Digi Data Visualization Series: Intro to ggplot2

Keiko Bridwell

10/13/2021

Welcome to a second installment of our series on ggplot2 for data visualization!
If you are following along, please make sure you have R and R Studio installed for your operating system (os).

More on how to install R and R Studio here.

You can find the first part of this series, Introduction to ggplot2 (by Katie Ireland Kuiper), at https://digi.uga.edu/tutorials/. This workshop provides basic instructions on how the components of ggplot work, including:

- the basic syntax of a ggplot() command
- how to plot boxplots, line graphs, barplots, and scatterplots
- how to change the appearance of a plot using preset themes
- how to change colors and shapes using theme()
- how to add axis and plot titles

Now, let's get started with some more advanced features of ggplot2!

Preparing R

First, install either ggplot2 or tidyverse using the commands below.

```
# install.packages("ggplot2")
library(ggplot2)
# install.packages("tidyverse")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.2     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

The dataset that we will be using today is from the datasets library, and is called `ChickWeights`. It is from an experiment in which baby chicks were fed one of four different diets, and their weights measured from 0 to 21 days.

```
# install.packages("datasets")
library(datasets)
```

Next, let's take a look at the data.

```
summary(ChickWeight)
```

```
##      weight           Time           Chick     Diet
##  Min.   : 35.0   Min.   : 0.00   13     : 12   1:220
##  1st Qu.: 63.0   1st Qu.: 4.00   9      : 12   2:120
##  Median :103.0   Median :10.00   20     : 12   3:120
##  Mean   :121.8   Mean   :10.72   10     : 12   4:118
##  3rd Qu.:163.8   3rd Qu.:16.00   17     : 12
##  Max.   :373.0   Max.   :21.00   19     : 12
##                                  (Other):506
```

For the Diet variable, let's relabel it as "Diet 1", "Diet 2", etc. This will make graphs more intuitive at later steps!

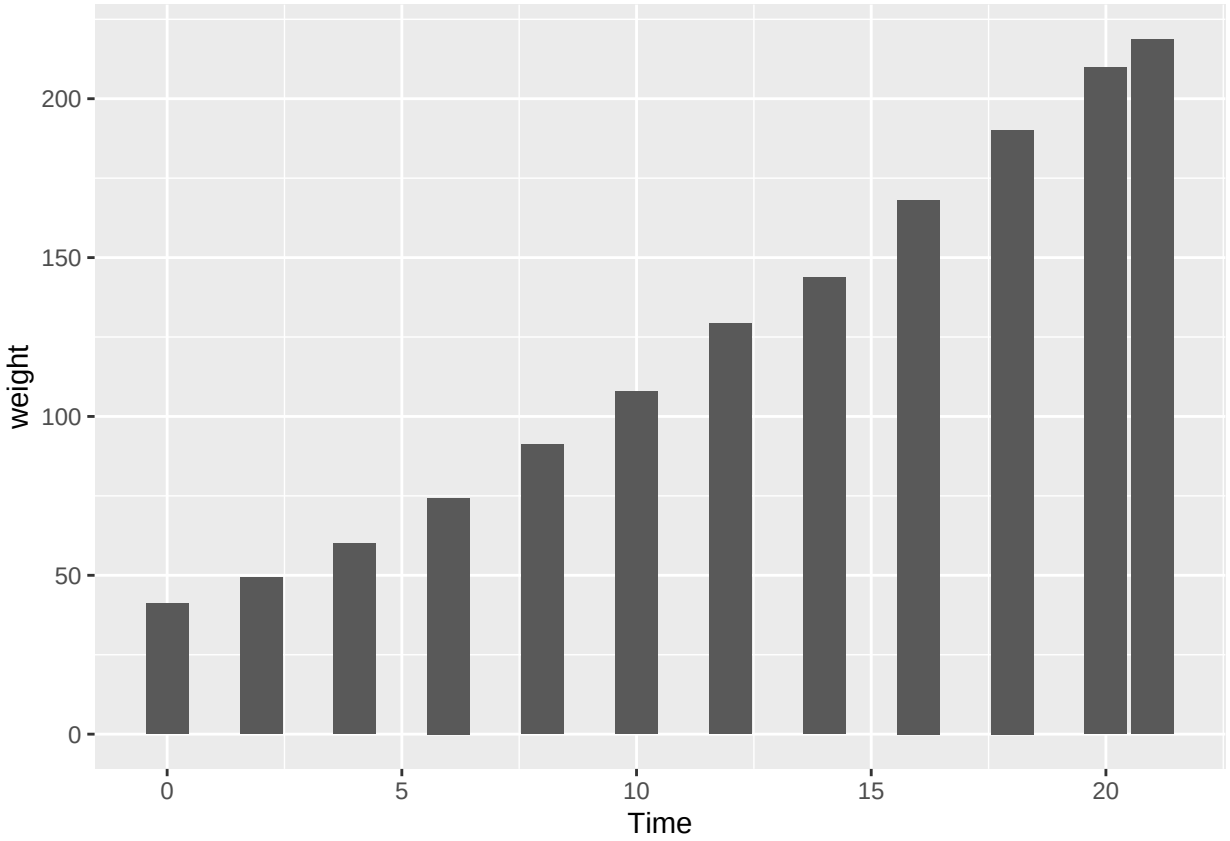
```
ChickWeight$Diet <- factor(ChickWeight$Diet, labels = c("Diet 1", "Diet 2", "Diet 3", "Diet 4"))
summary(ChickWeight)
```

```
##      weight           Time           Chick         Diet
##  Min.   : 35.0   Min.   : 0.00   13     : 12   Diet 1:220
##  1st Qu.: 63.0   1st Qu.: 4.00   9      : 12   Diet 2:120
##  Median :103.0   Median :10.00   20     : 12   Diet 3:120
##  Mean   :121.8   Mean   :10.72   10     : 12   Diet 4:118
##  3rd Qu.:163.8   3rd Qu.:16.00   17     : 12
##  Max.   :373.0   Max.   :21.00   19     : 12
##                                  (Other):506
```

Color coding in bar graphs

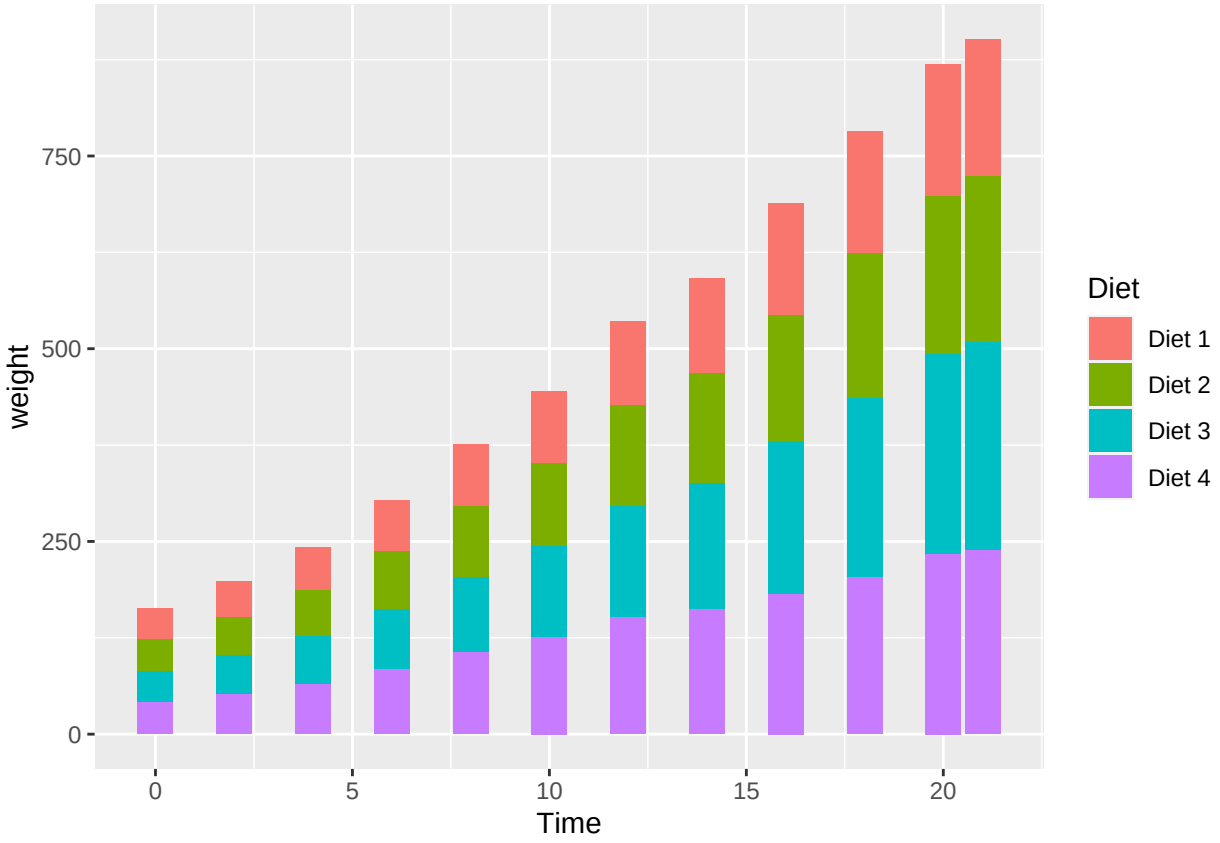
Let's take a look at the basic patterns in the data: how does the weight of chicks change over time?

```
ggplot(ChickWeight, aes(x=Time, y=weight)) +
  geom_bar(stat="summary", fun="mean")
```

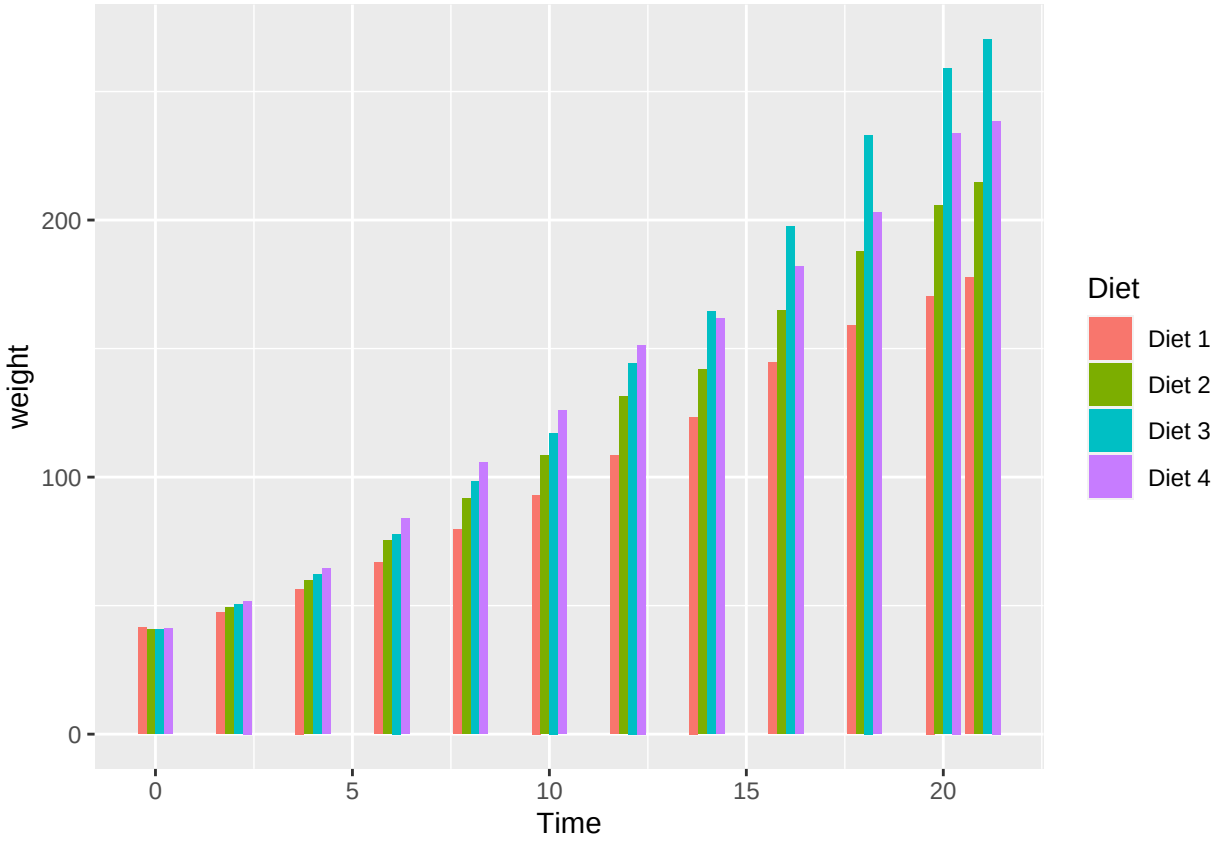



```
# The stat="summary" section tells R to use the mean of all the values for each timestamp.
# When you use stat="summary", you should also include fun="mean".
# stat="identity" is used when you only have one value per bar; here it would tell R to sum all of the
# stat="count" counts the number of times that a category (in this case, each timestamp) appears.
```

```
# Now use fill=Diet to color code the bars based on the chicks' diet. What happens?
ggplot(ChickWeight, aes(x=Time, y=weight, fill=Diet)) +
  geom_bar(stat="summary", fun="mean")
```



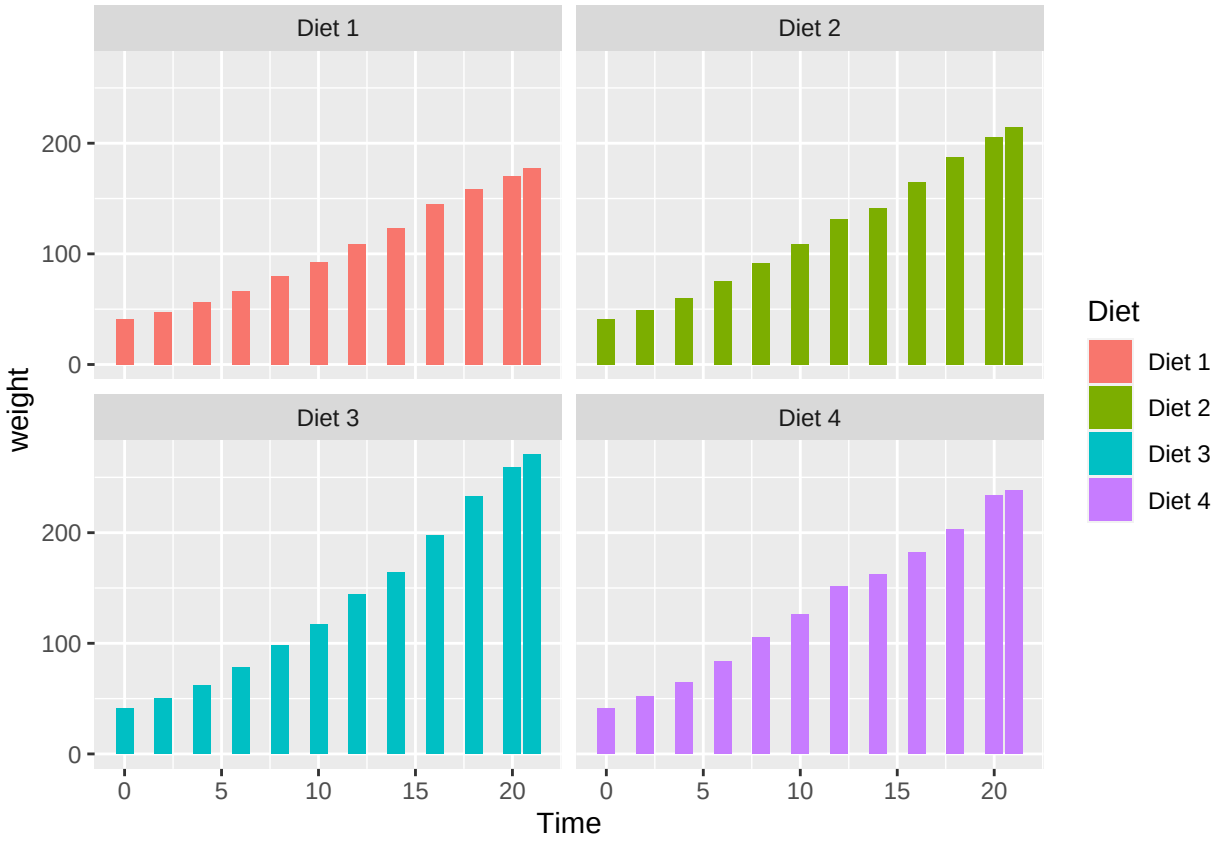

```
# to display the different-colored bars beside each other instead of stacked, use position="dodge"
ggplot(ChickWeight, aes(y=weight, x=Time, fill=Diet)) +
  geom_bar(stat="summary", fun="mean", position="dodge")
```



Faceting plots

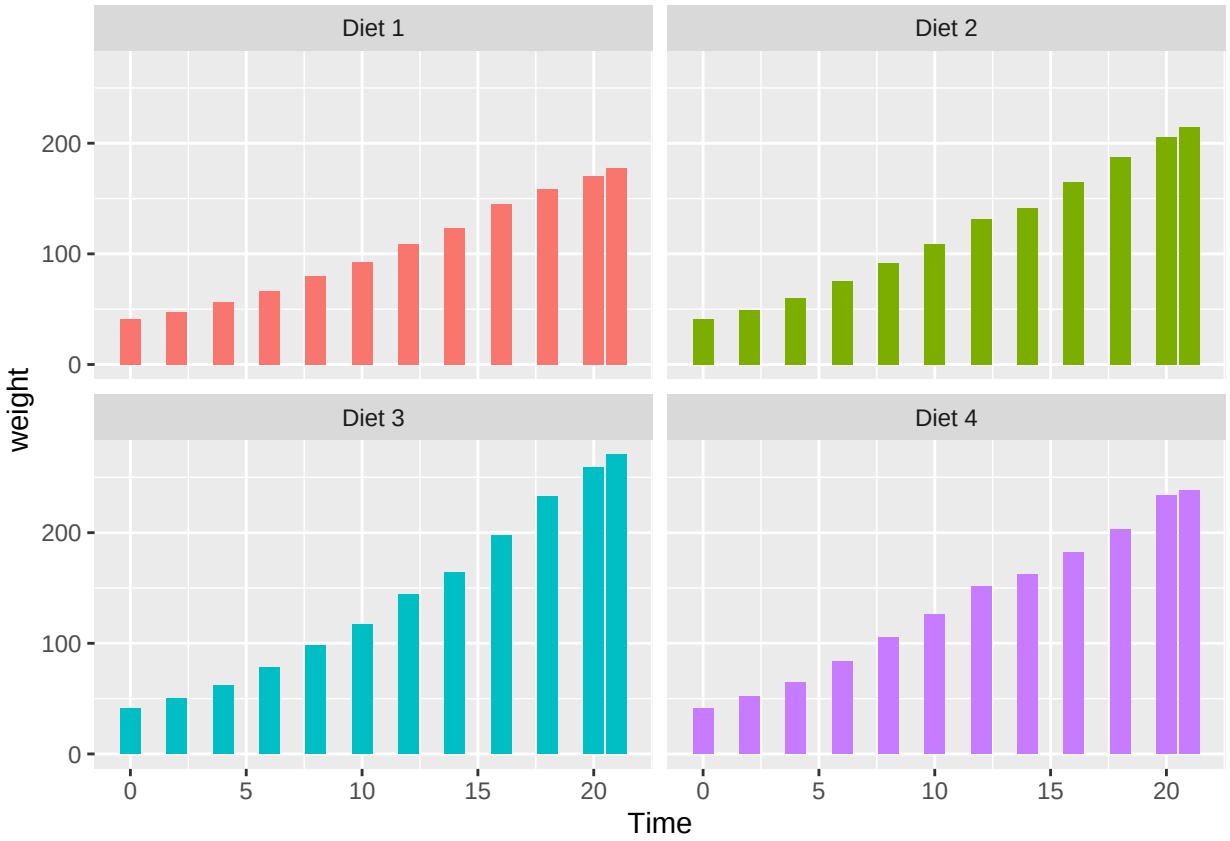
In the plot above, it can be hard to see the bars since they're so thin and tightly clustered together. There are many potential ways to go about making a graph more visually appealing, but one that would work well here is `facet_wrap`. `facet_wrap` and `facet_grid` divide plots into multiple panels by variable!

```
# facet the barplot into panels
ggplot(ChickWeight, aes(y=weight, x=Time, fill=Diet)) +
  geom_bar(stat="summary", fun="mean") +
  facet_wrap(vars(Diet))
```



Now that the different diets are all separated out and labeled, the legend to the side is redundant and taking up room. We can remove it by using `legend.position` in the `theme()` part of our ggplot function.

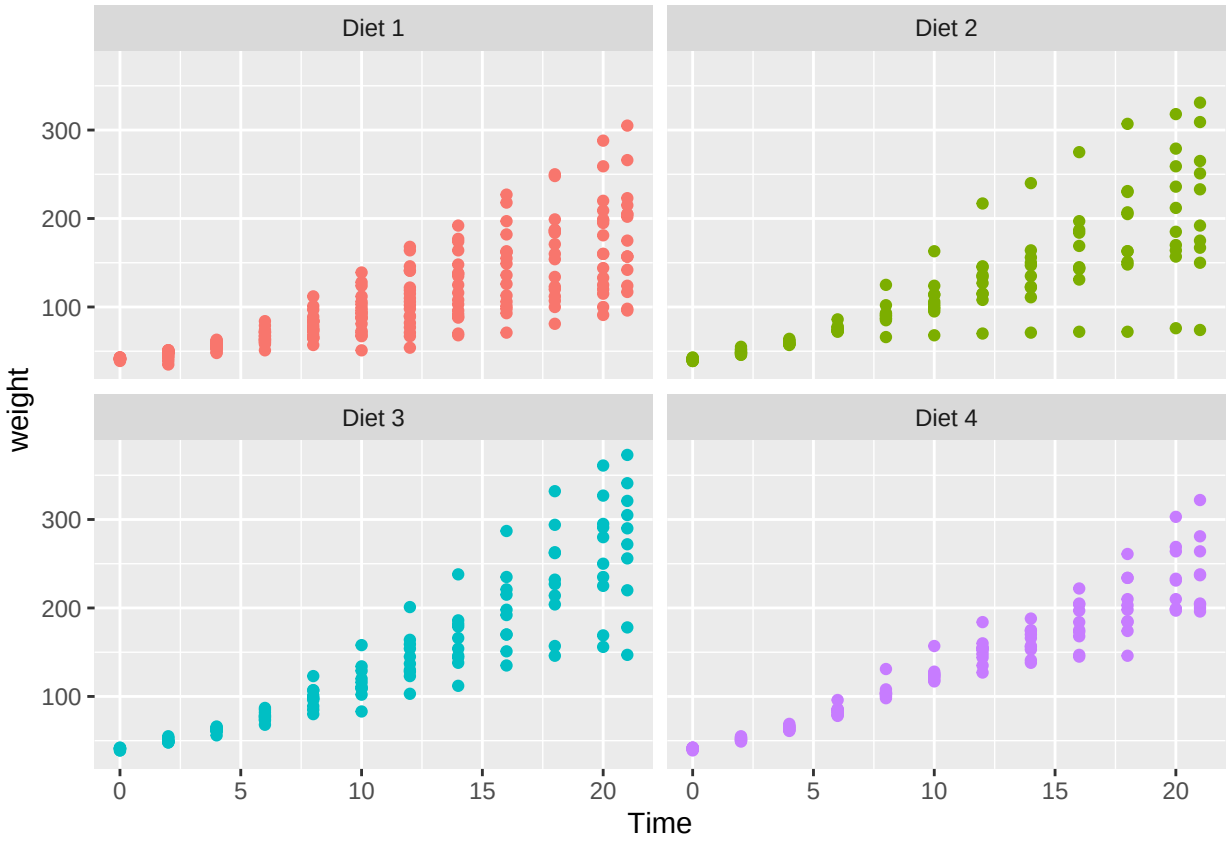
```
# remove the legend
ggplot(ChickWeight, aes(y=weight, x=Time, fill=Diet)) +
  geom_bar(stat="summary", fun="mean") +
  facet_wrap(vars(Diet)) +
  theme(legend.position = "none")
```



Customizing scatterplots

Is a bar graph really the best way to look at this data? After all, there are 30 chicks in this study, and there's only one bar for each time they were weighed, so there's a lot of variation here that's being hidden. One way to see some of this variation is by using a scatterplot.

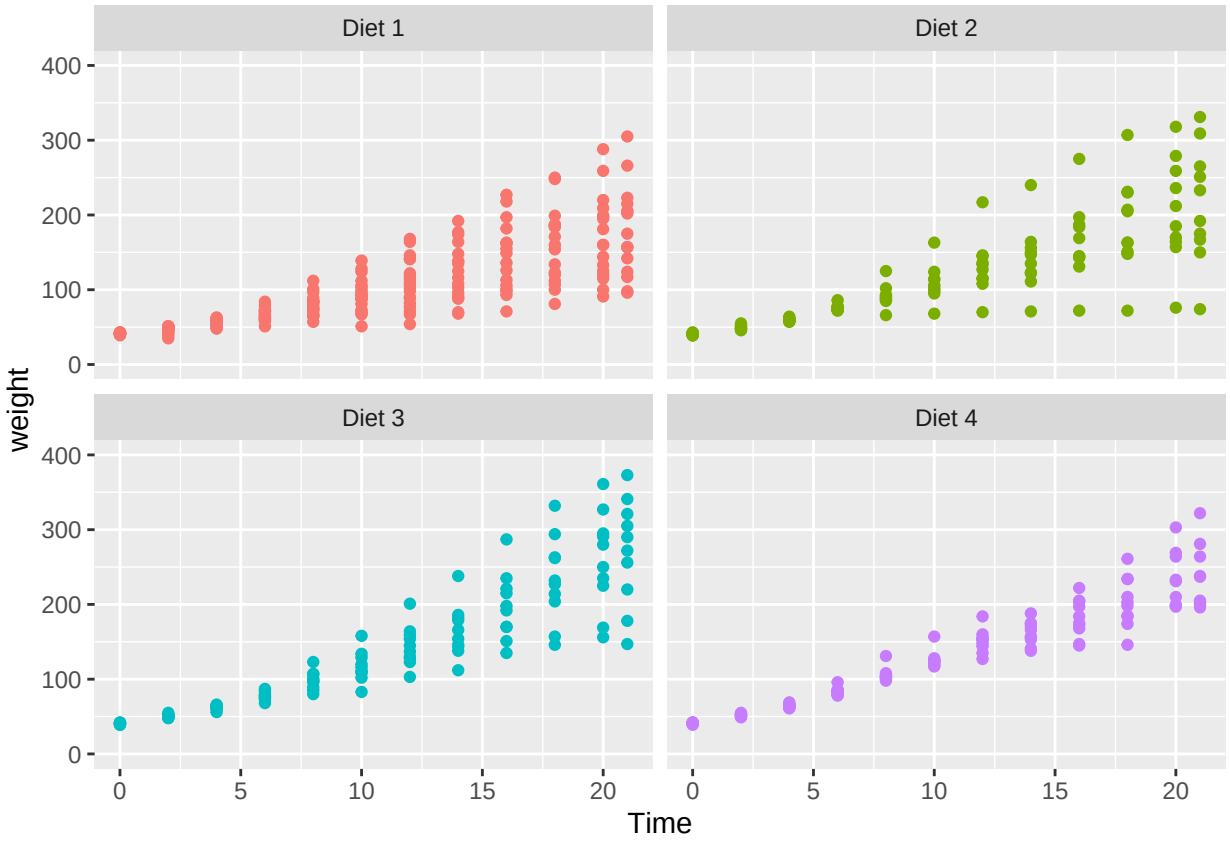
```
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_point() +
  facet_wrap(vars(Diet)) +
  theme(legend.position = "none")
```



This provides a lot more information: for instance, we see that all the chicks started out at almost exactly the same weight, and that they then showed much more variation over time. We also see that Diet 4 had the narrowest range (these chickens had the most consistent weights), and that there were some outliers with Diet 2.

However, one issue that we see with this graph is that the y-axis doesn't start at 0. We can adjust that by adding a `scale` function. Here, we're using `scale_y_continuous` because it's a continuous numerical value (there are other functions for the x-axis and for categorical variables), and we're setting the range from 0 to 400. There are other parameters that can be set using this function: for example, `breaks` allows you to set where the tick marks fall on the axis.

```
# add a custom range for the y-axis
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_point() +
  facet_wrap(vars(Diet)) +
  scale_y_continuous(limits = c(0,400)) +
  theme(legend.position = "none")
```

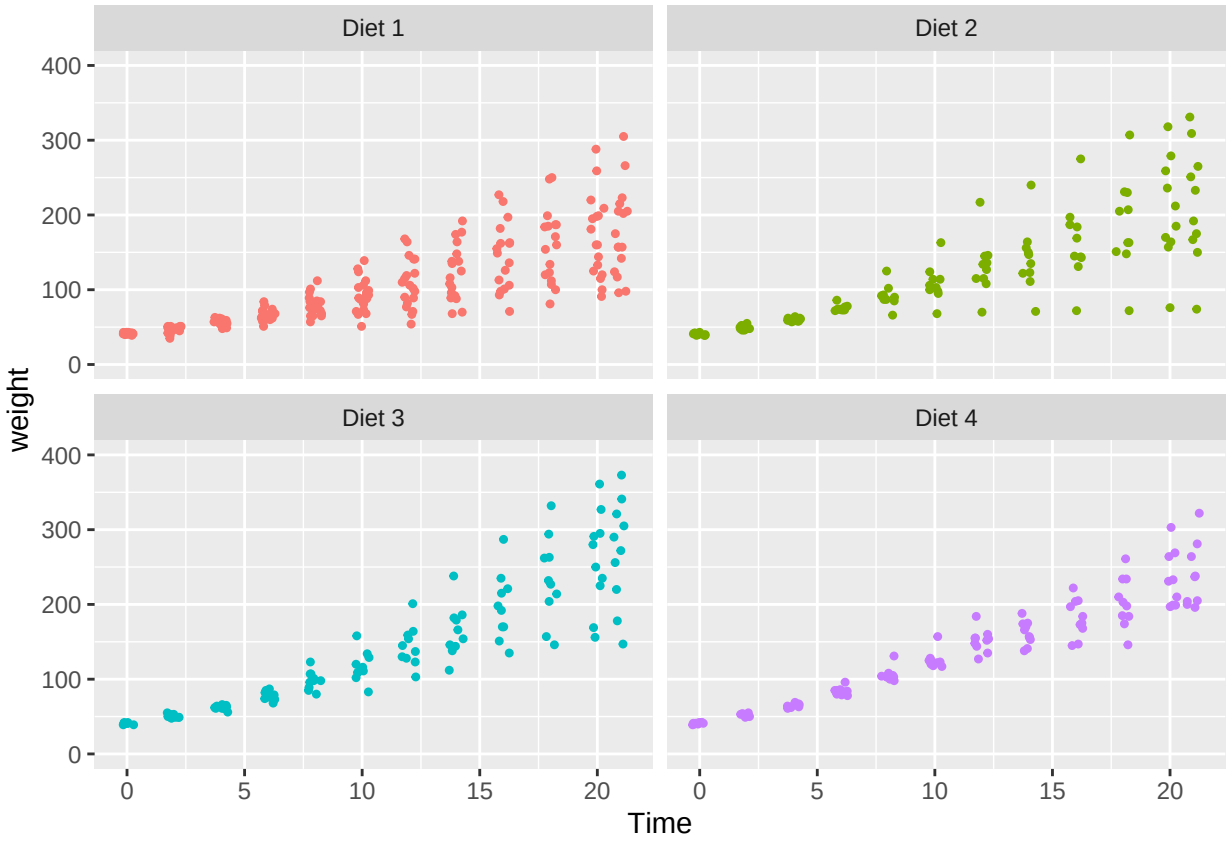


Something else we see is that, especially at the earlier timepoints, all of the chicks have very similar weights, so it's hard to tell how the weights are distributed. (For example, on Day 8, do more of the Diet 1 chicks weigh ~50g or ~100g?)

One way to make this easier to see is by using `geom_jitter` instead of `geom_point`. `geom_jitter` slightly offsets all the dots randomly so that they end up spaced out from each other. Here, we're only going to allow the dots to move left-to-right, not up-and-down, since movement along the vertical axis would represent a change in weight, while the days on the x-axis are more discrete. You can adjust how much your data spreads out by using the `width` parameter.

To make the points even easier to distinguish, you can change their size using the `size` parameter.

```
# jitter and shrink the points; set the height to 0 so it will only jitter left-to-right
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_jitter(size=0.9, width=.3, height=0) +
  facet_wrap(vars(Diet)) +
  scale_y_continuous(limits = c(0,400)) +
  theme(legend.position = "none")
```



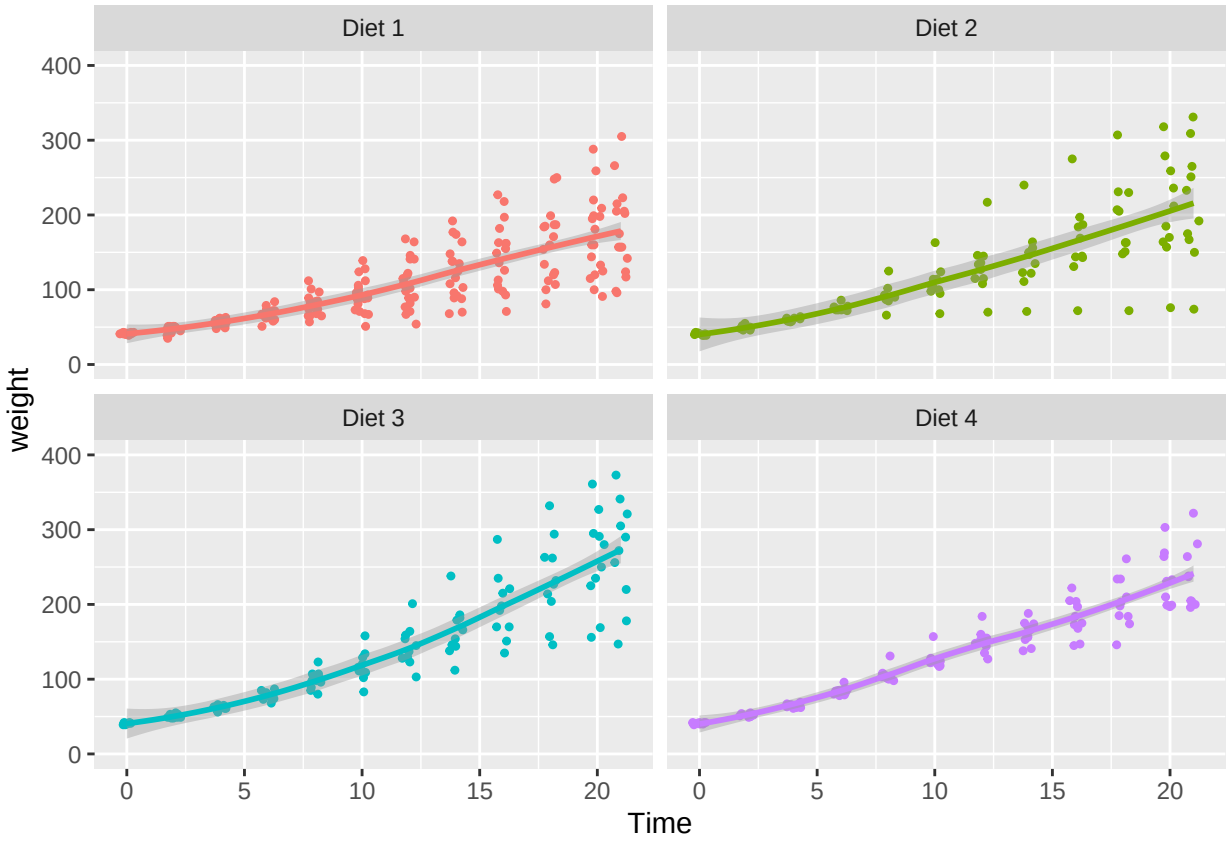
Regression lines

From the graphs above, we can see a pretty clear linear trajectory in the chicks' weight over time. Without getting into the process of calculating a statistical linear regression, we can overlay a line of best fit on top of the plot by using `geom_smooth`. (You can use as many geoms as you like in a single plot!)

To calculate a linear regression, we use `geom_smooth(method="lm")`. But what happens if we just use `geom_smooth()`?

```
# overlay the plot with a smoothed line...what happens?
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_jitter(size=0.9, width=.3, height=0) +
  geom_smooth() +
  facet_wrap(vars(Diet)) +
  scale_y_continuous(limits = c(0,400)) +
  theme(legend.position = "none")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

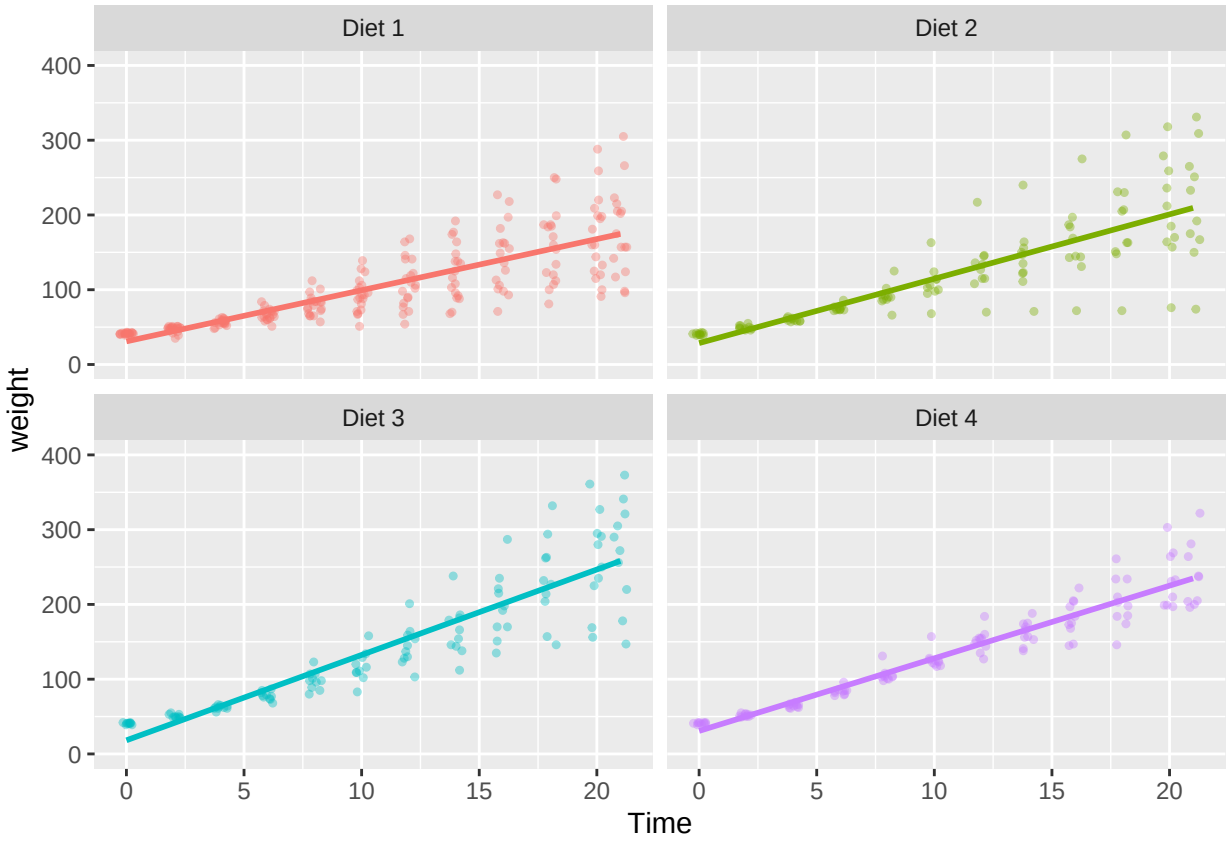


Using this method, we get a slightly curved line with a shaded confidence interval. (With different data, we might get a much more curved line, or even one that wiggles up and down.) While this may more accurately represent the trajectory of the data, it's less useful if you want to calculate a linear regression as part of your statistics. For a straight line, use the code below:

(To make the line stand out more, it also helps to make the points on the graph slightly transparent. This is done by adding `alpha` to `geom_jitter`; 0=fully transparent and 1=fully opaque.)

```
# overlay a straight regression line (with no confidence interval) on the scatterplot
# make the points more transparent
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_jitter(size=0.9, width=.3, height=0, alpha=0.4) +
  geom_smooth(method="lm", se=FALSE) +
  facet_wrap(vars(Diet)) +
  scale_y_continuous(limits = c(0,400)) +
  theme(legend.position = "none")
```

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

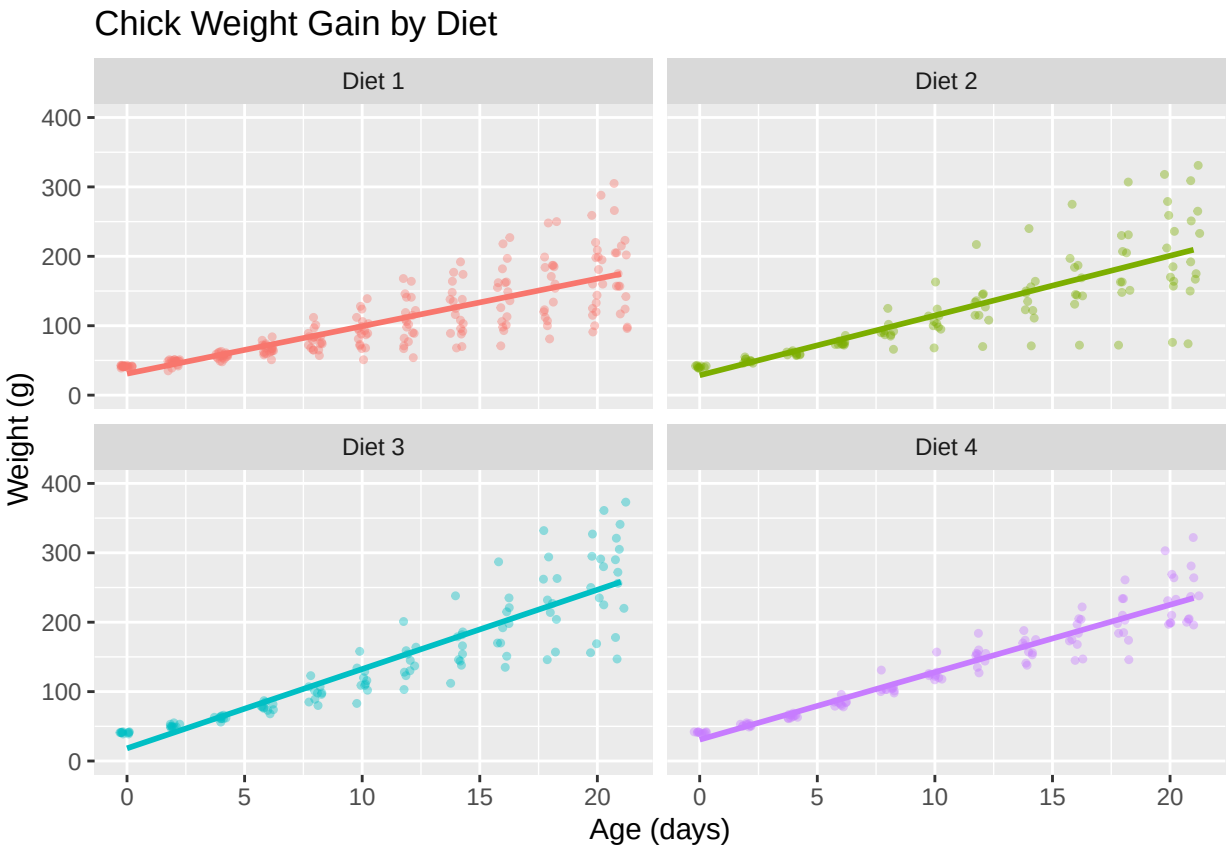


Modifying the theme

Now that we have an interesting type of graph, let's play with its appearance. First, let's make the axis labels more accurate and add a title.

```
# add axis labels and title
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_jitter(size=0.9, width=.3, height=0, alpha=0.4) +
  geom_smooth(method="lm", se=FALSE) +
  facet_wrap(vars(Diet)) +
  scale_y_continuous(limits = c(0,400)) +
  xlab("Age (days)") + ylab("Weight (g)") + ggtitle("Chick Weight Gain by Diet") +
  theme(legend.position = "none")
```

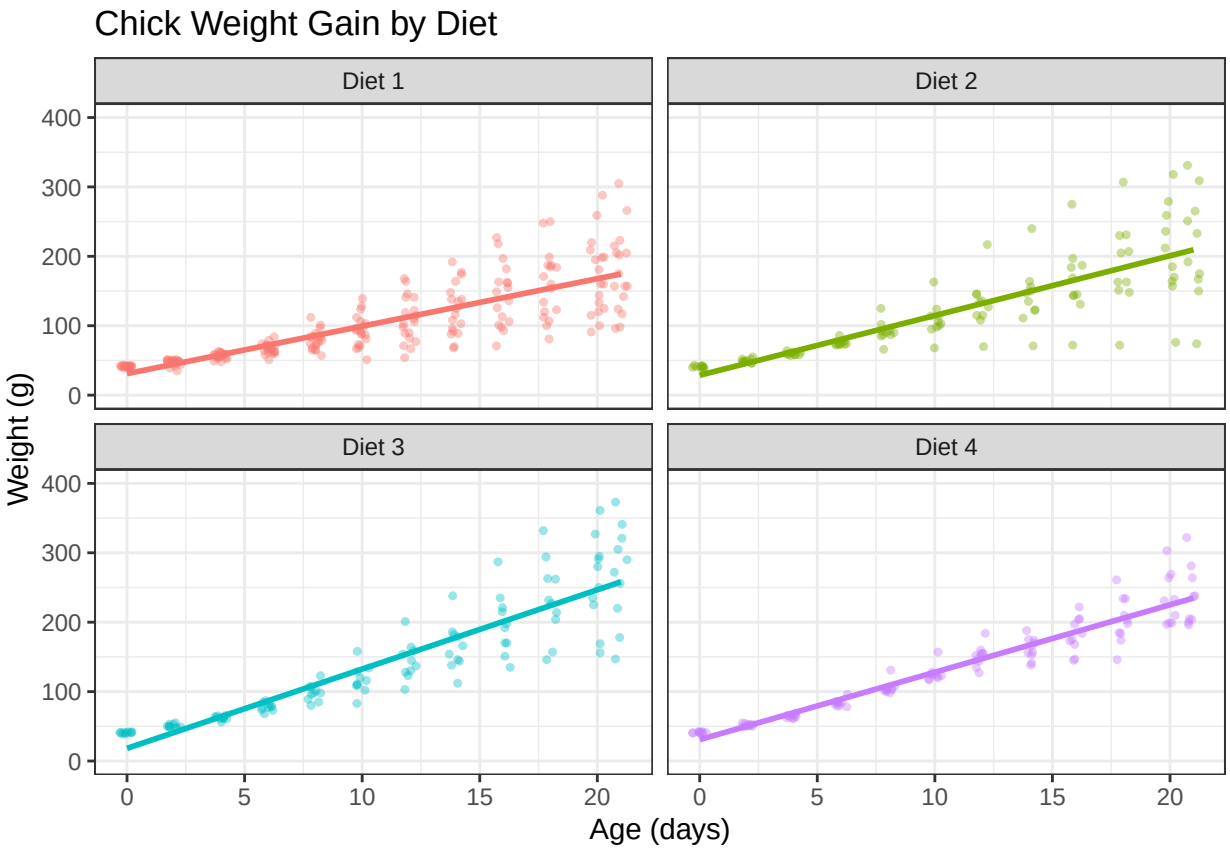
```
## `geom_smooth()` using formula 'y ~ x'
```



Now, play around with the theme. Some of the preset options are `theme_bw()`, `theme_classic()`, `theme_minimal()`, `theme_light()`, `theme_dark()`...pick a favorite. Make sure to add your theme BEFORE the `theme` function where you've removed the legend, or it will erase all the theme edits you make.

```
# add a preset theme
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_jitter(size=0.9, width=.3, height=0, alpha=0.4) +
  geom_smooth(method="lm", se=FALSE) +
  facet_wrap(vars(Diet)) +
  scale_y_continuous(limits = c(0,400)) +
  xlab("Age (days)") + ylab("Weight (g)") + ggtitle("Chick Weight Gain by Diet") +
  theme_bw() +
  theme(legend.position = "none")
```

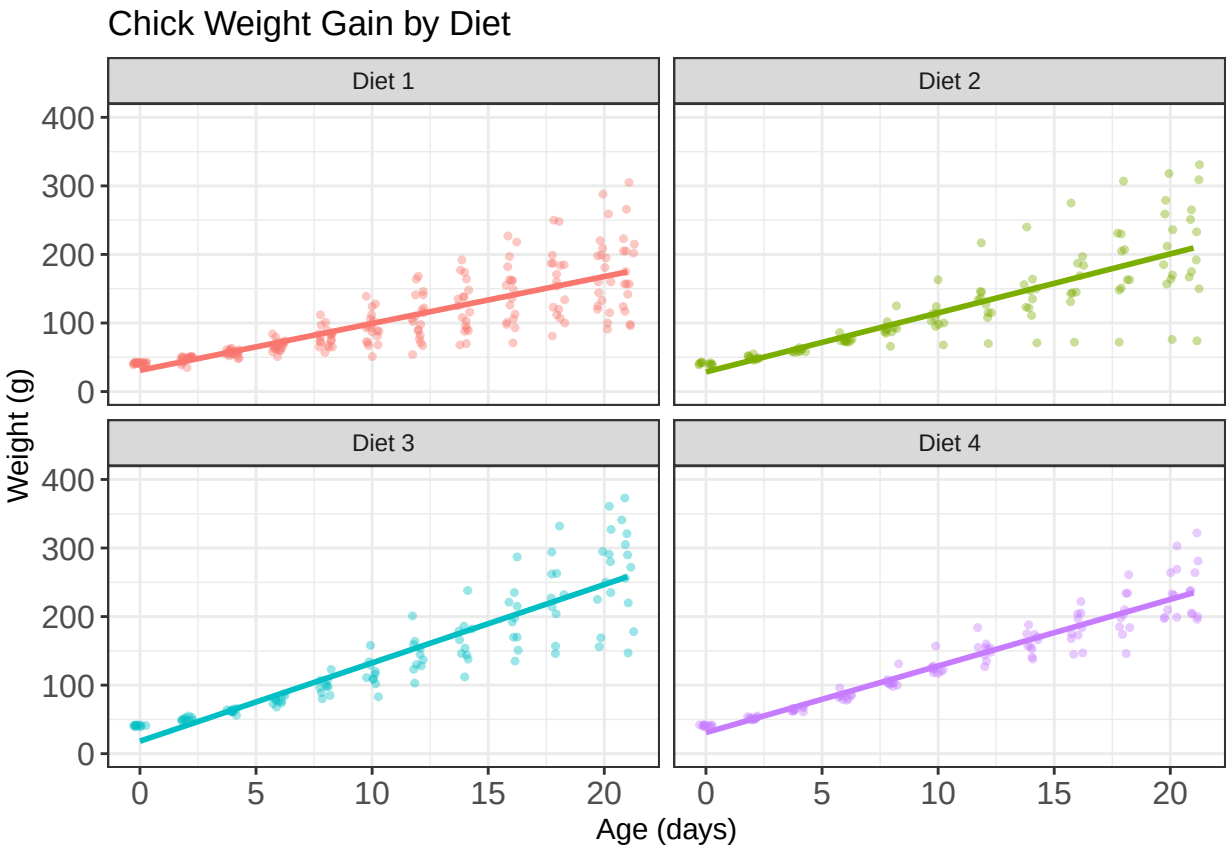
```
## `geom_smooth()` using formula 'y ~ x'
```



Next, let's adjust the size of the text in various places on the graph.

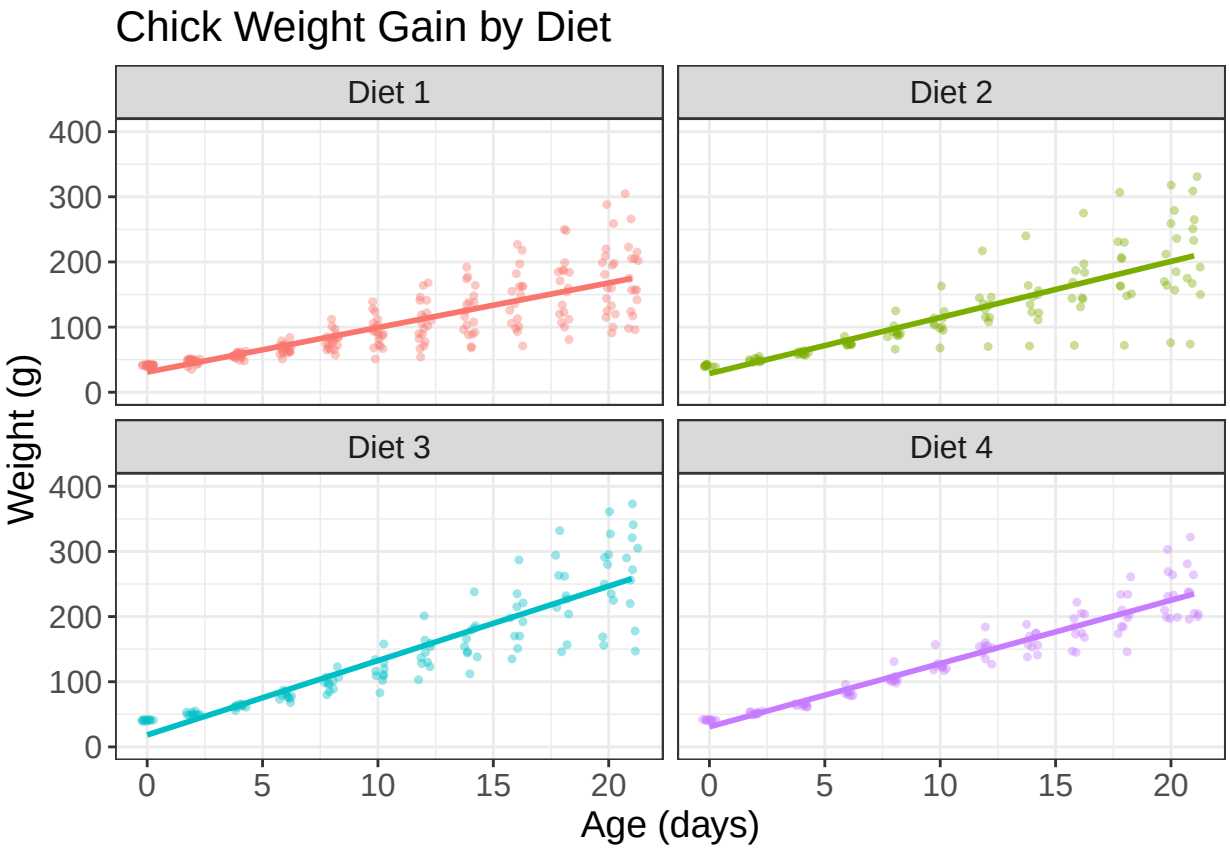
```
# enlarge font size on axis ticks
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_jitter(size=0.9, width=.3, height=0, alpha=0.4) +
  geom_smooth(method="lm", se=FALSE) +
  facet_wrap(vars(Diet)) +
  scale_y_continuous(limits = c(0,400)) +
  xlab("Age (days)") + ylab("Weight (g)") + ggtitle("Chick Weight Gain by Diet") +
  theme_bw() +
  theme(legend.position = "none",
        axis.text = element_text(size=12))
```

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




```
# enlarge font size on axis labels, title, and the panel headers (where it says "Diet #")
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_jitter(size=0.9, width=.3, height=0, alpha=0.4) +
  geom_smooth(method="lm", se=FALSE) +
  facet_wrap(vars(Diet)) +
  scale_y_continuous(limits = c(0,400)) +
  xlab("Age (days)") + ylab("Weight (g)") + ggtitle("Chick Weight Gain by Diet") +
  theme_bw() +
  theme(legend.position = "none",
        axis.text = element_text(size=12),
        axis.title = element_text(size=14),
        strip.text = element_text(size=12),
        plot.title = element_text(size=16))
```

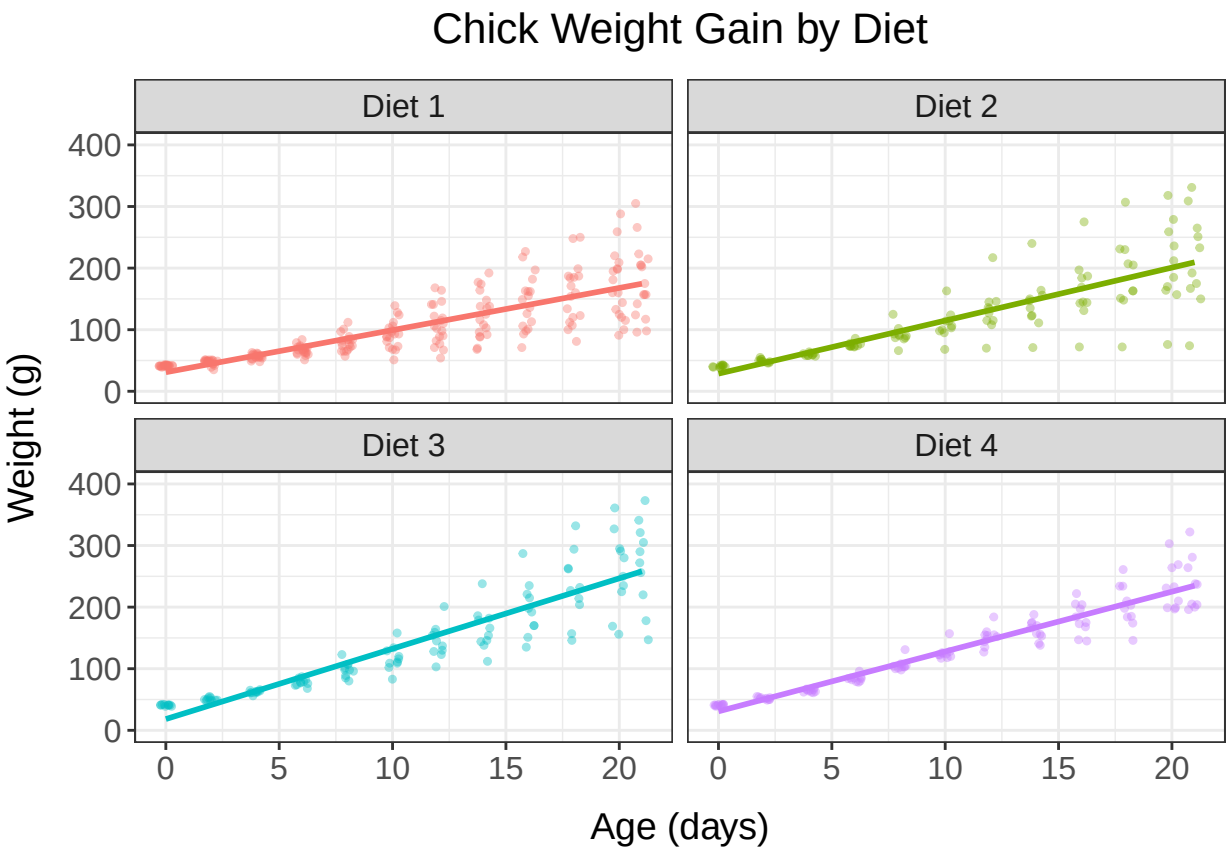
```
## `geom_smooth()` using formula 'y ~ x'
```



You can also change the position of labels. For the axis labels, they look a bit too close to the numbers on the tick marks, so we can add a margin of 10 pixels at the top of the x-axis label and to the right of the y-axis label. We can also add some space under the title and center it using `hjust` (0=left, 0.5=centered, 1=right).

```
# add margins around labels and title, center title
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_jitter(size=0.9, width=.3, height=0, alpha=0.4) +
  geom_smooth(method="lm", se=FALSE) +
  facet_wrap(vars(Diet)) +
  scale_y_continuous(limits = c(0,400)) +
  xlab("Age (days)") + ylab("Weight (g)") + ggtitle("Chick Weight Gain by Diet") +
  theme_bw() +
  theme(legend.position = "none",
        axis.text = element_text(size=12),
        axis.title = element_text(size=14),
        strip.text = element_text(size=12),
        plot.title = element_text(size=16, hjust=0.5, margin = margin(b=10)),
        axis.title.x = element_text(margin = margin(t=10)),
        axis.title.y = element_text(margin = margin(r=10)))
```

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



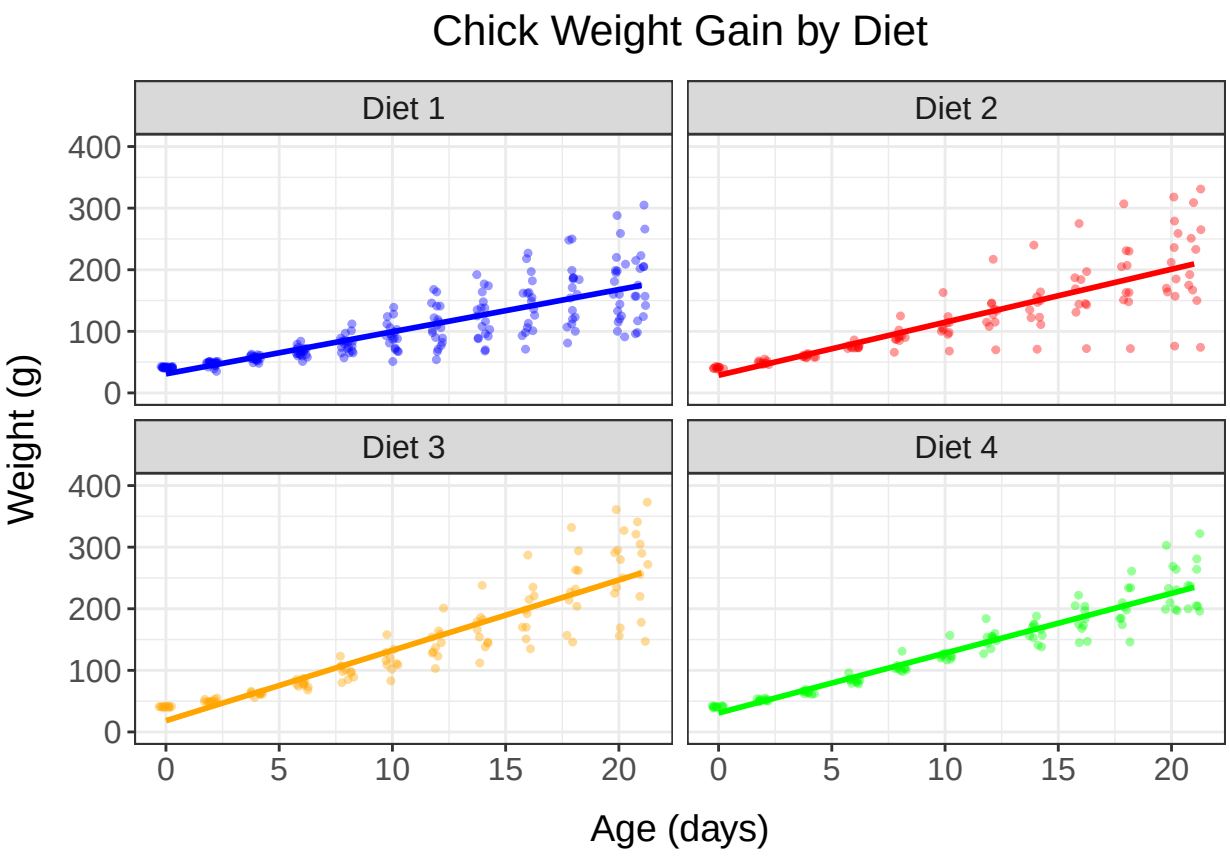
Color

The color scheme in the above graph is ggplot2's default color setting, which is designed to be evenly spaced around the color wheel. However, if you don't like this color scheme, or it isn't working well for a particular plot, you can set your own color schemes, either manually or by using a preset palette.

To set a color scheme manually, use `scale_color_manual` and list the colors that you want to appear. R recognizes the names of many colors and uses them as shortcuts for specific RGB/hexadecimal values, and you can search for a specific color using this manual.

```
# set colors manually
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_jitter(size=0.9, width=.3, height=0, alpha=0.4) +
  geom_smooth(method="lm", se=FALSE) +
  facet_wrap(vars(Diet)) +
  scale_y_continuous(limits = c(0,400)) +
  scale_color_manual(values = c("blue", "red", "orange", "green")) +
  xlab("Age (days)") + ylab("Weight (g)") + ggtitle("Chick Weight Gain by Diet") +
  theme_bw() +
  theme(legend.position = "none",
        axis.text = element_text(size=12),
        axis.title = element_text(size=14),
        strip.text = element_text(size=12),
        plot.title = element_text(size=16, hjust=0.5, margin = margin(b=10)),
        axis.title.x = element_text(margin = margin(t=10)),
        axis.title.y = element_text(margin = margin(r=10)))
```

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

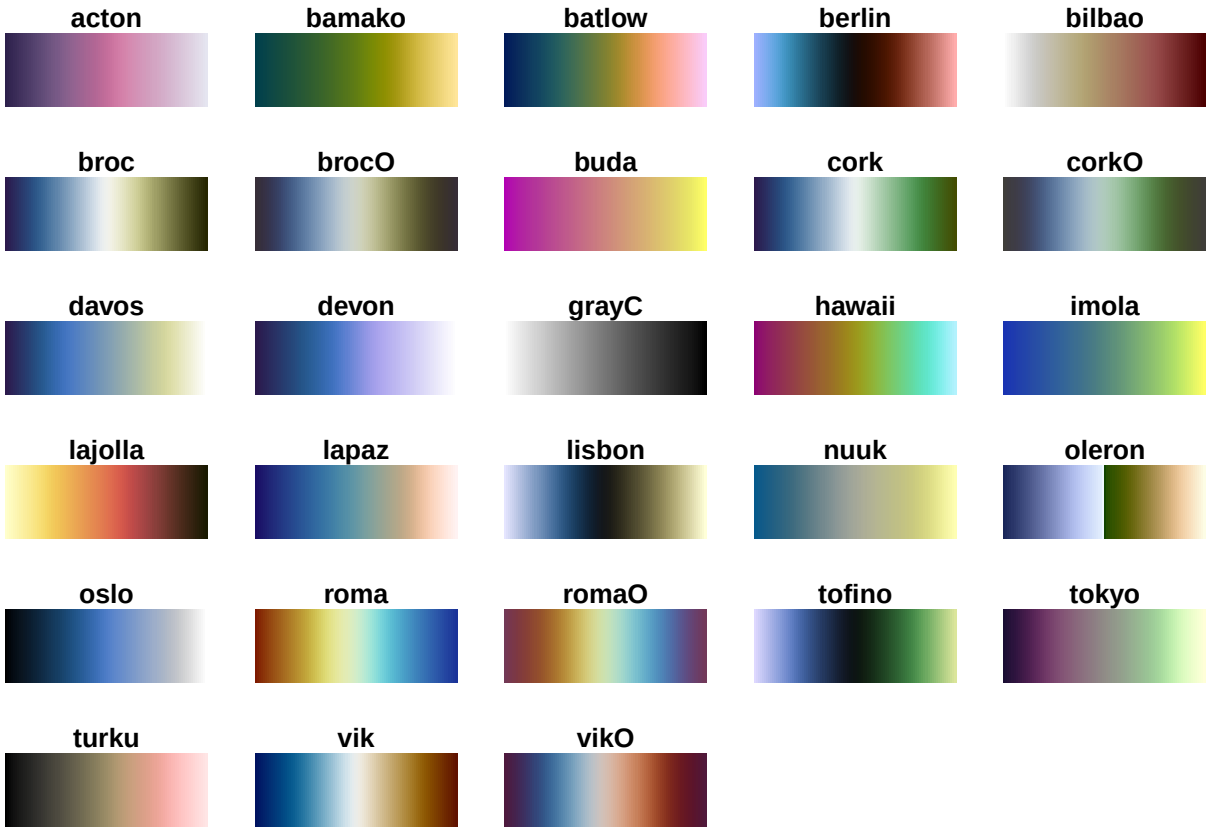


Alternatively, you can use one of the preset color palettes defined by R, which can be viewed here. There are also several packages that you can add with additional attractive color scales, such as viridis, scico (shown below), and even one designed to match the color palettes of Wes Anderson movies, wesanderson.

```
# install.packages("scico")
library(scico)
```

```
## Warning: package 'scico' was built under R version 4.1.1
```

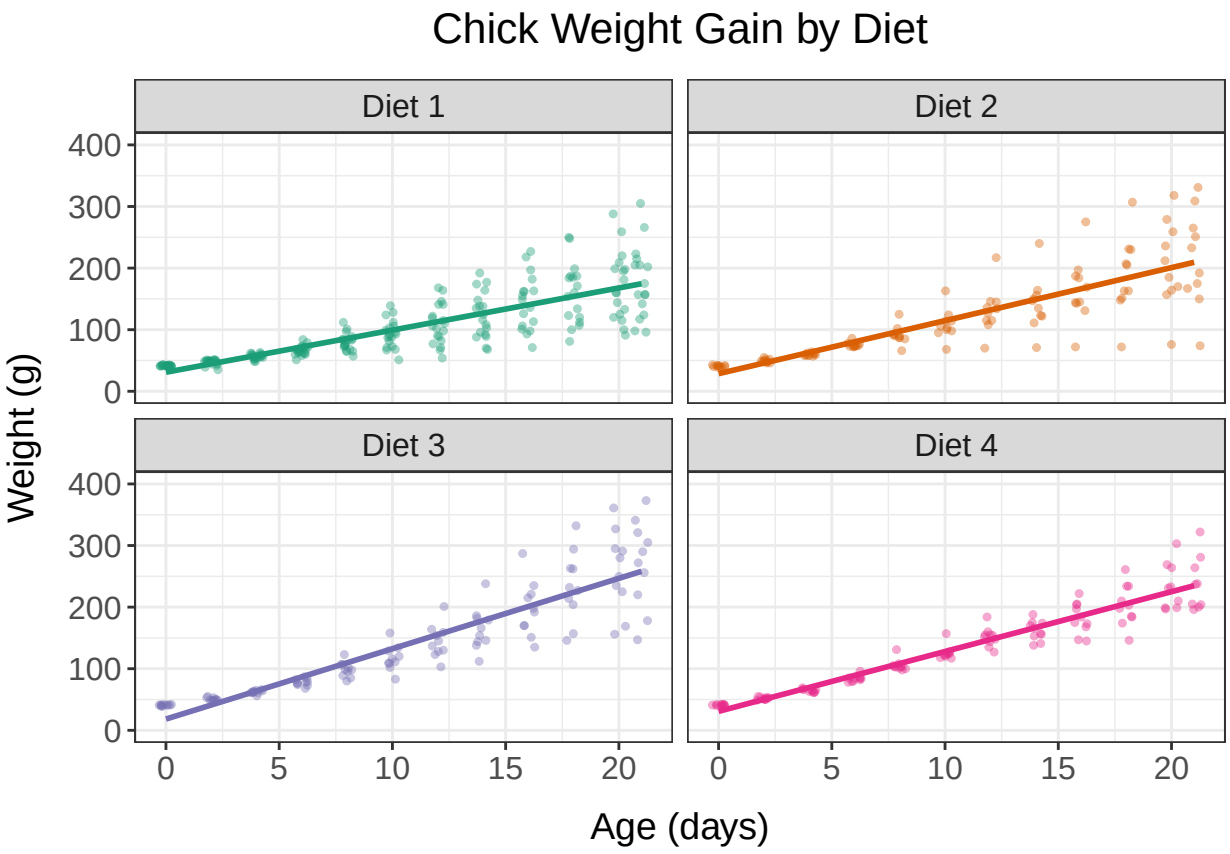
```
scico_palette_show()
```



To use one of these palettes for your graph, use `scale_color_brewer`.

```
# choose a color palette
ggplot(ChickWeight, aes(y=weight, x=Time, color=Diet)) +
  geom_jitter(size=0.9, width=.3, height=0, alpha=0.4) +
  geom_smooth(method="lm", se=FALSE) +
  facet_wrap(vars(Diet)) +
  scale_y_continuous(limits = c(0,400)) +
  scale_color_brewer(type = "qual", palette = "Dark2") +
  xlab("Age (days)") + ylab("Weight (g)") + ggtitle("Chick Weight Gain by Diet") +
  theme_bw() +
  theme(legend.position = "none",
        axis.text = element_text(size=12),
        axis.title = element_text(size=14),
        strip.text = element_text(size=12),
        plot.title = element_text(size=16, hjust=0.5, margin = margin(b=10)),
        axis.title.x = element_text(margin = margin(t=10)),
        axis.title.y = element_text(margin = margin(r=10)))
```

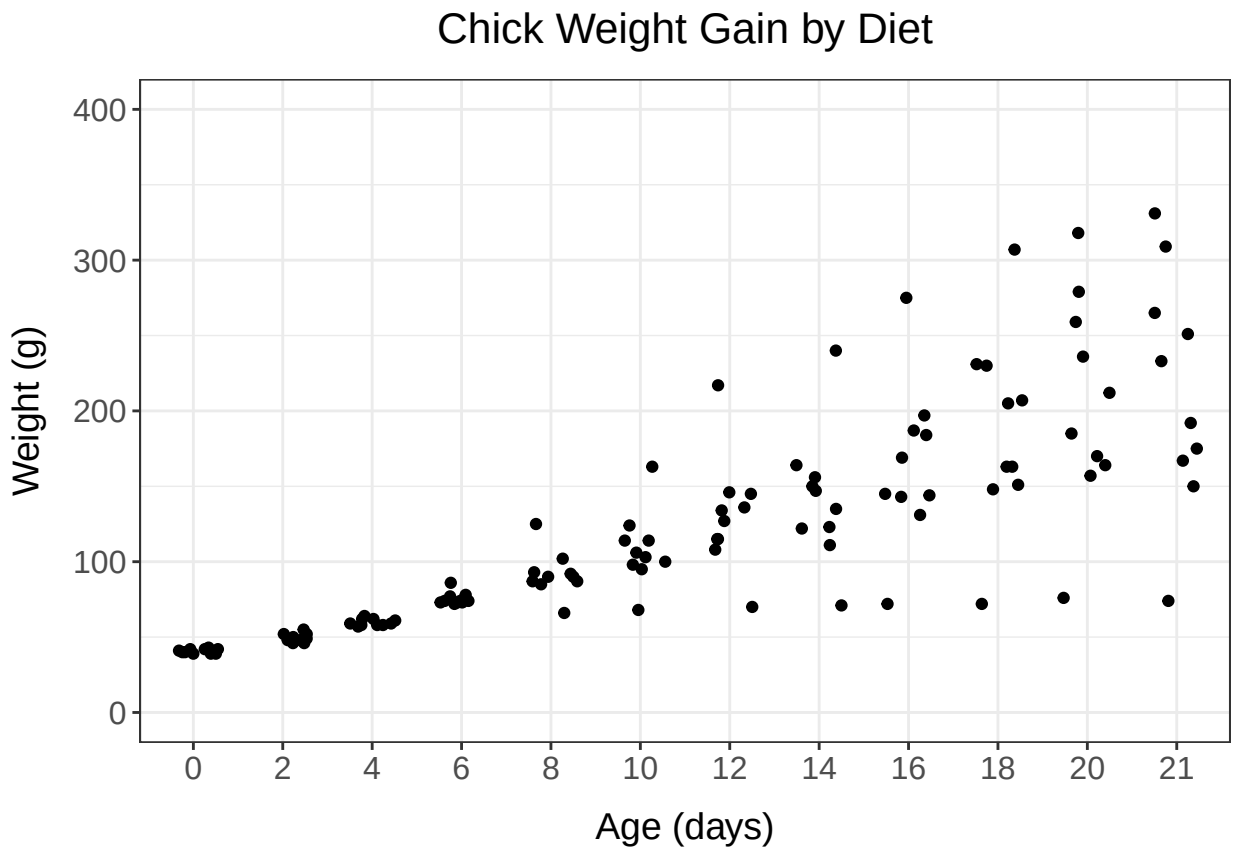
```
## `geom_smooth()` using formula 'y ~ x'
```



Text labels

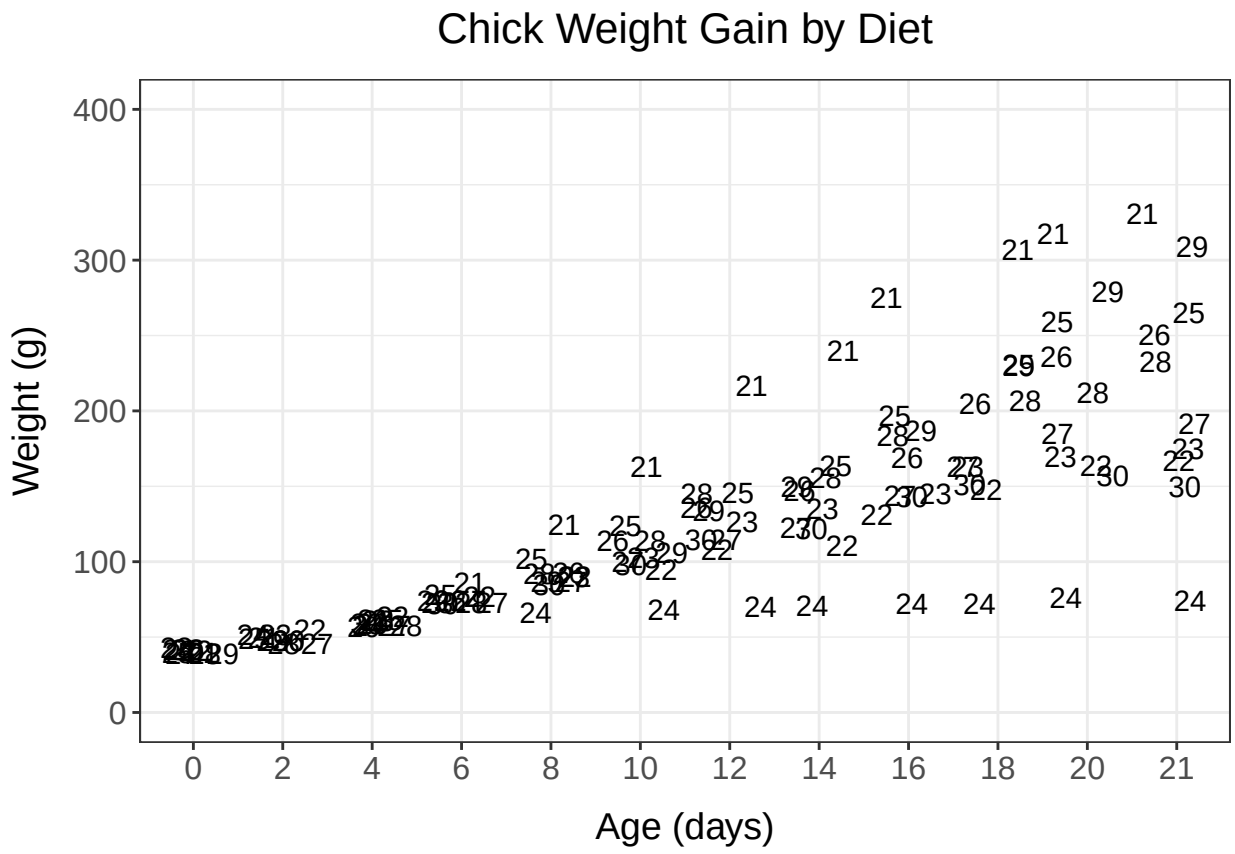
As a last point, let's look at just one of the sets of chickens, the Diet 2 group.

```
ggplot(subset(ChickWeight, Diet == "Diet 2"), aes(y=weight, x=factor(Time))) +
  geom_jitter(size=1.5, width=.3, height=0) +
  scale_y_continuous(limits = c(0,400)) +
  scale_colour_brewer(type = "qual", palette = "Dark2") +
  xlab("Age (days)") + ylab("Weight (g)") + ggtitle("Chick Weight Gain by Diet") +
  theme_bw() +
  theme(legend.position = "none",
        axis.text = element_text(size=12),
        axis.title = element_text(size=14),
        strip.text = element_text(size=12),
        plot.title = element_text(size=16, hjust=0.5, margin = margin(b=10)),
        axis.title.x = element_text(margin = margin(t=10)),
        axis.title.y = element_text(margin = margin(r=10)))
```



Now, instead of using dots, let's label each data point with the chicken's ID number using `geom_text`. (If you want a rectangle around the text, you can use `geom_label` instead.) So that the labels won't overlap too much, add `position = position_jitter` as an attribute.

```
ggplot(subset(ChickWeight, Diet == "Diet 2"), aes(y=weight, x=factor(Time))) +
  geom_text(aes(label=Chick), position = position_jitter(height=0)) +
  scale_y_continuous(limits = c(0,400)) +
  scale_colour_brewer(type = "qual", palette = "Dark2") +
  xlab("Age (days)") + ylab("Weight (g)") + ggtitle("Chick Weight Gain by Diet") +
  theme_bw() +
  theme(legend.position = "none",
        axis.text = element_text(size=12),
        axis.title = element_text(size=14),
        strip.text = element_text(size=12),
        plot.title = element_text(size=16, hjust=0.5, margin = margin(b=10)),
        axis.title.x = element_text(margin = margin(t=10)),
        axis.title.y = element_text(margin = margin(r=10)))
```



By doing this, we can see that all the outlier dots that were higher than the others belonged to the same chick (#21), and all the ones that were lower belonged to chick #24.

Conclusions

This tutorial discussed some of the customization abilities of ggplot2, including faceting plots, altering the color and size of points, adding regression lines, changing font sizes of labels, and using color palettes; however, there are still many more ways to customize the appearance of a graph using ggplot2! To learn more about these methods, we recommend the resources below!

Web-Based Resources

- Ggplot2: Elegant Graphics for Data Analysis, by Wickam et al.
- Ggplot2 cheat sheet
- Plotting anything with ggplot2 webinar
- R Graphics Cookbook
- R Color Names
- Palettes in RColorBrewer

Resources at UGA

- Linked-in Learning Tutorials
- Previous and upcoming DigiLab Tutorials
- DigiLab Data Consultations

- R Ladies Athens

Questions, comments, or suggestions? Get in touch at: - DigiLab Data Consultations - keiko.bridwell@uga.edu - katherine.kuiper25@uga.edu

Coming up next:

- Introduction to network analysis on October 20th
- Mapping Geographical Data on October 27th

Works Cited

Anderson, Edgar. 1935. The irises of the Gaspe Peninsula, Bulletin of the American Iris Society, 59, 2–5.
Chang, Winston. 2021. R Graphics Cookbook.
Dawson, Robert J. MacG. 1995. The 'Unusual Episode' Data Revisited. Journal of Statistics Education, 3. doi: 10.1080/10691898.1995.11910499.
De Vries, Andrie, and Joris Meys. 2015. R For Dummies. John Wiley & Sons.
Fay, Colin. Visualizing text data with ggplot2. Hill, Asha. 2018. 9 Useful R Data Visualization Packages for Any Discipline. Mode. https://mode.com/blog/r-data-visualization-packages/
Mejia, Mandy. 2013. 10 reasons to switch to ggplot. mandymejia.com

Pandey, Parul. 2019. A Comprehensive Guide to Data Visualization in R For Beginners. Towardsdatascience.com
Paradis, Emmanuel. 2005. R For Beginners.
Pederson, Thomas Lin. 2020. plotting anything with ggplot2 webinar.
R Studio Team. 2018. Ggplot2 Cheat Sheet
Wickam, Hadley. 2010. A layered grammar of graphics. Journal of Computational and Graphical Statistics.
Wickam, Hadley, Danielle Navarro, and Thomas Lin Pederson. 2021. ggplot2: Elegant Graphics for Data Analysis.