Facets (ggplot2)

Problem

You want to do split up your data by one or more variables and plot the subsets of data together.

Solution

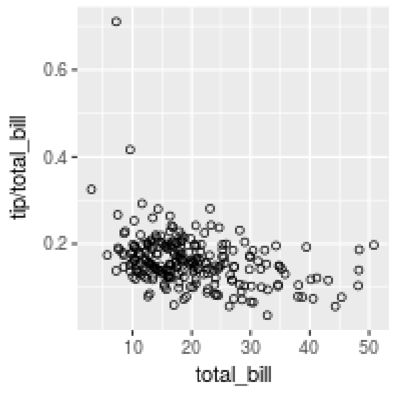
Sample data

We will use the `tips` dataset from the `reshape2` package.

```
library(reshape2)

# Look at first few rows

head(tips)

#>   total_bill  tip    sex smoker day   time size

#> 1      16.99 1.01 Female     No Sun Dinner    2

#> 2      10.34 1.66   Male     No Sun Dinner    3

#> 3      21.01 3.50   Male     No Sun Dinner    3

#> 4      23.68 3.31   Male     No Sun Dinner    2

#> 5      24.59 3.61 Female     No Sun Dinner    4

#> 6      25.29 4.71   Male     No Sun Dinner    4
```

This is a scatterplot of the tip percentage by total bill size.

```
library(ggplot2)

sp <- ggplot(tips, aes(x=total_bill, y=tip/total_bill)) + geom_point(shape=1)

sp
```

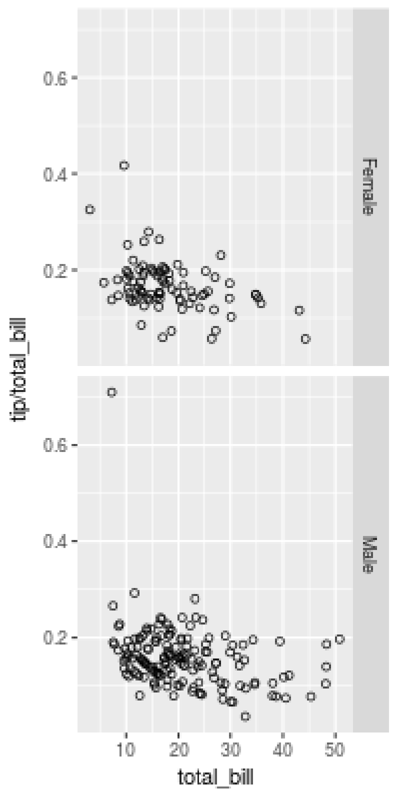


facet_grid

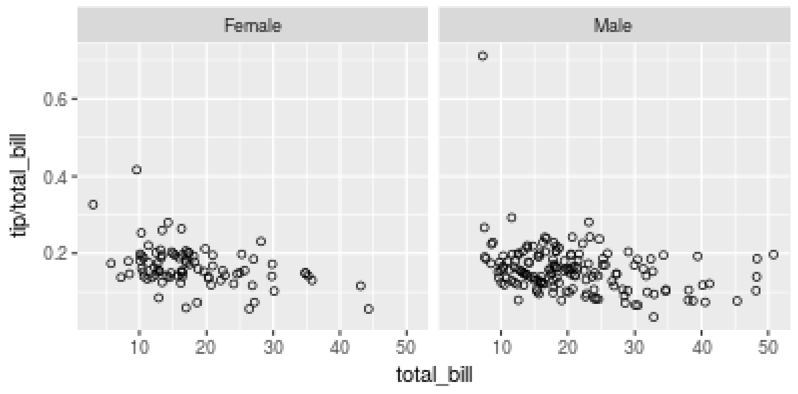
The data can be split up by one or two variables that vary on the horizontal and/or vertical direction.

This is done by giving a formula to `facet_grid()`, of the form `vertical ~ horizontal`.

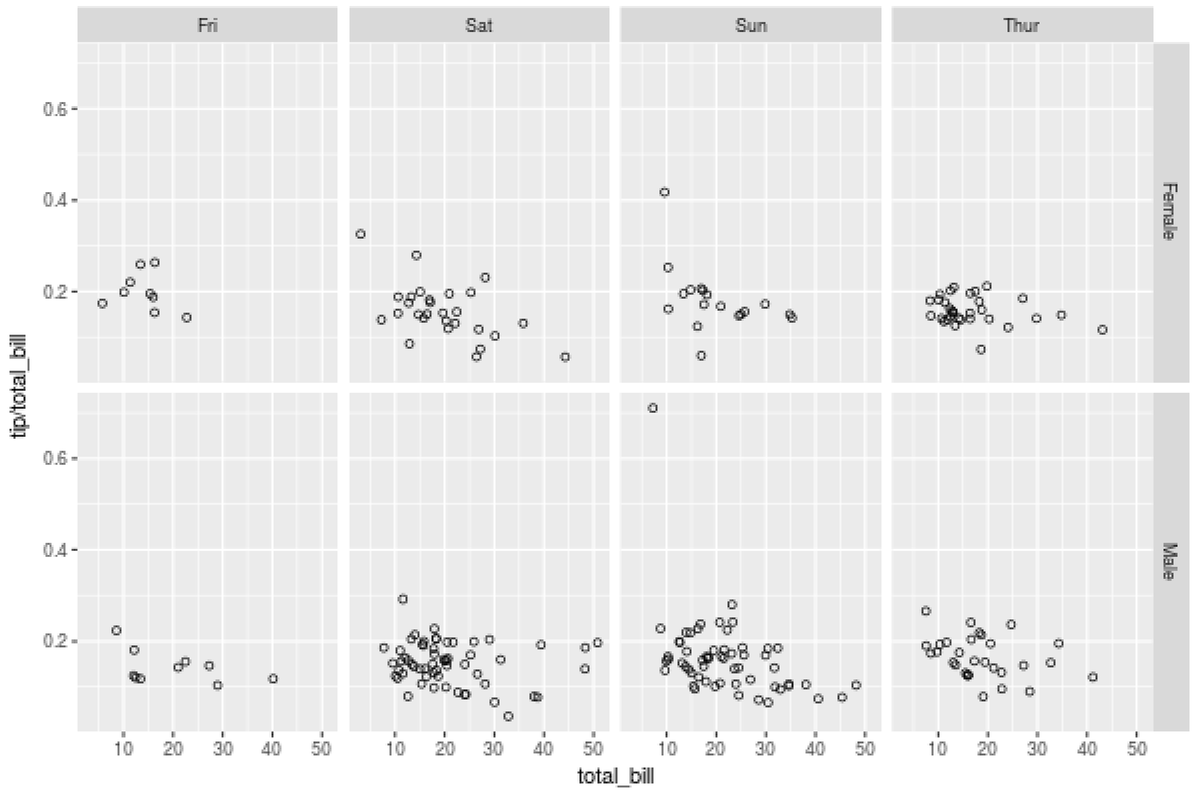
```r
# Divide by levels of "sex", in the vertical direction

sp + facet_grid(sex ~ .)
```



```r
# Divide by levels of "sex", in the horizontal direction

sp + facet_grid(. ~ sex)
```



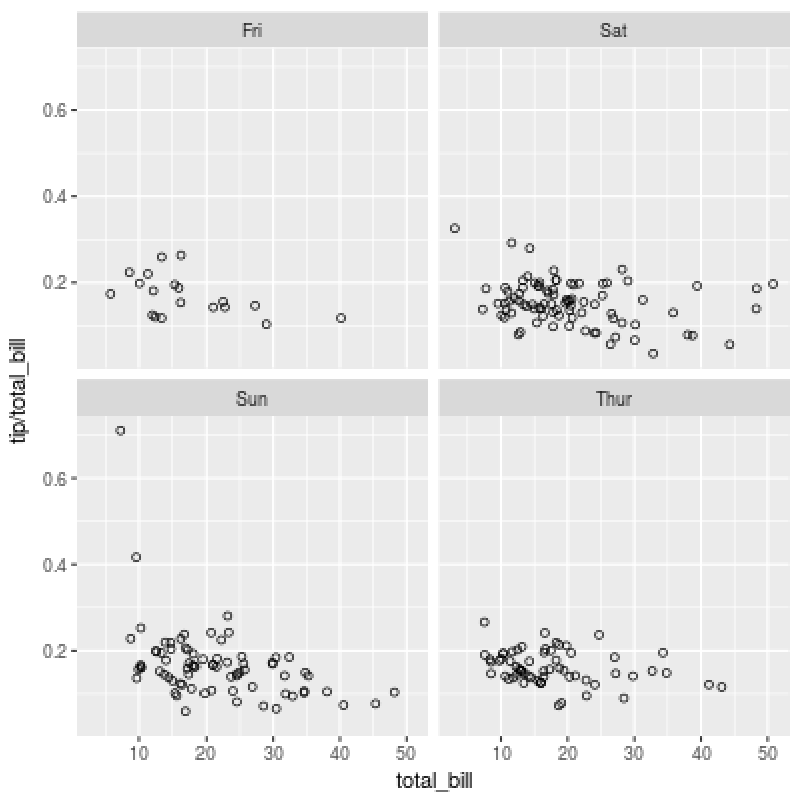
```
# Divide with "sex" vertical, "day" horizontal

sp + facet_grid(sex ~ day)
```



facet_wrap

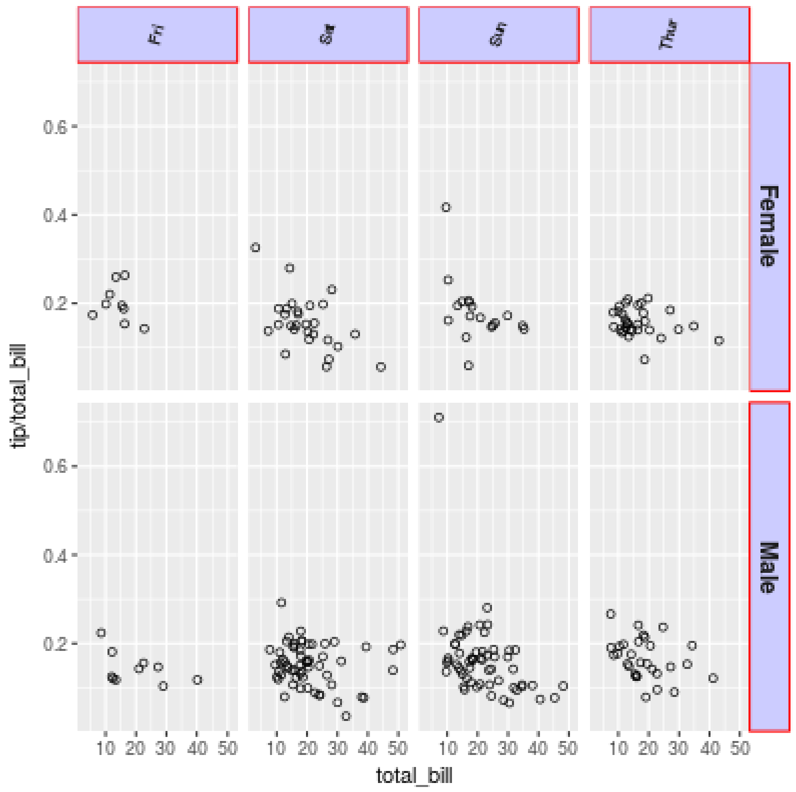
Instead of faceting with a variable in the horizontal or vertical direction, facets can be placed next to each other, wrapping with a certain number of columns or rows. The label for each plot will be at the top of the plot.

```
# Divide by day, going horizontally and wrapping with 2 columns

sp + facet_wrap( ~ day, ncol=2)
```



Modifying facet label appearance

```
sp + facet_grid(sex ~ day) +
    theme(strip.text.x = element_text(size=8, angle=75),
          strip.text.y = element_text(size=12, face="bold"),
          strip.background = element_rect(colour="red", fill="#CCCCFF"))
```



Modifying facet label text

There are a few different ways of modifying facet labels. The simplest way is to provide a named vector that maps original names to new names. To map the levels of `sex` from Female==>Women, and Male==>Men:

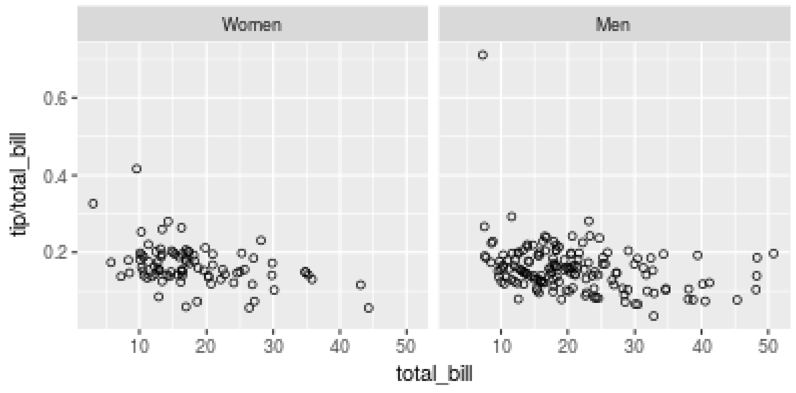
```
labels <- c(Female = "Women", Male = "Men")

sp + facet_grid(. ~ sex, labeller=labeller(sex = labels))
```

Another way is to modify the data frame so that the data contains the desired labels:

```
tips2 <- tips

levels(tips2$sex)[levels(tips2$sex)=="Female"] <- "Women"

levels(tips2$sex)[levels(tips2$sex)=="Male"]   <- "Men"

head(tips2, 3)

#>   total_bill  tip   sex smoker day   time size

#> 1      16.99 1.01 Women     No Sun Dinner    2

#> 2      10.34 1.66   Men     No Sun Dinner    3

#> 3      21.01 3.50   Men     No Sun Dinner    3



# Both of these will give the same output:
```

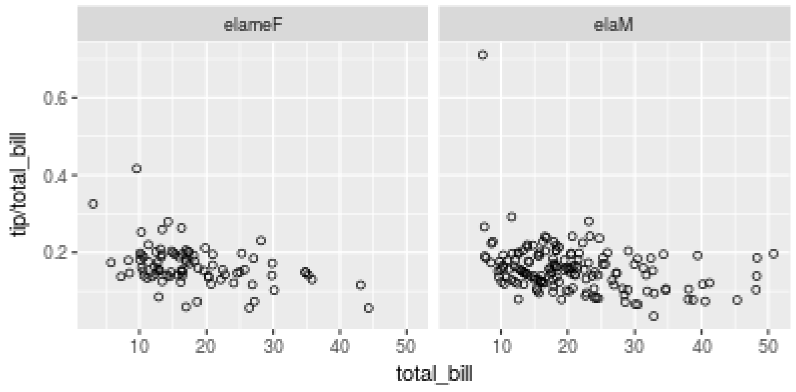
```
sp2 <- ggplot(tips2, aes(x=total_bill, y=tip/total_bill)) + geom_point(shape=1)

sp2 + facet_grid(. ~ sex)
```

Both of these will give the same result:



`labeller()` can use any function that takes a character vector as input and returns a character vector as output. For example, the `capitalize` function from the Hmisc package will capitalize the first letters of strings. We can also define our own custom functions, like this one, which reverses strings:

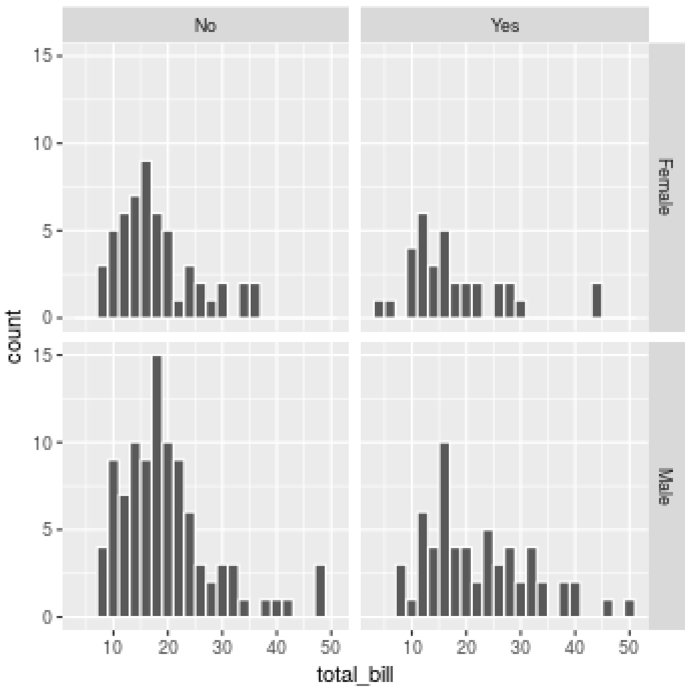
```
# Reverse each strings in a character vector

reverse <- function(strings) {

    strings <- strsplit(strings, "")

    vapply(strings, function(x) {

        paste(rev(x), collapse = "")

    }, FUN.VALUE = character(1))

}



sp + facet_grid(. ~ sex, labeller=labeller(sex = reverse))
```

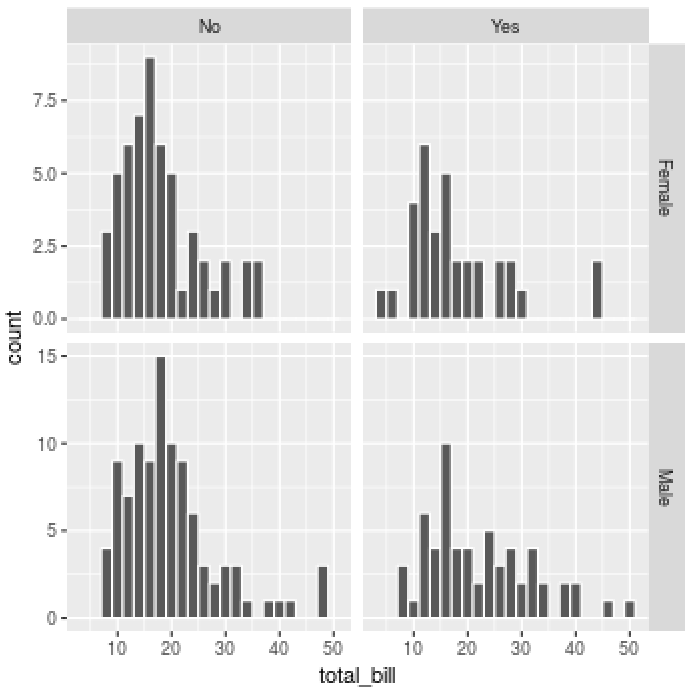


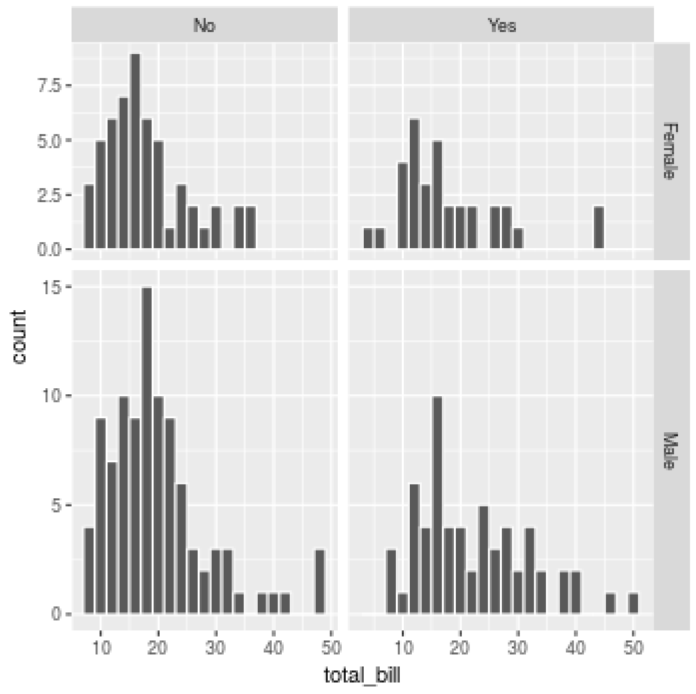
Free scales

Normally, the axis scales on each graph are **fixed**, which means that they have the same size and range. They can be made independent, by setting `scales` to `free`, `free_x`, or `free_y`.

```r
# A histogram of bill sizes

hp <- ggplot(tips, aes(x=total_bill)) + geom_histogram(binwidth=2,colour="white")



# Histogram of total_bill, divided by sex and smoker

hp + facet_grid(sex ~ smoker)



# Same as above, with scales="free_y"

hp + facet_grid(sex ~ smoker, scales="free_y")



# With panels that have the same scaling, but different range (and therefore different
physical sizes)

hp + facet_grid(sex ~ smoker, scales="free", space="free")
```


No
Yes
Female
Male
count
total_bill
0
5
10
15
10
20
30
40
50


No
Yes
Female
Male
count
total_bill
0.0
2.5
5.0
7.5
0
5
10
15
10
20
30
40
50



Also see: https://ggplot2-book.org/facet.html

Multiple graphs on one page (ggplot2)

Problem

You want to put multiple graphs on one page.

Solution

The easy way is to use the `multiplot` function, defined at the bottom of this page. If it isn't suitable for your needs, you can copy and modify it.

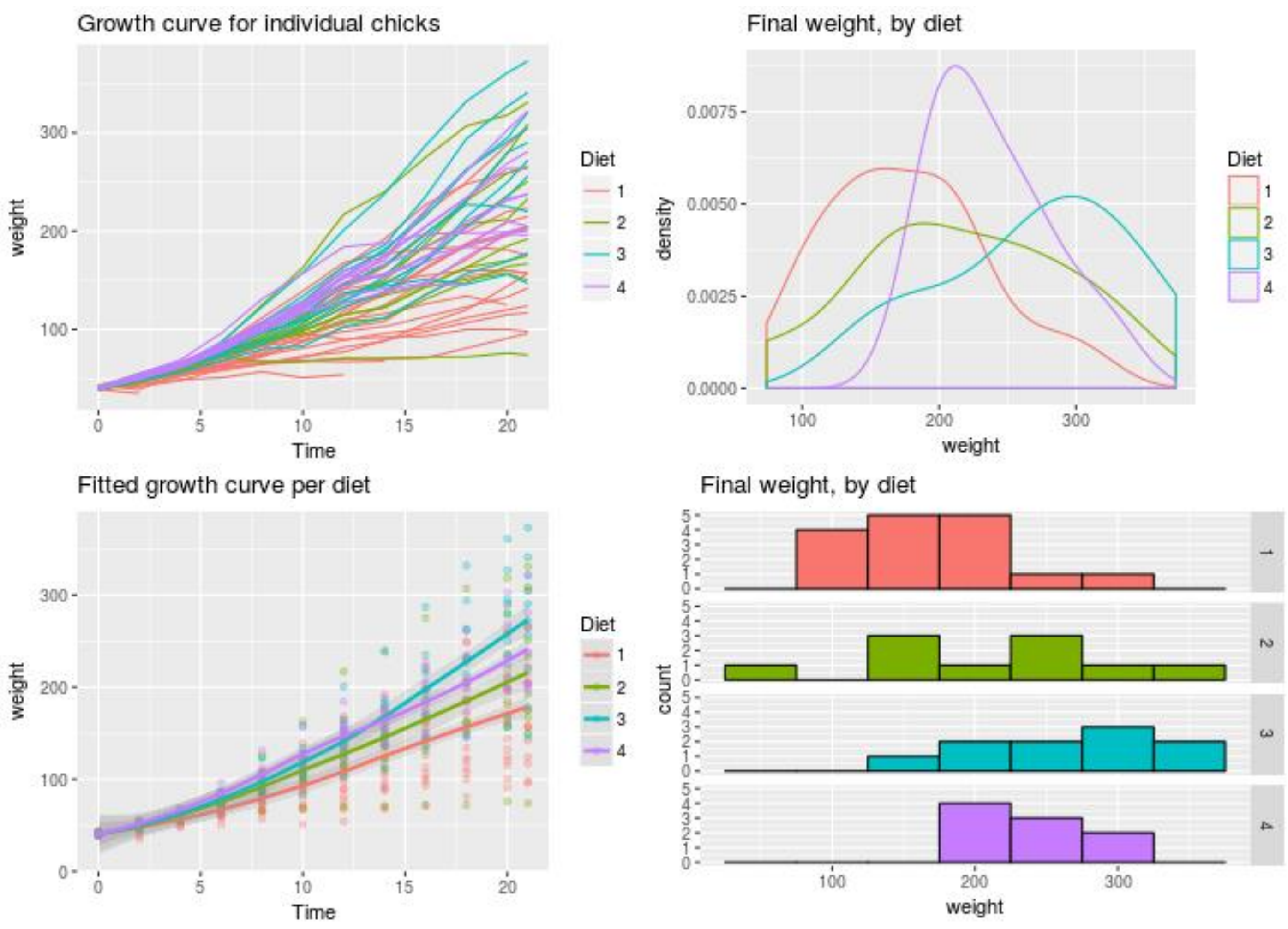
First, set up the plots and store them, but don't render them yet. The details of these plots aren't important; all you need to do is store the plot objects in variables.

```
library(ggplot2)


# This example uses the ChickWeight dataset, which comes with ggplot2

# First plot

p1 <- ggplot(ChickWeight, aes(x=Time, y=weight, colour=Diet, group=Chick)) +

    geom_line() +

    ggtitle("Growth curve for individual chicks")
```

```r
# Second plot
p2 <- ggplot(ChickWeight, aes(x=Time, y=weight, colour=Diet)) +
    geom_point(alpha=.3) +
    geom_smooth(alpha=.2, size=1) +
    ggtitle("Fitted growth curve per diet")

# Third plot
p3 <- ggplot(subset(ChickWeight, Time==21), aes(x=weight, colour=Diet)) +
    geom_density() +
    ggtitle("Final weight, by diet")

# Fourth plot
p4 <- ggplot(subset(ChickWeight, Time==21), aes(x=weight, fill=Diet)) +
    geom_histogram(colour="black", binwidth=50) +
    facet_grid(Diet ~ .) +
    ggtitle("Final weight, by diet") +
    theme(legend.position="none")        # No legend (redundant in this graph)
```

Once the plot objects are set up, we can render them with `multiplot`. This will make two columns of graphs:

```r
multiplot(p1, p2, p3, p4, cols=2)
#> `geom_smooth()` using method = 'loess'
```



multiplot function

This is the definition of `multiplot`. It can take any number of plot objects as arguments, or if it can take a list of plot objects passed to `plotlist`.

```
# Multiple plot function

#

# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)

# - cols:   Number of columns in layout

# - layout: A matrix specifying the layout. If present, 'cols' is ignored.

#

# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),

# then plot 1 will go in the upper left, 2 will go in the upper right, and

# 3 will go all the way across the bottom.

#

multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
```

```r
  library(grid)


  # Make a list from the ... arguments and plotlist

  plots <- c(list(...), plotlist)


  numPlots = length(plots)


  # If layout is NULL, then use 'cols' to determine layout

  if (is.null(layout)) {

    # Make the panel

    # ncol: Number of columns of plots

    # nrow: Number of rows needed, calculated from # of cols

    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),

                    ncol = cols, nrow = ceiling(numPlots/cols))

  }


 if (numPlots==1) {

    print(plots[[1]])


  } else {

    # Set up the page

    grid.newpage()

    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))


    # Make each plot, in the correct location

    for (i in 1:numPlots) {

      # Get the i,j matrix positions of the regions that contain this subplot

      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))


      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,

                                      layout.pos.col = matchidx$col))

    }
    }
  }
```