User manual for ggsubplot

Garrett Grolemund

September 3, 2012

1 Introduction

ggsubplot expands the ggplot2 package to help users create multi-level plots, or "embedded plots." Embedded plots embed subplots into a larger graph. ggsubplot is built on the observation that a subplot can be a ggplot geom.

2 New geoms

ggsubplot introduces two new geoms for making embedded plots: geom_subplot and geom_subplot2d. Both create a layer that uses subplots as a geom. geom_subplot uses a grouping variable to divide data into subplots. geom_subplot2d performs 2d binning on the xy field and represents each bin with a subplot. ggsubplot also provides geom_star and geom_coxcomb, two new geoms that are helpful for creating polar glyphs in an embedded plot.

2.1 geom_subplot

geom_subplot can be used like other geoms in ggplot2 function calls. The syntax of geom_-subplot also parallels ggplot2 syntax, but geom_subplot requires a new aesthetic, subplot, and recognizes additional arguments. The subplot aesthetic should be set to a complete ggplot2 layer call (e.g, layer(...), geom_point(...), stat_bin(...), etc.). The layer call assigned to the subplot aesthetic should not include a data set, as each subplot will inherit part of the data set assigned to the geom_subset layer.

```
## geom_point(aes(surftemp, temperature)) makes
## a complete layer
ggplot(nasa) +
  geom_point(aes(surftemp, temperature))

## ...so it can be used as a subplot aesthetic
## geom_subplot requires x, y, group, and subplot aesthetics
```

Figure 1: The nasa data presents meterological data collected at 576 separate sites evenly spaced on a grid. We can compare all the observations together in a single scatterplot (left), or we can group observations into separate scatterplots - one for each measurement site, laid out on the map according to the locations of the measurement sites (right).

```r
ggplot(nasa) +
  map_americas +
  geom_subplot(aes(long, lat, group = id,
    subplot = geom_point(aes(surftemp, temperature), size = 1/4))) +
  coord_map()
```

`geom_subplot` attempts to pick a pretty size for the subplots based on the spacing of the subplots in the final graph. This size can be overwriten with the `height` and `width` parameters. `height` and `width` can be set to numbers, which will be interpretted as units along the x and y axes, or to proportions of the original pretty size with the `rel()` function. For example, `width = rel(0.5)` would reduce the width of the subplots to approximately half of their original width.

2.2 `geom_subplot2d`

`geom_subplot2d` is similar to `geom_subplot`, but it does not take a group aesthetic. Instead it bins the coordinate plane into two dimensional bins and represents each bin with a subplot. The number of bins in the x and y directions can be specified with the `bins` or `binwidth` parameters, which expect vector arguments of length two. If a single number is entered for either parameter, it will be recycled to apply to both the x and y axis. Only one of `bins` or `binwidth` should be specfied as they create conflicting results.

Figure 2: Casualties in the Afghan War Diary by location (left) and the same information displayed by binning the data set according to latitude and longitude and drawing a barchart for each bin (right).

```r
## Afghan War Journal raw data
ggplot(casualties) +
  map_afghanistan +
  geom_point(aes(lon, lat, color = victim)) +
  coord_map()

## 2d bin with bar chart subplots displaying data in each region
ggplot(casualties) +
  map_afghanistan +
  geom_subplot2d(aes(lon, lat,
    subplot = geom_bar(aes(victim, ..count.., fill = victim))),
    bins = c(15,12), ref = NULL, width = rel(0.8)) +
  coord_map()

Using binwidth 0.905Using binwidth 0.754
```

subsubsectionscales By default, x and y scales are held constant across subplots in both `geom_subplot` and `geom_subplot2d`. However, scales can be allowed to vary across subplots by setting `x_scale = free` or `y_scale = free`. This behavior parallels the use of scales while faceting in ggplot2. `y_scale = free` has the effect of showing conditional distributions, `y_scale = identity` displays marginal distributions.

subsectiongeom_star and geom_coxcomb

geom_star and geom_coxcomb provide a way to include polar glyphs in cartesian embedded plots. geom_star creates a star glyph and geom_coxcomb creates a coxcomb glyph akin to the plots popularized by Florence Nightingale. Either geom can be used by itself, similar to geom_boxplot, but for individual glyphs, superior legend and axis details can be had by adding coord_polar() to a line graph or a histogram.

```
## A single star
one_nasa <- nasa[nasa$id == "1-1", ]
ggplot(one_nasa) +
  geom_star(aes(x = 0, y = 0, r = surftemp, angle = date,
    fill = mean(temperature)), r.zero = FALSE)

## r.zero determines whether the origin of the star should
## correspond to r = 0? If FALSE, origin corresponds to the
## lowest value of r.

## Stars in an embedded plot
ggplot(nasa) +
  map_americas +
  geom_subplot(aes(long, lat, group = id,
    subplot = geom_star(aes(x = 0, y = 0, r = surftemp,
    angle = date, fill = mean(surftemp)), r.zero = FALSE))) +
  coord_map()
```

```
## A single coxcomb
ggplot(casualties) +
  geom_coxcomb(aes(angle = month, fill = month))

## Coxcombs in an embedded plot
ggplot(casualties) +
  map_afghanistan +
  geom_subplot2d(aes(lon, lat,
    subplot = geom_coxcomb(aes(angle = month, fill = month))),
    bins = c(15, 12), ref = NULL) +
  coord_map()

Using binwidth 0.905Using binwidth 0.754
```

Figure 3:

Figure 4:

3 Reference objects

ggsubplot does not have the ability to draw a separate axis for each subplot. Moreover, such axes would be hard to read. Instead, subplots can be drawn with reference objects, such as a bounding box, that facilitate comparisons across subplots. Reference objects can be added to subplots by setting the `ref` parameter in `geom_subplot` and `geom_subplot2d`. The `ref` parameter takes the output of `ref_box`, which creates bounding boxes, `ref_hline`, which creates horizontal lines, or `ref_vline`, which creates vertical lines, see Figure 5. `ref` can also be set to `NULL` to omit reference objects from the plot. Reference objects are ggplot2 geoms with aesthetics such as fill, color, and transparency. These aesthetics can be set with the mapping parameter in the `ref_...` functions. `ref_hline` and `ref_vline` also recognize a `thickness` parameter which determines how thick the lines should be as a fraction of the width or height of the entire subplot.

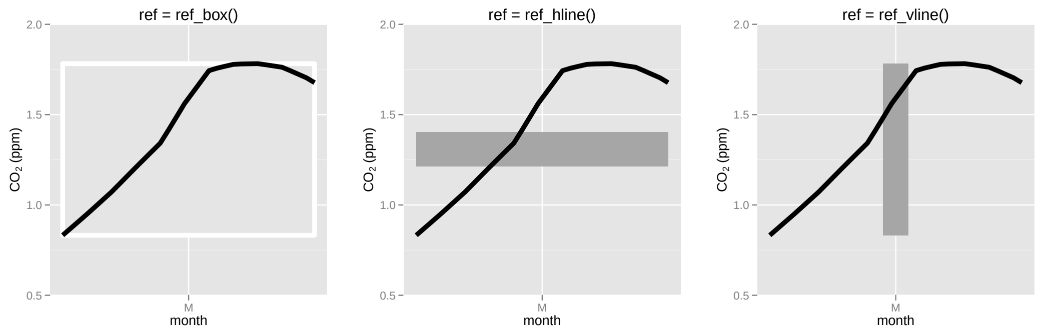

Figure 5: `ref = ref_box()` creates bounding boxes for each subplot (left). `ref = ref_hline()` creates horizontal lines (middle). `ref = ref_vline()` creates vertical lines (right).

```r
## Using reference object aesthetics to display additional information
ggplot(casualties) +
  map_afghanistan +
  geom_subplot2d(aes(lon, lat,
    subplot = geom_freqpoly(aes(date), binwidth = 13151560)),
    bins = c(18, 14),
    ref = ref_box(aes(fill = length(date)), alpha =0.75)) +
  coord_map()
## length(date) is a proxy for how many casualties
## occured in each region
```

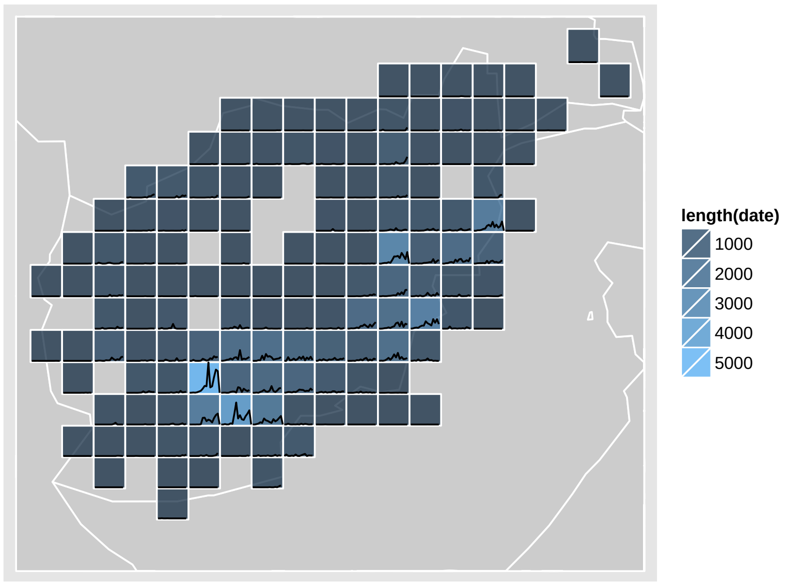

Figure 6: Frequency of casualties by region 2004-2010. Total casualties are indicated by the background fill of each subplot.

4 Using embedded plots

The best graphics tell a clear story, and for this reason users should pause before considering an embedded plot. Embedded plots contain more information than a normal plot and more dimensions that must be understood before the graph can be comprehended. Embedded plots also reduce the size at which many details are shown. This makes it easier to perceive overall patterns, but difficult to see precise details. Users should choose a simpler graph over an embedded plot whenever a simpler plot will suffice. Users should also plot subplots by themselves, at full scale, whenever details need to be measured from the graph accurately.

However, simple plots sometimes fail to tell a story clearly and many stories cannot be told at all with a simple plot. In these scenarios, embedded plots provide a useful tool. Embedded plots can make patterns clear where simpler plots suffer from overplotting, Figure 2. An embedded plot will not suffer from overplotting even when the data set contains millions, billions, or trillions of points. Embedded plots are also particularly useful for exploring spatio-temporal data, Figure 3. The latitude, longitude, and time dimensions of spatio-temporal data consume three dimensions within a plot, which leaves little graphical power left over to observe variables of interest. The extra axes provided by subplots relieve this problem. These dimensions also make it easier to explore complicated, multi-dimensional relationships, such as interaction effects.

Embedded plots have the capacity to display large amounts of information and they organize this information in an admirably intuitive way. When used wisely, embedded plots can reward a thoughtful viewer with insights that cannot not be perceived clearly or at all in simpler plots.