STA 141A
Fundamentals of Statistical Data Science

Fall 2016

Instructor: Debashis Paul

Lecture 10

Saving graphs as objects

- Any graphical object created in **ggplot2**, including layers, can be saved like any other R object.

```
library(ggplot2)
data(mpg)  # a data set on car milage
pl.mpg = qplot(displ,hwy,data=mpg, color=factor(cyl))
summary(pl.mpg)  # gives a summary of  pl.mpg as an object
print(pl.mpg) # or, simply pl.mpg  ,  plots the graph
save(pl.mpg, "mpg_plot.Rdata")  # you may specify the folder where you want to save
load("mpg_plot.Rdata")
```

Layered graphics : diamonds data

```
data(diamonds)
pl.diam = ggplot(diamonds, aes(carat,price, color=cut))
# x = carat, y = price, color = cut (not treated as categorical when using statistical transformations)
# layers, performing specific statistical/graphical summary can be stored as R objects
bestscatter = geom_point(alpha=I(1/2), shape='.', size=I(4))
# controls appearance with specification of transparency, shape and size of the points, apart from color
bestfit = geom_smooth(method="lm", se=F, color=alpha("green",0.5), size=2)
# linear regression fit; se=T will add standard error bands, color and size control the fitted line
pl.diam + bestscatter + bestfit  + scale_x_continuous(limits=c(0,4.5)) +
        scale_y_continuous(limits=c(0,25000))
# plots the points and adds regression fit of price on carat; also changes the scales of x and y axes
```

Faceting and transformation

```
# plot the scatter plots corresponding different values of cut
pl.diam + bestscatter + facet_grid(cut ~ .)
# also fit regression line to each stratum
pl.diam + bestscatter + facet_grid(cut ~ .) + bestfit
# allow the scales of the different panels to be different;
# default: scale = "free"; other options "free_x", "free_y"
pl.diam + bestscatter + facet_grid(cut ~ .,  scale="free") + bestfit
# perform log10 transform of the variables and add least square fits; also change the labels on the axes
pl.diam + geom_point() + scale_x_log10( ) + scale_y_log10( ) +
                    xlab("log10(carat)") + ylab("log10(price)") + bestfit
```

Different types of geom

```
dfx = data.frame(x=c(3,1,5,4), y=c(2,4,6,8), label=letters[1:4])  # data frame with 4 labeled rows and 2 columns
pl.dfx = ggplot(dfx,aes(x=x,y=y,label=label)) + xlab(NULL) + ylab(NULL)
pl.dfx + geom_bar(stat="identity")  + xlab("bar")   # bar plot
pl.dfx + geom_point() + xlab("point")  # scatter plot
pl.dfx + geom_line()  + xlab("line")   # line plot (connecting points ordered by x-value)
pl.dfx + geom_area() + xlab("area")   # area plot
pl.dfx + geom_path() + xlab("path")    # path plot (connecting successive points using the order in which they appear)
pl.dfx + geom_text() + xlab("text")   # adding text (label) to points
pl.dfx + geom_tile() + xlab("tile")   # tiles around points
pl.dfx + geom_polygon() + xlab("polygon")  # polygon formed by connecting successive points
```

Displaying univariate data : histogram

- We can use histograms, density plots or boxplots for displaying univariate data, often stratified by a categorical variable.

```
depth.di = ggplot(diamonds,aes(depth)) + xlim(58,68)
# plot relative density histograms of depth stratified by different values of cut
depth.di + geom_histogram(aes(y=..density..), binwidth=0.2) + facet_grid(cut ~ .)
# plot histogram of depth with different colors representing the actual count for different values of cut
depth.di + geom_histogram(aes(fill=cut), binwidth=0.2, position = "stack")
# plot relative contribution of different levels  of cut to each bin of the histogram for depth
depth.di + geom_histogram(aes(fill=cut), binwidth=0.2, position = "fill")
# plot the histogram of depth for different values of cut separately
depth.di + geom_histogram(aes(fill=cut), binwidth=0.2, position = "dodge")
```

Displaying univariate data : boxplot

- Boxplot feature in **ggplot2** uses an y variable (categorical), which we choose to be the variable cut in this example.

```
depcut.di = ggplot(diamonds,aes(x=cut,y=depth))
depcut.di +  geom_boxplot( )  # plots boxplots for depth stratified by cut
depcut.di + aes(color=cut) + geom_boxplot( )  # use different colors for different values of  cut
depcut.di + aes(color=factor(clarity)) + geom_boxplot( ) # stratify further by categorical variable clarity
depcut.di +  aes(color=cut) + geom_boxplot( ) + facet_grid(clarity ~ .) # plots in a column of  panels
depcut.di +  aes(color=cut) + geom_boxplot( ) + facet_grid(. ~ clarity) # plots  in a row of  panels
depcut.di +  aes(color=cut) + geom_boxplot( ) + facet_wrap(~ clarity) # better arrangement of  panels
```

Overriding grouping

- We can override grouping by an existing aesthetic feature (like color), by using aesthetic group

```
# in the previous example, we can stratify according to values of clarity rather than cut
depcut.di + geom_boxplot(aes(group=clarity, color=clarity)) + scale_x_discrete("clarity",breaks=c())
# Another example involving a longitudinal data set
library(nlme)  # load package nlme
data(Oxboys) # heights of boys in Oxford from the nlme package
boysOx = ggplot(Oxboys, aes(Occasion,height))
boysOx + geom_boxplot( ) # display boxplots stratified by Occasion
# overlay the plot with height trajectories, overriding original grouping by using Subject for grouping
boysOx + geom_boxplot( ) + geom_line(aes(group=Subject),color="steelblue")
```

Displaying bivariate data : jitter plot

- We can use the “jitter” geom to mitigate the effects of overplotting

```
td = ggplot(diamonds,aes(table,depth))
td + geom_point()   # scatter plot too crowded because of too many observations
td + geom_jitter()  # jitter plot
jit = position_jitter(width=0.5)  # creates a position variable that “jitters” the points over a specified width
td + geom_jitter(position=jit)
# also add transparency to the points for better visible effect
td + geom_jitter(position=jit,color=alpha("black",1/20))
```

Displaying bivariate data : histogram

- We can use bivariate histogram with square or hexagonal binning.

```
di = ggplot(diamonds,aes(carat,price)) + xlim(1,3)

di + stat_bin2d(bins=20,na.rm=T)  # bivariate histogram using square binning 20 x 20 bins

di + stat_bin2d(bins=20,na.rm=T,color="yellow")  # adds yellow border to the bins

di + stat_bin2d(binwidth=c(0.02,200))  # bins are determined by the widths of the rectangular bins

di + stat_bin2d(binwidth=c(0.02,200), aes(fill=..density..))  # use relative frequency instead of count

di + stat_binhex(bins=20, na.rm=T) # use hexagonal binning rather than square bins

di + stat_binhex(binwidth=c(0.02,200), na.rm=T) + scale_fill_gradient(low="blue",high="red")

# changes the shape of the hexagonal bins and changes a color scale
```

Displaying bivariate data : density plot

- We can also use bivariate kernel density estimator.

```
di + geom_density2d(na.rm=T)  # contour plot of the density estimator
# contour plot of the density together with scatter plot
di + geom_point(na.rm=T) + stat_density2d(na.rm=T)
di + stat_density2d(na.rm=T,aes(fill=..level..),geom="polygon") # surface plot of the density
# surface plot of the density with added transparency to the surface to make
# the points underneath visible; the points are themselves made transparent
di + geom_point(na.rm=T,size=1,alpha=I(1/10)) +
    stat_density2d(na.rm=T,aes(fill = ..level..,alpha=..level..), geom="polygon")  +
     scale_fill_gradient(low="blue",high="red")
```