xkcd plotting in R

Kayla Peck

Wednesday, February 24, 2016

XKCD is a popular stick figure web comic with themes in mathematics, science, language, and romance created by Randall Munroe. The **xkcd** package was developed in order to allow XKCD style graphs to be plotted in R, using a ggplot2 style. This package allows you to plot axes and lines that look handdrawn and plot stick figures (with the head represented as a circle and the arms, legs, and torso as straight lines).

This tutorial was developed and expanded upon based on the document xkcd: An R Package for Plotting XKCD Graphs.

Installing the XKCD fonts

The **xkcd** package uses the XKCD fonts which must be installed on your computer. Installation runs through the **extrafont** package.

```
install.packages("extrafont")
library(extrafont)
```

You can try the following code to download the font file (.ttf) from an external site:

```
download.file("http://simonsoftware.se/other/xkcd.ttf",
              dest="xkcd.ttf", mode="wb")
system("mkdir ~/.fonts/")
system("cp xkcd.ttf ~/.fonts")
font_import(pattern = "[X/x]kcd", prompt=FALSE)
fonts()
fonttable()
if(.Platform$OS.type != "unix"){
  ##Register fonts for Windows bitmap output
  loadfonts(device="win")
} else{
  loadfonts()
}
```

The above code, however, didn't work on my own machine (Windows). So instead, I used the url to manually download the file (click here). Next, I dragged/copied the .ttf file into the **C:/Windows/Fonts** folder.

After copying the font file over, use the following code (same as above without downloading the file):

```
font_import(pattern = "[X/x]kcd", prompt=FALSE)
fonts()
fonttable()
if(.Platform$OS.type != "unix"){
  ##Register fonts for Windows bitmap output
  loadfonts(device="win")
} else{
  loadfonts()
}
```

Installing xkcd

Use the following code to install and load the **xkcd** package:

```
install.packages("xkcd", dependencies = TRUE)
```

```
library(xkcd)
```

Loading **xkcd** automatically loads **ggplot2** and **extrafont**. Including `dependencies = TRUE` should install those packages if you have not done so before. You may also need to install **Rcpp** (a dependency of **ggplot2**) if you have not done so before and receive an error message for it.

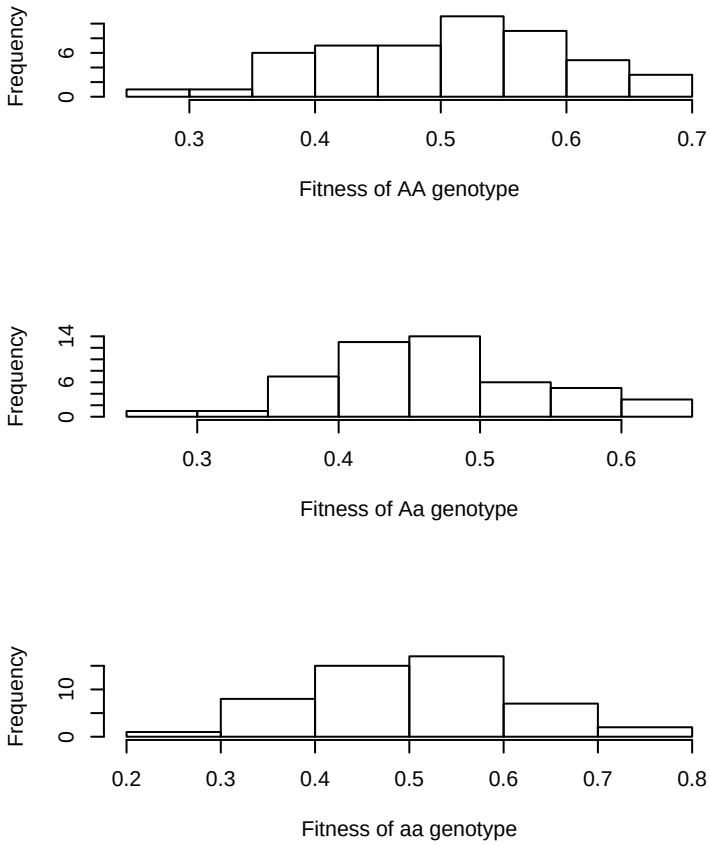
A quick introduction to ggplot2

There are three common ways to generate plots in R. Base, Lattice, and ggplot2. I use base on a daily basis because it is the most intuitive. However, latticce and ggplot2 can often produce more aesthetically pleasing graphs and allows for more elaborate specifications. As an example, here is the same plot in each style using only the most basic code.

```
mydat <- data.frame(genotype=rep(c("AA","Aa","aa"),50), fitness=rnorm(150,0.5,0.1))
```

Base

```
par(mfrow=c(3,1))
hist(mydat$fitness[mydat$genotype=="AA"], xlab="Fitness of AA genotype", main="")
hist(mydat$fitness[mydat$genotype=="Aa"], xlab="Fitness of Aa genotype", main="")
hist(mydat$fitness[mydat$genotype=="aa"], xlab="Fitness of aa genotype", main="")
```

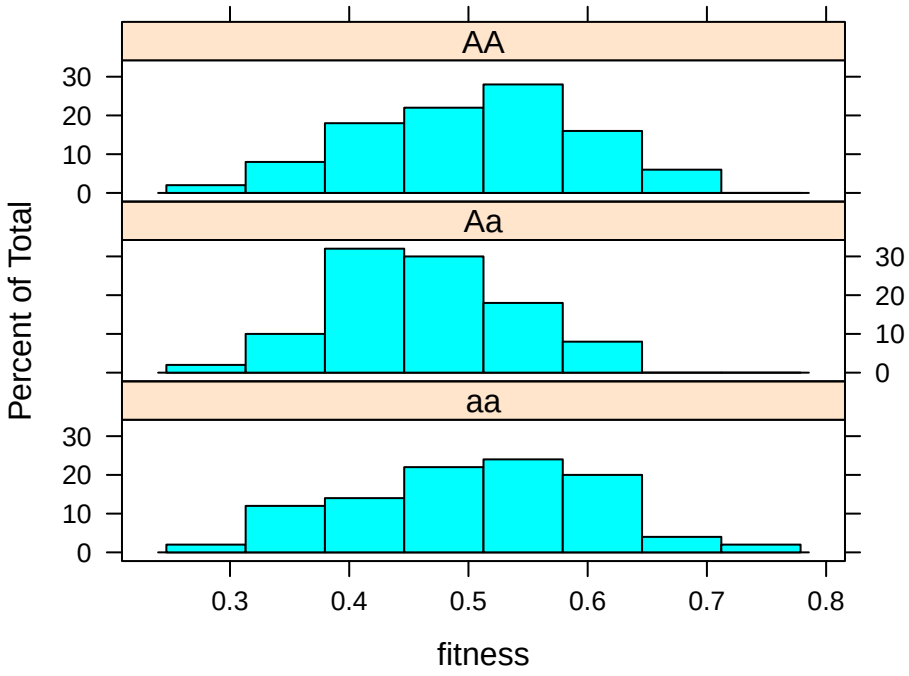


Three panels in base graphics.

Lattice

```
install.packages("lattice")
```

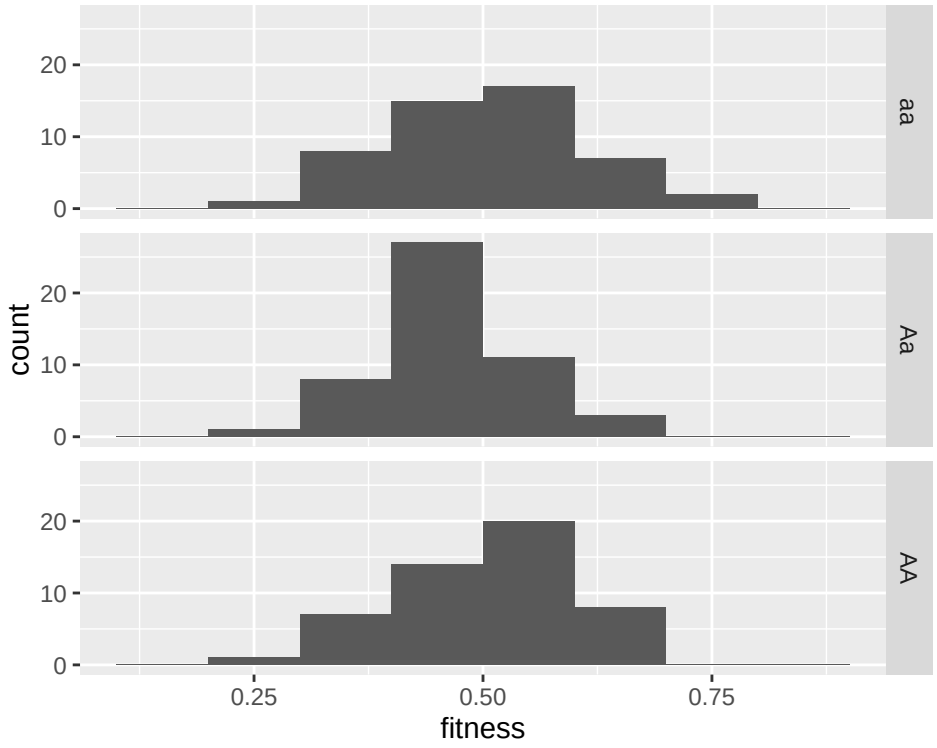
```
library(lattice)
histogram(~fitness | factor(genotype), data=mydat, layout=c(1,3))
```



Lattice histograms.

ggplot2

```
library(ggplot2)
qplot(fitness,geom="histogram",data=mydat,facets=genotype~., binwidth=0.1)
```



ggplot2 histograms.

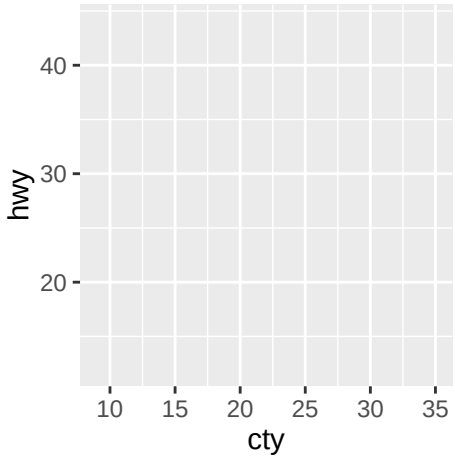
You can get an idea of the potential benefits of lattice and ggplot2, including using less lines of code to plot the same thing. You can check out examples of ggplot2 plots here.

Basics of ggplot2

We used `qplot()` above but that function generates a complete plot. The useful feature of ggplot2 is being able to add to a plot using layers.

Each plot begins with the format:

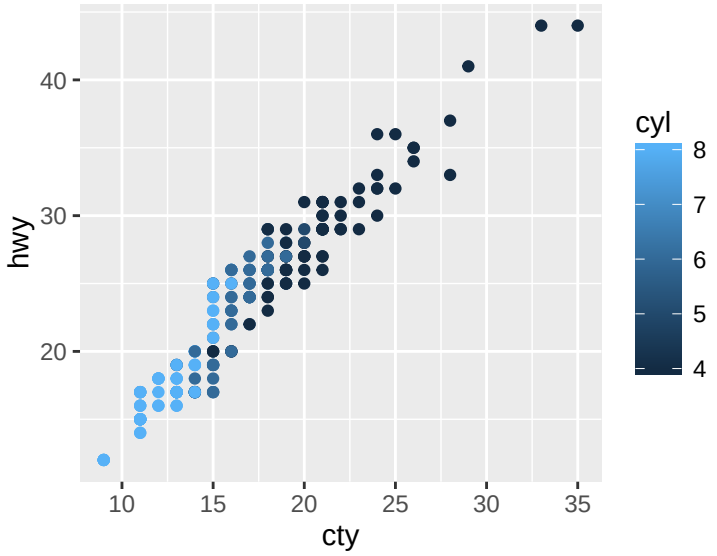
```
ggplot(data = mpg, aes(x = cty, y = hwy))
```



Here, `data` specifies the dataset being used, while `aes` specifies your x and y variables from within mpg. `aes()` is used to generate aesthetic mappings that describe how variables in the data are mapped to visual properties (aesthetics) of geoms. As you noticed, the above code creats an empty plot because we haven't designated what layers to add.

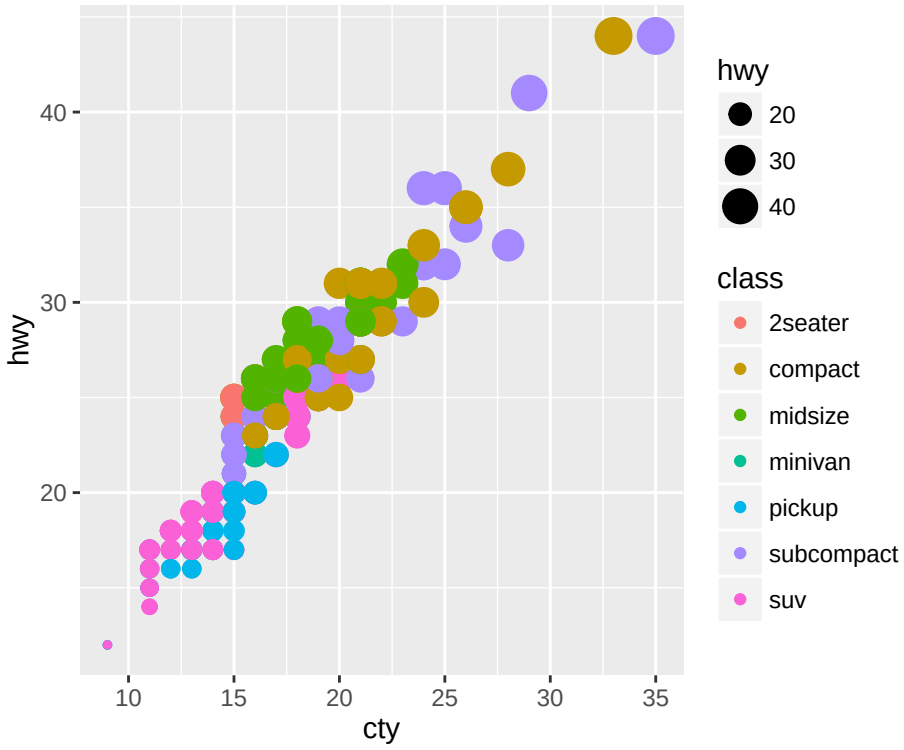
Add layers to your plot using a set of `geom_*()` functions. For example, to create a scatterplot, use `geom_point()`:

```
ggplot(data = mpg, aes(x = cty, y = hwy)) + geom_point(aes(color = cyl))
```



Or try:

```
ggplot(data = mpg, aes(x = cty, y = hwy)) + geom_point(aes(color = class, size=hwy))
```



Here the data is colored based on the class of the car and the size of the point designates the highway mpg.

In addition to `geom_point()`, you can use functions such as:

- geom_density()
- geom_histogram()
- geom_line()
- geom_jitter()
- geom_smooth()
- geom_bar()
- geom_boxplot()
- geom_contour()

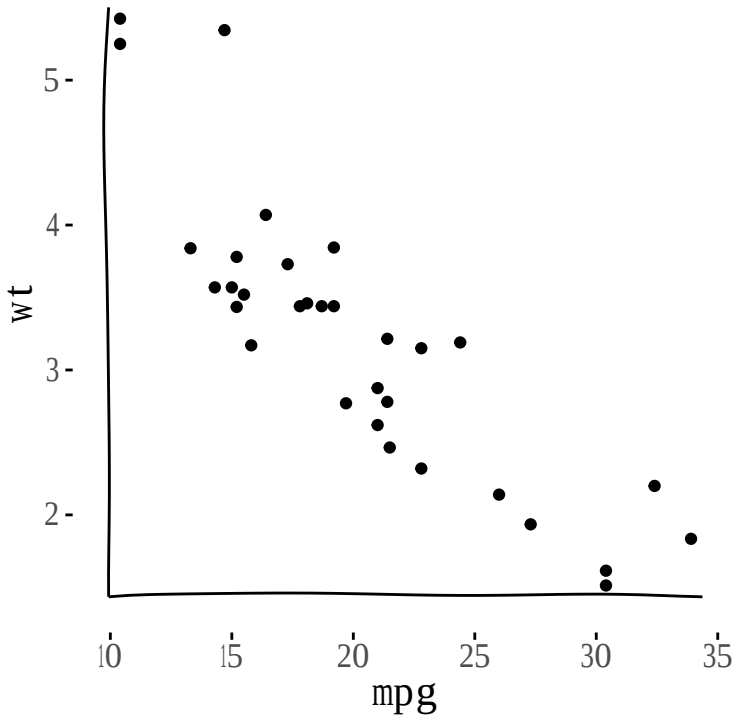
Check out this cheatsheet to familiarize yourself with the basic capabilities.

Axis, Stick Figures, and Facets

Drawing the axis

In order to get a hand-drawn look to the axis, use the function `xkcdaxis()`.

```
xrange <- range(mtcars$mpg)
yrange <- range(mtcars$wt)
set.seed(123)
plot <- ggplot() + geom_point(aes(mpg,wt), data=mtcars) +
  xkcdaxis(xrange, yrange)
plot
```



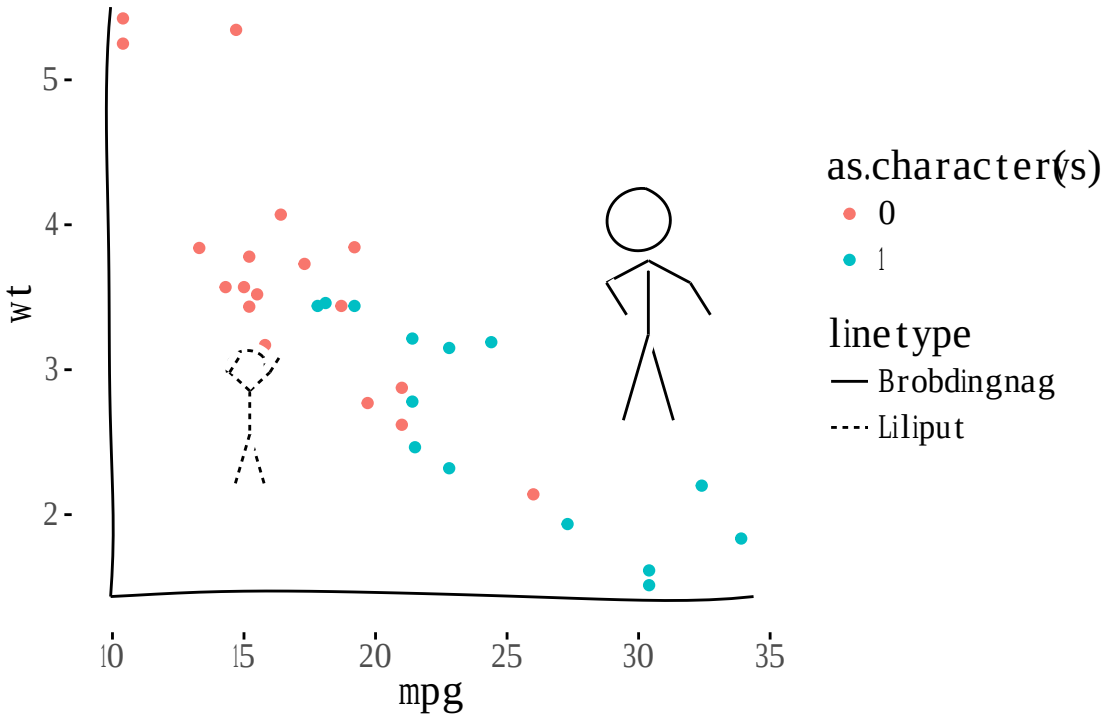
Here, the `set.seed()` function is used to reproduce the same white noise every time (for the axis). Use the same number to generate the same figure.

Drawing a stick figure

The `xkcdman()` function draws a stick figure with different positions, angles, and lengths.

```
ratioxy <- diff(xrange)/diff(yrange)
#designate the characters you are going to map
mapping <- aes(x, y, scale, ratioxy, angleofspine,
                anglerighthumerus, anglelefthumerus,
                anglerightradius, angleleftradius,
                anglerightleg, angleleftleg, angleofneck,
                linetype=city)
#Use vectors to make 2 stick men at once, one at (15,3) and one at (30,4)
dataman <- data.frame(x= c(15,30), y=c(3, 4),
                      scale = c(0.3,0.51) ,
                      ratioxy = ratioxy,
                      angleofspine = -pi/2 ,
                      anglerighthumerus = c(pi/4, -pi/6),
```

```
                        anglelefthumerus = c(pi/2 + pi/4, pi +pi/6),
                        anglerightradius = c(pi/3, -pi/3),
                        angleleftradius = c(pi/3, -pi/3),
                        anglerightleg = 3*pi/2 - pi / 12,
                        angleleftleg = 3*pi/2 + pi / 12 ,
                        angleofneck = runif(1, 3*pi/2-pi/10, 3*pi/2+pi/10),
                        city=c("Liliput","Brobdingnag"))
plot <- ggplot() + #Set up an empty plot
  geom_point(aes(mpg, wt, colour=as.character(vs)), data=mtcars) +
  xkcdaxis(xrange,yrange) + #Set up the xkcd-themed axis
  xkcdman(mapping, dataman) #Add the xkcd stick figures
plot
```

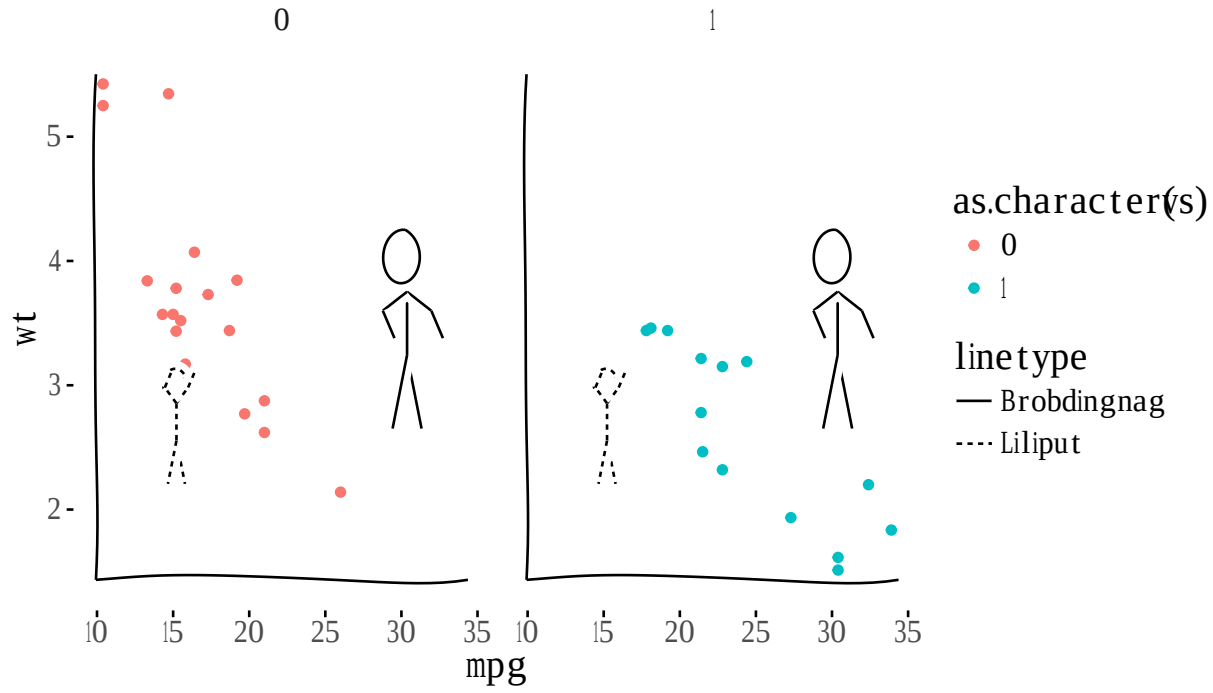


Note that two stick figures are drawn because two sets of coordinates were given in the dataman data frame. Using `linetype=city` caused the lines of the stick figures to be different. Additionally, the angles of the lines are in units of radians.

Facets

Use the facet feature of **ggplot2** to split your data up by one or more variables and then plot the subsets of data together.

```
plot + facet_grid(.~vs)
```

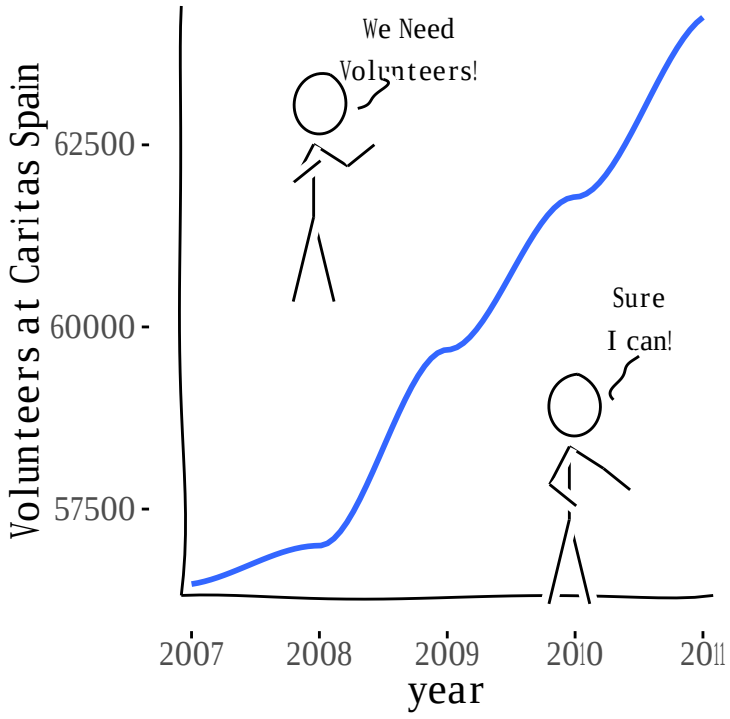


Here, `.~vs` designates to separate the data by the `vs` variable. See `head(mtcars)` to understand.

Some Basic Examples

```
volunteers <- data.frame(year=c(2007:2011),
                         number=c(56470, 56998, 59686, 61783, 64251))
xrange <- range(volunteers$year)
yrange <- range(volunteers$number)
ratioxy <- diff(xrange) / diff(yrange)
datalines <- data.frame(xbegin=c(2008.3,2010.5),ybegin=c(63000,59600),
                        xend=c(2008.5,2010.3), yend=c(63400,59000))
mapping <- aes(x, y, scale, ratioxy, angleofspine,
                 anglerighthumerus, anglelefthumerus,
                 anglerightradius, angleleftradius,
                 anglerightleg, angleleftleg, angleofneck)
dataman <- data.frame( x= c(2008,2010), y=c(63000, 58850),
                         scale = 1000 ,
                         ratioxy = ratioxy,
                         angleofspine = -pi/2 ,
                         anglerighthumerus = c(-pi/6, -pi/6),
                         anglelefthumerus = c(-pi/2 - pi/6, -pi/2 - pi/6),
                         anglerightradius = c(pi/5, -pi/5),
                         angleleftradius = c(pi/5, -pi/5),
                         angleleftleg = 3*pi/2 + pi / 12 ,
                         anglerightleg = 3*pi/2 - pi / 12,
                         angleofneck = runif(1, 3*pi/2-pi/10, 3*pi/2+pi/10))
plot <- ggplot() + geom_smooth(mapping=aes(x=year, y =number),
```

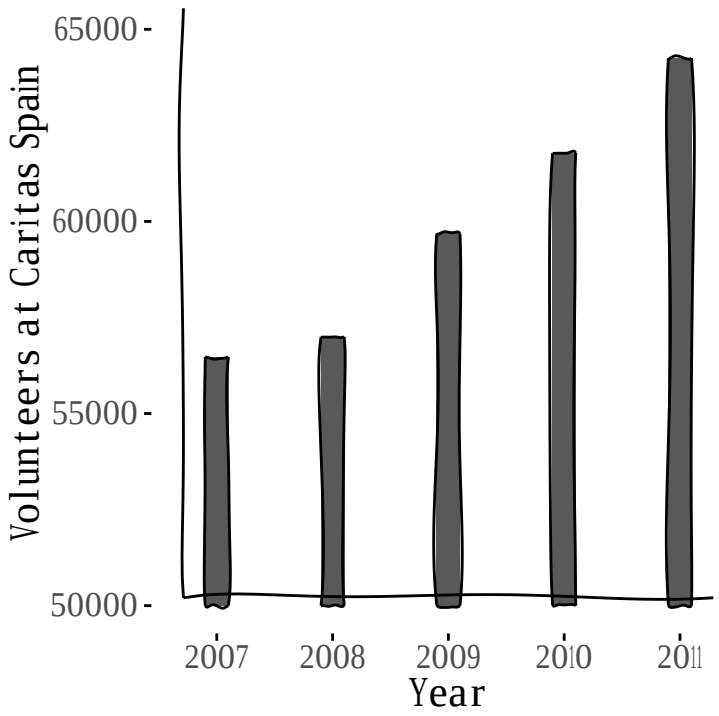
```
                              data =volunteers, method="loess") +
  xkcdaxis(xrange,yrange) +
  ylab("Volunteers at Caritas Spain") +
  xkcdman(mapping, dataman) +
  annotate("text", x=2008.7, y = 63800,
             label = "We Need\nVolunteers!", family="xkcd" ) +
  annotate("text", x=2010.5, y = 60100,
             label = "Sure\nI can!", family="xkcd" ) +
  xkcdline(aes(xbegin=xbegin,ybegin=ybegin,xend=xend,yend=yend),
             datalines, xjitteramount = 0.12)
plot
```



One cool thing about **xkcd** figures is that when lines intersect, there is a space in blank in the crossing, following XKCD style.

The following code plots the line chart as a bar chart:

```
data <- volunteers
data$xmin <- data$year - 0.1
data$xmax <- data$year + 0.1
data$ymin <- 50000
data$ymax <- data$number
xrange <- range(min(data$xmin)-0.1, max(data$xmax)+0.1)
yrange <- range(min(data$ymin)+500, max(data$ymax)+1000)
mapping <- aes(xmin=xmin, ymin=ymin, xmax=xmax, ymax=ymax)
plot <- ggplot() + xkcdrect(mapping, data) +
  xkcdaxis(xrange,yrange) +
  xlab("Year") + ylab("Volunteers at Caritas Spain")
plot
```



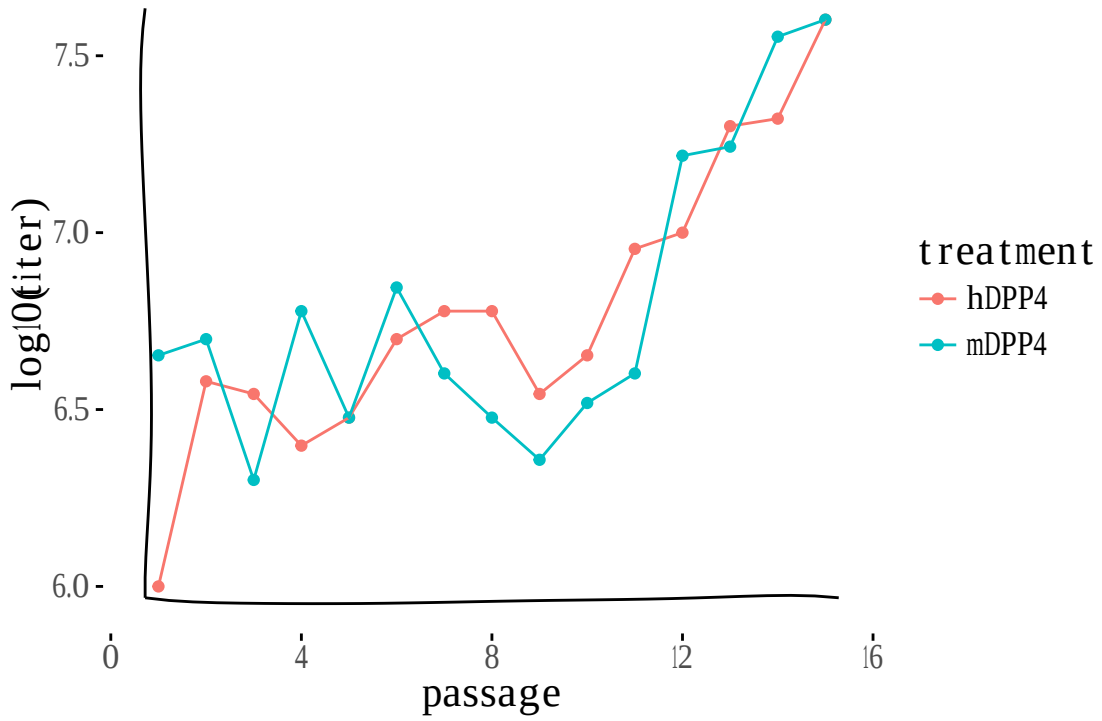
See `example(xkcdrect)` for how to specify the width, height, and colors of the bars.

Example with Real Data

I've been passaging virus to try to adapt it, and for each passage, I titer the virus (essentially see how many viruses I still have). If I see a decrease in titer over time, I can assume my virus population is dying out. If the titer remains constant, there is a good chance I'm achieving adaptation.

First, read in the data and plot titer against passage and also group by treatment (hDPP4 and mDPP4):

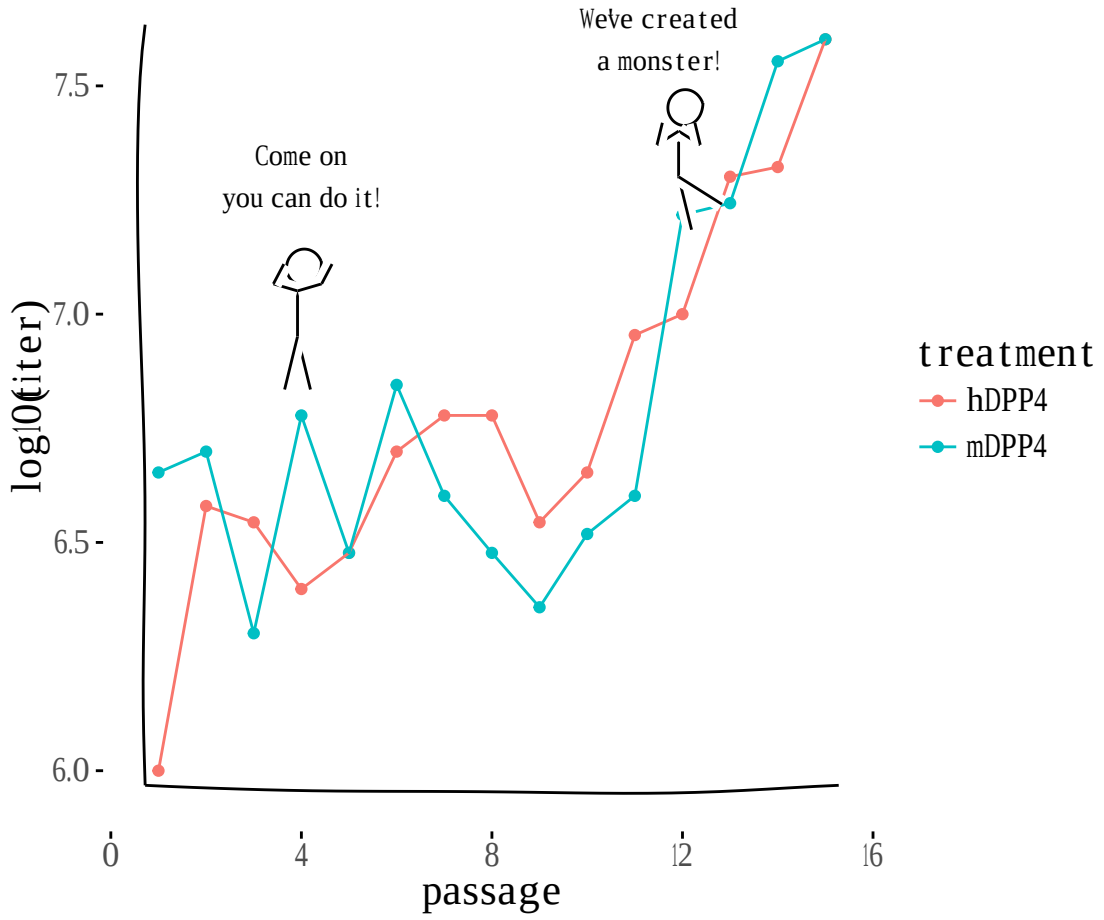
```
dat <- read.csv("host_invasion_titers.csv")
xrange <- range(dat$passage)
yrange <- range(log10(dat$titer))
set.seed(11)
plot <- ggplot() + geom_point(aes(passage,log10(titer),color=treatment), data=dat) +
  geom_line(aes(passage,log10(titer),color=treatment), data=dat) +
  xkcdaxis(xrange, yrange)
plot
```



Now let's try to add in a few stick figures. Note that in the above example code, they used pi and -pi and adding pis and it gets weird. For my own stick figures, I just used a classic radian circle for reference. Same thing, but easier on my brain.

```
ratioxy <- diff(xrange)/diff(yrange)
#designate the characters you are going to map
mapping <- aes(x, y, scale, ratioxy, angleofspine, anglerighthumerus, anglelefthumerus,
               anglerightradius, angleleftradius,
               anglerightleg, angleleftleg, angleofneck)
#Use vectors to make 2 stick men at once, one at (15,3) and one at (30,4)
dataman <- data.frame(x= c(4,12), y=c(7.1, 7.45),
                      scale = 0.1,
                      ratioxy = ratioxy,
                      angleofspine = -pi/2 ,
                      anglerighthumerus = c(pi/12, 11*pi/6),
                      anglelefthumerus = c(11*pi/12, 7*pi/6),
                      anglerightradius = c(pi/3, 7*pi/12),
                      angleleftradius = c(pi/3, 5*pi/12),
                      anglerightleg = c(3*pi/2 - pi / 12,11*pi/6),
                      angleleftleg = 3*pi/2 + pi / 12 ,
                      angleofneck = runif(1, 3*pi/2-pi/10, 3*pi/2+pi/10))
plot <- ggplot() + geom_point(aes(passage,log10(titer),color=treatment), data=dat) +
  geom_line(aes(passage,log10(titer),color=treatment), data=dat) +
  xkcdaxis(xrange,yrange) + #Set up the xkcd-themed axis
  annotate("text", x=4, y = 7.3,
           label = "Come on\nyou can do it!", family="xkcd" ) +
  annotate("text", x=11.5, y = 7.6,
           label = "We've created\na monster!", family="xkcd" ) +
```

```
  xkcdman(mapping, dataman) #Add the xkcd stick figures
plot
```



Note: for the record, we haven't created a monster.

Saving plots

In **ggplot2**, the function `ggsave()` is the preferred function for saving plots. To save as a .png file, use the code:

```
#plot <- qplot(1:10,1:10)
ggsave("plot.png", plot)
```

In order to save a plot as a PDF, the fonts needed to be embedded into the PDF using `embed_fonts()`. This also requires Ghostscript to be downloaded for Windows users: click here.

```
ggsave("plot.pdf", plot=plot, width=7, height=6)
if(.Platform$OS.type != "unix"){
  Sys.setenv(R_GSCMD =
```

```
                "C:/Program Files/gs/gs9.18/bin/gswin64c.exe") #path to Ghostscript executable
}
embed_fonts("plot.pdf")
```

If you're using RStudio, you can always generate the plot and use the *Export* drop-down menu and choose the *Save as PDF...* option.

References

- This document was generated using R markdown, cheatsheet here.
- Original xkcd tutorial document here.
- Also check out the ggplot2 cheatsheet.
- Radian circle for angles here.