# A Minimal Demo of `glyphmaps`

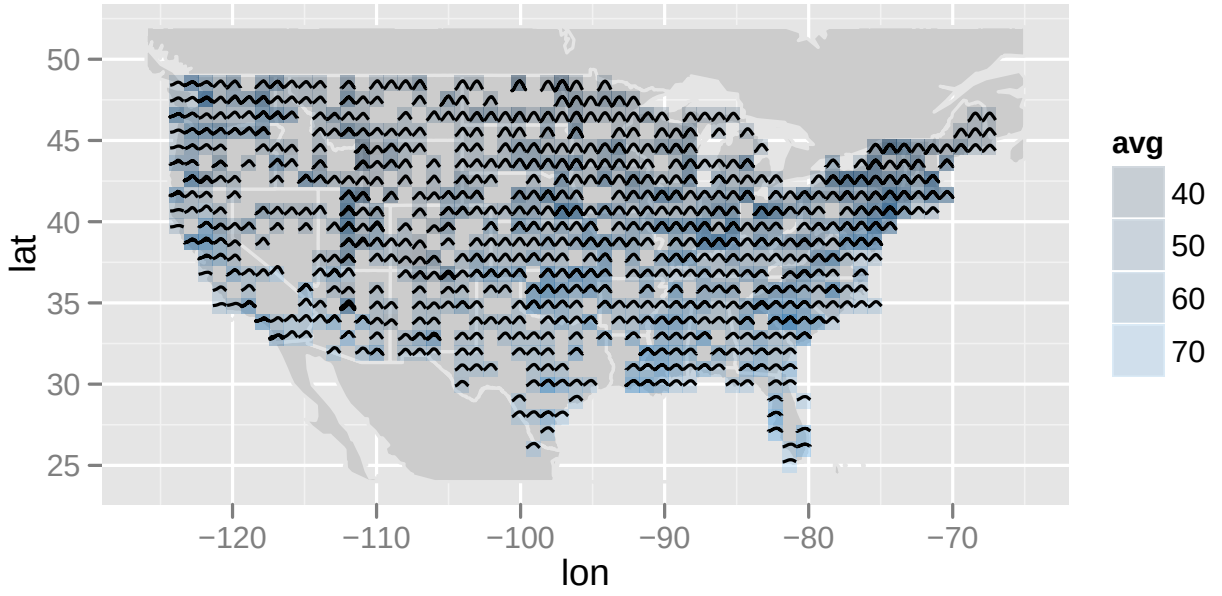
Garrett Grolemund

June 4, 2012

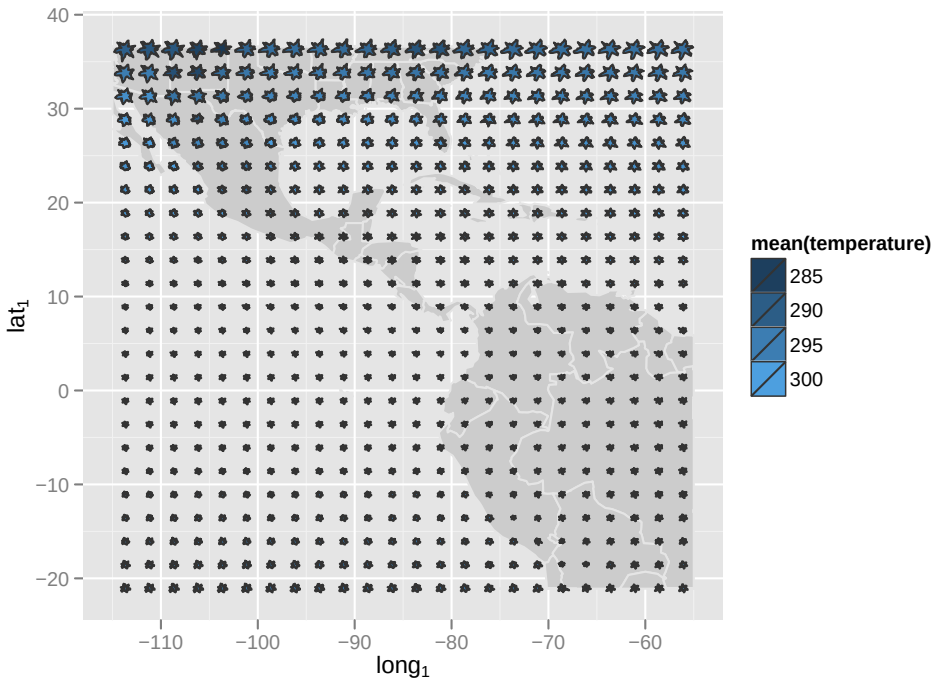
This document demonstrates the basic features of the `glyphmaps` package (name may change). `glyphmaps` was written to create plots like the ones below, which recreate plots similar to those in Wickham (2011) and Wickham et al. (Submitted). Both plots were made with `glyphmaps`. To run the examples in this document please download the `glyphmaps` package with the code below.

```
library(devtools)
install_github("ggplyr", "garrettgman", "glyphmaps")
library(glyphmaps)
```

```
ggplot(seasons) + map_us +
  grid(geom_line(aes(x = time, y = pred)),
    grid.aes = aes(lon, lat), x.nbin = 58, y.nbin = 25,
    ref = ref_box(aes(fill = avg), alpha = 0.2, color = NA))
```




```
ggplot(nasa) +  map_nasa +
  glyph(geom_star(aes(r = ozone, angle = date, x = 0, y = 0, fill = mean(temperature))),
  major.aes = aes(long[1], lat[1]), glyph.by = c("long", "lat"))
```




`glyphmaps` helps users create embedded graphics: plots that have subplots built into them. The package is built around three main functions: `grid`, `glyph`, and `ply_aes`. This demo will demonstrate these three functions, their options, and the various uses of embedded graphics.

# 1 Glyphs

Glyphs are geometric objects (i.e, geoms) that display information within the geom. In other words, a glyph can display information even if it is drawn by itself, without references to an external coordinate system. In reality, all geoms are a type of glyph, but the term glyph is usually reserved for complicated geoms, such as those that contain their own internal coordinate systems. The star glyphs in Figure 1 illustrate how glyphs can contain an internal (minor) coordinate system and can still be plotted in an external (major) coordinate system.

Glyphs provide one connection between the grammar and graphics and embedded plots. Every glyph is an embedded plot and each embedded plot is a type of glyph. Hence, we can treat embedded plots as a type of geom. Embedded plots can also be thought of as facetting on a continuous scale. However, since embedded plots are often used with other non-facetted layers, it is easier to implement embedded plots as a type of geom.

## 1.1 glyph

`glyph` provides a general method for creating glyph geoms in `ggplot2`. `glyph` works in conjunction with `ggplot2` layer objects in the following way. Each layer specifies a type of graph through a combination of geom, stat, aesthetics, parameters, and a position adjustment. `glyph` uses this specification to build the individual glyphs (i.e, `glyph` builds glyphs that match the plot specified by the layer). `glyph` will plot a separate glyph for different subsets of the layer's data. Users specify how to split the data into subsets with the `glyph.by` argument. This argument behaves similarly to the `.variables` argument in the `plyr` package (Wickham, 2011). Finally, `glyph` plots each glyph on the x and y axis specified by the `major.aes` argument. These "major axiis" are mapped to the data just as other `ggplot2` aesthetics are mapped to the data. Hence they should be specified with the `aes` function.

To summarize, a layer can be turned into a set of glyphs with the glyph function and three arguments:

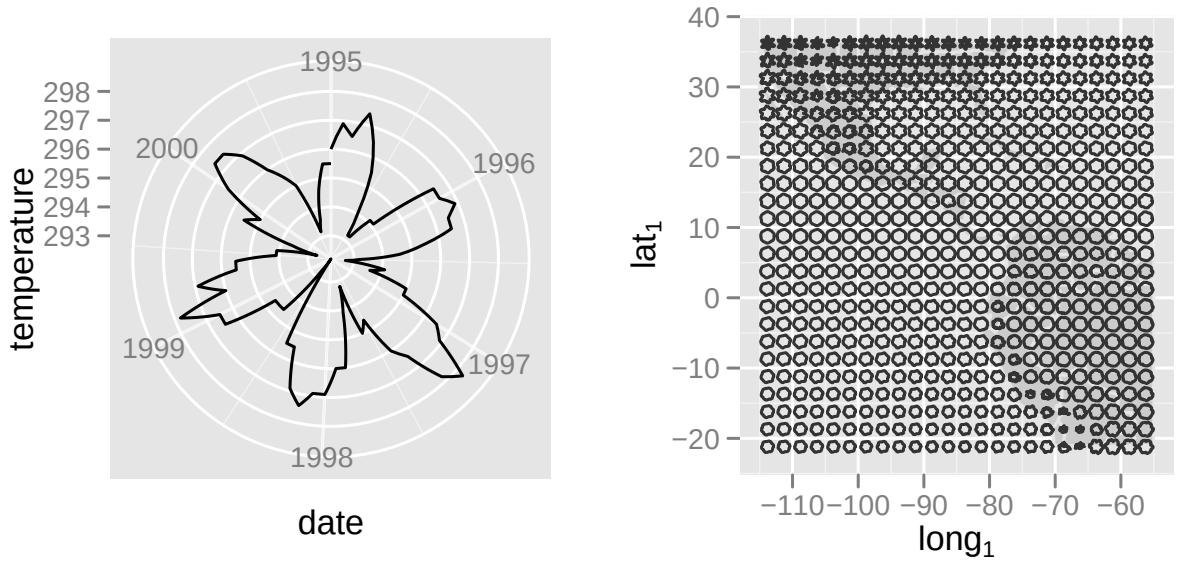
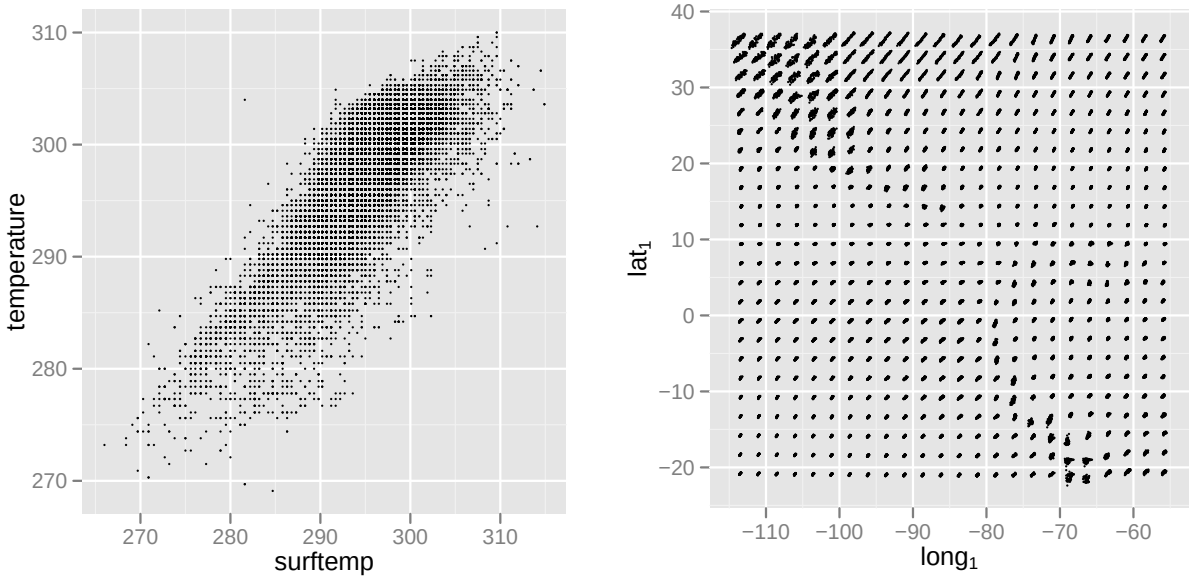



Figure 1: 'Individual star glyphs comprise a complete polar plot with an internal coordinate system (r = temperature and angle = date). Multiple star glyphs can be organized with an external coordinate sytem to reveal second order effects.'

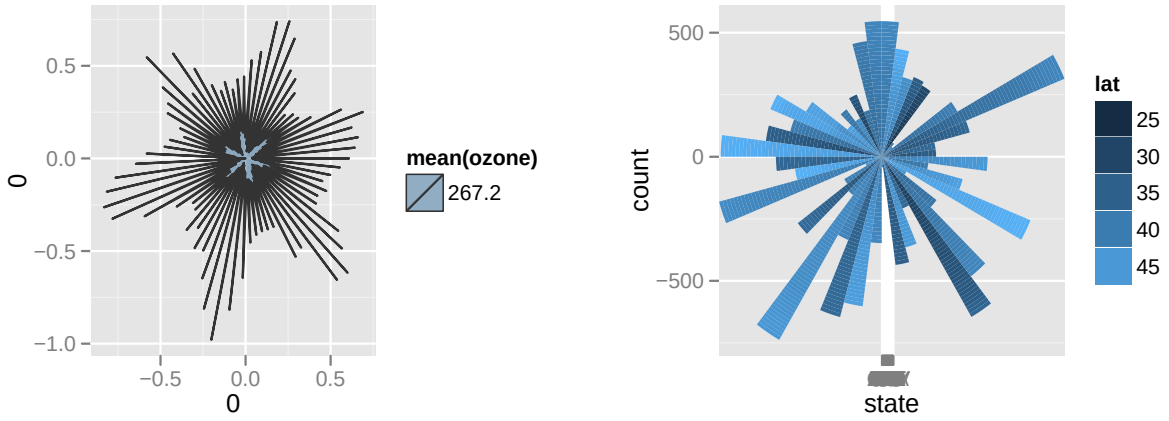
```
points <- geom_point(aes(x = surftemp, y = temperature), size = 1/2)
## A normal plot
ggplot(nasa) + points
## Glyphed plot
ggplot(nasa) + glyph(layer = points, major = aes(x = long[1], y = lat[1]),
    glyph.by = c("lat", "long"))
```




glyph gives users incredible freedom to design their own types of glyphs. Any graph that can be specified with a single layer in ggplot2 can be turned into a glyph. glyphmaps also provides two new geoms that can be used to create frequently seen glyphs. These are geom_star and geom_coxcomb, see

Figure **??**. Although coxcomb glyphs don't seem much better than Venn pie-agrams (which I suppose we could also do with glyphmaps).

```
ggplot(nasa) + geom_star(aes(r = ozone, angle = date, x = 0, y = 0,
    fill = mean(ozone)), alpha = 0.5)
ggplot(seasons) + geom_coxcomb(aes(angle = state, fill = lat))
```

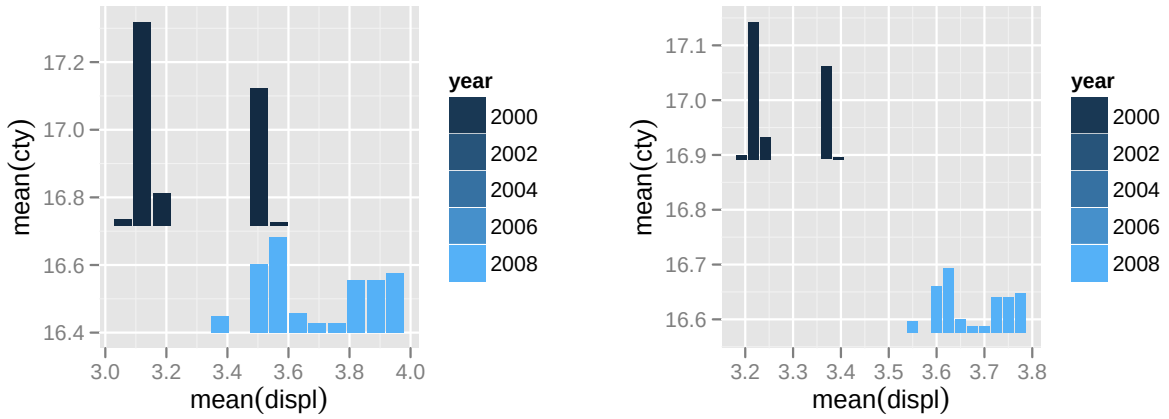



`glyph` also has a variety of optional arguments that allow users to customize glyph plots. These allow the user to control the dimensions of each glyph, to control the scale within each glyph, to add reference objects to each glyph, and to merge glyphs that overlap into a single glyph.

### width, height

The `width` and `height` arguments of `glyph` control the width and height of each glyph in the plot.

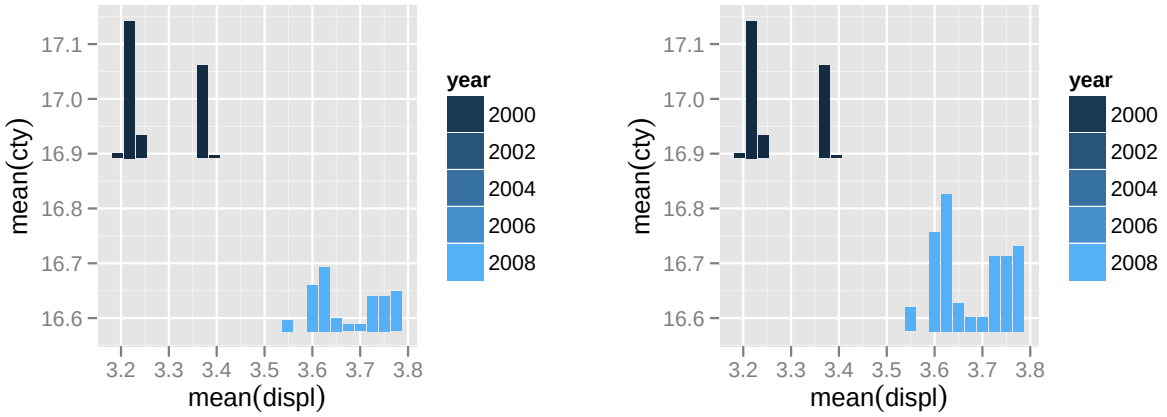
```
ggplot(mpg) + glyph(geom_bar(aes(x = trans, fill = year)),
  aes(x = mean(displ), y = mean(cty)), c("year"))
ggplot(mpg) + glyph(geom_bar(aes(x = trans, fill = year)),
  aes(x = mean(displ), y = mean(cty)), c("year"),
  width = 1/4, height = 1/4)
```




### x_scale, y_scale

The `x_scale` and `y_scale` arguments of `glyph` control the scaling within each glyph. As with facets, glyphs share the same scaling by default. The x and y scales can also be set to 'free'.

```
## glyphs share y scale
ggplot(mpg) + glyph(geom_bar(aes(x = trans, fill = year)),
  aes(x = mean(displ), y = mean(cty)), c("year"),
  width = 1/4, height = 1/4)
## independent y scales
ggplot(mpg) + glyph(geom_bar(aes(x = trans, fill = year)),
  aes(x = mean(displ), y = mean(cty)), c("year"),
  width = 1/4, height = 1/4, y_scale = free)
```




**reference objects**

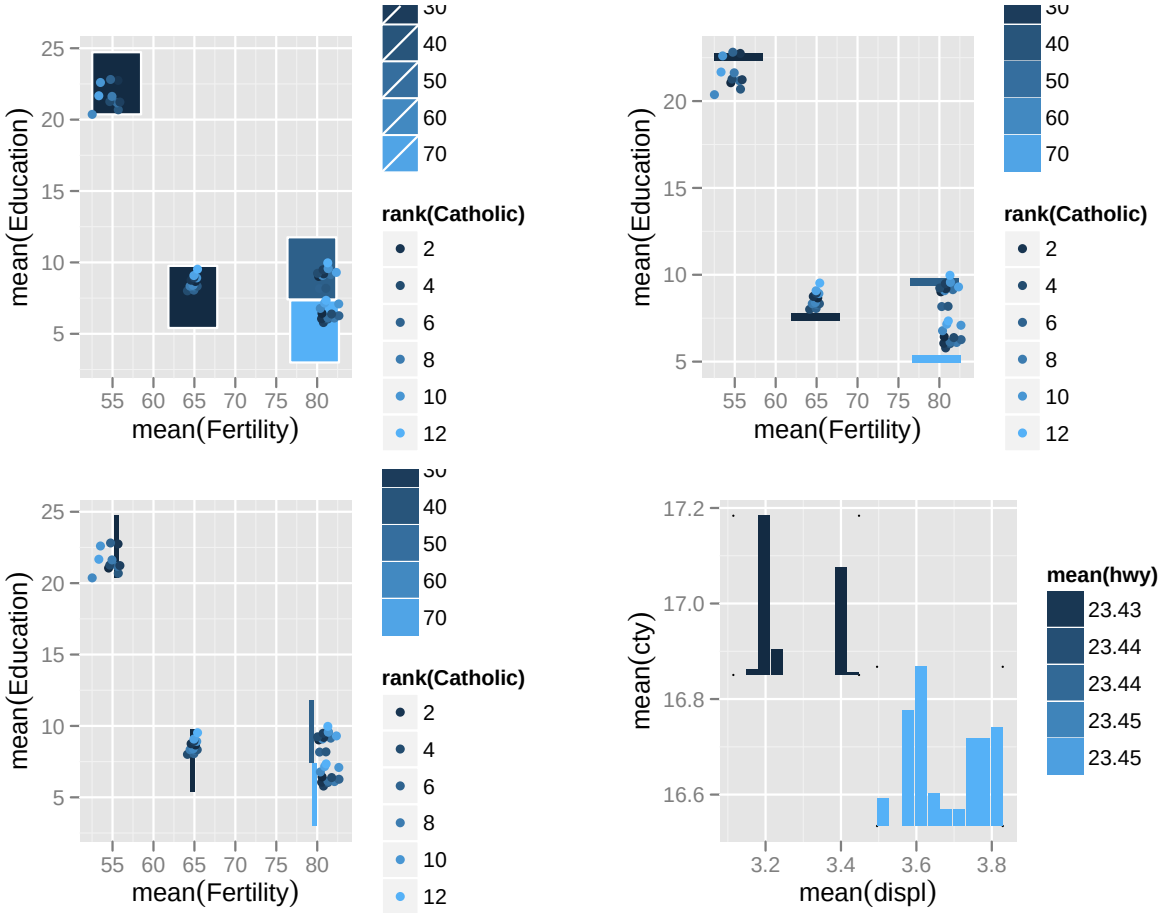
Users can set the `reference` argument to include reference objects with each glyph. These objects provide a frame of reference which facilitates comparing between glyphs. The `reference` argument should be set to one of `ref_box`, `ref_hline`, `ref_vline`, `ref_points`, or `NULL`. Each of these functions creates a different type of reference object, see Figure **??**. Each also accepts a mapping and parameters to customize the appearance of the reference objects.

```
# boxes
ggplot(test.data) + glyph(geom_point(aes(Fertility, Agriculture,
  color = rank(Catholic))), glyph.by = c("lat", "long"),
  major = aes(mean(Fertility), mean(Education)), ref = ref_box(aes(fill =
  mean(Catholic))))

# hlines
ggplot(test.data) + glyph(geom_point(aes(Fertility, Agriculture,
  color = rank(Catholic))), glyph.by = c("lat", "long"),
  major = aes(mean(Fertility), mean(Education)), ref = ref_hline(aes(fill =
  mean(Catholic))))

# vlines
ggplot(test.data) + glyph(geom_point(aes(Fertility, Agriculture,
  color = rank(Catholic))), glyph.by = c("lat", "long"),
  major = aes(mean(Fertility), mean(Education)), ref = ref_vline(aes(fill =
  mean(Catholic))))

# points
ggplot(mpg) + glyph(geom_bar(aes(x = trans, fill = year)),
  aes(x = mean(displ), y = mean(cty)), c("year"), y_scale = free,
  width = 1/3, height = 1/3, reference = ref_points(aes(fill = mean(hwy))))
```



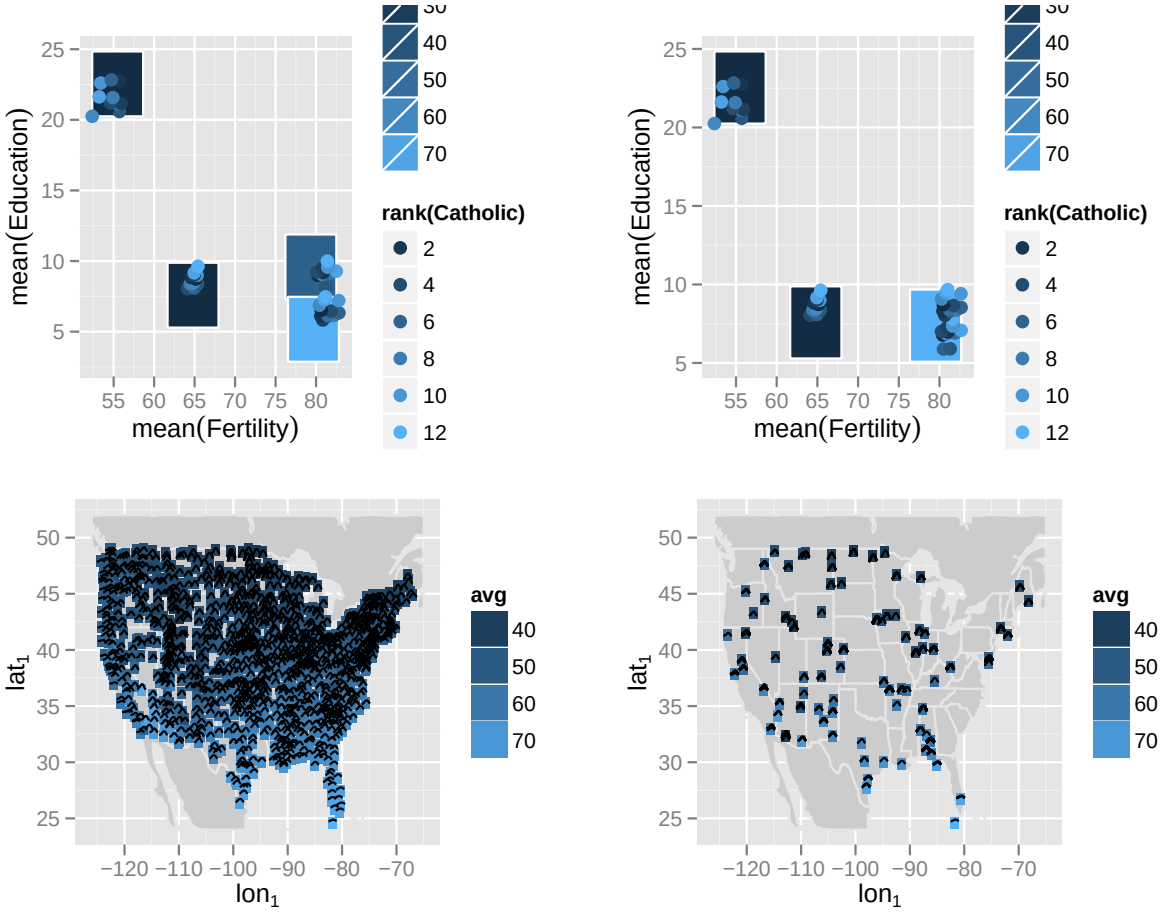

## 1.2 merge overlaps

Often two or more glyphs will overlap each other when plotted. It is possible to combine such overlapping glyphs into a single glyph by setting the `merge.overlaps` argument to `TRUE`. `glyphmaps` will then screen the glyph output for sets of overlapping graphs. Each set will be combined into one glyph and plotted at the at the average location of the combined glyphs, see Figure **??**. Caution should be used when setting `merge.overlaps` to `TRUE`. Merge overlaps currently works *exactly* as described, which means that for very dense graphs, `merge.overlaps` is likely to thin out the plot a bit more than the user expects.

```
## no merging
ggplot(test.data) + glyph(geom_point(aes(Fertility, Agriculture,
  color = rank(Catholic)), size = 3), glyph.by = c("lat", "long"),
  major = aes(mean(Fertility), mean(Education)), ref = ref_box(aes(fill =
  mean(Catholic))), width = rel(1), height = rel(1))
## merging
ggplot(test.data) + glyph(geom_point(aes(Fertility, Agriculture,
  color = rank(Catholic)), size = 3), glyph.by = c("lat", "long"),
  major = aes(mean(Fertility), mean(Education)), ref = ref_box(aes(fill =
  mean(Catholic))), merge = TRUE, width = rel(1), height = rel(1))

## compare with dense situations
ggplot(seasons) + map_us +
  glyph(geom_line(aes(x = time, y = pred)),
    aes(lon[1], lat[1]), c("stn"), height = 1, width = 1.5,
```

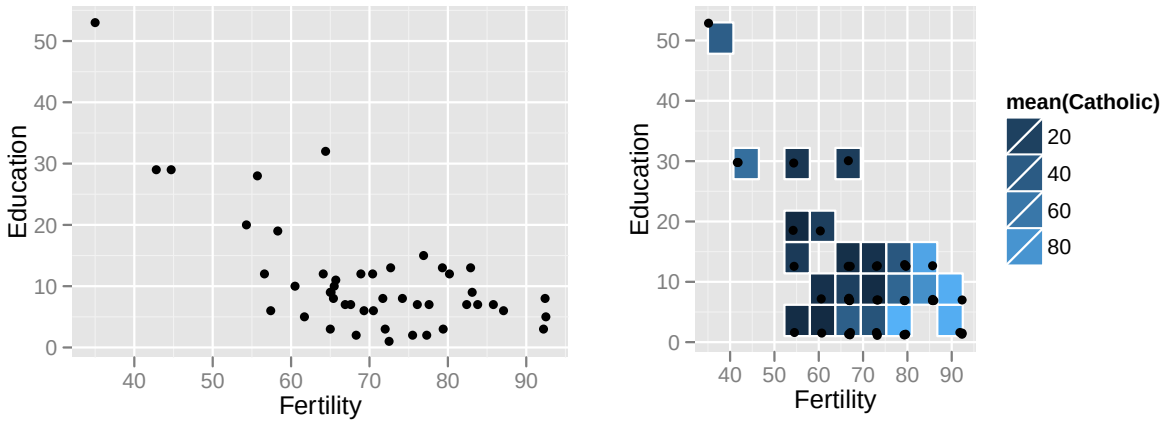
```
      ref = ref_box(aes(fill = avg), color = NA))
## vs
ggplot(seasons) + map_us +
  glyph(geom_line(aes(x = time, y = pred)),
    aes(lon[1], lat[1]), c("stn"), height = 1, width = 1.5,
      ref = ref_box(aes(fill = avg), color = NA), merge = TRUE)
```




## 1.3 grid

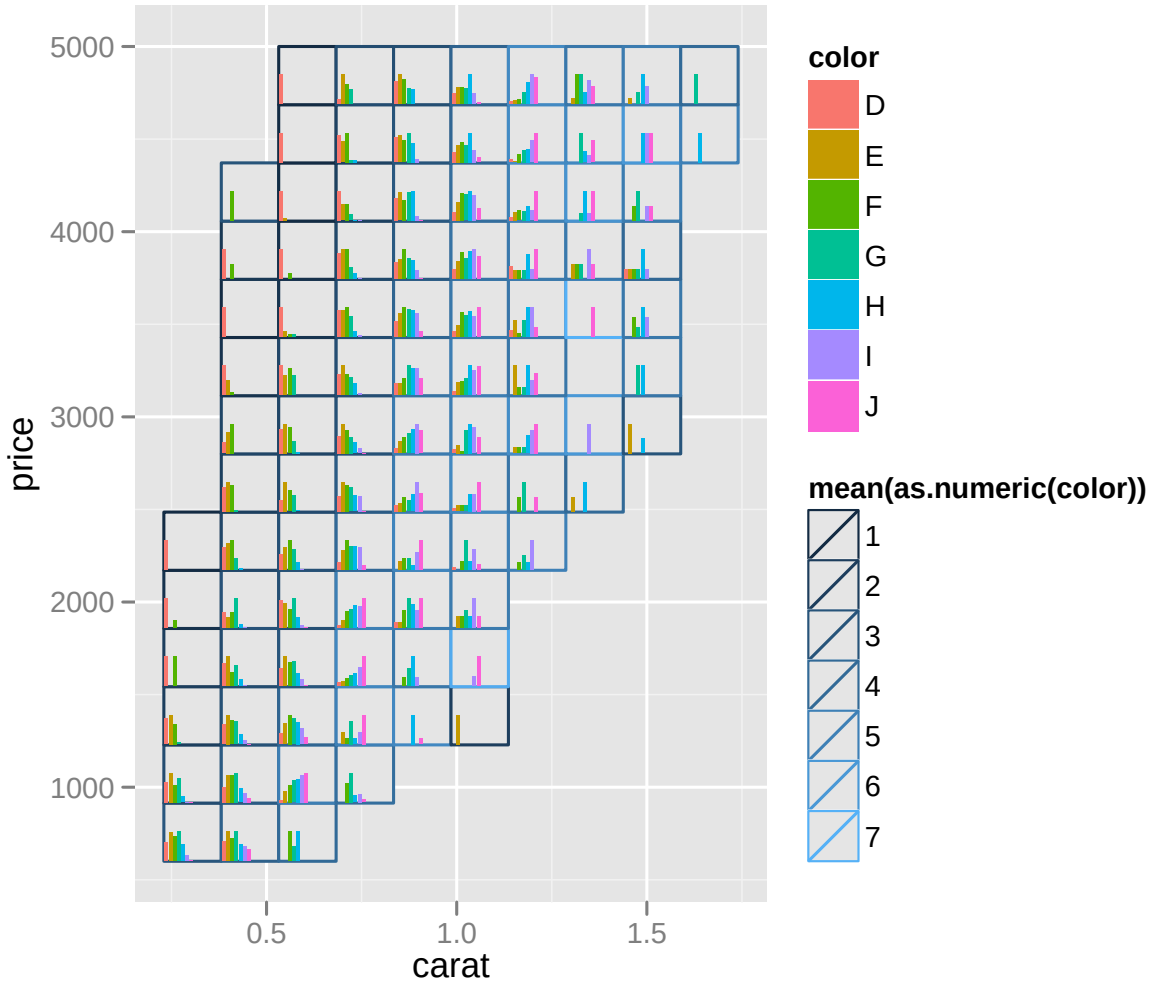
Gridding is the simplest way to embed subplots in a graph. `grid` basically bins a layer into a 2d grid of subplots and plots a glyph at every bin location. The syntax of `grid` is very similar to that of `glyph`. However, `grid` does not take `width`, `height`, or `merge.overlaps` arguments. the dimension of the glyphs (and the lack of overlaps) is implicit in the intention to grid a layer. Instead, `grid` takes two new arguments" `x.nbin` and `y.nbin`. These specify the number of bins to use on the x and y axiis. They are set to ten by default. Like `glyph`, `grid` can take reference and scale arguments.

```
## without gridding
ggplot(test.data) + geom_point(aes(Fertility, Education))
## with gridding
ggplot(test.data) +
  grid(geom_point(aes(Fertility, Education)), x.nbin = 10, y.nbin = 10,
       ref= ref_box(aes(fill = mean(Catholic))))
```




Gridding provides a new way to approach the problem of overplotting. A grid of glyphs can work like a bin2d but provides more information. Figure **??** demonstrates this approach by recreating an overplotting graph that has garnered recent interest on the ggplot2 mailing list.

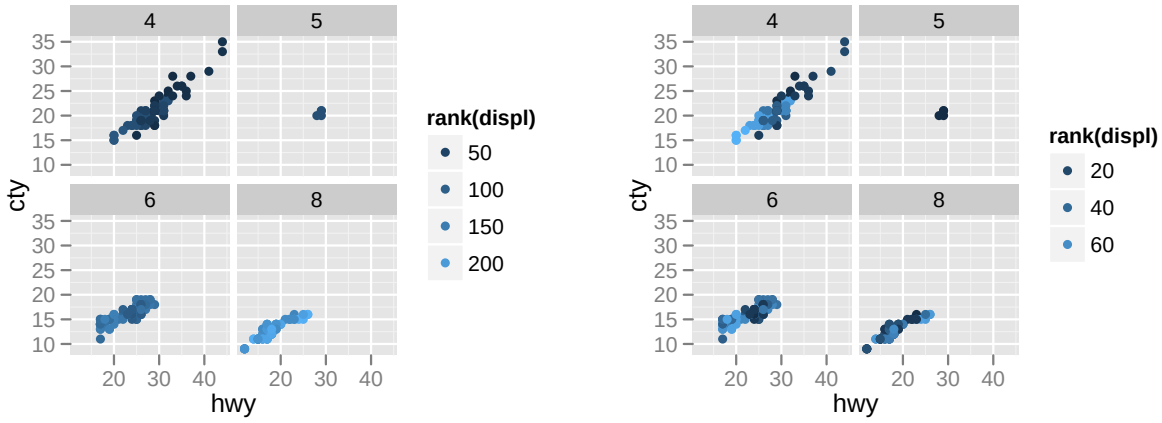
```
## without gridding
cheap.diamonds <- subset(diamonds, price <= 5000 & price >= 600)
ggplot(cheap.diamonds) +
  grid(geom_bar(aes(x = color, fill = color), position = "dodge"),
    grid.aes = aes(x = carat, y = price), x.nbin = 10, y.nbin = 14,
    y_scale = free, height.adjust = 0.5, width.adjust = 0.5,
    ref = ref_box(aes(color = mean(as.numeric(color)))))
```




## 1.4 ply_aes

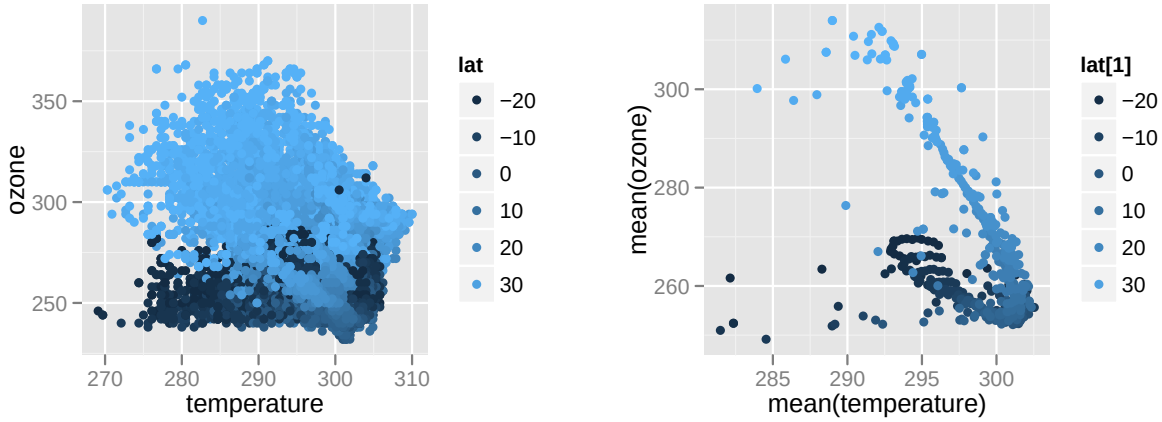
`ply_aes` wraps a layer like `grid` or `glyph` and causes the aesthetics to be calculated in a groupwise manner. Users can specify the groups to use, or `ply_aes` will default to any groupings provided by glyph.by or group.by parameters, or a group aesthetic.

```
## normal \texttt{ggplot2} behavior
ggplot(mpg) + geom_point(aes(hwy, cty, color = rank(displ))) +
  facet_wrap(~cyl)
## groupwise aesthetics
ggplot(mpg) + ply_aes(geom_point(aes(hwy, cty, color = rank(displ))),
  c("cyl")) + facet_wrap(~cyl)
```




`ply_aes` doesn't actually create embedded plots, but is closely related to the glyphing process. By default, `ggplot2` does not calculate statistics in a groupwise manner. However, such calculations can be useful in certain situations. For example, `ply_aes` is built into `glyph` and `grid` by default; embedded plots quickly become boring when aesthetics do not vary from subplot to subplot. Groupwise aesthetics can be interesting in other situations as well. For example, Figure **??** demonstrates how groupwise aesthetics can be used to avoid overplotting.

```
## overplotted
ggplot() + geom_point(aes(x = temperature, y = ozone, color = lat), data = nasa)
# using ply_aes to plot just summaries
ggplot() + ply_aes(geom_point(aes(x = mean(temperature), y = mean(ozone),
  color = lat[1]), data = nasa), c("lat", "long"))
```




## 2 speed considerations

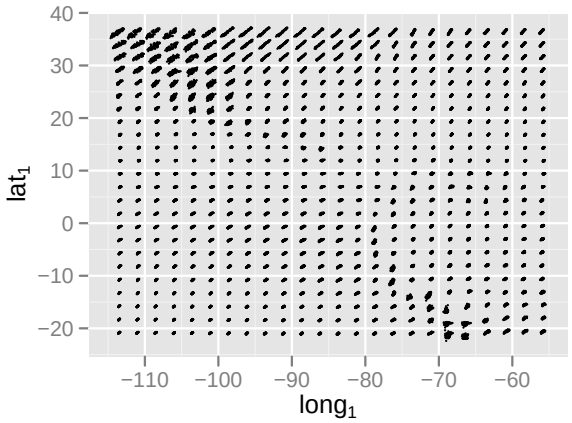
Embedding plots is computationally expensive. Such plots require much of both the computational engine of R and the graphics device. Hence, it is not unreasonable to expect `glyphmaps` to be slow, which it can be. However, relative to similar visual tasks `glyphmaps` is surprisingly fast. The final code chunk compares the speed of plotting forty thousand points in an embedded plot to plotting the same points in a facetted plot. `glyphmaps` performs quite well in comparison.

```
## embedded plot
system.time(print(ggplot(nasa) + glyph(geom_point(aes(x = surftemp,
  y = temperature), size = 1/2), aes(x = long[1], y = lat[1]), c("lat", "long"))))
```

```
##    user  system elapsed
##  12.460   2.282  15.981


## facetted plot
system.time(print(ggplot(nasa) + glyph(geom_point(aes(x = surftemp,
  y = temperature), size = 1/2) + facet_grid(long~lat))))

## Error: argument "major.aes" is missing, with no default
## Timing stopped at: 0.012 0.007 0.019
```




## References


Hadley Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1): 1âĂŞ29, 2011. URL `http://www.jstatsoft.org/v40/i01/`.

Hadley Wickham, Heike Hofmann, Charlotte Wickham, and Diane Cook. Glyph-maps for visually exploring temporal patterns in climate data and models. *Environmetrics*, Submitted.