

# To customize this chunk of code to your dataset, replace the example group
  # names with the group names in your dataset - leaving the quotation marks and
  # commas as they are. The order of the list below determines the order of the
  # bars in the ensuing bar graph.
```

```r
df <- df %>%
      mutate(grouping = factor(grouping,
                               levels = c("Male-headed, typical",
                                          "Male-headed, large",
                                          "Female-headed")))
```

The gap of the mean income to the Living Income Benchmark - bar chart

```
## The first section of this code summarizes and formats the data to be graph-ready
df %>%
# Group by household type
  group_by(grouping) %>%
# For each household type, summarize the mean gap to the living income,
# the mean other income, and the mean main_crop income
  summarise(Gap = mean(benchmark - total_hh_income),
            Other = mean(total_hh_income - income_main_crop),
            main_crop = mean(income_main_crop)) %>%
# Gather each income components into one column so the data is in 'long' format
  gather(key = "Component", value = "Income", Gap:main_crop) %>%
# Re-level the income factors for the order you want them stacked on the graph
  mutate(Component = factor(Component,
                            levels = c("Gap", "Other", "main_crop"))) %>%
# Generate ggplot graph for income by groupings and income component
  ggplot(aes(y = Income, x = grouping, fill = Component)) +
# Assign graph as stacked bar chart
  geom_bar(position = "stack", stat = "identity") +
# Label the graph title, axis, and caption
  labs(title = "Mean values",
       y = paste("(", currency, "/year/household)", sep = ""),
       x = "",
# Add caption with observation numbers for each household type
       caption = paste("Based on: \n",
                   paste(names(table(df$grouping)),
                       ":",
                       as.numeric(table(df$grouping)),
                       "observations \n ", collapse = ''), collapse = '')) +
# Label the legend and assign custom colors
  scale_fill_manual(values=c(gap_color, other_color, main_color),
                    breaks=c("Gap", "Other", "main_crop"),
                    labels=c("Gap to the Living Income Benchmark",
                             "Other income",
                             "Income from main crop")) +
# Format y-axis labels with a comma
  scale_y_continuous(labels = comma) +
# Remove x-axis grid lines and tick marks
  theme(panel.grid.major.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.x = element_blank(),
# Center plot title
        plot.title = element_text(hjust = 0.5),
# Move the legend to the bottom of the graph
        legend.position="bottom",
# Remove legend title
        legend.title = element_blank(),
# Put a box around the legend
        legend.box.background = element_rect(),
# Move caption to desired location
        plot.caption = element_text(hjust = 0)) +
```

```
# Add incomes to prospective graph components
  geom_text(aes(label = round(Income)),
            position = position_stack(vjust = 0.5),
            size = 3)
```

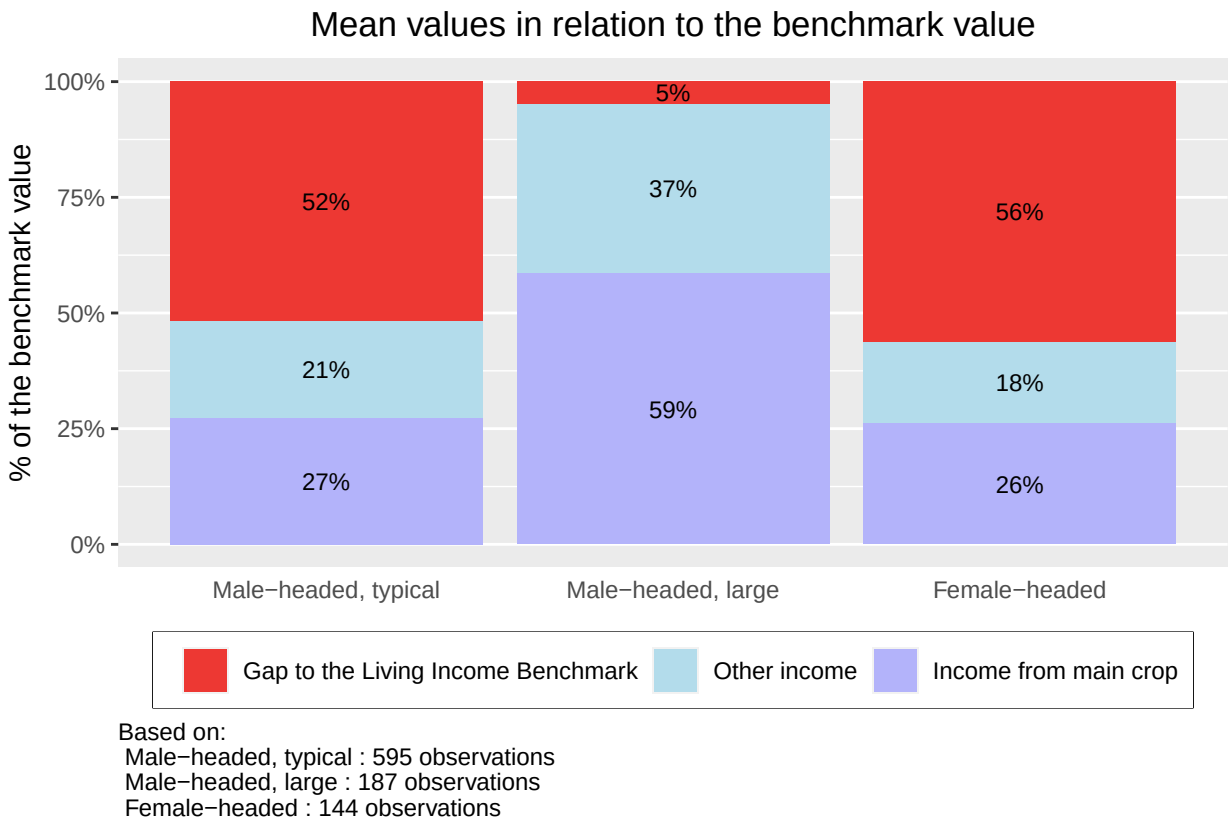


Based on:
Male−headed, typical : 595 observations
Male−headed, large : 187 observations
Female−headed : 144 observations

The gap of the relative mean income to the Living Income Benchmark - bar chart

```
## The first section of this code summarizes and formats the data to be graph-ready
df %>%
# Group by household type
  group_by(grouping) %>%
# For each household type, summarize the mean gap to the living income,
# the mean other income, and the mean main_crop income.
# Calculate all as percentages of the Living income benchmark - the total of the income and gap
  summarise(Gap = mean((benchmark - total_hh_income)/benchmark),
            Other = mean((total_hh_income - income_main_crop)/benchmark),
            main_crop = mean(income_main_crop/benchmark)) %>%
# Gather each income components into one column so the data is in 'long' format
  gather(key = "Component", value = "Income", Gap:main_crop) %>%
# Re-level the income factors for the order you want them stacked on the graph
  mutate(Component = factor(Component,
                            levels = c("Gap", "Other", "main_crop"))) %>%
# Generate ggplot graph for income by groupings and income component
  ggplot(aes(y = Income, x = grouping, fill = Component)) +
# Assign graph as stacked bar chart
  geom_bar(position = "stack", stat = "identity") +
# Label the graph title, axis, and caption
  labs(title = "Mean values in relation to the benchmark value",
       y = "% of the benchmark value",
       x = " ",
# Add caption with observation numbers for each household type
       caption = paste("Based on: \n",
                   paste(names(table(df$grouping)),
                         ":",
                         as.numeric(table(df$grouping)),
                         "observations \n ", collapse = ''), collapse = '')) +
# Label the legend
  scale_fill_manual(values=c(gap_color, other_color, main_color),
                    breaks=c("Gap",
                             "Other",
                             "main_crop"),
                    labels=c("Gap to the Living Income Benchmark",
                             "Other income",
                             "Income from main crop")) +
# Format y-axis labels with a comma
  scale_y_continuous(labels = percent) +
# Remove x-axis grid lines and tick marks
  theme(panel.grid.major.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.x = element_blank(),
# Center plot title
        plot.title = element_text(hjust = 0.5),
# Move the legend to the bottom of the graph
        legend.position="bottom",
# Remove legend title
        legend.title = element_blank(),
# Put a box around the legend
```

```
        legend.box.background = element_rect(),
# Move caption to desired location
        plot.caption = element_text(hjust = 0)) +
# Add incomes to prospective graph components
  geom_text(aes(label = label_percent(accuracy = 1L)(Income)),
            position = position_stack(vjust = 0.5),
            size = 3)
```

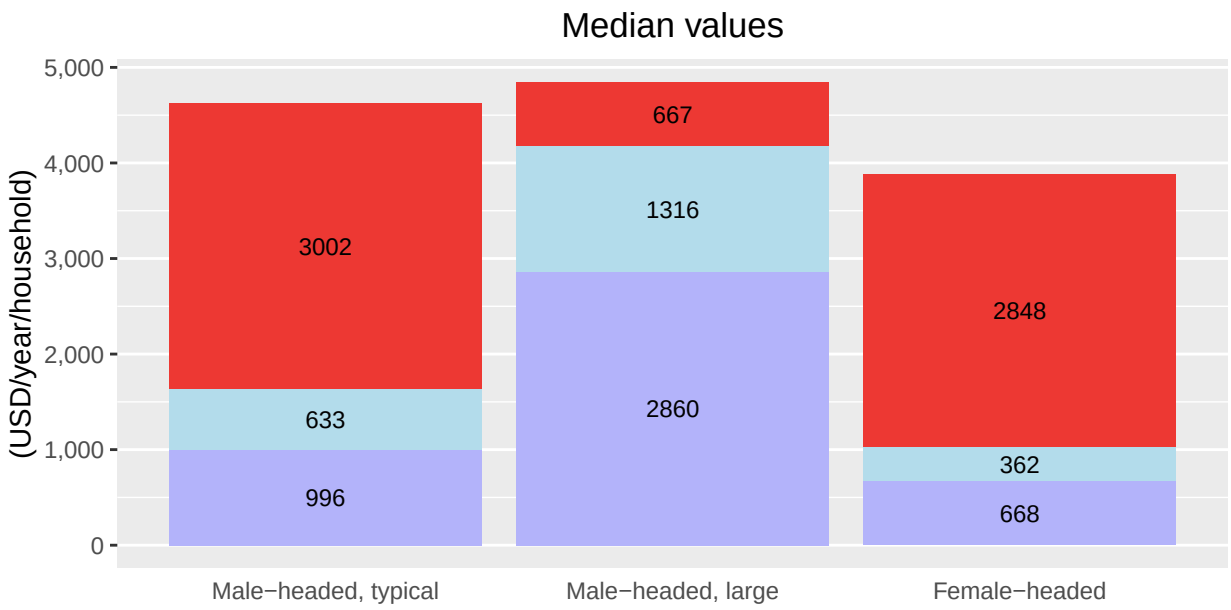


Based on:
Male−headed, typical : 595 observations
Male−headed, large : 187 observations
Female−headed : 144 observations

The gap of the absolute median income to the Living Income Benchmark - bar chart

```
## The first section of this code summarizes and formats the data to be graph-ready
df %>%
# Group by household type
  group_by(grouping) %>%
# For each grouping, summarize the median gap to the living income,
# the median other income, and the median main_crop income.
  summarise(Gap = median(benchmark - total_hh_income),
            Other = median(total_hh_income - income_main_crop),
            main_crop = median(income_main_crop)) %>%
# Gather each income components into one column so the data is in 'long' format
  gather(key = "Component", value = "Income", Gap:main_crop) %>%
# Re-level the income factors for the order you want them stacked on the graph
  mutate(Component = factor(Component,
                            levels = c("Gap", "Other", "main_crop"))) %>%
# Generate ggplot graph for income by grouping and income component
  ggplot(aes(y = Income, x = grouping, fill = Component)) +
# Assign graph as stacked bar chart
  geom_bar(position = "stack", stat = "identity") +
# Label the graph title, axis, and caption
  labs(title = "Median values",
       y = paste("(", currency, "/year/household)", sep = ""),
       x = "",
# Add caption with observation numbers for each household type
       caption = paste("Based on: \n",
                   paste(names(table(df$grouping)),
                       ":",
                       as.numeric(table(df$grouping)),
                       "observations \n ", collapse = ''), collapse = '')) +
# Label the legend and assign custom colors
  scale_fill_manual(values=c(gap_color, other_color, main_color),
                    breaks=c("Gap", "Other", "main_crop"),
                    labels=c("Gap to the Living Income Benchmark",
                             "Other income",
                             "Income from main crop")) +
# Format y-axis labels with a comma
  scale_y_continuous(labels = comma) +
# Remove x-axis grid lines and tick marks
  theme(panel.grid.major.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.x = element_blank(),
# Center plot title
        plot.title = element_text(hjust = 0.5),
# Move the legend to the bottom of the graph
        legend.position="bottom",
# Remove legend title
        legend.title = element_blank(),
# Put a box around the legend
        legend.box.background = element_rect(),
# Move caption to desired location
        plot.caption = element_text(hjust = 0)) +
```

```
# Add incomes to prospective graph components
  geom_text(aes(label = round(Income)),
            position = position_stack(vjust = 0.5),
            size = 3)
```

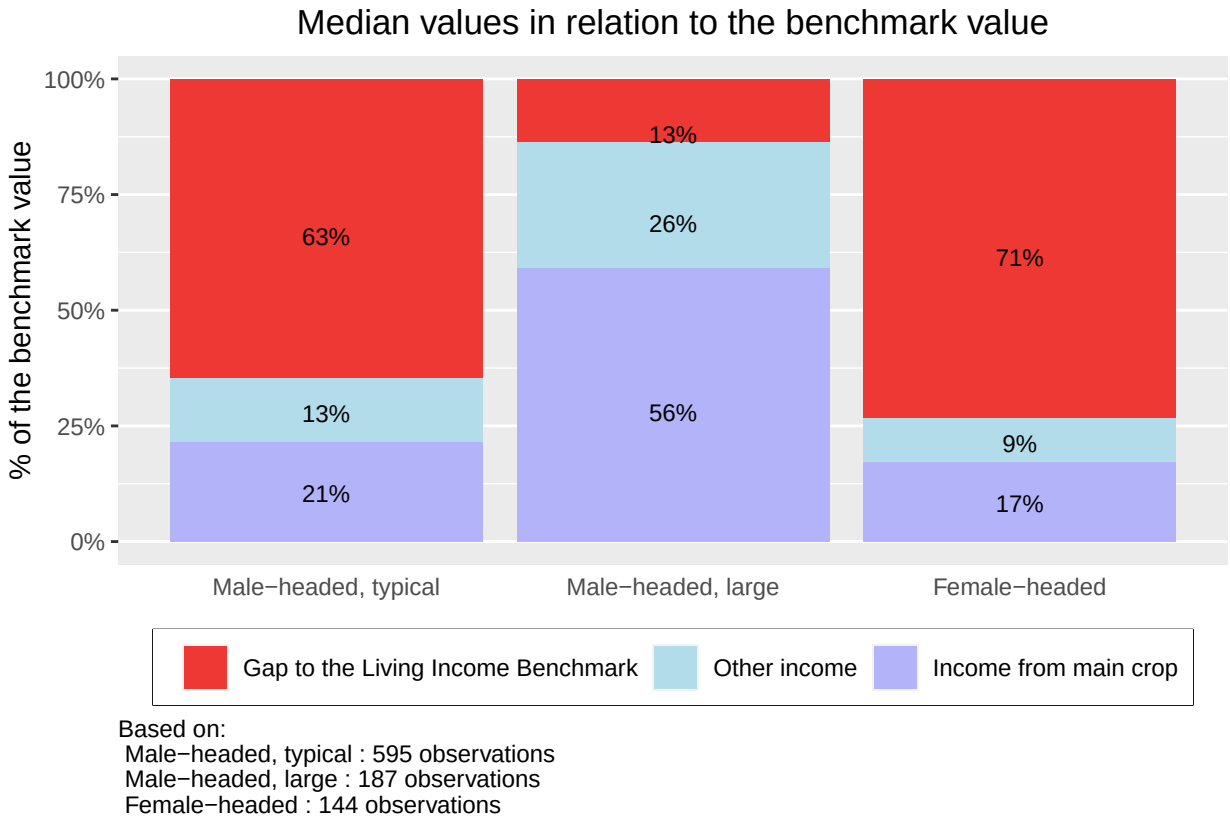


Based on:
Male−headed, typical : 595 observations
Male−headed, large : 187 observations
Female−headed : 144 observations

The gap of the relative median income to the Living Income Benchmark - bar chart

```
## The first section of this code summarizes and formats the data to be graph-ready
df %>%
# Group by household type
  group_by(grouping) %>%
# For each grouping, summarize the median gap to the living income,
# the median other income, and the median main_crop income.
# Calculate all as percentages of the Living income benchmark - the total of the income and gap
  summarise(Gap = median((benchmark - total_hh_income)/benchmark),
            Other = median((total_hh_income - income_main_crop)/benchmark),
            main_crop = median(income_main_crop/benchmark)) %>%
# Gather each income components into one column so the data is in 'long' format
  gather(key = "Component", value = "Income", Gap:main_crop) %>%
# Re-level the income factors for the order you want them stacked on the graph
  mutate(Component = factor(Component,
                            levels = c("Gap", "Other", "main_crop"))) %>%
# Generate ggplot graph for income by grouping and income component
  ggplot(aes(y = Income, x = grouping, fill = Component)) +
# Assign graph as stacked bar chart
  geom_bar(position = "fill", stat = "identity") +
# Label the graph title, axis, and caption
  labs(title = "Median values in relation to the benchmark value",
       y = "% of the benchmark value",
       x = "",
# Add caption with observation numbers for each household type
       caption = paste("Based on: \n",
                  paste(names(table(df$grouping)),
                      ":",
                      as.numeric(table(df$grouping)),
                      "observations \n ", collapse = ''), collapse = '')) +
# Label the legend and assign custom colors
  scale_fill_manual(values=c(gap_color, other_color, main_color),
                    breaks=c("Gap", "Other", "main_crop"),
                    labels=c("Gap to the Living Income Benchmark",
                             "Other income",
                             "Income from main crop")) +
# Format y-axis labels with a comma
  scale_y_continuous(labels = percent) +
# Remove x-axis grid lines and tick marks
  theme(panel.grid.major.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.x = element_blank(),
# Center plot title
        plot.title = element_text(hjust = 0.5),
# Move the legend to the bottom of the graph
        legend.position="bottom",
# Remove legend title
        legend.title = element_blank(),
# Put a box around the legend
        legend.box.background = element_rect(),
# Move caption to desired location
```

```
        plot.caption = element_text(hjust = 0)) +
# Add incomes to prospective graph components
  geom_text(aes(label = label_percent(accuracy = 1L)(Income)),
            position = position_stack(vjust = 0.5),
            size = 3)
```



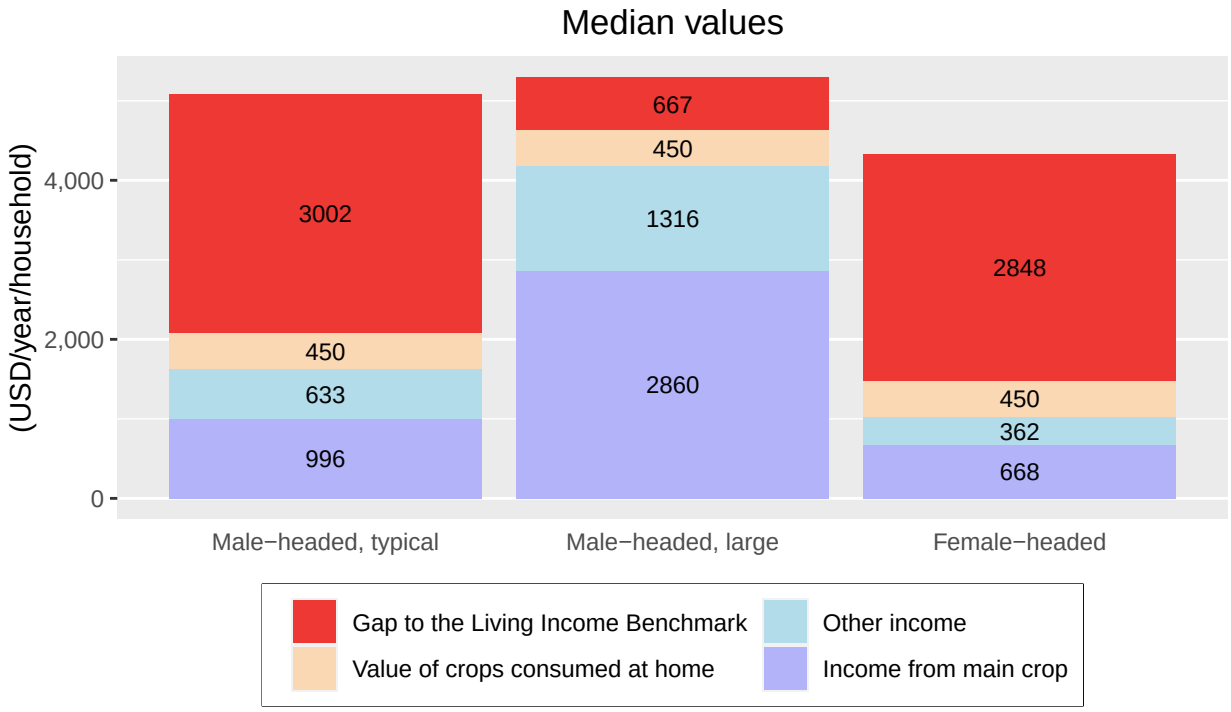
Based on:
Male−headed, typical : 595 observations
Male−headed, large : 187 observations
Female−headed : 144 observations

Bar charts including the intrinsic value of food produced and consumed at home

The gap of the absolute median income to the Living Income Benchmark - bar chart

```
## The first section of this code summarizes and formats the data to be graph-ready
df %>%
# Group by grouping
  group_by(grouping) %>%
# For each grouping, summarize the median gap to the living income,
# the median other income, and the median main_crop income.
  summarise(Gap = median(benchmark - total_hh_income),
            Other = median(total_hh_income - income_main_crop),
            Food = median(food_value),
            main_crop = median(income_main_crop)) %>%
# Gather each income components into one column so the data is in 'long' format
  gather(key = "Component", value = "Income", Gap:main_crop) %>%
# Re-level the income factors for the order you want them stacked on the graph
  mutate(Component = factor(Component,
                            levels = c("Gap", "Food", "Other", "main_crop"))) %>%
# Generate ggplot graph for income by grouping and income component
  ggplot(aes(y = Income, x = grouping, fill = Component)) +
# Assign graph as stacked bar chart
  geom_bar(position = "stack", stat = "identity") +
# Label the graph title, axis, and caption
  labs(title = "Median values",
       y = paste("(", currency, "/year/household)", sep = ""),
       x = "",
# Add caption with observation numbers for each household type
       caption = paste("Based on: \n",
                   paste(names(table(df$grouping)),
                       ":",
                       as.numeric(table(df$grouping)),
                       "observations \n ", collapse = ''), collapse = '')) +
# Label the legend and assign custom colors
  scale_fill_manual(values=c(gap_color, home_color, other_color, main_color),
                    breaks=c("Gap", "Food", "Other", "main_crop"),
                    labels=c("Gap to the Living Income Benchmark",
                             "Value of crops consumed at home",
                              "Other income",
                              "Income from main crop")) +
# Wrap legend onto 2 lines to fit everything neatly
  guides(fill = guide_legend(nrow = 2)) +
# Format y-axis labels with a comma
  scale_y_continuous(labels = comma) +
# Remove x-axis grid lines and tick marks
  theme(panel.grid.major.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.x = element_blank(),
# Center plot title
        plot.title = element_text(hjust = 0.5),
# Move the legend to the bottom of the graph
```

```
        legend.position="bottom",
# Remove legend title
        legend.title = element_blank(),
# Put a box around the legend
        legend.box.background = element_rect(),
# Move caption to desired location
        plot.caption = element_text(hjust = 0)) +
# Add incomes to prospective graph components
  geom_text(aes(label = round(Income)),
            position = position_stack(vjust = 0.5),
            size = 3)
```

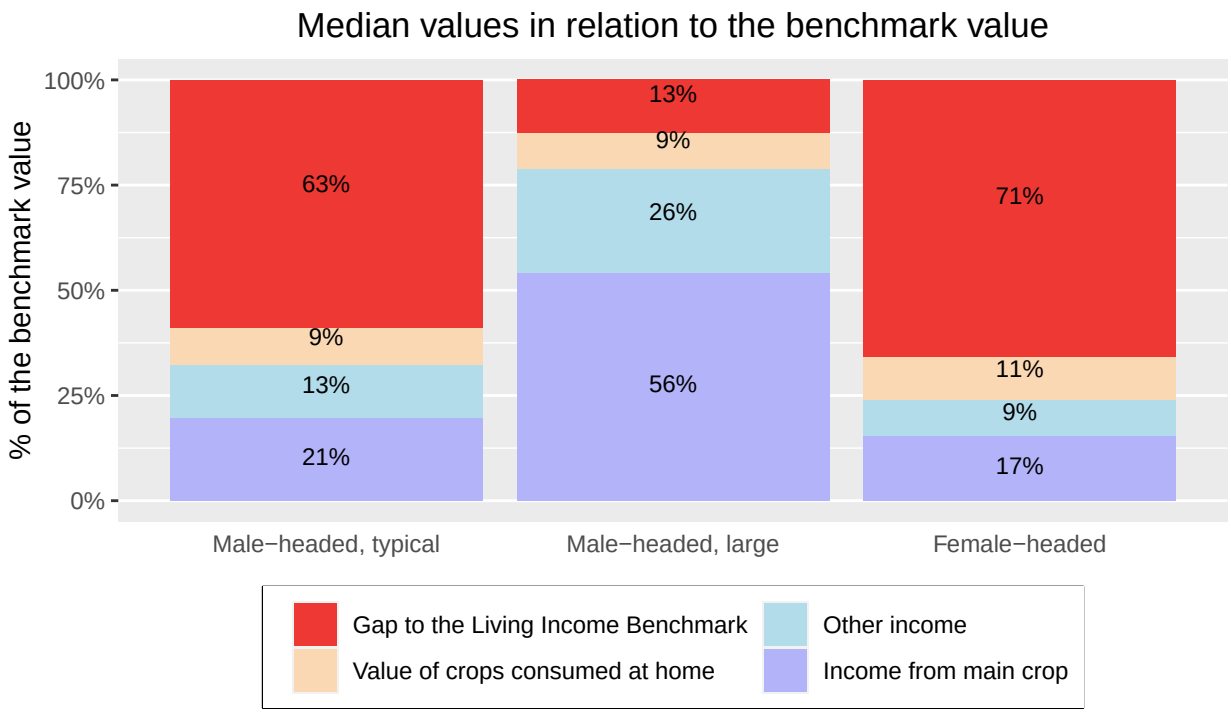


Based on:
Male−headed, typical : 595 observations
Male−headed, large : 187 observations
Female−headed : 144 observations

The gap of the relative median income to the Living Income Benchmark - bar chart

```
## The first section of this code summarizes and formats the data to be graph-ready
df %>%
# Group by household type
  group_by(grouping) %>%
# For each household type, summarize the median gap to the living income,
# the median other income, and the median main_crop income.
# Calculate all as percentages of the Living income benchmark - the total of the income and gap
  summarise(Gap = median((benchmark - total_hh_income)/benchmark),
            Food = median(food_value/benchmark),
            Other = median((total_hh_income - income_main_crop)/benchmark),
            main_crop = median(income_main_crop/benchmark)) %>%
# Gather each income components into one column so the data is in 'long' format
  gather(key = "Component", value = "Income", Gap:main_crop) %>%
# Re-level the income factors for the order you want them stacked on the graph
  mutate(Component = factor(Component,
                            levels = c("Gap", "Food", "Other", "main_crop"))) %>%
# Generate ggplot graph for income by grouping and income component
  ggplot(aes(y = Income, x = grouping, fill = Component)) +
# Assign graph as stacked bar chart
  geom_bar(position = "fill", stat = "identity") +
# Label the graph title, axis, and caption
  labs(title = "Median values in relation to the benchmark value",
       y = "% of the benchmark value",
       x = "",
# Add caption with observation numbers for each household type
       caption = paste("Based on: \n",
                   paste(names(table(df$grouping)),
                       ":",
                       as.numeric(table(df$grouping)),
                       "observations \n ", collapse = ''), collapse = '')) +
# Label the legend and assign custom colors
  scale_fill_manual(values=c(gap_color, home_color, other_color, main_color),
                    breaks=c("Gap", "Food", "Other", "main_crop"),
                    labels=c("Gap to the Living Income Benchmark",
                             "Value of crops consumed at home",
                             "Other income",
                             "Income from main crop")) +
# Wrap legend onto 2 lines to fit everything neatly
  guides(fill = guide_legend(nrow = 2)) +
# Format y-axis labels with a comma and assign limits 0-100%
  scale_y_continuous(labels = percent, limits = c(0,1)) +
# Remove x-axis grid lines and tick marks
  theme(panel.grid.major.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.x = element_blank(),
# Center plot title
        plot.title = element_text(hjust = 0.5),
# Move the legend to the bottom of the graph
        legend.position="bottom",
# Remove legend title
        legend.title = element_blank(),
```

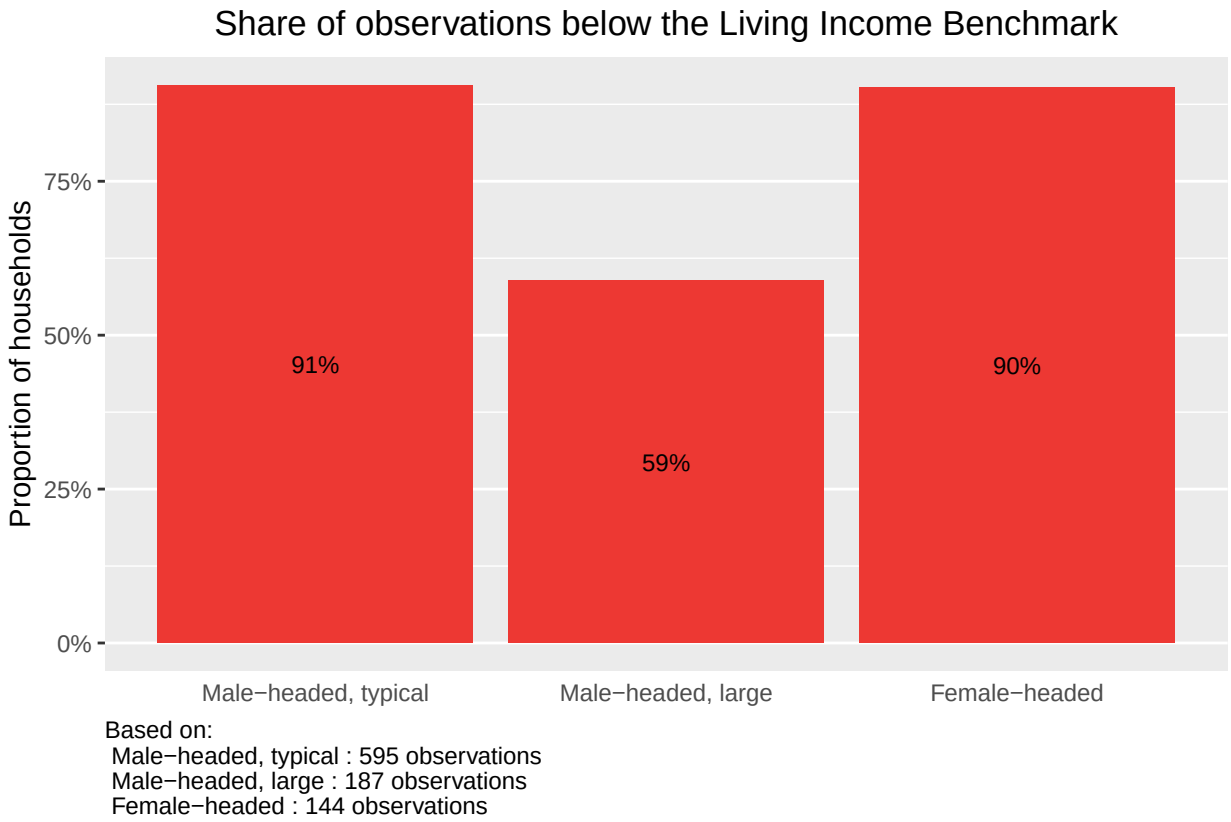
```
# Put a box around the legend
        legend.box.background = element_rect(),
# Move caption to desired location
        plot.caption = element_text(hjust = 0)) +
# Add incomes to prospective graph components
  geom_text(aes(label = label_percent(accuracy = 1L)(Income)),
            position = position_stack(vjust = 0.5),
            size = 3)
```



Based on:
Male−headed, typical : 595 observations
Male−headed, large : 187 observations
Female−headed : 144 observations

Share of those below the Living Income benchmark - bar chart

```
df %>%
# Group by household type
  group_by(grouping) %>%
# For each household type, calculate the percentage above the living income benchmark
  summarise(Below = sum(below_benchmark)/n()) %>%
# Generate ggplot graph for percentage by grouping
  ggplot(aes(x = grouping, y = Below)) +
# Assign graph as bar graph and color bars red for aesthetics
  geom_col(fill= share_color) +
# Label the graph title, axis, and caption
  labs(title = "Share of observations below the Living Income Benchmark",
       y = "Proportion of households",
       x = "",
# Add caption with observation numbers for each household type
       caption = paste("Based on: \n",
                   paste(names(table(df$grouping)),
                         ":",
                         as.numeric(table(df$grouping)),
                         "observations \n ", collapse = ''), collapse = '')) +
# Format y-axis labels with a percent
  scale_y_continuous(labels = percent) +
# Remove x-axis grid lines and tick marks
  theme(panel.grid.major.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.x = element_blank(),
# Center plot title
        plot.title = element_text(hjust = 0.5),
# Move caption to desired location
        plot.caption = element_text(hjust = 0)) +
# Add percents to each graph bar
  geom_text(aes(label = label_percent(accuracy = 1L)(Below)),
            position = position_stack(vjust = 0.5),
            size = 3)
```


Share of observations below the Living Income Benchmark
Proportion of households
75%
50%
25%
0%
91%
59%
90%
Male−headed, typical
Male−headed, large
Female−headed
Based on:
Male−headed, typical : 595 observations
Male−headed, large : 187 observations
Female−headed : 144 observations

Share of those below the Living Income Benchmark - density plot

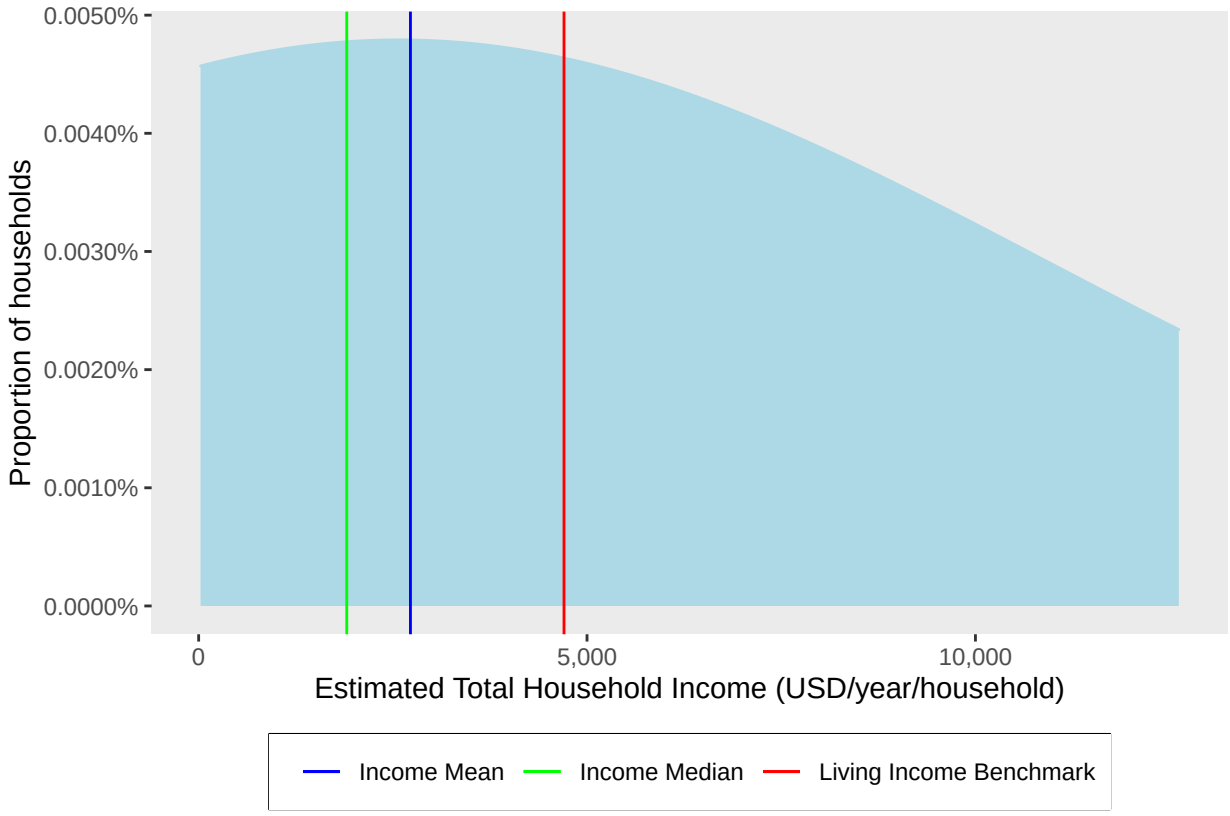
By household type

```
##### These first few lines of code draft a dataframe of the percent above the
  # LI benchmark for each group. The character strings in this dataframe are
  # then used as descriptive comments within the following graph
below <-
  df %>%
  group_by(grouping) %>%
  summarise(Percent = percent(sum(total_hh_income < benchmark)/n()))

# Formatting the strings as phrases with punctuation
below <- paste("Living Income", below$grouping, ": ",
               below$Percent, " below", sep = "")

##### Graph code begins here
df %>%
# Set x-axis to total household income and color by household type
  ggplot(aes(x = total_hh_income)) +
# Add density line. You can change the fill and line colors.
  geom_density(aes(fill = grouping), alpha = 0.9, color = "grey") +

#   for (i in 1:length(unique(df$grouping))) {
#
# ### Add vertical lines for Living Income benchmark(s)
# # Add and/or subtract lines based on number of household types
# # and if you have a different benchmark for each.
#   paste(as.character(geom_vline(aes(xintercept = unique(df$grouping)[i],
#                   color = below[i]),
#               key_glyph = "path")), "+" )
#   }
#     +
  scale_fill_manual(name = "Legend",
                    values = c(color_1, color_2, color_3, color_4),
                    breaks = levels(df$grouping),
                    labels = levels(df$grouping)) +
# Add graph and axis labels
  labs(x = paste("Estimated Total Household Income (",
                 currency, "/year/household)", sep = ""),
       y = "Income Probability",
# Add caption with observation numbers for each household type
       caption = paste("Based on: \n",
                   paste(names(table(df$grouping)),
                       ":",
                       as.numeric(table(df$grouping)),
                       "observations \n ", collapse = ''), collapse = '')) +
# Format x-axis labels as numbers with commas
# If your data has extreme income outliers,
    # you may need to filter them or add x-axis limits so they do not warp the graph
  scale_x_continuous(labels = comma, breaks=pretty_breaks(10)) +
# Remove y-axis labels and ticks
  theme(axis.text.y = element_blank(), axis.ticks.y = element_blank(),
# Remove background grid (the grid is the default)
```

```
        panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
# Move the legend to the bottom of the graph
        legend.position="bottom",
# Remove legend title
        legend.title = element_blank(),
# Remove gray background for legend symbols
        legend.key=element_rect(fill=NA),
# Put a box around the legend
        legend.box.background = element_rect(),
# Move caption to desired location
        plot.caption = element_text(hjust = 0),

  panel.background = element_rect(fill = "white", colour = NA)) +
# Wrap legend onto 3 lines to fit everything neatly
  guides(color = guide_legend(nrow = 3)) +
  geom_density(color = "dark blue")
```

With mean and median

```
df %>%
  # Set x-axis to key variable
  ggplot(aes(x = total_hh_income, y =(..count..)/n), fill = dataset) +
  # Add density line. You can change the fill and line colors.
  geom_density(color = "#add8e6", fill = "#add8e6", bw=8000) +
  # Add vertical line for living income
  geom_vline(aes(xintercept = mean(benchmark),
                 color = "Living Income Benchmark"), key_glyph = "path") +
  # Add vertical line for mean income
  geom_vline(aes(xintercept = mean(df$total_hh_income),
                 color = "Income Mean"), key_glyph = "path") +
  # Add vertical line for median income
  geom_vline(aes(xintercept = median(df$total_hh_income),
                 color = "Income Median"), key_glyph = "path") +
  # Add graph and axis labels
  labs(x = paste("Estimated Total Household Income (",
                 currency, "/", "year/household)",
                 sep = ""),
       y = "Proportion of households",
# Add caption with observation numbers for each household type
       caption = paste("N =", nrow(df), ", bin size = ",
                       "BINSIZE", collapse = '')) +
  # Format x-axis labels as numbers with commas
  # If your data has extreme income outliers,
    # you may need to filter them or add x-axis limits so they do not warp the graph
  scale_x_continuous(labels = comma) +
  scale_y_continuous(labels = percent) +
  # Remove y-axis labels and ticks
  theme(#axis.text.y = element_blank(), axis.ticks.y = element_blank(),
  # Remove background grid (the grid is the default)
        panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
  # Move the legend to the bottom of the graph
        legend.position="bottom",
  # Remove legend title
        legend.title = element_blank(),
  # Remove gray background for legend symbols
        legend.key=element_rect(fill=NA),
  # Put a box around the legend
        legend.box.background = element_rect(),
  # Move caption to desired location
        plot.caption = element_text(hjust = 0)) +
  # Manually add legend
  scale_color_manual(values = c("Living Income Benchmark"= "red",
                                "Income Mean"= "blue",
                                "Income Median"="green"))
```



N = 926 , bin size = BINSIZE

Share of those below the Living Income Benchmark - histogram

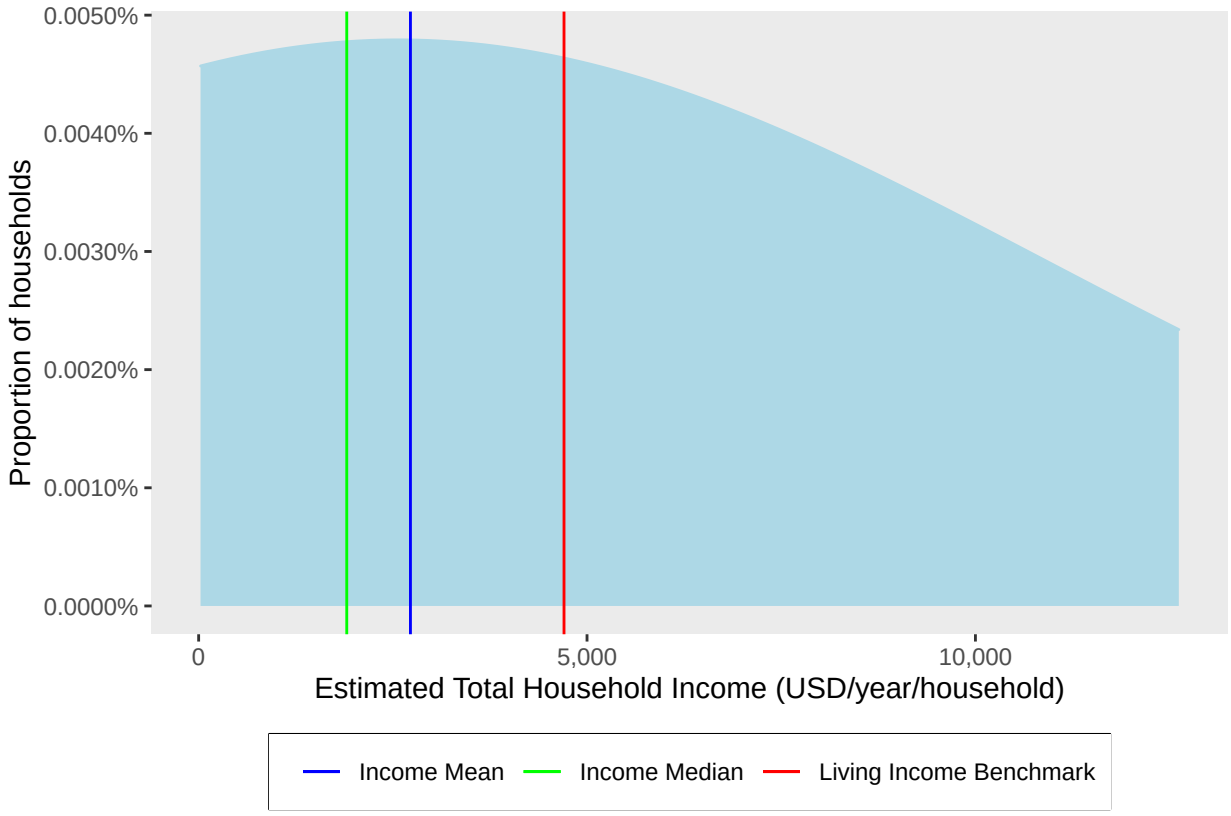
By household type

```
# This code drafts a dataframe of the percent above the LI benchmark for each group
# The character strings in the dataframe are then used as descriptive comments
  # within the graph
below <-
  df %>%
  group_by(grouping) %>%
  summarise(Percent = percent(sum(total_hh_income < benchmark)/n()))

# Formatting the strings as phrases with punctuation
below <- paste("Living Income", below$grouping, ": ",
               below$Percent, " below", sep = "")

df %>%
# Set x-axis to total household income
  ggplot(aes(x = total_hh_income,
# Color by household type
             fill = grouping)) +
# Add density line. You can change the fill and line colors.
  geom_density(alpha = 0.9, color = "grey") +

#   for (i in 1:length(unique(df$grouping))) {
#
# ### Add vertical lines for Living Income benchmark(s)
# # Add and/or subtract lines based on number of household types
# # and if you have a different benchmark for each.
#   paste(as.character(geom_vline(aes(xintercept = unique(df$grouping)[i],
#                   color = below[i]),
#               key_glyph = "path")), "+" )
#   }
#     +
  scale_color_manual(name = "Legend",
                     values = c(paste("Living Income 90% below" = color_1,
                                      "Living Income Male-headed, typical: 91% below"= color_2,
                                      "Living Income Male-headed, large: 59% below"= color_3))) +
# Add graph and axis labels
  labs(x = paste("Estimated Total Household Income (",
                 currency, "/year/household)", sep = ""),
       y = "Proportion of households",
# Add caption with observation numbers for each household type
       caption = paste("Based on: \n",
                   paste(names(table(df$grouping)),
                       ":",
                       as.numeric(table(df$grouping)),
                       "observations \n ", collapse = ''), collapse = '')) +
# Format x-axis labels as numbers with commas
# If your data has extreme income outliers,
    # you may need to filter them or add x-axis limits so they do not warp the graph
  scale_x_continuous(labels = comma) +
# Remove y-axis labels and ticks
  theme(axis.text.y = element_blank(), axis.ticks.y = element_blank(),
```

```
# Remove background grid (the grid is the default)
        panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
# Move the legend to the bottom of the graph
        legend.position="bottom",
# Remove legend title
        legend.title = element_blank(),
# Remove gray background for legend symbols
        legend.key=element_rect(fill=NA),
# Put a box around the legend
        legend.box.background = element_rect(),
# Move caption to desired location
        plot.caption = element_text(hjust = 0)) +
# Wrap legend onto 2 lines to fit everything neatly
  guides(fill = FALSE) +
# Wrap legend onto 3 lines to fit everything neatly
  guides(color = guide_legend(nrow = 3))
```

With mean and median

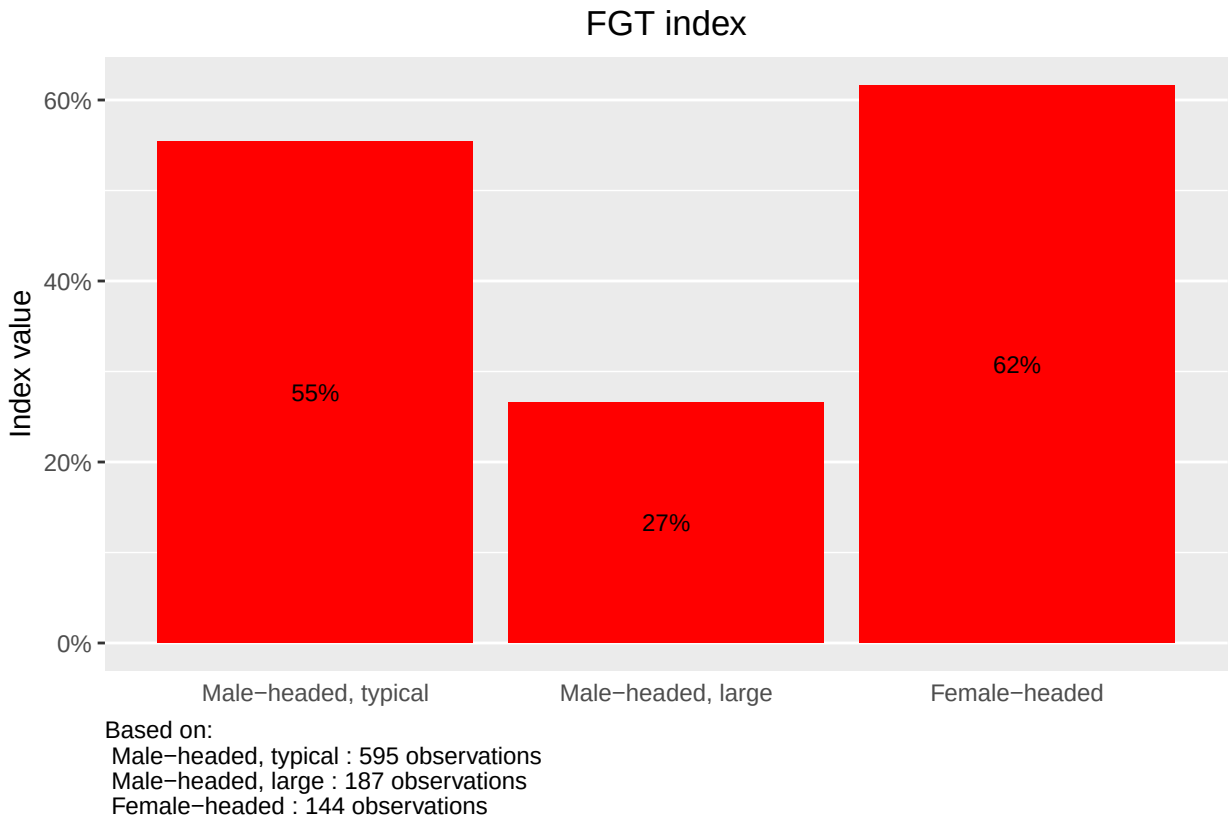
```
df %>%
  # Set x-axis to key variable
  ggplot(aes(x = total_hh_income, y =(..count..)/n), fill = dataset) +
  # Add density line. You can change the fill and line colors.
  geom_density(color = "#add8e6", fill = "#add8e6", bw=8000) +
  # Add vertical line for living income
  geom_vline(aes(xintercept = mean(benchmark),
                 color = "Living Income Benchmark"), key_glyph = "path") +
  # Add vertical line for mean income
  geom_vline(aes(xintercept = mean(df$total_hh_income),
                 color = "Income Mean"), key_glyph = "path") +
  # Add vertical line for median income
  geom_vline(aes(xintercept = median(df$total_hh_income),
                 color = "Income Median"), key_glyph = "path") +
  # Add graph and axis labels
  labs(x = paste("Estimated Total Household Income (",
                 currency, "/", "year/household)",
                 sep = ""),
       y = "Proportion of households",
# Add caption with observation numbers for each household type
       caption = paste("N =", nrow(df), ", bin size = ",
                       "BINSIZE", collapse = '')) +
  # Format x-axis labels as numbers with commas
  # If your data has extreme income outliers,
    # you may need to filter them or add x-axis limits so they do not warp the graph
  scale_x_continuous(labels = comma) +
  scale_y_continuous(labels = percent) +
  # Remove y-axis labels and ticks
  theme(#axis.text.y = element_blank(), axis.ticks.y = element_blank(),
  # Remove background grid (the grid is the default)
        panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
  # Move the legend to the bottom of the graph
        legend.position="bottom",
  # Remove legend title
        legend.title = element_blank(),
  # Remove gray background for legend symbols
        legend.key=element_rect(fill=NA),
  # Put a box around the legend
        legend.box.background = element_rect(),
  # Move caption to desired location
        plot.caption = element_text(hjust = 0)) +
  # Manually add legend
  scale_color_manual(values = c("Living Income Benchmark"= "red",
                                "Income Mean"= "blue",
                                "Income Median"="green"))
```



N = 926 , bin size = BINSIZE

Foster–Greer–Thorbecke (FGT) index

```
df %>%
# Group by household type
  group_by(grouping) %>%
# For each grouping, calculate the average Foster-Greer-Thorbecke (FGT) index
  summarise(FGT = mean(fgt_gap)) %>%
# Generate ggplot graph for percentage by grouping
  ggplot(aes(x = grouping, y = FGT)) +
# Assign graph as bar graph and color bars red for aesthetics
  geom_col(fill= "red") +
# Label the graph title, axis, and caption
  labs(title = "FGT index",
       y = "Index value",
       x = "",
# Add caption with observation numbers for each household type
       caption = paste("Based on: \n",
                   paste(names(table(df$grouping)),
                         ":",
                         as.numeric(table(df$grouping)),
                         "observations \n ", collapse = ''), collapse = '')) +
# Format y-axis labels with a percent
  scale_y_continuous(labels = percent) +
# Remove x-axis grid lines and tick marks
  theme(panel.grid.major.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.x = element_blank(),
# Center plot title
        plot.title = element_text(hjust = 0.5),
# Move caption to desired location
        plot.caption = element_text(hjust = 0)) +
# Add percents to each graph bar
  geom_text(aes(label = label_percent(accuracy = 1L)(FGT)),
            position = position_stack(vjust = 0.5),
            size = 3)
```



Based on:
Male−headed, typical : 595 observations
Male−headed, large : 187 observations
Female−headed : 144 observations