Hitchhiker's Guide to the Tidyverse

Cory Lanker

2019-07-22

Contents

Introduction

Introducing the tidyverse with two data sets:

1. Basic plots with `tibble` and `ggplot2` using `Boston` house prices.
2. Preprocessing with `tidyr` and `dplyr` using `Lahman` baseball data.

Though some of these commands will be used, we won't go deeply into the following tidyverse packages:

1. Reading in data with `readr`.
2. String manipulation with `stringr`.
3. Dates and times with `lubridate`.
4. Handling factors with `forcats`.

R proficiency is assumed. These notes aim to bring a functional R coder into the tidyverse realm.

```r
# To install the necessary packages in the tidyverse:
install.packages("tidyverse", dependencies = TRUE)
```

This document is built with R Markdown, **knitr** (Xie, 2015), and the **bookdown** package (Xie, 2019).

Chapter 1

tibbles, ggplot2, and the *tidyverse*

The tidyverse universe includes:

In general, the tidyverse is the following:

1. provided the `pipe` command `%>%`

- `x %>% f(y, z, ...)` is `f(x, y, z, ...)`
- allows chained commands for better coherence
- e.g., `mtcars %>% apply(2, mean)` is error without `tidyr::%>%`

2. `tibble` is the improved data structure of the tidyverse

- easier to read-in data to a useful format
- automatic type conversion
- nicer printing options

3. `dplyr` provides tibble manipulation commands

- understandable data processing with `pipe` streams
- **filter** data faster
- **arrange** rows of data easily
- **select** columns quickly
- **mutate** variables
- **summarize** according to `group_by()`
- also provides SQL relational operations

4. `ggplot2` is a plotting syntax (grammar of graphics)

- `qplot()` provides a sensible **q**uick **plot**
- apply plot types to data rather than the reverse

- e.g. `ggplot(data) + plot_type(aes(xvar, yvar, groups), options)`
- allows grid of plots by group using **facets**
- overlays statistical summaries, e.g. `+ geom_smooth(x, y)`
- "add" options such as transformed axes, labels, coordinates, etc.

5. `readr` is a faster, less painful read-in method

- `read_fun` denotes `readr` functions (instead of `read.fun`)
- guesses column types
- offers writing functions, too
- allows read and write with RDS, R's binary format

6. `tidyr` recharacterizes tibbles

- `spread()` turns key and value columns into key-category columns
- e.g., `state, year, pop` into `state, 1990, 1991, ...` of pop values
- `gather()` turns expands data frames by condensing columns
- e.g., condenses `1990, 1991, ...` into two `year, pop` columns

7. Other helpful tidyverse packages:

- `stringr` offers many useful `str_fun` operations
- `forcats` has operations _for cat_egorical variables
- `lubridate` provides date and time control
- `purrr`

The examples I'll use is the Boston housing database and the Lahman baseball database. By doing analysis on these two data sets, I hope to introduce the power of the tidyverse.

1.1 Tibbles: Boston housing data

Load, convert, print a tibble.

```
# Convert to a tibble so it prints nicely
library(MASS)
select <- dplyr::select
boston <- as_tibble(MASS::Boston)
boston
```

```
## # A tibble: 506 x 14
##       crim    zn indus  chas   nox    rm   age   dis   rad   tax ptratio
##      <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl> <int> <dbl>   <dbl>
##  1 0.00632  18    2.31     0 0.538  6.58  65.2  4.09     1   296    15.3
##  2 0.0273    0    7.07     0 0.469  6.42  78.9  4.97     2   242    17.8
##  3 0.0273    0    7.07     0 0.469  7.18  61.1  4.97     2   242    17.8
##  4 0.0324    0    2.18     0 0.458  7.00  45.8  6.06     3   222    18.7
```

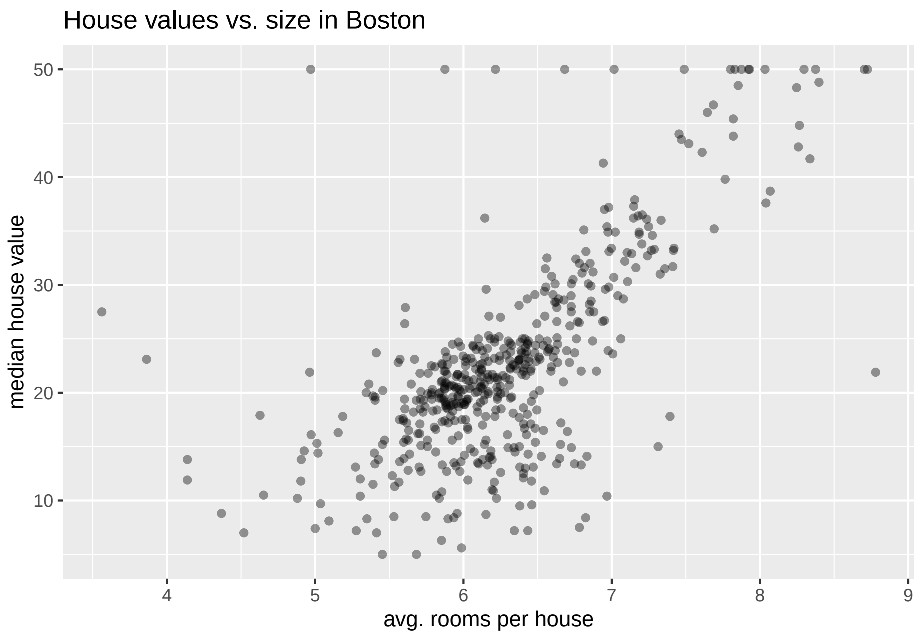
```
##  5 0.0690    0    2.18     0 0.458  7.15  54.2  6.06     3   222    18.7
##  6 0.0298    0    2.18     0 0.458  6.43  58.7  6.06     3   222    18.7
##  7 0.0883   12.5  7.87     0 0.524  6.01  66.6  5.56     5   311    15.2
##  8 0.145    12.5  7.87     0 0.524  6.17  96.1  5.95     5   311    15.2
##  9 0.211    12.5  7.87     0 0.524  5.63 100    6.08     5   311    15.2
## 10 0.170    12.5  7.87     0 0.524  6.00  85.9  6.59     5   311    15.2
## # ... with 496 more rows, and 3 more variables: black <dbl>, lstat <dbl>,
## #   medv <dbl>
```

```
?MASS::Boston
```

- crim per capita crime rate by town.
- zn proportion of residential land zoned for lots over 25,000 sq.ft.
- indus proportion of non-retail business acres per town.
- chas Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- nox nitrogen oxides concentration (parts per 10 million).
- rm average number of rooms per dwelling.
- age proportion of owner-occupied units built prior to 1940.
- dis weighted mean of distances to five Boston employment centres.
- rad index of accessibility to radial highways.
- tax full-value property-tax rate per $10,000.
- ptratio pupil-teacher ratio by town.
- black $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town.
- lstat lower status of the population (percent).
- medv median value of owner-occupied homes in $1000s.

A ggplot is the first declaration (usually variable `data` is defined), followed by graphics definitions (operations on the data):

```
ggplot(data = boston) +
  geom_point(mapping = aes(x = rm, y = medv), alpha=0.4) +
  labs(x = "avg. rooms per house",
       y = "median house value",
       title = "House values vs. size in Boston")
```



Making a histogram of all numeric variables. First step, gather all variables.

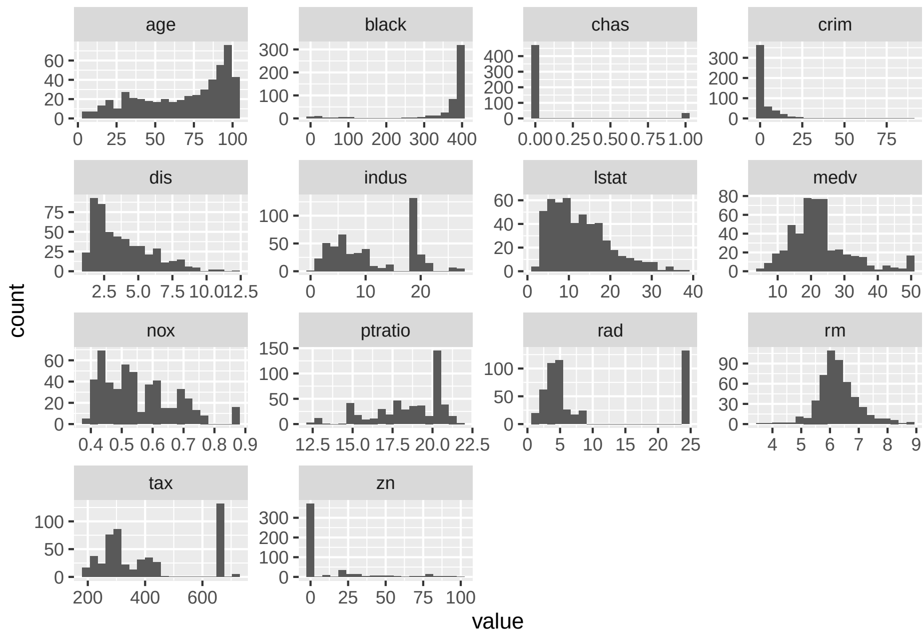
```
boston %>%
  keep(is.numeric) %>%  # strips all non-numeric columns (unnecessary here)
  gather()  # puts all variable values in a single column 'value'
```

```
## # A tibble: 7,084 x 2
##    key      value
##    <chr>    <dbl>
##  1 crim   0.00632
##  2 crim   0.0273
##  3 crim   0.0273
##  4 crim   0.0324
##  5 crim   0.0690
##  6 crim   0.0298
##  7 crim   0.0883
##  8 crim   0.145
##  9 crim   0.211
## 10 crim   0.170
## # ... with 7,074 more rows
```

Facet wrap allows plotting each `key` level separately.

```
boston %>%
  gather() %>%
  ggplot() +
    facet_wrap(~ key, scales = "free") +
```

```r
  geom_histogram(mapping = aes(value), bins=20)
```



From the histograms, there seems to be only a few values of `crim` over 30.

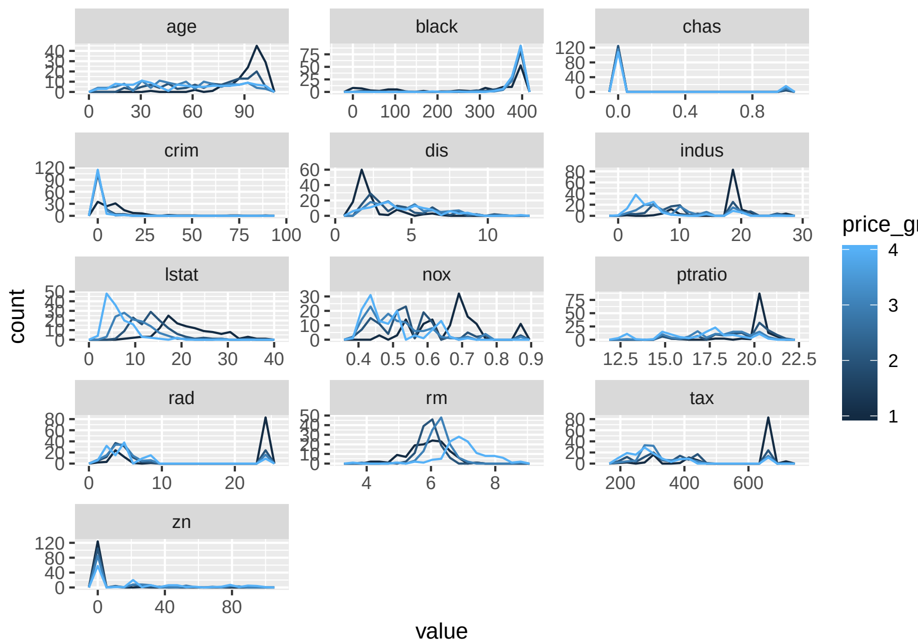
```r
boston %>%
  filter(crim > 30)
```

```
## # A tibble: 8 x 14
##     crim    zn indus  chas   nox    rm   age   dis   rad   tax ptratio black
##    <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl> <int> <dbl>   <dbl> <dbl>
## 1  89.0      0  18.1     0 0.671  6.97  91.9  1.42    24   666    20.2 397.
## 2  38.4      0  18.1     0 0.693  5.45 100    1.49    24   666    20.2 397.
## 3  41.5      0  18.1     0 0.693  5.53  85.4  1.61    24   666    20.2 329.
## 4  67.9      0  18.1     0 0.693  5.68 100    1.43    24   666    20.2 385.
## 5  51.1      0  18.1     0 0.597  5.76 100    1.41    24   666    20.2   2.6
## 6  45.7      0  18.1     0 0.693  4.52 100    1.66    24   666    20.2  88.3
## 7  73.5      0  18.1     0 0.679  5.96 100    1.80    24   666    20.2  16.4
## 8  37.7      0  18.1     0 0.679  6.20  78.7  1.86    24   666    20.2  18.8
## # ... with 2 more variables: lstat <dbl>, medv <dbl>
```

1.2 ggplot2 and EDA

But we want to know the conditional distributions according to `medv`. First, showing this with conditional densities.

```
boston %>%
  gather('key', 'value', -one_of('medv')) %>%
  mutate(price_gr = ntile(medv, 4)) %>%
  ggplot(aes(value, color = price_gr, group = price_gr)) +
    facet_wrap(~ key, ncol = 3, scales = "free") +
    geom_freqpoly(bins = 20)
```




```
# Click on the expand icon at the top right to make bigger.
```

Appears chas is categorical.

```
boston <- boston %>%
  mutate(chas = factor(chas))
```

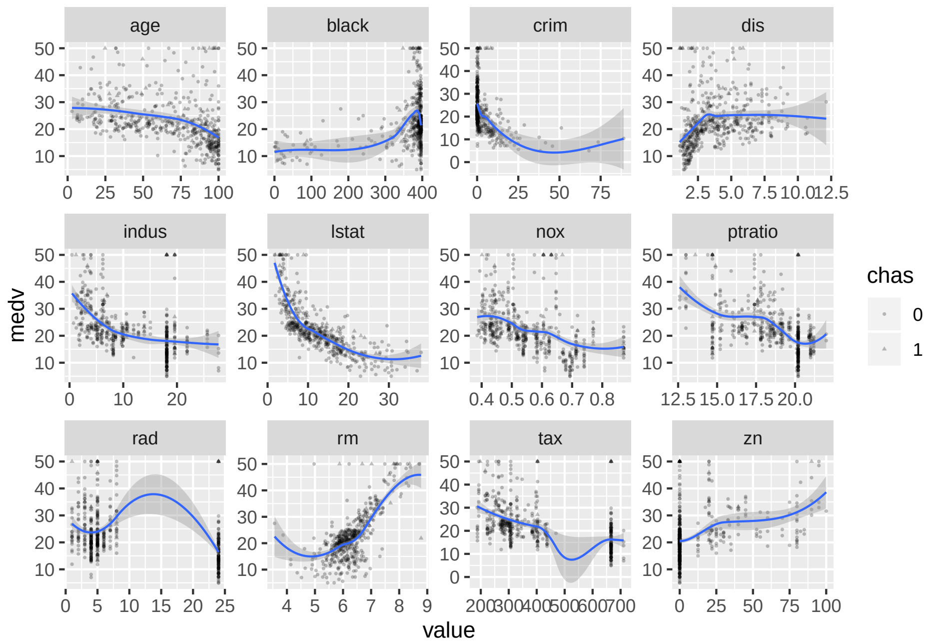
Second, scatterplots of median value vs. all variables.

```
boston %>%
  gather('key', 'value', -one_of(c("medv", "chas"))) %>%
  ggplot(aes(x = value, y = medv)) +
    facet_wrap( ~ key, scales = "free") +
    geom_point(aes(shape = chas), size = 0.5, alpha = 0.25) +
    geom_smooth(lwd = 0.5, se = TRUE) +
  ggsave('plots/medv-scatter.pdf')
```

```
## Saving 6.5 x 4.5 in image
```

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

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

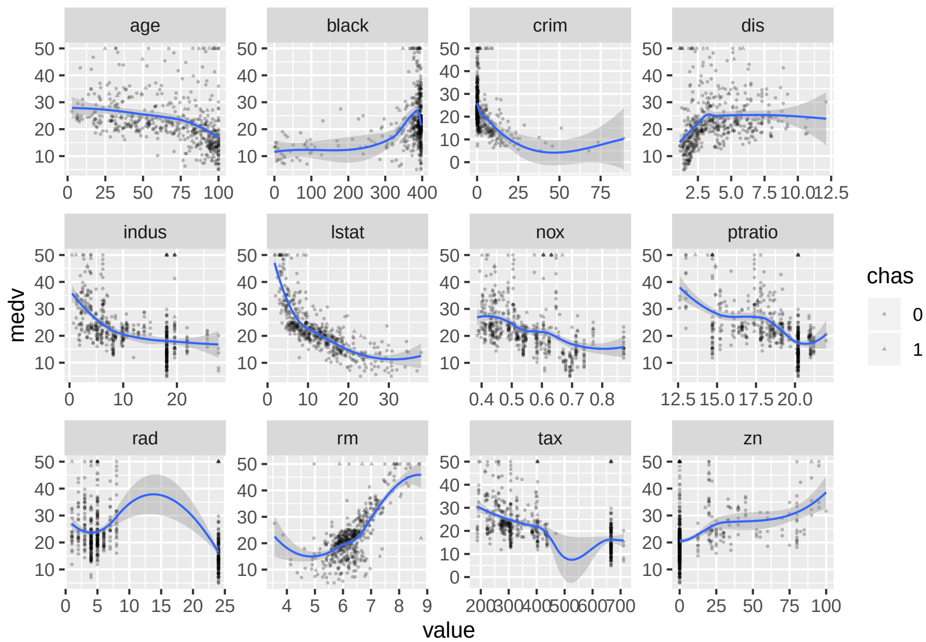



```
# Click on the expand icon at the top right to make bigger.
```

There are ggplot jitter options, but none worked for me.

```
boston %>%
  gather('key', 'value', -one_of(c("medv", "chas"))) %>%
  ggplot(aes(x = value, y = medv)) +
    facet_wrap( ~ key, scales = "free") +
    geom_jitter(aes(shape = chas), size = 0.5, alpha = 0.25) +
    geom_smooth(lwd = 0.5, se = TRUE)
```

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

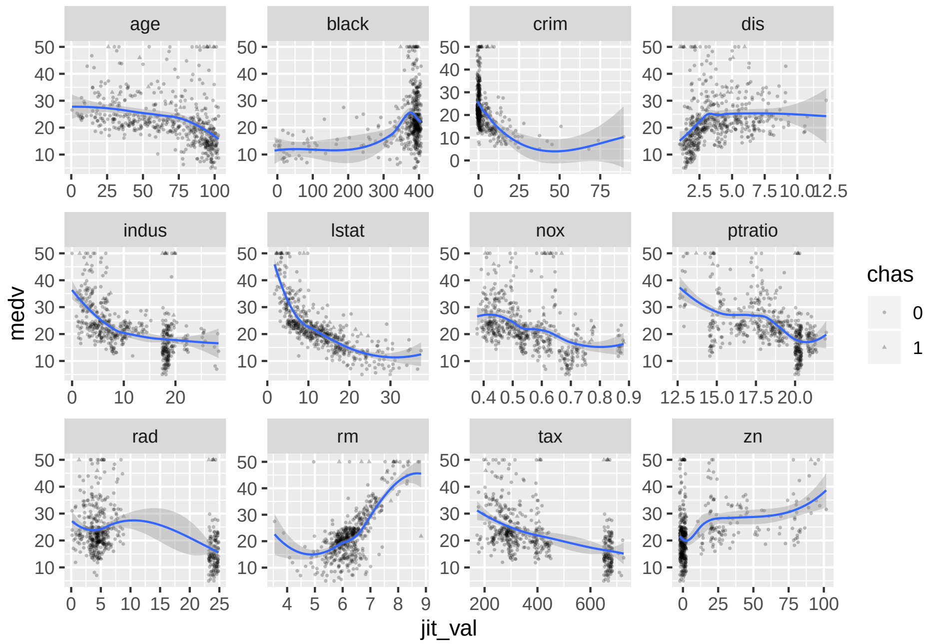


Tinkering to get a jittered plot.

```
var_sd <- boston %>%
  gather('key', 'value', -one_of(c("medv", "chas"))) %>%
  group_by(key) %>%
  summarize(var_sd = sd(value))
#var_sd <- boston %>%
#  keep(is.numeric) %>%
#  summarize_all(sd)
boston %>%
  gather('key', 'value', -one_of(c("medv", "chas"))) %>%
  left_join(y = var_sd, by = "key") %>%
  mutate(jit_val = value + var_sd * runif(nrow(boston), -0.1, 0.1)) %>%
  ggplot(aes(x = jit_val, y = medv)) +
    facet_wrap( ~ key, scales = "free") +
    geom_jitter(aes(shape = chas), size = 0.5, alpha = 0.25) +
    geom_smooth(lwd = 0.5, se = TRUE) +
  ggsave('plots/medv-jitter.pdf')
```

```
## Saving 6.5 x 4.5 in image
```

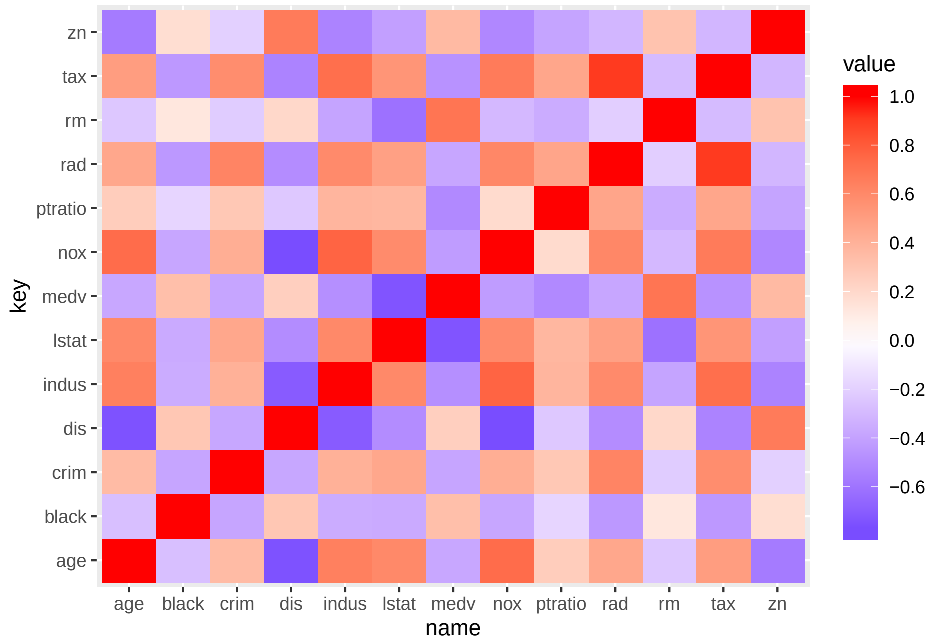
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# Click on the expand icon at the top right to make bigger.
```

Covariance plot of variables.

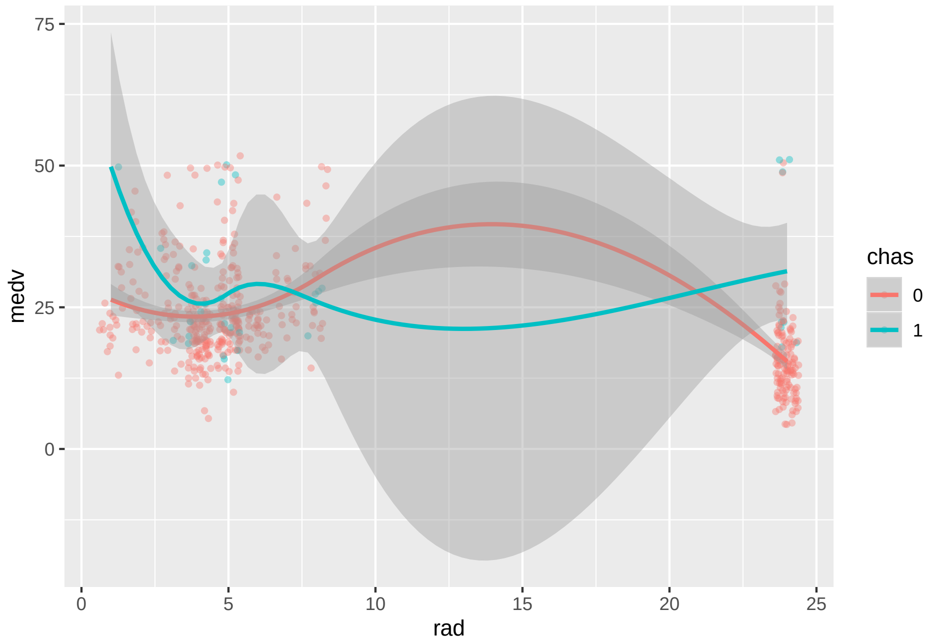
```
boston_num <- boston %>%
  keep(is.numeric)
boston_num %>%
  cor() %>%
  as_tibble() %>%
  mutate(name = colnames(boston_num)) %>%
  gather( , , -one_of("name")) %>%
  ggplot(aes(name, key, fill = value)) +
    geom_tile() +
    scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                         breaks = seq(-1, 1, , by = 0.2)) +
    theme(legend.key.height = unit(45, "pt"))
```



Analyze median value and highway access `rad`.

```
boston %>%
  ggplot(aes(rad, medv)) +
  geom_jitter(aes(color = chas),
              height = 2, width = NULL,
              size = 1, alpha = 0.4) +
  geom_smooth(aes(color = chas), lwd = 1)
```

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



Perhaps `rad = 24` is a missing value.

```
boston %>%
  count(rad)
```

```
## # A tibble: 9 x 2
##     rad     n
##   <int> <int>
## 1     1    20
## 2     2    24
## 3     3    38
## 4     4   110
## 5     5   115
## 6     6    26
## 7     7    17
## 8     8    24
## 9    24   132
```

```
boston %>%
  gather( , , -one_of("rad")) %>%
  group_by(key, rad) %>%
  mutate(value = as.numeric(value)) %>%  # necessary due to factor variable chas
  summarize(z = round(mean(value), 1)) %>%
  spread(rad, z)
```

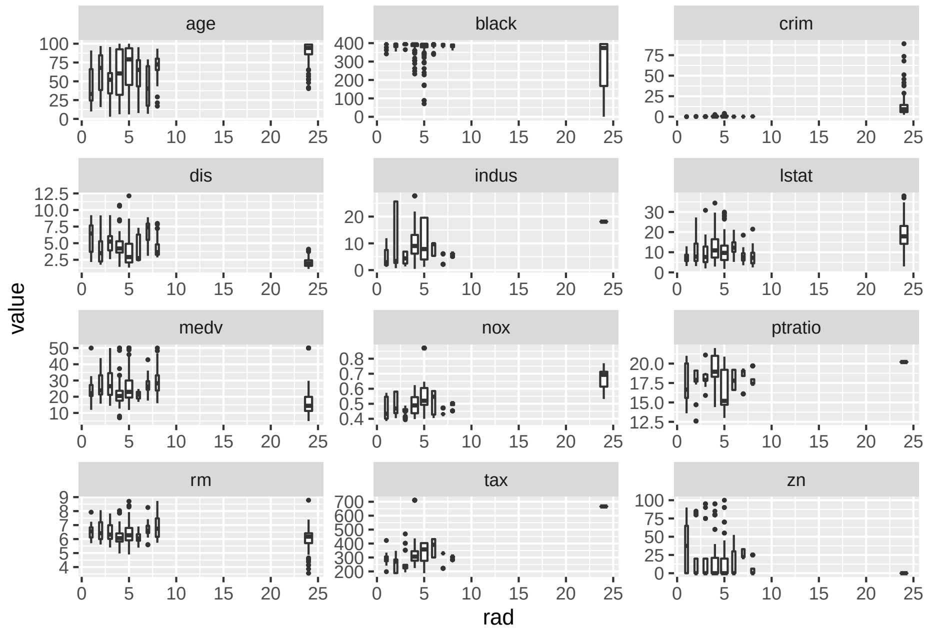
```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

```
## # A tibble: 13 x 10
## # Groups:   key [13]
##    key       `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`  `24`
##    <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 age      45    64.8  49.3  60.8  69.2  60.1  40.1  67.3  89.8
##  2 black   389.  386.  392.  383.  369.  387.  388.  385.  288. 
##  3 chas      0     0     0.1   0.1   0.1   0     0     0.2   0.1
##  4 crim      0     0.1   0.1   0.4   0.7   0.2   0.2   0.4  12.8
##  5 dis       6     4.1   5.1   4.4   3.7   4     6.5   4.4   2.1
##  6 indus     5.1   9.6   4.4  10.7   9.8   8.2   5     5.9  18.1
##  7 lstat     7.4  10     9.1  12.2  10.7  12.3   8     8    18.6
##  8 medv     24.4  26.8  27.9  21.4  25.7  21    27.1  30.4  16.4
##  9 nox       0.5   0.5   0.5   0.5   0.6   0.5   0.4   0.5   0.7
## 10 ptratio  17.6  17.3  18.2  19.1  16.5  17.8  18.4  18    20.2
## 11 rm        6.6   6.6   6.5   6.1   6.4   6.1   6.6   7     6  
## 12 tax     291.  261.  246.  336   332.  373.  304.  301.  666  
## 13 zn       39.9  20.4  16.4  14.7  11.1  13    26.7   6.2   0  
```

Or in helpful boxplot format.

```
boston %>%
  keep(is.numeric) %>%
  gather( , , -one_of("rad")) %>%
  group_by(key, rad) %>%
  ggplot(aes(x = rad, y = value, group = rad)) +
    geom_boxplot(outlier.size = 0.5, varwidth = T) +
    facet_wrap(~ key, ncol = 3, scales = "free") +
  ggsave('plots/rad-boxplot.pdf')
```

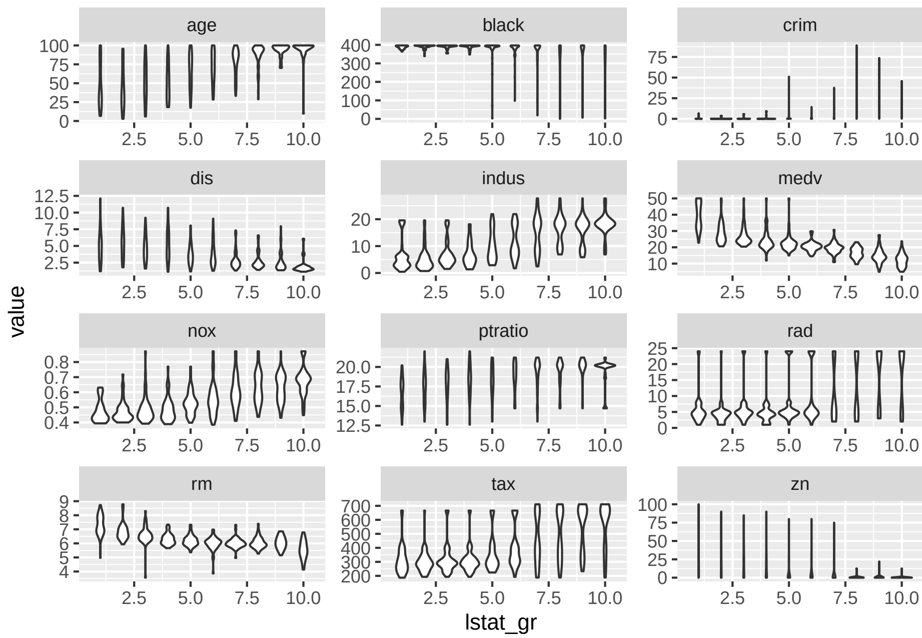
```
## Saving 6.5 x 4.5 in image
```



Looking at `lstat` relationships.

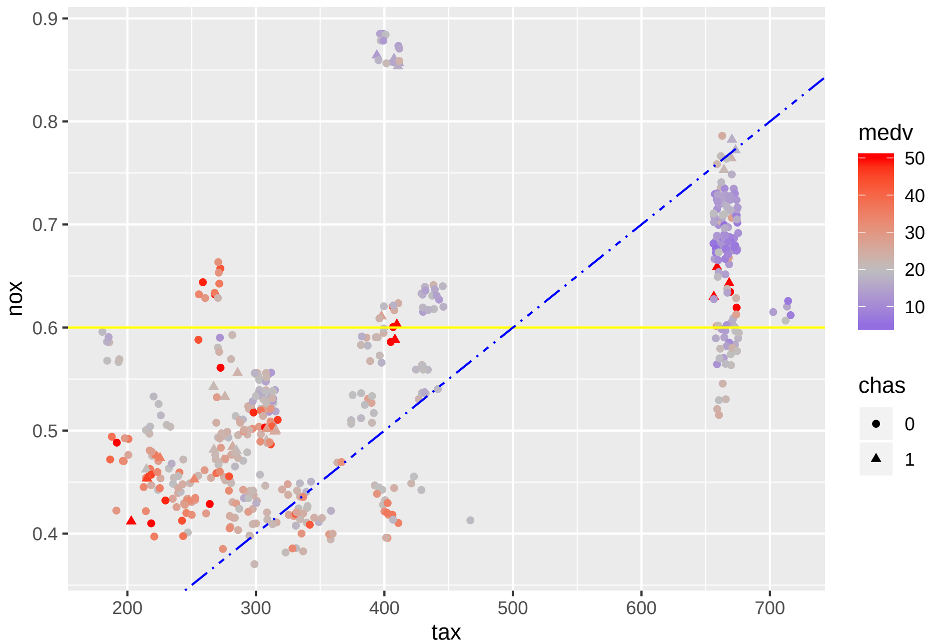
```
boston %>%
  keep(is.numeric) %>%
  gather( , , -one_of("lstat")) %>%
  mutate(lstat_gr = ntile(lstat, 10)) %>%
  group_by(key, lstat_gr) %>%
  ggplot(aes(x = lstat_gr, y = value, group = lstat_gr)) +
    geom_violin() +
    facet_wrap(~ key, ncol = 3, scales = "free") +
  ggsave('plots/lstat-violin.pdf')
```

```
## Saving 6.5 x 4.5 in image
```



Jittering works well for single plots.

```
boston %>%
  ggplot(aes(tax, nox)) +
    geom_jitter(aes(color = medv, shape = chas),
                height = 0.02, width = 10) +
    scale_color_gradient2(midpoint = 20,
                          low = "blue", mid = "gray75", high = "red") +
    geom_hline(yintercept = 0.6, color = "yellow") +
    geom_abline(slope = 0.001, intercept = 0.1, color = "blue", lty = "93133313")
```




```r
ggsave('plots/tax-nox.pdf')
```

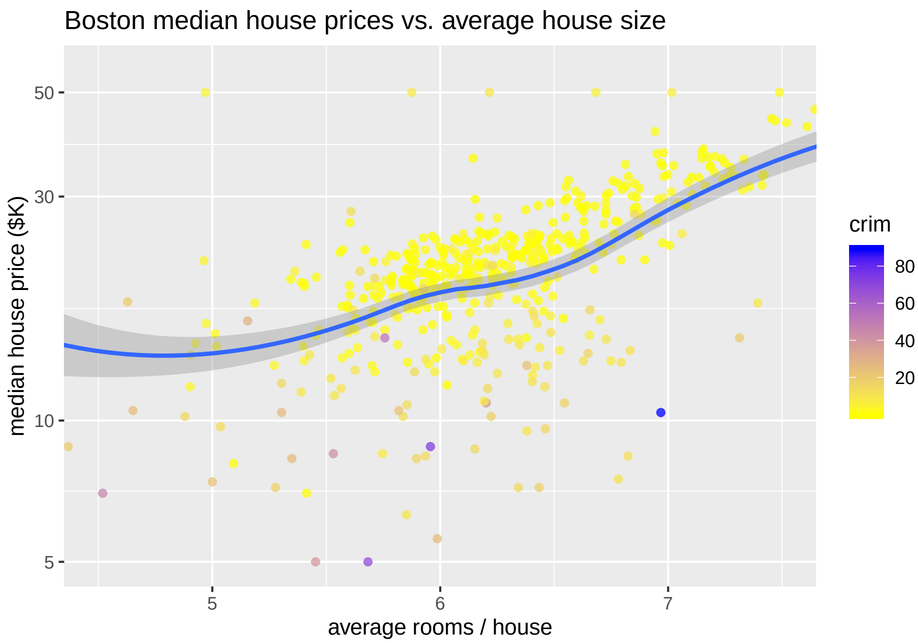
```
## Saving 6.5 x 4.5 in image
```

1.3 Many plotting options

Statistics can be added to the plot as an additional layer. Other layers are coordinates, facets, and scales.

```r
ggplot(data = boston) +
  geom_point(mapping = aes(x = rm, y = medv, color = crim), alpha=0.75) +
  geom_smooth(mapping = aes(x = rm, y = medv)) +
  coord_cartesian(xlim = c(4.5, 7.5)) +
  scale_y_log10() +
  scale_color_gradient(low = "yellow", high = "blue") +
  labs(x = "average rooms / house", y = "median house price ($K)",
       title = "Boston median house prices vs. average house size")
```

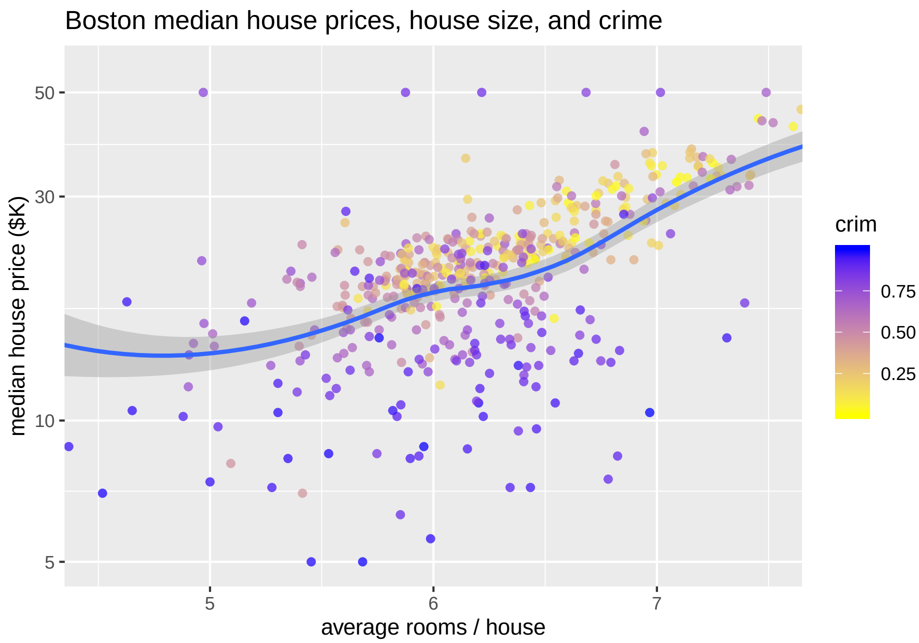
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Maybe more useful if colored by quantile of `crim` value.

```
boston %>%
  mutate(crim = cume_dist(crim)) %>%
  ggplot() +
    geom_point(mapping = aes(x = rm, y = medv, color = crim), alpha=0.75) +
    geom_smooth(mapping = aes(x = rm, y = medv)) +
    coord_cartesian(xlim = c(4.5, 7.5)) +
    scale_y_log10() +
    scale_color_gradient(low = "yellow", high = "blue") +
    labs(x = "average rooms / house", y = "median house price ($K)",
       title = "Boston median house prices, house size, and crime")
```

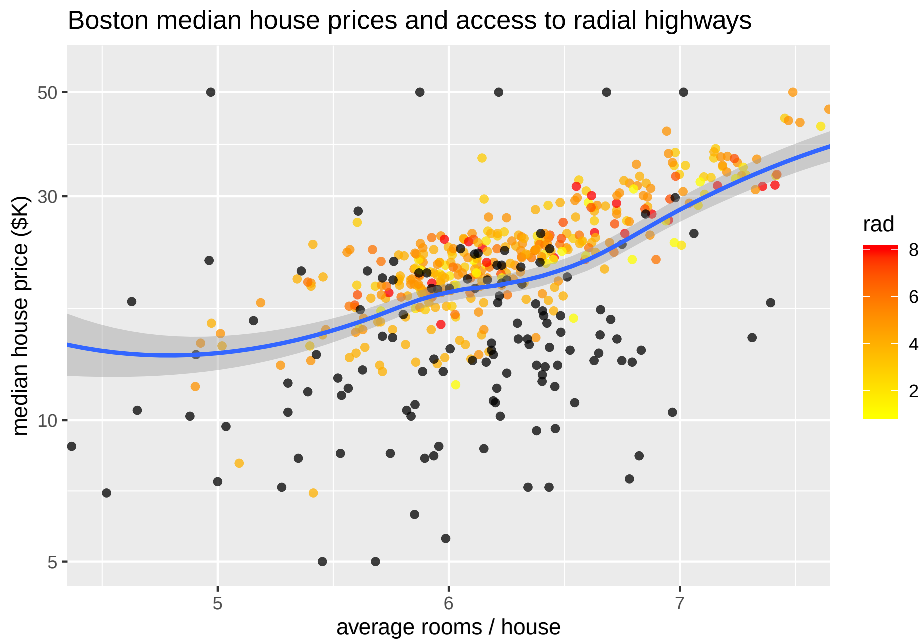
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Now color by `rad` but change all 24's to NA's.

```
boston %>%
  mutate(rad = ifelse(rad == 24, NA, rad)) %>%
  ggplot() +
    geom_point(mapping = aes(x = rm, y = medv, color = rad), alpha=0.75) +
    geom_smooth(mapping = aes(x = rm, y = medv)) +
    coord_cartesian(xlim = c(4.5, 7.5)) +
    scale_y_log10() +
    scale_color_gradient(low = "yellow", high = "red", na.value = "black") +
    labs(x = "average rooms / house", y = "median house price ($K)",
         title = "Boston median house prices and access to radial highways")
```

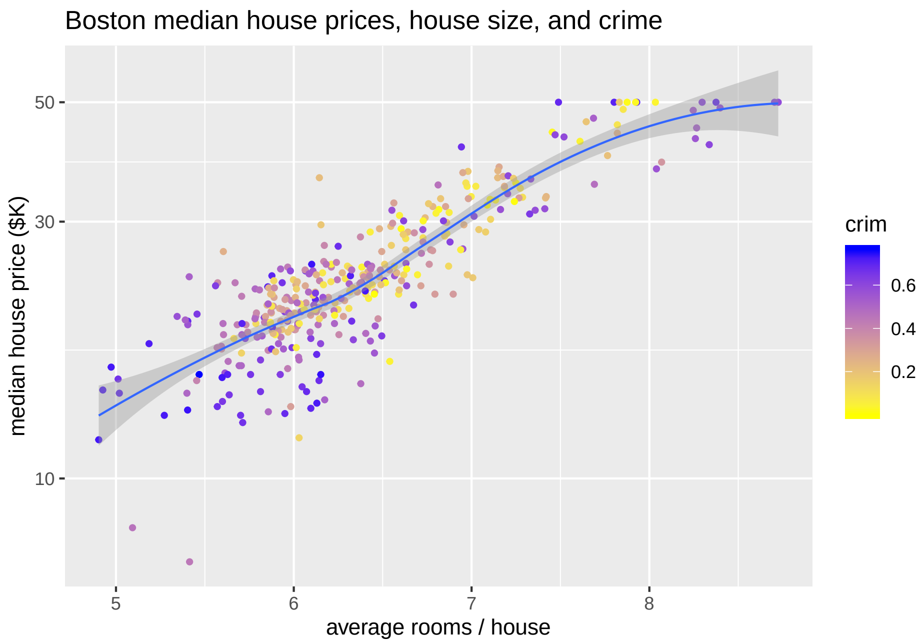
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Maybe excluding newly-NA'ed `rad` values helps the crime plot.

```
boston %>%
  mutate(rad = ifelse(rad == 24, NA, rad)) %>%
  mutate(crim = cume_dist(crim)) %>%
  filter(!is.na(rad)) %>%
  ggplot() +
    geom_point(mapping = aes(x = rm, y = medv, color = crim), size = 1) +
    geom_smooth(mapping = aes(x = rm, y = medv), lwd = 0.5) +
    scale_y_log10() +
    scale_color_gradient(low = "yellow", high = "blue") +
    labs(x = "average rooms / house", y = "median house price ($K)",
       title = "Boston median house prices, house size, and crime")
```

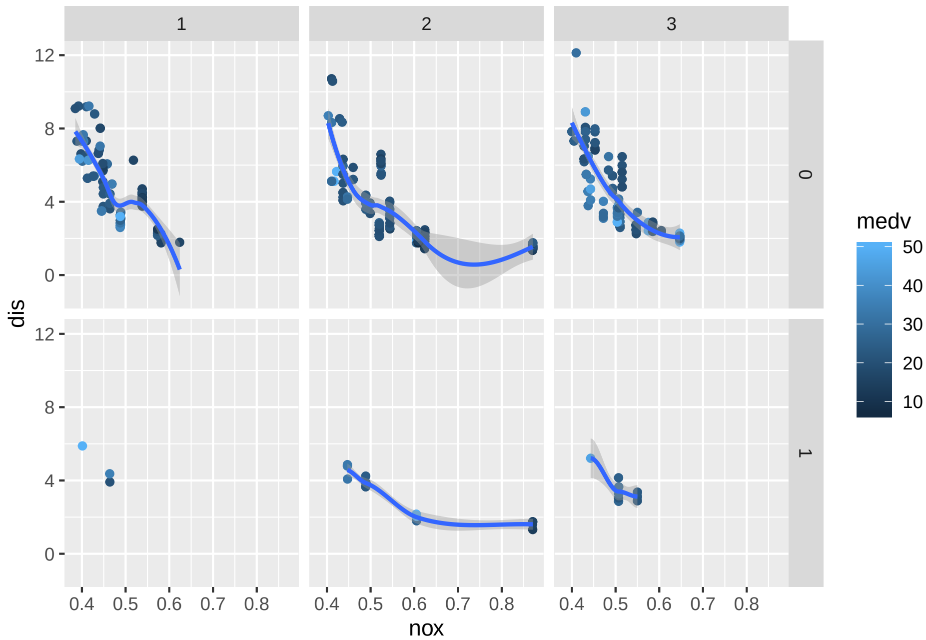
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



A grid of `nox` vs. `dis` plots according to `chas` (rows) and binned level (ntile) of `rad`.

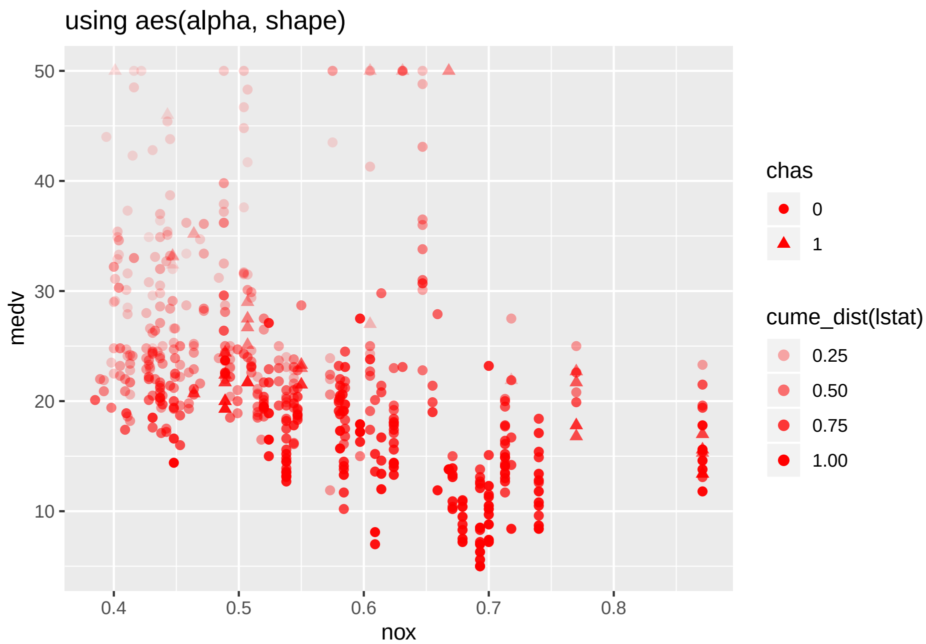
```
boston %>%
  mutate(rad = ifelse(rad == 24, NA, rad)) %>%
  filter(!is.na(rad)) %>%
  ggplot(aes(nox, dis, color = medv)) +
    geom_jitter() +
    facet_grid(chas ~ ntile(rad, 3)) +
    geom_smooth()
```

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



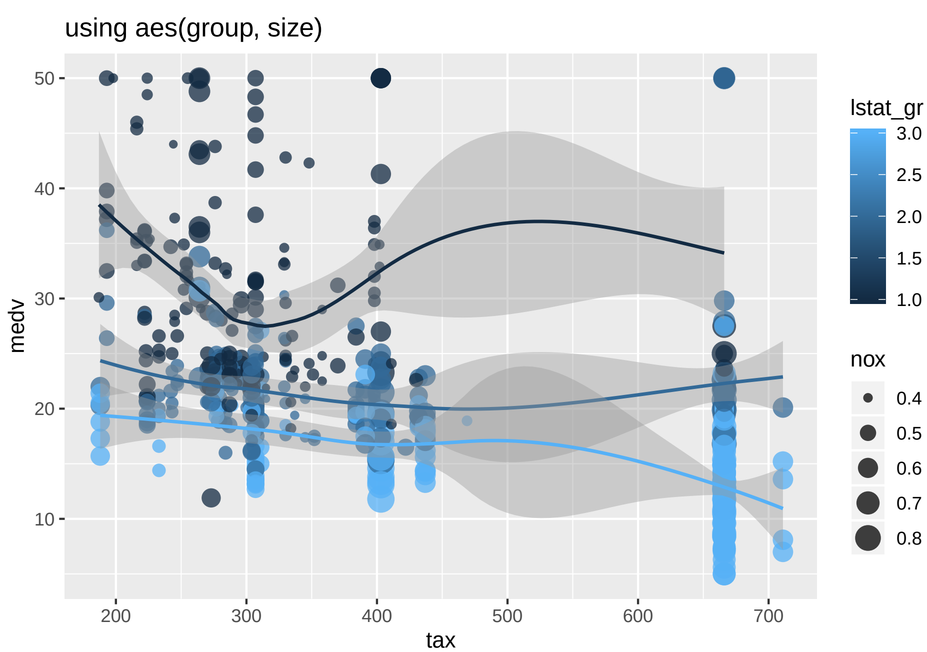
Multiplots available with **gridExtra**, used by ggplot2.

```
require(gridExtra)
p1 <- ggplot(boston) +
  geom_point(aes(nox, medv, shape = chas, alpha = cume_dist(lstat)),
             color = 'red', size = 2) +
  labs(title = 'using aes(alpha, shape)')
p2 <- boston %>%
  mutate(lstat_gr = ntile(lstat, 3)) %>%
  ggplot(aes(tax, medv, color = lstat_gr, size = nox)) +
  geom_point(shape = 16, alpha = 0.75) +
  geom_smooth(aes(group = lstat_gr), lwd = 0.8) +
  labs(title = 'using aes(group, size)')
p1
```




```
p2
```

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



```
ggsave('plots/two-plot.pdf', arrangeGrob(p1, p2))
```

```
## Saving 6.5 x 4.5 in image
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

Chapter 2

dplyr and tidyr

```
library(tidyverse)
library(gridExtra)
batting <- as_tibble(Lahman::Batting)
fielding <- as_tibble(Lahman::Fielding)
```

2.1 Hoofin' it with dplyr

Condense batting stats into player career totals, keep only those >= 500 games.

```
is_col <- names(batting)[c(1, 2, 4, 6:17)]
is_num <- names(batting)[sapply(batting, is.numeric)]
gt_500 <- batting %>%
  select(is_col) %>%
  select(-teamID) %>%
  drop_na() %>%
  group_by(playerID) %>%
  summarize_at(is_col[-(1:3)], sum, na.rm = T) %>%
  filter(G >= 500)
```

All Ha~ Green~ statistics:

```
batting %>%
  filter(str_detect(playerID, "greenha"))
```

```
## # A tibble: 14 x 22
##    playerID yearID stint teamID lgID      G    AB     R     H   X2B   X3B
##    <chr>     <int> <int> <fct>  <fct> <int> <int> <int> <int> <int> <int>
##  1 greenha~   1930     1 DET    AL        1     1     0     0     0     0
##  2 greenha~   1933     1 DET    AL      117   449    59   135    33     3
```

```
##  3 greenha~  1934      1 DET   AL      153   593   118   201    63     7
##  4 greenha~  1935      1 DET   AL      152   619   121   203    46    16
##  5 greenha~  1935      1 BRO   NL        2     0     0     0     0     0
##  6 greenha~  1936      1 DET   AL       12    46    10    16     6     2
##  7 greenha~  1937      1 DET   AL      154   594   137   200    49    14
##  8 greenha~  1938      1 DET   AL      155   556   144   175    23     4
##  9 greenha~  1939      1 DET   AL      138   500   112   156    42     7
## 10 greenha~  1940      1 DET   AL      148   573   129   195    50     8
## 11 greenha~  1941      1 DET   AL       19    67    12    18     5     1
## 12 greenha~  1945      1 DET   AL       78   270    47    84    20     2
## 13 greenha~  1946      1 DET   AL      142   523    91   145    29     5
## 14 greenha~  1947      1 PIT   NL      125   402    71   100    13     2
## # ... with 11 more variables: HR <int>, RBI <int>, SB <int>, CS <int>,
## #   BB <int>, SO <int>, IBB <int>, HBP <int>, SH <int>, SF <int>,
## #   GIDP <int>
```

Positions by game.

```
fielding %>%
  group_by(POS) %>%
  count(wt = G)
```

```
## # A tibble: 7 x 2
## # Groups:   POS [7]
##   POS         n
##   <chr>   <int>
## 1 1B     482698
## 2 2B     480968
## 3 3B     482320
## 4 C      497547
## 5 OF    1451301
## 6 P     1106574
## 7 SS     479045
```

Attach a column denoting their main fielding position.

```
is_field = names(fielding)[c(1, 6, 7, 9, 10, 11, 12, 13)]
fielding %>%
  select(is_field) %>%
  map(~ sum(is.na(.)))
```

```
## $playerID
## [1] 0
##
## $POS
## [1] 0
##
## $G
```

```
## [1] 0
##
## $InnOuts
## [1] 29929
##
## $PO
## [1] 0
##
## $A
## [1] 0
##
## $E
## [1] 1
##
## $DP
## [1] 0
```

Removing InnOuts is a good idea, too many missing.

```
is_field = names(fielding)[c(1, 6, 7, 10, 11, 12, 13)]
pos_tot <- fielding %>%
  select(is_field) %>%
  drop_na() %>%
  group_by(playerID, POS) %>%
  summarize_all(sum) %>%
  ungroup() %>%
  filter(G >= 100) %>%
  arrange(playerID, desc(G)) %>%
  group_by(playerID) %>%
  mutate(pos1 = first(POS)) %>%
  filter(POS == pos1) %>%
  select(-pos1)
```

2.2 tidyr and relational data

Add fielding info to batting tibble.

```
(batpos <- gt_500 %>%
   left_join(pos_tot, by = "playerID"))
```

```
## # A tibble: 2,667 x 19
##    playerID   G.x    AB     R     H   X2B   X3B    HR   RBI    SB    CS
##    <chr>    <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
##  1 aaronha~  3298 12364  2174  3771   624    98   755  2297   240    73
##  2 abbotku~   702  2044   273   523   109    23    62   242    22    11
```

```
##  3 abernte~   681   181    12    25     3     0     0     9     0     0
##  4 abramca~   521  1543   246   422    62    19    32   134    11    18
##  5 abreubo~  2425  8480  1453  2470   574    59   288  1363   400   128
##  6 abreujo~   742  2913   398   858   180    13   146   488     8     3
##  7 ackledu~   635  2125   261   512    94    18    46   216    31    12
##  8 adairje~  1165  4019   378  1022   163    19    57   366    29    29
##  9 adamsbo~   797  2604   395   701   107    31    25   188    25    30
## 10 adamsgl~   661  1617   152   452    79     5    34   225     6    10
## # ... with 2,657 more rows, and 8 more variables: BB <int>, SO <int>,
## #   POS <chr>, G.y <int>, PO <int>, A <int>, E <int>, DP <int>
```

Counts of positions.

```
batpos %>%
  group_by(POS) %>%
  count()
```

```
## # A tibble: 8 x 2
## # Groups:   POS [8]
##   POS       n
##   <chr> <int>
## 1 <NA>      2
## 2 1B      254
## 3 2B      277
## 4 3B      270
## 5 C       300
## 6 OF      890
## 7 P       378
## 8 SS      296
```

NAs are likely DHs.

```
pos_nas <- batpos %>%
  filter(is.na(POS))
batting %>%
  inner_join(pos_nas, by = "playerID")
```

```
## # A tibble: 26 x 40
##    playerID yearID stint teamID lgID      G  AB.x   R.x   H.x X2B.x X3B.x
##    <chr>     <int> <int> <fct>  <fct> <int> <int> <int> <int> <int> <int>
##  1 moraljo~   1973     1 OAK    AL        6    14     0     4     1     0
##  2 moraljo~   1973     2 MON    NL        5     5     0     2     0     0
##  3 moraljo~   1974     1 MON    NL       25    26     3     7     4     0
##  4 moraljo~   1975     1 MON    NL       93   163    18    49     6     1
##  5 moraljo~   1976     1 MON    NL      104   158    12    50    11     0
##  6 moraljo~   1977     1 MON    NL       65    74     3    15     4     1
##  7 moraljo~   1978     1 MIN    AL      101   242    22    76    13     1
##  8 moraljo~   1979     1 MIN    AL       92   191    21    51     5     1
```

```
##  9 moraljo~   1980     1 MIN    AL       97   241    36    73    17     2
## 10 moraljo~   1981     1 BAL    AL       38    86     6    21     3     0
## # ... with 16 more rows, and 29 more variables: HR.x <int>, RBI.x <int>,
## #   SB.x <int>, CS.x <int>, BB.x <int>, SO.x <int>, IBB <int>, HBP <int>,
## #   SH <int>, SF <int>, GIDP <int>, G.x <int>, AB.y <int>, R.y <int>,
## #   H.y <int>, X2B.y <int>, X3B.y <int>, HR.y <int>, RBI.y <int>,
## #   SB.y <int>, CS.y <int>, BB.y <int>, SO.y <int>, POS <chr>, G.y <int>,
## #   PO <int>, A <int>, E <int>, DP <int>
```

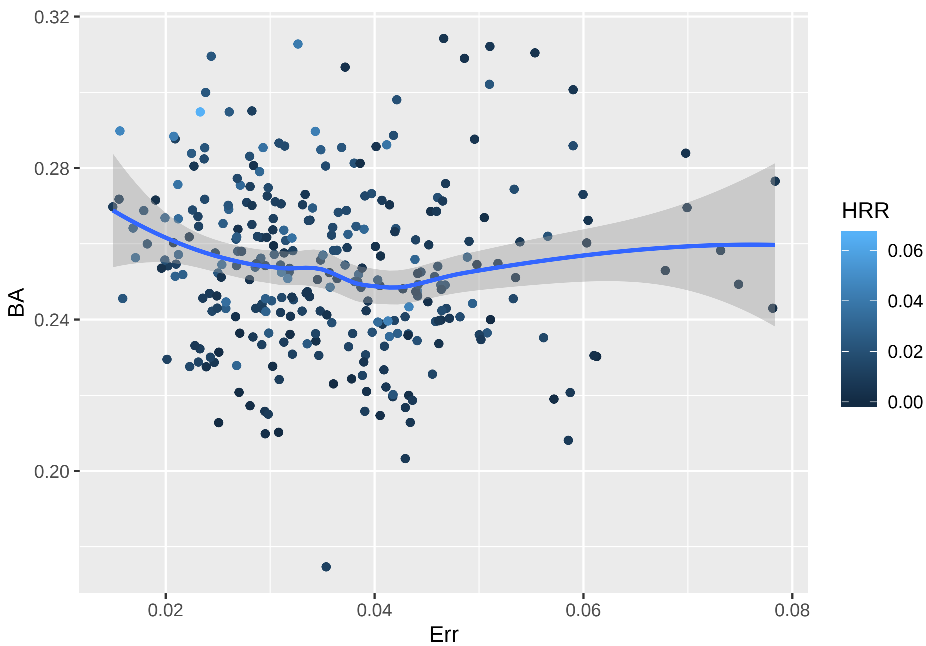
Drop these two DHs.

```
batpos <- batpos %>%
  drop_na()
```

Now we could explore many aspects of hitting stats vs. position, and see what position players were better fielders or better hitters, or if neither we can see if they played for the Expos.

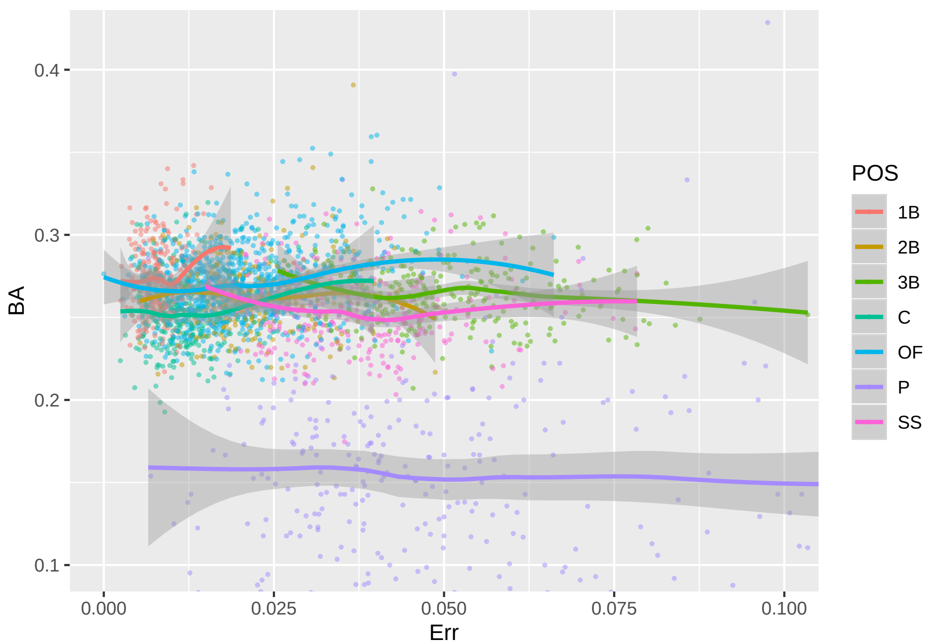
```
batpos %>%
  filter(POS == "SS") %>%
  mutate(BA = H / AB) %>%  # batting average, hits / at bats
  mutate(Err = E / (PO + A)) %>%   # error rate, errors / (put outs + assists)
  mutate(HRR = HR / AB) %>%  # home run rate, home runs / at bats
  ggplot(aes(Err, BA)) +
    geom_point(aes(color = HRR)) +
    geom_smooth()
```

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



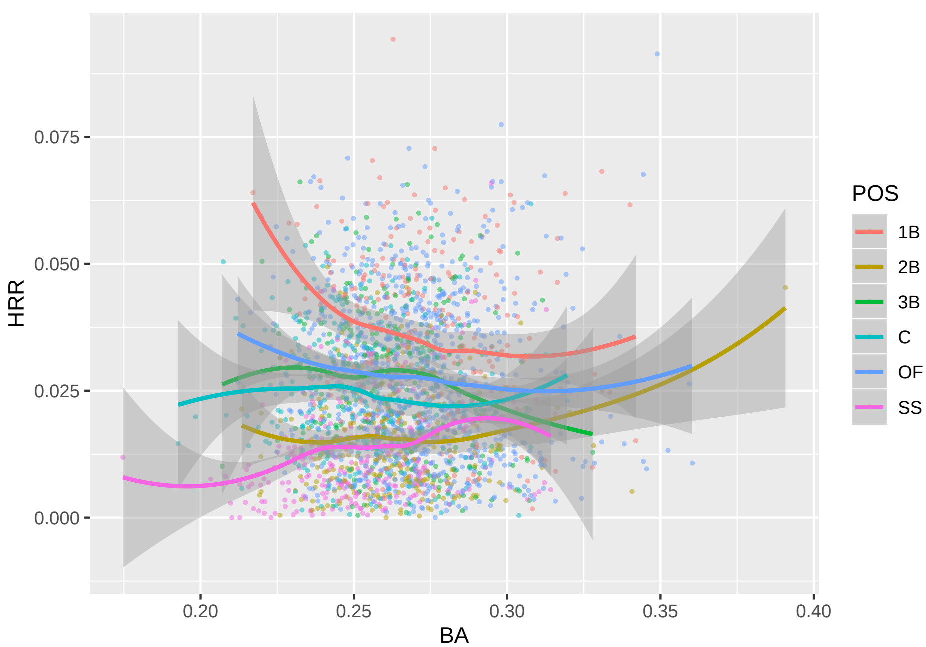

```
temp <- batpos %>%
  mutate(BA = H / AB) %>%  # batting average, hits / at bats
  filter(between(BA, 0.01, 0.49)) %>%
  mutate(Err = E / (PO + A)) %>%   # error rate, errors / (put outs + assists)
  mutate(HRR = HR / AB)  # home run rate, home runs / at bats
p1 <- temp %>%
  ggplot(aes(Err, BA, color = POS)) +
    geom_point(alpha = 0.5, size = 0.5) +
    geom_smooth(aes(group = POS)) +
    coord_cartesian(xlim = c(0, 0.1), ylim = c(0.1, 0.42))
p2 <- temp %>%
  filter(POS != "P") %>%
  ggplot(aes(BA, HRR, color = POS)) +
    geom_point(alpha = 0.5, size = 0.5) +
    geom_smooth(aes(group = POS))
p1
```

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




```
p2
```

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



```
ggsave('plots/pos-bat.pdf', arrangeGrob(p1, p2))
```

```
## Saving 6.5 x 4.5 in image
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2019). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.11.