Microdata R Code

Set the working directory

The default working directory for an RMarkdown file is the location of the file (e.g. your "Downloads" folder if you just downloaded and opened it). To save workshop files to a different location, in RStudio go to *File* -> *Save As* -> and choose a location you will remember.

Install and load required packages

```r
required_packages <- c("haven", "dplyr", "srvyr", "gtsummary", "ggplot2", "ggpubr", "cardx")

# Loop through each required package and install any that are missing
for (package in required_packages) {
    if (!package %in% installed.packages()) {
        install.packages(package)
    }
}

#Load required packages for use in this R session (HIDE OUTPUT)
library(haven)     #to import SPSS .sav file
library(dplyr)     #for data manipulation
library(srvyr)     #survey-specific functions
library(gtsummary) #create summary tables
library(ggplot2)   #create plots
library(flextable) # create tables
library(officer)   # manipulate word docs and power points
library(cardx)
```

Download and unzip the microdata

This workshop uses Public Use Microdata Files (PUMFs) from the *Canadian Tobacco and Nicotine Survey* (CTNS). PUMFs for the CTNS and other Statistics Canada surveys are available in Abacus, UBC Library's data repository (https://abacus.library.ubc.ca/).

Survey data can be downloaded by visiting Abacus with a browser, but R can automate the process using the Abacus API. Each Abacus file has a persistent identifier called a *handle*. Listed below are the CTNS data files used in this example. (Links are to the Abacus records where you'll also find codebooks and user guides.)

Survey	File description	File name	Handle
CTNS 2020	Microdata in SPSS .sav format	CTNS_2020_PUMF_SPSS_sav.zip	11272.1/AB2/UYC0Z8/XVITQW

Survey	File description	File name	Handle
CTNS 2022	Microdata in SPSS .sav format	CTNS_2022_SPSS_SAV.zip	11272.1/AB2/PWWFK3/4K96XZ

Run the code below to download and unzip data files for the 2020 and 2022 survey years

```
download.file("https://abacus.library.ubc.ca/api/access/datafile/:persistentId?persistentId=hdl:11272.1/

download.file("https://abacus.library.ubc.ca/api/access/datafile/:persistentId?persistentId=hdl:11272.1/

unzip("CTNS_2020.zip")
unzip("CTNS_2022.zip")
```

Look in the RStudio *Files* tab in the bottom right of the screen. You should see the unzipped ".sav" files in your working directory.

Read in your sav files

The *read_sav* function from the *haven* package imports SPSS .sav files as *data.frames*, which are similar to spreadsheets. It also imports variable and value labels to make the data easier to work with.

```
#Import the .sav files and store them as 'ctns2020' and 'ctns2022'
data2020 <- read_sav("ctns_2020_pumf_eng.sav")
data2022 <- read_sav("ctns_2022_pumf.sav")
```

It is important to check that your data imported correctly. Click 'data2020' and 'data2022' in the environment pane (top right) to view the imported data. Is it what you expect?

Why use sav files?

If we compare the codebook to the data imported into R as a sav file, we can see that the codebook variables have a description directly in the header of the dataframe. If we were using a plain text format, we would loose this information.

Set up your survey data for analysis

After confirming that the data imported correctly we can perform other other operations to prepare the data for analysis.

> **Note:** The code below uses the *pipe* operator from the *dplyr* package to perform multiple functions in sequence. The *%>%* at the end of each lines tells R to take the output of that line and "pipe" it into the next line for further processing.

```
ctns2020 <- data2020 %>%
  as_factor() %>%   #for better labels and data handling
  droplevels() %>%  #remove levels that have no data (tidier tables)
  as_survey(weights=WTPP)  #treat as a survey with weight variable WTPP

ctns2022 <- data2022 %>%
  as_factor() %>%
  droplevels() %>%
  as_survey(weights=WTPP)
```

srvyr package

In the code above we used the `as_survey` function in the *srvyr* package. We set the weight to be the WTPP variable. After this point, the data will automatically be weighted for graphs and analyses.

Identify variables for analysis

Codebooks help you identify variables for your analysis. Our example uses the variables below but you're welcome to experiment with others during practice.

> **Note:** Variables are not always consistent between survey years: names may change and variables may be added or removed.

CTNS 2020

variable	notes
GENDER	binary variable, confuses gender and sex (Female/Male)
AGEGROUP	age groups from 15 to 65+
HHLDSIZE	household size from 1 to 5+
DV_SSR	smoking status (current/former/never) *DV_SS in 2022*
PROV_C	province

CTNS 2022

variable	notes
GENDER	acknowledges non-binary identities but groups them for privacy (Women+/Men+)
AGEGROUP	age groups from 15 to 65+
HHLDSIZE	*not available in 2022 PUMF*
DV_SS	smoking status (current/former/never) *DV_SSR in 2020*
PROV_C	province

Create summary tables

gtsummary package

There are many ways to create tables in R. The `tbl_svysummary` function from the *gtsummary* package takes advantage of the survey weight we configured earlier and produces easy-to-read tables with little effort.

Here's a sample table with one variable…

```r
#Table estimating population by smoking status, CTNS 2020
tbl_svysummary(ctns2020, include=DV_SSR)
```

Characteristic	**N = 31,306,624**[1]
DV_SSR	

Current smoker	3,228,743 (10%)
Former smoker	7,746,431 (25%)
Never smoked	20,305,439 (65%)
Unknown	26,011

[1] n (%)

…and another with two variables:

```r
#Table estimating population by smoking status and province, CTNS 2020
tbl_svysummary(ctns2020, include=DV_SSR, by=PROV_C)
```

Characteristic	**Newfoundland and Labrador** N = 440,023[1]	**Prince Edward Island** N = 132
DV_SSR		
Current smoker	64,951 (15%)	15,196 (12%)
Former smoker	116,027 (26%)	35,852 (27%)
Never smoked	259,045 (59%)	80,177 (61%)
Unknown	0	891

[1] n (%)

Saving tables to a word document

HHLDSIZE

```r
# Create the summary table
summary_table <- tbl_svysummary(ctns2020, include = DV_SSR, by = PROV_C)

# Extract the data frame from the gtsummary object
summary_df <- summary_table %>%
  as_tibble() %>%
  select(-starts_with("row_type"))

# Convert the data frame to a flextable
summary_flextable <- flextable(summary_df)

# Create a Word document and add the flextable
doc <- read_docx() %>%
  body_add_flextable(summary_flextable)

# Save the Word document
print(doc, target = "summary_table.docx")
```

Create plots in ggplot2

Formatted tables are good for presentations but unformatted statistics are easier to plot. The workflow below uses functions from three packages to summarize and present survey data in bar graphs (*dplyr*, *srvyr*, and *ggplot2*).

Let's plot the percentages from the table about smoking status and household size.

Step 1: Create a summary table (Household size and smoking status) and store it as *plot1_data*

```
plot1_data <- ctns2020 %>%
  filter(!is.na(HHLDSIZE) & !is.na(DV_SSR)) %>%  #remove NA values
  group_by(HHLDSIZE, DV_SSR) %>%  #group by household size, then smoking status
  summarize(percent = survey_prop()*100) #calculate percentages for each grouped value
```

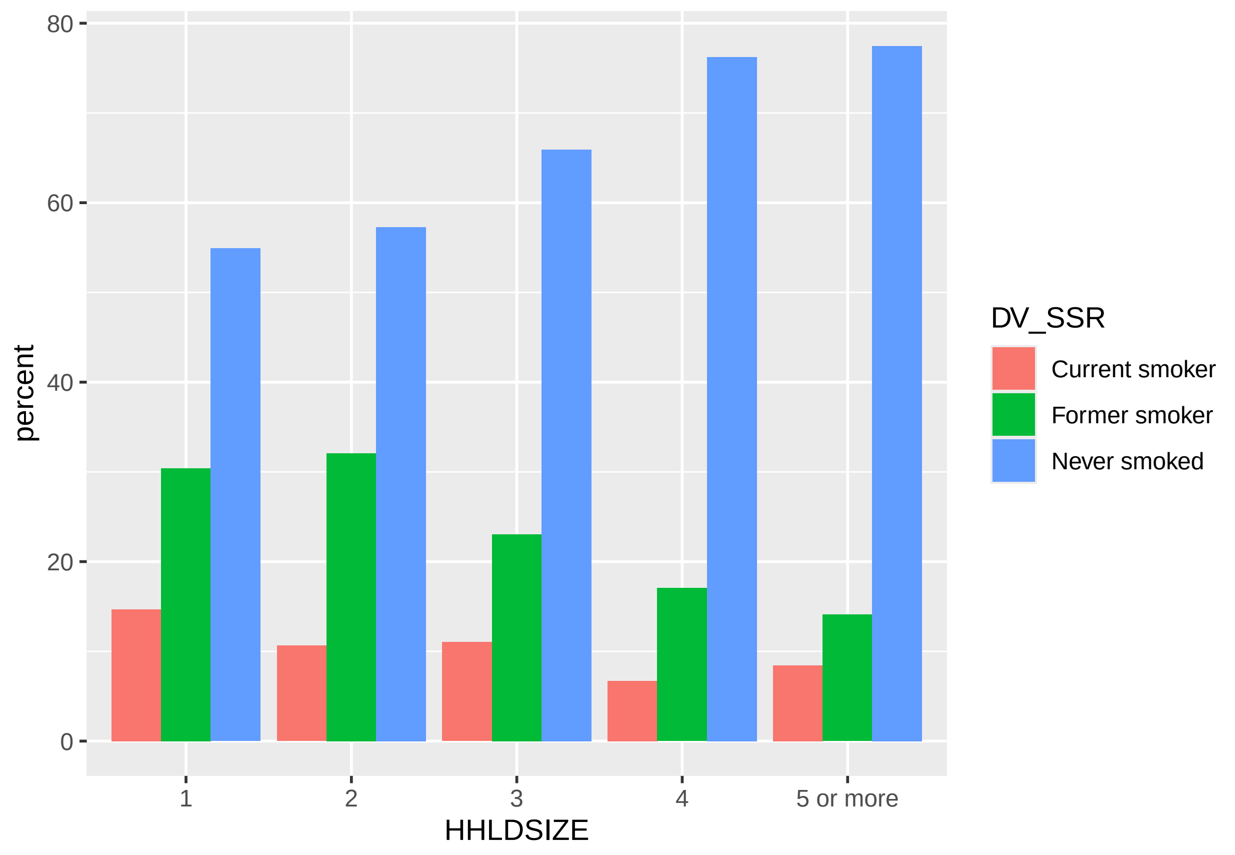
```
## When ‘proportion‘ is unspecified, ‘survey_prop()‘ now defaults to ‘proportion = TRUE‘.
## i This should improve confidence interval coverage.
## This message is displayed once per session.
```

Click *plot1_data* in the environment pane (top right) and see how percentages are calculated. (They total 100% for each household size).

Step 2: pipe the data into ggplot2

ggplot2 is a powerful and popular package for creating plots. Key components of a ggplot command include the *aes* function (identifies the variables) and the *geom* function (sets the plot type). There are many, many other optional components to customize the plot (see ggplot documentation).

```
plot1_data %>%
  ggplot(aes(y=percent, x=HHLDSIZE, fill=DV_SSR))+
  geom_col(position="dodge")
```



Current smoker percentage by age group, 2020 and 2022

This graph will compare the proportion of current smokers across two survey years, within each age group.

The structure of the command is similar to what we've seen already, but there are new components to data calculation step:

1. Specify *vartype="ci"* to calculate 95% confidence intervals
2. After calculating percentages, keep only the "Current smoker" rows
3. Add a *year* variable to distinguish between survey years
4. Rename the smoking status variable so it's the same for both years

Step 1: calculate 2020 values

```
plot2_data_2020 <- ctns2020 %>%
# filter(!is.na(DV_SSR)) %>%
  group_by(AGEGROUP, DV_SSR) %>%
  summarize(percent = survey_prop(vartype="ci")*100) %>%
  filter(DV_SSR == 'Current smoker') %>%
  mutate(year = "2020") %>%
  rename(smoking_status = DV_SSR)
```

Step 2: calculate 2022 values and join with 2020

```
plot2_data_2022 <- ctns2022 %>%
  filter(!is.na(DV_SS)) %>%
  group_by(AGEGROUP, DV_SS) %>%
  summarize(percent = survey_prop(vartype="ci")*100) %>%
  filter(DV_SS == 'Current smoker') %>%
  mutate(year = "2022") %>%
  rename(smoking_status = DV_SS)
```

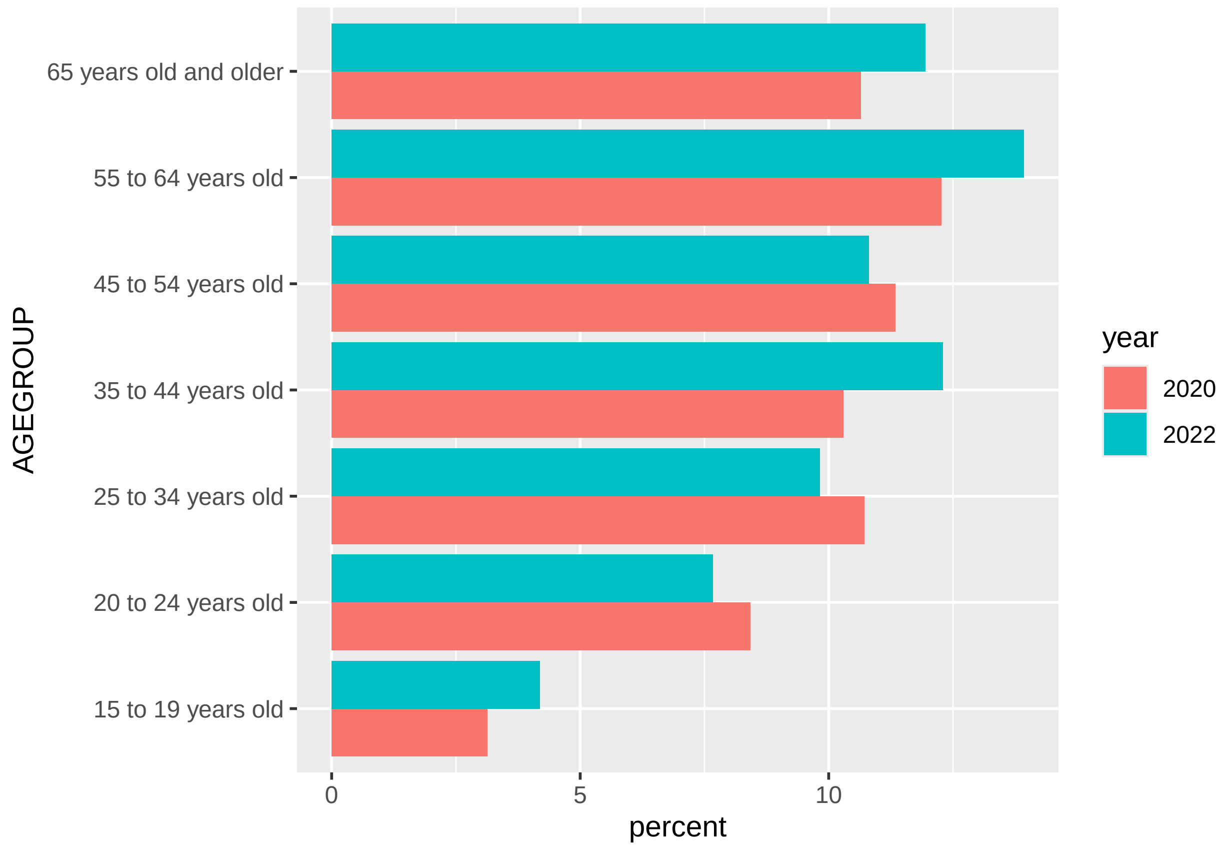
Step 3: join the 2020 and 2022 data together

```
plot2_data <- full_join(plot2_data_2020, plot2_data_2022)
```

```
## Joining with `by = join_by(AGEGROUP, smoking_status, percent, percent_low,
## percent_upp, year)`
```

Step 4: plot the data

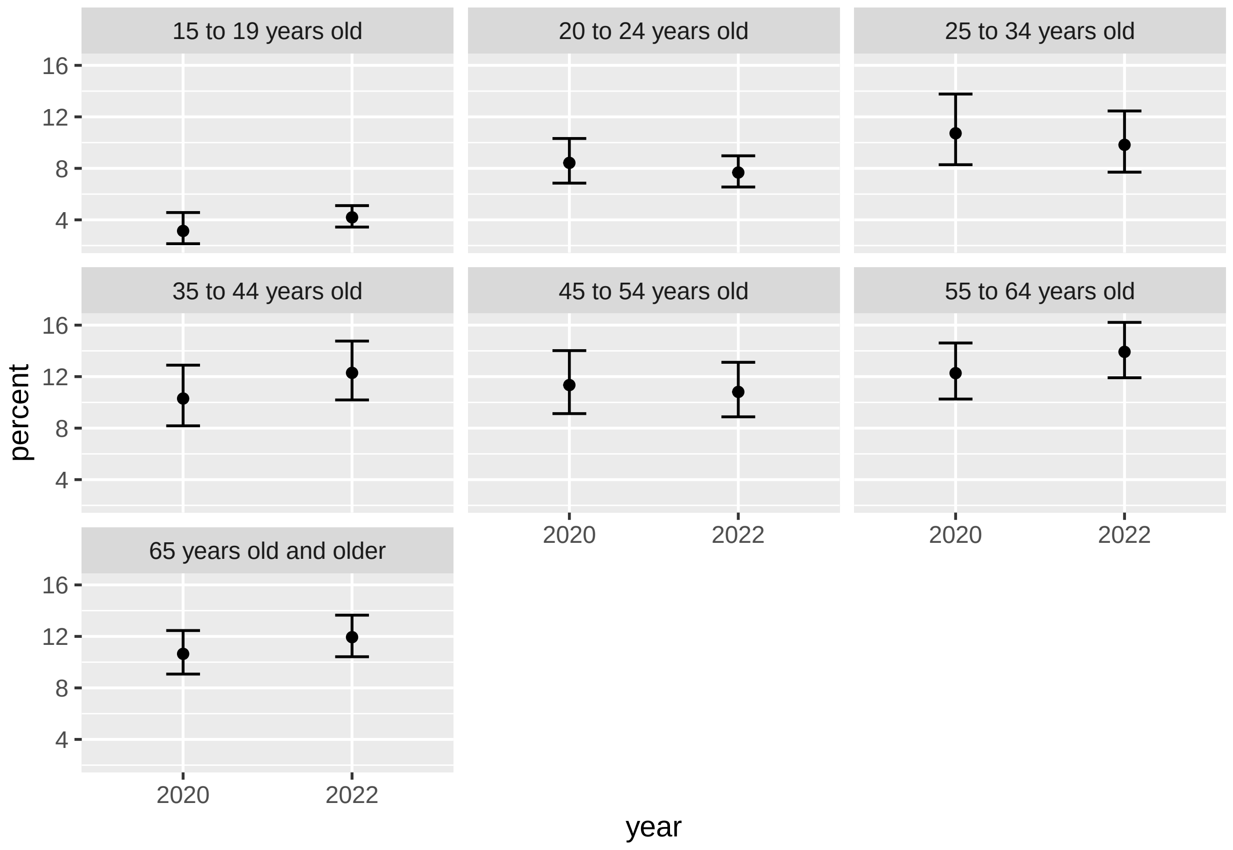
```
ggplot(plot2_data, aes(x=percent, y=AGEGROUP, fill=year))+
  geom_col(position="dodge")
```



The bar graph shows relatively small changes between the years within each age group. The percentage of current smokers in some age groups went up, in others it went down, and there's no strong pattern. Is the direction of change significant?

When calculating percentages we added the *vartype="ci"* parameter to generate confidence intervals. These can be plotted on a line graph that shows the current smoker proportions for each age group in black, along with a semi-transparent (*alpha=0.5*) gray band depicting the confidence intervals.

```r
ggplot(plot2_data, aes(x=year, y=percent, group=AGEGROUP))+
  geom_errorbar(aes(ymin=percent_low, ymax=percent_upp), width=0.2) +
  geom_point()+
  facet_wrap(vars(AGEGROUP))
```

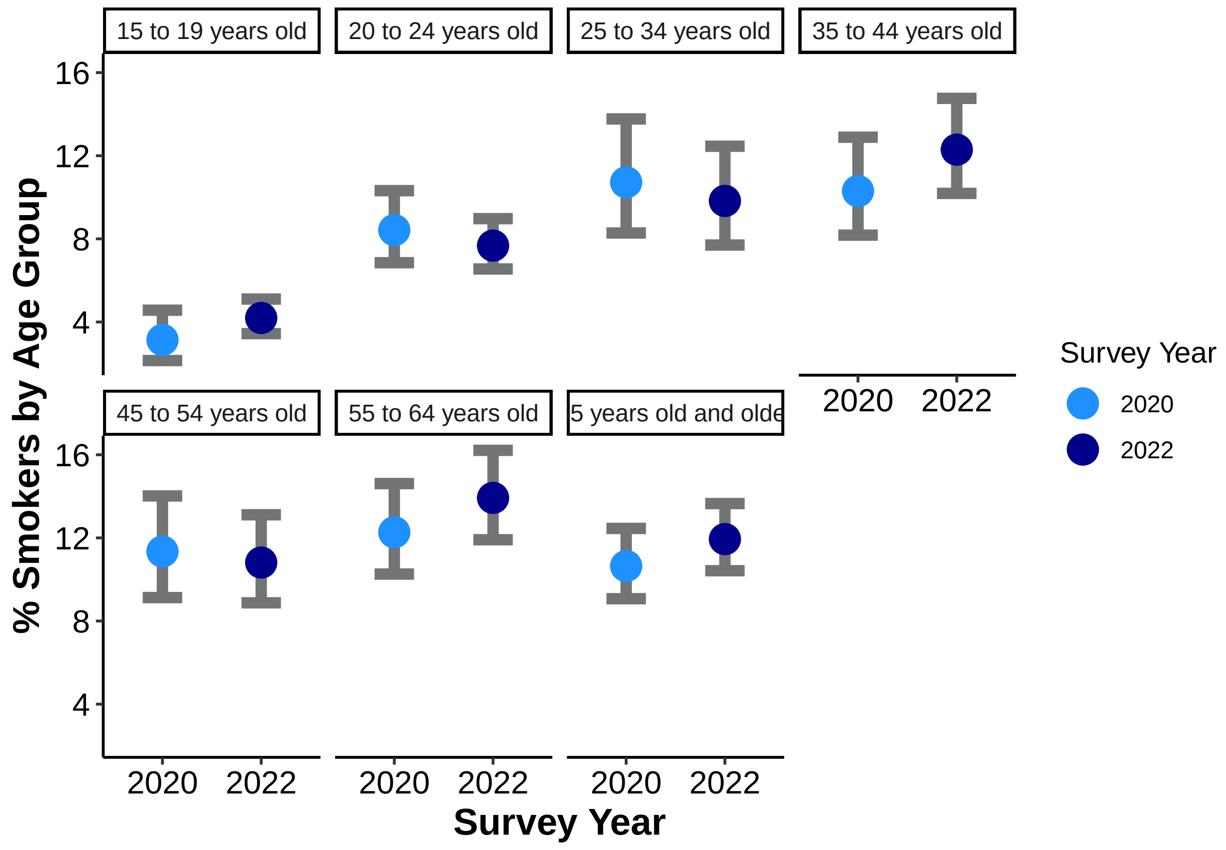


The graph makes it clear that the confidence intervals are relatively wide for some age groups: we can't say with 95% certainty that the actual direction of change in the percentage of smokers in Canada matches what we see in this survey sample.

Step 5: Customize the plot

This is meant as an illustrative example of how customization plots made with ggplot2 are. There are many options for customization, so this is by no means extensive.

```
ggplot(plot2_data, aes(x=year, y=percent, group=AGEGROUP))+
  geom_errorbar(aes(ymin=percent_low, ymax=percent_upp),
                width=0.4, ## change the witdh of the top and bottom horizontal line of the error bar
                color="#737574", ## change the color of error bars (hex color)
                linewidth=2) + ## linewidth makes the error bar lines wider or thinner
  geom_point(cex=5, aes(colour = year))+ # cex makes the points bigger or smaller, color sets the point
  facet_wrap(vars(AGEGROUP), ncol=4)+ ## setting the column number to distribut the facest
  theme_classic()+ ## change the background
  theme(axis.text=element_text(colour = "black", size = 12), ## set the axis text color and make the te
        axis.title = element_text(face = "bold", size = 14, colour = "black")) + ## set title size and
  labs(x="Survey Year", y="% Smokers by Age Group", color="Survey Year")+ # custom axis lables
  scale_color_manual(values=c("dodgerblue1", "blue4")) ## differnt point colours by year
```



Step 6: Export the plot

Now that we have a pretty plot, we can save it.

This can be done manually from the plots tab. The plot area can be stretched and re-sized by stretching the plot area, then by clicking *Export* and saving the plot as an image or from there.

To save time and export consistently sized plots, if is often better to use the package ggpubr.

```
ggsave(filename="smokers_by_age_and_year.pdf", width=10, height=8, units="in")
```