Lecture 5: Factors, Data frames, Tibbles

Finally working with data

Ingmar Sturm
UCSB
2024-07-01

Special thanks to Robin Liu for select course content used with permission.

Week 2: Working with data

Packages for this lecture

Before we proceed install the following packages:

`tidyverse` is a suite of R packages that streamline common data analysis tasks.

`gapminder` is a data set from a non-profit org. https://www.gapminder.org/

`datasets` is a default R package containing example data sets. It is usually loaded automatically.

```
install.packages(c("tidyverse", "gapminder"))
```

```
library(tidyverse)
library(gapminder)
library(datasets)
```

02:00

Summarizing Data

Measurements in centimeters of the petal length of 150 iris flowers

```
iris$Petal.Length
```

```
##   [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1 1.2 1.5 1.3 1.4
##  [19] 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5 1.4 1.6 1.6 1.5 1.5 1.4 1.5 1.2
##  [37] 1.3 1.4 1.3 1.5 1.3 1.3 1.3 1.6 1.9 1.4 1.6 1.4 1.5 1.4 4.7 4.5 4.9 4.0
##  [55] 4.6 4.5 4.7 3.3 4.6 3.9 3.5 4.2 4.0 4.7 3.6 4.4 4.5 4.1 4.5 3.9 4.8 4.0
##  [73] 4.9 4.7 4.3 4.4 4.8 5.0 4.5 3.5 3.8 3.7 3.9 5.1 4.5 4.5 4.7 4.4 4.1 4.0
##  [91] 4.4 4.6 4.0 3.3 4.2 4.2 4.2 4.3 3.0 4.1 6.0 5.1 5.9 5.6 5.8 6.6 4.5 6.3
## [109] 5.8 6.1 5.1 5.3 5.5 5.0 5.1 5.3 5.5 6.7 6.9 5.0 5.7 4.9 6.7 4.9 5.7 6.0
## [127] 4.8 4.9 5.6 5.8 6.1 6.4 5.6 5.1 5.6 6.1 5.6 5.5 4.8 5.4 5.6 5.1 5.1 5.9
## [145] 5.7 5.2 5.0 5.2 5.4 5.1
```

Numerical summaries help us understand the data

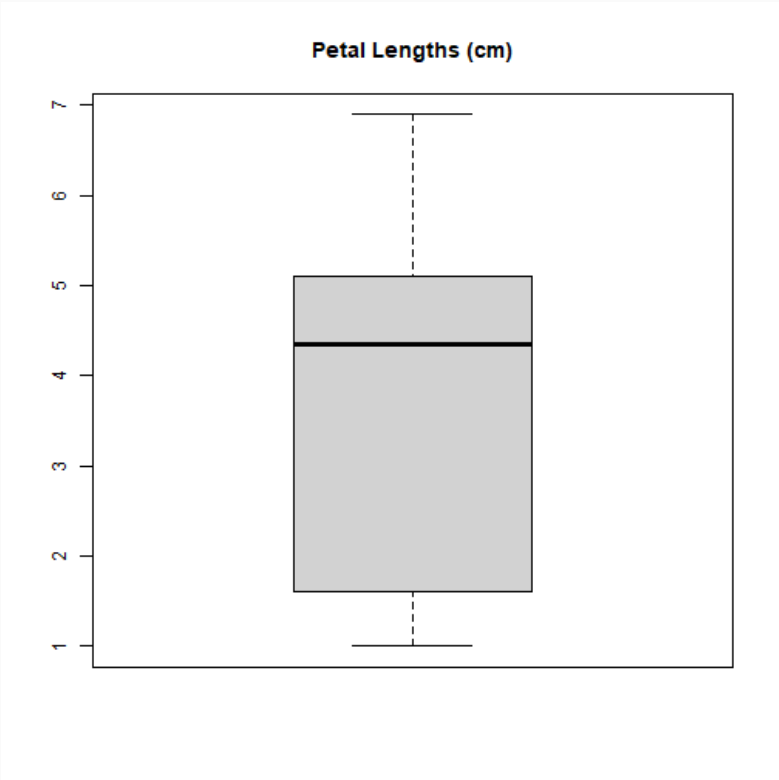
```
summary(iris$Petal.Length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.600   4.350   3.758   5.100   6.900
```

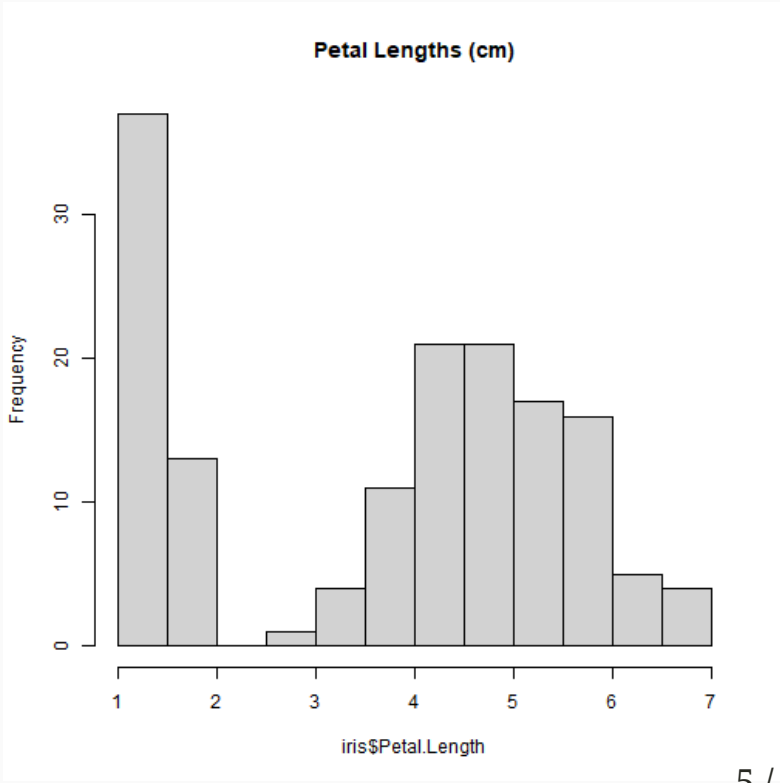
Summarizing Data

Visual summaries are even better

```
boxplot(iris$Petal.Length,
        main = "Petal Lengths (cm)")
```



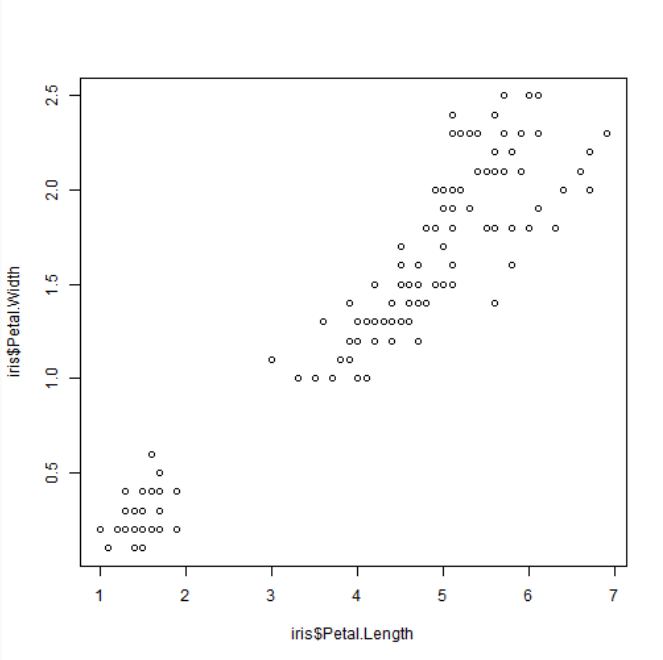

```
hist(iris$Petal.Length,
     main = "Petal Lengths (cm)")
```



Summarizing Data

Are two variables *correlated*?

```
plot(iris$Petal.Length, iris$Petal.Width)
```




```
cor(iris$Petal.Length, iris$Petal.Width)
```

```
## [1] 0.9628654
```

Summarizing Data

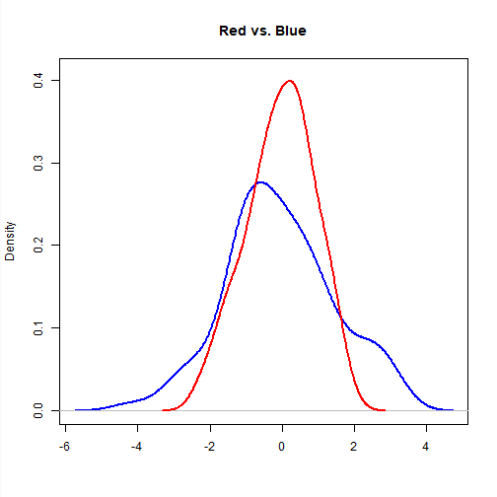
How "spread out" are the data?

```
summary(red)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.31933 -0.69237 -0.01444 -0.03719  0.57118  1.85215
```

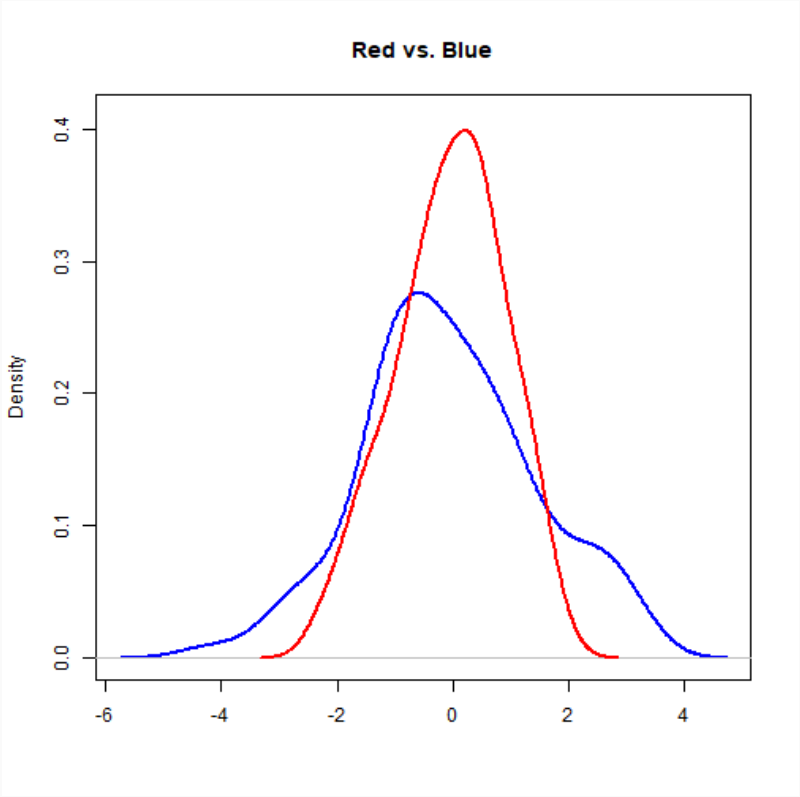
```
summary(blue)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.2345 -1.0976 -0.2257 -0.0630  0.8056  3.2002
```



Summarizing Data

The *standard deviation* measures the spread.




```
sd(red)
```

```
## [1] 0.934071
```

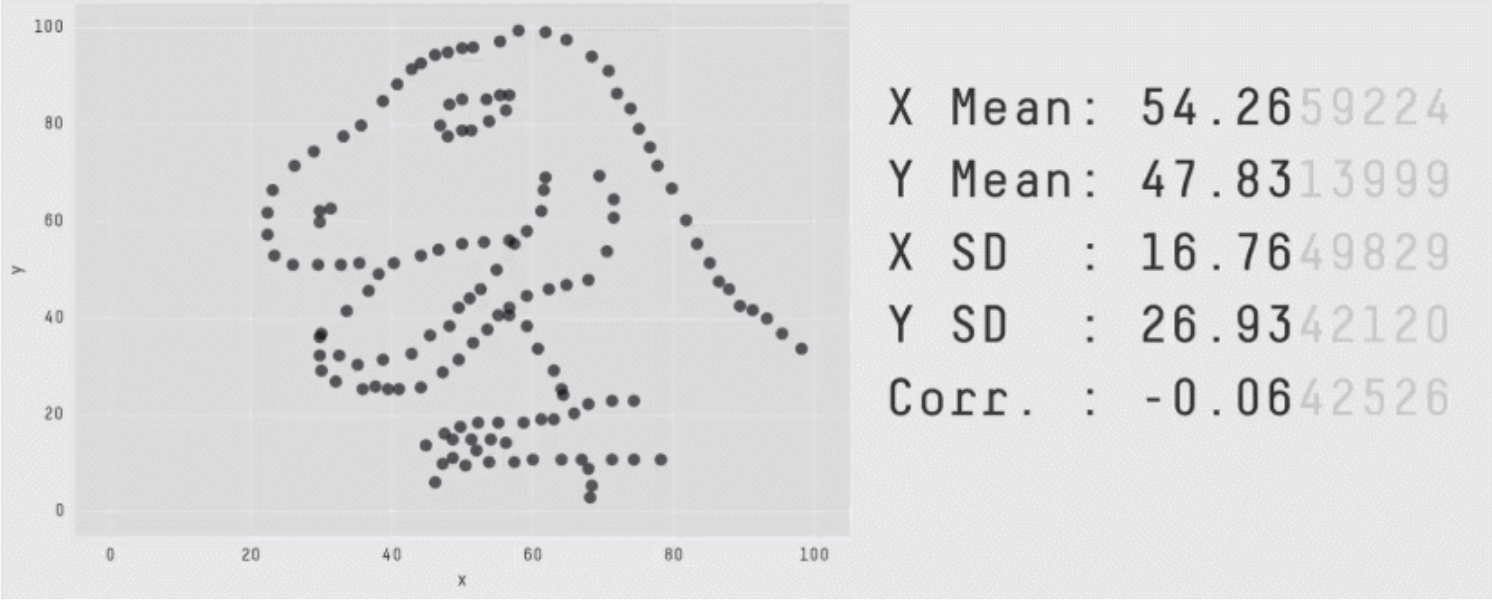
```
sd(blue)
```

```
## [1] 1.506259
```

The need for visualization

Exploratory data analysis is a crucial step in a data science project. Before we can apply any fancy machine learning method, we must understand the data through visual checks.

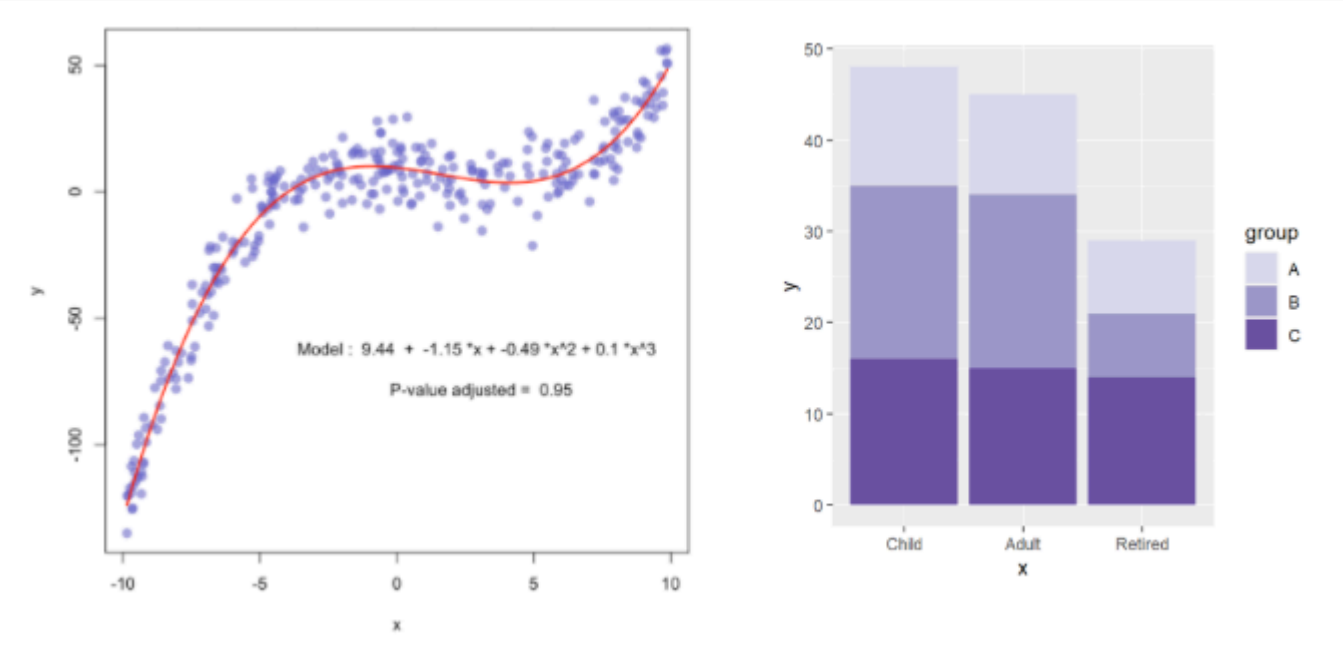
Numerical summaries (mean, median, mode, etc.) are not enough. Take a look at the *datasaurus dozen*:



Representing categorical data with **factors**

Categorical vs Quantitative data

- Numerics represent quantitative data
- Factors represent categorical data

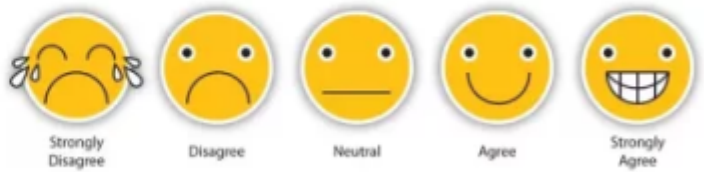


Factors

Categorical data

- Categorical variables have a fixed, known set of possible values
- They can be ordered or unordered

		Gender	
		f	m
Ice Cream	chocolate	60	20
	vanilla	25	30
	strawberry	15	50



Creating factors

Pass in the values along with the possible **levels**

```
flavors <- c("chocolate", "vanilla", "strawberry")
x <- c(rep("chocolate", 60),
       rep("vanilla", 25),
       rep("strawberry", 15))
ice_cream <- factor(x, levels = flavors)
```

```
summary(ice_cream)
```

```
##  chocolate    vanilla strawberry
##         60         25         15
```

Now run `ice_cream` in the console.

		Gender	
		f	m
Ice Cream	chocolate	60	20
	vanilla	25	30
	strawberry	15	50

Factors

```
ice_cream
```

```
##   [1] chocolate  chocolate  chocolate  chocolate  chocolate  chocolate
##   [7] chocolate  chocolate  chocolate  chocolate  chocolate  chocolate
##  [13] chocolate  chocolate  chocolate  chocolate  chocolate  chocolate
##  [19] chocolate  chocolate  chocolate  chocolate  chocolate  chocolate
##  [25] chocolate  chocolate  chocolate  chocolate  chocolate  chocolate
##  [31] chocolate  chocolate  chocolate  chocolate  chocolate  chocolate
##  [37] chocolate  chocolate  chocolate  chocolate  chocolate  chocolate
##  [43] chocolate  chocolate  chocolate  chocolate  chocolate  chocolate
##  [49] chocolate  chocolate  chocolate  chocolate  chocolate  chocolate
##  [55] chocolate  chocolate  chocolate  chocolate  chocolate  chocolate
##  [61] vanilla    vanilla    vanilla    vanilla    vanilla    vanilla
##  [67] vanilla    vanilla    vanilla    vanilla    vanilla    vanilla
##  [73] vanilla    vanilla    vanilla    vanilla    vanilla    vanilla
##  [79] vanilla    vanilla    vanilla    vanilla    vanilla    vanilla
##  [85] vanilla    strawberry strawberry strawberry strawberry strawberry
##  [91] strawberry strawberry strawberry strawberry strawberry strawberry
##  [97] strawberry strawberry strawberry strawberry
## Levels: chocolate vanilla strawberry
```

Factors

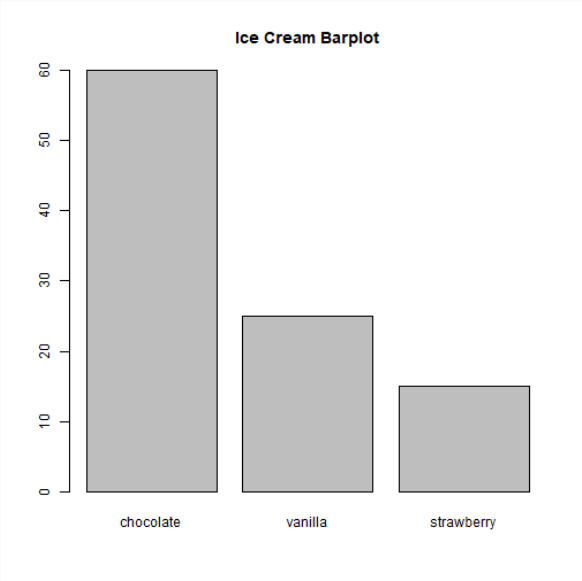
```
levels(ice_cream)
```

```
## [1] "chocolate"  "vanilla"    "strawberry"
```

```
nlevels(ice_cream)
```

```
## [1] 3
```

```
barplot(table(ice_cream), main="Ice Cream Barplot")
```



Data frames (and Tibbles)

Data frames

The fundamental object for data

Pretty much all of the data sets you will work with in R will be in the form of data frames.

Many data frames are readily available for you

```
library(datasets)
class(iris)
```

```
## [1] "data.frame"
```

```
class(mtcars)
```

```
## [1] "data.frame"
```

Data frames

First thing to do with a data frame is explore it:

```
str(iris) # "str" stands for structure
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
```

Data frames

```
View(iris) # View df in a separate window. Note the uppercase 'V' in View.
?iris # Brings out the help for a data set. May contain useful info.
```

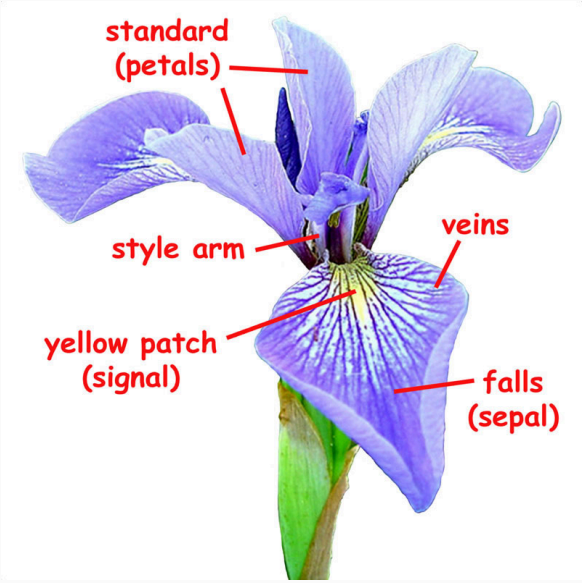
```
iris # prints entire df to console, undesirable for large dfs
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 2            4.9         3.0          1.4         0.2     setosa
## 3            4.7         3.2          1.3         0.2     setosa
## 4            4.6         3.1          1.5         0.2     setosa
## 5            5.0         3.6          1.4         0.2     setosa
## 6            5.4         3.9          1.7         0.4     setosa
## 7            4.6         3.4          1.4         0.3     setosa
## 8            5.0         3.4          1.5         0.2     setosa
## 9            4.4         2.9          1.4         0.2     setosa
## 10           4.9         3.1          1.5         0.1     setosa
## 11           5.4         3.7          1.5         0.2     setosa
## 12           4.8         3.4          1.6         0.2     setosa
## 13           4.8         3.0          1.4         0.1     setosa
## 14           4.3         3.0          1.1         0.1     setosa
## 15           5.8         4.0          1.2         0.2     setosa
```

Data frames

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```



Data frames

Extract a column in a df as a vector with `$`

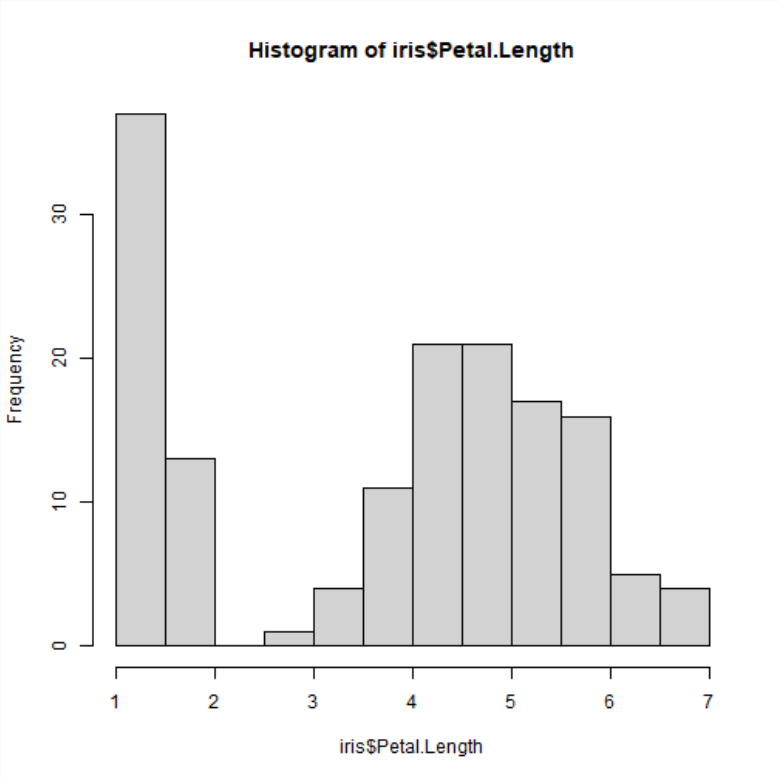
```
iris$Petal.Length
```

```
##   [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1 1.2 1.5 1.3 1.4
##  [19] 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5 1.4 1.6 1.6 1.5 1.5 1.4 1.5 1.2
##  [37] 1.3 1.4 1.3 1.5 1.3 1.3 1.3 1.6 1.9 1.4 1.6 1.4 1.5 1.4 4.7 4.5 4.9 4.0
##  [55] 4.6 4.5 4.7 3.3 4.6 3.9 3.5 4.2 4.0 4.7 3.6 4.4 4.5 4.1 4.5 3.9 4.8 4.0
##  [73] 4.9 4.7 4.3 4.4 4.8 5.0 4.5 3.5 3.8 3.7 3.9 5.1 4.5 4.5 4.7 4.4 4.1 4.0
##  [91] 4.4 4.6 4.0 3.3 4.2 4.2 4.2 4.3 3.0 4.1 6.0 5.1 5.9 5.6 5.8 6.6 4.5 6.3
## [109] 5.8 6.1 5.1 5.3 5.5 5.0 5.1 5.3 5.5 6.7 6.9 5.0 5.7 4.9 6.7 4.9 5.7 6.0
## [127] 4.8 4.9 5.6 5.8 6.1 6.4 5.6 5.1 5.6 6.1 5.6 5.5 4.8 5.4 5.6 5.1 5.1 5.9
## [145] 5.7 5.2 5.0 5.2 5.4 5.1
```

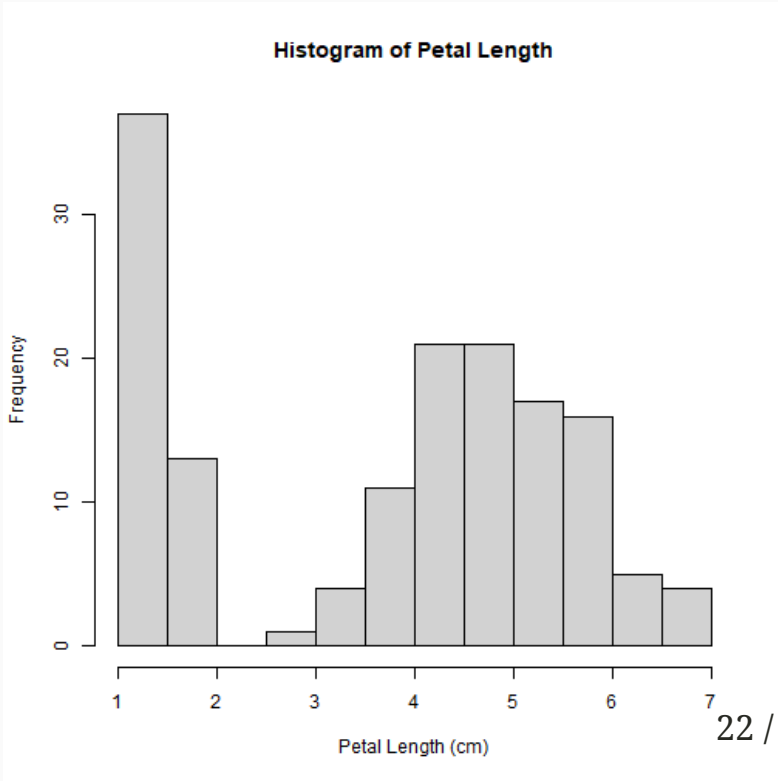
Data frames

A *histogram* plots the frequencies of the data after grouping them into *bins*.

```
hist(iris$Petal.Length)
```



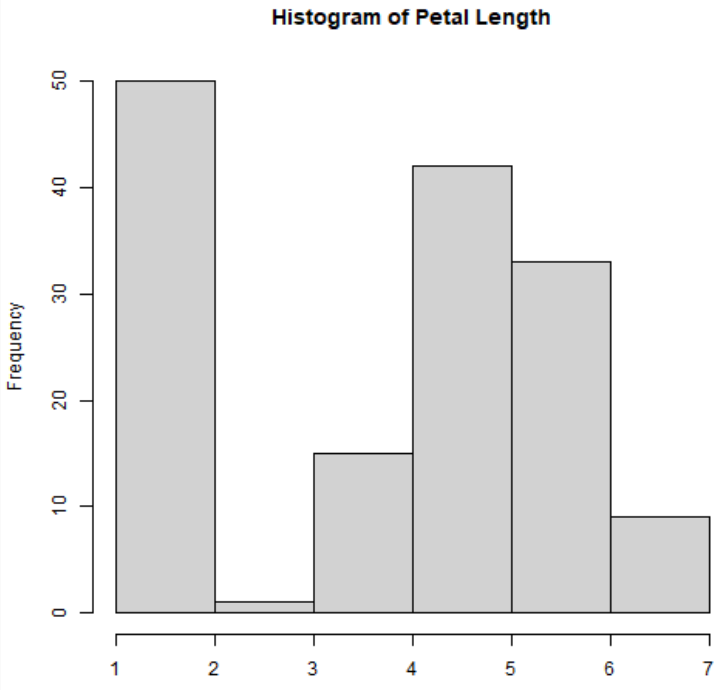

```
hist(iris$Petal.Length,
     main = "Histogram of Petal Length",
     xlab = "Petal Length (cm)")
```



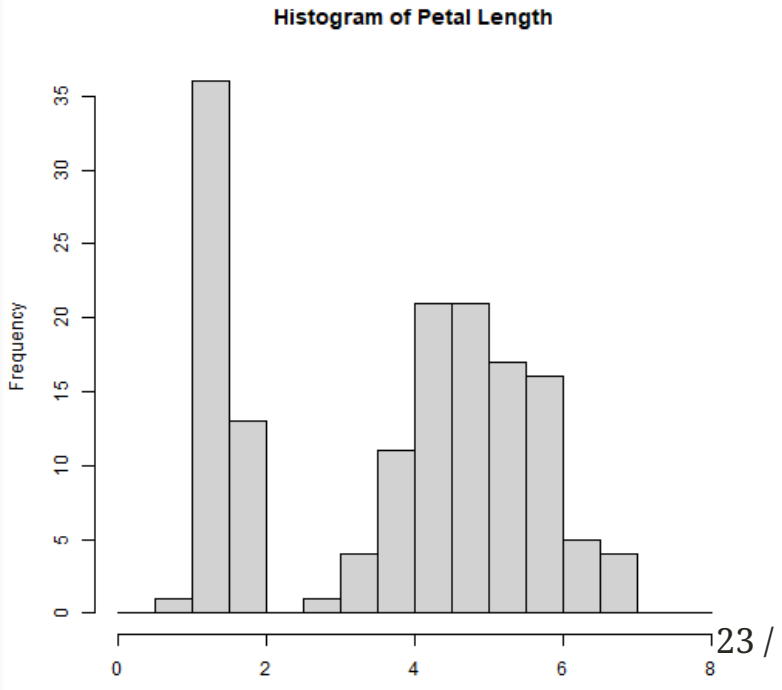
Data frames

Specifying *breaks* changes the number of bins.

```
hist(iris$Petal.Length,
     main = "Histogram of Petal Length",
     xlab = "Petal Length (cm)",
     breaks = 5)
```




```
hist(iris$Petal.Length,
     main = "Histogram of Petal Length",
     xlab = "Petal Length (cm)",
     breaks = seq(0, 8, by=0.5))
```



Tibbles

Tibbles are a special flavor of data frame. They have extended functionality and are usually easier to work with.

Make sure you have `gapminder` loaded.

```
library(gapminder)
```

Explore the `gapminder` data frame:

```
str(gapminder)
```

```
## tibble [1,704 x 6] (S3: tbl_df/tbl/data.frame)
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ year     : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp  : num [1:1704] 28.8 30.3 32 34 36.1 ...
##  $ pop      : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867
##  $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

In addition to being a data frame, it is also a *tibble*.

Tibbles

One difference between tibbles and data frames: tibbles have better printing to the console

```
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

Tibbles

Compare with printing a base data frame 🤢

```
as.data.frame(gapminder)
```

```
##                     country continent year  lifeExp        pop   gdpPercap
## 1               Afghanistan      Asia 1952 28.80100    8425333    779.4453
## 2               Afghanistan      Asia 1957 30.33200    9240934    820.8530
## 3               Afghanistan      Asia 1962 31.99700   10267083    853.1007
## 4               Afghanistan      Asia 1967 34.02000   11537966    836.1971
## 5               Afghanistan      Asia 1972 36.08800   13079460    739.9811
## 6               Afghanistan      Asia 1977 38.43800   14880372    786.1134
## 7               Afghanistan      Asia 1982 39.85400   12881816    978.0114
## 8               Afghanistan      Asia 1987 40.82200   13867957    852.3959
## 9               Afghanistan      Asia 1992 41.67400   16317921    649.3414
## 10              Afghanistan      Asia 1997 41.76300   22227415    635.3414
## 11              Afghanistan      Asia 2002 42.12900   25268405    726.7341
## 12              Afghanistan      Asia 2007 43.82800   31889923    974.5803
## 13                  Albania    Europe 1952 55.23000    1282697   1601.0561
## 14                  Albania    Europe 1957 59.28000    1476505   1942.2842
## 15                  Albania    Europe 1962 64.82000    1728137   2312.8890
## 16                  Albania    Europe 1967 66.22000    1984060   2760.1969
```

Tibbles

More tools to explore tibbles (and data frames)

```
head(gapminder)
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

Tibbles

More tools to explore tibbles (and data frames)

```
tail(gapminder)
```

```
## # A tibble: 6 x 6
##   country  continent  year lifeExp      pop gdpPercap
##   <fct>    <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Zimbabwe Africa     1982    60.4  7636524      789.
## 2 Zimbabwe Africa     1987    62.4  9216418      706.
## 3 Zimbabwe Africa     1992    60.4 10704340      693.
## 4 Zimbabwe Africa     1997    46.8 11404948      792.
## 5 Zimbabwe Africa     2002    40.0 11926563      672.
## 6 Zimbabwe Africa     2007    43.5 12311143      470.
```

Tibbles

```
names(gapminder)
```

```
## [1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

```
ncol(gapminder)
```

```
## [1] 6
```

```
length(gapminder)
```

```
## [1] 6
```

```
dim(gapminder)
```

```
## [1] 1704    6
```

```
nrow(gapminder)
```

```
## [1] 1704
```

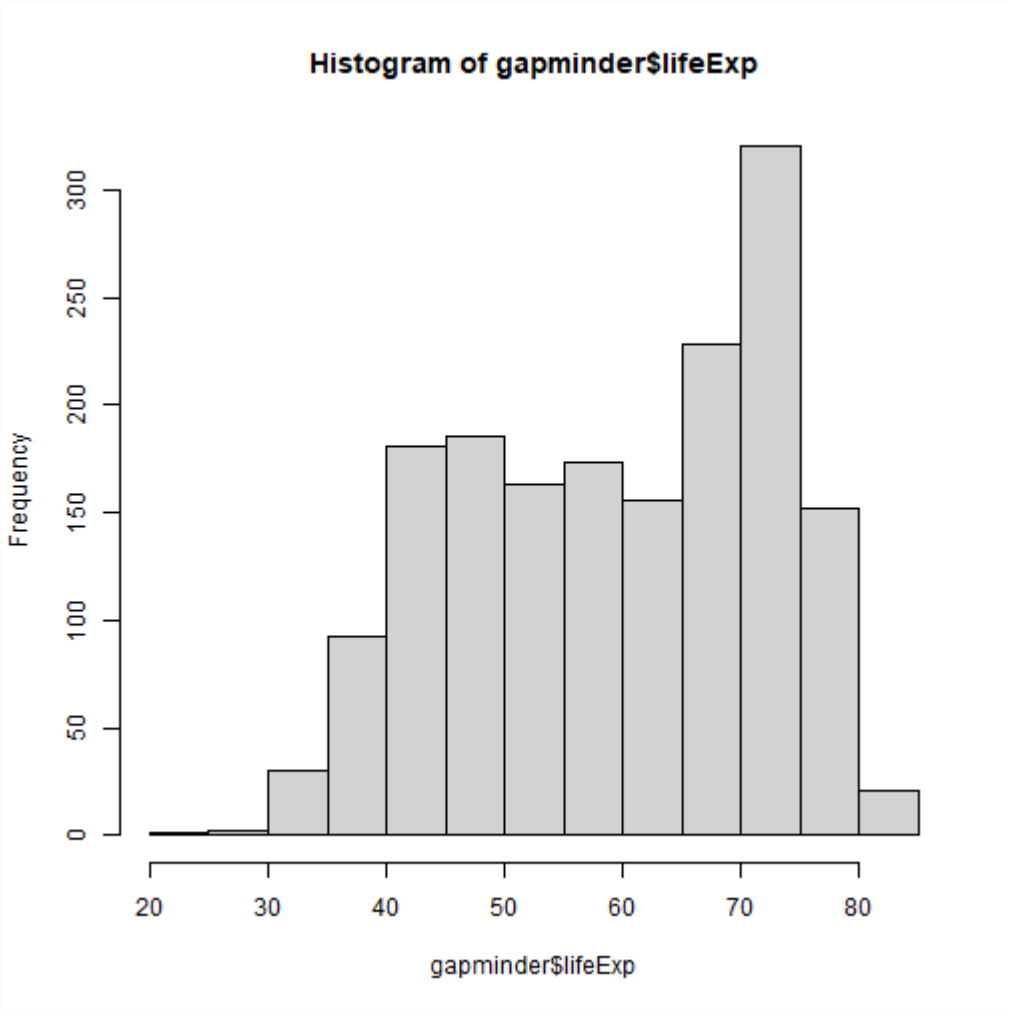
Tibbles

```
summary(gapminder)
```

```
##         country        continent        year         lifeExp
##  Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
##  Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
##  Algeria    :  12   Asia    :396   Median :1980   Median :60.71
##  Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47
##  Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85
##  Australia  :  12                  Max.   :2007   Max.   :82.60
##  (Other)    :1632
##       pop              gdpPercap
##  Min.   :6.001e+04   Min.   :   241.2
##  1st Qu.:2.794e+06   1st Qu.:  1202.1
##  Median :7.024e+06   Median :  3531.8
##  Mean   :2.960e+07   Mean   :  7215.3
##  3rd Qu.:1.959e+07   3rd Qu.:  9325.5
##  Max.   :1.319e+09   Max.   :113523.1
##
```

Basic Plotting

Create the following histogram of life expectancy:



02:00

Basic Plotting

Plot the counts for each continent. Not a histogram since data are not grouped into bins.

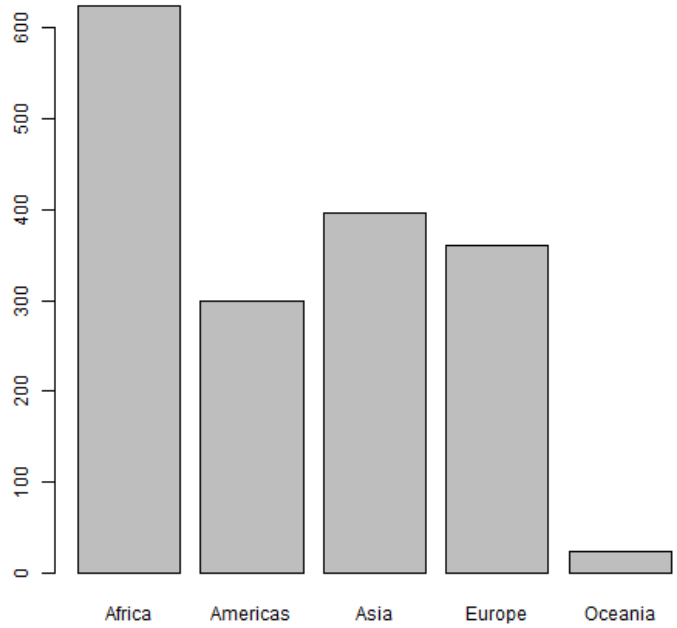
```
table(gapminder$continent)
```

```
##
##   Africa Americas     Asia   Europe  Oceania
##      624      300      396      360       24
```

```
is.factor(gapminder$continent)
```

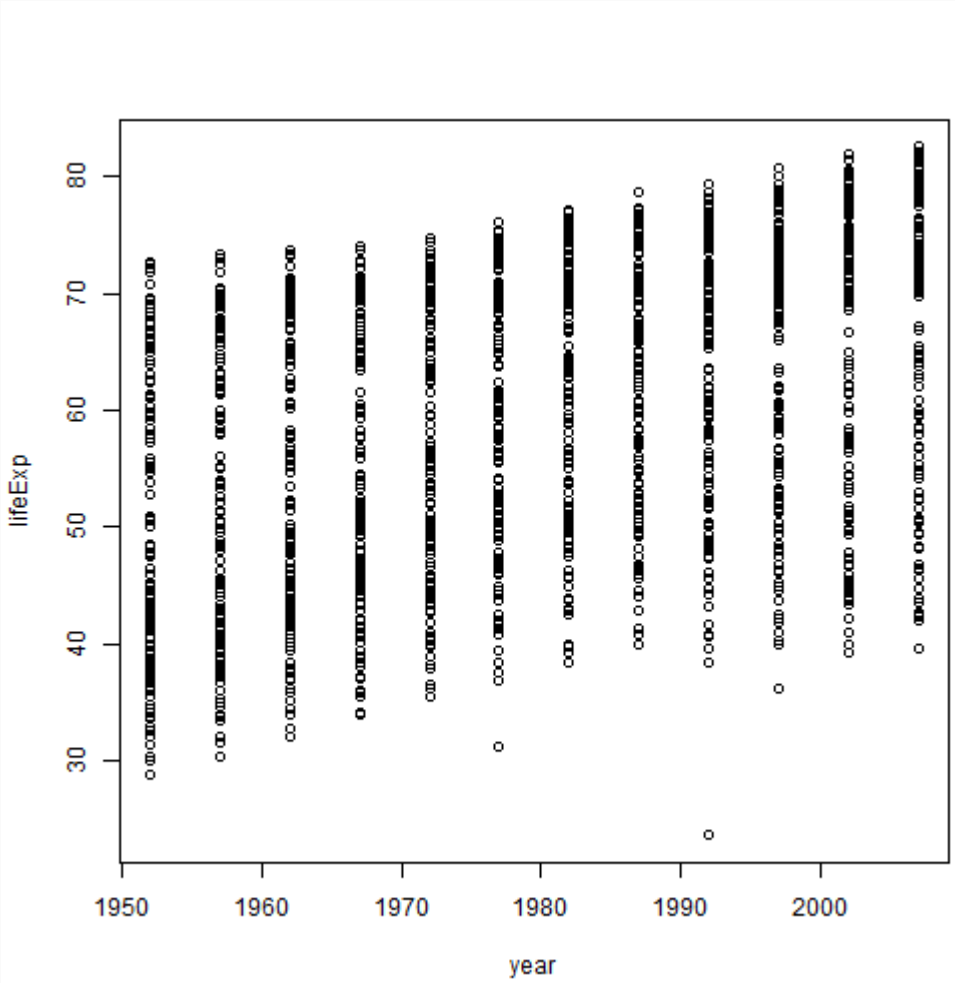
```
## [1] TRUE
```

```
barplot(table(gapminder$continent))
```



Plotting variables *against* each other

```
plot(lifeExp ~ year, gapminder)
```



Plotting variables *against* each other

Plot life expectancy against GDP per capita.

02:00

Convert data frames to tibbles

Tibbles are nicer to work with, but are not part of "base R".

It is easy to convert data frames to tibbles.

```
as_tibble(iris)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```