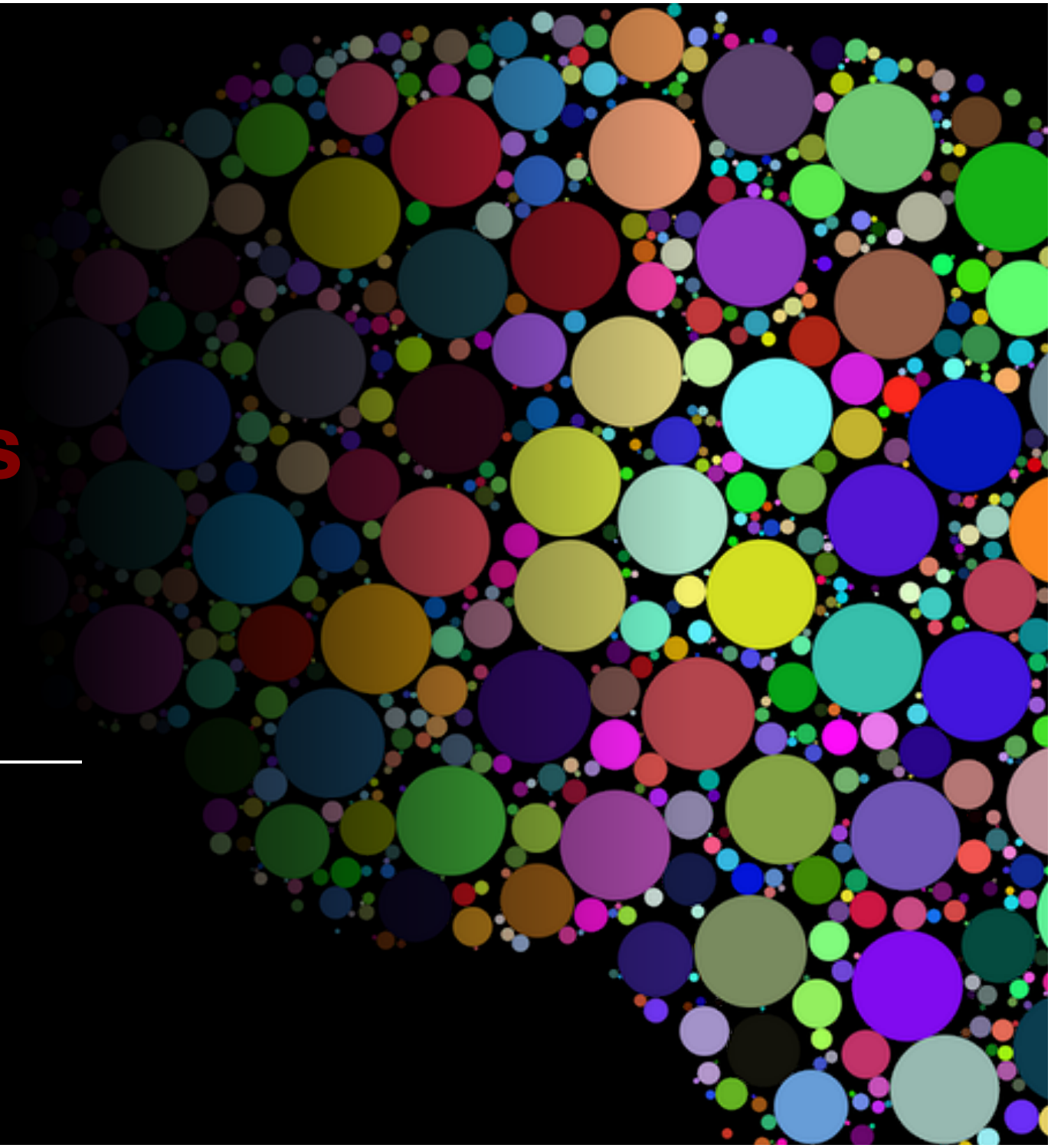
Functions and Graphs

Chapter 3



Random note

> The package DSUR from the Field book is not a thing.

- That’s ok! We’ll figure it out.

A Suggest-told thing to do

> Stack your code.

> The code in this section gets quite long.

> Stack your code.

> You (us) will be able to trouble shoot it easier, it helps see what's going on, and I am telling you to do it.

```
textline = ggplot(longtexting, aes(Time, Grammar_Score, color = Group))
textline +
  stat_summary(fun.y = mean,
               geom = "point") +
  stat_summary(fun.y = mean,
               geom = "line",
               aes(group = Group)) +
  stat_summary(fun.data = mean_cl_boot,
               geom = "errorbar",
               width = .2) +
  xlab("Measurement Time") +
  ylab("Mean Grammar Score") +
  cleanup +
  scale_color_manual(name = "Texting Option",
                     labels = c("All the texts", "None of the texts"),
                     values = c("Black", "Grey"))
```

File Functions

- We've already discussed how to use the import function in Rstudio.
  - However, that only really works well for csv files. What if we have something else?

File Functions

> Install the haven library.

> Be sure to load the library!

- library(haven)

File Functions

> Let’s load SPSS files:

- chickdata = read_spss(”*dataset*.sav")

Factor Functions

> Let's start with some fake data:

- notfactor = rep(1:3, 50)

> Now, let's make that data a factor variable.

> First, check and see what's in the data:

- table(notfactor)

Factor Functions

> factor(*column name*,

- levels = c(*1,2…*), ##things in the data
- labels = c(*“labels”, “labels”,…*)). ##labels to add

> This example:

> factored = factor(notfactor,

- levels=c(1,2,3),
- labels = c("swiss", "feta", "gouda"))

Stack your code!

Rearranging your data

> Install the reshape package and load the library!

> Load the Jiminy Cricket dataset.

```
> cricket = read.csv("c4 Jiminy Cricket.csv", header=TRUE)
> summary(cricket)
       ID            Strategy    Success_Pre     Success_Post
 Min.   :  1.00   Min.   :0.0   Min.   :23.00   Min.   :  2.00
 1st Qu.: 63.25   1st Qu.:0.0   1st Qu.:44.00   1st Qu.: 45.00
 Median :125.50   Median :0.5   Median :50.00   Median : 54.50
 Mean   :125.50   Mean   :0.5   Mean   :50.06   Mean   : 57.42
 3rd Qu.:187.75   3rd Qu.:1.0   3rd Qu.:56.00   3rd Qu.: 69.75
 Max.   :250.00   Max.   :1.0   Max.   :78.00   Max.   :100.00
>
```

Rearranging your data

> This data set is considered the WIDE format.

- In wide formats, rows are participants and columns are variables.

> There's another way?

- In the LONG format, rows are the multiple measurements of the participants.
- Wutz?

Rearranging your data

> Going from WIDE to LONG

- melt() function

> Going from LONG to WIDE

- cast() function*

Rearranging your data

> MELT(

- *dataframe*,
- id = c(*“column”, “column”*) – constant variables you do not want to change
  - > These will stay their own column but get repeated when necessary
- measured = c(*“column”, “column”*) – dependent variables you want to combine into one column
-)

Rearranging your data

> longcricket = melt(cricket,

- id = c("ID", "Strategy"),
- measured = c("Success_Pre", "Succcess_Post"))

> Let’s look at what that did.

Rearranging your data

```
       ID              Strategy    Success_Pre     Success_Post
 Min.   :  1.00   Min.   :0.0   Min.   :23.00   Min.   :  2.00
 1st Qu.: 63.25   1st Qu.:0.0   1st Qu.:44.00   1st Qu.: 45.00
 Median :125.50   Median :0.5   Median :50.00   Median : 54.50
 Mean   :125.50   Mean   :0.5   Mean   :50.06   Mean   : 57.42
 3rd Qu.:187.75   3rd Qu.:1.0   3rd Qu.:56.00   3rd Qu.: 69.75
 Max.   :250.00   Max.   :1.0   Max.   :78.00   Max.   :100.00
> summary(longcricket)
       ID              Strategy          variable       value
 Min.   :  1.0   Min.   :0.0   Success_Pre :250   Min.   :  2.00
 1st Qu.: 63.0   1st Qu.:0.0   Success_Post:250   1st Qu.: 44.00
 Median :125.5   Median :0.5                      Median : 52.00
 Mean   :125.5   Mean   :0.5                      Mean   : 53.74
 3rd Qu.:188.0   3rd Qu.:1.0                      3rd Qu.: 61.00
 Max.   :250.0   Max.   :1.0                      Max.   :100.00
>
```

Graphs Outline

> How to present data clearly

> R graphs

- Histograms
- Boxplots
- Error bar charts
- Scatterplots
- Line Graphs

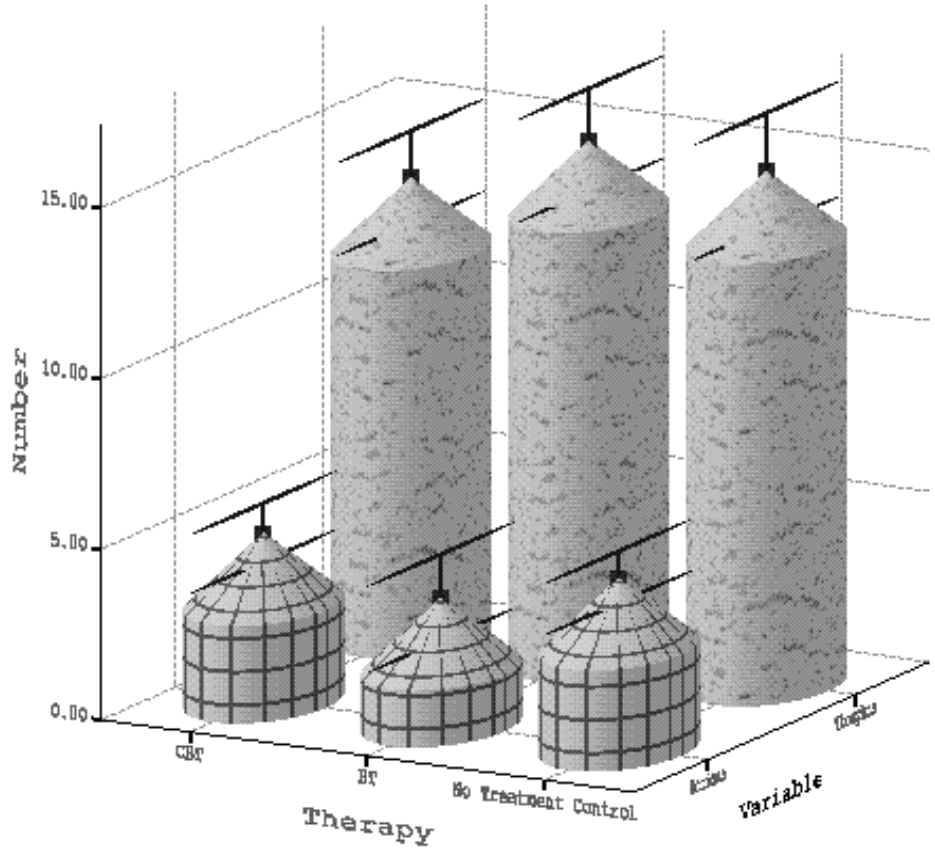
The Art of Presenting Data

> Graphs should (Tufte, 2001):

- Show the data.
- Induce the reader to think about the data being presented (rather than some other aspect of the graph).
- Avoid distorting the data.
- Present many numbers with minimum ink.
- Make large data sets (assuming you have one) coherent.
- Encourage the reader to compare different pieces of data.
- Reveal data.

Error Bars show 95.0 % CI of Mean

Bars show Means



Bad Graphs

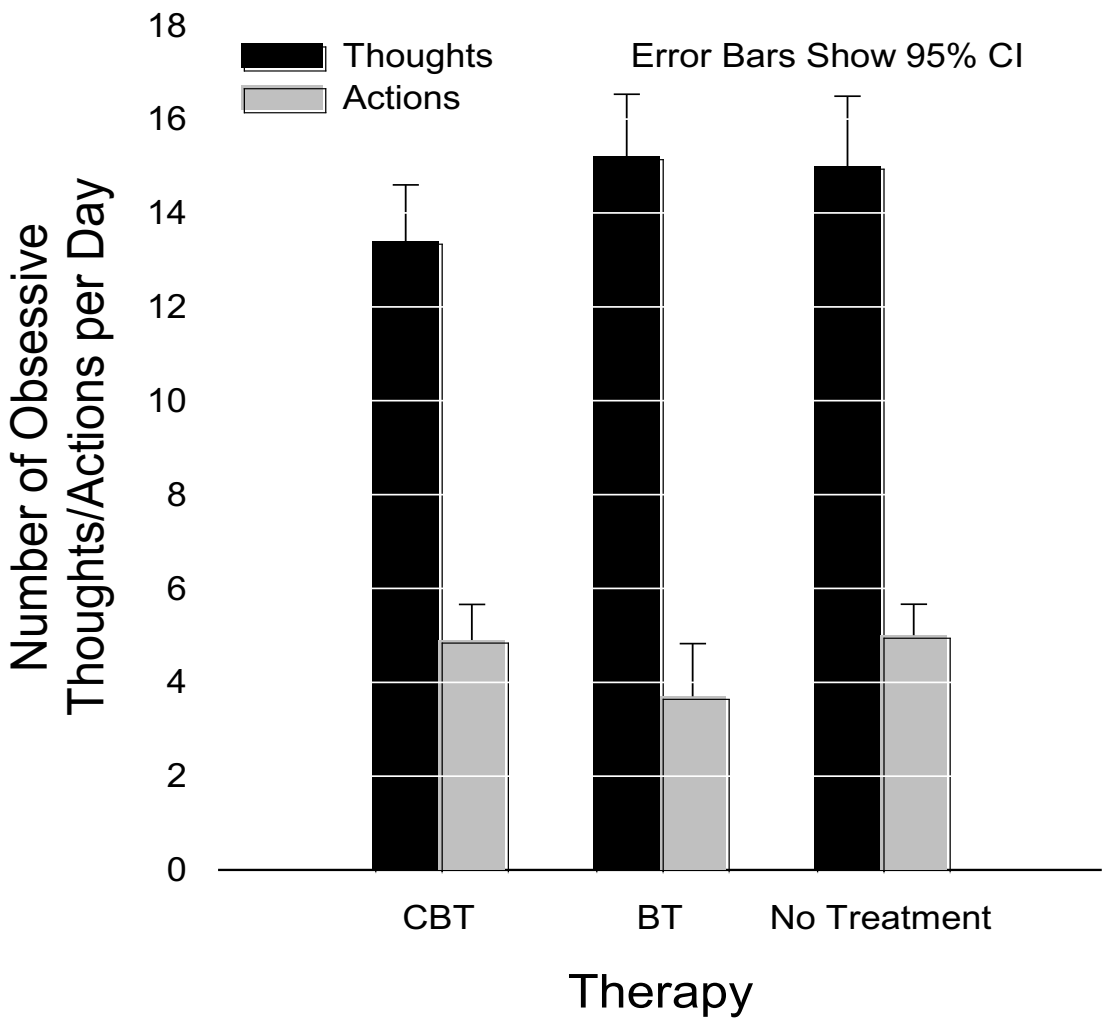
> 3D charts are bad.

> Patterns (depending)

> Cylindrical bars

> Bad axis labels

> Overlays


Thoughts
Actions
Error Bars Show 95% CI
Number of Obsessive Thoughts/Actions per Day
18
16
14
12
10
8
6
4
2
0
CBT
BT
No Treatment
Therapy

Deceiving the Reader

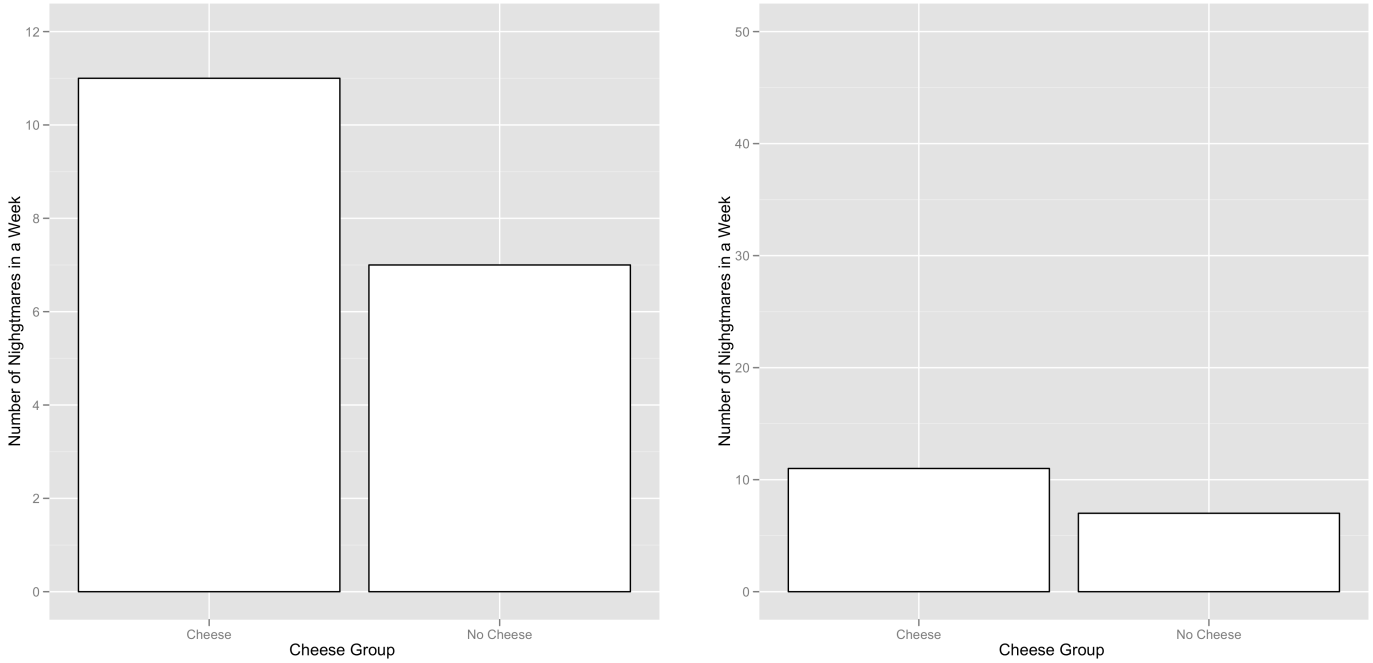
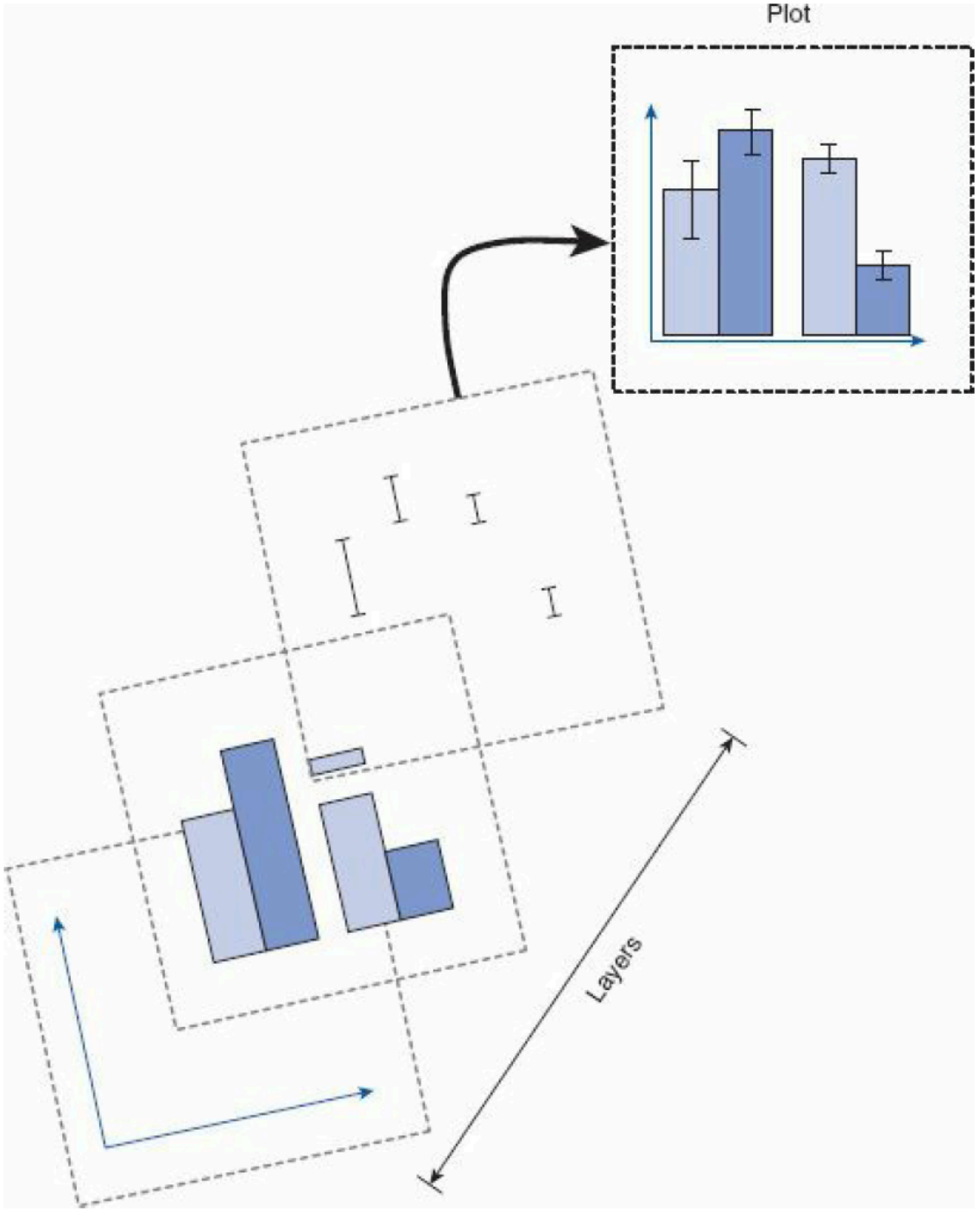


Figure 4.4: Two graphs about cheese

Plots in R

> Install ggplot2 package and load the library.

- There are lots of plot functions and packages, but ggplot2 is pretty nice.
- You may need to install Hmisc (error bars).


Plot
Layers

Too many options

> Ggplot2 is awesome because it has a zillion options.

- But that gets super overwhelming at the beginning.
- Remember that section 4.3.2 gives you a handy reference chart.

GGPLOT

> How ggplot works:

- First you define the basic structure of a plot you want.

> Example:

> myGraph = ggplot(*dataset*,

- aes(*column x axis*,

  > *Column y axis*,

  > color/fill = *legend column*))

GGPLOT

> Now we have mygraph saved, but nothing on it really.

> So we will add options and layers to create the graph.

GGPLOT

> myGraph + OTHER STUFF

> myGraph +

- geom_bar() +
- geom_point() +
- xlab(“xlabel”) +
- ylab(“ylabel”)

> So we are slowing adding components to the graph.

Histograms

- Histograms plot:
  - The continuous score (*x*-axis)
  - The frequency (*y*-axis)
- Histograms help us to identify:
  - The shape of the distribution
    - Skew
    - Kurtosis
    - Spread or variation in scores
  - Unusual scores

Histograms: Example

- I wanted to test the Disney philosophy that ‘Wishing upon a star makes all your dreams come true’.
- Measured the success of 250 people using a composite measure involving their salary, quality of life and how close their life matches their aspirations.
- Success was measured using a standardised technique :
  - Score ranged from 0 to 4
    - 0 = Complete failure
    - 4 = Complete success
- Participants were randomly allocated to either, Wish upon a star or work as hard as they can for next 5 years.
- Success was measured again after 5 years.

Histograms

> In ggplot:

- Load the Jiminy cricket data and call it cricket.

> Histogram – let's go!

- First build the plot:
- crickethist = ggplot(cricket, aes(Success_Pre))

  >(*datasetname*, aes(*column name*))

Histograms

> But that's going to be blank, so let's add a histogram on top of that:

- crickethist + geom_histogram()
- Make the bins different

```
>crickethist + geom_histogram(binwidth = 0.4)
```

- That color is hideous

```
>crickethist + geom_histogram(binwidth = 0.4,
  color = "green")
```

Histogram of Hygiene Scores for Day 1

> Create the plot object:

```
festivalhist = ggplot(festival, aes(day1))
```

> Add all the plot parts:

- geom_histogram(*binwidth, color, fill*)
- xlab()
- ylab()
- theme_bw() #we will end up doing something better than this clean up

Histograms

> Let's add some labels:

> crickethist +

- geom_histogram(binwidth = 0.4,

color =

"green") +

- xlab("Success Pre Test") +
- ylab("Frequency")

Histograms: Example

> A biologist was worried about the potential health effects of music festivals.

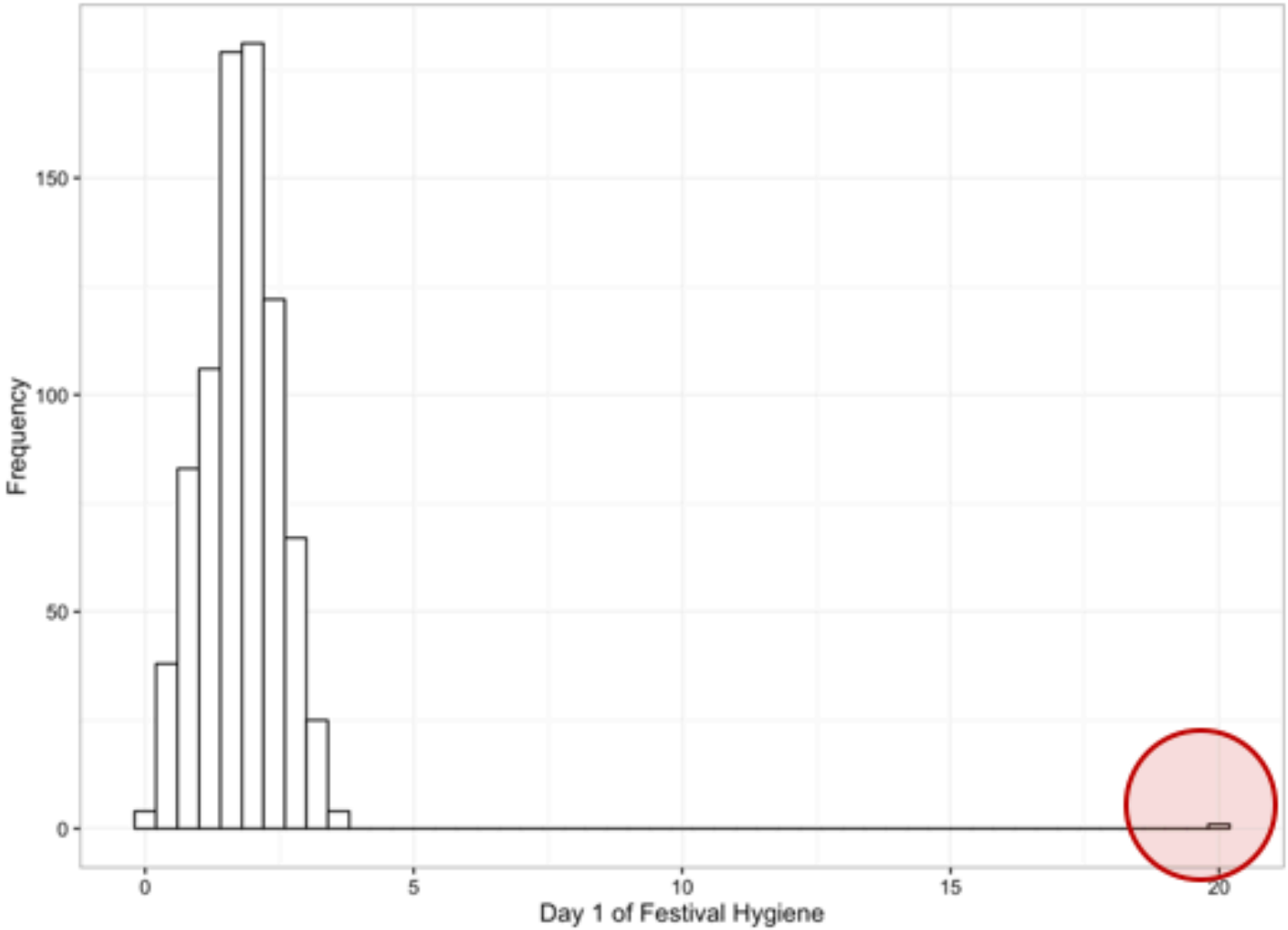
> Download Music Festival

> Measured the hygiene of 810 concert-goers over the three days of the festival.

> Hygiene was measured using a standardized technique :

- Score ranged from 0 to 4
  - > 0 = you smell like a corpse rotting up a skunk's arse
  - > 4 = you smell of sweet roses on a fresh spring day

DATASET: FESTIVAL DATA


Frequency
150
100
50
0
0
5
10
15
20
Day 1 of Festival Hygiene

Rules!

> All graphs should have:

- X and Y axis labels
- NO TITLES (this is APA)
- Labels for groups/legend
- Error bars when appropriate

> And they should be readable and not ugly.

Clean up code

> We are going to include some “cleanup” code for each group to help make them

```
##Other clean up option
cleanup = theme(panel.grid.major = element_blank(),
                panel.grid.minor = element_blank(),
                panel.background = element_blank(),
                axis.line.x = element_line(color = "black"),
                axis.line.y = element_line(color = "black"),
                legend.key = element_rect(fill = "white"),
                text = element_text(size = 15))
```

Save this code and then you can just do graph + cleanup

Scatterplots

> Simple scatter – two continuous variables

> Grouped scatter – two continuous variables + 1 categorical variable

Scatterplots: Example

> Anxiety and exam performance

> Participants:

- 103 students

> Measures

- Time spent revising (hours)
- Exam performance (%)
- Exam Anxiety (the EAQ, score out of 100)
- Gender

DATASET = EXAM ANXIETY

Scatterplots: Example

> First, let's change gender to a factor – so we can use it later for graphing appropriately.

- exam$Gender = factor(exam$Gender,

  >levels=c(1,2),

  >labels = c("Male", "Female"))

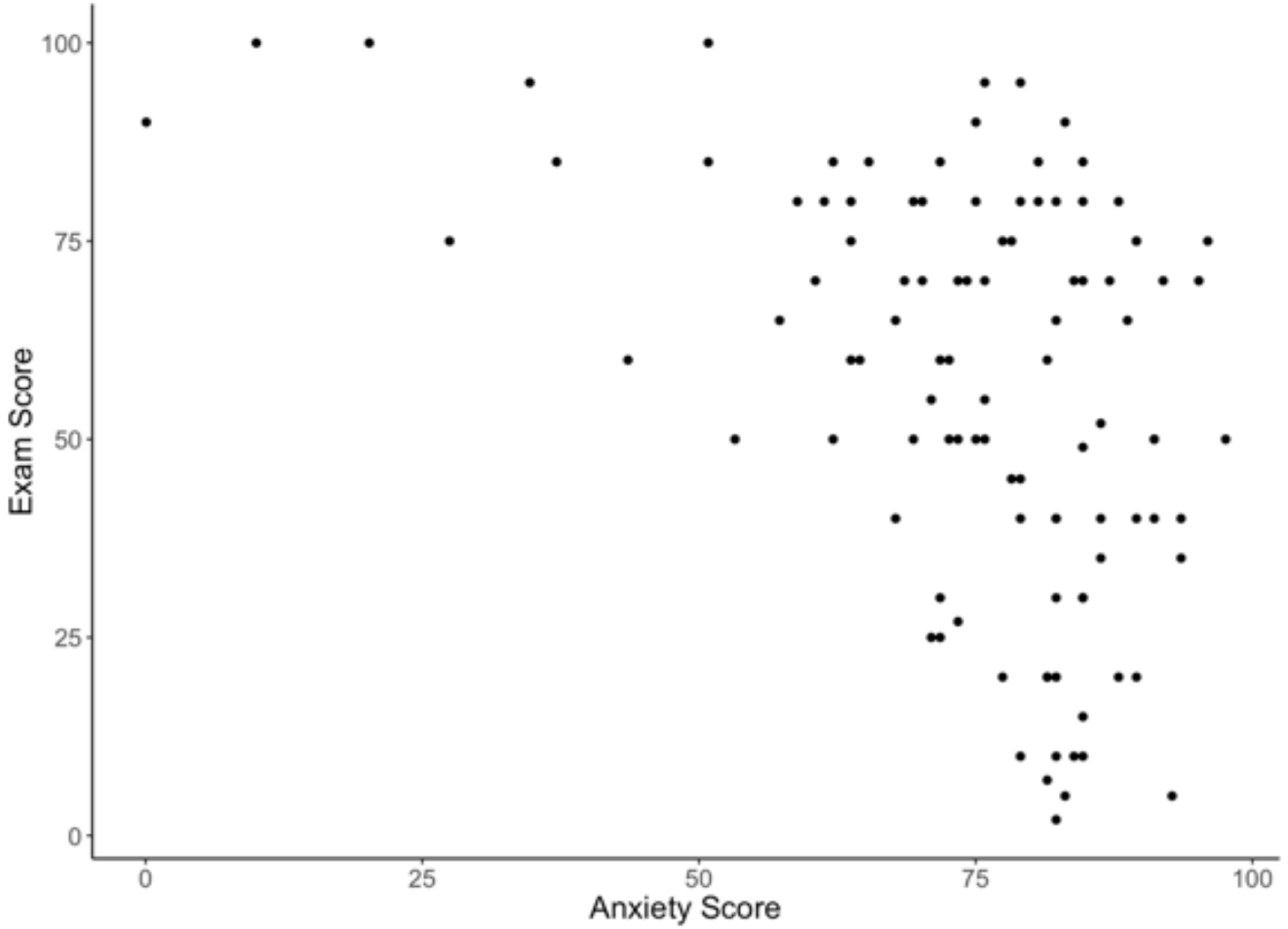
Simple Scatterplot

```
> scatter = ggplot(exam, aes(Anxiety, Exam))
```

```
> scatter +
```

- geom_point() +
- xlab("Success Pre Test") +
- ylab("Frequency") +
- cleanup

Simple Scatterplot



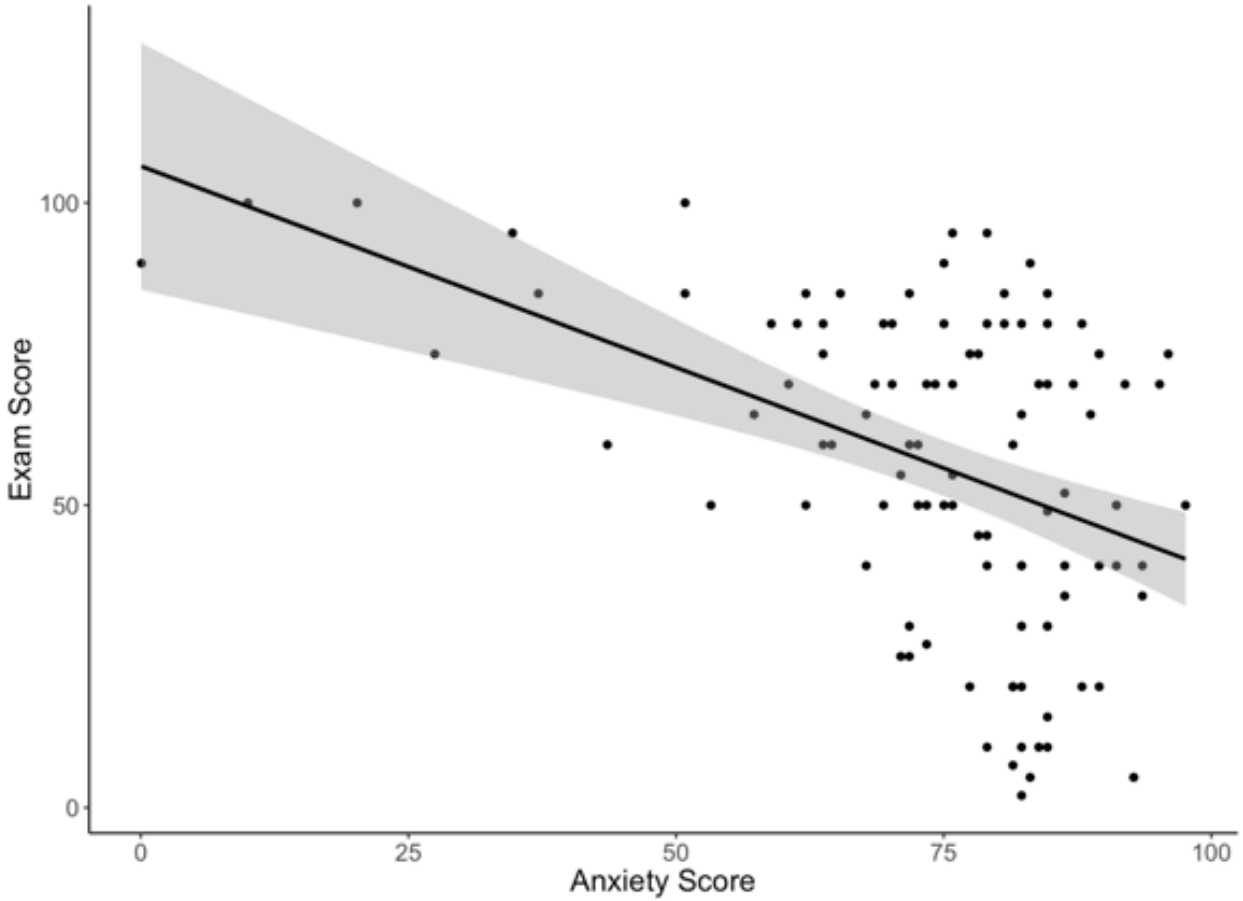
exam performance

Simple Scatterplot With Regression Line

```
scatter = ggplot(exam, aes(Anxiety, Exam))


scatter +
    geom_point() +
    geom_smooth(method = "lm", color = ”black")
+
    xlab("Anxiety Level") +
    ylab("Exam Performance") +
    cleanup
```

Simple Scatterplot



ession line

Grouped Scatterplot

```
scatter = ggplot(examData,
    aes(Anxiety, Exam, color = Gender, fill =
Gender))

scatter +
  geom_point() +
  geom_smooth(method = "lm") +
  xlab("Anxiety Level") +
  ylab("Exam Performance") +
  cleanup
```

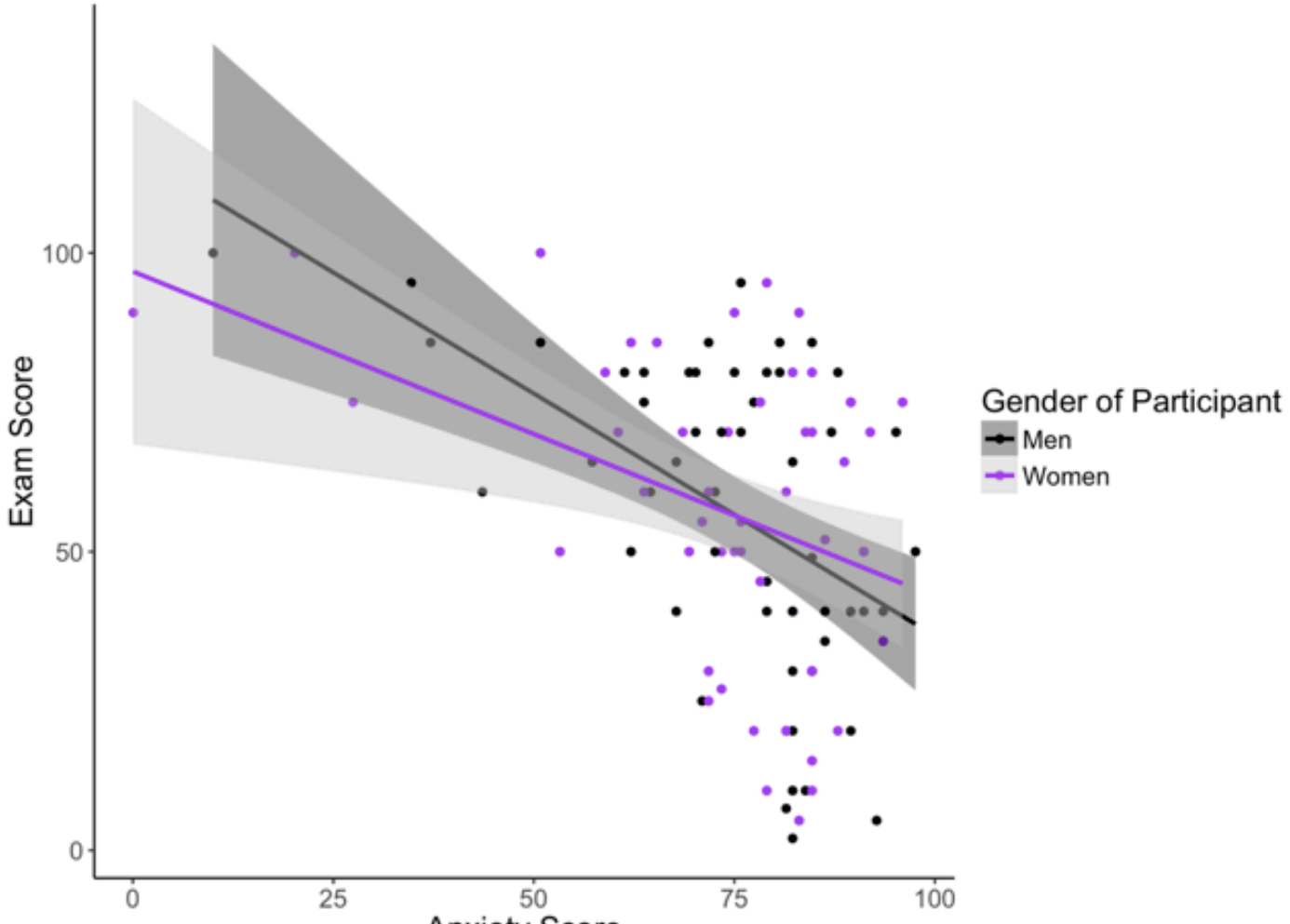
Grouped Scatterplot

> How to control the colors and fill with legends.

- scale_fill_manual(*name, labels, values*)
- scale_color_manual(*name, labels, values*)

```
scale_fill_manual(name = "Gender of Participant",
                  labels = c("Men", "Women"),
                  values = c("black", "grey")) +
scale_color_manual(name = "Gender of Participant",
                   labels = c("Men", "Women"),
                   values = c("black", "purple"))
```

Grouped Scatterplot



exam performance split by gender

Bar Charts + Error

> The bar (usually) shows the mean score

> The error bar displays the precision of the mean in one of three ways:

- The confidence interval (usually 95%)
- The standard deviation
- The standard error of the mean

Bar Chart: One Independent Variable

- Is there such a thing as a ‘chick flick’?
- Participants:
  - 20 men
  - 20 women
- Half of each sample saw one of two films:
  - A ‘chick flick’ (*Bridget Jones’s Diary*),
  - Control (*Memento*).
- Outcome measure
  - Physiological arousal as an indicator of how much they enjoyed the film.

DATASET = CHICK FLICK

Bar Chart Note

> To get bar charts to work appropriately, make sure your categorical IVs are factor variables

- Else you will get very strange errors in ggplot.

Bar Chart: One Independent Variable

> To plot the mean arousal score (*y*-axis) for each film (*x*-axis) first create the plot object:

```
chickbar = ggplot(chick, aes(film, arousal))
```

Bar Chart: One Independent Variable

> To add the mean, displayed as bars, we can add this as a layer to *bar* using the *stat_summary()* function:

```
chickbar +
        stat_summary(fun.y = mean,
                geom = "bar",
                fill = "White",
                color = "Black")
```

Bar Chart: One Independent Variable

> To add error bars, add these as a layer using *stat_summary()*:

```
+ stat_summary(fun.data = mean_cl_normal,
                geom = "errorbar",
                position = position_dodge(width =
0.90),
                width = 0.2)
```

Bar Chart: One Independent Variable

> Now, add the rest of things we've been doing

xlab("*label*") +

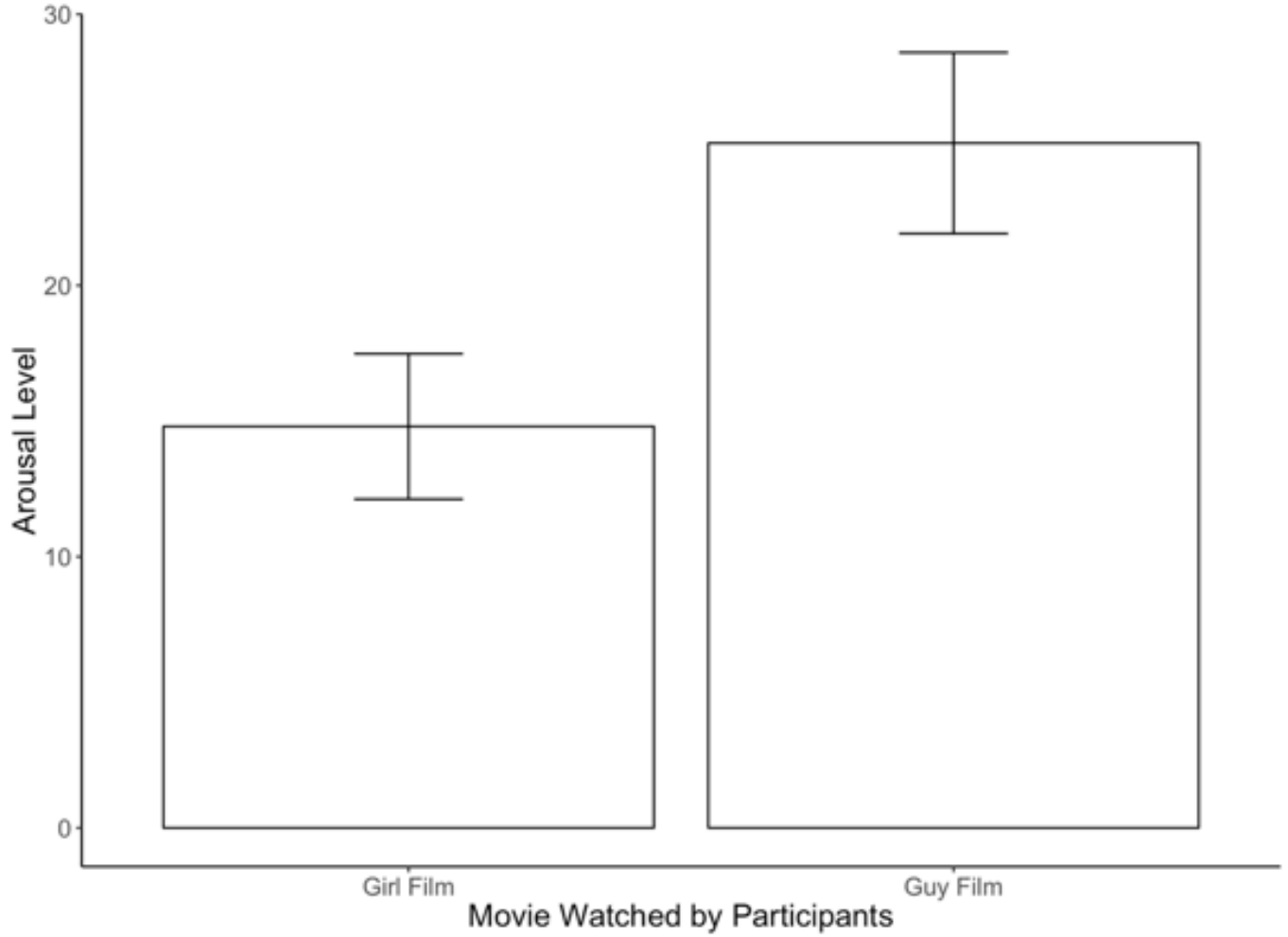
ylab("label") +

cleanup +

scale_x_discrete(labels = c("Girl Film", "Guy Film"))

**scale_x_discrete allows us to change the labels/names for the x axis but NOT color (that's scale_x_manual)

Bar Chart: One Independent Variable



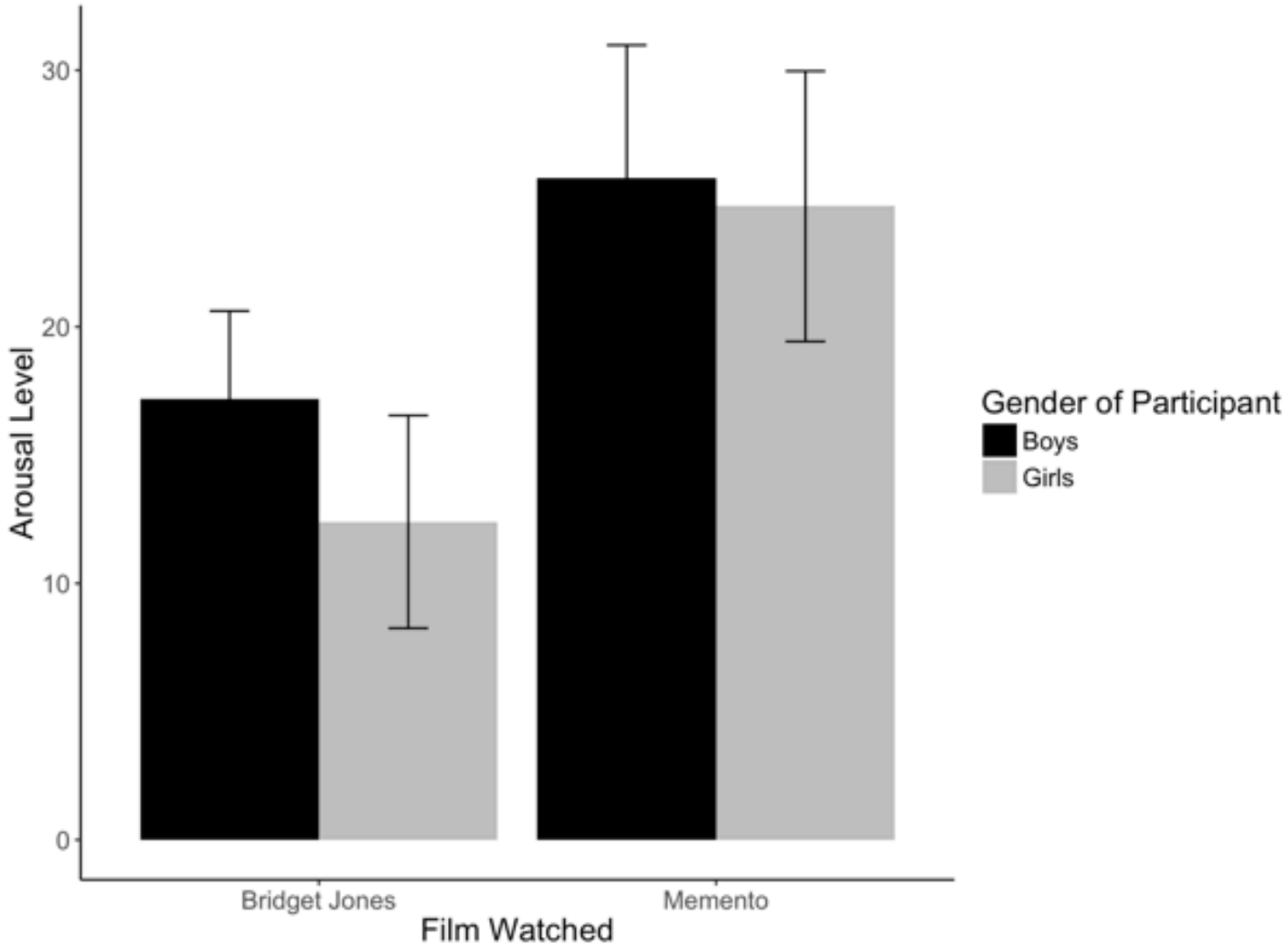
Bar Chart: Two Independent Variables

Chickbar2 = ggplot(chick, aes(film, arousal, fill = gender))

```
chickbar2 +
  stat_summary(fun.y = mean,
               geom = "bar",
               position = "dodge") +
  stat_summary(fun.data = mean_cl_normal,
               geom = "errorbar",
               position = position_dodge(width = 0.90),
               width = .2) +
  xlab("Film Watched") +
  ylab("Arousal Level") +
  cleanup +
  scale_fill_manual(name = "Gender of Participant",
                    labels = c("Boys", "Girls"),
                    values = c("Black", "Gray"))
```

So why no color

Bar Chart: Two Independent Variables



Line Graphs

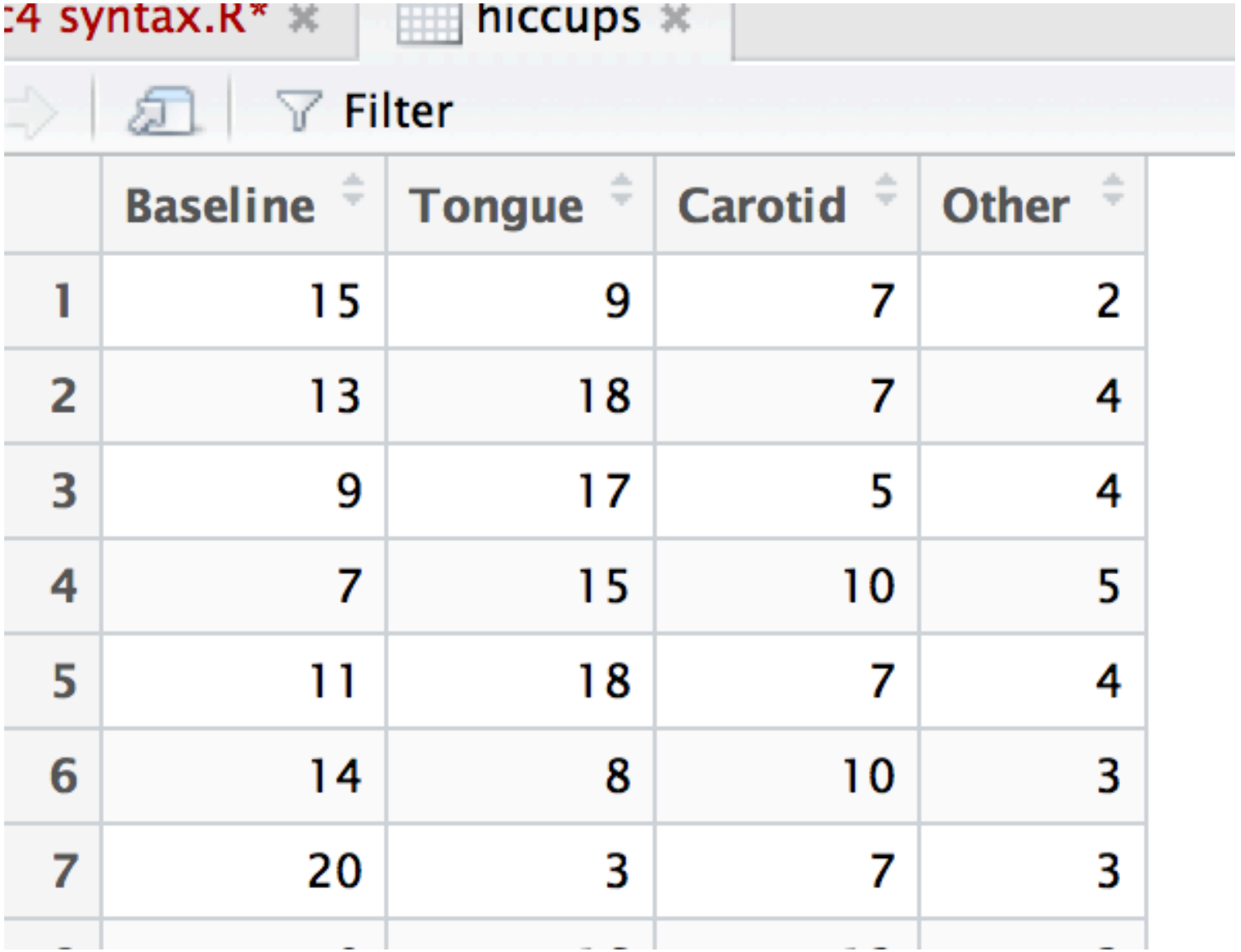
> When to use a line graph:

- With data that X is categorical, but is considered “mildly continuous”
- Usually with repeated measures data over time.
- (which is not what these examples are but oh well).

Line Graphs: One Independent Variable

- How to cure hiccups?
- Participants:
  - 15 hiccup sufferers
- Each tries four interventions (in random order):
  - Baseline
  - Tongue-pulling manoeuvres
  - Massage of the carotid artery
  - Other
- Outcome measure
  - The number of hiccups in the minute after each procedure

DATASET: HICCUP DATA



	Baseline	Tongue	Carotid	Other
1	15	9	7	2
2	13	18	7	4
3	9	17	5	4
4	7	15	10	5
5	11	18	7	4
6	14	8	10	3
7	20	3	7	3

Line Graphs: One Independent Variable

> These data are in the wrong format for *ggplot2* to use.

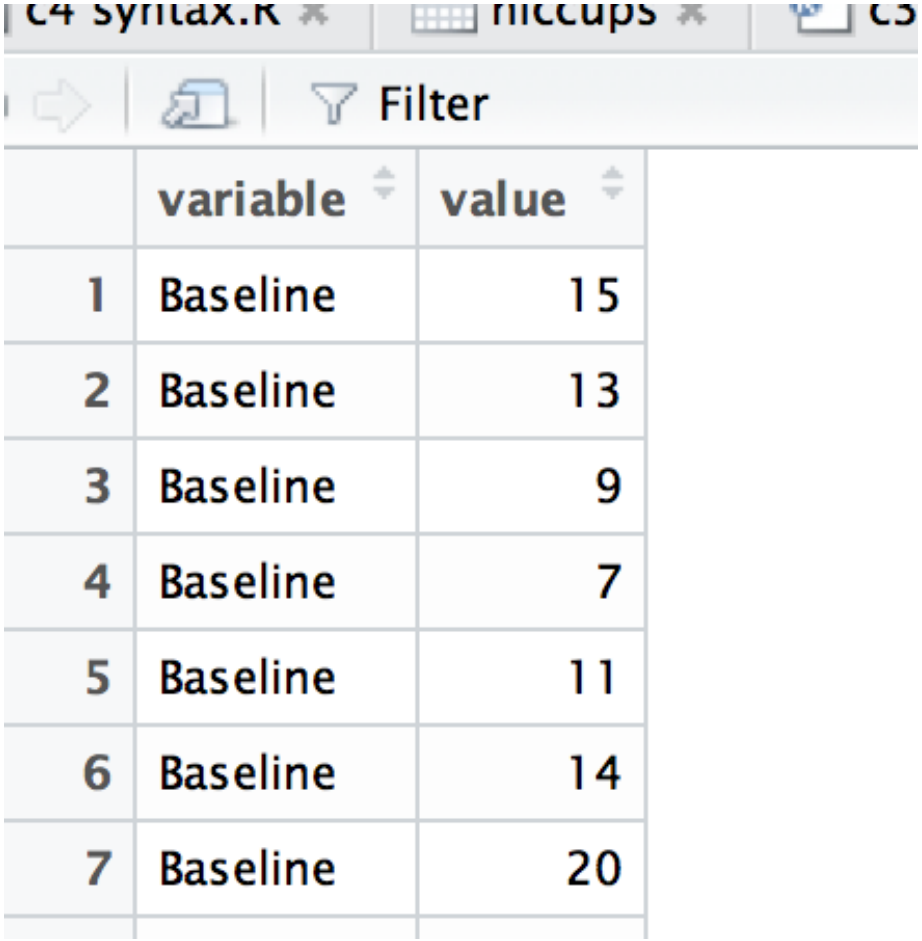
> We need all of the scores stacked up in a single column and then another variable that specifies the type of intervention.

> Melt time!

Melt the data

```
> longhiccups = melt(hiccups,
                     measured = c("Baseline",
"Tongue",                                    "Carotid",
"Other"))
```

Notice there's no ID, that's ok.

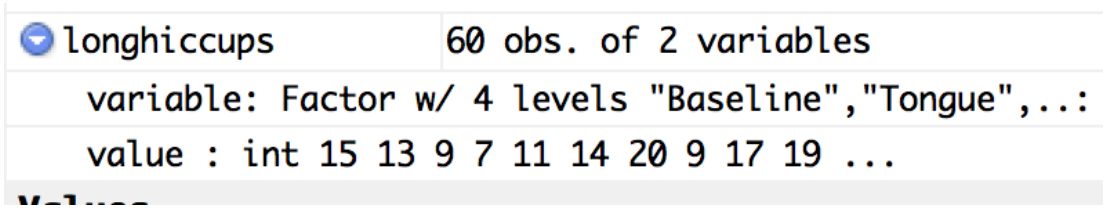


	variable	value
1	Baseline	15
2	Baseline	13
3	Baseline	9
4	Baseline	7
5	Baseline	11
6	Baseline	14
7	Baseline	20

Line Graphs: One Independent Variable

> Then you need to make sure your new variable is a factor.

- It is yay!
- But always check.



Line Graphs: One Independent Variable

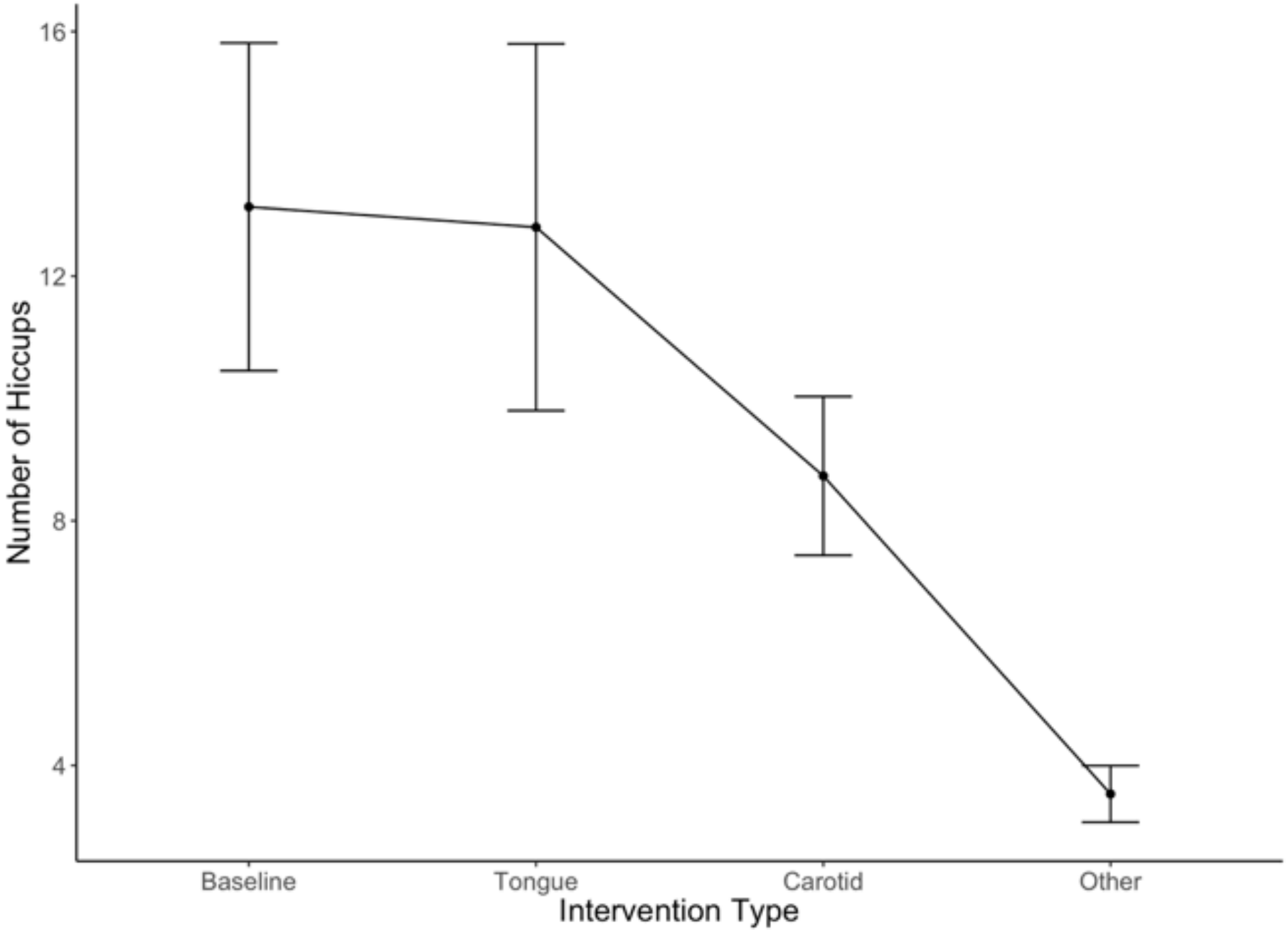
- We can then create the line graph:
  - hiccupline = ggplot(longhiccups, aes(variable, value))
  - Notice they are called variable and value.
  - We can change that to make more sense.

```
>colnames(longhiccups) = c("Intervention", "Hiccups")
>hiccupline = ggplot(longhiccups, aes(Intervention, Hiccups))
```

Line Graphs: One Independent Variable

> Add the rest of the formatting:

```
hiccupline +
  stat_summary(fun.y = mean, ##adds the points
               geom = "point") +
  stat_summary(fun.y = mean, ##adds the line
               geom = "line",
               aes(group=1)) +
  stat_summary(fun.data = mean_cl_normal, ##adds the error bars
               geom = "errorbar",
               width = .2) +
  xlab("Intervention Type") +
  ylab("Number of Hiccups") +
  cleanup
```



various interventions

Line Graphs for Several Independent Variables

- Is text-messaging bad for your grammar?
- Participants:
  - 50 children
- Children split into two groups:
  - Text-messaging allowed
  - Text-messaging forbidden
- Each child measures at two points in time:
  - Baseline
  - 6 months later
- Outcome measure
  - Percentage score on a grammar test

DATASET: TEXTING

Filter

	Group	Baseline	Six_months
1	1	52	32
2	1	68	48
3	1	85	62
4	1	47	16
5	1	73	63
6	1	57	53
7	1	63	59
8	1	50	58

First, let's fix the group problem:

```
texting$Group =
factor(texting$Group,
       levels=c(1,2),
```

Melt the data!

Filter

	Group	variable	value
1	Texting Allowed	Baseline	52
2	Texting Allowed	Baseline	68
3	Texting Allowed	Baseline	85
4	Texting Allowed	Baseline	47
5	Texting Allowed	Baseline	73
6	Texting Allowed	Baseline	57
7	Texting Allowed	Baseline	63
8	Texting Allowed	Baseline	50

ng 1 to 8 of 100 entries

Fix the column names

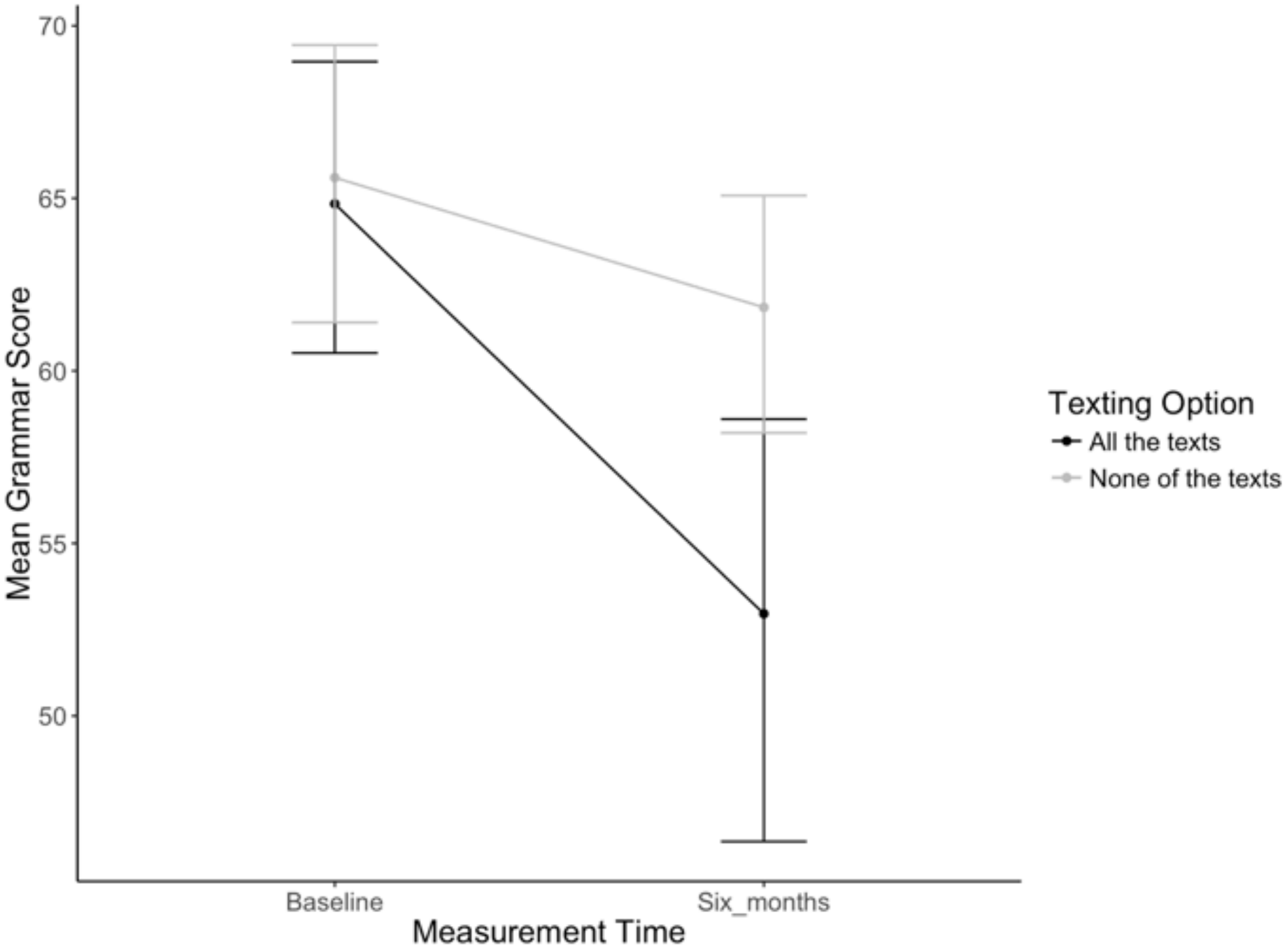
- colnames(textMessages) =

c(“Group”, “Time”, "Grammar_Score”)

Line Graphs for Several Independent Variables

textline = ggplot(longtexting, aes(Time, Grammar_Score, color = Group))

```
textline +
  stat_summary(fun.y = mean,
               geom = "point") +
  stat_summary(fun.y = mean,
               geom = "line",
               aes(group = Group)) +
  stat_summary(fun.data = mean_cl_normal,
               geom = "errorbar",
               width = .2) +
  xlab("Measurement Time") +
  ylab("Mean Grammar Score") +
  cleanup +
  scale_color_manual(name = "Texting Option",
                     labels = c("All the texts", "None of the texts"),
                     values = c("Black", "Grey"))
```


70
65
60
55
50
Mean Grammar Score
Baseline
Six_months
Measurement Time
Texting Option
All the texts
None of the texts