Lesson 07: Basic customization

LSC 563: Data Visualization – Spring 2022

Welcome! Class Starts at 5:15

1

Check-In: Blackboard

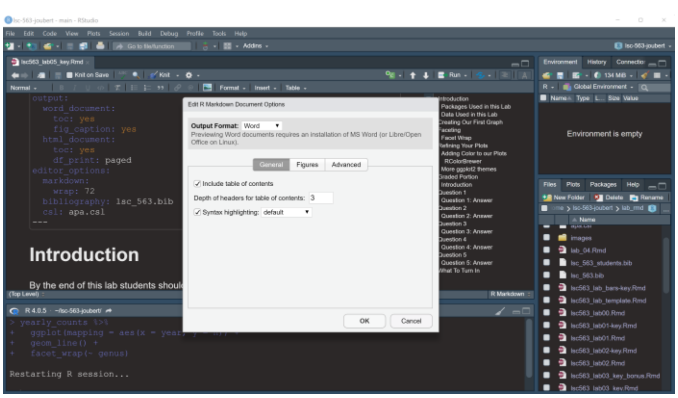
2

Check-In: Lessons

- Anything else?

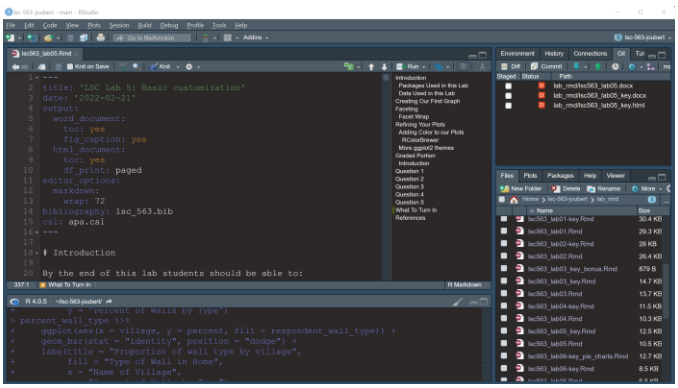
3

Check-In: Labs – RMD Options



4

Check-In: Labs – Bibliography



5

Grouped Data and the "Group" Aesthetic

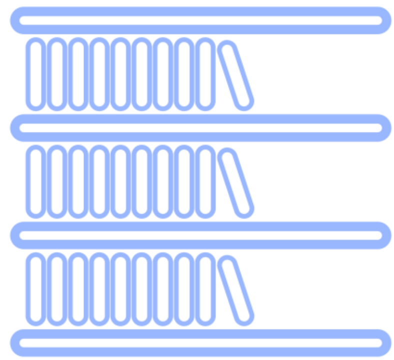
- Grouping: telling ggplot more about the internal structure of your data
- Faceting: breaking up your data into pieces for a plot
- Transforming: getting ggplot to perform some calculations on or summarize your data before producing the plot

Healy, 2018

6

Let's Load Our Libraries

- `library(gapminder)`
- `library(tidyverse)`
- `library(socviz)`
- `library(ggrepel)`
- `library(RColorBrewer)`



7

Grammar of Graphics

- Used by ggplot library
- Set of rules for producing graphics from data:
  - Taking pieces of data and mapping them to geometric objects (like points and lines)
  - Have aesthetic attributes (like position, color, and size)
  - Further rules for transforming the data if needed

Healy, 2018

8

Grammar of Graphics: Example - Data

- Gapminder dataset
- We need to map `year` to x and `gdpPercap` to y.
- How would we map this out in ggplot?

```
head(gapminder, 5)

## # A tibble: 5 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
```

Healy, 2018

9

Grammar of Graphics: Example - Data

- Gapminder dataset
- Plot the trajectory of life expectancy over time for each country in the data

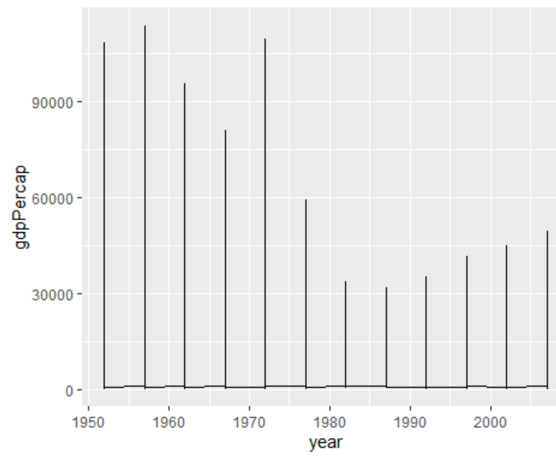
```
p <- ggplot(data = gapminder, mapping = aes(x = year,
y = gdpPercap))
```

Healy, 2018

10

Grammar of Graphics: Example - Data

- Wow, what happened?

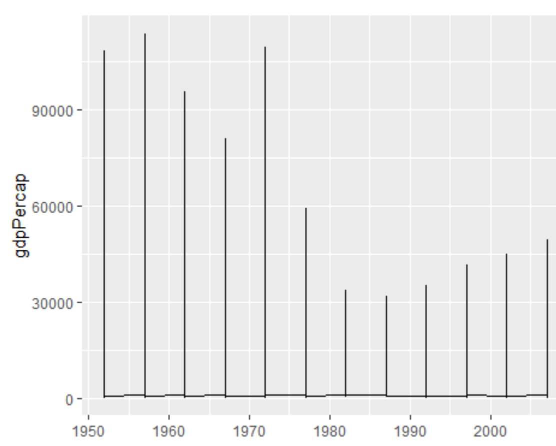


Healy, 2018

11

Grammar of Graphics: Example - Data

- ggplot made a good guess based on the structure of the data
- Need to tell ggplot that the yearly observations in the data are **grouped by country**
- `geom_line()` tries to join up all the lines for each particular year in the order they appear in the dataset

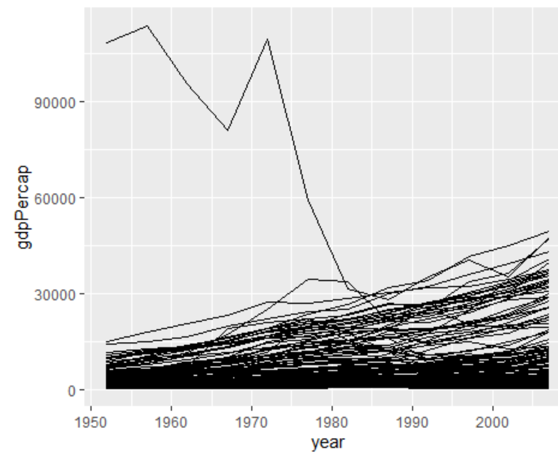


Healy, 2018

12

Grammar of Graphics: Example - Data

- Let us try this again, using the group aesthetic to tell ggplot explicitly about this country-level structure.
- How would we do this?



Healy, 2018

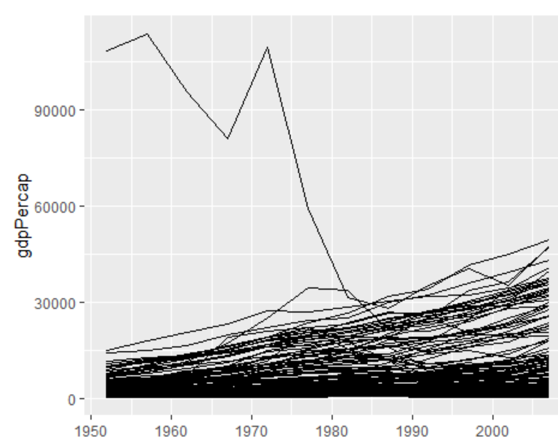
13

Grammar of Graphics: Example - Data

- Let us try this again, using the group aesthetic to tell ggplot explicitly about this country-level structure.

```
p +
  geom_line(aes(group = country))
```

- What do you think?

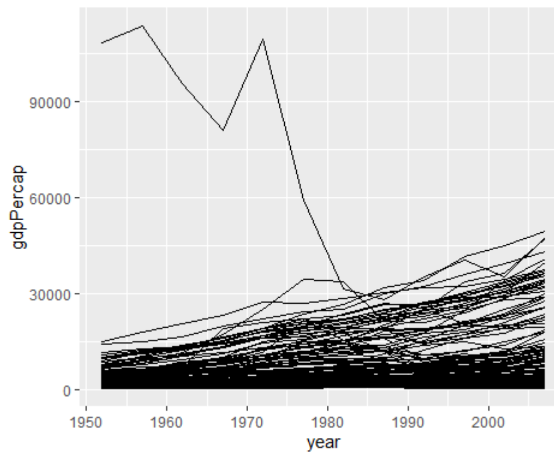


Healy, 2018

14

Grammar of Graphics: Example - Data

- Plot is still rough, but showing the data properly
- Each line representing the trajectory of a country over time
- Group aesthetic needed when the grouping information is not built into the variables being mapped



Healy, 2018

15

Facet to Make Small Multiples: Facet Wrap

- One option is to facet the data by some third variable, making a "small multiple" plot
- Allows a lot of information to be presented compactly and in a consistently comparable way
- Facets are not a geom but rather a way of organizing a series of geoms

Healy, 2018

16

Facet Wrap: Example 1

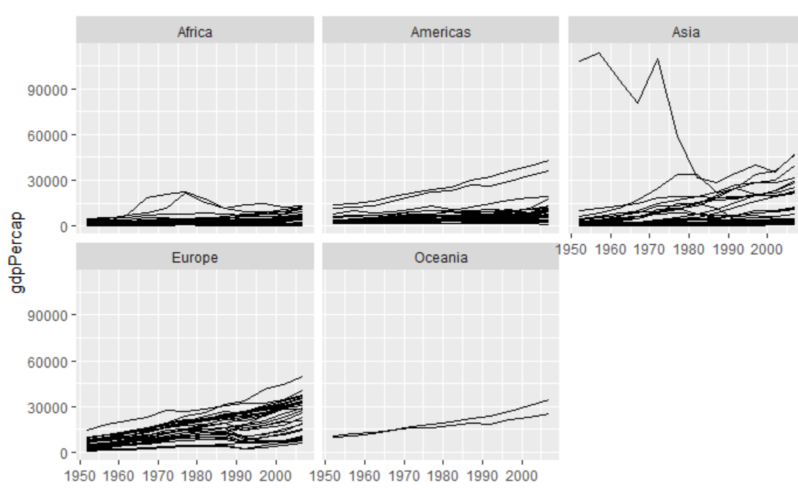
- Use `facet_wrap()` to split our plot by continent.
- The `facet_wrap()` uses facets = vars(continent) within the facet_wrap statement*
- Just like `aes()`, `vars()` is a quoting function that takes inputs to be evaluated in the context of a dataset. These inputs can be:
  - variable names
  - complex expressions

```
p +
  geom_line(aes(group = country)) +
  facet_wrap(facets = vars(continent))
```

Healy, 2018

17

Facet Wrap: Example 1



Healy, 2018

18

Facet Wrap: Example 2

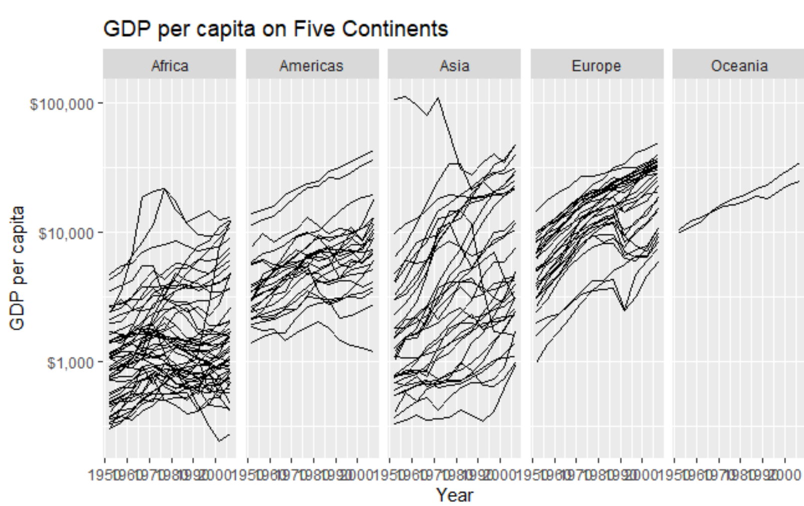
- We can also use the *ncol* argument to *facet_wrap()* to control the number of columns used to lay out the facets

```
 p +
  geom_line(aes(group = country)) +
   scale_y_log10(labels=scales::dollar ) +
  facet_wrap(facets = vars(continent), ncol = 5) +
  labs (x = "Year", y = "GDP per capita", title= "GDP per capita on Fi
ve Continents")
```

Healy, 2018

19

Facet Wrap: Example 2



Healy, 2018

20

Facet Wrap: Example 3

- Previous plot is kind of squished
- Let is save the previous plot to and object and then use ggsave to export the file.

```
plot_gdp <- p +
  geom_line(aes(group = country)) +
   scale_y_log10(labels=scales::dollar ) +
  facet_wrap(~continent, ncol = 5) +
  labs (x = "Year", y = "GDP per capita", title= "GDP per capita on Five
Continents")

ggsave("../figures/plot_gdp.pdf", plot = plot_gdp, width = 11, height = 8.5)
```

Healy, 2018

21

Facet to Make Small Multiples: Facet Grid

- `facet_grid()` forms a matrix of panels defined by row and column faceting variables
- Most useful when you have two discrete variables, and all combinations of the variables exist in the data
- If you have only one variable with many levels, try `facet_wrap()`.

Healy, 2018

22

Facet to Make Small Multiples: Facet Grid

- You control how you want your plots arranged using the following notation options:*
  - `p + facet_grid(rows = vars(drv))`
  - `p + facet_grid(cols = vars(cyl))`
  - `p + facet_grid(vars(drv), vars(cyl))`

* Implemented differently than Healy, because of updates

Healy, 2018

23

Facet Grid: Example 1 - Data

- `gss_sm`, a new dataset that we will use in the next few sections
- GSS is a long-running survey of American adults. See this [site] for full documentation of the [variables].
- Try `glimpse(gss_sm)`, which will give a compact summary of all the variables in the data.

24

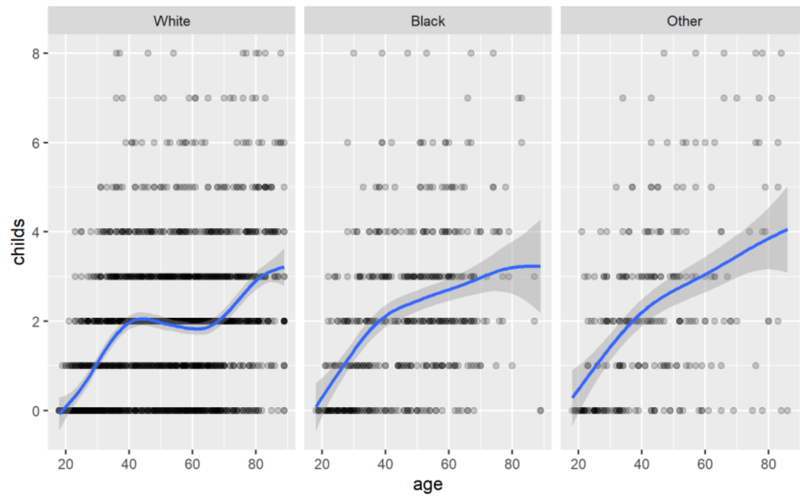
Facet Grid: Example 1 - Plan

- Smoothed scatterplot between the age of the respondent and the number of children they have.
- In `gss_sm` the childs variable is a numeric count of the respondent's children.
- We will then facet this relationship by sex and race of the respondent.
- We use R's formula notation in the `facet_grid` function to facet sex and race
- Because we are cross classifying our results, the formula is two-sided: `face_grid(sex ~ race).`

Facet Grid: Example 1 - Syntax

```
p <- ggplot(data = gss_sm, mapping = aes(x = age, y = childs))
p +
  geom_point(alpha = 0.2) +
  geom_smooth() +
  facet_grid(rows = vars(sex)) +
  facet_grid(cols = vars(race))
```

Facet Grid: Example 1 - Graph



Healy, 2018

27

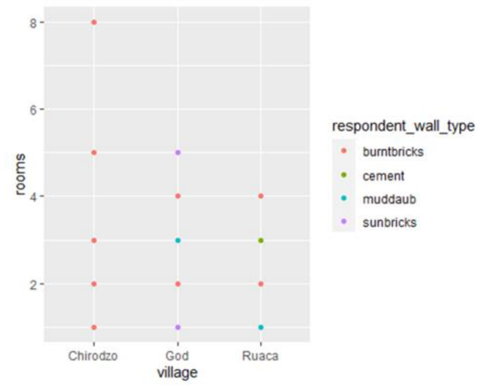
Exercise 1

- Create a new object called `interviews_plotting` and load the `interviews_plotting.csv` data into this object.
- Use what you just learned to create a scatter plot of `rooms (y)` by `village (x)` with the `respondent_wall_type` showing in different colors. Does this seem like a good way to display the relationship between these variables? Looking at the data and what is displayed on the graph…what is going on?

Break Out Rooms
15 minutes to discuss
Pick someone to report out

28

Exercise 1: Answer




```
interviews_plotting %>%
  ggplot(mapping = aes(x = village, y = rooms)) +
  geom_point(mapping = aes(col = respondent_wall_type))
```

29

Exercise 2

- Based on your examination of the data, try to fix your first graph, using another geom that Healy and Wilke discussed.
- HINT: It is **not** another specific chart type

Break Out Rooms
15 minutes to discuss
Pick someone to report out

30

Geoms Can Transform Data

31

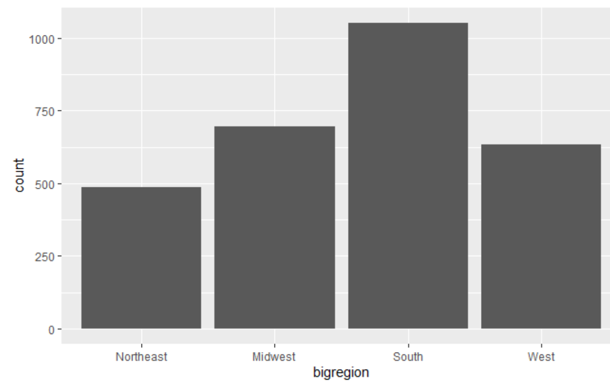
Geom and stat_function

- Every geom_ function has an associated stat_ function that it uses by default
- Two scenarios:
  - We want to calculate a statistic with the geom
  - Sometimes the calculations being done by the stat_ functions and geom_functions might not be immediately obvious

32

Geom and stat_function: Example 1

- Just one mapping, aes (x = bigregion)
- Bar chart has count of the number of (individual) observations
  - North
  - Midwest
  - South
  - West

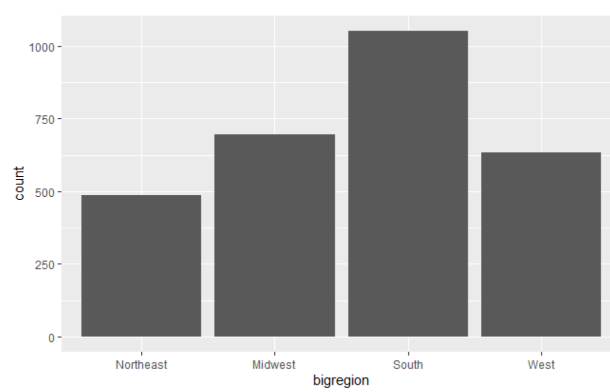



```
p <- ggplot(data = gss_sm,
mapping = aes(x = bigregion))
p + geom_bar()
```

33

Geom and stat_function: Example 1

- Also a y-axis variable count, that is not in the data
- geom_bar called the default stat_ function associated with it, stat_count()
- Function computes two new variables, count and prop (short for proportion)
- Count statistic is the one geom_bar() uses by default.

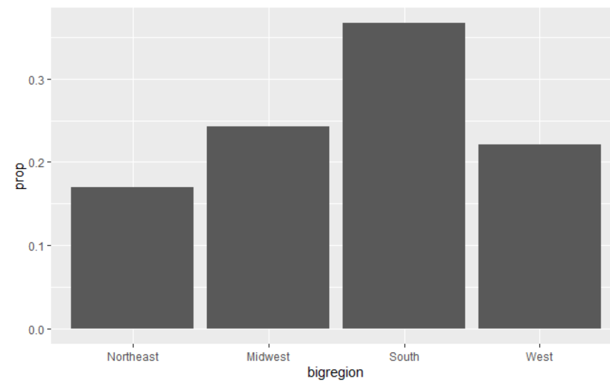



```
p <- ggplot(data = gss_sm,
mapping = aes(x = bigregion))
p + geom_bar()
```

34

Geom and stat_function: Example 2

- Anything other than default needs to be specified
- Relevant statistic is ..prop.. rather than prop
- Syntax makes sure these temporary variables won't be confused with others we are working with

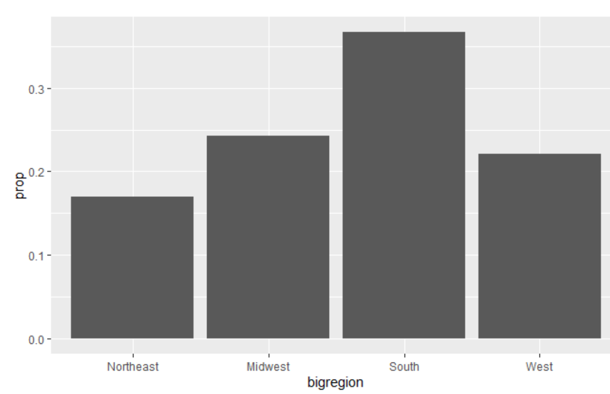



```
p <- ggplot(data = gss_sm, mapping = aes(x = bigregion))
p +
 geom_bar(mapping = aes(y = ..prop.., group = 1))
```

35

Geom and stat_function: Example 2

- we specify group = 1 inside the aes() call
- Value of 1 is a "dummy group" that tells ggplot to use the whole dataset when establishing the denominator for its prop calculations




```
p <- ggplot(data = gss_sm, mapping = aes(x = bigregion))
p +
 geom_bar(mapping = aes(y = ..prop.., group = 1))
```

36

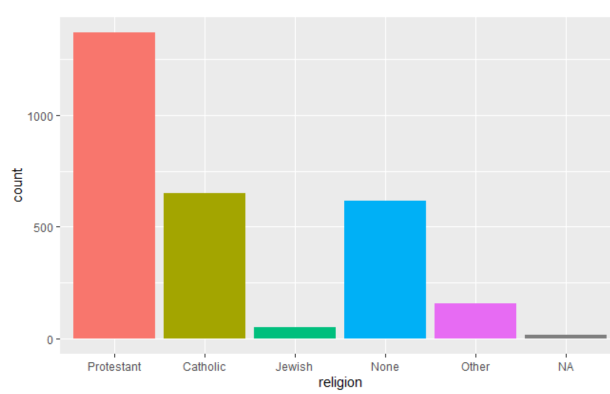
guides() function

- Controls whether guiding information about any mapping appears or not
- guides(fill = "none"), the legend is removed*
- Setting the guide for some mapping to "none" only works if there is a legend to turn off

37

guides() function: Example

```
p <- ggplot(data = gss_sm,
mapping = aes(x = religion, fill = religion))

p +
geom_bar() +
 guides(fill = "none")
```



38

Avoid Transformations When Necessary

39

```
titanic
```

fate <fct>	sex <fct>	n <dbl>	percent <dbl>
perished	male	1364	62.0
perished	female	126	5.7
survived	male	367	16.7
survived	female	344	15.6

4 rows

- So, what do we do when the data is already summarize?

40

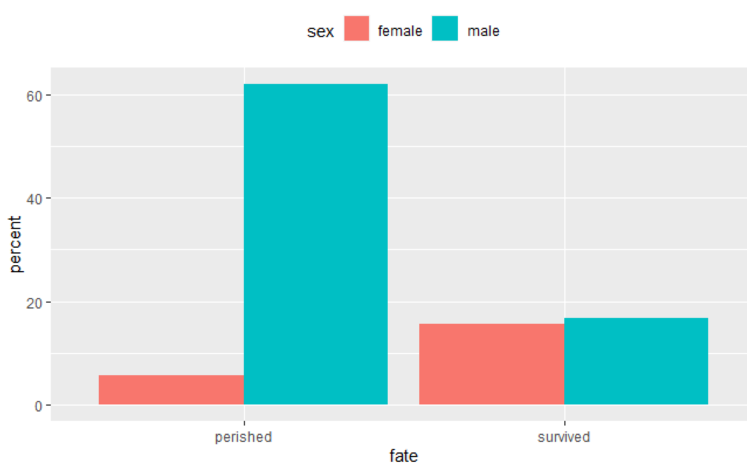
stat = 'identity'

- We do not need the *stat_* functions that *geom_bar()* would normally call, to count up the values
- We do this by adding *stat = 'identity'* in the *geom_bar()* call

```
p <- ggplot(data = titanic, mapping = aes(x = fate, y = percent, fill = sex))
p +
  geom_bar(position = "dodge", stat = "identity") +
  theme(legend.position = "top")
```

41

stat = 'identity': Example



geom_col(), has the same effect but assumes that *stat = "identity."*

42

geom_col()

- *geom_col()*, which has the same effect but assumes that *stat = "identity."*

{r titanic_3}

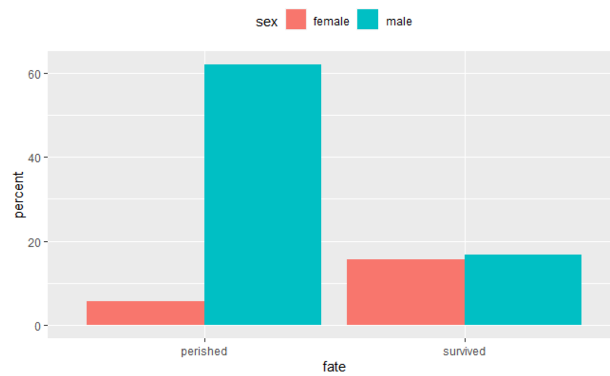
p <- ggplot(data = titanic, mapping = aes(x = fate,

y = percent, fill = sex))

p +

geom_col(position = "dodge") +

theme(legend.position = "top")



43

Using Color to Your Advantage

44

Color Considerations

- Color palette should be based on its ability to express the data you are plotting
- Unordered categorical variable like “country” requires distinct colors that won’t be easily confused with one another
- Ordered categorical variable like ”level of education’ requires a graded color scheme
- If your variable is ordered, your scale should be centered on a neutral midpoint with departures to extremes in each direction

Healy, 2018

45

RColorBrewer

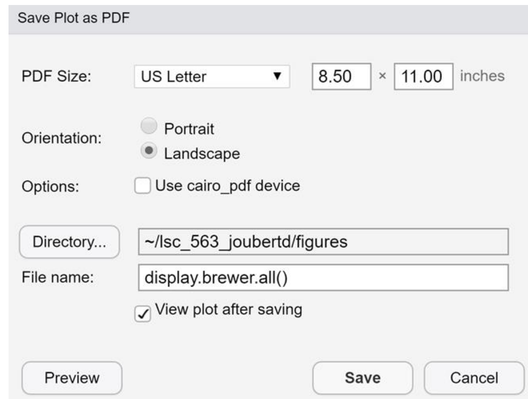
- RColorBrewer is an awesome package that employs a wide range of named color palettes.
- The nice thing about RColorBrewer is that it will show you all of the palettes in a graphics window.
- To make this work type the follow code in the Console:

```
display.brewer.all()
```

46

RColorBrewer

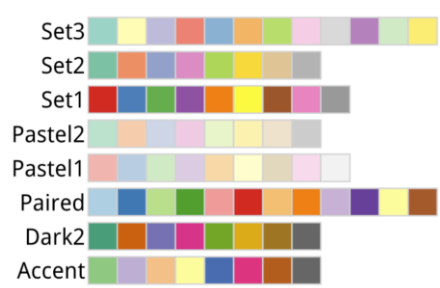
- Let us run this, and then save the plot as a PDF. Plots>Export>Save as PDF.
- You are going to need this image for the lab.



47

RColorBrewer: Qualitative Palettes

- Do not imply magnitude differences classes
- Hues are used to create the primary visual differences between classes
- Best suited to representing nominal or categorical data.



48

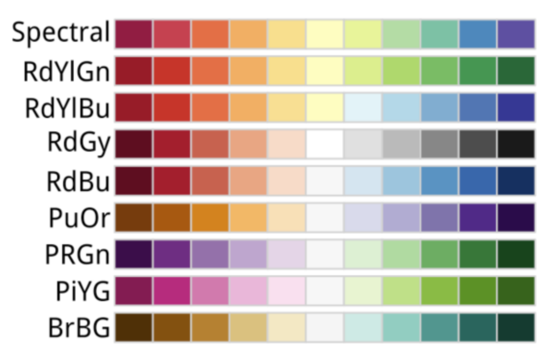
RColorBrewer: Sequential palettes

- Suited to ordered data that progress from low to high.
  - Lightness steps dominate the look of these schemes, with light colors for low data values, to dark colors for high data values.

49

RColorBrewer: Diverging Palettes

- Emphasis on mid-range critical values and extremes at both ends of the data range
  - Critical class or break in the middle emphasized with light colors
  - Low and high extremes are emphasized with dark colors that have contrasting hues
- Most useful for making comparisons with some critical value in the data



50

Specify Colors Manually

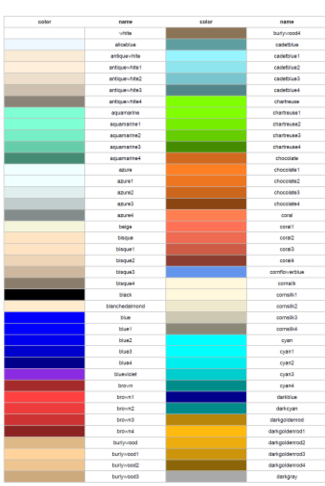
- via *scale_color_manual()* or *scale_fill_manual()*
  - *T*ake a *value* argument that can be specified as vector of color names or color values that R knows about.
- Try *demo('colors')* for an overview
- Color values can be specified via their hexadecimal RGB value
- Each channel can take a value from 0 to 255. A color hex value begins with a hash or pound character, #, followed by three pairs of hexadecimal or "hex" numbers

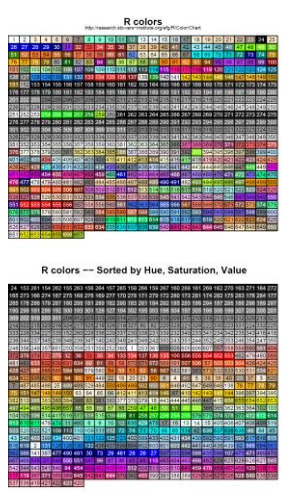
```
cb_palette <- c("#999999", "#E69F00", "#56B4E9", "#009E73",
                "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
```

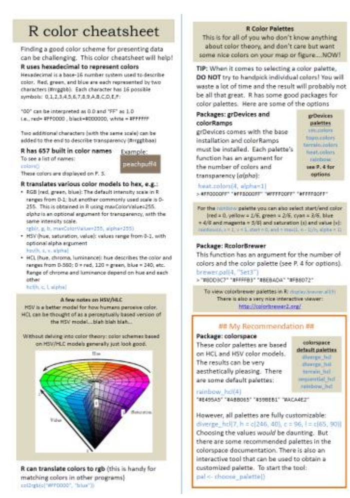
```
p4 + scale_color_manual(values = cb_palette)

## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```

51

Resources

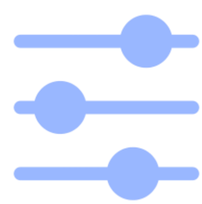






52

53



Refining Plots

54

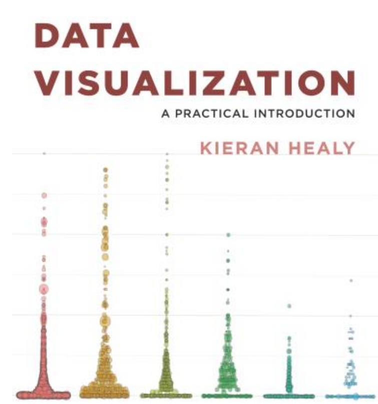
Making Changes

- During exploratory data analysis, the default settings in ggplot should work. However, we can refine a plot:
  - Customize based on personal tastes
  - Meet the expectations of a journal, or a conference presentation

55

Making Changes

- Read Healy chapter 8 (pages 199 - 201) and use as point of reference when working on labs and your final project



56