QCBS Workshop 3 Thoughts

I recently was an instructor for QCBS R workshop 3, on tidyr, dplyr, and ggplot2. I'm a big proponent of these packages and use them a ton, and I had a few suggestions of ways to improve the 3rd workshop. I thought it might be easier to type them up in a single document, instead of using the google sheets feedback form.

Broader conceptual stuff

The prezi is a little out of date in disucssing the "hadleyverse". Now, these packages are part of the "tidyverse", and more conceptually framed around the tidy data concept of having one observation per row. The weird sort of cult of personality around Hadley Wickham still exists, but is perhaps less prominent. Also, all the main packages used in this workshop (dplyr, tidyr, magrittr, ggplot2) are combined in one package, tidyverse, which includes all of these packages, and a few others. Would be easier to just have the students install and load that, instead of fussing with the individual packages.

I would also not split the tidyr and dplyr sections so clearly, and instead talk about all that stuff together as ways to manipulate data. I'd also add in a little more discussion of the advantages/disadvantages of tidy/long data versus wide data. Which format works best really depends on what analyses/tools will be used on the data later. But, tidyr and dplyr are great for manipulating both tidy and wide data.

On a related note, it may or may not be useful to separate ggplot2 from the rest of the tidyverse instruction. Already, the tidyr and dplyr sections are a lot of material to cover. While ggplot2 integrates really well with tidyr and dplyr for exploratory data analysis, the current prezi doesn't really get into that integration all that much, and is instead a bit more focused on just making nice plots. It might be useful to have a stand-alone "data visualization and making plots for publication" workshop that goes into those concepts in even more detail. I guess it all depends on what students are interested in.

Finally, I think that a lot of the advantages of dplyr/tidyr are most clear in comparison with their equivalent functions in base R. I'm not sure if it's worth it to do exact code comparisons (e.g., comparing filter to which statements and/or subsetting, or comparing summarize to aggregate). But, it might be useful to discuss the advantages in a bit more specific detail, especially for those students who are learning R from the ground up by taking the QCBS R course. That is, I can imagine that students, who were presumably presented with [] and $, and maybe apply and other functions in the earlier workshops, might not see the value in tidyr and dplyr functions that do more or less the same thing. But, particularly for students who are less interested in the programming nitty-gritty and just want to get stuff done with their data, I think dplyr and tidyr have huge advantages, both in ease of coding and understanding, but also technical advantages.

More specific suggestions

1. Get rid of all the grid extra stuff. At least in the way it is taught in this prezi, would be better to just use facets for all the grid extra examples. Might be useful to discuss packages that allow you to place multiple unrelated R plots onto the same graphing space (gridextra, cowplot, probably others) in a standalone workshop on publication-ready figures. But the example given in the prezi is much more suited to faceting.
2. To me, one of the most useful things about dplyr is the ability to use the whole split-apply-combine framework with functions outside of the summarise family of functions in dplyr. Specifically, using the do() function to apply custom functions, or run linear models, or somthing like that. I guess they may not have learned how to write custom functions yet, so that might be an issue. But I think this is a relatively common problem in analyzing biological datasets (e.g., I want to fit the same linear model a bunch of times to different trait measurements). A couple examples are below. One of these requires the "broom"" package, a package which turns model outputs (form lm(), glm(), glmer(), etc) into tidy

data frames. It's great for integrating with dplyr workflows, but would also be a useful package to talk about in some of the modeling classes: makes extracting coefficients and p-values and whatnot from model results much easier.

```
library(tidyverse)
# Example 1- Using a custom function in the split/apply/combine framework
# work with the built-in airquality dataset.
# This is kind of a bad example, as it could be accomplished without
# the custom function in a normal dplyr workflow.
# But, gets across the idea of applying a custom function.
# A better example would use a custom function that doesn't rely on dplyr functions.

# The main restrictions when using custom functions in dplyr is that they must
# (1) Take in a data frame as input
# (2) return a data frame as output
# (3) the output data frames must have the same # and names of columns,
# so they can be combined back together

# Often, we want to center variables so that the mean of a variable is 0.
# Let's write a custom function to do that.
center.variable <- function(tidy.df) {
  results.df <- tidy.df %>%
    mutate(centered.value = value - mean(value, na.rm = T)) %>%
    select(-value)
  return(results.df)
}

# We can apply that function to each subset created with group_by by using the
# do() function. The "." signifies the output from the pipe above.
centered.AQ <- airquality %>%
  gather(key = "measurement", value = "value", Ozone, Solar.R, Wind, Temp) %>%
  group_by(measurement) %>%
  do(center.variable(.)) %>%
  ungroup()
head(centered.AQ)
```

```
## # A tibble: 6 × 4
##   Month   Day measurement centered.value
##   <int> <int>       <chr>          <dbl>
## 1     5     1       Ozone       -1.12931
## 2     5     2       Ozone       -6.12931
## 3     5     3       Ozone      -30.12931
## 4     5     4       Ozone      -24.12931
## 5     5     5       Ozone             NA
## 6     5     6       Ozone      -14.12931
```

```
# and to check if the centering worked...
centered.AQ %>%
  group_by(measurement) %>%
  summarise(mean.centered = mean(centered.value, na.rm = T))
```

```
## # A tibble: 4 × 2
##   measurement mean.centered
##         <chr>         <dbl>
## 1       Ozone -9.830199e-16
## 2     Solar.R -1.949235e-15
```

```
## 3        Temp  6.686123e-15
## 4        Wind -8.720812e-17
```

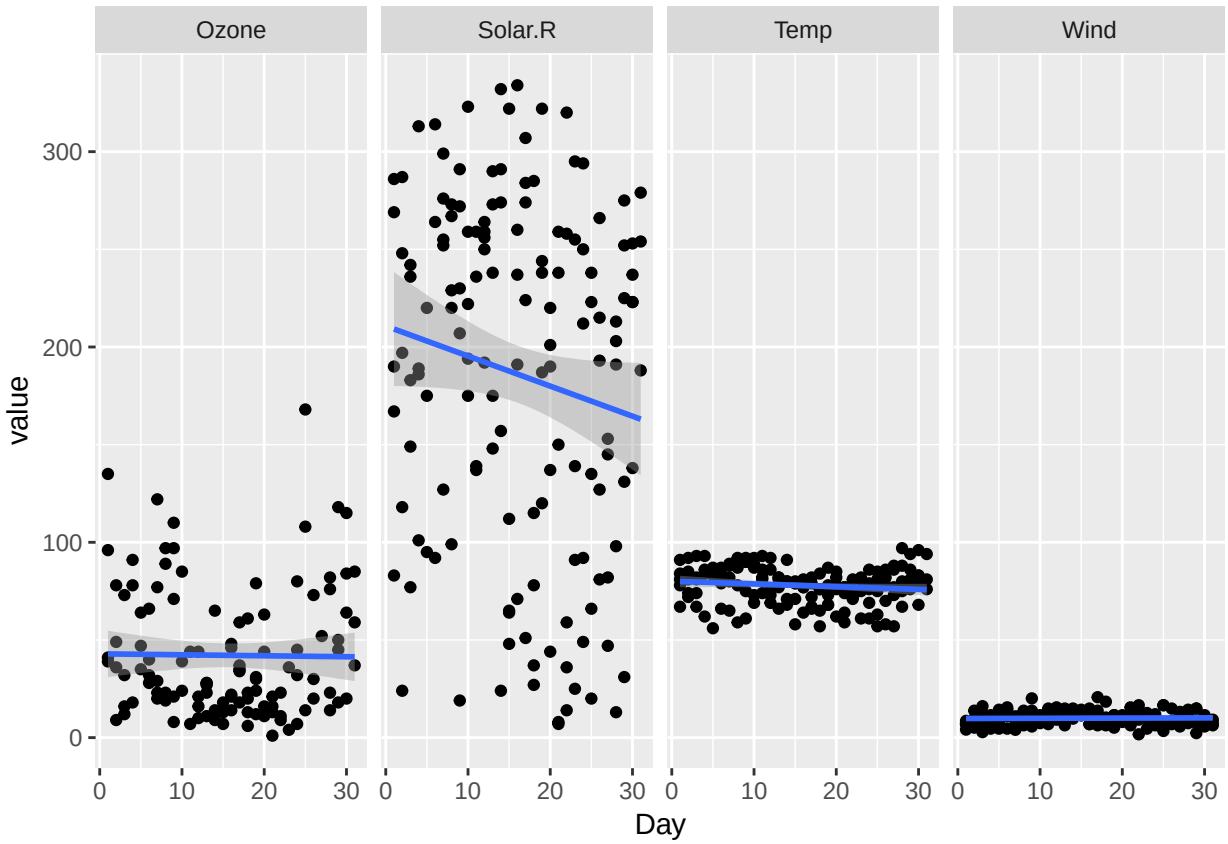
```
# Good, those are 0 (within float).

# Example 2-
# Running a bunch of stats on different subsets of the data.
# need the broom package to get model outputs as a tidy data frame
library(broom)

# Another bad example off the top of my head:
# do the measured variable show some trend throughout the month?
# Probably wouldn't do this in real life, but shows how to simultaneously fit
# linear models for all measured variables
lm.results <- airquality %>%
  gather(key = "measurement", value = "value", Ozone, Solar.R, Wind, Temp) %>%
  group_by(measurement) %>%
  do(tidy(lm(value ~ Day, data = .))) %>%
  ungroup()
lm.results
```

```
## # A tibble: 8 × 6
##   measurement        term     estimate   std.error  statistic      p.value
##         <chr>       <chr>        <dbl>       <dbl>      <dbl>        <dbl>
## 1       Ozone (Intercept)  42.90387169  6.28832252  6.8227849 4.532847e-10
## 2       Ozone         Day  -0.04986077  0.35306258 -0.1412236 8.879425e-01
## 3     Solar.R (Intercept) 210.74191208 15.48196383 13.6120918 1.205374e-27
## 4     Solar.R         Day  -1.53879319  0.84363069 -1.8240128 7.022338e-02
## 5        Temp (Intercept)  80.08610687  1.55983396 51.3427127 1.689319e-97
## 6        Temp         Day  -0.13944349  0.08614974 -1.6186176 1.076164e-01
## 7        Wind (Intercept)   9.78679559  0.58537342 16.7188931 3.549437e-36
## 8        Wind         Day   0.01080243  0.03233022  0.3341280 7.387466e-01
```

```
# And, just to show the integration with ggplot
lm.plot <- airquality %>%
  gather(key = "measurement", value = "value", Ozone, Solar.R, Wind, Temp) %>%
  ggplot(data = ., aes(x = Day, y = value)) +
  geom_point() +
  geom_smooth(method = "lm") +
  facet_grid(. ~ measurement)
lm.plot
```




```
# Nice thing is that the lm smoother in ggplot is the base R one, so the slopes and intercepts
# in the lm.results table are the same as the lines in the lm.plot.
```

3. Need to discuss the ungroup() function in dplyr. If you use group_by to create a new object with dplyr, the grouping persists in the newly created object. This can lead to unexpected behavior down the line. I usually just use ungroup() at the end of any dplyr blcok that used group_by.

4. RStudio cheatsheets- There are excellent cheatsheets for ggplot2 and data maniupulation with tidyr/dplyr available through the RStudio help menu (Help > Cheatsheets). Extremely useful, especially for showing the diversity of geoms available in ggplot2.