Data Visualization

Field Coordinator Training - R Track

Luiza Andrade, Leonardo Viotti & Rob Marty

June 2018

i2i DIME TRANSFORM DEVELOPMENT

WORLD BANK GROUP

1. Introduction
2. Exploratory graphs - Base plot
3. ggplot - Introduction
4. ggplot - Data structure
5. ggplot - Aesthetics
6. ggplot - Titles and labels
7. Saving a plot
8. Interactive graph

Outline

1. Introduction
2. Exploratory graphs - Base plot
3. ggplot - Introduction
4. ggplot - Data structure
5. ggplot - Aesthetics
6. ggplot - Titles and labels
7. Saving a plot
8. Interactive graph
9. References and recommendations
10. Appendix

Introduction

In this session, you'll learn how to use R to produce insightful, meaningful and (hopefully) nice-looking graphs. In particular, you'll use a package called `ggplot2`.

Similarly, to the previous session, you can find some references at the end of this presentation that include a more comprehensive discussions on data visualization.

Introduction

Before we start, let's make sure we're all set:

1. Make sure you the packages `ggplot2` and `plotly` are installed and loaded.
    - Preferably, do this by adding them to `packages` vector of your Master script.
2. Load the `lwh_clean.csv` data set that were created in the last session.
    - Run the Master script to load all the necessary packages and set file paths.
    - Remeber to disable the `PACKAGES` switch in the Master if you already installed them. This will save you a lot of time.

Introduction

```r
# Install packages
install.packages(c("ggplot2","plotly"),
                 dependencies = TRUE)

# Load packages
library(ggplot2)
library(plotly)

# Load CSV data
projectFolder  <- "YOUR/FOLDER/PATH"
finalData <- file.path(projectFolder,
                       "DataWork","DataSets","Final")
lwh <- read.csv(file.path(finalData,"lwh_clean.csv"),
                header = TRUE)
lwh_simp <- read.csv(file.path(finalData,"lwh_simp.csv"),
                     header = TRUE)
```

Introduction

In our workflow there are usually two distinct uses for plots:

1. **Exploratory analysis:** Quickly visualize your data in an insightful way
   - We'll do this using R's base plot, that allows you to quickly create some basic plots
2. **Publication/Reporting:** Make pretty graphs for a presentation, a project report, or to just show your boss something other than the laziest graph you could come up with
   - We'll do this using `ggplot`, a flexible and powerful tool to plot beautiful graphs
   - `ggplot`'s syntax is more complicated, but it's easier to make your graphs look good with it

Outline

1. Introduction
2. Exploratory graphs - Base plot
3. ggplot - Introduction
4. ggplot - Data structure
5. ggplot - Aesthetics
6. ggplot - Titles and labels
7. Saving a plot
8. Interactive graph
9. References and recommendations
10. Appendix

Exploratory graphs - Base plot

For exploratory plots, we're going to use Base plot. It is easy to use and can produce useful graphs with very few lines of code.

Exploratory graphs - Base plot

Exercise 1:

Plot the lwh_simp data set you constructed in the previous session using the plot() function.

1. If you don't have the code from the previous session. Here it is:

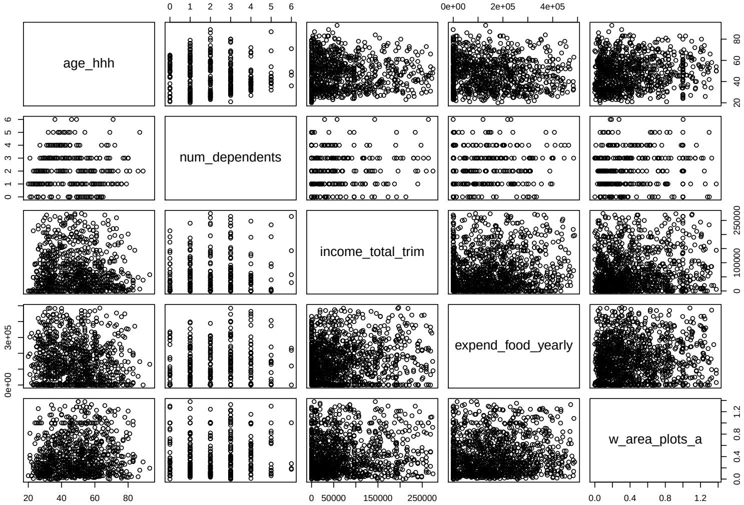
```r
covariates <- c("age_hhh",
                "num_dependents",
                "income_total_trim",
                "expend_food_yearly",
                "w_area_plots_a")

lwh_simp <- lwh[,covariates]
```

2. Now, pass the name of the data set as the only argument for the plot function

Exploratory graphs - Base plot

```r
plot(lwh_simp)
```



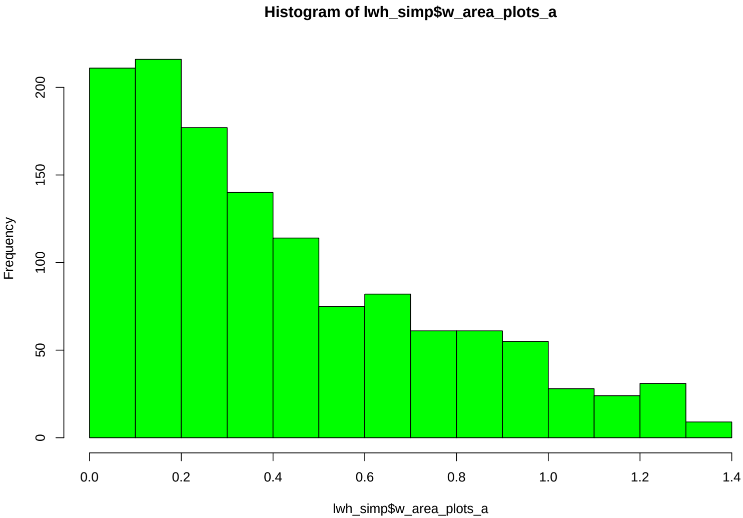
Exploratory graphs - Base plot

Exercise 2

1. Create a histogram of variable `w_area_plots_a` in data frame `lwh_simp`. Use the `hist()` function for that.
2. Try to set the color as `"green"`. Use the help if you are not sure how to do it.

Exploratory graphs - Base plot

```
hist(lwh_simp$w_area_plots_a, col = "green")
```



Exploratory graphs - Base plot

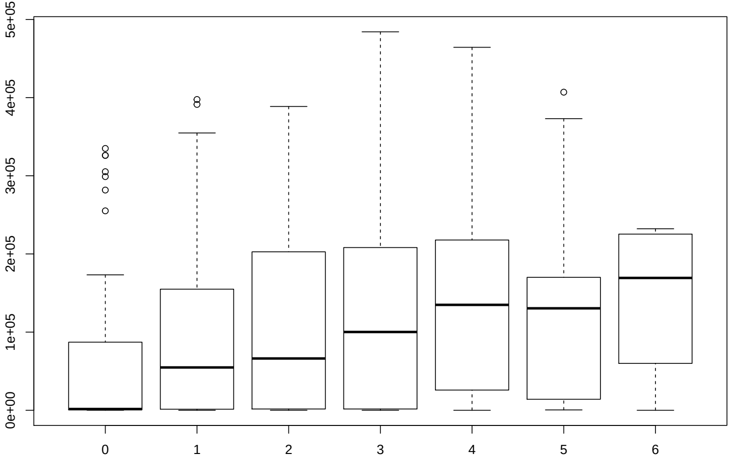
Exercise 3

Use the `boxplot()` function to see a description of the yearly food expenditure by number of dependents.

- TIP: The syntax for this function is `variable_to_be_plotted` ~ `variable_to_plot_over`.

Exploratory graphs - Base plot

```
boxplot(lwh_simp$expend_food_yearly ~ lwh_simp$num_dependents)
```



Outline

1. Introduction
2. Exploratory graphs - Base plot
3. ggplot - Introduction
4. ggplot - Data structure
5. ggplot - Aesthetics
6. ggplot - Titles and labels
7. Saving a plot
8. Interactive graph
9. References and recommendations
10. Appendix

ggplot - Introduction

The ggplot2 package is an implementation of the theoretical framework for the construction of quantitative graphs developed in The Grammar of Graphics, a book by Leland Wilkinson.

It is a powerful and easy to use tool (once you understand its logic) that produces complex and multifaceted plots.

ggplot - Introduction

There are a few reasons why we use ggplot:

- It is generally easier to customize and tailor plots to your needs.
- It works great with almost any kind of plot, including maps, with little variation in syntax. The notable exception is 3D plotting, which can be done, but it's not as straightforward.
- It looks good.

ggplot - Grammar

Here is a list of ggplot2 most commonly used "grammar" elements:

Element	Description
Data	The dataset(s) being used.
Aesthetics	How your data is maped. For example what goes in the X and Y axis, size, shape and color.
Geometries	The visual elements used to represent the data (e.g. lines, dots, bars, etc.)
Themes	All non-data formating.

ggplot - Data structure

As almost everything in R, ggplot is very flexible and can be used in different ways. It is easier to work, however, if your data is in a specific format.

- Data has to be a data frame.
- Long format, especially categorical values.
  - It's better to have one variable with all the categories than a set of dummies.
- Labelled factors.

This is very similar to Stata, with the advantage of not needing to preserve and restore anything.

ggplot - Data structure

Here's an example of how an aggregated (collapsed) data set should look like:

```
##    year gender income_total_trim
## 1  2012 Female          34331.47
## 2  2013 Female          57123.69
## 3  2014 Female          48560.15
## 4  2016 Female          53980.17
## 5  2018 Female          70238.30
## 6  2012   Male          40222.72
## 7  2013   Male          77518.86
## 8  2014   Male          78886.26
## 9  2016   Male          85901.84
## 10 2018   Male         100867.57
```

ggplot - Introduction

Exercise 4

First, let's create a very simple plot with the yearly expenditure on food on the Y axis and age of the household head on the X axis.

1. Use the ggplot function to store your plot in an object called plot1
   - Since it's the first one, Here's the code:

```
plot1 <- ggplot(data = lwh,
                aes(y = w_area_plots_b,
                    x = w_gross_yield_b))
```

2. Now, try to print your plot. What happens?
   - You can do it by using the print() function or just typing the object name in the console

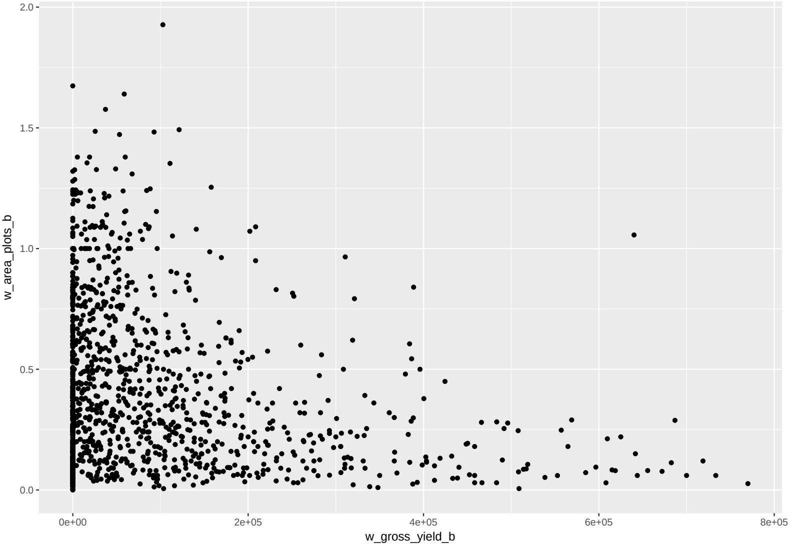
ggplot - Introduction

We mapped our variables, but without a geometry ggplot doesn't know how to represent the data.

Let's add a geometry

```
ggplot(data = lwh,
       aes(y = w_area_plots_b,
           x = w_gross_yield_b)) +
  geom_point()
```

ggplot - Introduction



Outline

1. Introduction
2. Exploratory graphs - Base plot
3. ggplot - Introduction
4. ggplot - Data structure
5. ggplot - Aesthetics
6. ggplot - Titles and labels
7. Saving a plot
8. Interactive graph
9. References and recommendations
10. Appendix

ggplot and aggregate()

- Since ggplot usually works better with data in a specific format, there are a few functions particularly useful to manipulate data
- Here, we'll use the aggregate() function
- aggregate() works similarly to collapse in Stata

ggplot and aggregate()

To create a line plot with ggplot, we will create a new data set containing the average of the total house hold annual income by year.

```
# Aggregate anual income by year
anualInc <-
    aggregate(x   = lwh["income_total_trim"], # data.frame
              by  = list(year = lwh$year), #list
              FUN = mean, na.rm = T) # function
```

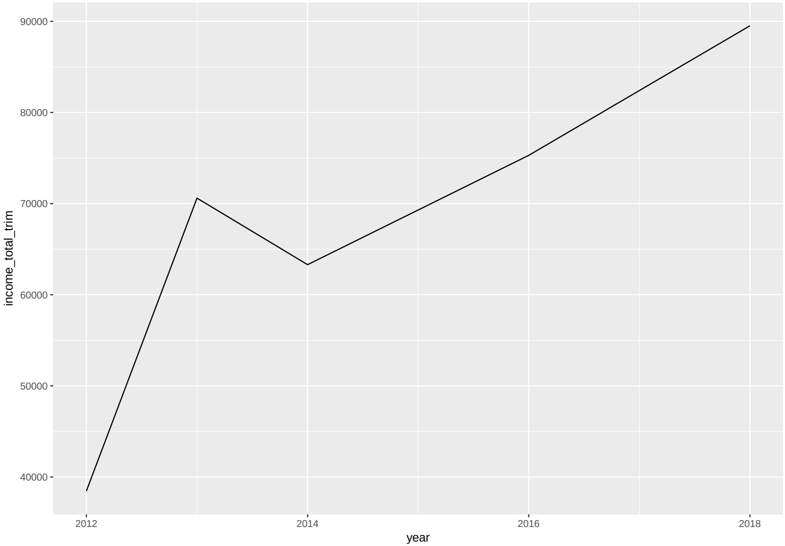
Exercise 5

1. Use the ggplot() function with the anualInc as the data argument, income_total_trim as y and year as x in aes()
2. This time add geom_line()

ggplot and aggregate()

```
# Plot code with arguments in geom
ggplot() +
    geom_line(data = anualInc,
              aes(y = income_total_trim,
                  x = year))
```

ggplot and aggregate()



ggplot and aggregate()

Ok, that worked, but was a really boring plot. Let's do a cooler one.

Exercise 6: Part 1

Use the aggregate() function to average total house hold annual income, now, by two variables: year and treatment_hh.

- TIP: add an argument to the list() function in the by argument. Like this:

```
by = list(Year = lwh$year,
          Treatment = lwh$treatment_hh)
```

ggplot and aggregate()

```r
# Aggregate data set
anualIncGen <- aggregate(x= lwh["income_total_win"],
                         by = list(year = lwh$year,
                                   treatment =  lwh$treatment_hh),
                         FUN = mean, na.rm = T)

anualIncGen
```

```
##    year treatment income_total_win
## 1  2012   Control         47958.37
## 2  2013   Control         63482.25
## 3  2014   Control         59232.51
## 4  2016   Control         71106.46
## 5  2018   Control         98294.61
## 6  2012 Treatment         41459.83
## 7  2013 Treatment        104023.15
## 8  2014 Treatment         80672.20
## 9  2016 Treatment         85192.96
## 10 2018 Treatment         99510.29
```

ggplot and aggregate()

Exercise 6: Part 2

Now, use ggplot as in the previous exercise to plot the aggregated data frame.

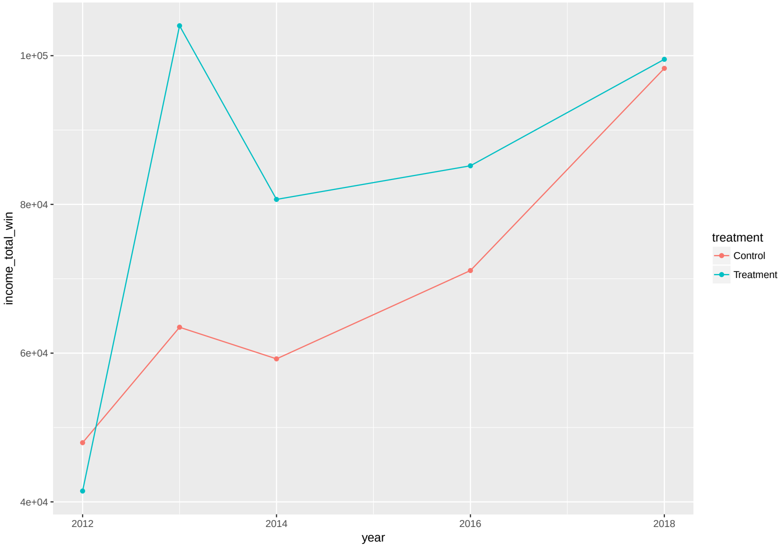
1. This time set the color and group arguments in the aes() function as the treatment variable.
2. Finally, add both line and point geoms.

ggplot and aggregate()

```r
# Aggregate data set
anualIncGen <- aggregate(x= lwh["income_total_win"],
                         by = list(year = lwh$year,
                                   treatment =  lwh$treatment_hh),
                         FUN = mean, na.rm = T)

# Plot aggregated data set
ggplot(anualIncGen,
       aes(y = income_total_win,
           x = year,
           color = treatment,
           group = treatment)) +
  geom_line() +
  geom_point()
```

ggplot and aggregate()



Outline

1. Introduction
2. Exploratory graphs - Base plot
3. `ggplot` - Introduction
4. `ggplot` - Data structure
5. `ggplot` - Aesthetics
6. `ggplot` - Titles and labels
7. Saving a plot
8. Interactive graph
9. References and recommendations
10. Appendix

ggplot - Aesthetics

In ggplot , the aesthetic element defines how data is represented in the aesthetics (e.g. color, shape and size) of geometric objects (e.g. points, lines and bars).

Exercise 7: Part 1

1. Create a data frame containing only 2018 observations of lwh in the Rwamangana 35 site.

   - TIP: This time we want to keep only observations (lines) that meet a certain condition. Use the left argument of the brackets operators. Like this:

```
lwh[CONDITION,]
```

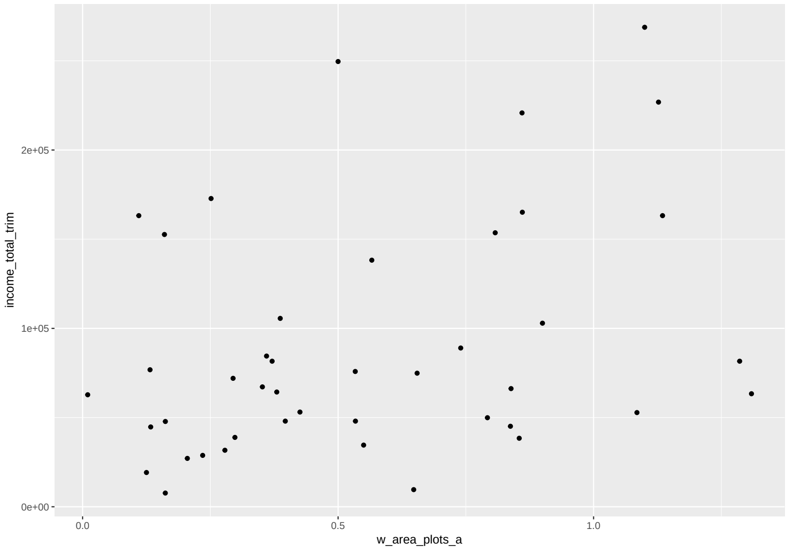
2. Use ggplot() to scatter plot area in season A and total annual income

ggplot - Aesthetics

```r
# Subset lwh to 2018 and Rwamangana 35
lwh_s <- lwh[lwh$year == 2018 & lwh$site_code == "Rwamangana 35", ]

# Plot
ggplot(lwh_s, aes(x = w_area_plots_a,
                  y = income_total_trim)) +
    geom_point()
```

ggplot - Aesthetics



ggplot - Aesthetics

Now add the women's dietary diversity score[1] to aes().

Exercise 7: Part 2

1. Use the wdds_score varaible as color in aes().
   - TIP: this is is a categorical variable, but it is stored in numeric format. Use the factor() function. Like this:

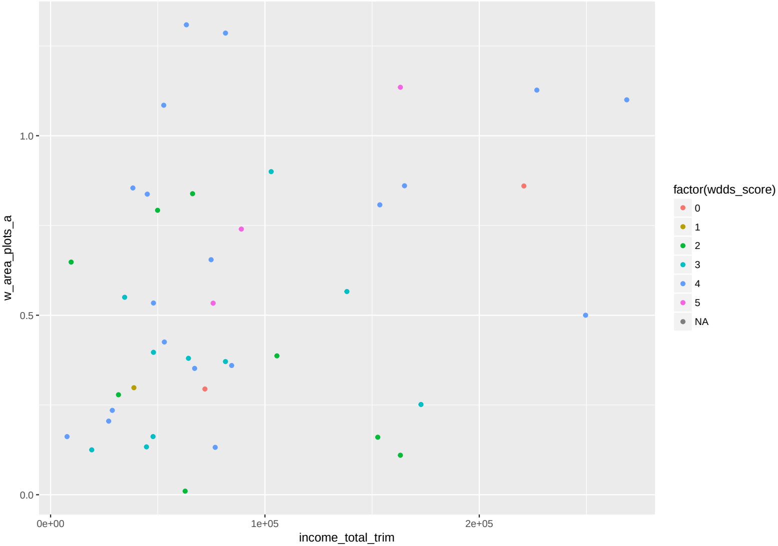
```
color = factor(wdds_score)
```

[1] Women's dietary diversity score is based on the questions asked specifically to an adult female respondent about her food consumption. We ask if she ate 16 food items categorized into 9 groups. Each group get 1 if any of the food belonging to the group is consumed. The final score is a simply sum of the categories.

ggplot - Aesthetics

```
# Plot code
ggplot(data = lwh_s,
       aes(y = w_area_plots_a,
           x = income_total_trim,
           color = factor(wdds_score))) +
    geom_point()
```

ggplot - Aesthetics



ggplot - Aesthetics

Now we will combine different arguments of `aes()`

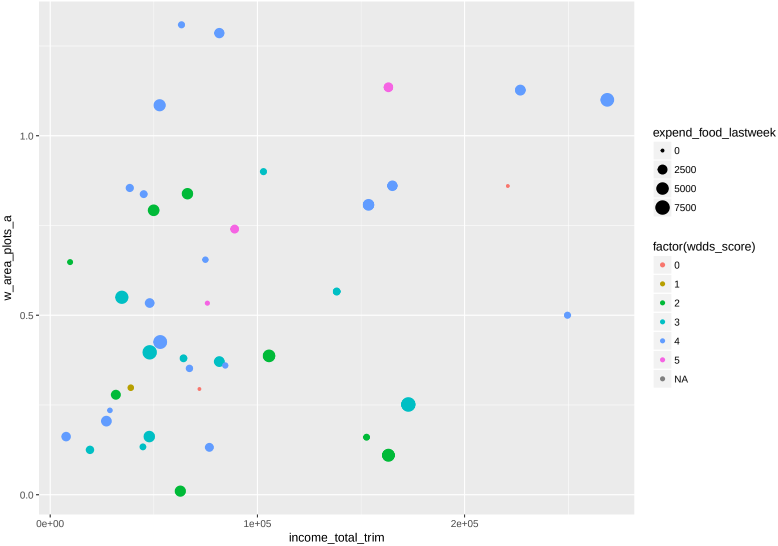
Exercise 7: Part 3

1. Add expenditure on food last week variable, `expend_food_lastweek` to the `size` argument of `aes()` in your last graph.

ggplot - Aesthetics

```
# Plot code
ggplot(data = lwh_s,
       aes(y = w_area_plots_a,
           x = income_total_trim,
           color = factor(wdds_score),
           size = expend_food_lastweek)) +
    geom_point()
```

ggplot - Aesthetics

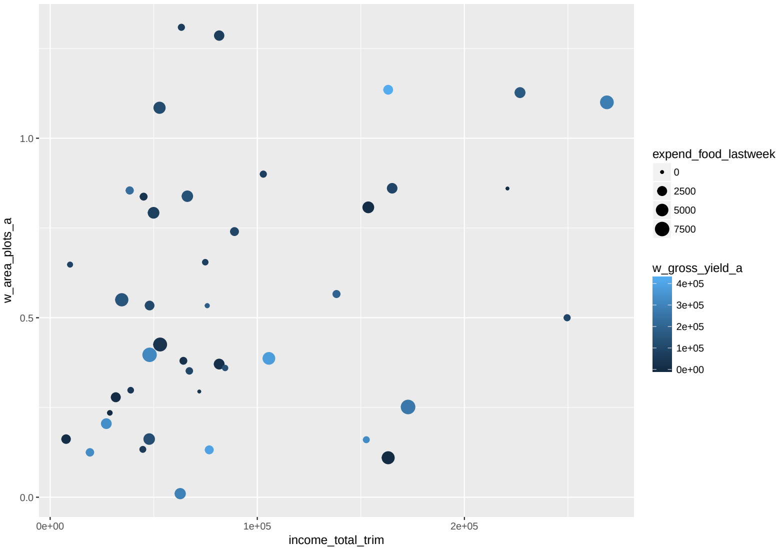


ggplot - Aesthetics

- Because we used a factor variable as the color in the aesthetic, it displayed different categories clearly
- If we had used a numeric variable instead, it would have created a gradient
- You'll not create this for time's sake, but here's how it would look:

```r
# Plot code
ggplot(data = lwh_s,
       aes(y = w_area_plots_a,
           x = income_total_trim,
           color = w_gross_yield_a,
           size = expend_food_lastweek)) +
    geom_point()
```

ggplot - Aesthetics



Outline

1. Introduction
2. Exploratory graphs - Base plot
3. ggplot - Introduction
4. ggplot - Data structure
5. ggplot - Aesthetics
6. ggplot - Titles and labels
7. Saving a plot
8. Interactive graph
9. References and recommendations
10. Appendix

ggplot - Titles and labels

Lastly, we can add titles, axis labels and format legends. This is done by adding additional layers to the plot.

To add a title and legend to the X and Y axis, we use the functions listed below. They all take a string scalar as argument.

- `ggtitle()` - adds a title to the plot
- `xlab()` - adds a label to X axis
- `ylab()` - adds a label to Y axis

ggplot - Titles and labels

To add legend titles, we will use two functions that control the formatting of the aesthetic. As they can take many different arguments, we use `name` argument sets legend title:

- `scale_color_discrete()` - formatting of the color aesthetic.
- `scale_size_continuous()` - formatting of the color aesthetic.

ggplot - Titles and labels

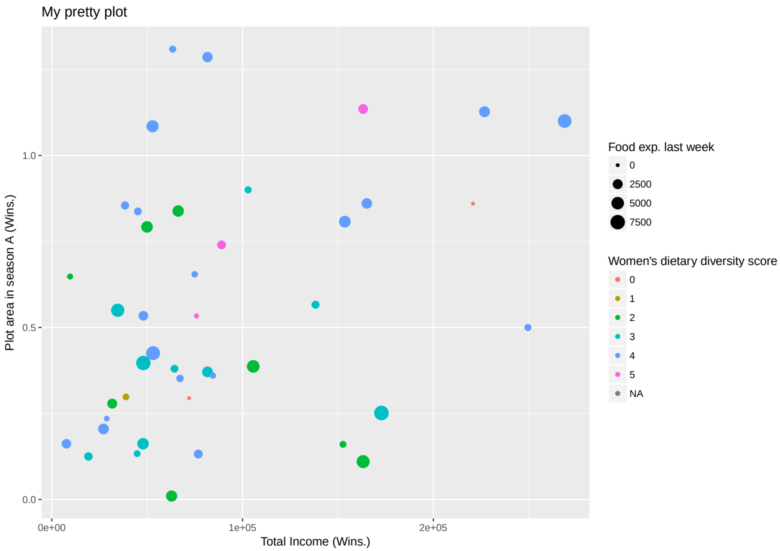
Exercise 8

1. Copy the code for the graph produced in the previous exercise.
2. Use the + symbol to add the layer.
3. `ggtitle()`, `xlab()` and `ylab()` take simple strings as inputs.
4. For `scale_color_continuous()` and `scale_size_continuous()` you need to specify `name` argument as the desired string.

ggplot - Titles and labels

```r
# A properly labelled plot
ggplot(lwh_s, aes(y = w_area_plots_a,
                  x = income_total_trim,
                  size  = expend_food_lastweek,
                  color = factor(wdds_score))) +
  geom_point() +
  ggtitle("My pretty plot") +
  xlab("Total Income (Wins.)") +
  ylab("Plot area in season A (Wins.)") +
  scale_color_discrete(name = "Women's dietary diversity score") +
  scale_size_continuous(name = "Food exp. last week")
```

ggplot - Titles and labels



Outline

1. Introduction
2. Exploratory graphs - Base plot
3. `ggplot` - Introduction
4. `ggplot` - Data structure
5. `ggplot` - Aesthetics
6. `ggplot` - Titles and labels
7. **Saving a plot**
8. Interactive graph
9. References and recommendations
10. Appendix

Saving a plot

Let's save the previous plot using the PDF graphics device

Exercise 9:

1. Open the PDF graphics device by using the `pdf()` function.
2. Set the `file` argument as your outputs path with the file name (including ".pdf")
3. Paste the code of the previous plot bellow the `pdf()` function.
4. Finally, use the `dev.off()` to close the device, i.e. tell R that you're finished plotting.

Saving a plot

```
# Open PDF graphics device
pdf(file = file.path(rawOutput, "plot1.pdf"))

 # Plot
 ggplot(data = lwh_s,
     aes(y = w_area_plots_a,
         x = income_total_trim,
         color = factor(wdds_score),
         size = expend_food_lastweek)) +
   geom_point() +
   ggtitle("My pretty plot") +
   xlab("Total Income (Wins.)") +
   ylab("Plot area in season A (Wins.)") +
   scale_color_discrete(name = "Women's dietary diversity score") +
   scale_size_continuous(name = "Food exp. last week")

# Close PDF graphics device
dev.off()
```

Outline

1. Introduction
2. Exploratory graphs - Base plot
3. `ggplot` - Introduction
4. `ggplot` - Data structure
5. `ggplot` - Aesthetics
6. `ggplot` - Titles and labels
7. Saving a plot
8. Interactive graph
9. References and recommendations
10. Appendix

Interactive graph

Now we'll use ggplotly() to create an interactive graph!

Bonus exercise - ggplotly

1. Load the `plotly` package
2. Store one the plots produced in the previous `ggplot` exercises in an object.
3. Pass that object as argument for the `ggplotly()` function

Interactive graph

```r
# Load package
library(plotly)

# Store graph in plot1
plot1 <- ggplot(data = lwh_s,
                aes(y = w_area_plots_a,
                    x = income_total_trim,
                    color = factor(wdds_score))) +
    geom_point()

# Use ggplotly to create an interactive plot
ggplotly(plot1)
```

Outline

1. Introduction
2. Exploratory graphs - Base plot
3. `ggplot` - Introduction
4. `ggplot` - Data structure
5. `ggplot` - Aesthetics
6. `ggplot` - Titles and labels
7. Saving a plot
8. Interactive graph
9. References and recommendations
10. Appendix

References and recommendations

Websites:

- The R Graph Gallery: https://www.r-graph-gallery.com/
- Gpplot official site: http://ggplot2.tidyverse.org/

Online courses:

- Johns Hopkins Exploratory Data Analysis at Coursera: https://www.coursera.org/learn/exploratory-data-analysis

Books:

- The grammar of graphics by Leland Wilkinson.
- Beautiful Evidence by Edward Tufte.
- R Graphics cook book by Winston Chang
- R for Data Science by Hadley Wickham andGarrett Grolemund

Outline

1. Introduction
2. Exploratory graphs - Base plot
3. `ggplot` - Introduction
4. `ggplot` - Data structure
5. `ggplot` - Aesthetics
6. `ggplot` - Titles and labels
7. Saving a plot
8. Interactive graph
9. References and recommendations
10. Appendix

Appendix

Slides that didn't make the cut

R common graphic devices

R has several built-in graphic devices. A graphic device is the medium where you visually represent a plot. Here are the most come:

- The standard screen device or the plot pane in RStudio.
- PDF file (file device).
- PNG (file device).
- JPEG (file device).

Differed ways of saving a plot

There are a couple of ways of saving plots in R:

- With ggplot you can use the `ggsave()` function that has a simple syntax and can do most of the dirty work for you.
- Choosing an specific graphics device (other than the default window). pdf(), png(), jpeg() and bmp() are functions that call graphic devices that save your plots in these formats. There are a few others built-in and you can check them by typing *?Devices*, and even more in CRAN.

Graphic systems

R has four main graphic systems:

- Base plot - most basic plotting system.
- Grid graphics - is a low-level graphics system mainly used by package developers.
- Lattice graphics - easy way to plot high dimensional data or many graphs at the same time.
- Ggplot2 - a flexible and powerful tool to plot beautiful graphs with intuitive syntax