COMP/STAT 112 Midsemester Review

Handy functions

You will be given this list of handy functions with your quiz.

- `ggplot` functions
  `ggplot()`, `geom_bar()`, `geom_boxplot()`, `geom_density()`, `geom_histogram()`, `geom_line()`, `geom_point()`, `geom_smooth()`, `facet_wrap()`
- wrangling functions
  `arrange()`, `count()`, `filter()`, `group_by()`, `mutate()`, `select()`, `summarize()`
- `pivot_` functions
  `pivot_longer()`, `pivot_wider()`
- `_join` functions
  `anti_join()`, `full_join()`, `inner_join()`, `left_join()`, `semi_join()`
- `fct_` functions
  `fct_recode()`, `fct_relevel()`, `fct_reorder()`
- `str_` functions
  `str_detect()`, `str_length()`, `str_replace()`, `str_replace_all()`, `str_sub()`, `str_to_lower()`, `separate()`

Mid-semester review Part 1: Data preparation functions

Kiva is a non-profit that allows people from around the world to lend small amounts to others to start or grow a business, go to school, access clean energy, etc. The `loans` data includes information about 10,000 different Kiva loans:

```
head(loans, 3)
## # A tibble: 3 x 6
##   borrower sector  partner_id  year funded  paid
##   <chr>    <chr>        <dbl> <dbl>  <dbl> <dbl>
## 1 Jose     Housing         23  2009    425  417.
## 2 Amor     Housing        126  2011   1175  783.
## 3 Jamshed  Food           100  2010   3000 3000
```

For each question below, simply name the **1 or 2 functions** you would need to complete the task related to the `loans` data. Do not write out the full code, and when there are multiple possible approaches, use the most efficient.

(1) Get a table of loans in the food sector.

(2) Get a table of the number of loans in each sector (eg: Housing, Food, Agriculture).

(3) The sector categories are currently stored in alphabetical order (eg: Agriculture, Arts, Clothing, …). Re-order these according to the typical funded loan amount in these sectors.

(4) Convert the funded amount of the loan from dollars to yen.

(5) Record only the first initial (letter) for each borrower's name.

(6) The number of unpaid loans by each year is shown below:

```
## # A tibble: 3 x 2
##    year     n
##   <dbl> <int>
## 1  2010  2348
## 2  2011  3075
## 3  2012   924
```

Put the data in this new format:

```
## # A tibble: 1 x 3
##   `2010` `2011` `2012`
##    <int>  <int>  <int>
## 1   2348   3075    924
```

(7) Get a smaller table with just the borrower name and their sector.

(8) Make a new variable that records the amount of each loan that is still unpaid, i.e. the difference between what was funded (promised) and what has been paid.

(9) Only keeps loans for borrowers that include the word "Group" in their name (eg: Gemma's Group, Benkadi 4 Group, …)

(10) Find a set of the highest paid loans.

(11) Calculate the average funded amount by sector. (This requires 2 functions. What are they, in order?)

(12) In the sector variable, abbreviate "Agriculture" to "Ag".

(13) The `partners` dataset includes data on each of the funding *partners* or lenders:

```
head(partners, 3)
## # A tibble: 3 x 3
##   partner_id countries.region total_amount_raised
##        <dbl> <chr>                          <dbl>
## 1          1 Africa                         26600
## 2          2 Middle East                     4400
## 3          3 Eastern Europe                 36700
```

For each loan in the `loans` data, attach on information about the corresponding funding partner: `loans %>% ...`.

(14) Get a dataset that keeps all data in `loans` *and* `partners`, even when there is no matching `partner_id`. `loans %>% ...`

Mid-semester review Part 2

NOTE: The actual quiz will cover Activities 1–13. The questions below focus mainly on Activities 2–11 as this material is less fresh. Be sure to completely review all activities for the quiz!

The `running` dataset includes information on runners that have competed in the annual Cherry Blossom 10-mile race in Washington, D.C.. Each row represents one race for one runner, including the `net` running time in minutes, `year` of the race, and the binary `sex`:

```
head(running, 4)
## # A tibble: 4 x 5
##   name.yob               sex     age   net  year
##   <chr>                  <chr> <int> <dbl> <int>
## 1 a. renee callahan 1966 F        37 100.   2003
## 2 aaren pastor 1991      F        14  89.4  2005
## 3 aaren pastor 1991      F        15  71.2  2006
## 4 aaron alton 1974       M        31  85.4  2005
```

(15) **What code should I use?**
For each prompt below, fill in the blanks with the appropriate information.

```
# Sort the net running times from shortest to longest
running %>%

  _________________(net)

# Record the (approximate) birth year of each runner
running %>%

  _________________(birth_year = year - age)

# Keep only data on the age and net running time
running %>%

  _________________(age, net)

# Keep only data on runners with "renee" in their name
running %>%
```

```
  filter(______________(name.yob, "renee"))

# Change the year variable to only include the last 2 digits
running %>%

  _____________(year = _____________(year, 3, 4))
```

(16) **What code did I use?** Fill in the blanks with the missing function name.

```
running %>%

  _________________(age > 85)
```

```
## # A tibble: 2 x 5
##   name.yob          sex     age   net  year
##   <chr>             <chr> <int> <dbl> <int>
## 1 hedy marque 1918 F        86  133.  2004
## 2 hedy marque 1918 F        87  118.  2005
```

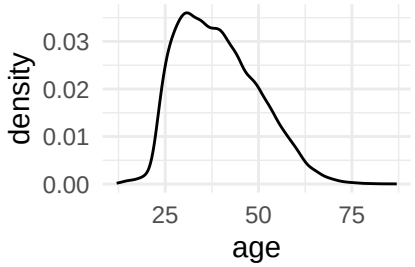
```
running %>%

  _________________(maximum_time = max(net))
```

```
## # A tibble: 1 x 1
##   maximum_time
##          <dbl>
## 1         176.
```

```
ggplot(running, aes(x = age)) +

  ____________________________() +
  theme_minimal()
```



(17) **Interpretation**
Summarize, in words and in context, what we learn from the plot above. Remember to keep your response to 1 sentence.

(18) **What would this code do?**
Consider a smaller data set, `running_small`:

```
running_small
## # A tibble: 6 x 5
##   name.yob          sex     age   net  year
##   <chr>             <chr> <int> <dbl> <int>
## 1 abby solomon 1984 F        21 115.   2005
## 2 abby solomon 1984 F        22 108.   2006
## 3 yungki kim 1955   M        45  83.6  2000
## 4 yungki kim 1955   M        47  81.2  2002
## 5 yungki kim 1955   M        48  82.0  2003
## 6 yungki kim 1955   M        49  81.9  2004
```

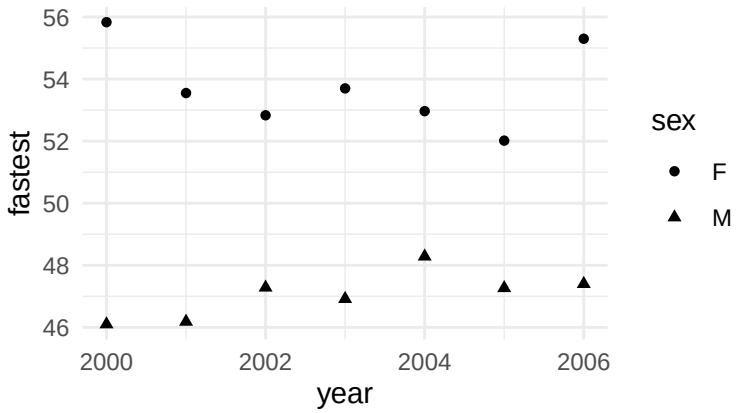
Fill in all 6 empty cells of the table below to communicate what the following code would produce:

```
running_small %>%
  count(name.yob)
```


(19) **Write the code**

```
head(running, 4)
## # A tibble: 4 x 5
##   name.yob                sex     age   net  year
##   <chr>                   <chr> <int> <dbl> <int>
## 1 a. renee callahan 1966 F        37 100.   2003
## 2 aaren pastor 1991      F        14  89.4  2005
## 3 aaren pastor 1991      F        15  71.2  2006
## 4 aaron alton 1974       M        31  85.4  2005
```

Below is a plot of the *fastest* net running time each year:



Fill in the code skeleton below with the code used to create this plot. NOTE: Some blanks require more than 1 "word".

```
_________________ %>%

  _________________(_________________________________) %>%

  _________________(_________________________________) %>%

  ggplot(aes(_____________________________________________)) +

    ____________________() +

    theme_minimal()
```

(20) **Who ran every year?**

The `running` data includes anybody who ran *any* of the races from 2000-2006. For example, it includes people that ran in only one year (e.g. only 2001) and people that ran *every* year from 2000-2006. The `all_years` data includes only the names of people that ran in *every* race from 2000-2006:

```
head(all_years, 3)
## # A tibble: 3 x 4
##   name.yob           first   last    yob
##   <chr>              <chr>   <chr> <dbl>
## 1 arthur scott 1960  arthur  scott  1960
## 2 bernard kelly 1956 bernard kelly  1956
## 3 betty blank 1953   betty   blank  1953
```

Suppose we want to obtain a subset of the `running` data for *only* these runners in `all_years`. Which join function(s) could we use? Circle the letter corresponding to any answer that could work.

a. `left_join()`
b. `inner_join()`
c. `full_join()`
d. `semi_join()`
e. `anti_join()`

(21) **Change the data**

Let's focus on just the runners that ran in each race from 2000-2006, stored in `running_2`:

```
head(running_2, 3)
## # A tibble: 3 x 5
##   name.yob          sex     age   net  year
##   <chr>             <chr> <int> <dbl> <int>
## 1 arthur scott 1960 M        40  83.2  2000
## 2 arthur scott 1960 M        41  85.7  2001
## 3 arthur scott 1960 M        42  79.7  2002
```

Suppose we wished to store this data in *wide* format:

```
## # A tibble: 3 x 8
##   name.yob        year2000 year2001 year2002 year2003 year2004 year2005 year2006
##   <chr>              <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 arthur scott 1~     83.2     85.7     79.7     79.8     83.3     81.7     81.3
## 2 bernard kelly ~     95.8     89.2     93.6     91.4     95.2     89.7     94.6
## 3 betty blank 19~     68.2     68.2     70.4     69.4     70.3     72.6     69.8
```

Fill in the code skeleton below with the code used to create this new data. NOTE: Some blanks require more than 1 "word".

```
_________________ %>%

  select(_____________________________________________________) %>%


  _________________(_________________ = _________________,

                   _________________ = _________________,

                   names_prefix = "_________________") %>%

  head(3)
```

(22) **Units of observation**
What are the units of observation in the *wide* data?