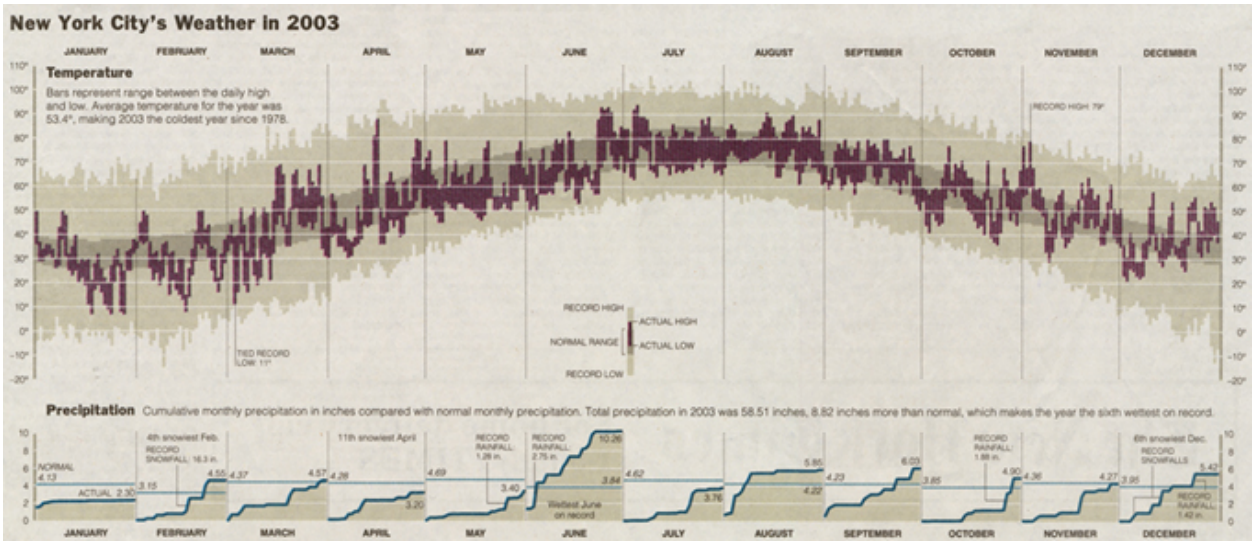
Quiz 3

Name:

1.-3.

On the following plot, label an example each of: (1) adding reference information (2) highlighting interesting data; and (3) small multiples. (1 point each.)



Source: New York Times.

4.

You run the following code:

```
library(faraway)
data(nepali)
ggplot(nepali, aes(x = ht, y = wt))
        + geom_point()
```

You get the following errors:

```
Error: No layers in plot
Error in +geom_point() : invalid argument to unary operator
```

What happened?

a. You have not yet loaded the `nepali` dataframe.
b. You are trying to create a scatterplot for a categorical variable.
c. Because you put the `+` at the start of the second line, instead of at the end of the first line, R tried to run the first line by itself, and this didn't work because the first line does not include a call for a geom.
d. The `nepali` data lacks one or both of the two columns (`ht`, `wt`) that you used in the `aes` statement of the call.

Answer c. Explanation: Anytime you try to run the `ggplot()` call without adding on a geom, you'll get the error that there are "No layers in plot". In this case, that's because R does not keep looking for additions to the first line because you've put the `+` on the beginning of the next land instead of the end of the first line. If you tried to create a scatterplot of a categorical variable, `ggplot` would do it, although it won't be a very interesting plot (try `ggplot(worldcup, aes(x = Team, y = Position)) + geom_point()`), so Answer b. is incorrect. If you hadn't loaded the data, you'd get a different error ("object `nepali` not found"), so Answer a. is incorrect. If you lacked one of the columns you're trying to plot, you'd also get a different error ("object `ht` not found", for example).

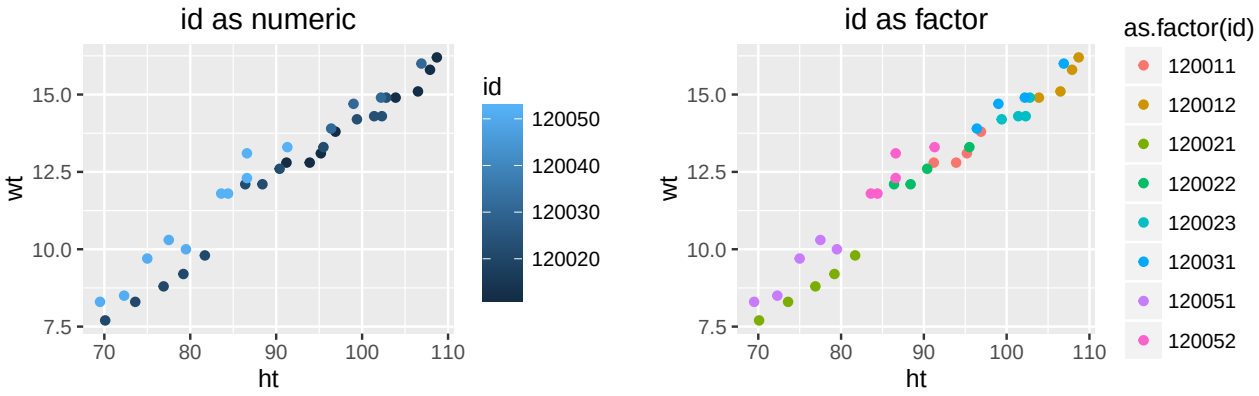
5.

You are creating a scatterplot of height (x-axis) by weight (y-axis) for the Nepali children in the `nepali` dataset. You know that there are multiple measurements per child, so you would like to plot each child in a separate color, to facilitate comparisons between children. Which of the following choices is a good strategy for doing that?

a. Convert the `id` column of `nepali`, which represents the child's id but is currently of a numeric class, to a character, then use `ggplot2` to create a scatterplot and specify `color = "id"` in the `geom_point()` call.
b. Convert the `id` column of `nepali`, which represents the child's id but is currently of a numeric class, to a factor, then use `ggplot2` to create a scatterplot and specify `color = id` in the `aes()` section of the `ggplot()` call.
c. Leave the `id` column of `nepali`, which represents the child's id, as a numeric class, then use `plot` to create a scatterplot, with the option `col = id`.
d. Convert the `id` column of `nepali`, which represents the child's id but is currently of a numeric class, to a factor, then use `plot` to create a scatterplot, and then run `plot(color = id, add = TRUE)`.

Answer b. Explanation: While there is a way to do this with base R graphics rather than ggplot, it's rather complex, and neither the code in Answer c. or Answer d. (which both use the base R graphic function `plot()`) would do it. To plot each child as a separate color, you'd need to enter `id` into the `ggplot` mapping as a vector with the "factor" class. If you do it as a number, ggplot will still try to map id to a color, but it will do it using a sliding scale. See the following two plots for an example.

```
library(ggplot2)
library(gridExtra)
library(faraway)
data(nepali)
nepali <- subset(nepali, id %in% unique(nepali$id)[1:8])
a <- ggplot(nepali, aes(x = ht, y = wt, color = id)) + geom_point() +
        ggtitle("id as numeric")
b <- ggplot(nepali, aes(x = ht, y = wt, color = as.factor(id))) +
        geom_point() +
        ggtitle("id as factor")
grid.arrange(a, b, ncol = 2)
```



6.-9.

Match the `ggplot` functions with the guideline for good graphics each can most obviously help with (match each function with one and only one guideline).

6. `geom_text`
7. `theme_few`
8. `geom_hline`
9. `facet_grid`

a. Highlight interesting data (e.g., label the point with the maximum value of one of the variables)
b. Create small multiples
c. Add references to provide context (e.g., show the average across all data)
d. Increase data-to-ink ratio

Question 10: **Answer a.** If you wanted to label the point with the maximum value, you could use `geom_text()` to label just that point. For example, think of the in-class exercise, where you labeled the point for the player with the most shots with his name.

Question 11: **Answer d.** Changing the theme to the "few" theme will remove some of the background color and grid lines and so help increase the data-to-ink ratio by taking out some non-data ink.

Question 12: **Answer c.** Lines (smooth or otherwise) can be great for providing references in your data. For example, if you plotted date on the x-axis and temperature on the y-axis, a horizontal line showing the mean temperature over the whole year would help point out which days were warmer than the year-round average and which were cooler.

Question 13: **Answer b.** Faceting in ggplot can be used to create small multiples, where a small separate graph is printed for each of the levels of the factor you use to facet.

15.

You want to pull out the vector on tackles from the `worldcup` dataset from the course notes and save it as the object `tackles`. Which of the following commands would **not** achieve this?

a. `tackles <- worldcup[ , "Tackles"]`
b. `tackles <- worldcup$Tackles`
c. `tackles <- worldcup[ , Tackles]`

d. `tackles <- worldcup[1:nrow(worldcup), "Tackles"]`

Answer c. Explanation: Answer c. omits quotation marks on `Tackles` in the indexing. By forgetting those, R will not look for a column named "Tackles", but instead will look for an object *outside of* `worldcup` named `Tackles`, and then will use the values in that to index the dataframe. The code in Answer d. would work because it's choosing all rows in the dataframe (`1:nrow(worldcup)` is indexing every row in the original data), although the faster way to run this line of code would be `tackles <- worldcup[ , "Tackles"]`. Answers a. and b. are both very straightforward ways to index out this column from the dataframe.