5.1 介绍

在前面已经介绍了一些新的理念，为了帮助你理解这些东西，我们这次使用一个数据，通过使用shiny来探索这个数据集，将之前学习的新思想都用起来。我们先在探索数据，然后将这部分打包成shiny app。
在这一章里面，我们将使用`vroom`读取数据，使用`tidyverse`包来对数据分析。

```
library(shiny)
library(vroom)
library(tidyverse)
```

5.2 数据

这个数据是记录了美国代表性医院的一些样本事故。这个数据可以在这个网站上获得：https://github.com/hadley/neiss 。

在这一章，我们将注意力转到2017年的数据，这样保证数据足够小。这个章节的代码都在这个网站里面： https://github.com/hadley/mastering-shiny/blob/master/neiss/data.R 。

我们使用的数据集叫`injuries`。包含250000个观测值。

```
injuries <- vroom::vroom("neiss/injuries.tsv.gz")
injuries
#> # A tibble: 255,064 x 10
#>   trmt_date    age sex   race  body_part diag  location prod_code weight
#>   <date>     <dbl> <chr> <chr> <chr>     <chr> <chr>        <dbl>  <dbl>
#> 1 2017-01-01    71 male  white Upper Tr… Cont… Other P…      1807   77.7
#> 2 2017-01-01    16 male  white Lower Arm Burn… Home           676   77.7
#> 3 2017-01-01    58 male  white Upper Tr… Cont… Home           649   77.7
#> 4 2017-01-01    21 male  white Lower Tr… Stra… Home          4076   77.7
#> 5 2017-01-01    54 male  white Head      Inte… Other P…      1807   77.7
#> 6 2017-01-01    21 male  white Hand      Frac… Home          1884   77.7
#> # … with 255,058 more rows, and 1 more variable: narrative <chr>
```

每一行代表一次事故，有着10个变量

1. `trmt_date`是人被送到医院的时间（不是事故发生的时间）
2. `age`、`sex`、`race`是这个人的一些特征
3. `body_part`是这个人身体受伤的位置（比如眼睛、脚踝之类的）。`location`是这个人受伤的地理位置
4. `diag`是伤情的基本诊断（骨折还是擦伤等）
5. `prod_code`是与受伤的原因（爬楼梯之类的）
6. `weight`是用来估计美国有多少人遇到相同的情况
7. `narrative`是关于这个事故的发生经过

我们还是用这个文件夹下的别的数据集：`products`。这个是将上面`injuries`数据里面的`prod_code`的编码和对应的具体内容的一个介绍。`population`数据是美国2017年的`age`和`sex`的人口统计数目：

```
products <- vroom::vroom("neiss/products.tsv")
```

```
products
#> # A tibble: 38 x 2
#>   prod_code title
#>       <dbl> <chr>
#> 1       464 knives, not elsewhere classified
#> 2       474 tableware and accessories
#> 3       604 desks, chests, bureaus or buffets
#> 4       611 bathtubs or showers
#> 5       649 toilets
#> 6       676 rugs or carpets, not specified
#> # … with 32 more rows

population <- vroom::vroom("neiss/population.tsv")
population
#> # A tibble: 170 x 3
#>     age sex    population
#>   <dbl> <chr>       <dbl>
#> 1     0 female    1924145
#> 2     0 male      2015150
#> 3     1 female    1943534
#> 4     1 male      2031718
#> 5     2 female    1965150
#> 6     2 male      2056625
#> # … with 164 more rows
```

5.3 探索

`prod_code`等于1842(爬楼梯，散步）的有多少：

```
selected <- injuries %>% filter(prod_code == 1842)
nrow(selected)
#> [1] 30647
```

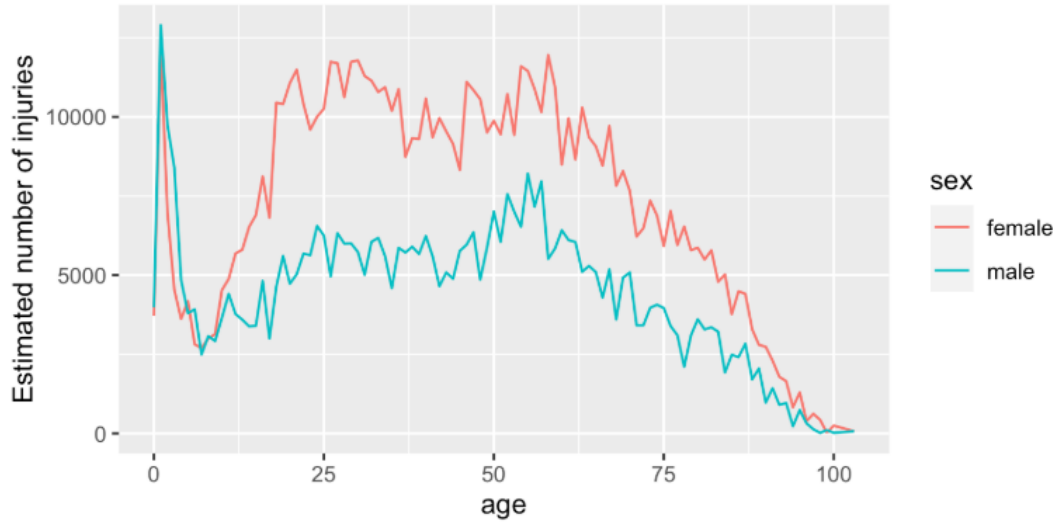
对`weight`的基本统计：

```
selected %>% count(diag, wt = weight, sort = TRUE)
#> # A tibble: 23 x 2
#>   diag                        n
#>   <chr>                   <dbl>
#> 1 Strain, Sprain        267892.
#> 2 Fracture              243082.
#> 3 Other Or Not Stated   227515.
#> 4 Contusion Or Abrasion 195172.
#> 5 Inter Organ Injury    111340.
#> 6 Laceration             89190.
#> # … with 17 more rows

selected %>% count(body_part, wt = weight, sort = TRUE)
#> # A tibble: 25 x 2
#>   body_part         n
#>   <chr>         <dbl>
#> 1 Ankle       183470.
#> 2 Head        174725.
#> 3 Lower Trunk 150459.
#> 4 Knee        112162.
```

```
#> 5 Upper Trunk  98197.
#> 6 Face         73815.
#> # … with 19 more rows

selected %>% count(location, wt = weight, sort = TRUE)
#> # A tibble: 8 x 2
#>   location                        n
#>   <chr>                       <dbl>
#> 1 Home                      647127.
#> 2 Unknown                   458802.
#> 3 Other Public Property      57625.
#> 4 School                     25146.
#> 5 Sports Or Recreation Place 11833.
#> 6 Street Or Highway           2148.
#> # … with 2 more rows
```

从上面可以看出来爬楼梯主要造成脚踝的扭伤、韧带拉伤、骨折。
还可以探索年龄和性别对`weight`的关系：

```
summary <- selected %>%
  count(age, sex, wt = weight)
summary
#> # A tibble: 204 x 3
#>     age sex         n
#>   <dbl> <chr>   <dbl>
#> 1     0 female  3714.
#> 2     0 male    3981.
#> 3     1 female 12155.
#> 4     1 male   12898.
#> 5     2 female  6949.
#> 6     2 male    9730.
#> # … with 198 more rows

summary %>%
  ggplot(aes(age, n, colour = sex)) +
  geom_line() +
  labs(y = "Estimated number of injuries")
```

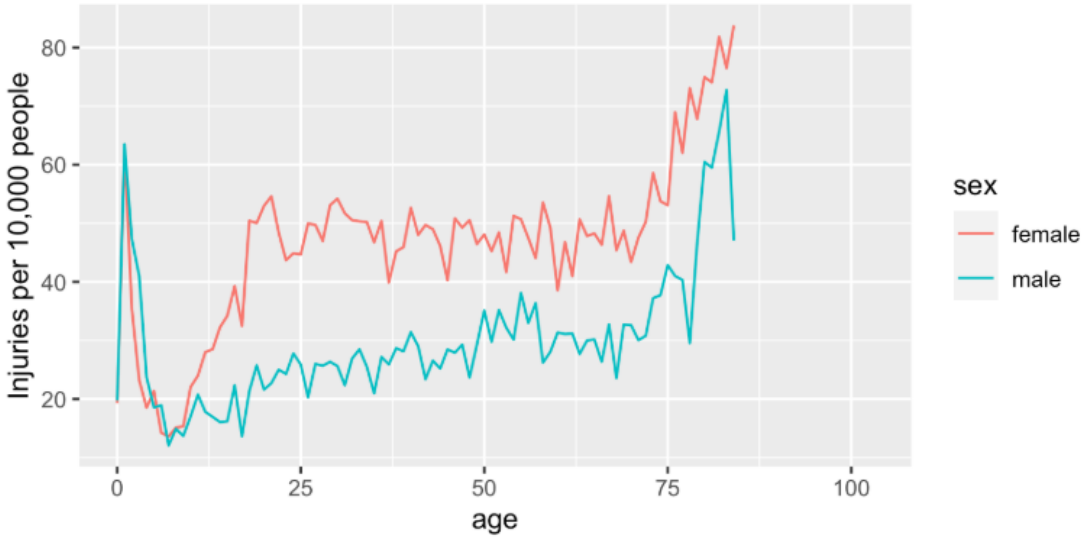


从上面的图可以看出来：当小孩开始学习行走的时候，达到一个高峰，到成年的时候逐渐变平。然后在50多岁的时候逐渐下降，有趣的是：女性受伤的次数比男性要多（难道是因为女的喜欢穿高跟鞋？）老年人受伤数比年轻人少（难道是因为老年人人少？），接下来计算受伤人在对应年龄段和性别的比率（这里又乘以1000）

```
summary <- selected %>%
  count(age, sex, wt = weight) %>%
  left_join(population, by = c("age", "sex")) %>%
  mutate(rate = n / population * 1e4)

summary
#> # A tibble: 204 x 5
#>     age sex         n population  rate
#>   <dbl> <chr>   <dbl>      <dbl> <dbl>
#> 1     0 female  3714.    1924145  19.3
#> 2     0 male    3981.    2015150  19.8
#> 3     1 female 12155.    1943534  62.5
#> 4     1 male   12898.    2031718  63.5
#> 5     2 female  6949.    1965150  35.4
#> 6     2 male    9730.    2056625  47.3
#> # … with 198 more rows
```

画图代码：

```
summary %>%
  ggplot(aes(age, rate, colour = sex)) +
  geom_line(na.rm = TRUE) +
  labs(y = "Injuries per 10,000 people")
```



上面的图中，可以注意到，50岁以后，受伤的人数因为爬楼梯这件事受伤人数增长。上面的图止于80岁（我也不知道为什么80岁以上的数据就没有了）

下面是一些事故的叙述：

```
selected %>%
  sample_n(10) %>%
  pull(narrative)
#>  [1] "63 YO FEMALE PUT LEG THROUGH NEIGHBORS STEPS. DX ACUTE ANKLE PAIN"

#>  [2] "57 YOM FELL ON STAIRS AND STRAINED LOWER BACK"

#>  [3] "A 66YOF HAD SYNCOPE EPISODE WITH FALL, LOST BALANCE AND FELL DOWN 5
STEPS, FX HIP"
#>  [4] "86 YOM WAS GOING DOWN STAIRS; MISSED A STEP, FELL, HIT HEAD ON
STEP.HEMATOMA OF HEAD."
#>  [5] "21YOF WITH ANKLE SPRAIN AFTER SLIPPING ON STAIRS AT HOME DX SPRAIN*"

#>  [6] "83 YO F C/O HIP PAIN S/P FALL DOWN STEPS AT HOME DX THYROID NODULE,
FALL, PAIN OF LEFT HIP JOINT, HYPOKALEMIA"
#>  [7] "9YOF SUST HEAD INJURY WHEN HE FELL DOWN STAIRS"

#>  [8] "29YOM HXOF R FT FX, PT ON CRUTCHES, PT WAS WALKING DN STAIRS & FELL
INJURED R FT @ 1800. DX R FT INJURY, HX OF FX"
#>  [9] "40YOM FELL DOWN 3 STAIRS: DX ACUTE ANK PN; L ANK SPRAIN/DELTOID LIG
ANKSPRAIN"
#> [10] "LEFT SHOULDER CONTUSION. 36 YOF WAS WALKING UPSTAIRS WHEN SHETRIPPED
AND FELL."
```

5.4 shiny 原型

这里开始制作一个shiny app的原型，原型越简单越好，如果前期做复杂了，后面修改很麻烦。
在第一版shiny app里面，第一行是输入，第二行三列都是输出。

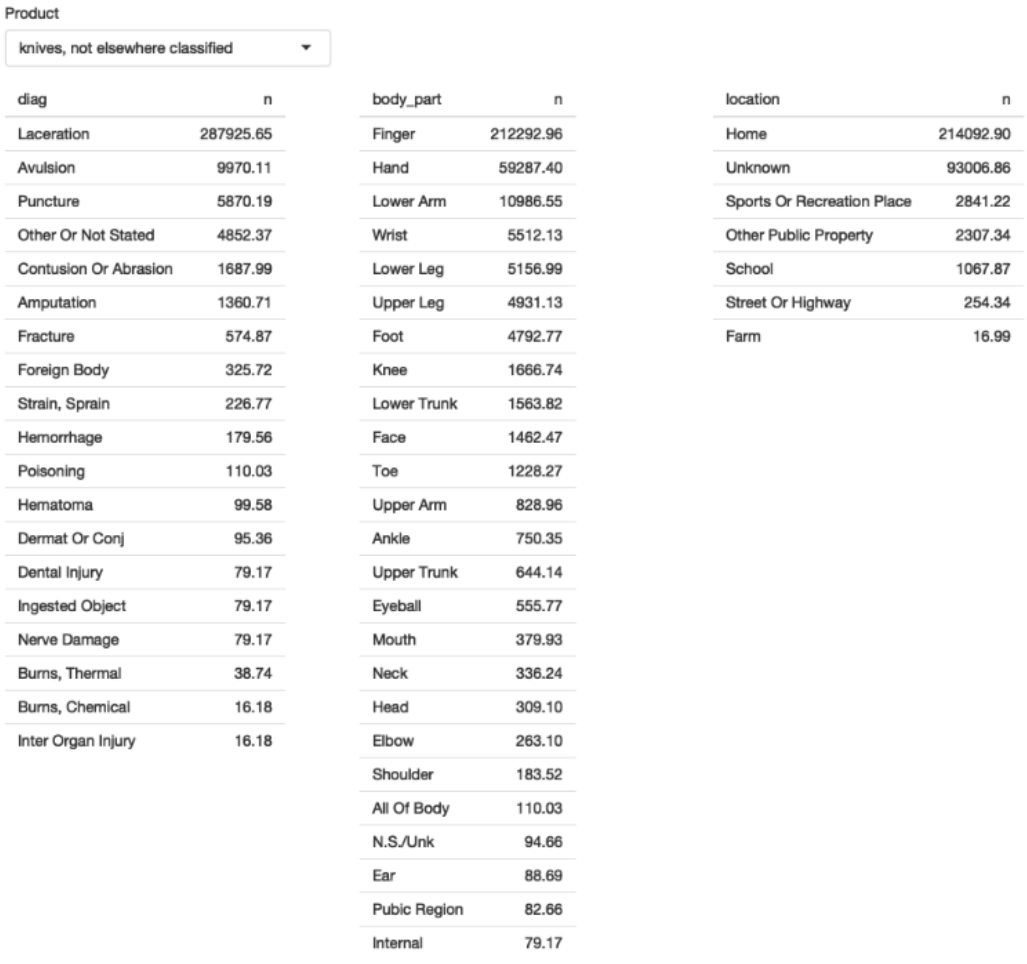
```
ui <- fluidPage(
  fluidRow(
    column(6,
      selectInput("code", "Product", setNames(products$prod_code,
products$title))
    )
  ),
  fluidRow(
    column(4, tableOutput("diag")),
    column(4, tableOutput("body_part")),
    column(4, tableOutput("location"))
  ),
  fluidRow(
    column(12, plotOutput("age_sex"))
  )
)
```

在下面的server部分，写起来和刚开始我们分析的代码一样，只不过是做成反应表达式的样子。通常最好在shiny app还没有启动的时候做一些数据处理工作。

```
server <- function(input, output, session) {
  selected <- reactive(injuries %>% filter(prod_code == input$code))

  output$diag <- renderTable(
    selected() %>% count(diag, wt = weight, sort = TRUE)
```

```
  )
  output$body_part <- renderTable(
    selected() %>% count(body_part, wt = weight, sort = TRUE)
  )
  output$location <- renderTable(
    selected() %>% count(location, wt = weight, sort = TRUE)
  )

  summary <- reactive({
    selected() %>%
      count(age, sex, wt = weight) %>%
      left_join(population, by = c("age", "sex")) %>%
      mutate(rate = n / population * 1e4)
  })

  output$age_sex <- renderPlot({
    summary() %>%
      ggplot(aes(age, n, colour = sex)) +
      geom_line() +
      labs(y = "Estimated number of injuries")
  }, res = 96)
}
```

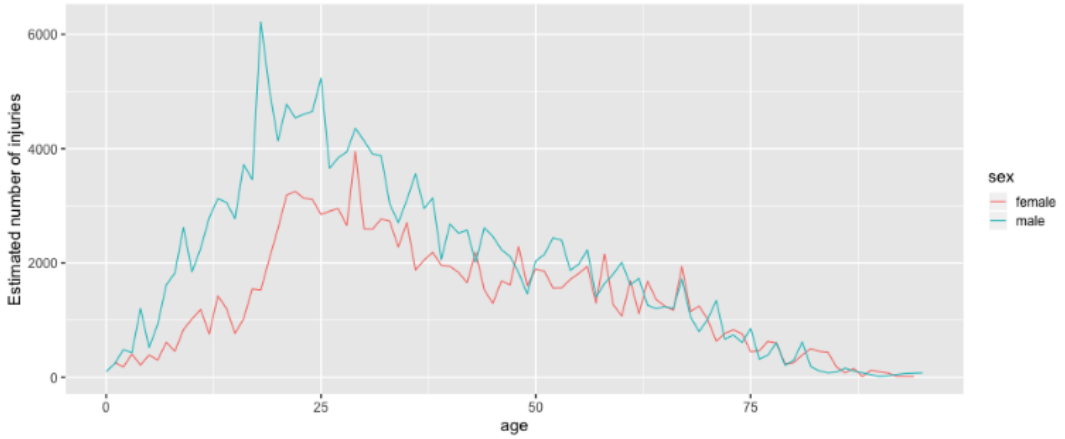
上面的代码 `summary()` 可能不是必须写成反应式的（因为太简单了）。但是实际上，最好将计算和绘图部分分开。这个第一版的源码在网站上：https://github.com/hadley/mastering-shiny/tree/master/neiss/prototype.R



diag	n
Laceration	287925.65
Avulsion	9970.11
Puncture	5870.19
Other Or Not Stated	4852.37
Contusion Or Abrasion	1687.99
Amputation	1360.71
Fracture	574.87
Foreign Body	325.72
Strain, Sprain	226.77
Hemorrhage	179.56
Poisoning	110.03
Hematoma	99.58
Dermat Or Conj	95.36
Dental Injury	79.17
Ingested Object	79.17
Nerve Damage	79.17
Burns, Thermal	38.74
Burns, Chemical	16.18
Inter Organ Injury	16.18

body_part	n
Finger	212292.96
Hand	59287.40
Lower Arm	10986.55
Wrist	5512.13
Lower Leg	5156.99
Upper Leg	4931.13
Foot	4792.77
Knee	1666.74
Lower Trunk	1563.82
Face	1462.47
Toe	1228.27
Upper Arm	828.96
Ankle	750.35
Upper Trunk	644.14
Eyeball	555.77
Mouth	379.93
Neck	336.24
Head	309.10
Elbow	263.10
Shoulder	183.52
All Of Body	110.03
N.S./Unk	94.66
Ear	88.69
Pubic Region	82.66
Internal	79.17

location	n
Home	214092.90
Unknown	93006.86
Sports Or Recreation Place	2841.22
Other Public Property	2307.34
School	1067.87
Street Or Highway	254.34
Farm	16.99



5.5 继续优化

上面的东西，有着基础的框架而且可以运行。我们继续改进。第一个问题是这个app显示太多的信息了。我们只是需要一些重点，因此我们要知道如何截断数据：将变量转换为因子，然后将前五个的所有因子汇众在一起。

```
injuries %>%
  mutate(diag = fct_lump(fct_infreq(diag), n = 5)) %>%
  group_by(diag) %>%
  summarise(n = as.integer(sum(weight)))
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # A tibble: 6 x 2
#>   diag                        n
#>   <fct>                   <int>
#> 1 Other Or Not Stated   1806436
#> 2 Fracture              1558961
#> 3 Laceration            1432407
#> 4 Strain, Sprain        1432556
#> 5 Contusion Or Abrasion 1451987
#> 6 Other                 1929147
```

上面的函数是Hadley写的，如果不懂也没关系，复制粘贴也可以：

```
count_top <- function(df, var, n = 5) {
  df %>%
    mutate({{ var }} := fct_lump(fct_infreq({{ var }}), n = n)) %>%
    group_by({{ var }}) %>%
    summarise(n = as.integer(sum(weight)))
}
```

在server部分我将部分代码修改了：

```
  output$diag <- renderTable(count_top(selected(), diag), width = "100%")
  output$body_part <- renderTable(count_top(selected(), body_part), width =
"100%")
  output$location <- renderTable(count_top(selected(), location), width =
"100%")
```

上面的代码强制表格使用最大的宽度，使得表更加的漂亮

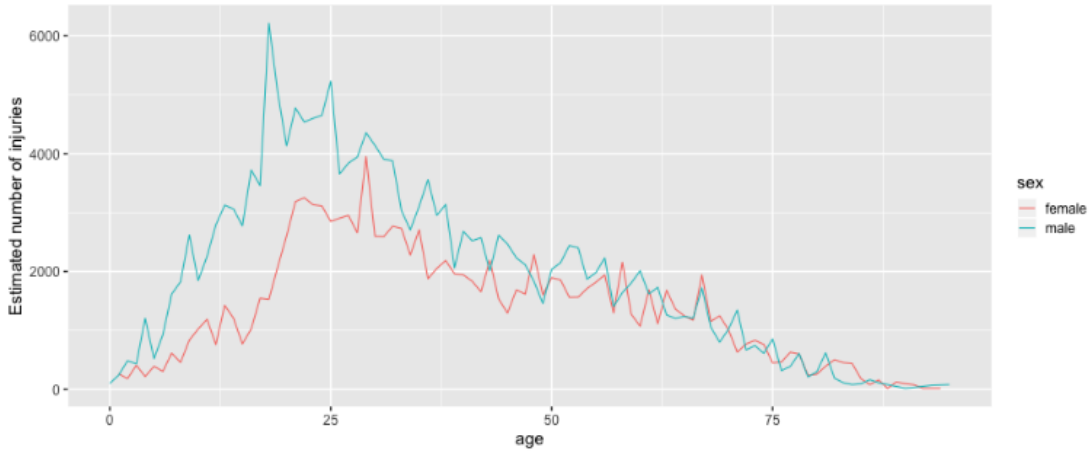
Product

knives, not elsewhere classified

diag	n
Laceration	287925
Avulsion	9970
Puncture	5870
Other Or Not Stated	4852
Contusion Or Abrasion	1687
Other	3281

body_part	n
Finger	212292
Hand	59287
Lower Arm	10986
Wrist	5512
Foot	4792
Other	20715

location	n
Home	214092
Unknown	93006
Other Public Property	2307
Sports Or Recreation Place	2841
School	1067
Other	271



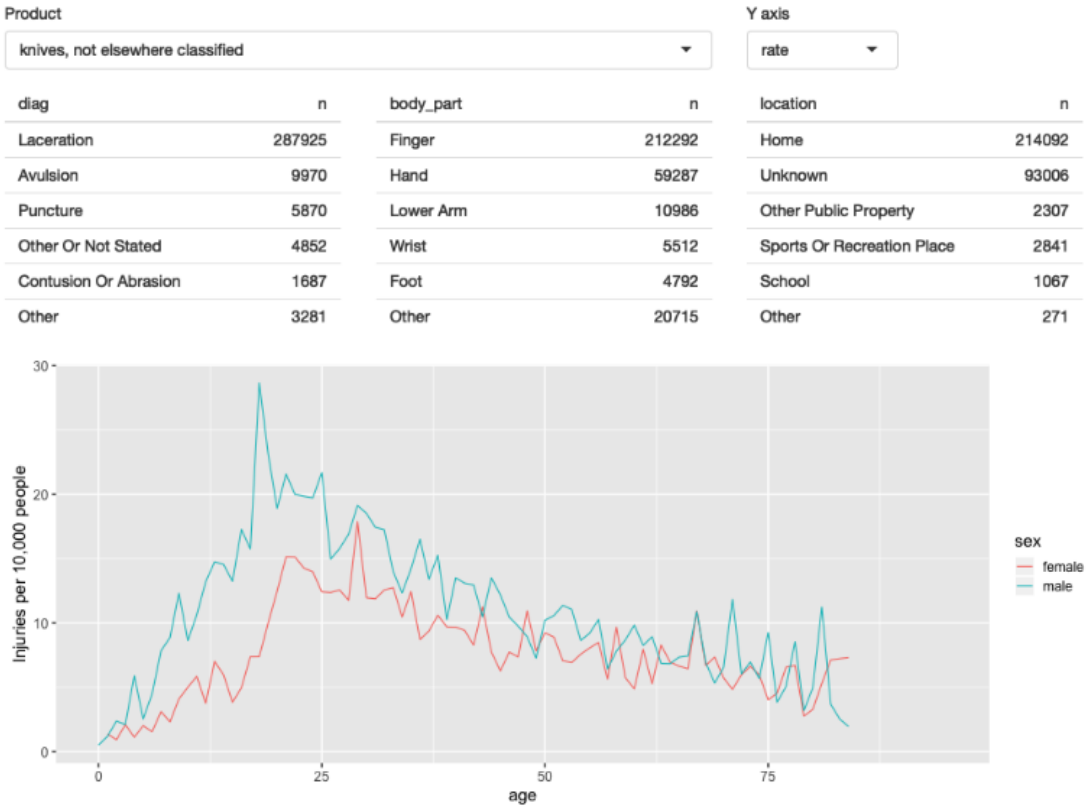
5.6 使用比率还是使用人数

上面我们已经讨论过了，比率和人数绘图的差异，这里，在shiny app中，可以留给用户自己选择。在shiny app中，我在第一行的第二列加入一个选择按钮：

```
fluidRow(
  column(8,
    selectInput("code", "Product",
      choices = setNames(products$prod_code, products$title),
      width = "100%"
    )
  ),
  column(2, selectInput("y", "Y axis", c("rate", "count")))
),
```

默认是`rate`，因为你不用理解年龄段的具体分布。
在画图部分，使用这部分代码：

```
output$age_sex <- renderPlot({
  if (input$y == "count") {
    summary() %>%
      ggplot(aes(age, n, colour = sex)) +
      geom_line() +
      labs(y = "Estimated number of injuries")
  } else {
    summary() %>%
      ggplot(aes(age, rate, colour = sex)) +
      geom_line(na.rm = TRUE) +
      labs(y = "Injuries per 10,000 people")
  }
}, res = 96)
```



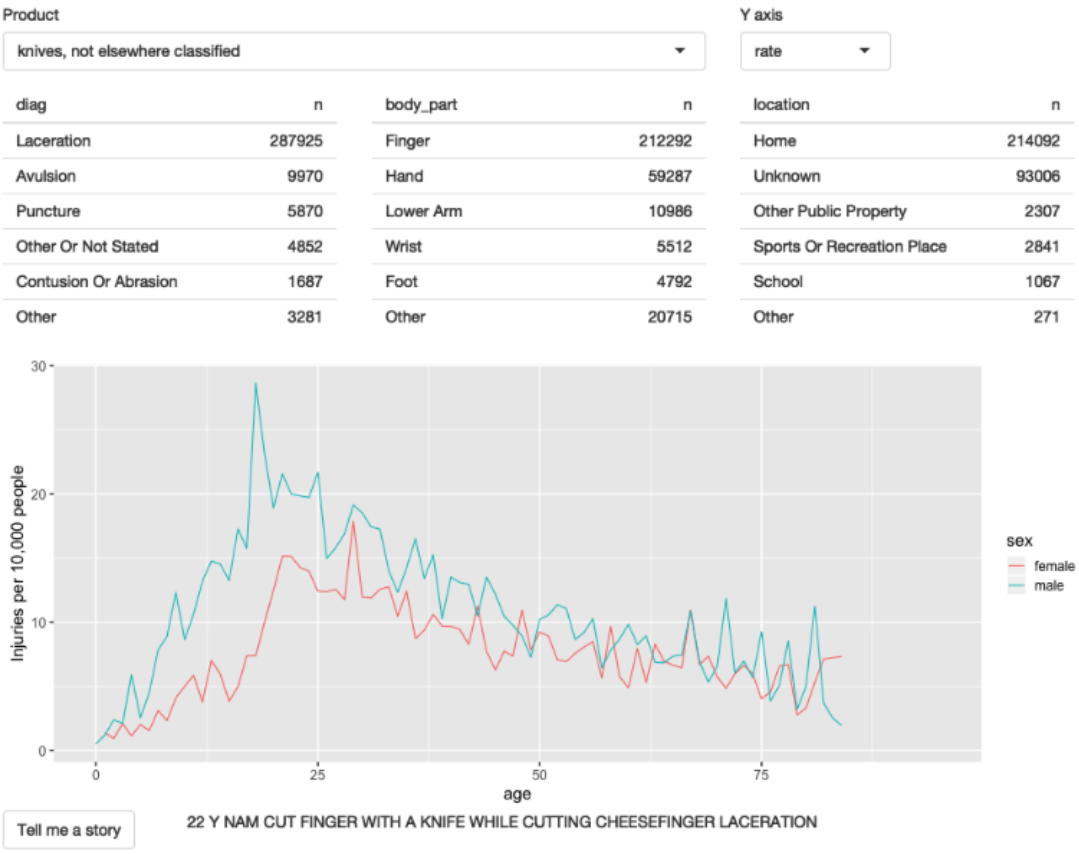
5.7 说故事

我觉得每个事件背后的一些叙述是非常有意识的，因此我也希望可以将这部分展示出来。同样，也有助于你理解图背后的事情。
在shiny app的代码，我加入这部分代码：

```
#ui part
  fluidRow(
    column(2, actionButton("story", "Tell me a story")),
    column(10, textOutput("narrative"))
  )
```

```
# server part
  output$narrative <- renderText({
    input$story
    selected() %>% pull(narrative) %>% sample(1)
  })
```

截图如下：



5.8 练习

内容可能会一直更新，可以查看我的微信公众号：pypi

获得最新的关于这部分的内容

微信扫一扫：

知乎：https://www.zhihu.com/people/fa-fa-1-94

csdn：https://blog.csdn.net/yuanzhoulvpi

github: https://github.com/yuanzhoulvpi2017/master_shiny_CN

如果有错误，欢迎指正，邮箱联系我：yuanzhoulvpi@outlook.com