SG1022 Seminar 3: Using survey data

Christopher Gandrud

2016

Objectives

- Getting (local) data into R
- Working with factors (categorical data)
- Using R to analyse surveys (cross-tabs, X^2, grouped bar plots)
- On you're own: Setting up Qualtrics for online survey administration

Pop Quiz: Preparing a survey

- What is a **survey mode**? Given an example of how the survey mode may affect your results.
- What is a **sampling frame**? How do you choose one?
- What is a **pilot** survey? Why are they important?
- What is **item non-response**?
- What is **acquiescence** and why might it be a problem in your survey?

File paths

Once you have gathered your survey data (and almost any type of data), you will store it on a computer in files. Files on a computer are organised **hierarchically** into (upside down) **trees**.

```
Root
|_
    Parent
    |_
        Child1
        Child2
```

You will see this week, and especially in the text analysis and reproducible research weeks, that knowing where your files are stored and how to access them is very important for doing computational data research.

Example file path

For example, the file path `C:\group_project\data\data_set.csv` represents the tree:

```
C
|_
    group_project
    |_
        data
        |_
            data_set.csv
```

Root naming conventions

Unfortunately, how your computer refers to the file tree depends on your operating system:

- **Windows**: The 'root' of the tree is a partition such as `C:\`. So, for example, `C:\group_project` indicates that the `group_project` directory is a child of the `C` partition.
- **Mac/Linux**: The 'root' of the tree is just denoted with a `/` with nothing before it. E.g. `/group_project` means that the `group_project` directory is a child of the root directory.

Sub (child) directories

Sub (child) directories are denoted with a `/` in Linux/Mac and `\` in Windows, e.g.:

```
# Linux/Mac
/group_project/data

# Windows
C:\group_project\data
```

R tip:

- In R for windows, use `/`, it will know what you mean.
- Or use two backslashes `\\` (nerd detail: `\` is the R escape character).

Working directories

A **working directory** is the directory where the program looks for files/other directories.

Always remember the working directory. Otherwise you may open/save files that you do not want to open/save.

Working directories

You can find out what your working directory is with the `getwd` function. List all of the files in that directory with `list.files`, and set your working directory with `setwd`.

```
# Find the working directory
getwd()

# List all files in the working directory
list.files()

# Set the U drive as your working directory
setwd('U:\\')
```

File & directory name conventions

Don't use spaces in your file names.

They can create problems for programs that treat spaces as an indication that the path has ended.

Alternatives:

- `CamelCase` (ex. `DataAnalysis.R`)
- `file_underscore` (ex. `data_analysis.R`)

Loading data into R

R can load data from many different file formats (e.g. `.sav` (SPSS), `.xlsx` (Excel), `.dta` (Stata), `.csv`).

The rio (R input/output) package makes it very easy to import many different types of data. It has two key functions `import` and `export`.

Install rio

Remember: You need to `install.packages` rio and then load it into your workspace with `library`.

In your **console**:

```
install.packages('rio')
```

In **a new R source code file**:

```
library(rio)
```

Rio `import` for loading data into R

First: create a folder in your `U` drive called `sg1022_data`.

Second: download the `ESS5_UKonly.sav` data set from Moodle (it's under Week 3).

Third, set your working directory to the folder where the data is located:

```
setwd('U:\\sg1022_data/')
```

Tip: If you begin typing the directory name and then hit the `TAB` key on your keyboard, RStudio will list possible directory names, so you don't have to type the whole thing.

Rio `import` for loading data into R

Now load the data using the file's name (`ESS5_UKonly.sav`)

```
library(rio)

# Load ESS 5 (UK) only data from SPSS format
ess5_uk <- import('ESS5_UKonly.sav')

# Show a selection of the data.
head(ess5_uk[1:3, 1:6])
```

```
##        name essround edition   proddate     idno          cntry
## 1 ESS5e03_1        5     3.1 29.04.2014 10100034 United Kingdom
## 2 ESS5e03_1        5     3.1 29.04.2014 10100059 United Kingdom
## 3 ESS5e03_1        5     3.1 29.04.2014 10100067 United Kingdom
```

Note: Always look at the data you imported to see if it's what you think you imported/see what needs cleaning.

Rio `export` for saving data

We can save the `ess5_uk` data into another format. For example, .csv "Comma Separated Values". Just add the csv file extension to the file name and `rio` does the rest.

```
# Save file in current working directory
export(ess5_uk, file = 'ess5_uk.csv')
```

Review: factors (categorical data)

In R, categorical data is coded using **factors**.

Let's load a simple data set into R for this example: `example_kw.csv`.

This file is located on Moodle (Week 3). Download it and place it into your `sg1022_data` directory. Now load it into a new object called `kanye_survey`.

```
kanye_survey <- import('example_kw.csv')
```

Review: factors (categorical data)

This data set has a numeric variable called `income`. We want to convert it to a factor with three category labels:

Number Code	Label
1	Low income
2	Medium income
3	High income

Factor labels

To convert this variable to a factor and add labels use the `factor` function.

```
# Create a vector of level labels
income_labels <- c('Low income', 'Medium income', 'High income')

# Convert income to factor and apply labels
kanye_survey$income <- factor(kanye_survey$income,
                              labels = income_labels)
```

```r
# Create simple frequency table
summary(kanye_survey$income)
```

```
##    Low income Medium income   High income
##           495           399           106
```

Converting from character strings to factors (1)

Sometimes you have data that is in character strings (R sees letters, but doesn't see any categories), but you want it to be a factor. For example:

```r
summary(kanye_survey$kanye_or_wiz)
```

```
##    Length     Class      Mode
##      1000 character character
```

Converting from character strings to factors (2)

Simply run it through `as.factor`. R will turn it into a factor and use the character strings as factor labels.

```r
kanye_survey$kanye_or_wiz <- as.factor(kanye_survey$kanye_or_wiz)

summary(kanye_survey$kanye_or_wiz)
```

```
## kanye   wiz
##   640   360
```

Frequency table (categorical variables)

Remember, use `summary` to create a basic frequency table of a factor variable in R.

```r
income_freq <- summary(kanye_survey$income)
income_freq
```

```
##    Low income Medium income   High income
##           495           399           106
```

We can convert these counts to proportions with `prop.table`:

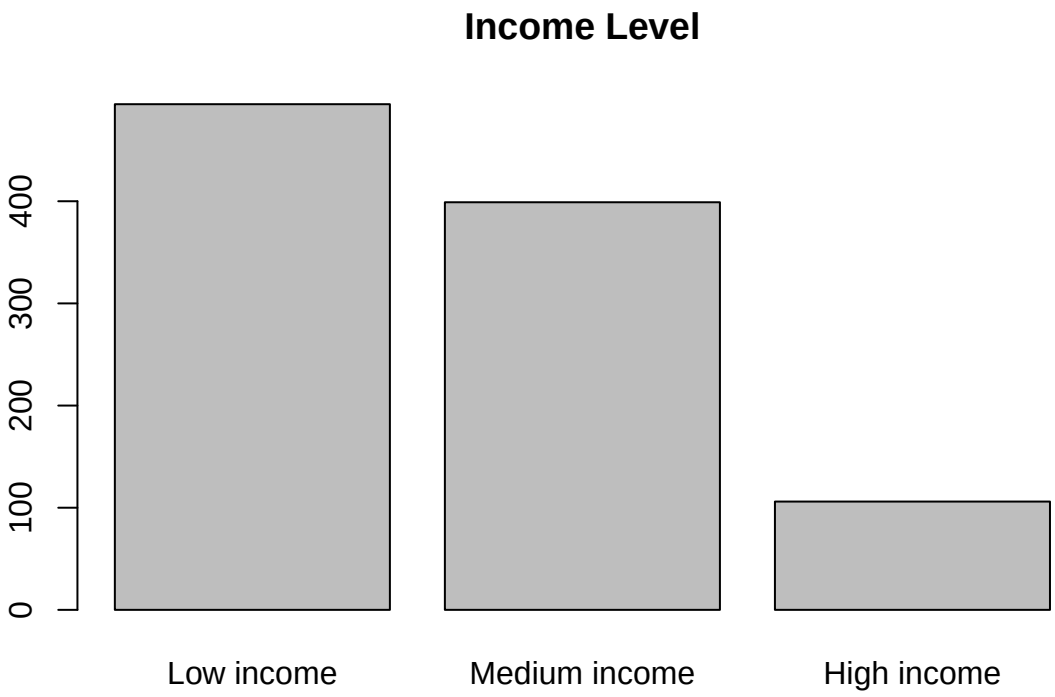
```r
income_prop <- prop.table(income_freq)
income_prop
```

```
##    Low income Medium income   High income
##         0.495         0.399         0.106
```

Barplot (frequencies)

Show the frequencies more effectively with a barplot. To create a barplot of a single factor variable just use `plot`:

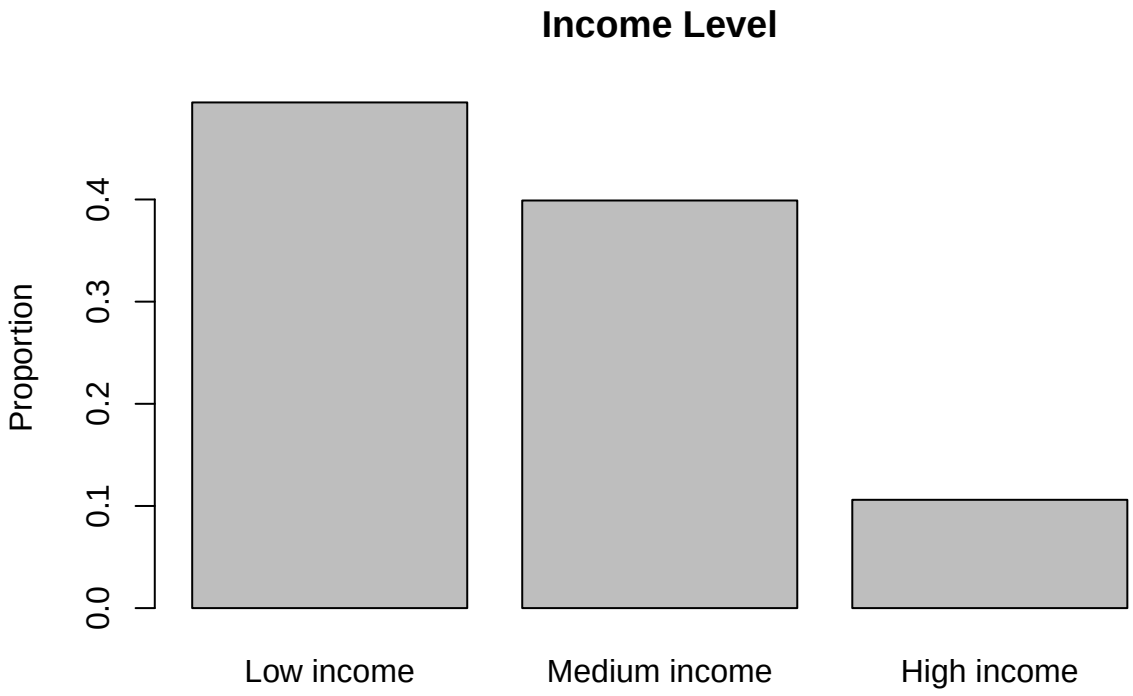
```
plot(kanye_survey$income, main = 'Income Level')
```



Barplot (proportions)

You can also plot the proportions you created before. This time you need to explicitly use `barplot`.

```
barplot(income_prop, main = 'Income Level', ylab = 'Proportion')
```



Joint distributions (categorical variables)

Use `table` to create a simple contingency table:

```
support <- table(kanye_survey$kanye_or_wiz, kanye_survey$income)
support
```

```
##
##         Low income Medium income High income
##   kanye        311           260          69
##   wiz          184           139          37
```

Note: the contingency table you create with the `table` function **is the basis for all** of the following cross-tabs, barplots, and X^2 tests.

Cross-tabs with proportions

Use `prop.table` again to find the contingency table proportions:

```
prop.table(support, margin = 1) # row proportions
```

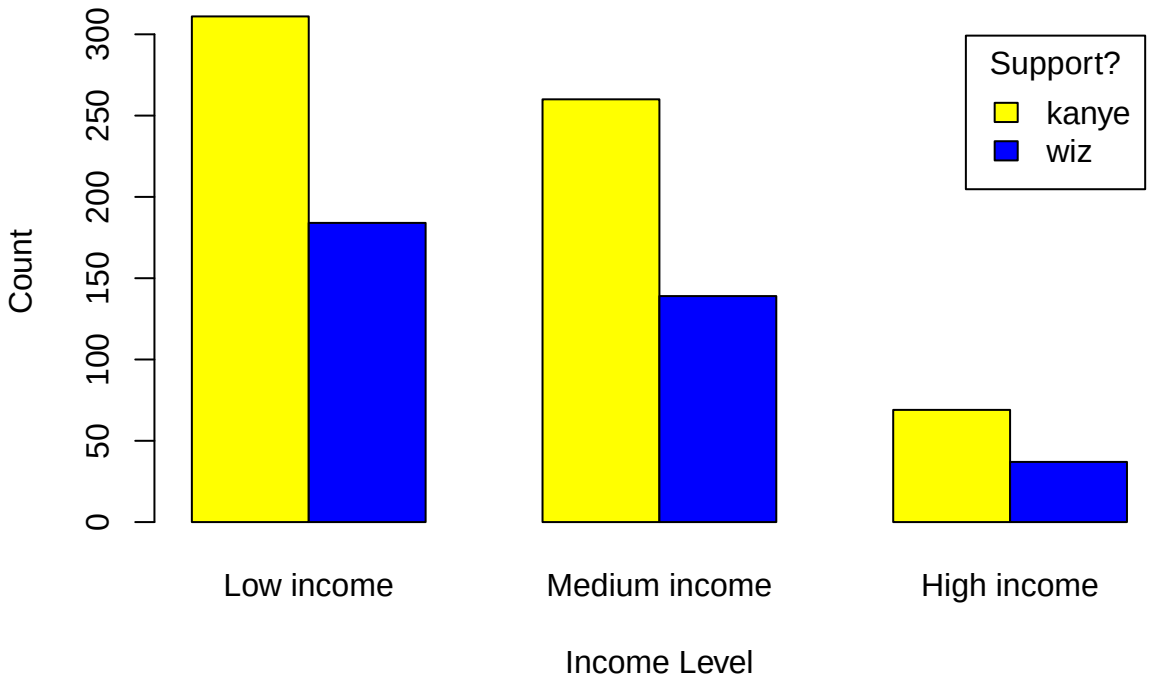
```
##
##         Low income Medium income High income
##   kanye  0.4859375     0.4062500   0.1078125
##   wiz    0.5111111     0.3861111   0.1027778
```

```r
prop.table(support, margin = 2) # column proportions
```

```
##
##         Low income Medium income High income
##   kanye  0.6282828     0.6516291   0.6509434
##   wiz    0.3717172     0.3483709   0.3490566
```

Plotting (grouped bar chart)–Base R

```r
barplot(support, beside = TRUE, col = c('yellow', 'blue'),
  xlab = 'Income Level', ylab = 'Count',
  legend = rownames(support), args.legend = list(title = 'Support?'))
```



Colours in R

For a list of R colour names see: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

Plotting Percentages–Base R (1)

To plot proportions (easier for cross-group comparability), first create a table of the proportions:

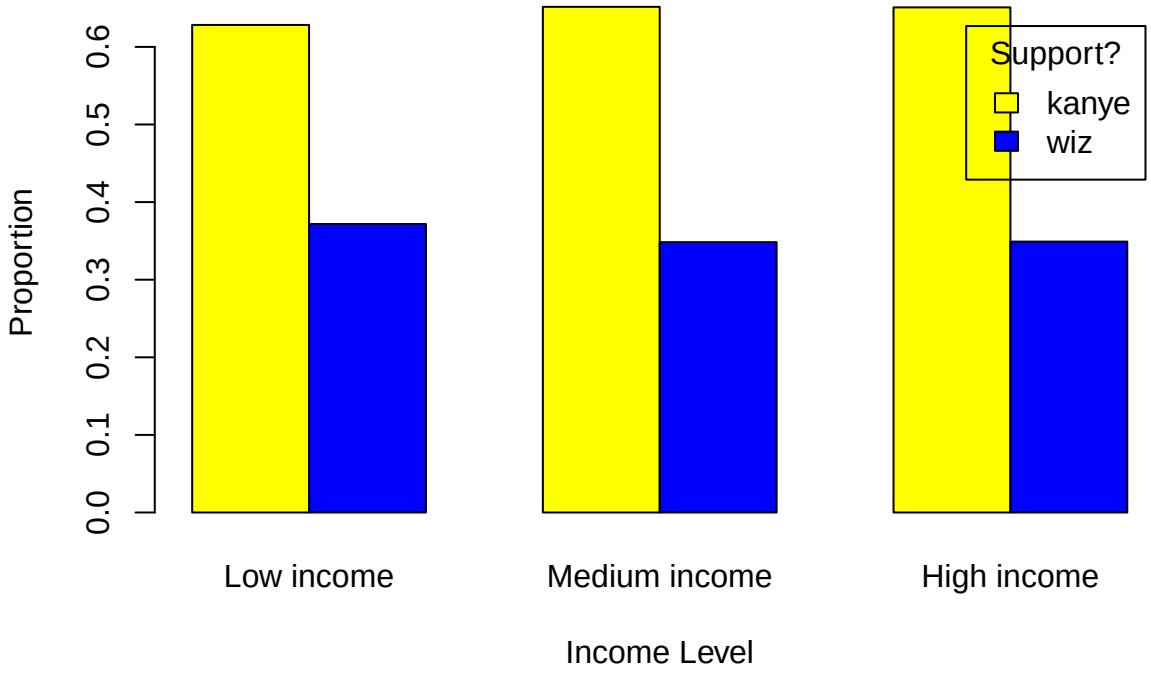
```r
support_prop <- prop.table(support, margin = 2) # column proportion
support_prop
```

```
##
##         Low income Medium income High income
##   kanye  0.6282828     0.6516291   0.6509434
##   wiz    0.3717172     0.3483709   0.3490566
```

Then give these to `barplot` as before . . .

Plotting Proportions–Base R (2)

```
barplot(support_prop, beside = TRUE, col = c('yellow', 'blue'),
  xlab = 'Income Level', ylab = 'Proportion',
  legend = rownames(support), args.legend = list(title = 'Support?'))
```



Plotting Percentages–Base R (1)

To plot percentages, simply convert the proportions to percents by multiplying them by 100:

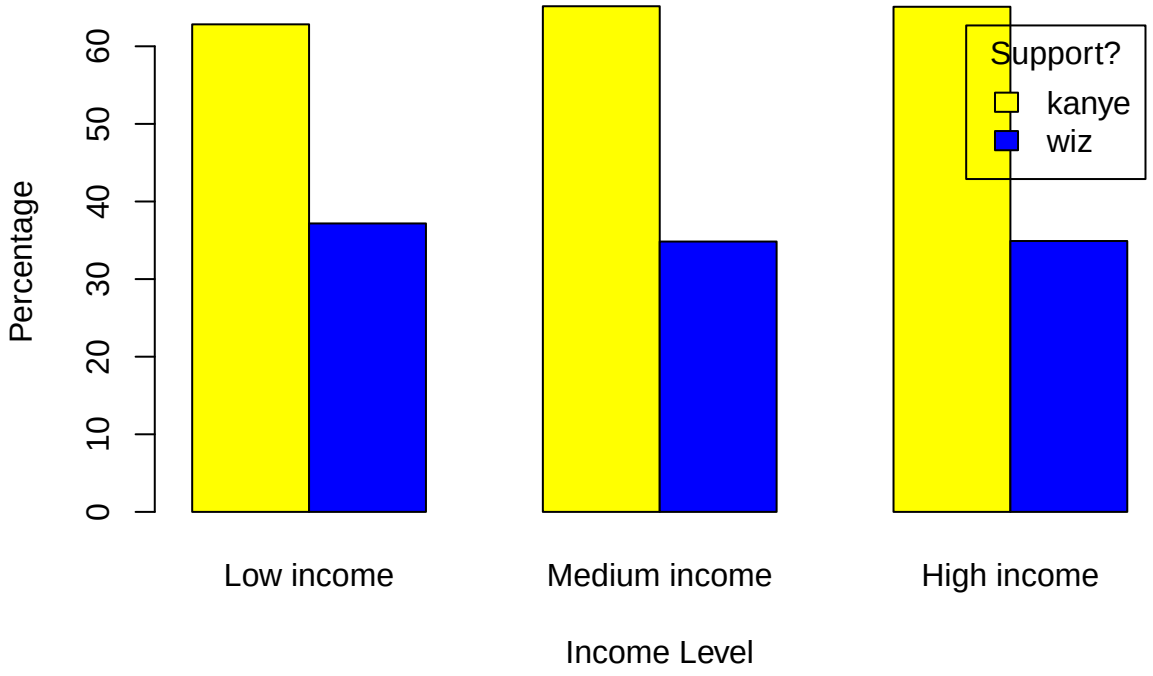
```
# Convert to percentages
support_perc <- support_prop * 100
support_perc
```

```
##
##         Low income Medium income High income
##   kanye   62.82828      65.16291    65.09434
##   wiz     37.17172      34.83709    34.90566
```

Then give these to `barplot` as before . . .

Plotting Percentages–Base R (2)

```r
barplot(support_perc, beside = TRUE, col = c('yellow', 'blue'),
  xlab = 'Income Level', ylab = 'Percentage',
  legend = rownames(support), args.legend = list(title = 'Support?'))
```



Joint distributions (categorical variables)

χ^2 Test

```r
chisq.test(support)
```

```
##
##  Pearson's Chi-squared test
##
## data:  support
## X-squared = 0.58426, df = 2, p-value = 0.7467
```

Based on this test: is there a statistically significant association between income level and support for Kanye West and Wiz Khalifa?

Set up and run Qualtrics survey

Now, set up a Qualtrics Survey using *Worksheet: introduction to Qualtrics* on Moodle under *Qualtrics Resources*.

You can use Qualtrics to administer a survey via the internet. This may be helpful for your group project if you choose to use surveys as your data pathway.

Set up a simple (no more than 3 or 4 questions) Qualtrics survey, administer it to 3 classmates, download and load the results into R.

Extras: reorder factors

You may want to **change the order** of a factor's variable's levels so that makes more substantive sense. For example:

```
library(MASS) # Contains example data set

# Relevel
survey$Smoke <- factor(survey$Smoke,
                       levels = c('Never', 'Occas', 'Regul', 'Heavy'))
survey$Exer <- factor(survey$Exer, levels = c('None', 'Some', 'Freq'))

# Create contingency table
smoking <- table(survey$Smoke, survey$Exer)
```