Lab Assignment1 Adv Informatics

Chapter 7/10

0. Create a small tibble manually.

```
library(tidyr)
library(tibble)
tibble(x = 1:5, y = 1, z = x ^ 2 + y)
```

```
## # A tibble: 5 x 3
##       x     y     z
##   <int> <dbl> <dbl>
## 1     1     1     2
## 2     2     1     5
## 3     3     1    10
## 4     4     1    17
## 5     5     1    26
```

1. How can you tell if an object is a tibble? (Hint: try printing mtcars, which is a regular data frame).

```
mtcars
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
library(tibble)
as.tibble(mtcars)
```

```
## # A tibble: 32 x 11
##      mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##  * <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  21       6  160    110  3.9   2.62  16.5     0     1     4     4
##  2  21       6  160    110  3.9   2.88  17.0     0     1     4     4
##  3  22.8     4  108     93  3.85  2.32  18.6     1     1     4     1
##  4  21.4     6  258    110  3.08  3.22  19.4     1     0     3     1
##  5  18.7     8  360    175  3.15  3.44  17.0     0     0     3     2
##  6  18.1     6  225    105  2.76  3.46  20.2     1     0     3     1
##  7  14.3     8  360    245  3.21  3.57  15.8     0     0     3     4
##  8  24.4     4  147.    62  3.69  3.19  20       1     0     4     2
##  9  22.8     4  141.    95  3.92  3.15  22.9     1     0     4     2
## 10  19.2     6  168.   123  3.92  3.44  18.3     1     0     4     4
## # ... with 22 more rows
```

Tibbles will say "A tibble" when displaying itself. A tibble will also show only the first 10 rows of the data instead of a data frame which will list out all of the rows. A tibble will also report the type for each of its columns as well.

2. Compare and contrast the following operations on a data.frame and equivalent tibble. What is different? Why might the default data frame behaviours cause you frustration?

```
df <- data.frame(abc = 1, xyz = "a")
df
```

```
##   abc xyz
## 1   1   a
```

```
df$x
```

```
## [1] a
## Levels: a
```

```
df[, "xyz"]
```

```
## [1] a
## Levels: a
```

```
df[, c("abc", "xyz")]
```

```
##   abc xyz
## 1   1   a
```

```
tibble1 <- tibble(abc=1, xyz="a")
tibble1
```

```
## # A tibble: 1 x 2
##     abc xyz
##   <dbl> <chr>
## 1     1 a
```

```
tibble1$x
```

```
## Warning: Unknown or uninitialised column: 'x'.
## NULL
```

```
tibble1[, "xyz"]
```

```
## # A tibble: 1 x 1
##   xyz
##   <chr>
## 1 a
```

```
tibble1[, c("abc", "xyz")]
```

```
## # A tibble: 1 x 2
##     abc xyz
##   <dbl> <chr>
## 1     1 a
```

```
tibble1$abc
```

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

```
tibble1$xyz
```

```
## [1] "a"
```

The dataframe will still return the value even when the full name of the column isn't used. Just by asking for "x" it returned a value for that column. If the same is done with the tibble, it does not work. The column names are "abc" and "xyz" and the tibble will return when the full column name is given. With the tibble this could prevent the wrong column or row or data in general from being called or worked on since the full name should be specified.

3. If you have the name of a variable stored in an object, e.g. var <- "mpg", how can you extract the reference variable from a tibble?

To do this, you would use `df[[var]]` rather than using the dollar sign as in previous questions. Using the dollar sign will look for a column with the exact name specified and not look for variables stored in objects.

4. Practice referring to non-syntactic names in the following data frame by:

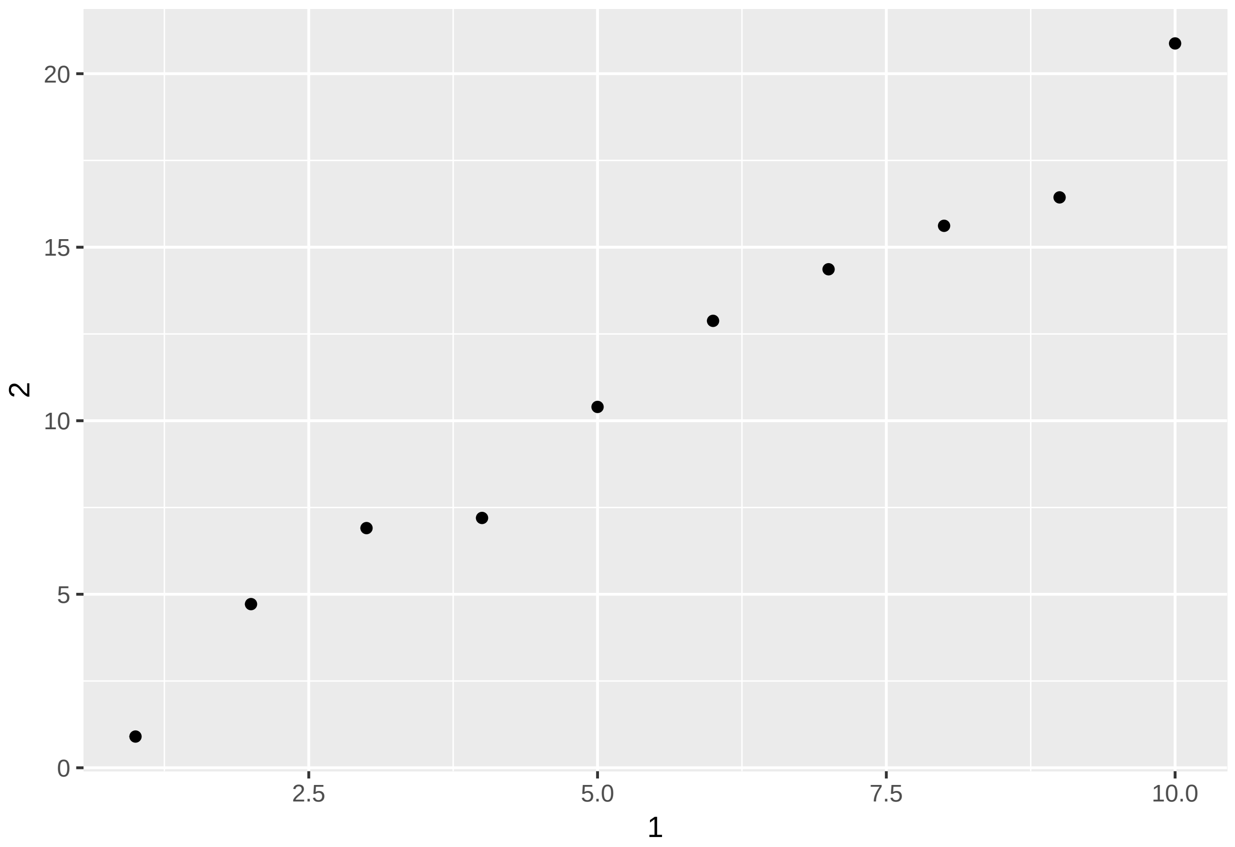
a. Extracting the variable called 1.

```
annoying <- tibble(
  `1` = 1:10,
  `2` = `1` * 2 + rnorm(length(`1`))
)
annoying$`1`
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

b. Plotting a scatterplot of 1 vs 2.

```
library(ggplot2)
ggplot(annoying, aes(x=`1`, y=`2`)) + geom_point()
```



c. Creating a new column called 3 which is 2 divided by 1.

The mutate function from dplyr can be used to add new columns

```
library(dplyr)
```

```
##
## 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
```

```
mutate(annoying, `3` = `2` / `1`)
```

```
## # A tibble: 10 x 3
##      `1`    `2`   `3`
##    <int>  <dbl> <dbl>
##  1     1  0.900 0.900
##  2     2  4.72  2.36
##  3     3  6.91  2.30
##  4     4  7.20  1.80
##  5     5 10.4   2.08
##  6     6 12.9   2.15
##  7     7 14.4   2.05
##  8     8 15.6   1.95
##  9      9 16.4   1.83
## 10     10 20.9   2.09
```

d. Renaming the columns to one, two and three.

The rename function from dplyr can do this task.

```
library(dplyr)
annoying %>%
+ rename(one = `1`, two = `2`, three = `3` )
```

5. What does `tibble::enframe()` do? When might you use it?

This function can take a vector that is named and turn it into a data frame/tibble. The columns will be name and value.

6. What option controls how many additional column names are printed at the footer of a tibble?

```
print(n=10)
```

With this function you can specify how many rows to display of the tibble as well as adjust how many columns with the width option as well.

Chapter 8/11

1. What function would you use to read a file where fields were separated with "|"?

The readr package has a function `read_delim` which can be used to specify what is separating the fields.

```
library(readr)
read_delim(file, delim="|")
```

2. Apart from file, skip, and comment, what other arguments do read_csv() and read_tsv() have in common?

```
library(readr)
union(names(formals(read_csv)), names(formals(read_tsv)))
```

```
##  [1] "file"            "col_names"       "col_types"
##  [4] "locale"          "na"              "quoted_na"
##  [7] "quote"           "comment"         "trim_ws"
## [10] "skip"            "n_max"           "guess_max"
## [13] "progress"        "skip_empty_rows"
```

4. Sometimes strings in a CSV file contain commas. To prevent them from causing problems they need to be surrounded by a quoting character. By convention, read_csv() assumes that the quoting character will be , and if you want to change it you'll need to use read_delim() instead. What arguments do you need to specify to read the following text into a data frame?

```
"x,y\n1,'a,b'"
```

To change the quoting character, you will specify `quote` in `read_delim()` from the readr package.

```
library(readr)
xquote <- "x,y\n1, 'a,b'"
read_delim(xquote, ",", quote = "'" )
```

```
## # A tibble: 1 x 2
##       x y
##   <dbl> <chr>
## 1     1 a,b
```

5. Identify what is wrong with each of the following inline CSV files. What happens when you run the code?

```
read_csv("a,b\n1,2,3\n4,5,6")
```

```
## Warning: 2 parsing failures.
## row col  expected    actual         file
##   1  -- 2 columns 3 columns literal data
##   2  -- 2 columns 3 columns literal data
```

```
## # A tibble: 2 x 2
##       a     b
##   <dbl> <dbl>
## 1     1     2
## 2     4     5
```

It is telling us that 3 columns are actually there but only 2 columns were asked for or "expected," so only the first 2 columns are shown.

```
read_csv("a,b,c\n1,2\n1,2,3,4")
```

```
## Warning: 2 parsing failures.
## row col  expected    actual         file
##   1  -- 3 columns 2 columns literal data
##   2  -- 3 columns 4 columns literal data
```

```
## # A tibble: 2 x 3
##       a     b     c
##   <dbl> <dbl> <dbl>
## 1     1     2    NA
## 2     1     2     3
```

There are 3 columns in the header, but for row 1 there are only 2 values in the first two columns and nothing in the third. For row 3 there are 4 columns of data compared to the 3 columns in the header so the 4th value is not shown.

```
read_csv("a,b\n\"1")
```

```
## Warning: 2 parsing failures.
## row col                     expected    actual         file
##   1  a  closing quote at end of file           literal data
##   1  -- 2 columns                    1 columns literal data
```

```
## # A tibble: 1 x 2
##       a b
##   <dbl> <chr>
## 1     1 <NA>
```

There is an extra quotation that is not closed.

```
read_csv("a,b\n1,2\na,b")
```

```
## # A tibble: 2 x 2
##   a     b
##   <chr> <chr>
## 1 1     2
## 2 a     b
```

The columns a and b are being treated as character vectors since the third row has "values" a and b.

```
read_csv("a;b\n1;3")
```

```
## # A tibble: 1 x 1
##   `a;b`
##   <chr>
## 1 1;3
```

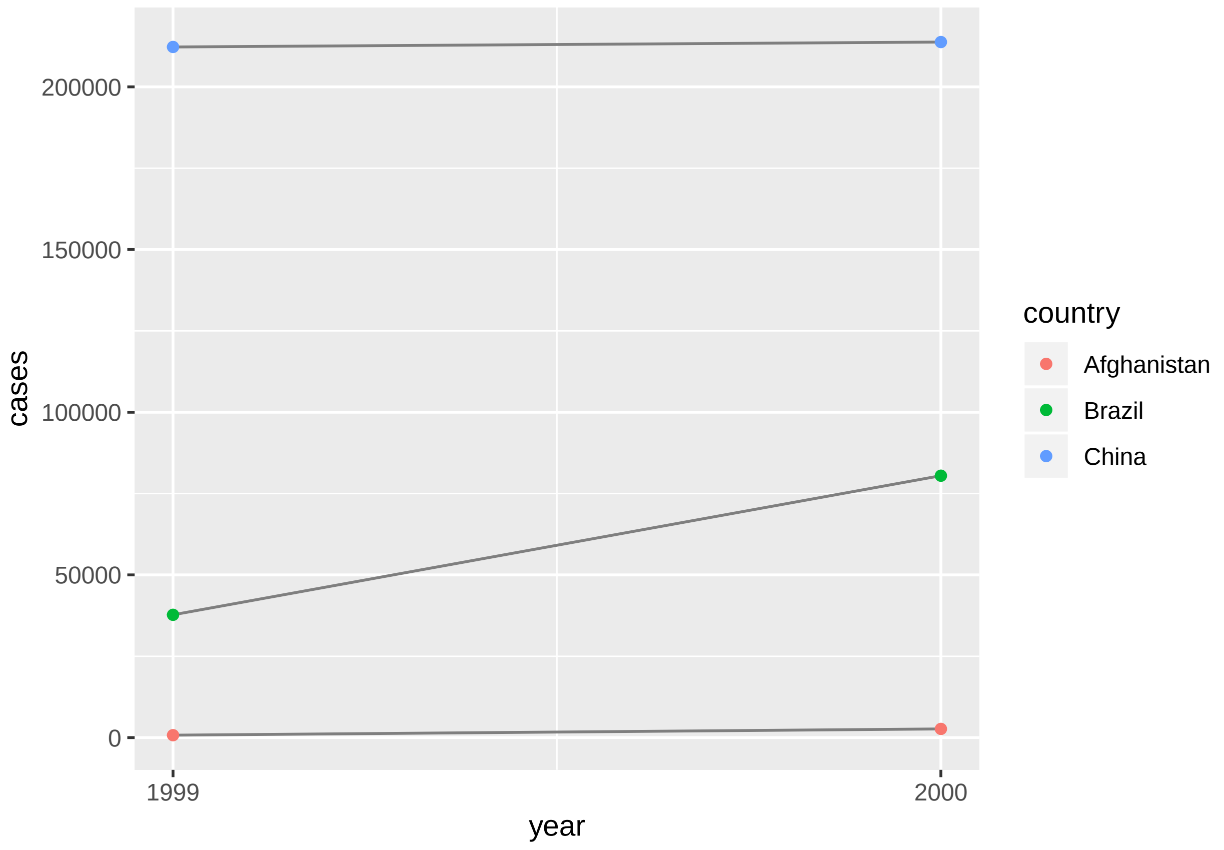
The values are being separated by a semicolon which isn't supported or read correctly as seen. There should be two columns, one named a and one named b and the values are 1 and 3 respectively.

Chapter 9/12

3. Recreate the plot showing change in cases over time using table2 instead of table1. What do you need to do first?

We can use `filter` first to only include rows with information about cases.

```
library(ggplot2)
library(tidyr)
table2 %>%
  filter(type == "cases") %>%
  ggplot(aes(year, count)) +
  geom_line(aes(group = country), colour = "grey50") +
  geom_point(aes(colour = country)) +
  scale_x_continuous(breaks = unique(table2$year)) +
  ylab("cases")
```



2. Why does this code fail?

```
table4a %>%
  gather(1999, 2000, key = "year", value = "cases")
#> Error in inds_combine(.vars, ind_list): Position must be between 0 and n
```

In this example, the function gather is looking for the columns that are number 1999 and 2000 (1999th or 2000th column) instead of looking for columns that are named 1999 or 2000. You could use backticks to show they are column names instead.

Chapter 10/13

1. Add a surrogate key to flights.

```
library("nycflights13")
flights %>%
  mutate(flight_key = row_number()) %>%
  glimpse()
```

```
## Observations: 336,776
## Variables: 20
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013,...
## $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 55...
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 60...
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2...
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 7...
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 7...
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -...
## $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV",...
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79...
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN...
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR"...
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL"...
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138...
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 94...
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5,...
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013...
## $ flight_key     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, ...
```

2. Add the location of the origin and destination (i.e. the lat and lon) to flights.

```
airport_locations <- airports %>%
  select(faa, lat, lon)

flights %>%
  select(year:day, hour, origin, dest) %>%
  left_join(
    airport_locations,
    by = c("origin" = "faa")
  ) %>%
  left_join(
    airport_locations,
    by = c("dest" = "faa")
  )
```

```
## # A tibble: 336,776 x 10
##     year month   day  hour origin dest  lat.x lon.x lat.y lon.y
##    <int> <int> <int> <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <dbl>
##  1  2013     1     1     5 EWR    IAH    40.7 -74.2  30.0 -95.3
##  2  2013     1     1     5 LGA    IAH    40.8 -73.9  30.0 -95.3
##  3  2013     1     1     5 JFK    MIA    40.6 -73.8  25.8 -80.3
##  4  2013     1     1     5 JFK    BQN    40.6 -73.8  NA    NA
##  5  2013     1     1     6 LGA    ATL    40.8 -73.9  33.6 -84.4
##  6  2013     1     1     5 EWR    ORD    40.7 -74.2  42.0 -87.9
##  7  2013     1     1     6 EWR    FLL    40.7 -74.2  26.1 -80.2
##  8  2013     1     1     6 LGA    IAD    40.8 -73.9  38.9 -77.5
##  9  2013     1     1     6 JFK    MCO    40.6 -73.8  28.4 -81.3
## 10  2013     1     1     6 LGA    ORD    40.8 -73.9  42.0 -87.9
## # ... with 336,766 more rows
```

Where x is the origin and y is the destination information.