In [1]:
```
library(tidyverse)
library(nycflights13)
library(lubridate)
```

```
── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──
✔ ggplot2 2.2.1     ✔ purrr   0.2.4
✔ tibble  1.3.4     ✔ dplyr   0.7.4
✔ tidyr   0.7.2     ✔ stringr 1.2.0
✔ readr   1.1.1     ✔ forcats 0.2.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()

Attaching package: ‘lubridate’

The following object is masked from ‘package:base’:

    date
```

STATS 306

Problem set 5: importing and tidying data

Each problem is worth two to four points, depending on difficulty, for a total of 20.

Note: you do not need to use `install.packages()` in this notebook. You may assume that we have already installed all of the necessary packages when we run your code.

Problem 1 (2 pts)

The following table lists annual pizza consumption (in kilograms) for the teaching staff of STATS 306:

person	2015	2016	2017
Kidus	0	8	2
Byoung	2	.	3
Luke	1	3	6
Prof. Terhorst	25	70	372

A `.` in this table denotes a missing value.

Use the `tidyverse` commands to import this table and store it as `table1`.

In [2]:
```
table1 = NA
### BEGIN SOLUTION
table1 = tribble(
    ~person, ~`2015`, ~`2016`, ~`2017`,
    "Kidus", 0, 8, 2,
    "Byoung", 2, NA, 3,
    "Luke", 1, 3, 6,
    "Prof. Terhorst", 25, 70, 372
)
### END SOLUTION
```

In [3]:
```
stopifnot(exists("table1"))
### BEGIN HIDDEN TESTS
table1_ans = tribble(
    ~person, ~`2015`, ~`2016`, ~`2017`,
    "Kidus", 0, 8, 2,
    "Byoung", 2, NA, 3,
    "Luke", 1, 3, 6,
    "Prof. Terhorst", 25, 70, 372
)
stopifnot(identical(table1$person, table1_ans$person))
for (v in 2015:2017) {
    c1 = table1[[as.character(v)]]
    c2 = table1_ans[[as.character(v)]]
    stopifnot(all(near(c1[!is.na(c1)], c2[!is.na(c2)])))
}
### END HIDDEN TESTS
```

Problem 2 (2 pts)

Convert `table1` to tidy form and store the result as `table2`.

In [4]:
```
table2 = NA
### BEGIN SOLUTION
table2 = gather(table1, 2:4, key="year", value="apc")
### END SOLUTION
```

In [5]:
```
stopifnot(exists("table2"))
### BEGIN HIDDEN TESTS
table2_ans = gather(table1, 2:4, key="year", value="apc") %>% arrange(person, year)
atable2 = arrange(table2, person, year)
stopifnot(all(atable2$person == table2_ans$person))
stopifnot(all(atable2$year == table2_ans$year))
stopifnot(all(near(select(atable2, -person, -year)[[1]], table2_ans$apc), na.rm=T))
### END HIDDEN TESTS
```

Problem 3 (2 pts)

Define a function `f3(v)` which uses the tidyverse commands to parse strings of European price data. For example,

```
{r}
> f3("€1.220,36 €3.002,18")
[1] 1220.36 3002.18
```

The outline of the function has been provided for you.

In [6]:
```
f3 = function(v) {
    ### BEGIN SOLUTION
    parse_number(str_split(v, " ", simplify = T),
                 locale=locale(grouping_mark = "."))
    ### END SOLUTION
}
```

In [7]:
```
stopifnot(exists("f3"))
stopifnot(near(f3("€4.112,86 €2,30"), c(4112.86, 2.30)))
### BEGIN HIDDEN TESTS
v = "€1.220,36 €3.002,18 €582,30 €800,64 €1.113,99 €100,00 0,00"
stopifnot(near(
    f3(v),
    c(1220.36, 3002.18, 582.30, 800.64, 1113.99, 100.00, 0.)
))
### END HIDDEN TESTS
```

Problem 4 (4 pts)

Define a function `f4(x,y)` which takes two arguments, `x` and `y`. Each argument is a comma-separated string whose first entry is a column name and whose remaining entries are column values. The function `f4` should return a tibble consisting of these two columns and their respective entries. For example:

```
{r}
> x = "col1,1,2,3,4"
> y = "col2,a,b,c,d"
> f4(x,y) %>% print
# A tibble: 4 x 2
   col1  col2
  <int> <chr>
1     1     a
2     2     b
3     3     c
4     4     d
```

In [8]:
```
f4 = function(x, y) {
### BEGIN SOLUTION
    write_csv(as_tibble(t(read_csv(paste(x,y,sep="\n"), col_names=F))),
              col_names=F, path="tmp.csv")
    read_csv("tmp.csv")
### END SOLUTION
}
```

In [9]:
```
x = "u,1.23,2.0,3.0,4.0"
y = "1col,2015-06-07,2014-01-01,2013-02-04,2014-01-01"
stopifnot(near(
    f4(x,y)$u, c(1.23,2.0,3.0,4.0)
    )
)
stopifnot(identical(
    f4(x,y)$`1col`,
    ymd(c("2015-06-07","2014-01-01","2013-02-04","2014-01-01"))
    )
)
stopifnot(near(
    f4(y,x)$u, c(1.23,2.0,3.0,4.0)
    )
)
stopifnot(identical(
    f4(y,x)$`1col`,
        ymd(c("2015-06-07","2014-01-01","2013-02-04","2014-01-01"))
    )
)
### BEGIN HIDDEN TESTS
x = "col1,a,b,c,d,'hello'"
y = "1col,-1,2,3,4,4"
stopifnot(identical(
    f4(x,y)$col1,
    c('a','b','c','d',"'hello'")
    )
)
stopifnot(near(
    f4(x,y)$`1col`,
    c(-1,2,3,4,4)
    )
)
### END HIDDEN TESTS
```

```
Parsed with column specification:
cols(
  u = col_double(),
  `1col` = col_date(format = "")
)
Parsed with column specification:
cols(
  u = col_double(),
  `1col` = col_date(format = "")
)
Parsed with column specification:
cols(
  `1col` = col_date(format = ""),
  u = col_double()
)
Parsed with column specification:
cols(
  `1col` = col_date(format = ""),
  u = col_double()
)
Parsed with column specification:
cols(
  col1 = col_character(),
  `1col` = col_integer()
)
Parsed with column specification:
cols(
  col1 = col_character(),
  `1col` = col_integer()
)
```

Problem 5 (4 pts)

A CSV file called `problem5.csv` has been distributed along with this notebook. Use the tidyverse commands to import it and store it in a table called `table5`. Be sure that the column types are correct, missing data is correctly handled, and any comments or other metadata are dealt with appropriately.

In [10]:
```
table5 = NA
### BEGIN SOLUTION
table5 = read_csv("problem5.csv", comment = "--", na="??", guess_max=2000)
### END SOLUTION
```

```
Parsed with column specification:
cols(
  `1` = col_double(),
  b = col_character(),
  c = col_date(format = "")
)
```

In [11]:
```
stopifnot(exists("table5"))
### BEGIN HIDDEN TESTS
table5_ans = read_csv("problem5.csv", comment = "--", na="??", guess_max=2000)
stopifnot(identical(table5, table5_ans))
### END HIDDEN TESTS
```

```
Parsed with column specification:
cols(
  `1` = col_double(),
  b = col_character(),
  c = col_date(format = "")
)
```

Problem 6

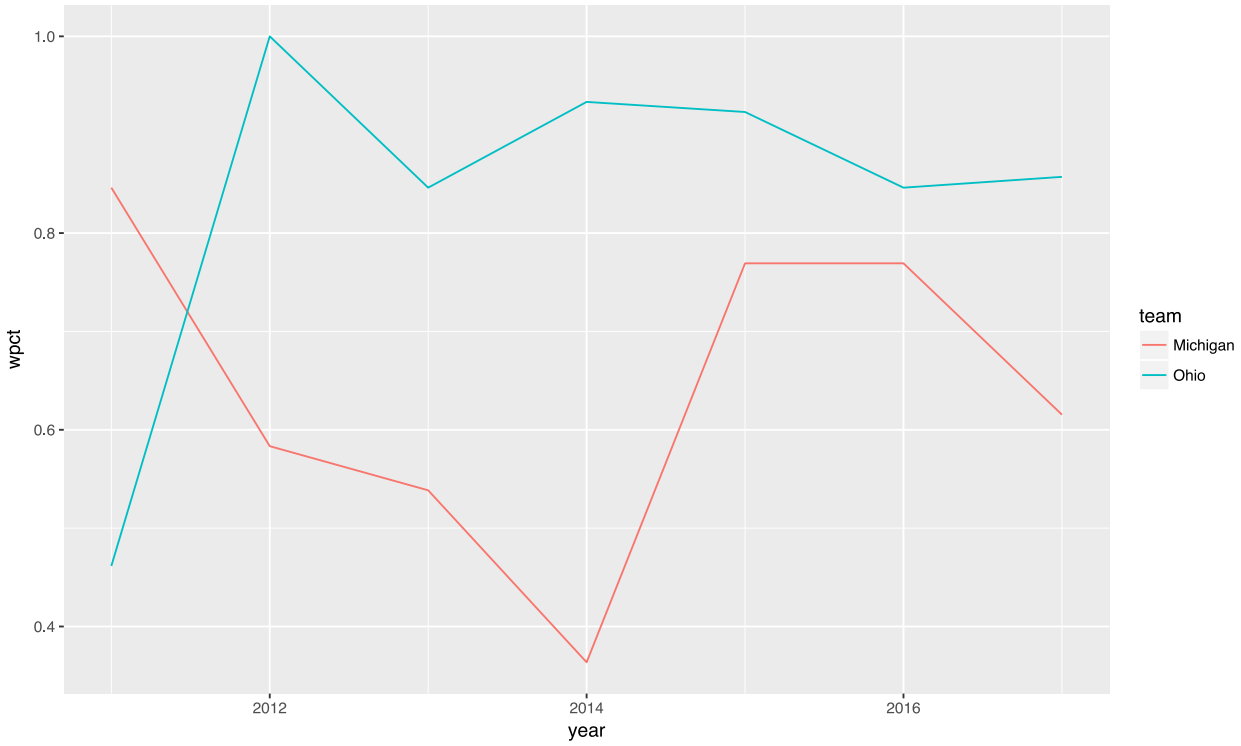
In PS4 you used the following data set to generate a plot:

```
  year mich_wpct oh_wpct
1 2011 0.8461538 0.4615385
2 2012 0.5833333 1.0000000
3 2013 0.5384615 0.8461538
4 2014 0.3636364 0.9333333
5 2015 0.7692308 0.9230769
6 2016 0.7692308 0.8461538
7 2017 0.6153846 0.8571429
```

Since these data are not tidy, you likely encountered difficulties using `ggplot()` to create the requested plot. Now that you know how to tidy up data, re-create the plot more easily using `read_table()` and `gather()`.

In [12]:
```
### BEGIN SOLUTION
mich_ohio = read_table("
year mich_wpct oh_wpct
2011 0.8461538 0.4615385
2012 0.5833333 1.0000000
2013 0.5384615 0.8461538
2014 0.3636364 0.9333333
2015 0.7692308 0.9230769
2016 0.7692308 0.8461538
2017 0.6153846 0.8571429", skip=1) %>% rename(Michigan=mich_wpct, Ohio=oh_wpct) %>%
                                        gather(2:3, key="team", value="wpct")
ggplot(mich_ohio) + geom_line(aes(x=year, y=wpct, colour=team))

### END SOLUTION
```



Problem 7 (2 points)

In this problem you will again analyze the `cfb` data set.

In [13]:
```
load('cfb.RData')
```

For every pair of teams that played seven or more times in `cfb`, add up the total number of points scored across all games. Store the resulting data set in a table called `table7`. The table should have three columns: `team_pair`, `total_points`, and `points_per_game`. `team_pair` should contain the two team names in alphabetical order, separated by a hyphen; for example, "Michigan-Ohio State". Note that there should be *one* row per team pair, for a total of 223 rows.

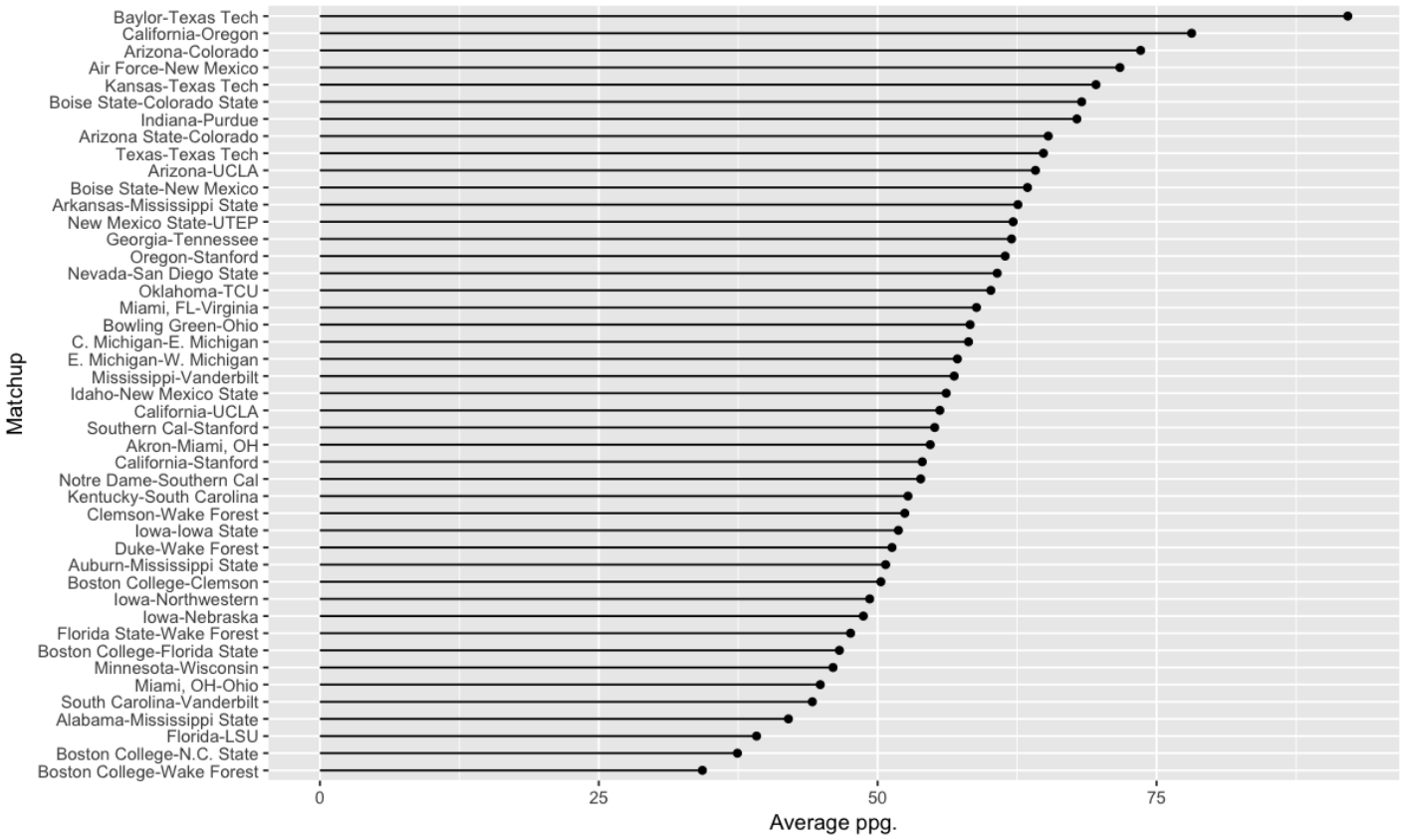
In [14]:
```
table7 = NA
### BEGIN SOLUTION
table7 = mutate(cfb, team1=pmin(winning, losing),
                team2=pmax(winning, losing),
                total_points=winning_points + losing_points) %>%
        unite(team_pair, team1, team2, sep = "-") %>%
        group_by(team_pair) %>%
        summarize(n=n(), total_points=sum(total_points)) %>%
        mutate(points_per_game=total_points/n) %>%
        filter(n>=7)
### END SOLUTION
```

In [15]:
```
stopifnot(exists("table7"))
stopifnot(nrow(table7) == 223)
### BEGIN HIDDEN TESTS
table7_ans = mutate(cfb, team1=pmin(winning, losing),
                    team2=pmax(winning, losing),
                    total_points=winning_points + losing_points) %>%
        unite(team_pair, team1, team2, sep = "-") %>%
        group_by(team_pair) %>%
        summarize(n=n(), total_points=sum(total_points)) %>%
        mutate(points_per_game=total_points/n) %>%
        filter(n>6)

stopifnot(identical(
    arrange(table7_ans, team_pair)$team_pair,
    arrange(table7, team_pair)$team_pair
    )
)
### END HIDDEN TESTS
```

Problem 8 (2 points)

Sort `table7` in descending order of `points_per_game` and sample every fifth row to create the following plot:



In [16]:
```
### BEGIN SOLUTION
arrange(table7, desc(points_per_game)) %>% slice(seq(1, nrow(table7), 5)) %>%
    mutate(team_pair=reorder(team_pair, points_per_game))  %>%
    ggplot + geom_point(aes(y=team_pair, x=points_per_game)) +
    geom_segment(aes(y=team_pair, yend=team_pair, x=0, xend=points_per_game)) +
    ylab("Matchup") + xlab("Average ppg.")
### END SOLUTION
```