Session 3: A consize guide to ggplot2

Ashwin Karanam

June 27, 2018

Contents

1 The ggplot2 package

A graphing package in the tidyverse based on "The Grammar of Graphics" (Lelanad Wilkson). It uses a layered structure to graphing which simplifies the process of coding plots in R. Intuitively, we know that any graph looks like Figure 1. ggplot adopts this concept.

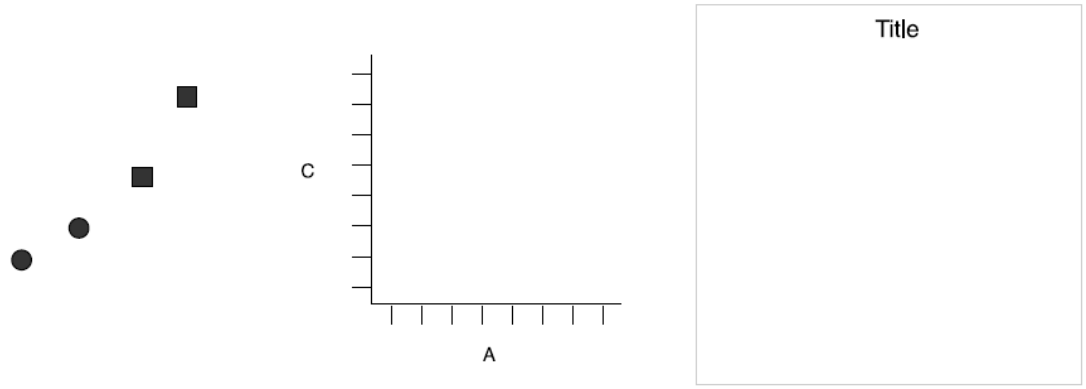


Figure 1. Graphics objects produced by (from left to right): geometric objects, scales and coordinate system, plot annotations.

1.1 The Grammar in ggplot2

Essential layers that dictate a plot:

1. a default dataset and set of mappings from variables to aesthetics,
2. one or more layers, with each layer having one geometric object, one statistical transformation,
3. one position adjustment, and optionally, one dataset and set of aesthetic mappings,
4. one scale for each aesthetic mapping used,
5. a coordinate system,
6. the facet specification

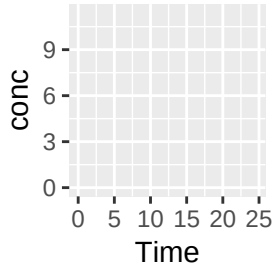
2 Hands on with ggplot2

Because we are interested in R for PMx, lets use PK data. The thoephylline data is good enough.

```
library(ggplot2)
library(dplyr)
library(gridExtra)
library(mrgsolve)

df <- data.frame(Theoph)
```

As we have already worked with this data before, let's skip the introductions and get right into the coding. Use the `ggplot()` function to create the first layer which is the base plot.

```
ggplot(df,
       aes(Time,conc))
```

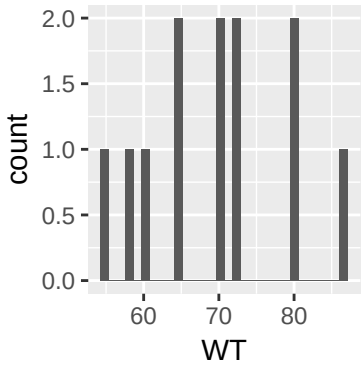


Every plot should start with `ggplot()` as this maps the x and y axis along with which data to use. To this we add layers that inform what type of plot to create using "geoms".

2.1 Histograms

Let's try a basic histogram plot of subject weights. We will use the `geom_histogram()` for this.

```
wt <- df %>%
  group_by(df$Subject) %>%
  summarise(WT = mean(Wt))

ggplot(wt,aes(WT))+
  geom_histogram()
```



There are several options within `geom_histogram()` that let you modify the histograms.

2.2 Scatterplots/Trendplots

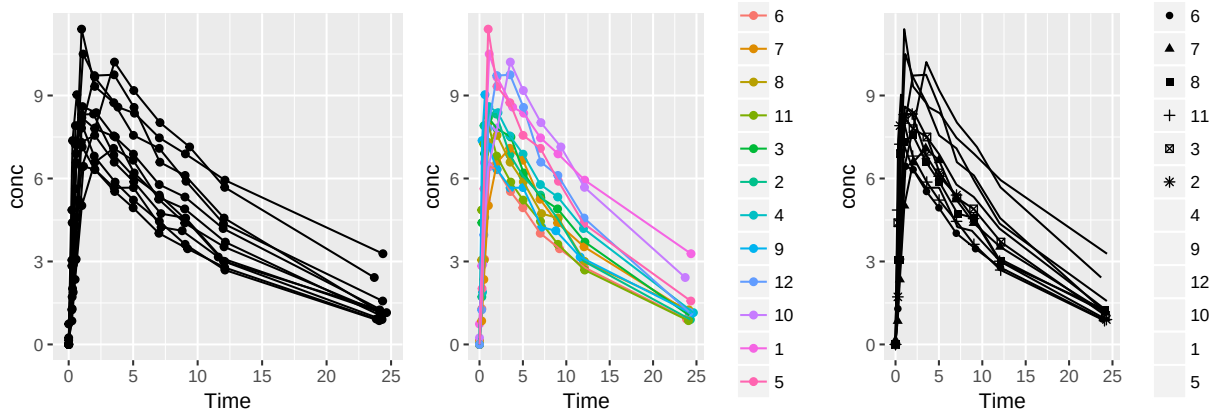
2.2.1 Sphagetti plots

Trendplots are one of the most important type of plots in PMx. `geom_line` is used for connecting observations in the order they appear in. The "group" argument allows you to map a group of factors. The other way of doing this would be to use the color and linetype arguments.

```
a <- ggplot(df,aes(Time,conc,group=Subject))+
  geom_point()+
  geom_line()

b <- ggplot(df,aes(Time,conc,color=Subject))+
  geom_point()+
  geom_line()

c <- ggplot(df,aes(Time,conc,group=Subject))+
  geom_point(aes(shape=Subject))+
  geom_line()

grid.arrange(a,b,c,ncol=3)
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values
## because more than 6 becomes difficult to discriminate; you have
## 12. Consider specifying shapes manually if you must have them.
```

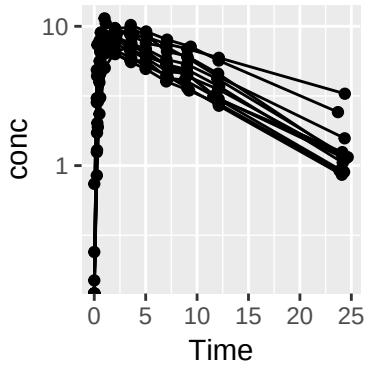
```
## Warning: Removed 66 rows containing missing values (geom_point).
```



As you see they do the same thing as group, but assigns a different color/linetype to each group. Of course this is not such a good idea because of limited number of shapes/lines!

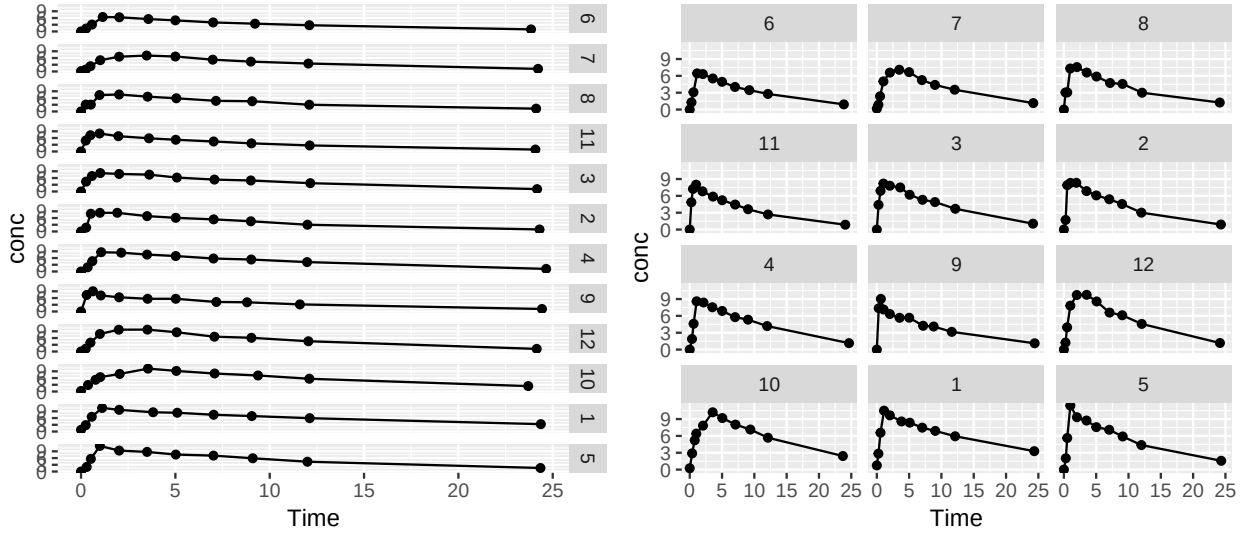
Another commonly used transformation is for the y-axis. For log scale Y-axis, we use the `scale_y_log10()` function which is an offshoot of `scale_y_continuous()`. Similar functions are available for x-axis transformations.

```
ggplot(df,aes(Time,conc,group=Subject))+
  geom_point()+
  geom_line()+
  scale_y_log10()
```



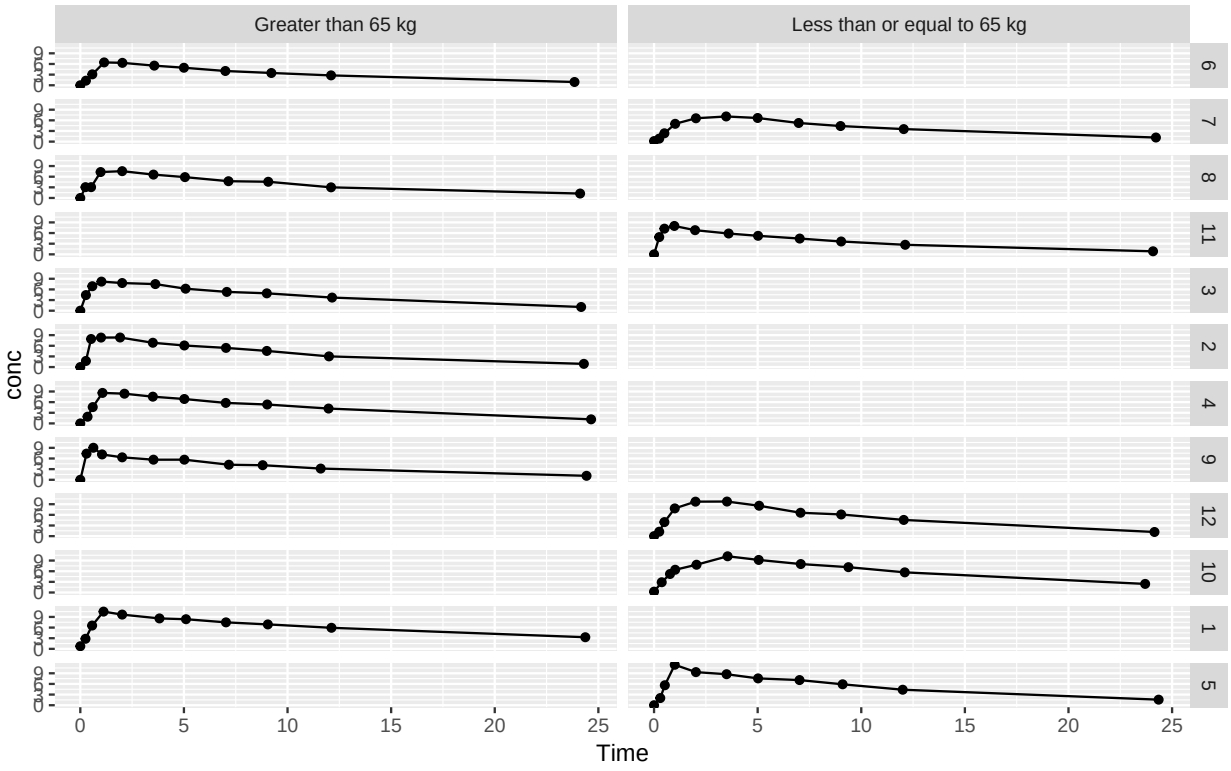
Now, let's create one plot for each individual. `facet_grid()` is one way to facet your data. This is a good verb to use for small IDs, but becomes useless when use large datasets. The other verb useful here is `facet_wrap()` which allows you to customize your grid.

```
c <- ggplot(df,aes(Time,conc))+
  geom_point()+
  geom_line()+
  facet_grid(Subject~.)

d <- ggplot(df,aes(Time,conc))+
  geom_point()+
  geom_line()+
  facet_wrap(~Subject,ncol=3)

grid.arrange(c,d,ncol=2)
```



We can add a second layer of faceting to this.

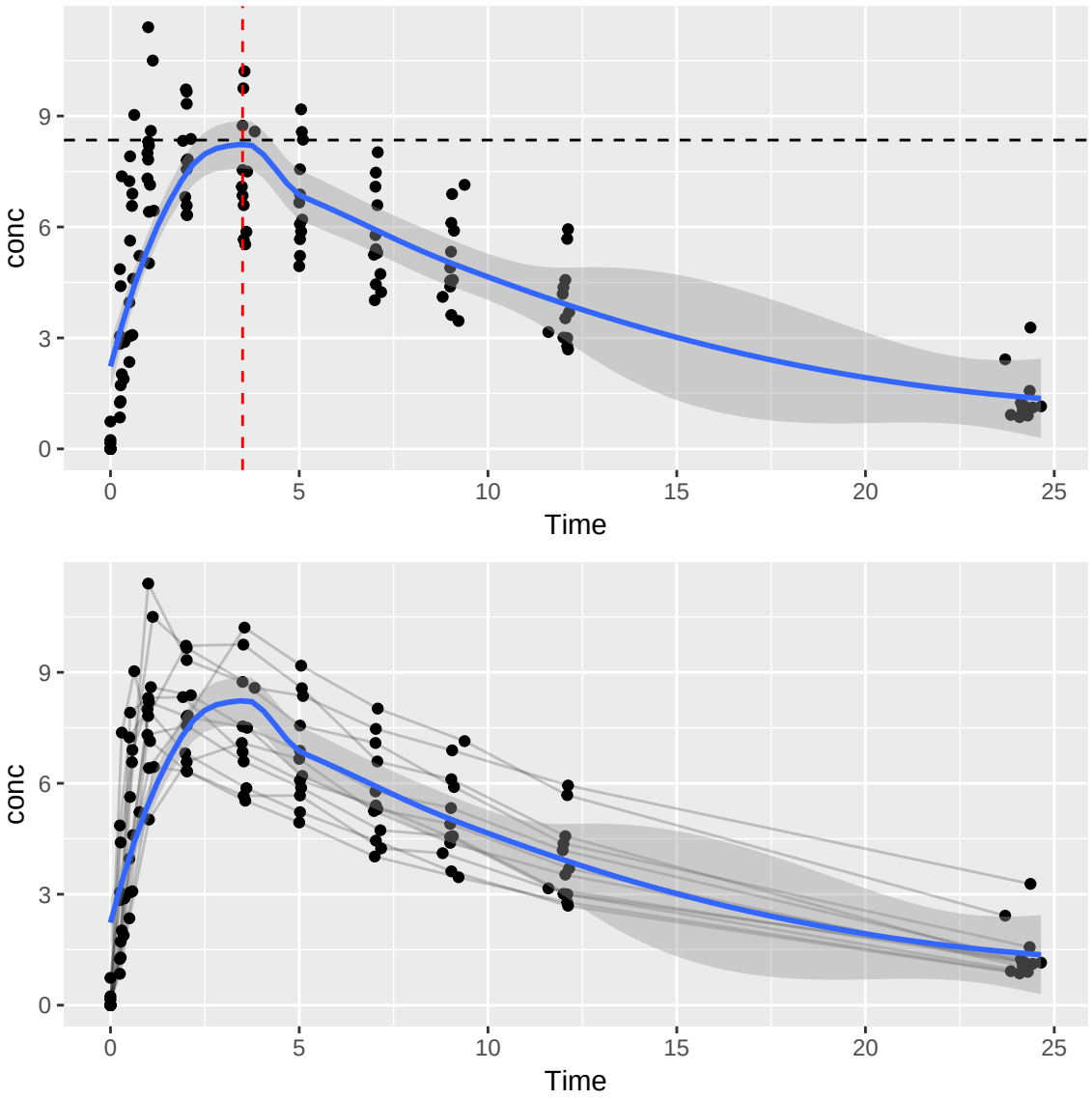
```
df2 <- df %>%
  mutate(Category = if_else(Wt <= 65,
                            "Less than or equal to 65 kg",
                            "Greater than 65 kg"))

ggplot(df2,aes(Time,conc))+
  geom_point()+
  geom_line()+
  facet_grid(Subject~Category)
```



2.2.2 Trendplots

With the same concentration time plot, you can add a trend line eg. a non-parametric smoother. The `geom_smooth` verb allows for this in ggplot.

```
e <- ggplot(df,aes(Time,conc))+
  geom_point()+
  geom_smooth(se=TRUE,method="loess")+
  geom_vline(xintercept = 3.5, linetype=2,color="red")+
  geom_hline(yintercept = 8.35, linetype=2,color="black")

f <- ggplot(df,aes(Time,conc))+
  geom_point()+
  geom_line(aes(group=Subject),alpha=0.2)+
  geom_smooth(se=TRUE,method="loess")

grid.arrange(e,f,ncol=1)
```



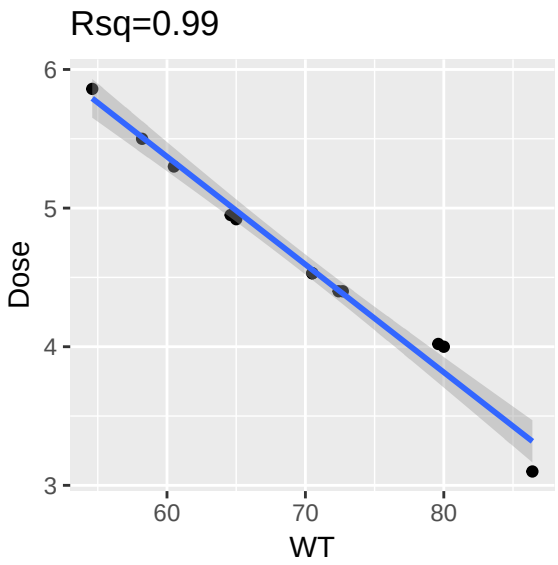
Couple of things to notice in the first plot:

1. the grouping asthetic is shifted to geom_point as geom_smooth does not need the grouping asthetic.
2. By default geom_smooth plots the SE and uses the loess method. Additional methods include lm, glm and gam.
3. Use `geom_hline()` and `geom_vline()` to create reference lines.

The second plot here is adding individual sphagetti plots to the average plot. I am using the `aplha` argument in the geom_line to make the individual lines lighter. You can use this for any geom available.

While we are on smoothers, let me introduce you to correlation plots. Let's say we want a correlation plot for weight and dose.

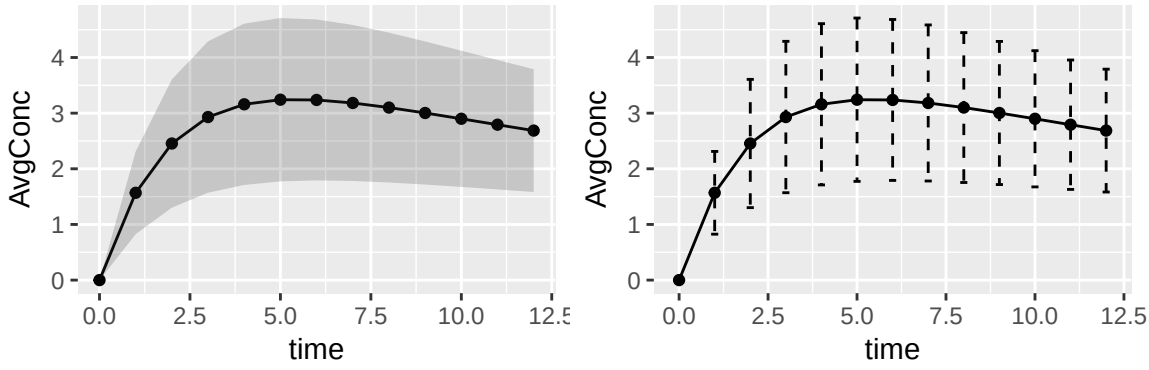
```
wt <- df %>%
  dplyr::group_by(Subject) %>%
  dplyr::summarise(WT = mean(Wt),Dose=mean(Dose))

cor <- cor(wt$WT,wt$Dose)

ggplot(wt,aes(WT,Dose))+
  geom_point()+
  geom_smooth(method="lm")+
  labs(title="Rsq=0.99")
```



Do not use these in-lieu of actual statistical analysis. Confirm your linear fits by running regression analysis.

Another common type of plot is an average trend plot with standard deviation. Use `geom_ribbon()` for creating a shaded region in your plot. We can also use `geom_errorbar()` for this which would give you error bars around the mean. Let us use something from mrgsolve for this exercise.

```
mod <- mread_cache("popExample",modlib())

idata <- data.frame(ID=1:10)
mat1 <- dmat(0.1,0.2)

set.seed(12358)
out <- mod %>%
  omat(mat1) %>%
  idata_set(object="idata")%>%
  ev(amt=100)%>%
  obsonly()%>%
  mrgsim(end=12,delta=1) %>%
  as.data.frame()

avg <- out %>%
  dplyr::group_by(time)%>%
  dplyr::summarise(AvgConc = mean(DV),
                   SD=sd(DV))

g <- ggplot(avg,aes(time,AvgConc))+
  geom_point()+
  geom_line()+
  geom_ribbon(aes(ymin=AvgConc-SD,ymax=AvgConc+SD),alpha=0.2)

h <- ggplot(avg,aes(time,AvgConc))+
  geom_point()+
  geom_line()+
  geom_errorbar(aes(ymin=AvgConc-SD,ymax=AvgConc+SD),
                linetype=2,
                size=0.5,
                width=0.2)

grid.arrange(g,h,ncol=2)
```

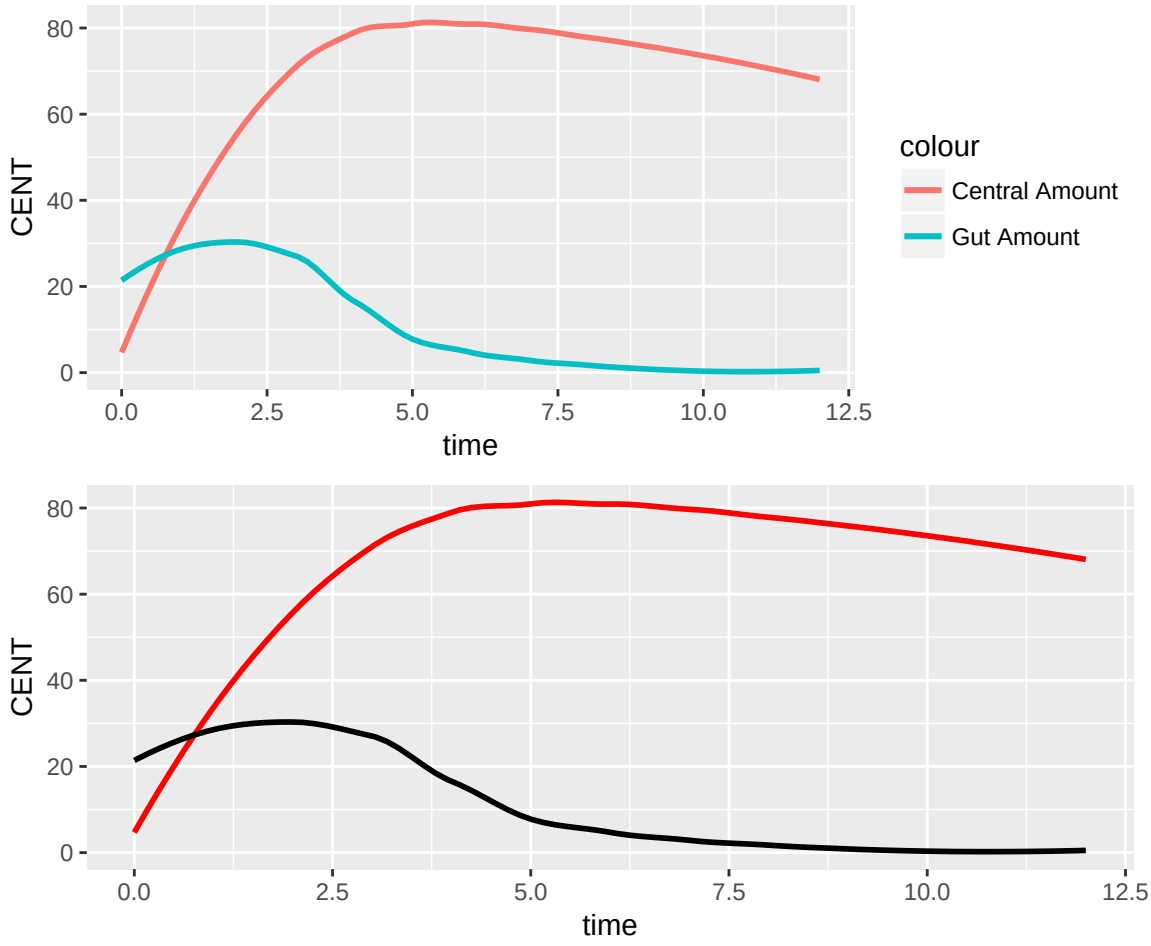


2.3 Adding multiple trend lines

2.3.1 Multiple smoothers of the same type of data

Use `geom_smooth` for adding multiple trends. Just add a seperate `geom_smooth` with new mappings.

```
i <- ggplot(out, aes(time,CENT))+
  geom_smooth(aes(color="Central Amount"), se=FALSE)+
  geom_smooth(aes(time, GUT,color="Gut Amount"), se=FALSE)

j <- ggplot(out, aes(time,CENT))+
  geom_smooth(color="red", se=FALSE)+
  geom_smooth(aes(time, GUT),color="black", se=FALSE)


grid.arrange(i,j,ncol=1)
```

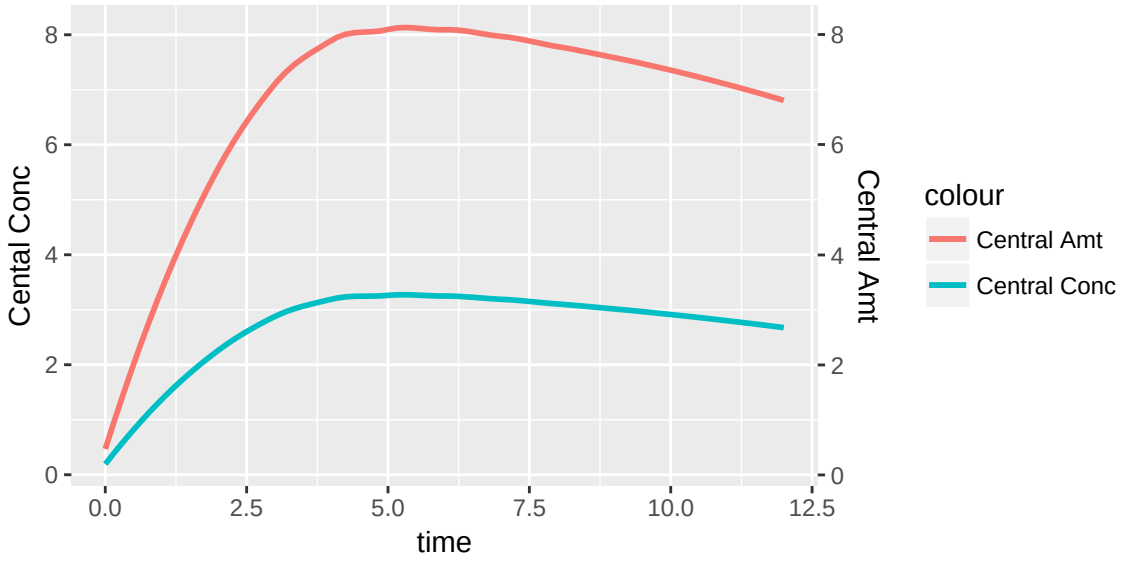


If you notice, in the first plot the color argument is within the aesthetics argument. This forces ggplot to consider all mappings in that geom as one entity. Here we are forcing ggplot to call the first layer as central concentration and second as central amount. In the second plot, we do not use the aesthetics mapping for color but use the RGB manual color selection. This does not give us a legend because you manually specified which layer has which color.

2.3.2 Multiple smoothers of different types of data

If you want to simulatenously plot a concentration and an amount curve, you'll need two Y-axis. `sec.axis` is the verb to create a secondary axis.

```
ggplot(out, aes(time,DV))+
  geom_smooth(aes(color="Central Conc"), se=FALSE)+
  geom_smooth(aes(time,(CENT/10),color="Central Amt"),se=FALSE)+
  scale_y_continuous( name="Cental Conc",
                      sec.axis = sec_axis(trans = ~. *1, name = "Central Amt"))
```



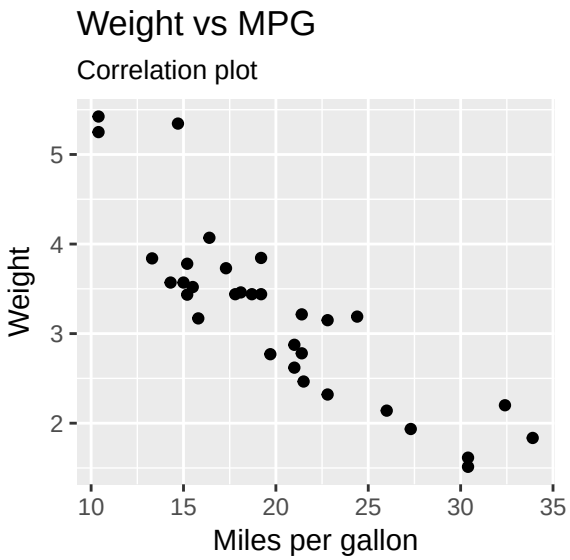
You can also use `sec_axis()` to create a secondary axis, be it for x or y axes.

2.4 Formatting in ggplot

2.4.1 Formatting the axis labels

Use the `labs()` to edit the plot title, and axis labels. You can also use `xlab()`, `ylab()` and `ggtitle()` to individually edit these. For this exercise let us use the "mtcars" dataset

```
cars <- data.frame(mtcars)

ggplot(cars, aes(mpg, wt))+
  geom_point()+
  labs(title="Weight vs MPG",
       subtitle="Correlation plot",
       x="Miles per gallon",
       y="Weight")
```

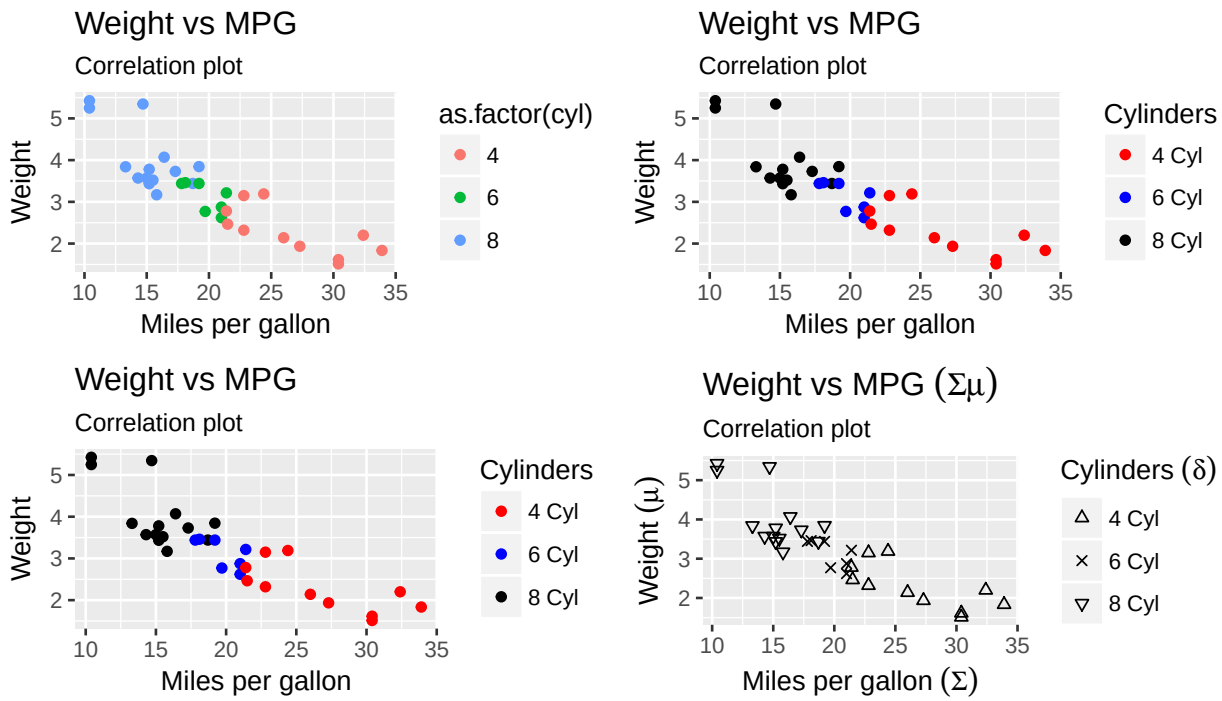


2.4.2 Formatting legends

ggplot uses the names of your columns or factors in said colums to determine the names of the legends. An esay way to override those is to use the `scale_color_manual()` verb. This allows you to choose customize not only the names and titles, but also the color. Similar functions are available for shape `scale_shape_manual()`, fill `scale_fill_manual()`, size `scale_size_manual()`, linetype `scale_linetype_manual()` and alpha `scale_alpha_manual()`. Similar off-shoots are available for discrete variables `scale_*_discrete()` and continious variables `scale_*_continuous()`.

```
k <- ggplot(cars, aes(mpg, wt, color=as.factor(cyl)))+
  geom_point()+
  labs(title="Weight vs MPG",
       subtitle="Correlation plot",
       x="Miles per gallon",
       y="Weight")

l <- ggplot(cars, aes(mpg, wt, color=as.factor(cyl)))+
  geom_point()+
  labs(title="Weight vs MPG",
       subtitle="Correlation plot",
       x="Miles per gallon",
       y="Weight")+
  scale_colour_manual(values=c("red", "blue", "black"),
                      name="Cylinders",
                      breaks=c("4", "6", "8"),
                      labels=c("4 Cyl", "6 Cyl", "8 Cyl"))

m <- ggplot(cars, aes(mpg, wt, color=as.factor(cyl)))+
  geom_point()+
  labs(title="Weight vs MPG",
       subtitle="Correlation plot",
       x="Miles per gallon",
       y="Weight")+
  scale_colour_manual(values=c("red", "blue", "black"),
```

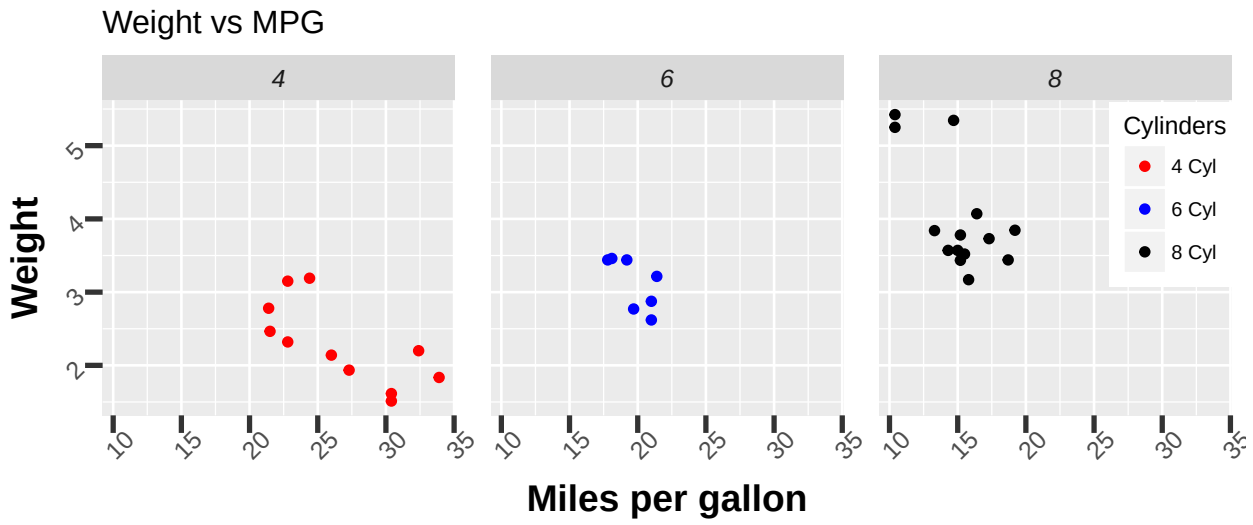
```
                        name="Cylinders",
                        breaks=c("4", "6", "8"),
                        labels=c("4 Cyl", "6 Cyl", "8 Cyl"))

n <- ggplot(cars, aes(mpg, wt, shape=as.factor(cyl)))+
  geom_point()+
  labs(title="Weight vs MPG"~(Sigma*mu),
       subtitle="Correlation plot",
       x="Miles per gallon"~(Sigma),
       y="Weight"~(mu))+
  scale_shape_manual(values=c(2,4,6),
                        name="Cylinders"~(delta),
                        breaks=c("4", "6", "8"),
                        labels=c("4 Cyl","6 Cyl","8 Cyl"))

grid.arrange(k,l,m,n,ncol=2)
```



2.4.3 Text size formatting

`theme()` is a great way of formatting all your text in one place. There are around 50 arguments within `theme()` all aimed at providing user control over the formatting.

```
ggplot(cars, aes(mpg, wt, color=as.factor(cyl)))+
  geom_point()+
  facet_grid(.~as.factor(cyl))+
  labs(title="Weight vs MPG",
       x="Miles per gallon",
       y="Weight")+
  scale_colour_manual(values=c("red", "blue", "black"),
                      name="Cylinders",
                      breaks=c("4", "6", "8"),
                      labels=c("4 Cyl", "6 Cyl", "8 Cyl")) +
  theme(axis.title.x=element_text(size=15,face="bold"),
        axis.title.y=element_text(size=15,face="bold"),
        axis.text = element_text(size=10,angle = 45),
        axis.ticks = element_line(size = 1),
        axis.ticks.length = unit(.25, "cm"),
        strip.text.x = element_text(size=10,face="italic"),
        legend.position = c(0.95,0.7),
        legend.title=element_text(size=10),
        legend.text=element_text(size=8),
        panel.spacing = unit(1, "lines"))
```



3 Colors, themes and exporting with ggplot2

3.1 R Color

This is a short section on colors and how to custmomize colors in R and in ggplot2

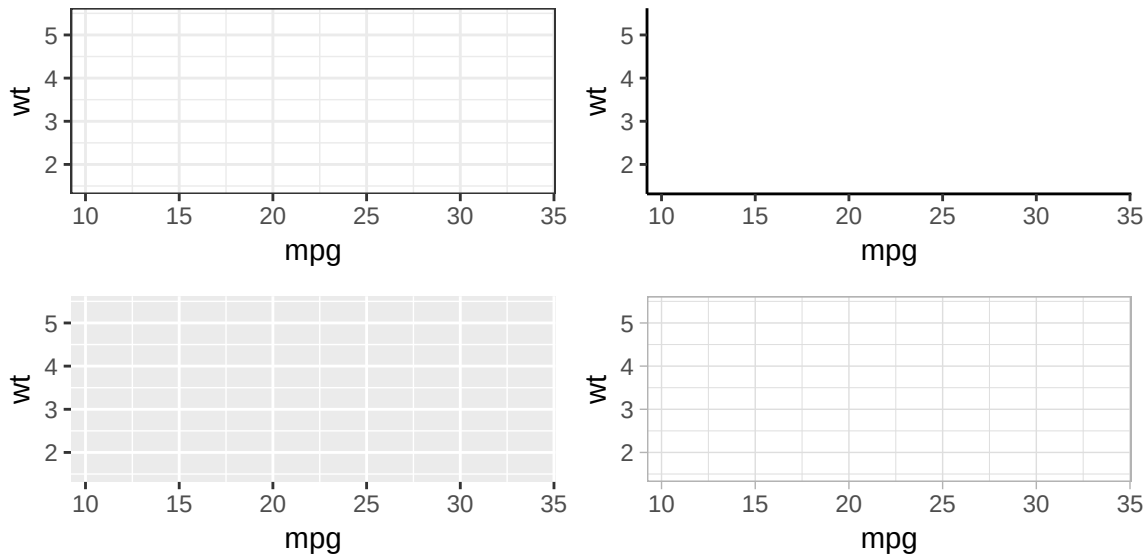
There are several help or how-to sections on the web about R colors. There are countless times when the defaults in ggplot look decent for a presentation, but when you print it out, they look really bad. Overcome them by:

1. Here is a consize guide to the color choices are given by `Columbia-Stats`. These colors go hand in hand with `scale_color_manual()`
2. If you're a fan of using the color pallete instead of the names, then I suggest use this `link`
3. If you want to be adventurous, there's several packages that give fun palletes like `wesanderson`

3.2 Themes in ggplot

There are several built in themes availabe for ggplot. Use the `theme_*()` verb to explore the options. Here are some for demostration.

```
o <- ggplot(cars, aes(mpg, wt, shape=as.factor(cyl)))+
  theme_bw()

p <- ggplot(cars, aes(mpg, wt, shape=as.factor(cyl)))+
  theme_classic()

q <- ggplot(cars, aes(mpg, wt, shape=as.factor(cyl)))+
  theme_grey()

r <- ggplot(cars, aes(mpg, wt, shape=as.factor(cyl)))+
  theme_light()

grid.arrange(o,p,q,r,ncol=2)
```



3.3 Exporting publication quality images

You can output almost any type of image file. The verbs are `tiff()`, `bmp()`, `jpeg()`, and `png()`.

```
tiff("test.tiff", width = 9, height = 8, units = 'in', res = 300)
l
dev.off()
```

4 Helpful References

1. `R in Action` by Robert Kabacoff
2. `Hands-On Programming` with R by Garrett Grolemund
3. `R Cookbook` by Paul Teetor
4. `THE ART OF R PROGRAMMING` by Norman Matloff
5. `The Grammar of Graphics` by Leland Wilkinson
6. `A Layered Grammar of Graphics` by Hadley Wickham

5 Session Information

```
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] bindrcpp_0.2         mrgsolve_0.8.10.9007 gridExtra_2.3
## [4] dplyr_0.7.4          ggplot2_2.2.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.15              bindr_0.1
##  [3] knitr_1.20                magrittr_1.5
##  [5] munsell_0.4.3             colorspace_1.3-2
##  [7] R6_2.2.2                  RcppArmadillo_0.8.300.1.0
##  [9] rlang_0.2.0               stringr_1.3.1
## [11] plyr_1.8.4                tools_3.4.3
## [13] grid_3.4.3                gtable_0.2.0
## [15] htmltools_0.3.6           yaml_2.1.16
## [17] lazyeval_0.2.1            rprojroot_1.3-2
```

```
## [19] digest_0.6.15            assertthat_0.2.0
## [21] tibble_1.4.2             reshape2_1.4.3
## [23] glue_1.2.0               evaluate_0.10.1
## [25] rmarkdown_1.8            labeling_0.3
## [27] stringi_1.1.7            compiler_3.4.3
## [29] pillar_1.1.0             scales_0.5.0
## [31] backports_1.1.2          pkgconfig_2.0.1
```