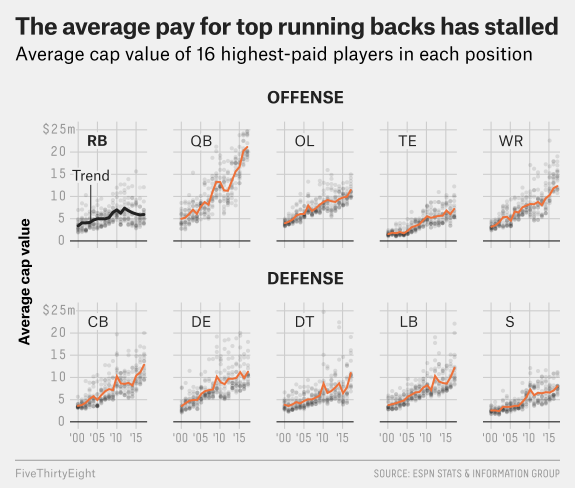
R Notebook for Tidy Tuesday

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook for the [#TidyTuesday](https://github.com/rfordatascience/tidytuesday/blob/master/README.md) challenge: a weekly social data project (in R), which builds off #makeovermonday style projects but aimed at the R ecosystem. An emphasis will be placed on understanding how to summarize and arrange data to make meaningful charts with ggplot2, tidyr, dplyr, and other tools in the tidyverse ecosystem.

This code is for Week 2, and my effort to recreate



this image, from 538

library(tidyverse, quietly = TRUE, warn.conflicts = FALSE)

## ── Attaching packages ────────────────────────────────────────────────────────────────────────── tidyverse 1.2.1 ──

## ✔ ggplot2 2.2.1 ✔ purrr 0.2.4  
## ✔ tibble 1.4.2 ✔ dplyr 0.7.4  
## ✔ tidyr 0.8.0 ✔ stringr 1.3.0  
## ✔ readr 1.1.1 ✔ forcats 0.3.0

## ── Conflicts ───────────────────────────────────────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(readxl) # To read in Excel data the tidyverse way

# import data

football <- read\_excel("data/tidy\_tuesday\_week2.xlsx")  
football

## # A tibble: 800 x 11  
## year Cornerback `Defensive Lineman` Linebacker `Offensive Lineman`  
## <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 2011. 11265916. 17818000. 16420000. 15960000.  
## 2 2011. 11000000. 16200000. 15623000. 12800000.  
## 3 2011. 10000000. 12476000. 11825000. 11767500.  
## 4 2011. 10000000. 11904706. 10083333. 10358200.  
## 5 2011. 10000000. 11762782. 10020000. 10000000.  
## 6 2011. 9244117. 11340000. 8150000. 9859166.  
## 7 2011. 8000000. 10000000. 7812500. 9500000.  
## 8 2011. 7900000. 9482500. 7700000. 9420000.  
## 9 2011. 7400000. 8450000. 7200000. 8880000.  
## 10 2011. 7000000. 8383266. 7100000. 8686750.  
## # ... with 790 more rows, and 6 more variables: Quarterback <dbl>,  
## # `Running Back` <dbl>, Safety <dbl>, `Special Teamer` <dbl>, `Tight  
## # End` <dbl>, `Wide Receiver` <dbl>

# tidydata, and match names with labels

football\_adj <- football %>%   
 rename(QB = Quarterback, RB = `Running Back`, S = Safety, ST = `Special Teamer`, TE = `Tight End`, WR = `Wide Receiver`, CB = Cornerback, DL = `Defensive Lineman`, LB = Linebacker, OL = `Offensive Lineman`)

# mutate data to divide by 1M

football\_adj <- football\_adj %>%   
 mutate\_at(vars(CB:WR), funs(. / 1000000))

# select top 16 salares of each year, for each position

football\_16 <- football\_adj %>%   
 group\_by(year) %>%   
 top\_n(16) %>%   
 gather(key = position, value = salary, -year)

## Selecting by WR

football\_16

## # A tibble: 1,290 x 3  
## # Groups: year [8]  
## year position salary  
## <dbl> <chr> <dbl>  
## 1 2011. CB 11.3   
## 2 2011. CB 11.0   
## 3 2011. CB 10.0   
## 4 2011. CB 10.0   
## 5 2011. CB 10.0   
## 6 2011. CB 9.24  
## 7 2011. CB 8.00  
## 8 2011. CB 7.90  
## 9 2011. CB 7.40  
## 10 2011. CB 7.00  
## # ... with 1,280 more rows

# find means for line graph

Here is a code block in that I ultimately didn’t, but thought I would. The image looked like a smooth fit mean line would ‘fit’ each of the years, so made an ‘avg’ df that I anticipated using. When I was bulidng the ggplot code block (below), discovered that ggplot would do this for me. Thanks ggplot!

football\_16\_avg <- football\_adj %>%   
 group\_by(year) %>%   
 top\_n(16) %>%   
 summarise\_each(funs(mean))

## Selecting by WR

## `summarise\_each()` is deprecated.  
## Use `summarise\_all()`, `summarise\_at()` or `summarise\_if()` instead.  
## To map `funs` over all variables, use `summarise\_all()`

football\_16\_avg

## # A tibble: 8 x 11  
## year CB DL LB OL QB RB S ST TE WR  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 2011. 8.12 9.99 8.82 9.68 11.7 6.47 6.25 2.90 5.44 8.68  
## 2 2012. 8.62 10.5 10.9 9.36 11.2 7.44 6.67 3.32 5.50 9.36  
## 3 2013. 8.55 10.4 9.68 8.82 13.4 6.73 6.41 3.23 5.60 8.13  
## 4 2014. 8.21 12.1 9.01 9.42 15.6 6.42 6.72 3.33 5.60 9.57  
## 5 2015. 10.3 11.3 8.59 9.66 16.2 5.74 6.49 3.31 6.36 10.1   
## 6 2016. 11.1 11.0 10.2 9.95 20.0 5.85 7.07 3.56 6.03 11.9   
## 7 2017. 11.5 13.0 12.2 11.7 19.9 5.99 7.38 3.78 7.29 12.7   
## 8 2018. 12.3 16.0 12.5 12.5 23.8 6.38 9.03 3.82 7.65 13.2

# need to get a mean for each year and each position (probably put in new df so you can add it as a ggplot element)

Wasn’t sure how to do this, so skipped it; but again, didn’t need to.

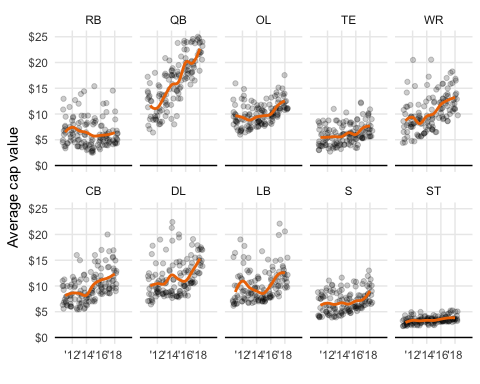
# plot

football\_16$position\_f = factor(football\_16$position, levels=c('RB', 'QB', 'OL', 'TE', 'WR', 'CB', 'DL', 'LB', 'S', 'ST'))  
  
ggplot(football\_16) +  
 geom\_jitter(aes(x = year, y = salary), alpha = 0.2) +  
 geom\_smooth(aes(x = year, y = salary), color = "darkorange2", method = "auto", span = 0.4, se = FALSE) +  
 #coord\_cartesian(ylim = c(0,25), xlim = c(2010,2020)) +  
 scale\_y\_continuous(limits = c(0,25), breaks=c(0,5,10,15,20,25), minor\_breaks = NULL, labels = scales::dollar) +  
 scale\_x\_continuous(limits = c(2010,2020), breaks = c(2012,2014,2016,2018), minor\_breaks = NULL, label = c("'12", "'14", "'16", "'18")) +  
 geom\_hline(aes(yintercept = 0)) +  
 facet\_wrap(~ position\_f, ncol = 5) +  
 theme\_minimal() +  
 labs(y = "Average cap value", x = "")

## `geom\_smooth()` using method = 'loess'

## Warning: Removed 3 rows containing non-finite values (stat\_smooth).

## Warning: Removed 3 rows containing missing values (geom\_point).



# save

ggsave("nfl\_salary.png", last\_plot(), height = 8, width = 12, units = "in", dpi = 600)

## `geom\_smooth()` using method = 'loess'

## Warning: Removed 3 rows containing non-finite values (stat\_smooth).

## Warning: Removed 3 rows containing missing values (geom\_point).

#default is last plot, but can name objects here

# Conclusion

First step was to make an outline of anticipated steps to go from raw data to final image; I made these key headers of each section, and then began to work my way through. Ultimately I didn’t need two of the anticipated steps. Here are some of the new things that I needed to learn/figure out for the final image:

* ggplot
* factor
* top\_n (to select top #)
* mutate\_at (to make a new df)
* gather(key = position, value = salary, -year)
* [ggsave](http://ggplot2.tidyverse.org/reference/ggsave.html)

This was the closest that I’ve gotten to making the image look even remotely close; so needless to say, I’m hooked.