R Lab 2

Zeren Li

9/2/2019

Roadmap

- RStudio server, R Style
- `tidyverse`
- T-test

Advanced programming for data science

`tidyverse`

"The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures."

The core tidyverse includes the packages that you're likely to use in everyday data analyses. As of tidyverse 1.2.0, the following packages are included in the core tidyverse:

- `magrittr`: offers a set of operators which make your code more readable
- `dplyr`: a grammar of data manipulation
- `ggplot2`:a system for declaratively creating graphics
- `tidyr`: a set of functions that help you get to tidy data
- `readr`: provides a fast and friendly way to read rectangular data
- `purrr`: enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors
- `tibble`: a modern re-imagining of the data frame
- `stringr`: a cohesive set of functions designed to make working with strings as easy as possible
- `forcats`: provides a suite of useful tools that solve common problems with factors

`magrittr`

The `magrittr` does the following:

- structuring sequences of data operations left-to-right (as opposed to from the inside and out)
- avoiding nested function calls
- minimizing the need for local variables and function definitions
- making it easy to add steps anywhere in the sequence of operations

You can think about the following sequence of actions - find key, unlock car, start car, drive to school, park.

Expressed as a set of nested functions in R pseudocode this would look like:

```
park(drive(start_car(find("keys")), to="campus"))
```

Writing it out using pipes give it a more natural (and easier to read) structure:

```
find("keys") %>%
    start_car() %>%
    drive(to = "campus") %>%
    park()
```

Approaches

Basic piping `x %>% f` is equivalent to `f(x)` `x %>% f(y)` is equivalent to `f(x, y)` `x %>% f %>% g` %>% h is equivalent to `h(g(f(x)))`

The argument placeholder

`x %>% f(y, .)` is equivalent to `f(y, x)` `x %>% f(y, z = .)` is equivalent to `f(y, z = x)`

`dplyr` - A Grammar of Data Manipulation

`dplyr` is based on the concepts of functions as verbs that manipulate data frames.

Single data frame functions / verbs:

- – `filter()`: filter rows by condition(s)
- – `slice()`: filter rows using index(es)
- – `select()`: select columns by name
- – `rename()`: rename variables
- – `arrange()`: reorder rows
- – `mutate()`: add new variables
- – `distinct()`: filter for unique rows
- – `sample_n()` / `sample_frac()`: randomly sample rows
- – `summarise()`: reduce variables to values
- – ... (many more)

CEO Data Analysis

Read CEO data

```
# set working directory
setwd("~/PS630-R-Lab/lab-2") # change to your own working directory
# read dta (Stata)
ceo <- read_dta("./CEOSAL2.DTA") # read CEO dataset using haven
```

`filter()` - filter by condition(s)

```
ceo %>% filter(age > 50)
```

```
## # A tibble: 139 x 15
##    salary   age college  grad comten ceoten sales profits mktval lsalary
##     <dbl> <dbl>   <dbl> <dbl>  <dbl>  <dbl> <dbl>   <dbl>  <dbl>   <dbl>
##  1    379    51       1     1      9      3   169      40   1100    5.94
##  2    651    55       1     0     22     22  1100     -54   1000    6.48
##  3   1067    64       1     1      7      7 19000     614   3900    6.97
##  4    945    59       1     0     35     10   536      24    623    6.85
##  5   1261    63       1     1     32      8  4800     191   2100    7.14
##  6   1094    64       1     1     39      5  2900     230   3900    7.00
```

```
##  7    601    54       1     1     26      7  1200      34    533    6.40
##  8    355    66       1     0     39      8   560       8    477    5.87
##  9   1200    72       1     0     37     37   796      35    678    7.09
## 10    697    51       1     0     25      1  8200     234   5700    6.55
## # ... with 129 more rows, and 5 more variables: lsales <dbl>,
## #   lmktval <dbl>, comtensq <dbl>, ceotensq <dbl>, profmarg <dbl>
```

```
ceo %>% filter(age == 50, salary > 1000 )
```

```
## # A tibble: 2 x 15
##   salary   age college  grad comten ceoten sales profits mktval lsalary
##    <dbl> <dbl>   <dbl> <dbl>  <dbl>  <dbl> <dbl>   <dbl>  <dbl>   <dbl>
## 1   1301    50       1     1     19     15  1800     130   1600    7.17
## 2   1142    50       1     1     11      3  2600     166   2200    7.04
## # ... with 5 more variables: lsales <dbl>, lmktval <dbl>, comtensq <dbl>,
## #   ceotensq <dbl>, profmarg <dbl>
```

slice() - first 10 columns

```
ceo %>% slice(1:10)
```

```
## # A tibble: 10 x 15
##    salary   age college  grad comten ceoten sales profits mktval lsalary
##     <dbl> <dbl>   <dbl> <dbl>  <dbl>  <dbl> <dbl>   <dbl>  <dbl>   <dbl>
##  1   1161    49       1     1      9      2  6200     966  23200    7.06
##  2    600    43       1     1     10     10   283      48   1100    6.40
##  3    379    51       1     1      9      3   169      40   1100    5.94
##  4    651    55       1     0     22     22  1100     -54   1000    6.48
##  5    497    44       1     1      8      6   351      28    387    6.21
##  6   1067    64       1     1      7      7 19000     614   3900    6.97
##  7    945    59       1     0     35     10   536      24    623    6.85
##  8   1261    63       1     1     32      8  4800     191   2100    7.14
##  9    503    47       1     1      4      4   610       7    454    6.22
## 10   1094    64       1     1     39      5  2900     230   3900    7.00
## # ... with 5 more variables: lsales <dbl>, lmktval <dbl>, comtensq <dbl>,
## #   ceotensq <dbl>, profmarg <dbl>
```

slice() - last 5 columns

```
ceo %>% slice((n()-4):n())
```

```
## # A tibble: 5 x 15
##   salary   age college  grad comten ceoten sales profits mktval lsalary
##    <dbl> <dbl>   <dbl> <dbl>  <dbl>  <dbl> <dbl>   <dbl>  <dbl>   <dbl>
## 1    264    63       1     0     42      3   334      43    480    5.58
## 2    185    58       1     0     39      1   766      49    560    5.22
## 3    387    71       1     1     32     13   432      28    477    5.96
## 4   2220    63       1     1     18     18   277     -80    540    7.71
## 5    445    69       1     0     23      0   249      31    828    6.10
## # ... with 5 more variables: lsales <dbl>, lmktval <dbl>, comtensq <dbl>,
## #   ceotensq <dbl>, profmarg <dbl>
```

select() - select columns

```
ceo %>% select(salary, profmarg)
```

```
## # A tibble: 177 x 2
##    salary profmarg
##     <dbl>    <dbl>
##  1   1161    15.6
##  2    600    17.0
##  3    379    23.7
##  4    651    -4.91
##  5    497     7.98
##  6   1067     3.23
##  7    945     4.48
##  8   1261     3.98
##  9    503     1.15
## 10   1094     7.93
## # ... with 167 more rows
```

select() - exclude columns

```
ceo %>% select(-salary, -profmarg)
```

```
## # A tibble: 177 x 13
##      age college  grad comten ceoten sales profits mktval lsalary lsales
##    <dbl>   <dbl> <dbl>  <dbl>  <dbl> <dbl>   <dbl>  <dbl>   <dbl>  <dbl>
##  1    49       1     1      9      2  6200     966  23200    7.06   8.73
##  2    43       1     1     10     10   283      48   1100    6.40   5.65
##  3    51       1     1      9      3   169      40   1100    5.94   5.13
##  4    55       1     0     22     22  1100     -54   1000    6.48   7.00
##  5    44       1     1      8      6   351      28    387    6.21   5.86
##  6    64       1     1      7      7 19000     614   3900    6.97   9.85
##  7    59       1     0     35     10   536      24    623    6.85   6.28
##  8    63       1     1     32      8  4800     191   2100    7.14   8.48
##  9    47       1     1      4      4   610       7    454    6.22   6.41
## 10    64       1     1     39      5  2900     230   3900    7.00   7.97
## # ... with 167 more rows, and 3 more variables: lmktval <dbl>,
## #   comtensq <dbl>, ceotensq <dbl>
```

```
ceo %>% select(-c(salary, profmarg))
```

```
## # A tibble: 177 x 13
##      age college  grad comten ceoten sales profits mktval lsalary lsales
##    <dbl>   <dbl> <dbl>  <dbl>  <dbl> <dbl>   <dbl>  <dbl>   <dbl>  <dbl>
##  1    49       1     1      9      2  6200     966  23200    7.06   8.73
##  2    43       1     1     10     10   283      48   1100    6.40   5.65
##  3    51       1     1      9      3   169      40   1100    5.94   5.13
##  4    55       1     0     22     22  1100     -54   1000    6.48   7.00
##  5    44       1     1      8      6   351      28    387    6.21   5.86
##  6    64       1     1      7      7 19000     614   3900    6.97   9.85
##  7    59       1     0     35     10   536      24    623    6.85   6.28
##  8    63       1     1     32      8  4800     191   2100    7.14   8.48
##  9    47       1     1      4      4   610       7    454    6.22   6.41
```

```
## 10    64       1     1     39      5  2900     230   3900    7.00   7.97
## # ... with 167 more rows, and 3 more variables: lmktval <dbl>,
## #   comtensq <dbl>, ceotensq <dbl>
```

select() - ranges

```
ceo %>% select(salary:college)
```

```
## # A tibble: 177 x 3
##    salary   age college
##     <dbl> <dbl>   <dbl>
##  1   1161    49       1
##  2    600    43       1
##  3    379    51       1
##  4    651    55       1
##  5    497    44       1
##  6   1067    64       1
##  7    945    59       1
##  8   1261    63       1
##  9    503    47       1
## 10   1094    64       1
## # ... with 167 more rows
```

select() - excluse ranges

```
ceo %>% select(-c(salary:college))
```

```
## # A tibble: 177 x 12
##     grad comten ceoten sales profits mktval lsalary lsales lmktval comtensq
##    <dbl>  <dbl>  <dbl> <dbl>   <dbl>  <dbl>   <dbl>  <dbl>   <dbl>    <dbl>
##  1     1      9      2  6200     966  23200    7.06   8.73   10.1        81
##  2     1     10     10   283      48   1100    6.40   5.65    7.00      100
##  3     1      9      3   169      40   1100    5.94   5.13    7.00       81
##  4     0     22     22  1100     -54   1000    6.48   7.00    6.91      484
##  5     1      8      6   351      28    387    6.21   5.86    5.96       64
##  6     1      7      7 19000     614   3900    6.97   9.85    8.27       49
##  7     0     35     10   536      24    623    6.85   6.28    6.43     1225
##  8     1     32      8  4800     191   2100    7.14   8.48    7.65     1024
##  9     1      4      4   610       7    454    6.22   6.41    6.12       16
## 10     1     39      5  2900     230   3900    7.00   7.97    8.27     1521
## # ... with 167 more rows, and 2 more variables: ceotensq <dbl>,
## #   profmarg <dbl>
```

rename() - change column names

```
names(ceo)
```

```
##  [1] "salary"   "age"      "college"  "grad"     "comten"   "ceoten"
##  [7] "sales"    "profits"  "mktval"   "lsalary"  "lsales"   "lmktval"
## [13] "comtensq" "ceotensq" "profmarg"
```

```
ceo_new <- ceo %>% rename(profit_margin = profmarg)
names(ceo_new)
```

```
##  [1] "salary"        "age"           "college"       "grad"
##  [5] "comten"        "ceoten"        "sales"         "profits"
##  [9] "mktval"        "lsalary"       "lsales"        "lmktval"
## [13] "comtensq"      "ceotensq"      "profit_margin"
```

arrange() - sort data

```
ceo %>%
  # filter if age is larger than 50
  filter(age > 50) %>%
  # sort by age and then salary
  arrange(age,salary)
```

```
## # A tibble: 139 x 15
##    salary   age college  grad comten ceoten sales profits mktval lsalary
##     <dbl> <dbl>   <dbl> <dbl>  <dbl>  <dbl> <dbl>   <dbl>  <dbl>   <dbl>
##  1    246    51       1     0      8      8    78      13    458    5.51
##  2    379    51       1     1      9      3   169      40   1100    5.94
##  3    541    51       1     0     30      4  1400      82   1200    6.29
##  4    697    51       1     0     25      1  8200     234   5700    6.55
##  5   1487    51       1     0      3      3 22200     182   2800    7.30
##  6    483    52       1     1     18     14  1000      35    548    6.18
##  7    515    52       1     1     27      1  1100      51    889    6.24
##  8    552    52       1     0     30      1  2800     308   3500    6.31
##  9    704    52       1     1      6      6    50       8    903    6.56
## 10    999    52       1     0     28     17   159      21    398    6.91
## # ... with 129 more rows, and 5 more variables: lsales <dbl>,
## #   lmktval <dbl>, comtensq <dbl>, ceotensq <dbl>, profmarg <dbl>
```

```
ceo %>%
  # filter if age is larger than 50
  filter(age > 50) %>%
  # sort by age (descend)
  arrange(desc(age),salary)
```

```
## # A tibble: 139 x 15
##    salary   age college  grad comten ceoten sales profits mktval lsalary
##     <dbl> <dbl>   <dbl> <dbl>  <dbl>  <dbl> <dbl>   <dbl>  <dbl>   <dbl>
##  1    425    86       1     1     13     13    36      11    644    6.05
##  2    396    80       1     0     58     28   513      53    963    5.98
##  3    300    77       0     0     45     26  6900     483   4700    5.70
##  4   1946    73       1     0     25     21  7800     484   8000    7.57
##  5    971    72       1     1     33     24  1400      69    609    6.88
##  6   1200    72       1     0     37     37   796      35    678    7.09
##  7    387    71       1     1     32     13   432      28    477    5.96
##  8   1675    71       0     0     31     12   674     115   1200    7.42
##  9    174    69       1     0     13     13    29       6    390    5.16
## 10    445    69       1     0     23      0   249      31    828    6.10
## # ... with 129 more rows, and 5 more variables: lsales <dbl>,
## #   lmktval <dbl>, comtensq <dbl>, ceotensq <dbl>, profmarg <dbl>
```

`mutate()` - modify columns

```
ceo %>%
  # add a new variable salary_l: log(salary)
  mutate(salary_l = log(salary)) %>%
  # select salary and salary_l
  select(salary, salary_l, lsalary)
```

```
## # A tibble: 177 x 3
##    salary salary_l lsalary
##     <dbl>    <dbl>   <dbl>
##  1   1161     7.06    7.06
##  2    600     6.40    6.40
##  3    379     5.94    5.94
##  4    651     6.48    6.48
##  5    497     6.21    6.21
##  6   1067     6.97    6.97
##  7    945     6.85    6.85
##  8   1261     7.14    7.14
##  9    503     6.22    6.22
## 10   1094     7.00    7.00
## # ... with 167 more rows
```

`distinct()` - find unique rows

```
ceo %>%
  select(age, ceoten) %>%
  distinct() %>%
  arrange(age, ceoten)
```

```
## # A tibble: 147 x 2
##      age ceoten
##    <dbl>  <dbl>
##  1    33      9
##  2    38      3
##  3    39      3
##  4    39      8
##  5    40      1
##  6    40      5
##  7    40     11
##  8    41      2
##  9    42     12
## 10    43      2
## # ... with 137 more rows
```

`sample_n()` sampling rows

```
ceo %>% sample_n(100)
```

```
## # A tibble: 100 x 15
##    salary   age college  grad comten ceoten sales profits mktval lsalary
##     <dbl> <dbl>   <dbl> <dbl>  <dbl>  <dbl> <dbl>   <dbl>  <dbl>   <dbl>
```

```
##  1    733    60       1     0      8      8   347      18    778    6.60
##  2    585    60       1     1     36     10  1700      33    449    6.37
##  3    491    43       1     1     21      2   561      54    521    6.20
##  4   1268    47       1     0     20      4  1100      47   2500    7.15
##  5   1067    64       1     1      7      7 19000     614   3900    6.97
##  6   1162    58       1     0     24      6  3800     226   1800    7.06
##  7    449    56       1     0     31      1   661      37    538    6.11
##  8    379    51       1     1      9      3   169      40   1100    5.94
##  9    834    61       1     1     32      0  7600     364   5300    6.73
## 10    581    54       1     0     19     19   408      23    403    6.36
## # ... with 90 more rows, and 5 more variables: lsales <dbl>,
## #   lmktval <dbl>, comtensq <dbl>, ceotensq <dbl>, profmarg <dbl>
```

summarise() - summarize data

```
ceo %>%
  # provide summary statistic: # of obs, min, max
  summarize(n = n(),
            mean = mean(salary,na.rm = T),
            min = min(salary,na.rm = T),
            max = max(salary,na.rm = T))
```

```
## # A tibble: 1 x 4
##       n  mean   min   max
##   <int> <dbl> <dbl> <dbl>
## 1   177  866.   100  5299
```

Tabulate Data by grad

```
# creat your own contingency table
ceo_tab = ceo %>%
  # define subgroups
  group_by(grad) %>%
  # provide summary statistic: # of obs, min, max
  summarize(n = n(),
            mean = mean(salary,na.rm = T),
            sd = var(salary,na.rm = T) %>% sqrt(.),
            min = min(salary,na.rm = T),
            max = max(salary,na.rm = T))
ceo_tab
```

```
## # A tibble: 2 x 6
##    grad     n  mean    sd   min   max
##   <dbl> <int> <dbl> <dbl> <dbl> <dbl>
## 1     0    83  868.  675.   174  5299
## 2     1    94  864.  501.   100  2265
```

Using xtable() to export

```
xtable(ceo_tab)
```

% latex table generated in R 3.5.1 by xtable 1.8-3 package % Thu Sep 12 16:14:47 2019

	grad	n	mean	sd	min	max
1	0.00	83	867.73	675.22	174.00	5299.00
2	1.00	94	864.21	501.39	100.00	2265.00

T-test

Tabulate your data

```
# creat your own table
tab <- ceo %>%
  # define subgroups
  group_by(grad) %>%
  # provide summary statistic: # of obs, min, max
  summarize(n = n(),
            mean = mean(profmarg,na.rm = T),
            sd = var(profmarg,na.rm = T) %>% sqrt(.),
            min = min(profmarg,na.rm = T),
            max = max(profmarg,na.rm = T))
tab
```

```
## # A tibble: 2 x 6
##    grad     n  mean    sd    min   max
##   <dbl> <int> <dbl> <dbl>  <dbl> <dbl>
## 1     0    83  6.71  8.23  -48.1  23.9
## 2     1    94  6.16 23.3  -203.   47.5
```

Construct T-statistic

$$t = \frac{\bar{X}_t - \bar{X}_c}{\hat{\sigma}_{\bar{X}_t - \bar{X}_c}}$$

where:

$$\hat{\sigma}_{\bar{X}_t - \bar{X}_c} = \sqrt{\frac{\hat{\sigma}_{\bar{X}_t}}{n_t} + \frac{\hat{\sigma}_{\bar{X}_c}}{n_c}}$$

```
# number of observations
n_c <- tab$n[1]
n_t <- tab$n[2]

# mean
mean_c <- tab$mean[1]
mean_t <- tab$mean[2]

# standard deviation
sd_c <- tab$sd[1]
sd_t <- tab$sd[2]

# compute sigma
signma_tc <- sqrt(sd_c^2/n_c + sd_t^2/n_t)
```

```r
# compute t-statistic
t_test <- (mean_t - mean_c)/ signma_tc

t_test
```

```
## [1] -0.2138428
```

The degrees of freedom

R uses Welch DoF, which is estimated as follows:

$$\nu_W = \frac{\left(\frac{s_t^2}{n_t} + \frac{s_c^2}{n_c}\right)^2}{\frac{s_t^4}{n_t^2\nu_t} + \frac{s_c^4}{n_c^2\nu_c}}$$

```r
# numerator
num = (sd_t^2/n_t + sd_c^2/n_c)^2

# denominator
den = sd_t^4/( (n_t^2) * (n_t - 1)) + sd_c^4/( (n_c^2) * (n_c - 1))

# degrees of freedom
dof = num/den
dof
```

```
## [1] 118.4336
```

Confidence Interval

$$CI = (\bar{X} - t^* * \hat{\sigma}_{\bar{X}_t - \bar{X}_c}, \bar{X} + t^* * \hat{\sigma}_{\bar{X}_t - \bar{X}_c})$$

where $\sigma_{\bar{X}_t - \bar{X}_c}$ is the estimated standard error, t^* is the critical value of t for the desired level of confidence.

```r
mean_dif <-  mean_c - mean_t
mean_dif
```

```
## [1] 0.5494467
```

```r
critical_t <- qt(0.05/2, dof)

mean_dif - critical_t*signma_tc
```

```
## [1] 5.637355
```

```r
mean_dif + critical_t*signma_tc
```

```
## [1] -4.538462
```

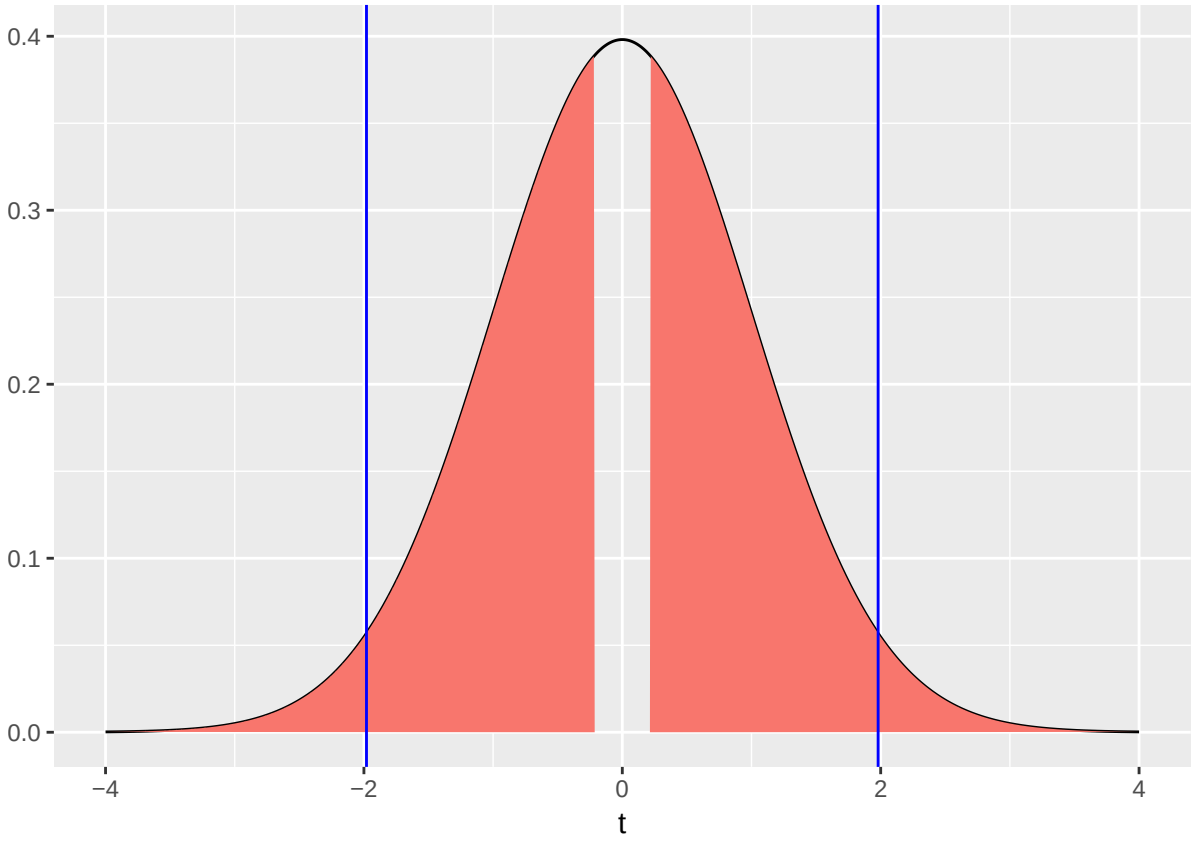
P value

$$|T| > t_{1-\alpha/2,\nu}$$

```r
2*pt(-abs(t_test),df = dof) # pt() is the distribution function of t Distribution
```

```
## [1] 0.8310373
```

Visualizing P value



T-test using `t.test`

```r
t.test( profmarg ~ grad  , data = ceo )
```

```
##
##  Welch Two Sample t-test
##
## data:  profmarg by grad
## t = 0.21384, df = 118.43, p-value = 0.831
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.538462  5.637355
## sample estimates:
## mean in group 0 mean in group 1
##        6.711907        6.162460
```