Chapter 5: Data Summarization, Manipulation with dplyr

M Affouf

1/8/2018

part1

Data Summarization

- Basic statistical summarization
  - `mean(x)`: takes the mean of x
  - `sd(x)`: takes the standard deviation of x
  - `median(x)`: takes the median of x
  - `quantile(x)`: displays sample quantities of x. Default is min, IQR, max
  - `range(x)`: displays the range. Same as c(min(x), max(x))

Some examples

We can use the `mtcars` and Charm City Circulator datasets to explore different ways of summarizing data.

```
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1
```

Statistical summarization

```
mean(mtcars$hp)
```

```
## [1] 146.6875
```

```
quantile(mtcars$hp)
```

```
##    0%   25%   50%   75%  100%
##  52.0  96.5 123.0 180.0 335.0
```

Statistical summarization

```r
median(mtcars$wt)
```

```
## [1] 3.325
```

```r
quantile(mtcars$wt, probs = 0.6)
```

```
##   60%
## 3.44
```

Statistical summarization

Note that many of these functions have additional inputs regarding missing data, typically requiring the `na.rm` argument.

```
x = c(1,5,7,NA,4,2, 8,10,45,42)
mean(x)
```

```
## [1] NA
```

```
mean(x,na.rm=TRUE)
```

```
## [1] 13.77778
```

```
quantile(x,na.rm=TRUE)
```

```
##   0%  25%  50%  75% 100%
##    1    4    7   10   45
```

Data Summarization on matrices/data frames

- Basic statistical summarization
  - `rowMeans(x)`: takes the means of each row of x
  - `colMeans(x)`: takes the means of each column of x
  - `rowSums(x)`: takes the sum of each row of x
  - `colSums(x)`: takes the sum of each column of x
  - `summary(x)`: for data frames, displays the quantile information

Charm City Circulator data

Please download the Charm City Circulator data:

```
circ = read.csv("Charm_City_Circulator_Ridership.csv",
                header=TRUE,as.is=TRUE)
```

Subsetting to specific columns

Let's just take columns that represent average ridership:

```
library(dplyr,quietly = TRUE)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
circ2 = select(circ, date, day, ends_with("Average"))
```

column and row means

```
avgs = select(circ2, ends_with("Average"))
colMeans(avgs,na.rm=TRUE)
```

```
## orangeAverage purpleAverage  greenAverage bannerAverage
##     3033.1611     4016.9345     1957.7814      827.2685
```

```
circ2$daily = rowMeans(avgs,na.rm=TRUE)
head(circ2$daily)
```

```
## [1]  952.0  796.0 1211.5 1213.5 1644.0 1490.5
```

Summary

```
summary(circ2)
```

```
##       date                day             orangeAverage  pu
##  Length:1146        Length:1146        Min.   :   0   Mi
##  Class :character   Class :character   1st Qu.:2001   1s
##  Mode  :character   Mode  :character   Median :2968   Me
##                                        Mean   :3033   Me
##                                        3rd Qu.:4020   3r
##                                        Max.   :6926   Ma
##                                        NA's   :10     NA
##   greenAverage    bannerAverage         daily
##  Min.   :   0   Min.   :   0.0   Min.   :   0
##  1st Qu.:1491   1st Qu.: 632.5   1st Qu.:2097
##  Median :2079   Median : 763.0   Median :2846
##  Mean   :1958   Mean   : 827.3   Mean   :2878
##  3rd Qu.:2340   3rd Qu.: 945.9   3rd Qu.:3646
##  Max.   :5094   Max.   :4617.0   Max.   :6123
##  NA's   :661    NA's   :876      NA's   :10
```

Apply statements

You can apply more general functions to the rows or columns of a matrix or data frame, beyond the mean and sum.

```
apply(X, MARGIN, FUN, ...)
```

X : an array, including a matrix.
MARGIN : a vector giving the subscripts which the function will be applied over. E.g., for a matrix 1 indicates rows, 2 indicates columns, c(1, 2) indicates rows and columns. Where X has named dimnames, it can be a character vector selecting dimension names.
FUN : the function to be applied: see 'Details'.
... : optional arguments to FUN.

Apply statements

```
apply(avgs,2,mean,na.rm=TRUE) # column means
```

```
##  orangeAverage purpleAverage  greenAverage bannerAverage
##      3033.1611      4016.9345      1957.7814       827.2685
```

```
apply(avgs,2,sd,na.rm=TRUE) # columns sds
```

```
##  orangeAverage purpleAverage  greenAverage bannerAverage
##      1227.5779      1406.6544       592.8969       436.0487
```

```
apply(avgs,2,max,na.rm=TRUE) # column maxs
```

```
##  orangeAverage purpleAverage  greenAverage bannerAverage
##         6926.5         8089.5         5094.0         4617.0
```

Other Apply Statements

- `tapply()`: 'table' apply
- `lapply()`: 'list' apply [tomorrow]
- `sapply()`: 'simple' apply [tomorrow]
- Other less used ones...

See more details here: http://nsaunders.wordpress.com/2010/08/20/a-brief-introduction-to-apply-in-r/

tapply()

From the help file: "Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors."

```
tapply(X, INDEX, FUN = NULL, ..., simplify = TRUE)
```

Simply put, you can apply function `FUN` to `X` within each categorical level of `INDEX`. It is very useful for assessing properties of continuous data by levels of categorical data.

tapply()

For example, we can estimate the highest average daily ridership for each day of the week in 1 line in the Circulator dataset.

```
tapply(circ2$daily, circ2$day, max, na.rm=TRUE)
```

```
##    Friday    Monday  Saturday    Sunday  Thursday   Tues
##   5600.75   5002.25   6123.00   3980.25   4820.50   485
```

Data Summarization

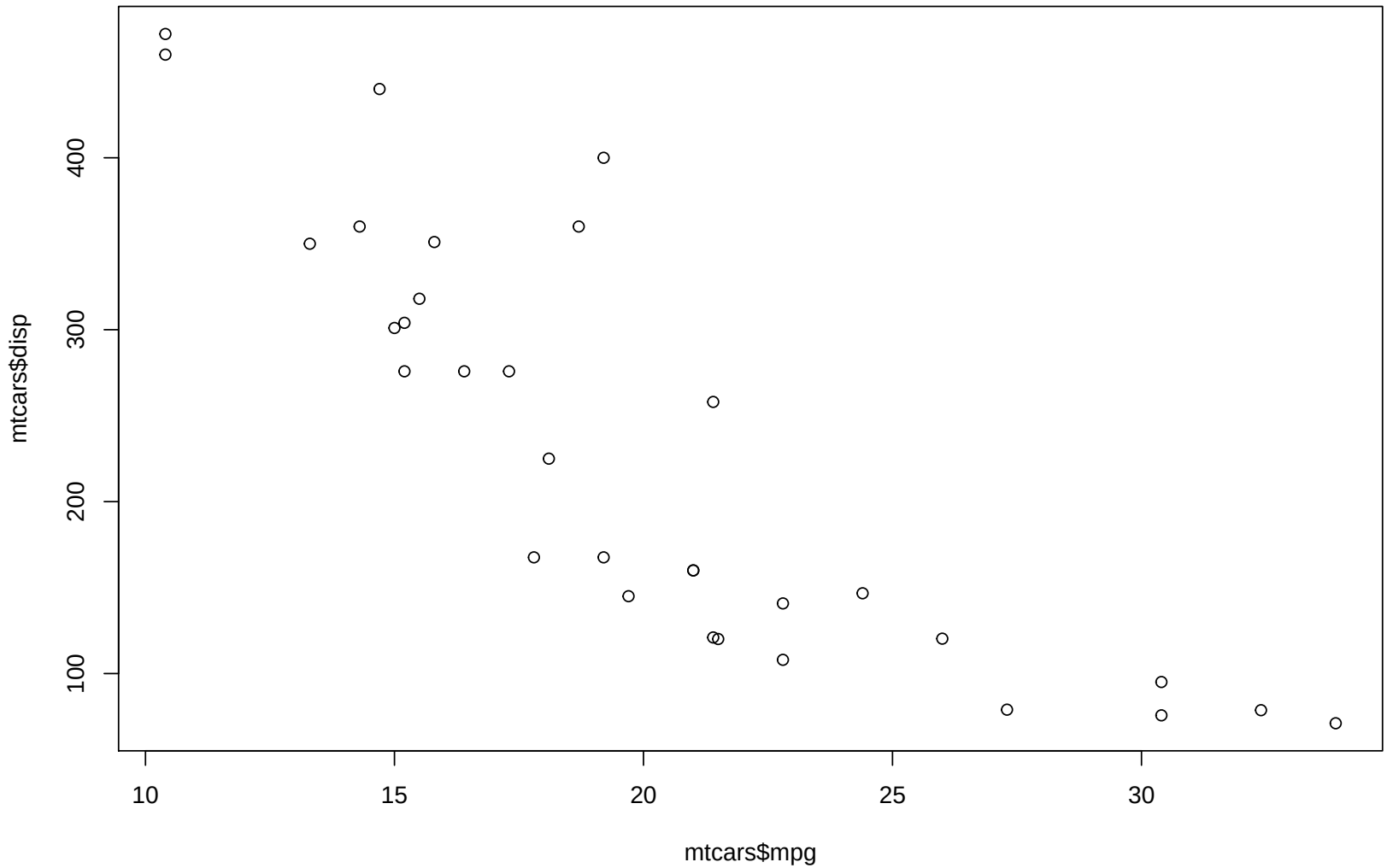
- Basic summarization plots
  - `plot(x,y)`: scatterplot of x and y
  - `boxplot(y~x)`: boxplot of y against levels of x
  - `hist(x)`: histogram of x
  - `density(X)`: kernel density plot of x

Basic Plots

Plotting is an important component of exploratory data analysis. We will review some of the more useful and informative plots here. We will go over formatting and making plots look nicer in additional lectures.

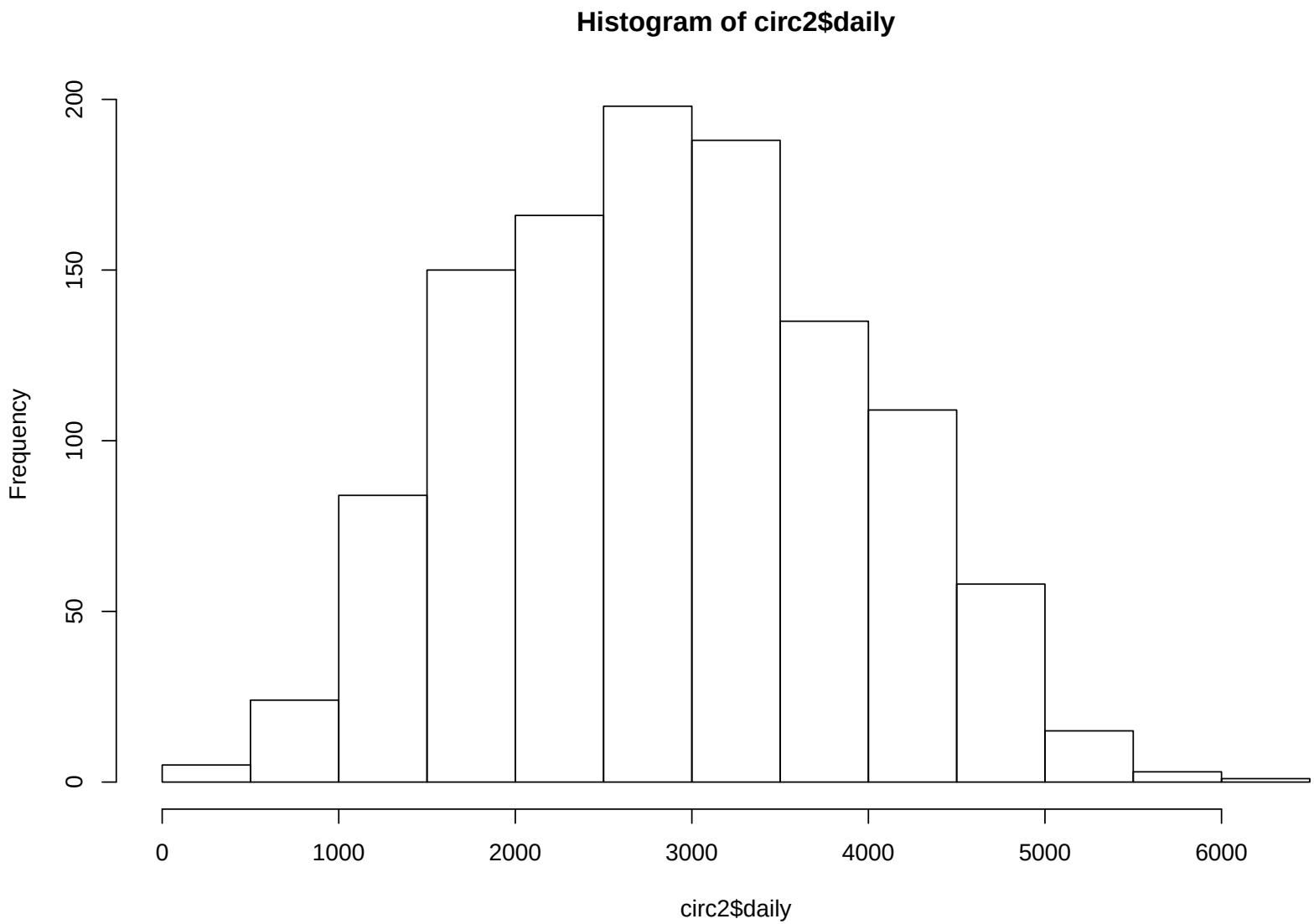
Scatterplot

```
plot(mtcars$mpg, mtcars$disp)
```



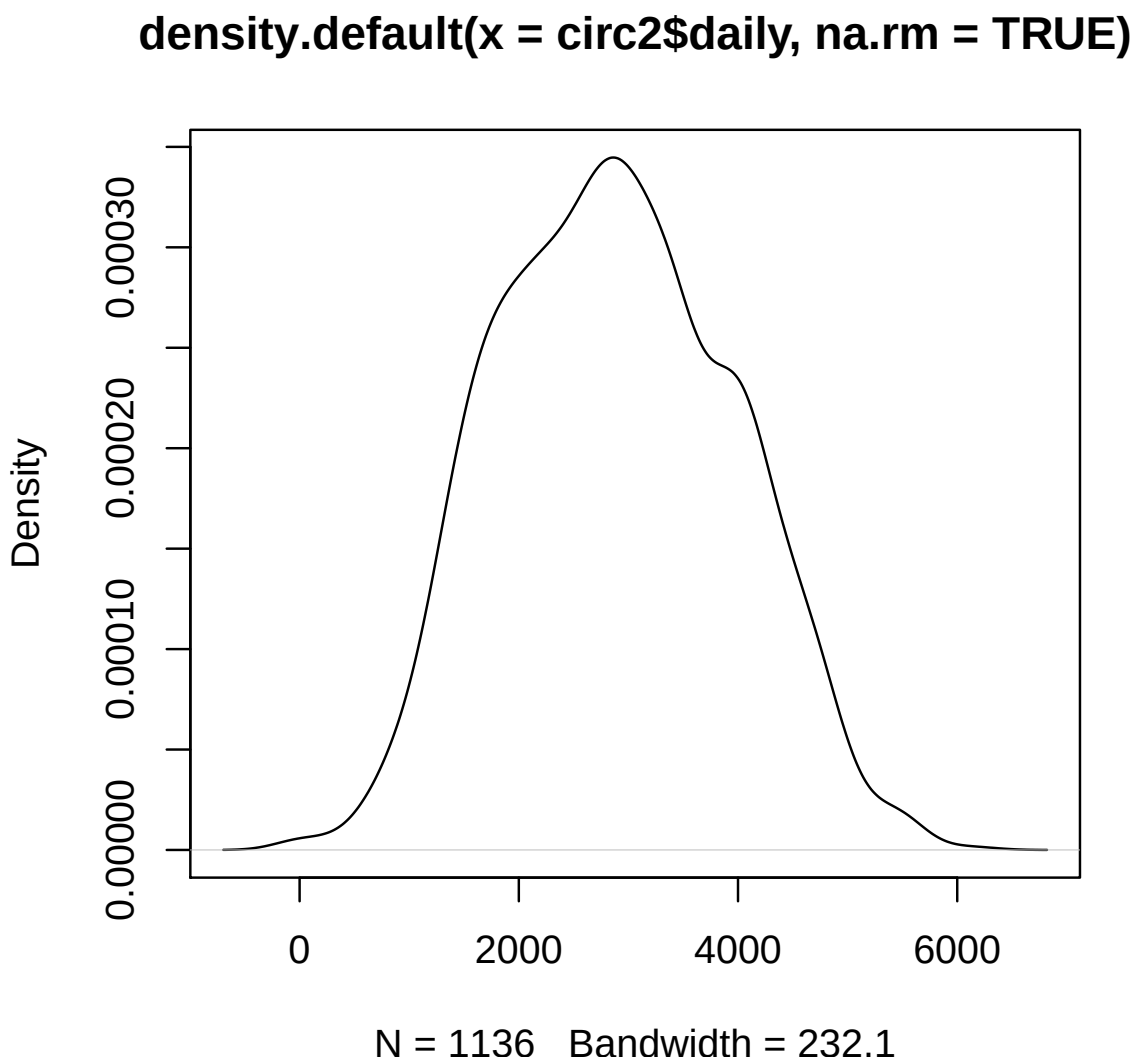
Histograms

```
hist(circ2$daily)
```



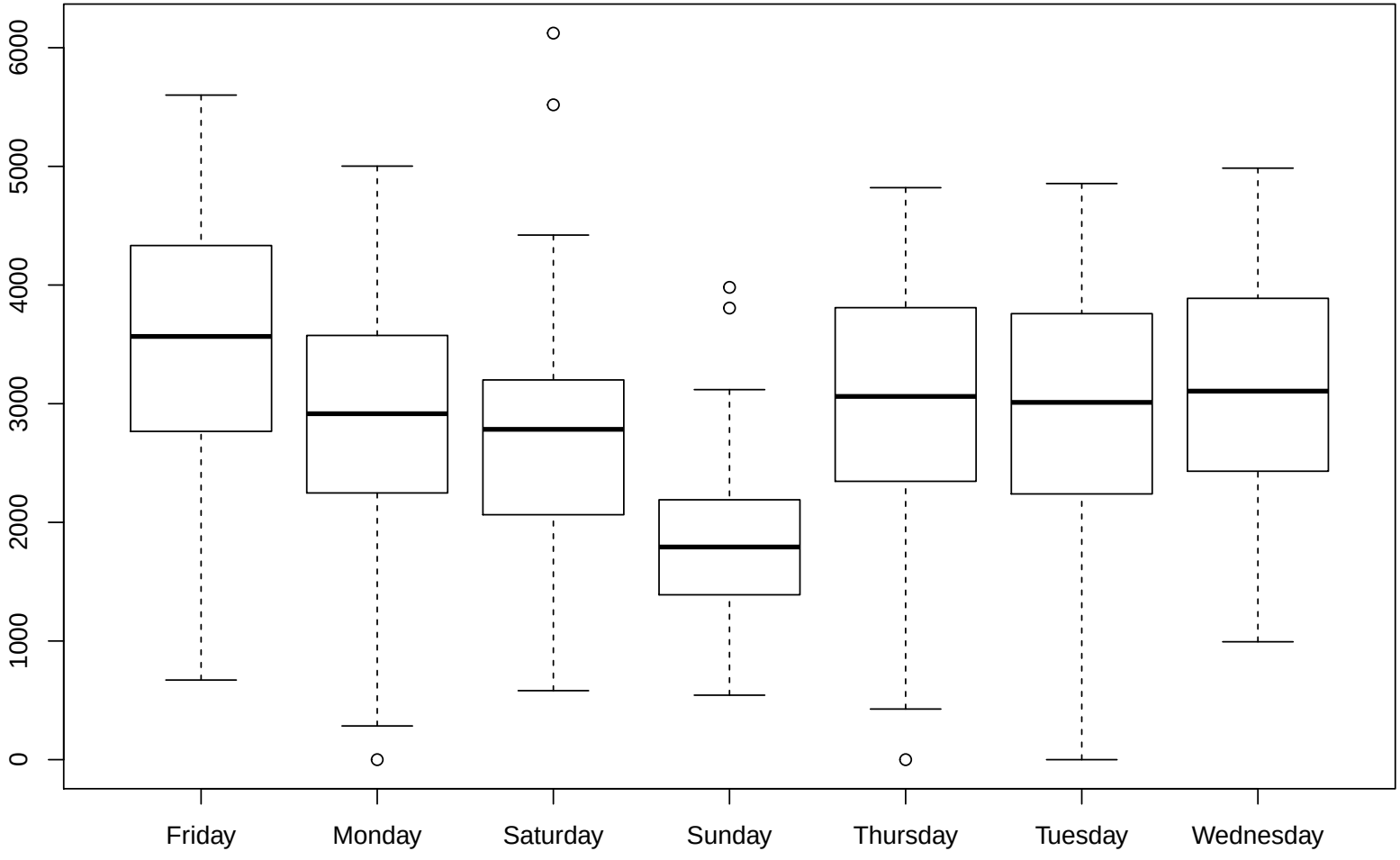
Density

```
## plot(density(circ2$daily))
plot(density(circ2$daily,na.rm=TRUE))
```



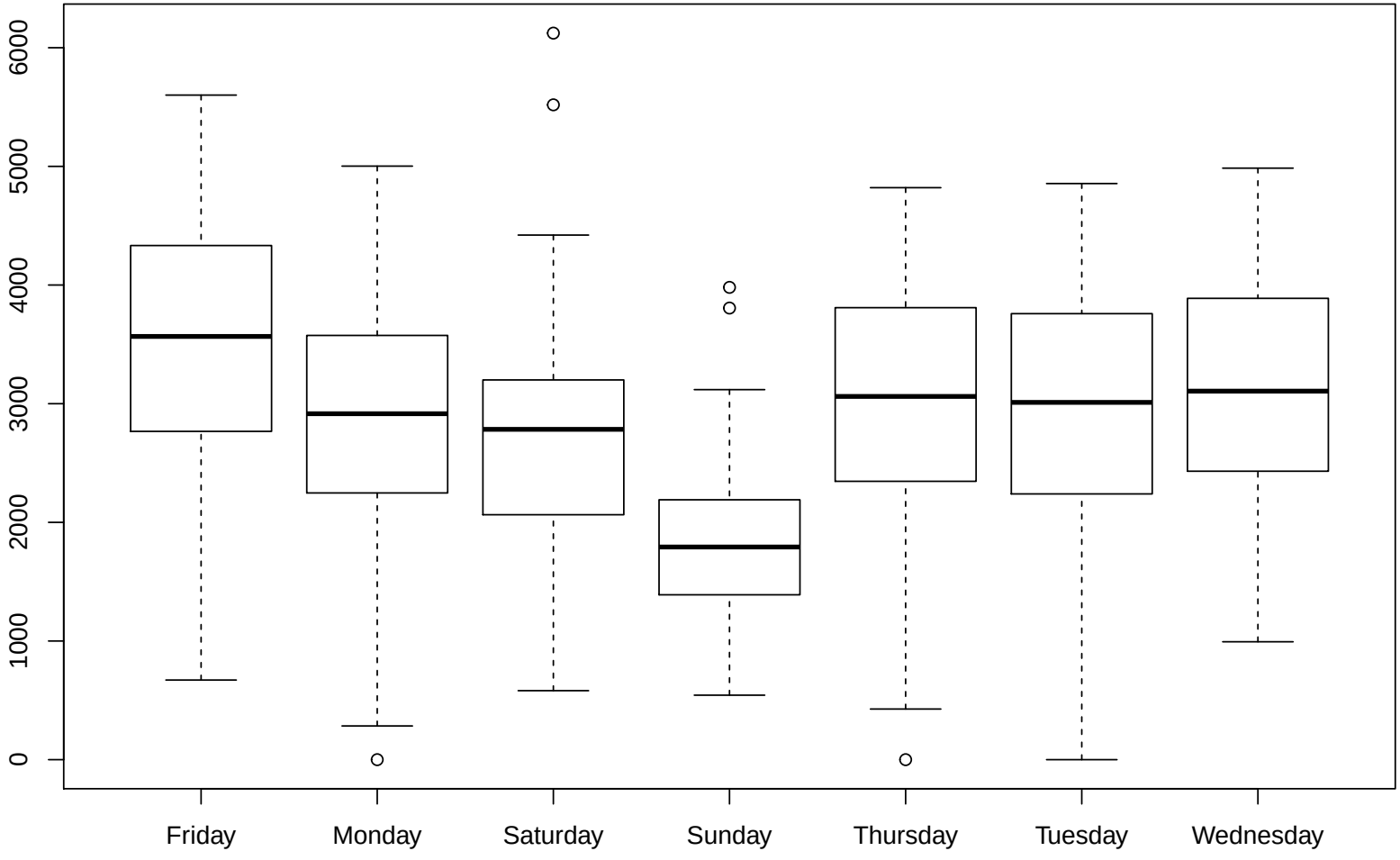
Boxplots

```
boxplot(circ2$daily ~ circ2$day)
```



Boxplots

```
boxplot(daily ~ day, data=circ2)
```

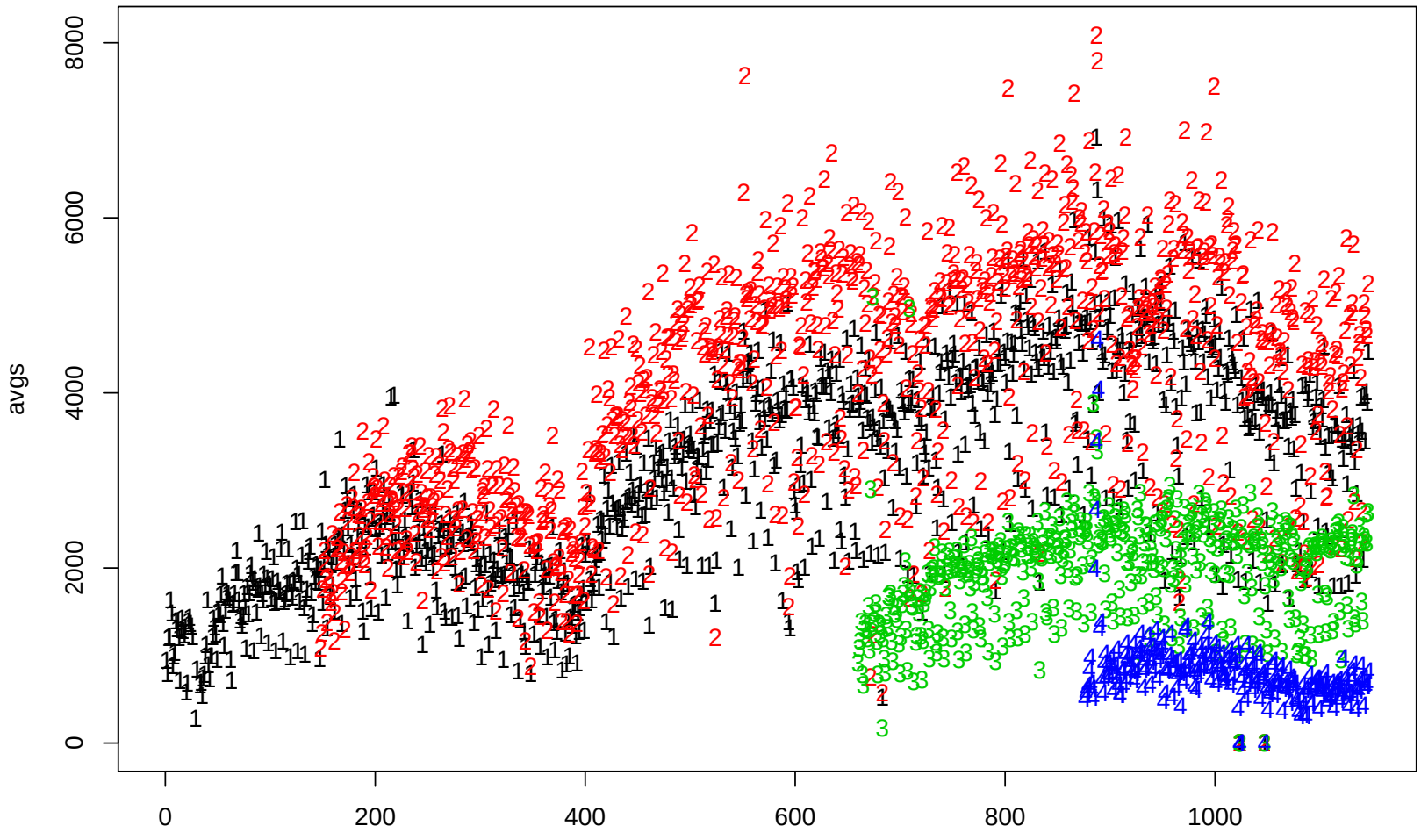


Data Summarization for data.frames

- Basic summarization plots
  - `matplot(x,y)`: scatterplot of two matrices, `x` and `y`
  - `pairs(x,y)`: plots pairwise scatter plots of matrices `x` and `y`, column by column

Matrix plot

```
matplot(avgs)
```



Part 2

Agenda

- Summaries with the aggregate() function
- Standard graphics

Getting started: birthwt data set

- We're going to start by operating on the `birthwt` dataset from the MASS library
- Let's get it loaded and see what we're working with

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
str(birthwt)
```

```
## 'data.frame':    189 obs. of  10 variables:
##  $ low   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ age   : int  19 33 20 21 18 21 22 17 29 26 ...
##  $ lwt   : int  182 155 105 108 107 124 118 103 123 113
```

Renaming the variables

- The dataset doesn't come with very descriptive variable names
- Let's get better column names (use `help(birthwt)` to understand the variables and come up with better names)

```
colnames(birthwt)
```

```
##  [1] "low"   "age"   "lwt"   "race"  "smoke" "ptl"   "ht
##  [9] "ftv"   "bwt"
```

```
# The default names are not very descriptive

colnames(birthwt) <- c("birthwt.below.2500", "mother.age",
    "race", "mother.smokes", "previous.prem.labor", "hypert
    "physician.visits", "birthwt.grams")

# Better names!
```

Renaming the factors

- All the factors are currently represented as integers
- Let's use the `transform()` and `mapvalues()` functions to convert variables to factors and give the factors more meaningful levels

```
library(plyr)
```

```
## ------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cau
## If you need functions from both plyr and dplyr, please l
## library(plyr); library(dplyr)

## ------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
```

Summary of the data

- Now that things are coded correctly, we can look at an overall summary

```
summary(birthwt)
```

```
##  birthwt.below.2500   mother.age     mother.weight       r
##  no :130            Min.   :14.00   Min.   : 80.0   blac
##  yes: 59            1st Qu.:19.00   1st Qu.:110.0   othe
##                     Median :23.00   Median :121.0   whit
##                     Mean   :23.24   Mean   :129.8
##                     3rd Qu.:26.00   3rd Qu.:140.0
##                     Max.   :45.00   Max.   :250.0
##  mother.smokes previous.prem.labor hypertension uterine
##  no :115       Min.   :0.0000      no :177      no :161
##  yes: 74       1st Qu.:0.0000      yes: 12      yes: 28
##                Median :0.0000
##                Mean   :0.1958
##                3rd Qu.:0.0000
```

A simple table

- Let's use the `tapply()` function to see what the average birthweight looks like when broken down by race and smoking status

```
with(birthwt, tapply(birthwt.grams, INDEX = list(race, moth
```

```
##                no       yes
## black    2854.500 2504.000
## other    2815.782 2757.167
## white    3428.750 2826.846
```

What if we wanted nicer looking output?

- Let's use the header `{r, results='asis'}`, along with the `kable()` function from the `knitr` library

```
library(knitr)
bwt.tbl <- with(birthwt, tapply(birthwt.grams, INDEX = list
kable(bwt.tbl, format = "markdown")
```

	no	yes
black	2854.500	2504.000
other	2815.782	2757.167
white	3428.750	2826.846

- `kable()` outputs the table in a way that Markdown can read and nicely display
- Note: changing the CSS changes the table appearance

aggregate() function

- Let's first recall what `tapply()` does
- Command: `tapply(X, INDEX, FUN)`
  - Applies `FUN` to `X` grouped by factors in `INDEX`
- **`aggregate()`** performs a similar operation, but presents the results in a form that is at times more convenient
- There are many ways to call the `aggregate()` function
- Analog of `tapply` call: `aggregate(X, by, FUN)`
  - Here, `by` is exactly like `INDEX`

Example: tapply vs aggregate

```
library(MASS)
with(birthwt, tapply(birthwt.grams, INDEX = list(race, moth
```

```
##                no       yes
## black  2854.500 2504.000
## other  2815.782 2757.167
## white  3428.750 2826.846
```

```
with(birthwt, aggregate(birthwt.grams, by = list(race, moth
```

```
##   Group.1 Group.2        x
## 1   black      no 2854.500
## 2   other      no 2815.782
## 3   white      no 3428.750
## 4   black     yes 2504.000
## 5   other     yes 2757.167
## 6   white     yes 2826.846
```

Example: different syntax

- Here's a convenient alternative way to call `aggregate`
- It uses the R `formula` syntax, which we'll learn more about when we discuss regression

```
aggregate(birthwt.grams ~ race + mother.smokes, FUN=mean, d
```

```
##    race mother.smokes birthwt.grams
## 1 black            no      2854.500
## 2 other            no      2815.782
## 3 white            no      3428.750
## 4 black           yes      2504.000
## 5 other           yes      2757.167
## 6 white           yes      2826.846
```

- We'll see later that `aggregate` output can be more convenient for plotting

A closer look at low birth weight

```
weight.smoke.tbl <- with(birthwt, table(birthwt.below.2500,
weight.smoke.tbl
```

```
##                     mother.smokes
## birthwt.below.2500 no yes
##                 no  86  44
##                 yes 29  30
```

- The odds of low bwt among non-smoking mothers is

```
or.smoke.bwt <- (weight.smoke.tbl[2,2] / weight.smoke.tbl[1
or.smoke.bwt
```

```
## [1] 2.021944
```

- So the odds of low birth weight are 2 times higher when the mother smokes

continued...

- Is the mother's age correlated with birth weight?

```
with(birthwt, cor(birthwt.grams, mother.age))  # Calculate
```

```
## [1] 0.09031781
```

- Does this change when we account for smoking status?

```
with(birthwt, cor(birthwt.grams[mother.smokes == "yes"], mo
```

```
## [1] -0.1441649
```

```
with(birthwt, cor(birthwt.grams[mother.smokes == "no"], mot
```

```
## [1] 0.2014558
```

Faster way: by() function

- Think of the `by(data, INDICES, FUN)` function as a `tapply()` function that operates on data frames instead of just vectors
- When using `tapply(X, INDEX, FUN)`, `X` is generally a numeric vector
- To calculate correlations, we need to allow `X` to be a data frame or matrix

```
by(data = birthwt[c("birthwt.grams", "mother.age")],
   INDICES = birthwt["mother.smokes"],
   FUN = function(x) {cor(x[,1], x[,2])})
```

```
## mother.smokes: no
## [1] 0.2014558
## ------------------------------------------------------------
## mother.smokes: yes
## [1] -0.1441649
```

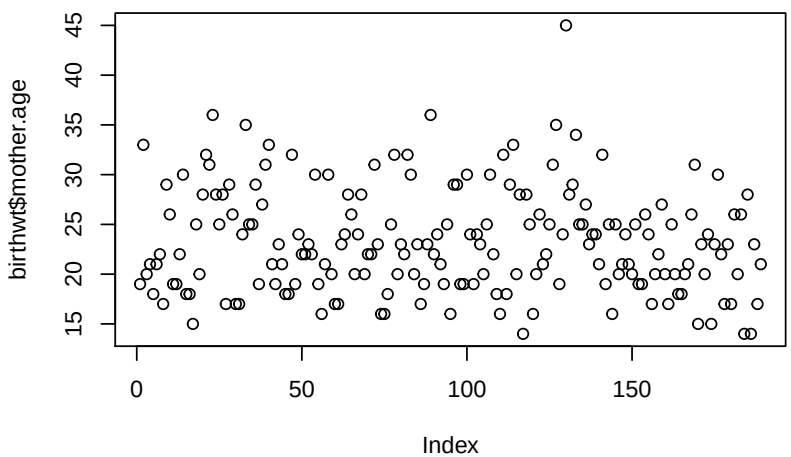
Standard graphics in R

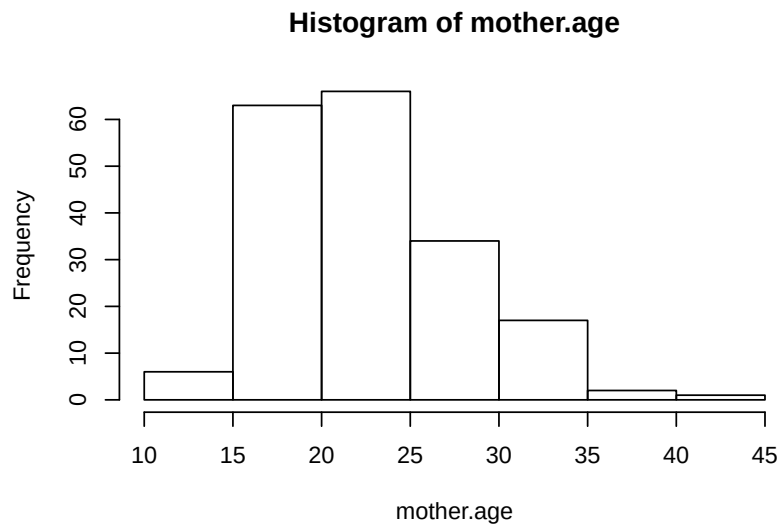
Single-variable plots

Let's continue with the `birthwt` data from the `MASS` library.

Here are some basic single-variable plots.

```
par(mfrow = c(2,2)) # Display plots in a single 2 x 2 figu
plot(birthwt$mother.age)
with(birthwt, hist(mother.age))
plot(birthwt$mother.smokes)
plot(birthwt$birthwt.grams)
```





(much) better graphics with ggplot2

Introduction to ggplot2

ggplot2 has a slightly steeper learning curve than the base graphics functions, but it also generally produces far better and more easily customizable graphics.

There are two basic calls in ggplot:

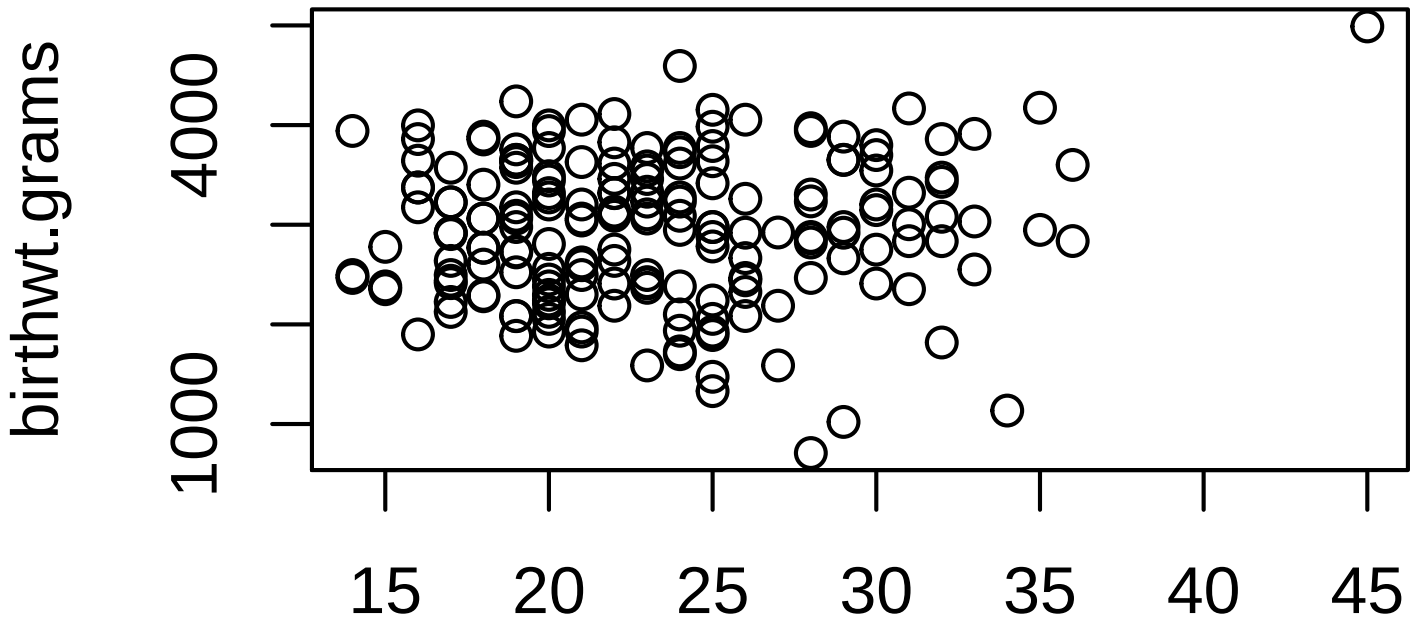
- `qplot(x, y, ..., data)`: a "quick-plot" routine, which essentially replaces the base `plot()`
- `ggplot(data, aes(x, y, ...), ...)`: defines a graphics object from which plots can be generated, along with *aesthetic mappings* that specify how variables are mapped to visual properties.

```
library(ggplot2)
```

plot vs qplot

Here's how the default scatterplots look in ggplot compared to the base graphics. We'll illustrate things by continuing to use the birthwt data from the `MASS` library.

```
with(birthwt, plot(mother.age, birthwt.grams))   # Base gra
```



ggplot function

The ggplot2 library comes with a dataset called diamonds. Let's look at it

```
dim(diamonds)
```

```
## [1] 53940    10
```

```
head(diamonds)
```

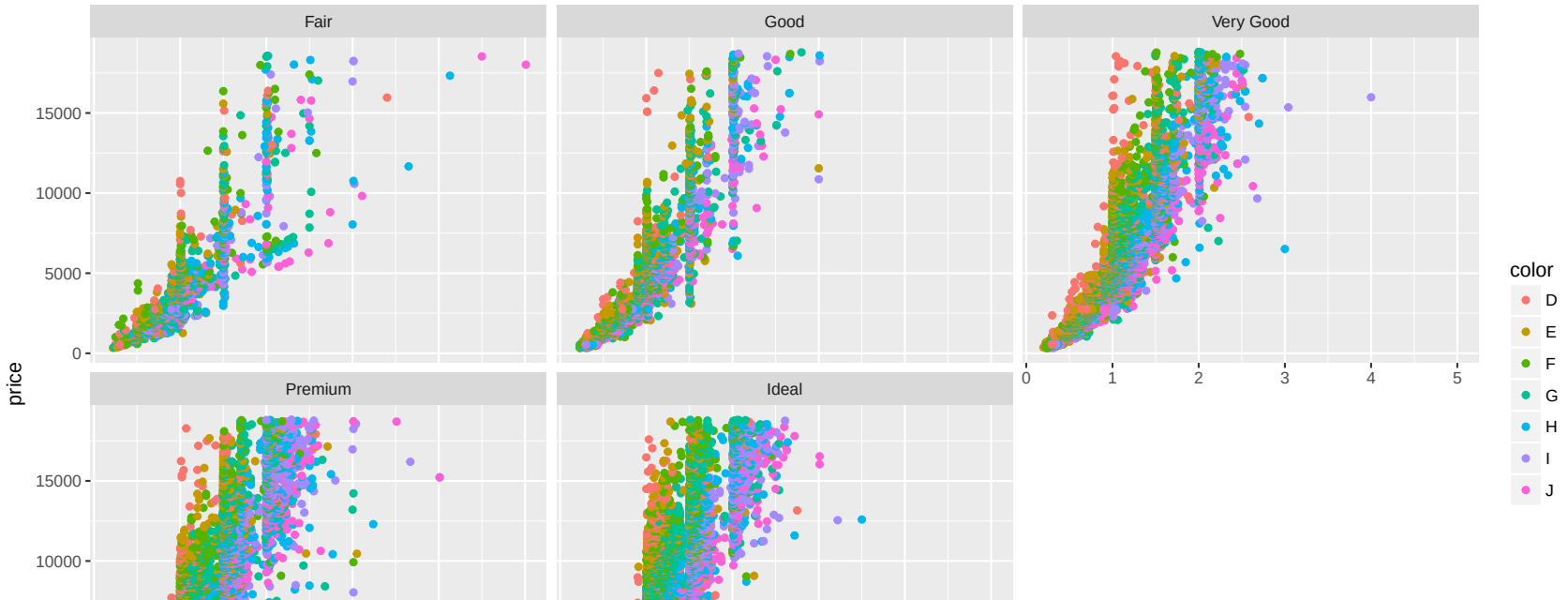
```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl>
## 1 0.230 Ideal     E     SI2      61.5  55.0   326  3.95
## 2 0.210 Premium   E     SI1      59.8  61.0   326  3.89
## 3 0.230 Good      E     VS1      56.9  65.0   327  4.05
## 4 0.290 Premium   I     VS2      62.4  58.0   334  4.20
## 5 0.310 Good      J     SI2      63.3  58.0   335  4.34
## 6 0.240 Very Good J     VVS2     62.8  57.0   336  3.94
```

Conditional plots

We can create plots showing the relationship between variables across different values of a factor. For instance, here's a scatterplot showing how diamond price varies with carat size, conditioned on color. It's created using the `facet_wrap(~ factor1 + factor2 + ... + factorn)` command.

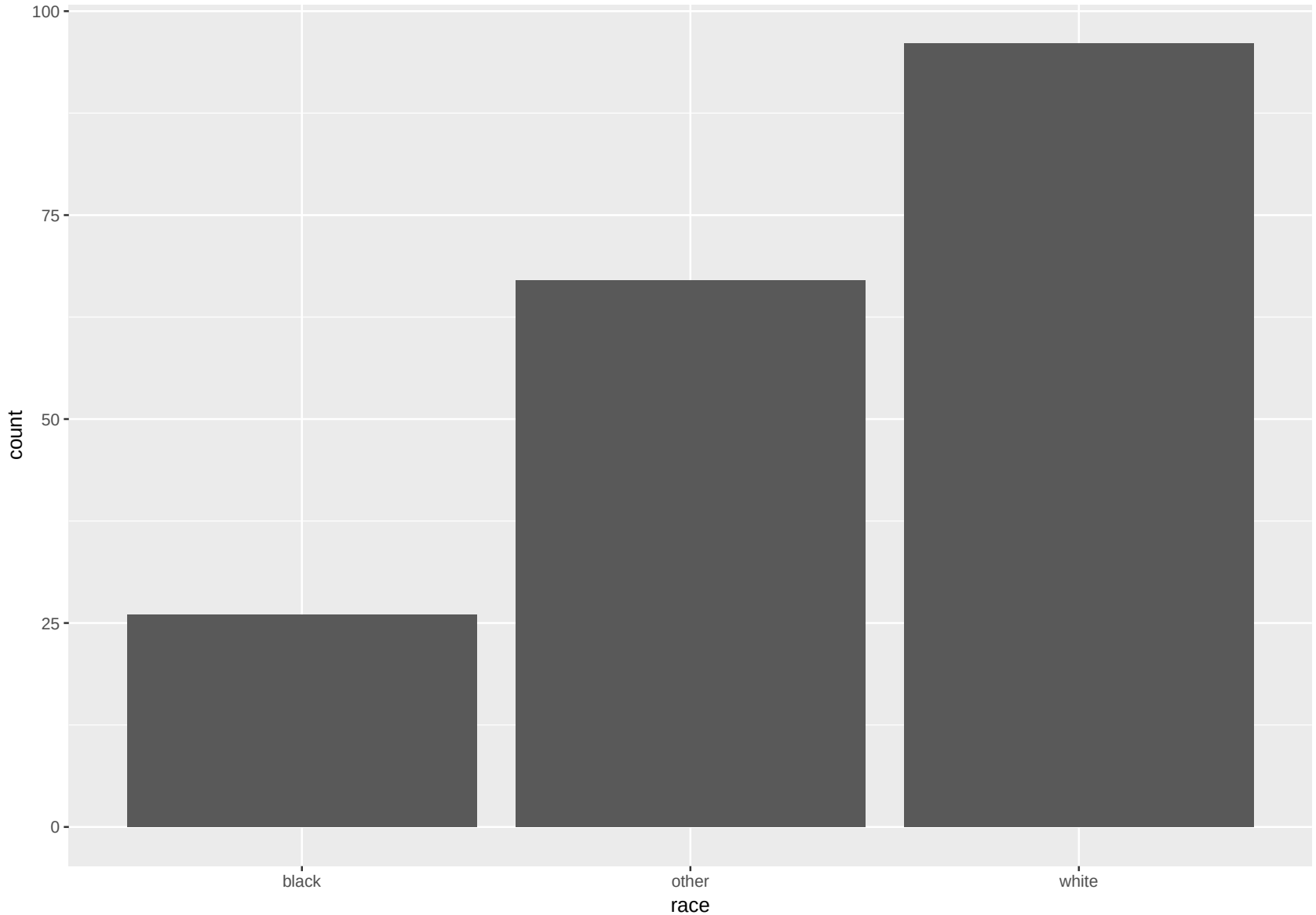
```
diamond.plot <- ggplot(data=diamonds, aes(x=carat, y=price,

diamond.plot + geom_point() + facet_wrap(~ cut)
```



A bar chart

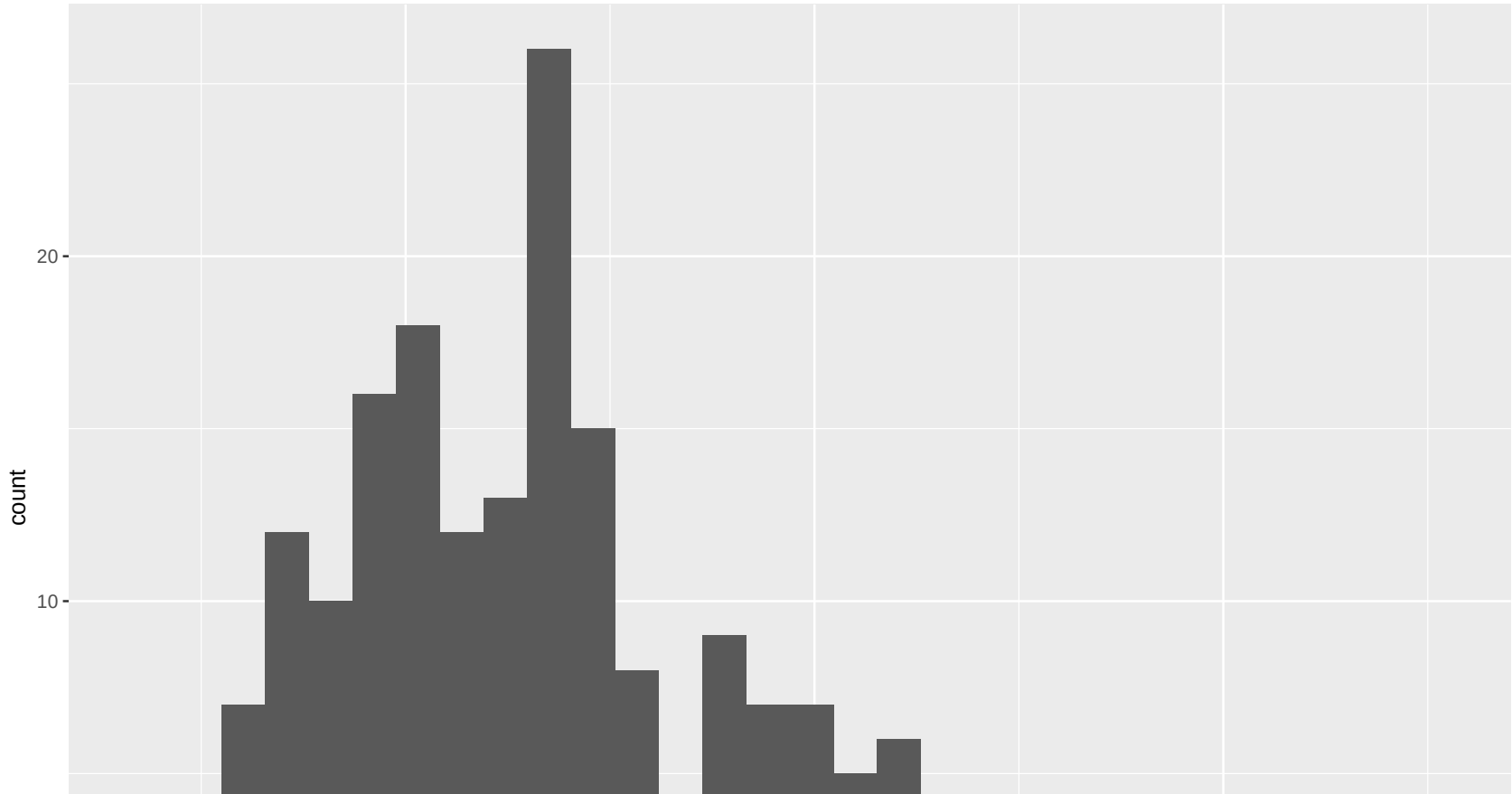
```
qplot(x = race, data = birthwt, geom = "bar")
```



Histograms and density plots

```
base.plot <- ggplot(birthwt, aes(x = mother.age)) +
  xlab("Mother's age")
base.plot + geom_histogram()
```

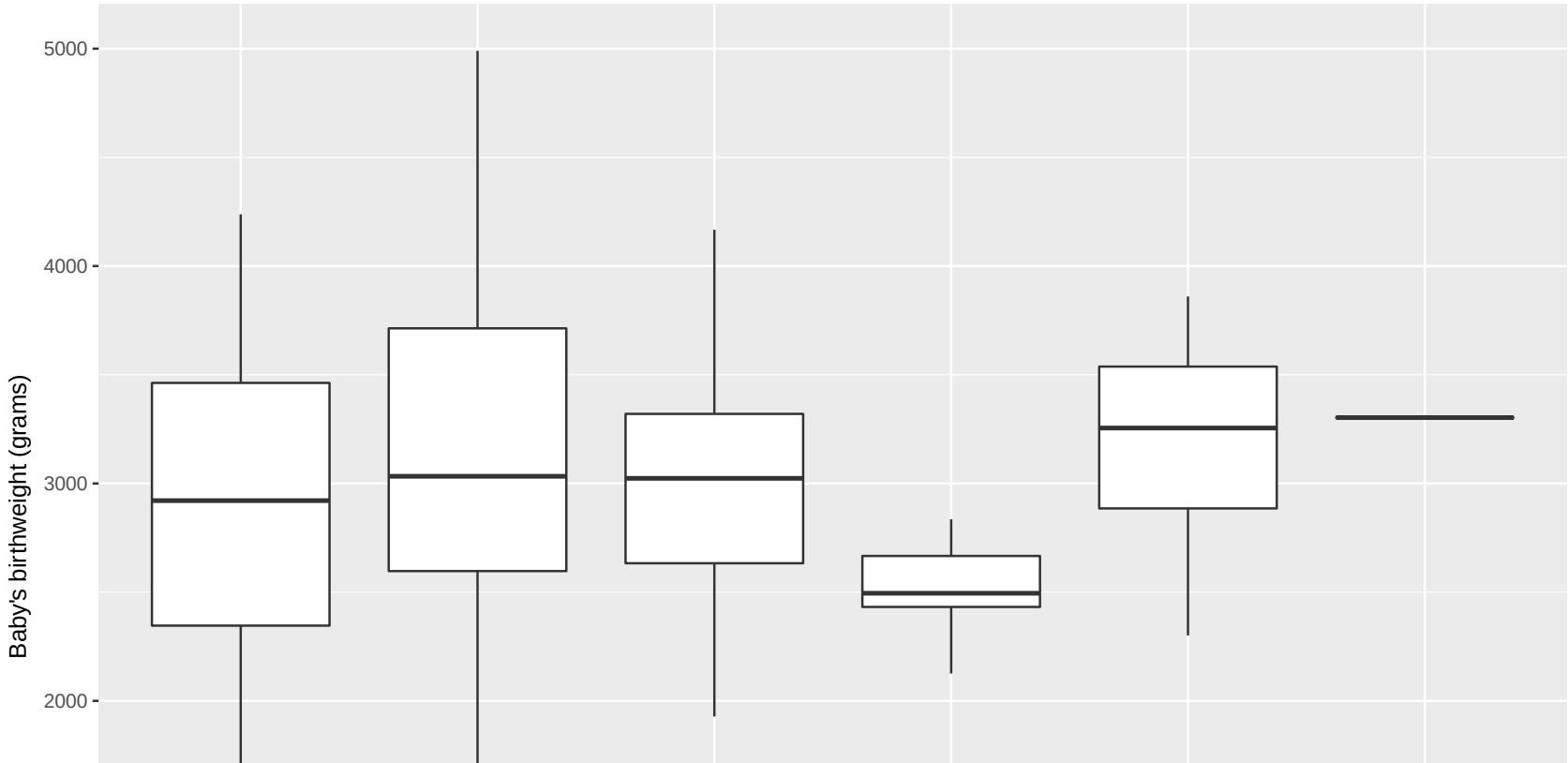
```
## `stat_bin()` using `bins = 30`. Pick better value with `
```



Box plots and violin plots

```
base.plot <- ggplot(birthwt, aes(x = as.factor(physician.vi
  xlab("Number of first trimester physician visits") +
  ylab("Baby's birthweight (grams)")

# Box plot
base.plot + geom_boxplot()
```



Visualizing means

Previously we calculated the following table:

```
bwt.summary <- aggregate(birthwt.grams ~ race + mother.smok
bwt.summary
```

```
##    race mother.smokes birthwt.grams
## 1 black            no      2854.500
## 2 other            no      2815.782
## 3 white            no      3428.750
## 4 black           yes      2504.000
## 5 other           yes      2757.167
## 6 white           yes      2826.846
```

We can plot this table in a nice bar chart as follows:

```
# Define basic aesthetic parameters
p.bwt <- ggplot(data = bwt.summary, aes(y = birthwt.grams,

# Pick colors for the bars
```

Does the association between birthweight and mother's age depend on smoking status?

We previously ran the following command to calculate the correlation between mother's ages and baby birthweights.

```
by(data = birthwt[c("birthwt.grams", "mother.age")],
   INDICES = birthwt["mother.smokes"],
   FUN = function(x) {cor(x[,1], x[,2])})
```

```
## mother.smokes: no
## [1] 0.2014558
## ------------------------------------------------------------
## mother.smokes: yes
## [1] -0.1441649
```

Here's a visualization of our data that allows us to see what's going on.

```
ggplot(birthwt, aes(x=mother.age, y=birthwt.grams, shape=mo
  geom_point() + # Adds points (scatterplot)
```