Teaching Introductory Statistics with R/RStudio

Categorical Data

Scatterplots

Correlation

Categorical Data

Recall that a **Categorical Variable** is any data whose values are names or labels rather than a numerical value.

For categorical variables, we can make a table of counts for each category (level).

The data set called `loans_full_schema` contains data on 10,000 loans (rows) and 55 variables (columns).

There is a categorical variable called `homeownership` with three levels: ***rent, mortgage, own***.

Here is the table of counts:

homeownership	Count
rent	3858
mortgage	4789
own	1353
Total	10000

To make tables for categorical variable in R, use the `tally()` function:

```
tally(~ homeownership, data = loans_full_schema)
```

2-By-2 Contingency Table

Here is a Contingency table of categorical variables **`application_type`** and **`homeownership`**:

		homeownership			
		rent	mortgage	own	Total
app_type	individual	3496	3839	1170	8505
	joint	362	950	183	1495
	Total	3858	4789	1353	10000

- Use **`tally()`** function to make a contingency table for two categorical variable :

```
tally(~ application_type + homeownership, data = loans_full_schema)
```

- To include row/column totals along the margins:

```
tally(~ application_type + homeownership, data = loans_full_schema, margins = TRUE)
```

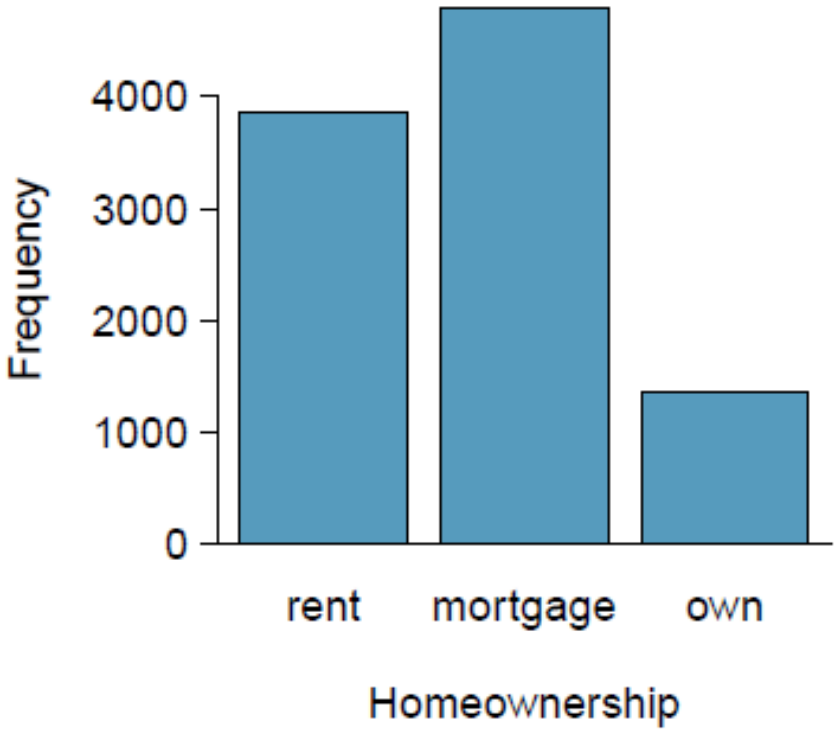
Bar Plots

Below are Bar Plots of **`homeownership`** variable.

The first Bar plot shows total counts for each level.

For the second plot, counts are converted to proportions (relative frequency)

For example, 3858/10000 = 0.3858 for *rent*.



Bar Plots in R

Total Counts on the Y-Axis:

```
gf_counts(~ homeownership, data = loans_full_schema)
```

Proportion (Relative Frequency: Count/Total) on Y-Axis

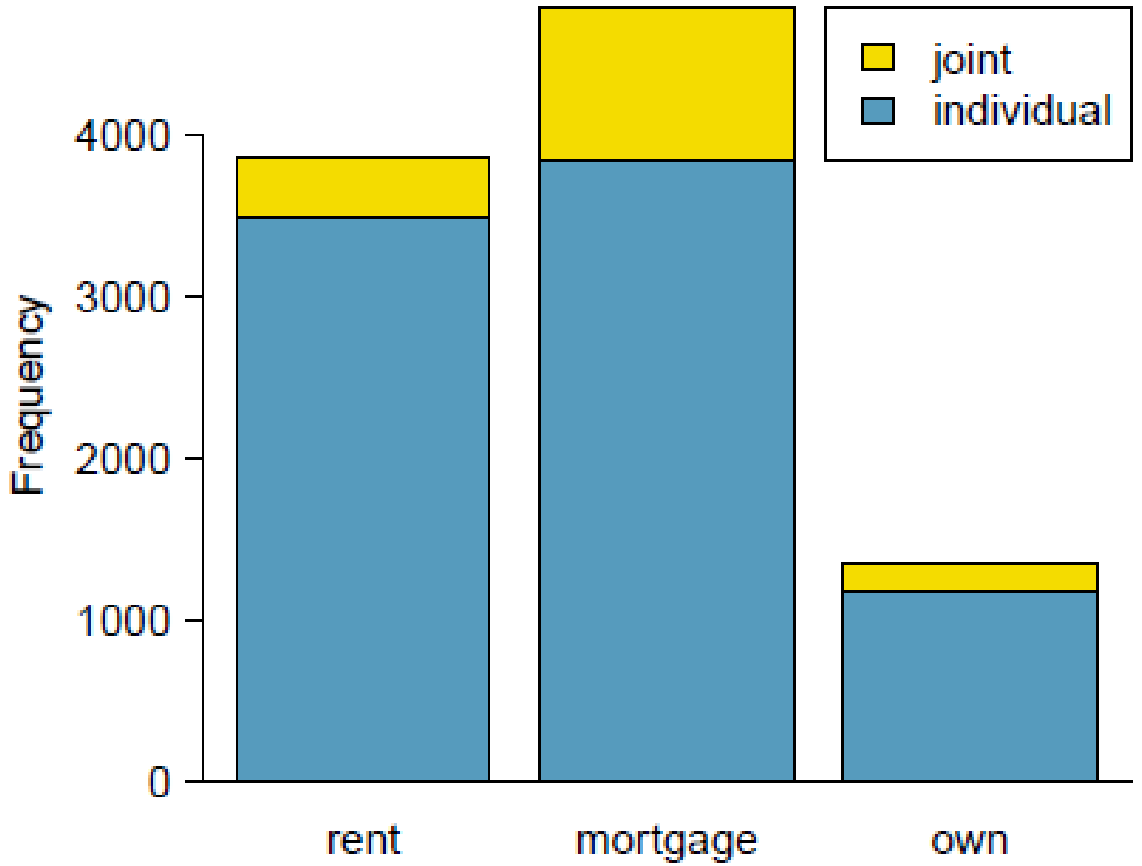
```
gf_props(~ homeownership, data = loans_full_schema)
```

Percentage (out of 100) on Y-Axis

```
gf_percents(~ homeownership, data = loans_full_schema)
```

Stacked Bar Plots in R

Contingency tables using row or column proportions help us examine how two categorical variables are related. **Stacked bar plots** provide a way to visualize the information in these tables.

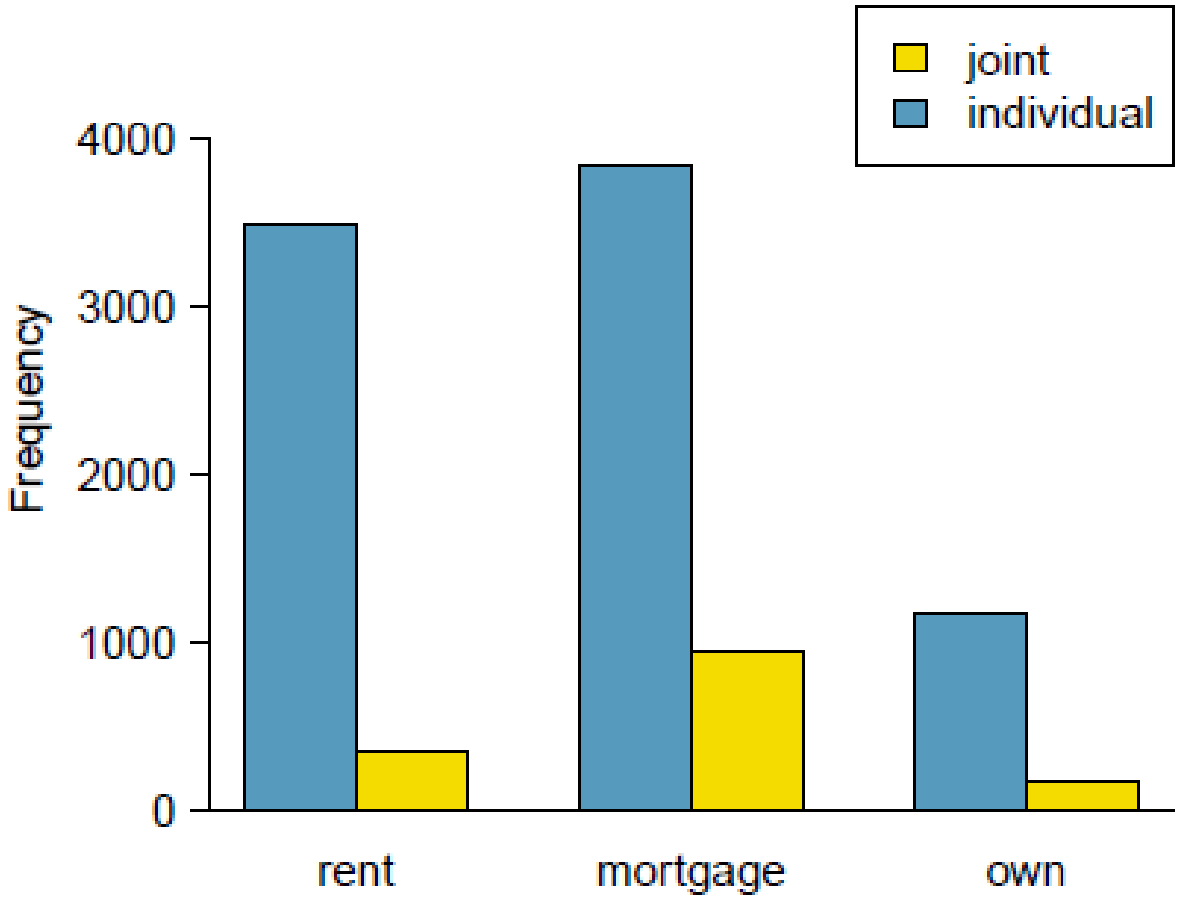


Notice a tilde here due to variable

```
gf_counts(~ homeownership, data = loans_full_schema, fill = ~ application_type)
```

Side-By-Side Bar Plots in R

Side-by-Side Bar Plot for two categorical variables.



Notice tilde here

```
gf_counts(~ homeownership, data = loans_full_schema, fill = ~ application_type,
          position = "dodge")
```

Mosaic Plots

A **mosaic plot** (no relation to our mosaic package!) is a data visualization technique for contingency tables It uses box areas to represent the number of cases in each category.

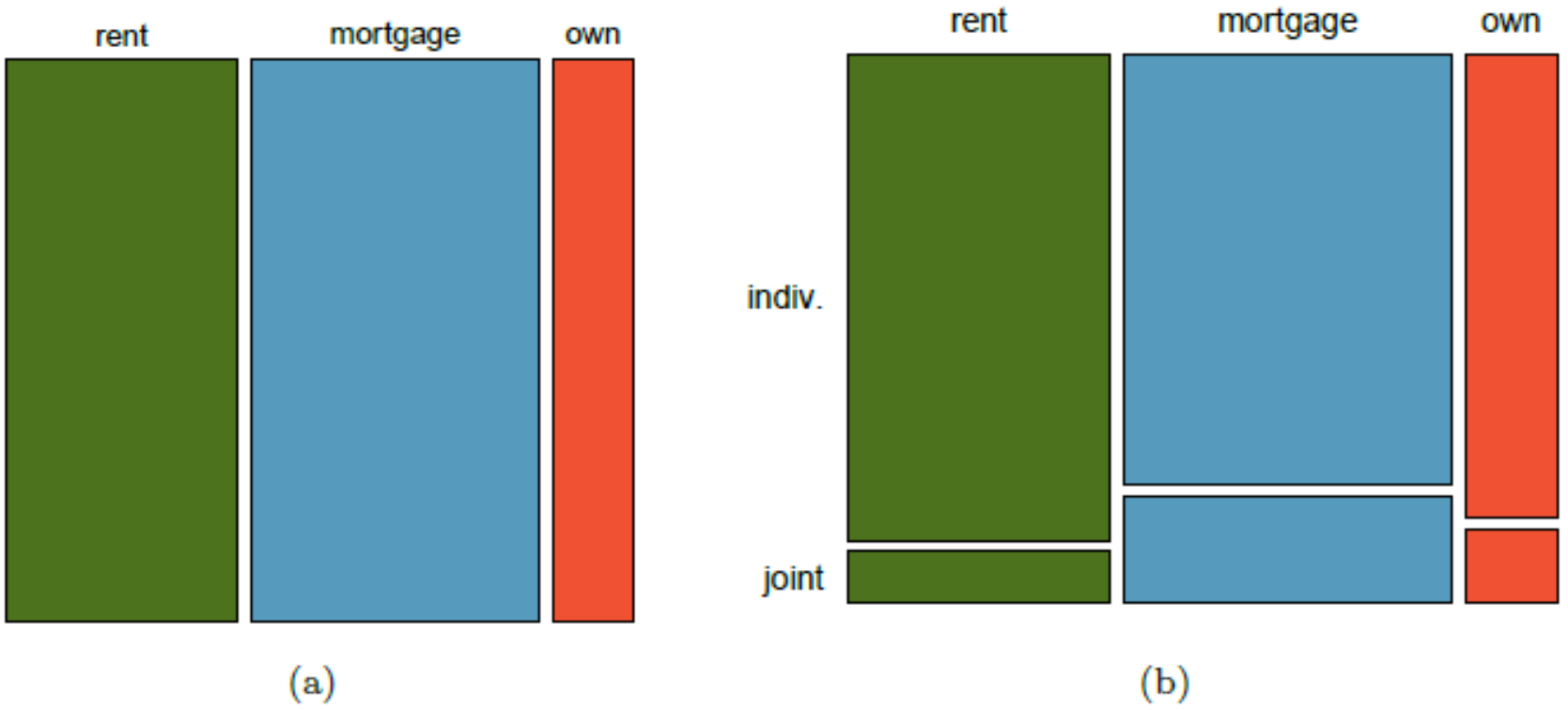


Figure 2.24: (a) The one-variable mosaic plot for `homeownership`. (b) Two-variable mosaic plot for both `homeownership` and `app_type`.

Mosaic Plots in R

To create a mosaic plot in R, there is a `mosaic()` function inside of a package called **vcd**.

And before, go to the **Packages** tab and check box for **vcd**. Or type: `library(vcd)`

Now we can use the `mosaic()` function.

Let's use the `Titanic` data set. It contains the following categorical variables:

Categorical Variable	Levels of Caterigorical Variable
Age	Adult or Child
Sex	Female or Male
Survived	Yes or No

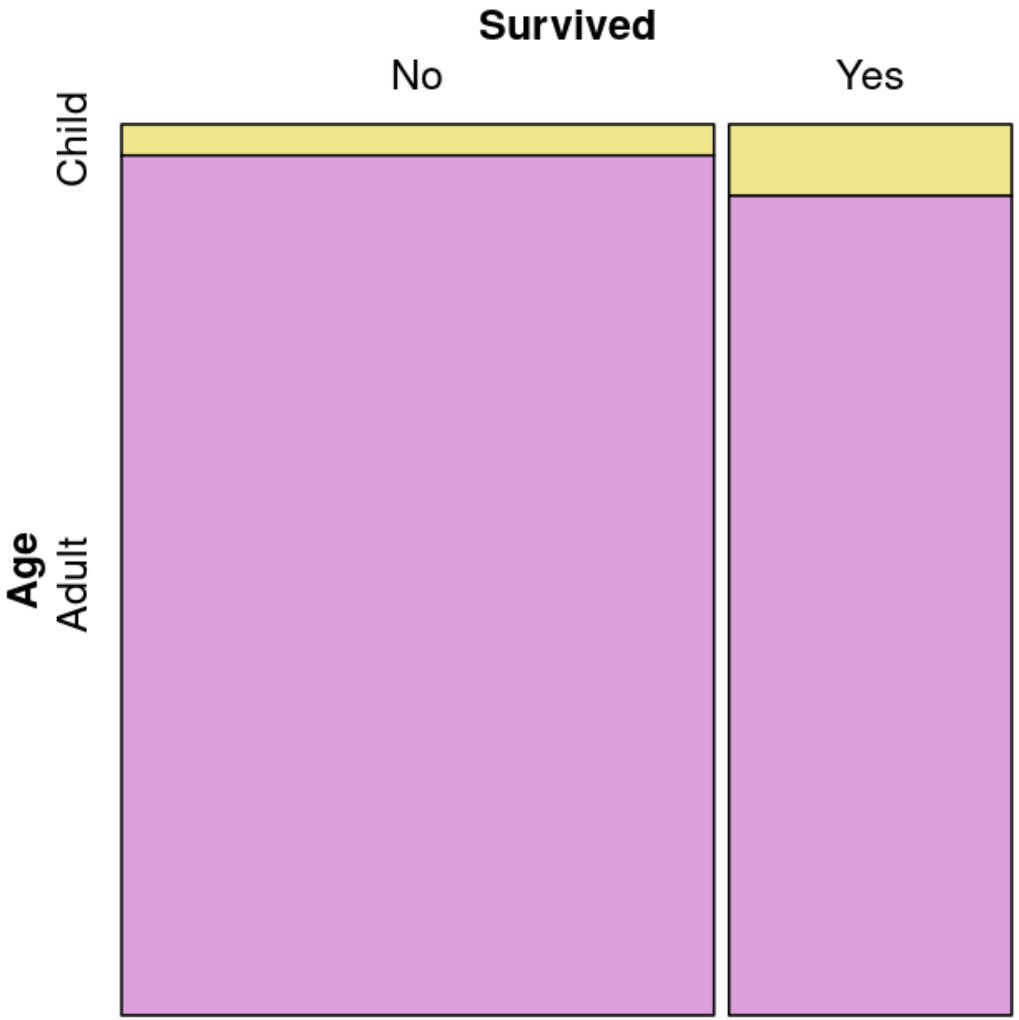
To view the Titantic data set, just type the name of the data set in the console. (Do not use view)

`Titanic` ← **Note that the name of this data set uses capital "T"**

Mosaic Plots in R

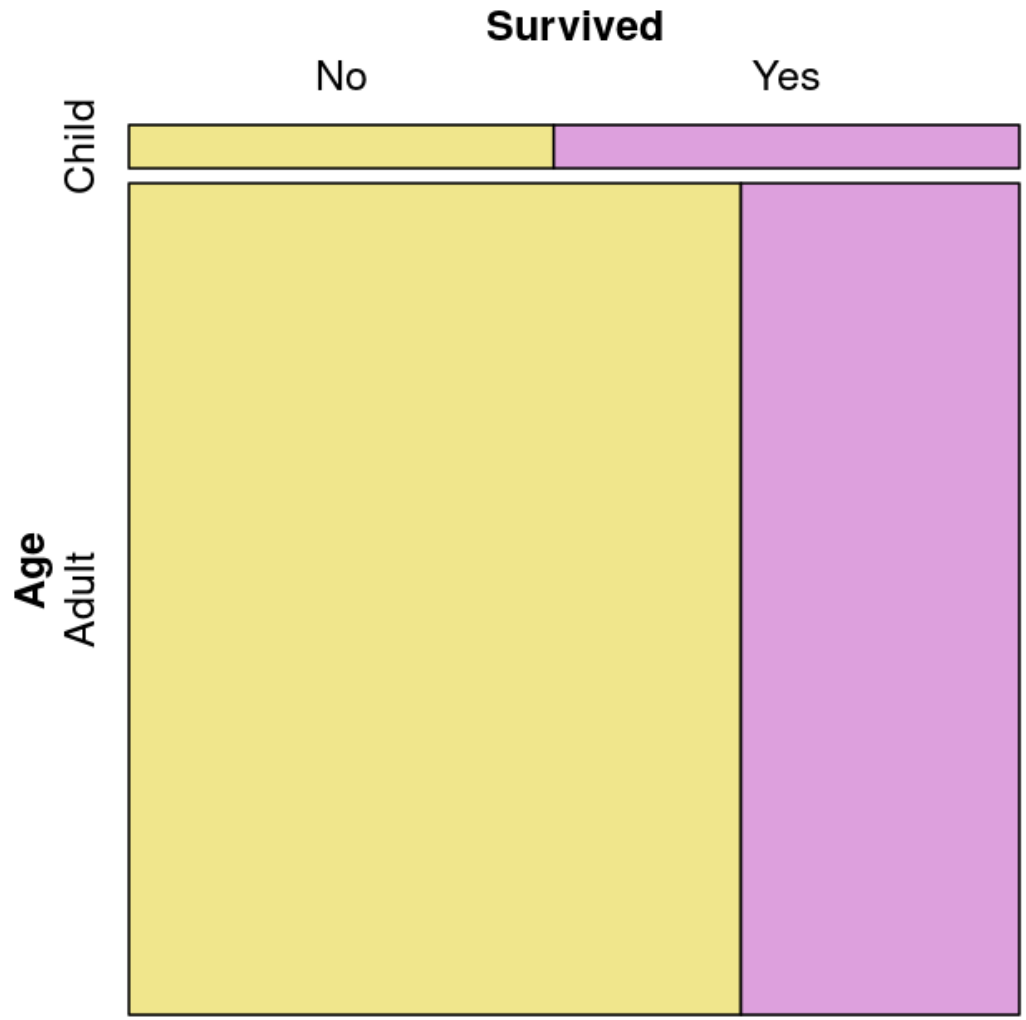

```
mosaic(~ Survived, data = Titanic)
```

Highlighting Age

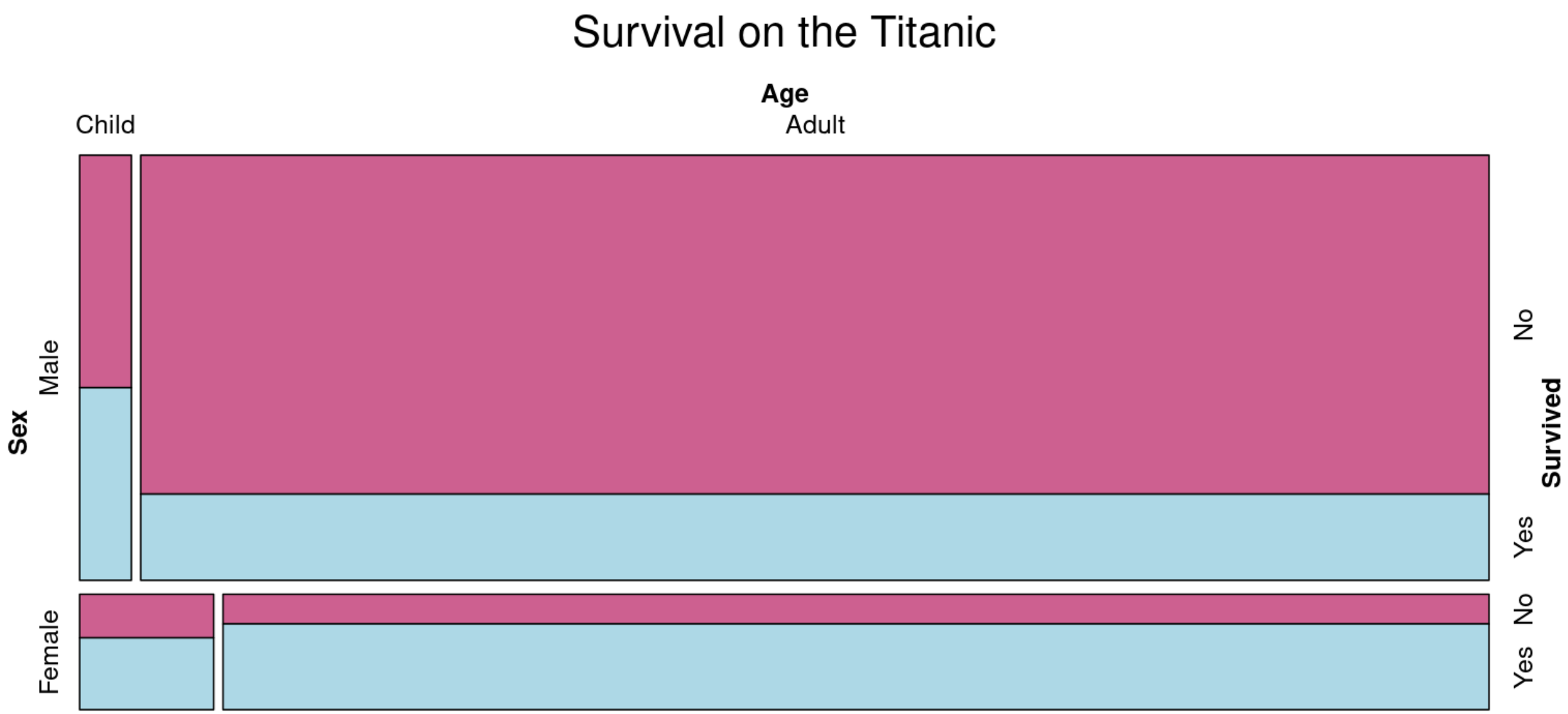



```
mosaic(~ Age + Survived, data = Titanic,
    highlighting = "Age",
    highlighting_fill = c("khaki", "plum"),
    main = "Highlighting Age")
```

Highlighting Survived




```
mosaic(~ Age + Survived, data = Titanic,
    highlighting = "Survived",
    highlighting_fill = c("khaki", "plum"),
    main = "Highlighting Survived")
```




```
mosaic(~ Sex + Age + Survived, data = Titanic,
           highlighting = "Survived",
           highlighting_fill = c("hotpink3", "lightblue"),
           main = "Survival on the Titanic")
```

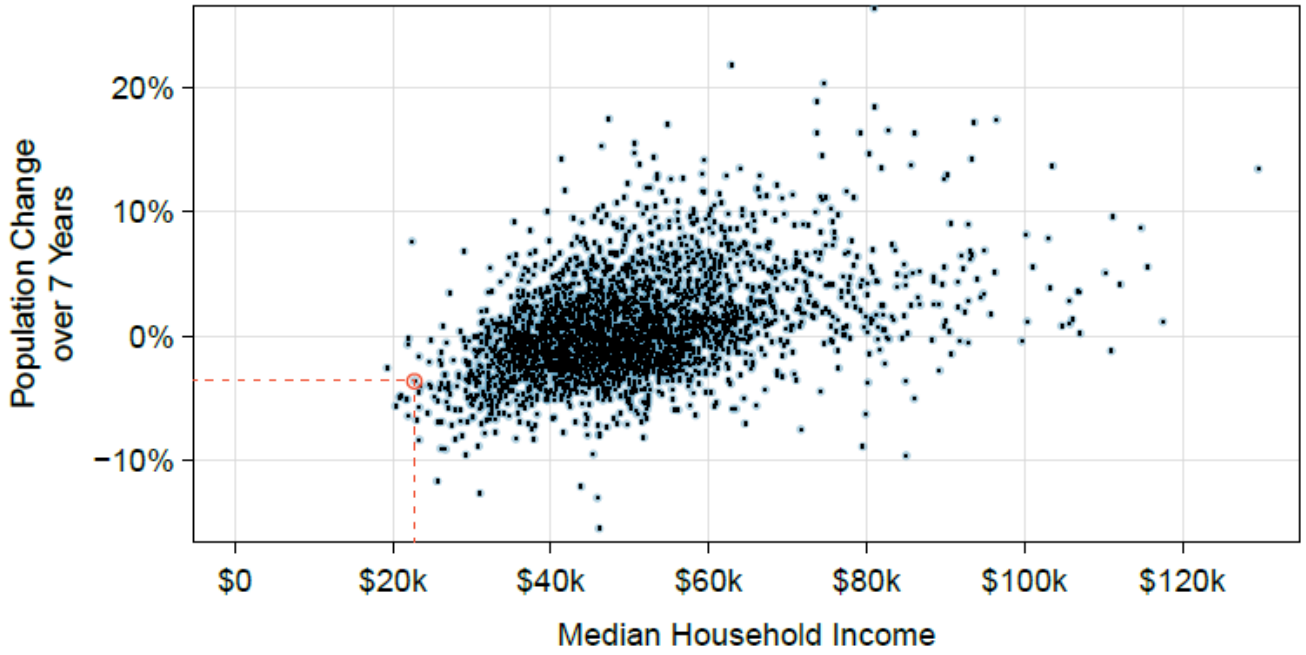
Scatterplots in R

Scatterplot: Map two numerical variables to the x and y coordinates.

Mosaic function for scatterplots: `gf_point(y ~ x, data = ...)`

Plot **`county`** data:

Y-variable: population change **`pop_change`** X-variable: Median household income (**`median_hh_income`**)




```
gf_point(pop_change ~ median_hh_income, data = county)
```

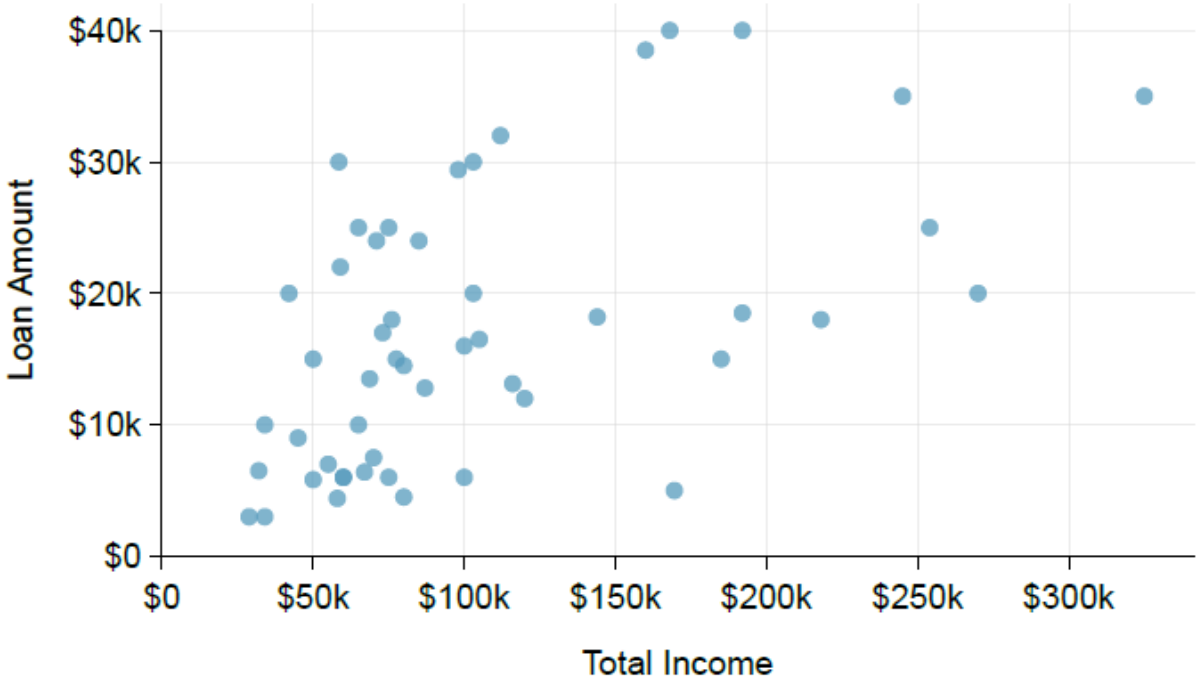
Scatterplots in R

`gf_point(y ~ x, data = ...)`

Color the Points

`loan50` data.

loan_amount plotted against total_income

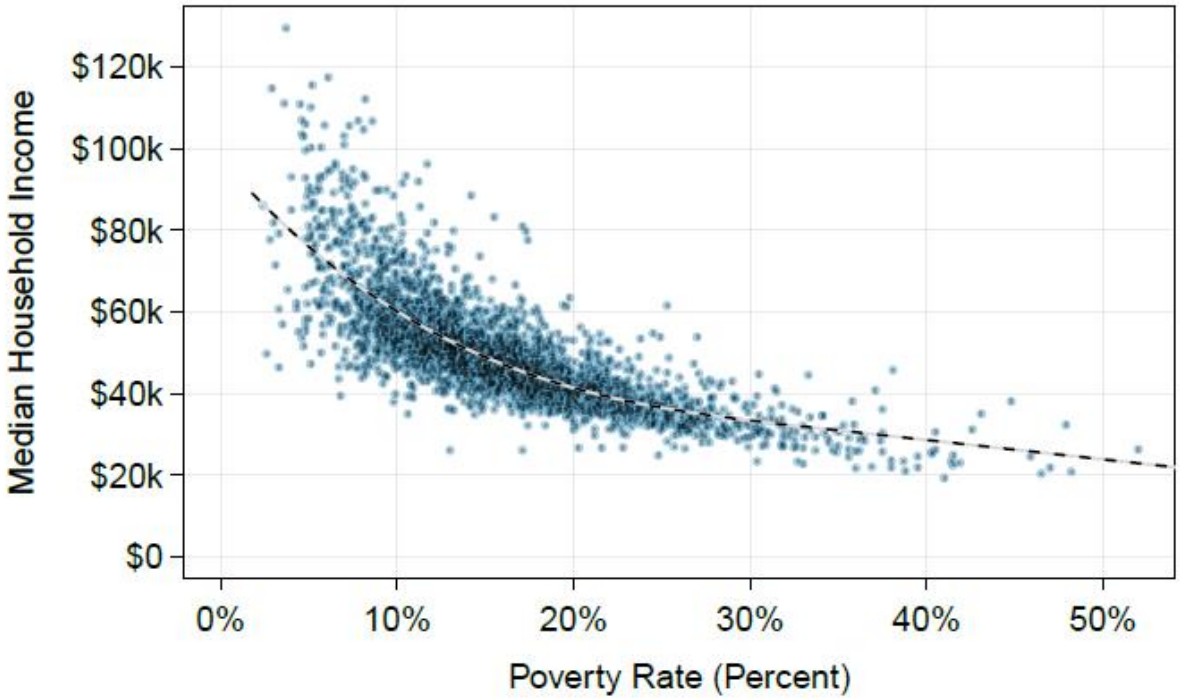



```
gf_point(loan_amount ~ total_income,
         data = loan50, color = "blue")
```

Add Smooth Curve Model and Axis Labels

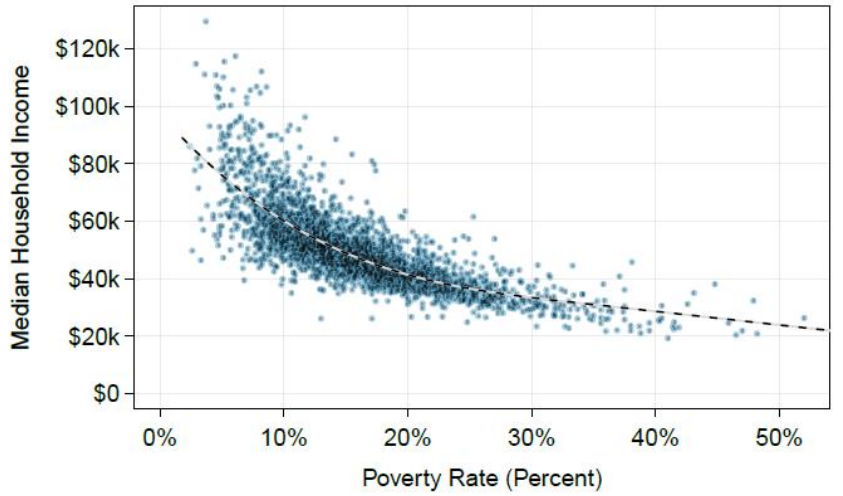
`county` data.

Median Income plotted against Poverty Rate




```
gf_point(median_hh_income ~ poverty, data = county) +
         ylab("Median Household Income") +
         xlab("Poverty Rate (Percent)") +
         geom_smooth()
```

Teaching With R Is FUN!
But Sometimes Hard Decisions (and Headaches!)



GOAL: Fit smooth trend line to data

OPTION 1:

Consistent Mosaic Functions, but needs pipe %>% and defaults to linear model for large data sets!

```
gf_point(median_hh_income ~ poverty, data = county) %>% gf_smooth()
```

Can override linear and specify a locally estimated smoother:

```
gf_point(median_hh_income ~ poverty, data = county) %>%
      gf_smooth(method = "loess")
```
← Can use `method = "lm"` for linear

OPTION 2:

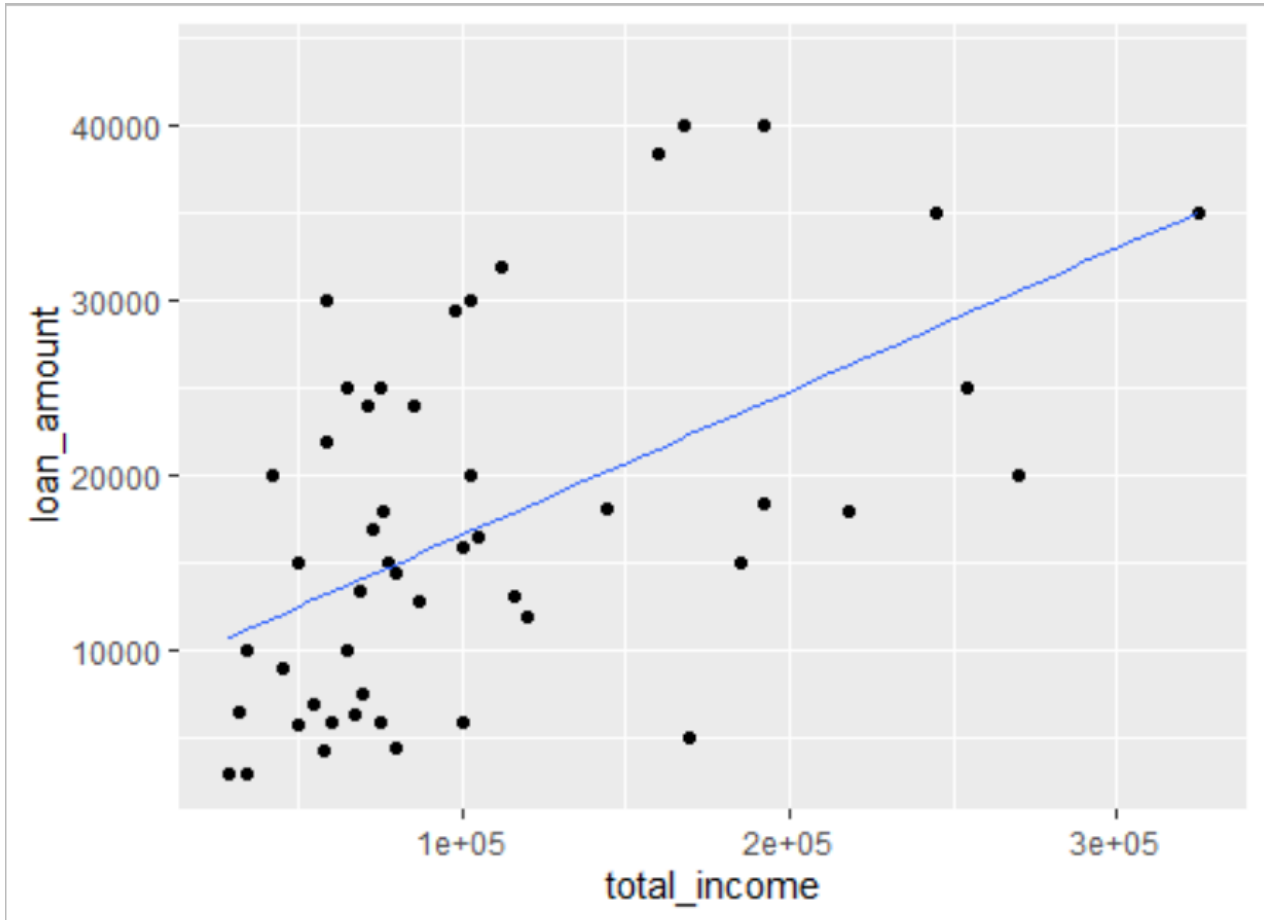
Another way is to borrow from the tidyverse (ggplot). Simpler "+" connecter.

```
gf_point(median_hh_income ~ poverty, data = county) + geom_smooth()
```

For linear model:

```
gf_point(median_hh_income ~ poverty, data = county) + geom_lm()
```

Pearson Correlation Coefficient



The first code makes the scatterplot.
An extra plot layer is added using `+ geom_lm()`
This adds the blue straight "Line of Best".
(Note: "**lm**" stands for "**linear model**".)

The second code calculates correlation.
It is about 0.54, indicating a moderate linear relationship between the two variables.

`use = "complete"` is needed for missing data.
Gives R permission to only use complete (x,y) pairs and omit incomplete pairs caused by missing data.

```
gf_point(loan_amount ~ total_income, data = loan50) + geom_lm()
```

```
cor(loan_amount ~ total_income, data = loan50, use = "complete")
[1] 0.5351341
```