A Gentle Guide to the Grammar of Graphics with `ggplot2`

Tampa R Users Meetup

Garrick Aden-Buie
@grrrck

2018-01-23

Follow along: **bit.ly/trug-ggplot2**

Brought to you by the letter...
G
THE CLASSIC
CRAYON FACTORY

Why *ggplot2*?

Hadley Wickham

The transferrable skills from ggplot2 are not the idiosyncracies of plotting syntax, but a powerful way of thinking about visualisation, as a way of **mapping between variables and the visual properties of geometric objects** that you can perceive.

http://disq.us/p/sv640d

Why *ggplot2*?

My personal reasons

- Functional data visualization

    1. Wrange data
    2. Map data to visual elements
    3. Tweak scales, guides, axis, labels, theme

- Easy to reason about how data drives visualization

- Easy to iterate

- Easy to be consistent

What are we getting into?

`ggplot2` is a huge package: philosophy + functions

...but it's very well organized

What are we getting into?

`ggplot2` is a huge package: philosophy + functions
...but it's very well organized

Lots of examples of not-so-great plots in these slides
...but that's okay

What are we getting into?

`ggplot2` is a huge package: philosophy + functions
...but it's very well organized

Lots of examples of not-so-great plots in these slides
...but that's okay

Going to throw a lot at you
...but you'll know *where* and *what* to look for

What are we getting into?

`ggplot2` is a huge package: philosophy + functions
...but it's very well organized

Lots of examples of not-so-great plots in these slides
...but that's okay

Going to throw a lot at you
...but you'll know *where* and *what* to look for

What are we getting into?

`ggplot2` is a huge package: philosophy + functions
...but it's very well organized

Lots of examples of not-so-great plots in these slides
...but that's okay

Going to throw a lot at you
...but you'll know *where* and *what* to look for

G is for getting started

Easy: install the tidyverse

```
install.packages('tidyverse')
```

Medium: install just `ggplot2`

```
install.pacakages('ggplot2')
```

Expert: install from GitHub

```
devtools::install_github('tidyverse/ggplot2')
```

G is for getting started

Load the tidyverse

```r
library(tidyverse)
```

```
## ── Attaching packages ───────────────────────────────────────────────────────

## ✔ ggplot2 2.2.1     ✔ purrr   0.2.4
## ✔ tibble  1.4.1     ✔ dplyr   0.7.4
## ✔ tidyr   0.7.2     ✔ stringr 1.2.0
## ✔ readr   1.1.1     ✔ forcats 0.2.0

## ── Conflicts ──────────────────────────────────────────────────────────────
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

G is for getting started

Other packages you'll need for this adventure

```r
library(lubridate)        # tidyverse
library(reshape2)         # install.packages("reshape2")
library(babynames)        # install.packages("babynames")
```

gg is for Grammar of Graphics

Data

```
g ← ggplot()
```

Tidy Data

1. Each variable forms a column
2. Each observation forms a row
3. Each observational unit forms a table

The following example draws from

```
data(population, package = "tidyr")
```

gg is for Grammar of Graphics

Data

```
g ← ggplot()
```

country	1995	2000	2005	2010
Canada	29.2949	30.69742	32.25309	34.12624
China	1237.5314	1280.42858	1318.17683	1359.82146
USA	268.0397	284.59440	298.16580	312.24712

year	Canada	China	USA
1995	29.29490	1237.531	268.0397
2000	30.69742	1280.429	284.5944
2005	32.25309	1318.177	298.1658
2010	34.12624	1359.821	312.2471

gg is for Grammar of Graphics

Data

```
tidy1 <- gather(messy1, 'year', 'population', -country)
```

```
g <- ggplot()
```

country	year	population
Canada	1995	29.295
China	1995	1237.531
USA	1995	268.040
Canada	2000	30.697
China	2000	1280.429
USA	2000	284.594
Canada	2005	32.253
China	2005	1318.177
USA	2005	298.166
Canada	2010	34.126
China	2010	1359.821

gg is for Grammar of Graphics

Data

```
tidy2 <- gather(messy2, 'country', 'population', -year)
```

```
g <- ggplot()
```

year	country	population
1995	Canada	29.295
2000	Canada	30.697
2005	Canada	32.253
2010	Canada	34.126
1995	China	1237.531
2000	China	1280.429
2005	China	1318.177
2010	China	1359.821
1995	USA	268.040
2000	USA	284.594
2005	USA	298.166

gg is for Grammar of Graphics

Data

Aesthetics

```
g + aes()
```

Map data to visual elements or parameters

- year
- population
- country

gg is for Grammar of Graphics

Data

Aesthetics

```
g + aes()
```

Map data to visual elements or parameters

- year → **x**
- population → **y**
- country → *shape*, *color*, etc.

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
g + geom_*()
```

Geometric objects displayed on the plot:

Type	Function
Point	`geom_point()`
Line	`geom_line()`
Bar	`geom_bar()`, `geom_col()`
Histogram	`geom_histogram()`
Regression	`geom_smooth()`
Boxplot	`geom_boxplot()`
Text	`geom_text()`
Vert./Horiz. Line	`geom_{vh}line()`
Count	`geom_count()`
Density	`geom_density()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
g + geom_*()
```

Those are just the top 10 most popular geoms[1]

See http://ggplot2.tidyverse.org/reference/ for many more options

Or just start typing `geom_` in RStudio

```
##  [1] "geom_abline"     "geom_area"       "geom_bar"        "geom_bin2d"
##  [5] "geom_blank"      "geom_boxplot"    "geom_col"        "geom_contour"
##  [9] "geom_count"      "geom_crossbar"   "geom_curve"      "geom_density"
## [13] "geom_density_2d" "geom_density2d"  "geom_dotplot"    "geom_errorbar"
## [17] "geom_errorbarh"  "geom_freqpoly"   "geom_hex"        "geom_histogram"
## [21] "geom_hline"      "geom_jitter"     "geom_label"      "geom_line"
## [25] "geom_linerange"  "geom_map"        "geom_path"       "geom_point"
## [29] "geom_pointrange" "geom_polygon"    "geom_qq"         "geom_quantile"
## [33] "geom_raster"     "geom_rect"       "geom_ribbon"     "geom_rug"
## [37] "geom_segment"    "geom_smooth"     "geom_spoke"      "geom_step"
## [41] "geom_text"       "geom_tile"       "geom_violin"     "geom_vline"
```

[1] https://eric.netlify.com/2017/08/10/most-popular-ggplot2-geoms/

Our first plot!

```
ggplot(tidy1)
```

Our first plot!

```
ggplot(tidy1) +
  aes(x = year,
      y = population)
```

Our first plot!

```
ggplot(tidy1) +
  aes(x = year,
      y = population) +
  geom_point()
```

Our first plot!

```
ggplot(tidy1) +
  aes(x = year,
      y = population,
      color = country) +
  geom_point()
```

Our first plot!

```
ggplot(tidy1) +
  aes(x = year,
      y = population,
      color = country) +
  geom_point() +
  geom_line()
```

```
geom_path: Each group consists
of only one observation.
Do you need to adjust the
group aesthetic?
```

Our first plot!

```
ggplot(tidy1) +
  aes(x = year,
      y = population,
      color = country) +
  geom_point() +
  geom_line(
    aes(group = country))
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
g + geom_*()
```

```
geom_*(mapping, data, stat, position)
```

- `data` Geoms can have their own data
  - Has to map onto global coordinates
- `map` Geoms can have their own aesthetics
  - Inherits global aesthetics
  - Have geom-specific aesthetics
    - `geom_point` needs `x` and `y`, optional `shape`, `color`, `size`, etc.
    - `geom_ribbon` requires `x`, `ymin` and `ymax`, optional `fill`
  - `?geom_ribbon`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
g + geom_*()
```

```
geom_*(mapping, data, stat, position)
```

- `stat` Some geoms apply further transformations to the data
  - All respect `stat = 'identity'`
  - Ex: `geom_histogram` uses `stat_bin()` to group observations
- `position` Some adjust location of objects
  - `'dodge'`, `'stack'`, `'jitter'`

Example: Stat and Position

Star Wars Characters

```
sw_chars ← starwars %>%
  mutate(
    n_movies = map_int(films, length),
    gender = ifelse(
      !gender %in% c('female', 'male'),
      'other', gender)
  ) %>%
  select(name, gender, n_movies)
```

name	gender	n_movies
Luke Skywalker	male	5
C-3PO	other	6
R2-D2	other	7
Darth Vader	male	4
Leia Organa	female	5
Owen Lars	male	3
Beru Whitesun lars	female	3
R5-D4	other	1
Biggs Darklighter	male	1
Obi-Wan Kenobi	male	6
Anakin Skywalker	male	3
Wilhuff Tarkin	male	2
Chewbacca	male	5

Example: Stat and Position

```
ggplot(sw_chars) +
  aes(x = n_movies) +
  geom_bar(stat = "count")
```

Example: Stat and Position

```
ggplot(sw_chars) +
  aes(x = n_movies,
      fill = gender) +
  geom_bar(stat = "count")
```

Example: Stat and Position

```
sw_chars_id ← sw_chars %>%
  group_by(n_movies, gender) %>%
  tally
```

n_movies	gender	n
1	female	9
1	male	34
1	other	3
2	female	6
2	male	12
3	female	3
3	male	9
3	other	1
4	male	2
5	female	1
5	male	4
6	male	1
6	other	1

Example: Stat and Position

```
ggplot(sw_chars_id) +
  aes(x = n_movies,
      y = n,
      fill = gender) +
  geom_bar(stat = 'identity')
```

Note: `geom_col()` is alias for `geom_bar(stat = 'identity')`

Example: Stat and Position

```
ggplot(sw_chars_id) +
  aes(x = n_movies,
      y = n,
      fill = gender) +
  geom_col(position = "fill")
```

Example: Stat and Position

```
ggplot(sw_chars_id) +
  aes(x = n_movies,
      y = n,
      fill = gender) +
  geom_col(position = "dodge")
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

```
g+facet_wrap()

g+facet_grid()
```

```
g ← ggplot(sw_chars) +
  aes(x = n_movies,
      fill = gender) +
  geom_bar()
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

```
g+facet_wrap()

g+facet_grid()
```

```
g + facet_wrap(~ gender)
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

```
g+facet_wrap()

g+facet_grid()
```

```
g + facet_grid(gender ~ hair_color)
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

```
g+facet_wrap()

g+facet_grid()
```

```
g + facet_grid(gender ~ hair_color, scales = 'free_y')
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

```
g + labs()
```

```
g ← g +
  labs(
    x = "Film Appearances",
    y = "Count of Characters",
    title = "Recurring Star Wars Characters",
    subtitle = "How often do characters appear?",
    fill = "Gender"
  )
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

```
g + labs()
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Scales

```
g + scale_*_*()
```

`scale` + `_` + `<aes>` + `_` + `<type>` + `()`

What parameter do you want to adjust? → `<aes>`
What type is the parameter? → `<type>`

- I want to change my discrete x-axis
  `scale_x_discrete()`
- I want to change point size from continuous variable
  `scale_size_continuous()`
- I want to rescale y-axis as log
  `scale_y_log10()`
- I want to use a different color palette
  `scale_fill_discrete()`
  `scale_color_manual()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Scales

```
g + scale_*_*()
```

```
g ← g + scale_fill_brewer(palette = 'Set1')
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Scales

Theme

```
g + theme()
```

Change the appearance of plot decorations
i.e. things that aren't mapped to data

A few "starter" themes ship with the package

- `g + theme_bw()`
- `g + theme_dark()`
- `g + theme_gray()`
- `g + theme_light()`
- `g + theme_minimal()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Scales

Theme

```
g + theme()
```

Huge number of parameters, grouped by plot area:

- Global options: `line`, `rect`, `text`, `title`
- `axis`: x-, y- or other axis title, ticks, lines
- `legend`: Plot legends
- `panel`: Actual plot area
- `plot`: Whole image
- `strip`: Facet labels

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Scales

Theme

```
g + theme()
```

Theme options are supported by helper functions:

- `element_blank()` removes the element
- `element_line()`
- `element_rect()`
- `element_text()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Scales

Theme

```
g + theme()
```

```
g + theme_bw()
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Scales

Theme

```
g + theme()
```

```
g + theme_minimal() + theme(text = element_text(family = "Palatino"))
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Scales

Theme

```
g + theme()
```

You can also set the theme globally with `theme_set()`

```
my_theme <- theme_bw() +
  theme(
    text = element_text(family = "Palatino", size = 12),
    panel.border = element_rect(colour = 'grey80'),
    panel.grid.minor = element_blank()
  )

theme_set(my_theme)
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Scales

Theme

```
g + theme()
```

```
g
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Scales

Theme

```
g + theme()
```

```
g + theme(legend.position = 'bottom')
```

You have the power!

"Live" Coding

```
data(tips, package = "reshape2")
```

head(tips)

total_bill	tip	sex	smoker	day	time	size
16.99	1.01	Female	No	Sun	Dinner	2
10.34	1.66	Male	No	Sun	Dinner	3
21.01	3.50	Male	No	Sun	Dinner	3
23.68	3.31	Male	No	Sun	Dinner	2
24.59	3.61	Female	No	Sun	Dinner	4
25.29	4.71	Male	No	Sun	Dinner	4
8.77	2.00	Male	No	Sun	Dinner	2
26.88	3.12	Male	No	Sun	Dinner	4
15.04	1.96	Male	No	Sun	Dinner	2
14.78	3.23	Male	No	Sun	Dinner	2

tips: tip histogram

```
ggplot(tips) +
  aes(x = tip) +
  geom_histogram(
    binwidth = 0.25
  )
```

tips: tip density

```
ggplot(tips) +
  aes(x = tip) +
  geom_density(
    aes(fill = day)
  )
```

tips: tip density

```
ggplot(tips) +
  aes(x = tip) +
  geom_density(
    aes(fill = day),
    alpha = 0.4
  )
```

tips: tip density

```
ggplot(tips) +
  aes(x = tip/total_bill) +
  geom_density(
    aes(fill = day)
  ) +
  facet_wrap(~ day)
```

tips: tip vs total

```
ggplot(tips) +
  aes(x = total_bill,
      y = tip) +
  geom_point()
```

tips: tip vs total

```
ggplot(tips) +
  aes(x = total_bill,
      y = tip) +
  geom_point() +
  geom_smooth(method = "lm")
```

tips: tip vs total

```r
ggplot(tips) +
  aes(x = total_bill,
      y = tip) +
  geom_point() +
  geom_smooth(method = "lm")+
  geom_abline(
    slope = c(0.2, 0.15),
    intercept = 0,
    color = c('#69b578',
              "#dd1144"),
    linetype = 3)
```

tips: tip vs total

```
ggplot(tips) +
  aes(x = total_bill,
      y = tip/total_bill) +
  geom_point() +
  geom_hline(
    yintercept = c(0.2, 0.15),
    color = c('#69b578',
              "#dd1144"),
    linetype = 1)
```

tips: tip vs total

```
tips$percent <-
  tips$tip/tips$total_bill

ggplot(tips) +
  aes(x = size,
      y = percent,
      color = smoker) +
  geom_point()
```

tips: tip vs total

```
tips$percent <-
  tips$tip/tips$total_bill

ggplot(tips) +
  aes(x = size,
      y = percent,
      color = smoker) +
  geom_jitter(width = 0.25)
```

tips: tip vs total

```
ggplot(tips) +
  aes(x = day,
      y = percent,
      color = sex) +
  geom_jitter(width = 0.25) +
  facet_grid(time ~ smoker)
```

tips: tip vs total

```r
tips <- mutate(tips,
  time = factor(time,
    c("Lunch", "Dinner")),
  day = factor(day,
    c("Thur", "Fri",
      "Sat", "Sun")
  ))

ggplot(tips) +
  aes(x = day,
      y = percent,
      color = sex) +
  geom_jitter(width = 0.25) +
  facet_grid(time ~ smoker)
```

tips: tip vs total

```
ggplot(tips) +
  aes(x = day,
      y = percent,
      fill = time) +
  geom_boxplot() +
  facet_grid(. ~ smoker)
```

tips: tip vs total

```
ggplot(tips) +
  aes(x = day,
      y = percent,
      color = smoker,
      fill  = smoker) +
  geom_violin(alpha = 0.3) +
  facet_wrap(~ smoker)
```

tips: tip vs total

```
g ← ggplot(tips) +
  aes(x = day,
      y = percent,
      color = smoker,
      fill = smoker) +
  geom_violin(alpha = 0.3) +
  geom_jitter(alpha = 0.4,
              width = 0.25,
              size  = 0.8)+
  facet_wrap(~ smoker)
g
```

tips: tip vs total

```
g + guides(color = FALSE,
           fill  = FALSE) +
  labs(x = '',
       y = 'Tip Rate') +
  scale_y_continuous(
    labels = scales::percent
  )
```

Level up

```
data(babynames, 'babynames')
```

head(babynames)

The babynames package contains data provided by the USA social security administration:

- `babynames`: For each year from 1880 to 2015, the number of children of each sex given each name. All names with more than 5 uses are given.

year	sex	name	n	prop
1989	F	Taunya	8	0.000
1982	M	Zebulin	5	0.000
1884	M	Wheeler	10	0.000
1989	F	Neelam	18	0.000
1957	F	Kelly	1909	0.001
2004	F	Gizell	8	0.000

Most popular baby names in 2015

```
babynames_pop2015 <- babynames %>%
  filter(year == 2015) %>%
  mutate(
    n = n/1000,
    sex = case_when(
      sex == "F" ~ "Girl Names",
      TRUE ~ "Boy Names"
  )) %>%
  group_by(sex) %>%
  top_n(10, n)
```

year	sex	name	n	prop
2015	Boy Names	Noah	19.511	0.010
2015	Boy Names	Liam	18.281	0.009
2015	Boy Names	Mason	16.535	0.008
2015	Boy Names	Jacob	15.816	0.008
2015	Girl Names	Emma	20.355	0.011
2015	Girl Names	Olivia	19.553	0.010
2015	Girl Names	Sophia	17.327	0.009
2015	Girl Names	Ava	16.286	0.008

Most popular baby names in 2015

```
g_babynames <- ggplot(babynames_pop2015) +
  aes(y = n, x = name) +
  geom_col()
```

Most popular baby names in 2015

```
g_babynames <- ggplot(babynames_pop2015) +
  aes(y = n, x = name) +
  geom_col() +
  coord_flip()
```

Most popular baby names in 2015

```
g_babynames <- ggplot(babynames_pop2015) +
  aes(y = n, x = fct_reorder(name, n)) +
  geom_col() +
  coord_flip()
```

📦 `fct_reorder` comes from the tidyverse package `forecats`

Most popular baby names in 2015

```
g_babynames <- ggplot(babynames_pop2015) +
  aes(y = n, x = fct_reorder(name, n), fill = sex) +
  geom_col() +
  coord_flip()
```

Most popular baby names in 2015

```
g_babynames <- ggplot(babynames_pop2015) +
  aes(y = n, x = fct_reorder(name, n), fill = sex) +
  geom_col() +
  coord_flip() +
  facet_wrap( ~ sex, scales = 'free_y')
```

Most popular baby names in 2015

```
g_babynames <- ggplot(babynames_pop2015) +
  aes(y = n, x = fct_reorder(name, n), fill = sex) +
  geom_col() +
  geom_text(
    aes(label = format(n*1000, big.mark = ',')),
    size = 9, hjust = 1.1,
    color = 'white', family = 'Fira Sans'
  ) +
  coord_flip() +
  facet_wrap( ~ sex, scales = 'free_y')
```

Most popular baby names in 2015

```
g_babynames +
  labs(x = '',
       y = 'Number of Babies Born in 2015 (thousands)') +
  guides(fill = FALSE) +
  scale_fill_manual(
    values = c("Boy Names"  = "#77cbb9",
               "Girl Names" = "#a077cb")) +
  theme(
    strip.text = element_text(face = 'bold', size = 20),
    strip.background = element_blank(),
    text = element_text(size = 24)
  )
```

Most popular baby names in 2015

Gender-bending baby names

Find babynames that were

1. More "boyish" or "girlish" in pre-1900s and opposite in post-1900s
2. Pick top 10 boy ↔ girl names

Gender-bending baby names

Find babynames that were

1. More "boyish" or "girlish" in pre-1900s and opposite in post-1900s

2. Pick top 10 boy ↔ girl names

Boy → Girl Names:
Madison, Ashley, Alexis, Lauren, Taylor, Addison, Sydney, Allison, Morgan, Aubrey

Girl → Boy Names:
Ollie, Jean, Lou, Cruz, Frankie, Alpha, Artie, Vinnie, Donnie, Lue

Gender-bending baby names

Data-preprocessing:

1. Un-tidy `sex` column into `Female` and `Male`
2. Calculate difference in proportion by name
3. Add groups for area plot (thank you stackoverflow!)

Check out `babynames-prep.R` in repo

year	name	prop	prop_group
1884	Taylor	-0.00017	1
1901	Jean	0.00103	1
1917	Frankie	0.00027	1
1937	Allison	-0.00002	1
1967	Frankie	-0.00024	2
1981	Donnie	-0.00024	2
1984	Allison	0.00334	4
1994	Ollie	-0.00001	4
2010	Addison	0.00518	2
2014	Morgan	0.00134	2

Gender-bending baby names

```
ggplot(sel_change_babynames) +
  aes(x = year, y = prop)
```

Gender-bending baby names

```
ggplot(sel_change_babynames) +
  aes(x = year, y = prop) +
  geom_line(color = "grey50", aes(group=name))
```

Gender-bending baby names

```
ggplot(sel_change_babynames) +
  aes(x = year, y = prop, fill = prop > 0) +
  geom_area(aes(group = prop_group)) +
  geom_line(color = "grey50", aes(group=name))+
  facet_wrap(~ name, scales = 'free_y', ncol = 5)
```

Gender-bending baby names

```r
g_bnc <- ggplot(sel_change_babynames) +
  aes(x = year, y = prop, fill = prop > 0) +
  geom_area(aes(group = prop_group)) +
  geom_line(color = "grey50", aes(group=name))+
  facet_wrap(~ name, scales = 'free_y', ncol = 5) +
  scale_fill_manual(values = c("#6ec4db", "#fa7c92")) +
  guides(fill = FALSE) +
  labs(x = '', y = '')

g_bnc
```

Gender-bending baby names

```
g_bnc <- g_bnc +
  theme_minimal(base_family = 'Palatino') +
  theme(
    axis.text.y = element_blank(),
    strip.text = element_text(size = 18, face = 'bold'),
    panel.grid.major.y = element_blank(),
    panel.grid.minor.y = element_blank(),
    panel.grid.minor.x = element_blank(),
    panel.grid.major.x = element_line(color = "grey80", linetype = 3))
```

Gender-bending baby names

g is for Goodbye

Stack Exchange is Awesome

Stack Exchange is Awesome

1 Answer

active oldest votes

7

You need to make a new grouping variable for each positive/negative segment. To make the transitions less "blocky", you can just first interpolate the data:

```
require(ggplot2)

# Load data
df = read.table('data.txt', header=T)
df$created = as.POSIXct(df$created, tz='UTC')

# Interpolate data
lin_interp = function(x, y, length.out=100) {
    approx(x, y, xout=seq(min(x), max(x), length.out=length.out))$y
}
created.interp = lin_interp(df$created, df$created)
created.interp = as.POSIXct(created.interp, origin='1970-01-01', tz='UTC')
score.interp   = lin_interp(df$created, df$score)
df.interp = data.frame(created=created.interp, score=score.interp)

# Make a grouping variable for each pos/neg segment
cat.rle = rle(df.interp$score < 0)
```

ggplot2 Extensions: ggplot2-exts.org

ggplot2 and beyond

Learn more

- **ggplot2 docs:** http://ggplot2.tidyverse.org/
- **R4DS - Data visualization:** http://r4ds.had.co.nz/data-visualisation.html
- **Hadley Wickham's ggplot2 book:** https://www.amazon.com/dp/0387981403/

Noteworthy RStudio Add-Ins

- ggplotThemeAssist: Customize your ggplot theme interactively
- ggedit: Layer, scale, and theme editing

Practice and Review

Fun Datasets

- `fivethirtyeight`
- `nycflights`
- `ggplot2movies`
- `population` and `who` in `tidyr`

Review

- Slides and code on GitHub: http://github.com/gadenbuie/trug-ggplot2

Thanks!

@grrrck
github.com/gadenbuie
Garrick Aden-Buie