Statistical tests

Abhijit Dasgupta

BIOF 339

Comparing two groups

The t-test

The t-test compares whether the mean of a variable differs between two groups.

It does assume the normal distribution for the data, but is robust to deviations from normality

Do not test for normality before doing the t-test. It isn't necessary and screws up your error rates

The t-test

In R, there is a convenient function `t.test`

```
t.test(NP_958782 ~ ER.Status, data = brca)
```

```

    Welch Two Sample t-test

data:  NP_958782 by ER.Status
t = 0.63522, df = 41.807, p-value = 0.5287
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3523151  0.6759226
sample estimates:
mean in group Negative mean in group Positive 
             0.4292798              0.2674761 
```

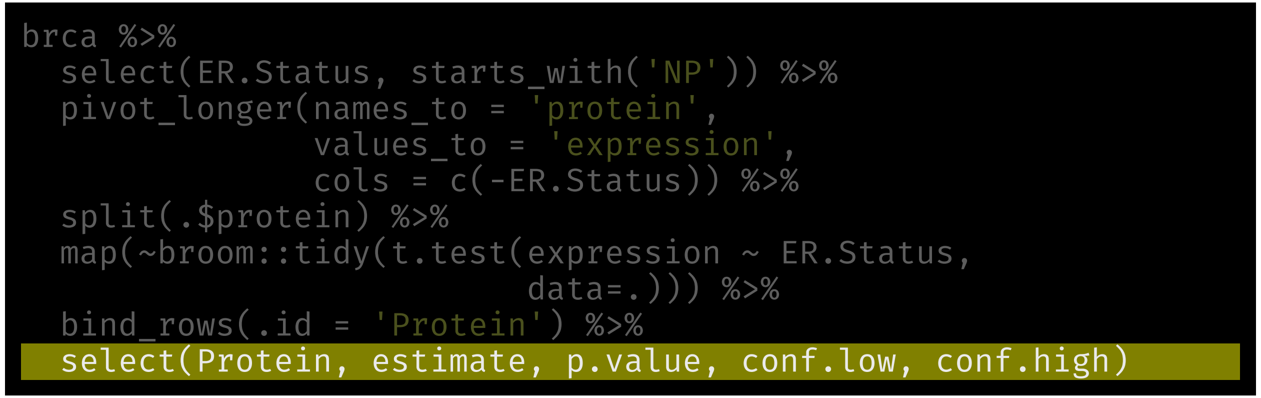
Read the code as

"Do a t-test to see if (the mean of) `NP_958782` differs by `ER.Status`, where both are taken from the data set `brca`"

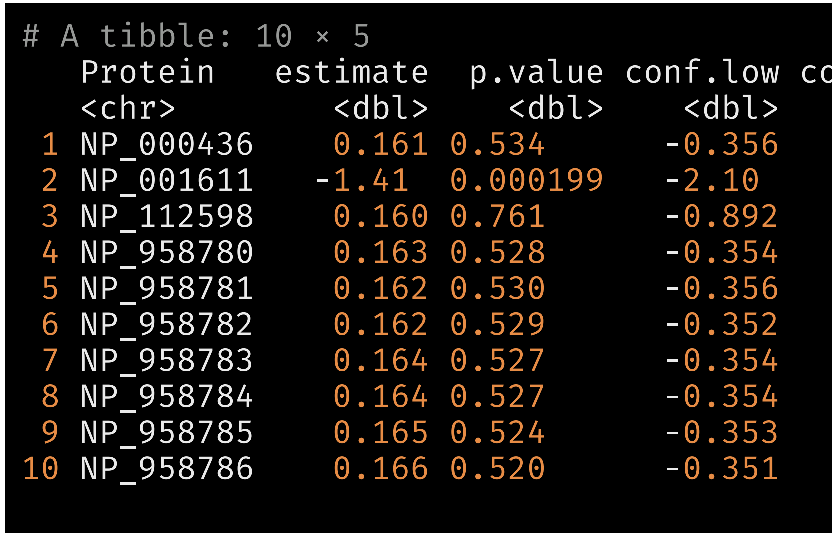
You can read the `~` as "by", as in "t-test of NP_958782 by ER.Status"

Using broom

The fact that broom::tidy makes the results of tests into tibbles is in fact extremely useful in high-throughput work



```
brca %>%
  select(ER.Status, starts_with('NP')) %>%
  pivot_longer(names_to = 'protein',
               values_to = 'expression',
               cols = c(-ER.Status)) %>%
  split(.$protein) %>%
  map(~broom::tidy(t.test(expression ~ ER.Status,
                          data=.))) %>%
  bind_rows(.id = 'Protein') %>%
  select(Protein, estimate, p.value, conf.low, conf.high)
```



```
# A tibble: 10 × 5
   Protein   estimate  p.value conf.low co
   <chr>        <dbl>    <dbl>    <dbl>
 1 NP_000436    0.161 0.534      -0.356
 2 NP_001611   -1.41  0.000199   -2.10
 3 NP_112598    0.160 0.761      -0.892
 4 NP_958780    0.163 0.528      -0.354
 5 NP_958781    0.162 0.530      -0.356
 6 NP_958782    0.162 0.529      -0.352
 7 NP_958783    0.164 0.527      -0.354
 8 NP_958784    0.164 0.527      -0.354
 9 NP_958785    0.165 0.524      -0.353
10 NP_958786    0.166 0.520      -0.351
```

Back to testing

Wilcoxon test, nonparametric t-test

```
wilcox.test(NP_958782 ~ ER.Status, data=brca) %>%
  broom::tidy()
```

```
# A tibble: 1 × 4
  statistic p.value method                                            alternative
      <dbl>   <dbl> <chr>                                             <chr>
1       755   0.590 Wilcoxon rank sum test with continuity correction two.sided
```

```

	Wilcoxon rank sum test with continuity correction

data:  NP_958782 by ER.Status
W = 755, p-value = 0.5897
alternative hypothesis: true location shift is not equal to 0
```

Wilcoxon test

```
brca %>%
  select(ER.Status, starts_with('NP')) %>%
  tidyr::gather(protein,expression, -ER.Status) %>%
  split(.$protein) %>%
  map(~broom::tidy(wilcox.test(expression ~ ER.Statu
                         data=.))) %>%
  bind_rows(.id='Protein') %>%
  select(Protein, p.value)
```

```
# A tibble: 10 × 2
   Protein     p.value
   <chr>         <dbl>
 1 NP_000436 0.583
 2 NP_001611 0.0000928
 3 NP_112598 0.939
 4 NP_958780 0.583
 5 NP_958781 0.576
 6 NP_958782 0.590
 7 NP_958783 0.583
 8 NP_958784 0.576
 9 NP_958785 0.576
10 NP_958786 0.576
```

Using tableone

```
CreateTableOne(
  data = brca %>% filter(!is.na(ER.Status)),
  vars = brca %>%
    select(starts_with('NP')) %>%
    names(),
  strata = 'ER.Status',
  test = T,
  testNormal = t.test
)
```

```
                        Stratified by ER.Status
                         Negative     Positive     p      test
  n                         38           69
  NP_958782 (mean (SD))  0.43 (1.13)  0.27 (0.93)  0.498
  NP_958785 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.492
  NP_958786 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.487
  NP_000436 (mean (SD))  0.43 (1.14)  0.27 (0.93)  0.502
  NP_958781 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.499
  NP_958780 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.496
  NP_958783 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.495
  NP_958784 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.495
  NP_112598 (mean (SD)) -0.20 (2.28) -0.36 (1.97)  0.748
  NP_001611 (mean (SD)) -0.57 (1.54)  0.84 (1.19) <0.001
```

This is not quite the same results as before

Using tableone

```
CreateTableOne(
  data = brca %>% filter(!is.na(ER.Status)),
  vars = brca %>%
    select(starts_with('NP')) %>%
    names(),
  strata = 'ER.Status',
  test = T,
  testNormal = t.test,
  argsNormal = list(var.equal=F)
)
```

```
                        Stratified by ER.Status
                         Negative      Positive      p      test
  n                         38            69
  NP_958782 (mean (SD))  0.43 (1.13)  0.27 (0.93)  0.529
  NP_958785 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.524
  NP_958786 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.520
  NP_000436 (mean (SD))  0.43 (1.14)  0.27 (0.93)  0.534
  NP_958781 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.530
  NP_958780 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.528
  NP_958783 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.527
  NP_958784 (mean (SD))  0.44 (1.14)  0.27 (0.93)  0.527
  NP_112598 (mean (SD)) -0.20 (2.28) -0.36 (1.97)  0.761
  NP_001611 (mean (SD)) -0.57 (1.54)  0.84 (1.19) <0.001
```

Tests for discrete data

Testing whether the distribution of a categorical variable differs by levels of another categorical variable can be done using either the Chi-square test (`chisq.test`) or the Fisher's test (`fisher.test`). Both require you to create a 2x2 table first.

```
fisher.test(table(brca$Tumor, brca$ER.Status))
```

```

	Fisher's Exact Test for Count Data

data:  table(brca$Tumor, brca$ER.Status)
p-value = 0.6003
alternative hypothesis: two.sided
```

Tests for discrete data

Testing whether the distribution of a categorical variable differs by levels of another categorical variable can be done using either the Chi-square test (`chisq.test`) or the Fisher's test (`fisher.test`). Both require you to create a 2x2 table first.

```
chisq.test(table(brca$Tumor, brca$ER.Status))
```

```

	Pearson's Chi-squared test

data:  table(brca$Tumor, brca$ER.Status)
X-squared = 2.094, df = 3, p-value = 0.5531
```

Tests for discrete data

We can use `broom::tidy` for either of these

```
chisq.test(table(brca$Tumor, brca$ER.Status)) %>%
  broom::tidy()
```

```
# A tibble: 1 × 4
  statistic p.value parameter method
      <dbl>   <dbl>     <int> <chr>
1      2.09   0.553         3 Pearson's Chi-squared test
```

Using tableone

```
CreateCatTable(vars = c('Tumor','Node','Metastasis'),
               data = filter(brca, !is.na(ER.Status)),
               strata = 'ER.Status',
               test = T) # chisq.test
```

```
                     Stratified by ER.Status
                      Negative   Positive   p      test
  n                   38         69
  Tumor (%)                                  0.553
     T1                6 (15.8)  10 (14.5)
     T2               26 (68.4)  40 (58.0)
     T3                5 (13.2)  14 (20.3)
     T4                1 ( 2.6)   5 ( 7.2)
  Node (%)                                   0.685
     N0               22 (57.9)  32 (46.4)
     N1                8 (21.1)  21 (30.4)
     N2                5 (13.2)  10 (14.5)
     N3                3 ( 7.9)   6 ( 8.7)
  Metastasis = M1 (%)  1 ( 2.6)   1 ( 1.4)   1.000
```

Using tableone

```
c1 <- CreateCatTable(vars = c('Tumor','Node','Metastasis'),
               data = filter(brca, !is.na(ER.Status)),
               strata = 'ER.Status',
               test = T)
print(c1, exact = c('Tumor','Node','Metastasis')) # fisher.test
```

```
                     Stratified by ER.Status
                      Negative   Positive   p      test
  n                   38         69
  Tumor (%)                                  0.600 exact
     T1                6 (15.8)  10 (14.5)
     T2               26 (68.4)  40 (58.0)
     T3                5 (13.2)  14 (20.3)
     T4                1 ( 2.6)   5 ( 7.2)
  Node (%)                                   0.695 exact
     N0               22 (57.9)  32 (46.4)
     N1                8 (21.1)  21 (30.4)
     N2                5 (13.2)  10 (14.5)
     N3                3 ( 7.9)   6 ( 8.7)
  Metastasis = M1 (%)  1 ( 2.6)   1 ( 1.4)   1.000 exact
```