Aula 3

JAMES HUNTER, PH.D.

28 de maio de 2020

1

Objetivos para Hoje

- Simulação
- Probabilidade e distribuições dos dados
- Inferência simples
- Desenvolver modelos do mundo – regressão

2

Simulação

3

Método Simples de Analisar um Problema

- Formular seu problema em expressões matemáticas - com pelo menos uma variável probabilística
- Calcular o resultado **MUITAS** vezes
- Fazer uma análise estatística dos resultados
- Já fizeram isso com Excel?

4

Problema - Camisetas de Sustentare

- Um guru de marketing sugere que Dr. Wilmar vende camisetas com o nome e logótipo da escola
  - Ajuda fidelizar os alunos e ex-alunos da escola
  - Alunos e ex-alunos expressam orgulho de associação com a escola com o produto
  - Pode ser mais fonte de renda para a Sustentare
  - Muito popular como técnica nos EUA
- Quantos camisetas pode ser vendidos?
- Quanto lucro vai produzir para a escola nos 1º 5 anos?

VSS: Dados são totalmente inventados por Prof. Jim

5

Premissas do Problema

- Sustentare tem agora 4.000 alunos e ex-alunos (`publico_ano1`)
  - Esse número aumenta 250 por ano (`novo_publico`)
- Custo da camiseta já estampada com logótipo este ano é R$32 (`custo_ano1`)
- Custo aumenta 5.0% por ano (`inflacao`)
- Os custos de desenvolver o projeto: R$25.000 (`desenv_ano1`)
  - Todos ocorrem no 1º ano
- A escola vai colocar uma marcação no custo de 100% (`markup`)
- Número dos anos do projeção: 5 anos (`anos`)
- O guru estima que entre 15% (`prop_min`) e 40% (`prop_max`) do público vai comprar em qualquer ano

6

Exercício 9 – Parte A – Colocar as Premissas em Código

- Serie de atribuições
- Colocar todos os valores na forma de variável - Permitir mudança de valor facilmente

7

Parte A – Código

```
publico_ano1 <- 4000 #número total dos alunos/ex-alunos
novo_publico <- 250 #número novo de alunos por ano
custo_ano1 <- 32 #custo da camiseta estampada com logo e nome em reais
inflacao <- .05 #aumento de custos anual (inflação)
markup <- 1 #fator para adicionar ao custo para determinar preço de venda
            #ex. 1 = 100%
desenv_ano1 <- 20000 # cobrança em ano 1 para desenvolver produto, mktg, etc.
anos <- 5L #número de anos para julgar se programa vale a pena
min_prop <- 0.15 # min e max proporções do público que vai comprar
max_prop <- 0.40
```

8

Simulação – Modelo

- Como vamos usar essas variáveis?
- Queremos determinar o lucro para a escola (`lucro`)
  - Que é que sobra depois dos 5 anos de `renda` menos (`custo` * `compras`)
  - Também reduzido pelo custo de desenvolvimento (`desenv`)
- `renda` é resultado do `preco` multiplicado pelas `compras`
- `compras` em qualquer ano é a proporção do público (`prop_compras`) que compra a camiseta
- `preco` em qualquer ano é produto do `custo` por camiseta multiplicado pela marcação (`markup`)

9

Agora, o modelo incorpora todos as variáveis que precisamos

10

Criar uma Base de Dados para as Variáveis

Criar uma Base de Dados para as Variáveis

- Vai ser uma base de `anos` fileiras e 7 colunas
- 1ª 4 colunas podem ser calculados simplesmente das variáveis com valores fixos
- Últimas 3 (`compras`, `renda` e `lucro`) dependem do cálculo probabilístico
- Fazer em 2 grupos: variáveis fixas e variáveis estocásticas
- Pode ver que todas os valores são 0

11

Cálculo de Variáveis Fixas

- `desenv` só precisa um valor, relacionado ao ano 1:
  - `camis$desenv[1] <- desenv_ano1`
- Calcular um valor para cada um dos outras 3 variáveis fixas por cada ano
- Invés de fazer uma por vez, podemos instruir R a fazer os cálculos
- `for` loop - técnica de cálculos repetitidos

12

glimpse(camis)

```
camis <- tibble(desenv = numeric(length = anos),
                publico =  numeric(length = anos),
                custo = numeric(length = anos),
                preco = numeric(length = anos),
                compras = numeric(length = anos),
                renda = numeric(length = anos),
                lucro = numeric(length = anos))
```

13

for Loops

- Técnica clássica de programação
- Permite que você pode fazer um cálculo *n* vezes
  - loop executa *n* vezes
  - *n* é uma quantidade fixa
- Corpo do loop pode ter comandos múltiplos
- Outro tipo de loop: `while()`
  - Loop repete enquanto uma condição fica verdade

14

Exemplo Simples

- 4 variáveis: *a*, *b*, *c*, *d* - todas com 10 números aleatórios
- Queremos determinar a mediana de cada variável

15

Tibble

```
set.seed(42)
df <- tibble(
  a = rnorm(10),
  b = rnorm(10),
  c = rnorm(10),
  d = rnorm(10)
)

glimpse(df)
```

```
## Rows: 10
## Columns: 4
## $ a <dbl> 1.37095845, -0.56469817, 0.36312841, 0.63286260, 0.40426832, ...
## $ b <dbl> 1.3048697, 2.2866454, -1.3888607, -0.2787888, -0.1333213, ...
## $ c <dbl> -0.3066386, -1.7813084, -0.1719174, 1.2146747, 1.8951935,  ...
## $ d <dbl> 0.45545012, 0.70483734, 1.03510352, -0.60892638, 0.50495512, ...
```

16

Método Tradicional para Calcular a Mediana

- Copiar-Colar

```
median(df$a)
## [1] 0.3836984
median(df$b)
## [1] -0.2060551
median(df$c)
## [1] -0.281954
median(df$d)
## [1] -0.2864019
```

17

Pode Trocar para um `for` Loop

- Evitar erros gerados por copiar e colar
- 3 Componentes
  1. Estabelecer lugar para salvar o resultado do loop
  2. Sequência de repetições
  3. Corpo do loop

18

Output

- Um vetor ou tibble onde vai colocar os resultados do corpo do loop
- Alocação do espaço na memoria para os resultados
- Precisa vir antes da especificação do loop
- No caso de `camis`, já existe no tibble
- No caso de `df`, um vetor de números para salvar os resultados

```
(output <- numeric(length = ncol(df)))
## [1] 0 0 0 0
```

19

Sequência

- Contar para loop quantas vezes vai repetir e com quais índices
- Use uma variável índice, normalmente chamada `i`
- Gramática `(i in xxx)` atribuir a `i` todos os valores em `xxx`
- Para `camis`, vamos usar os anos: 1, 2, 3, 4, 5
- Para `df`, usar o número dos casos: 1 até 10 - `(i in 1:anos)`
- Facilitar sequenciamento com `seq_along()`
  - Quer dizer: usar todos as colunas numa sequência

```
seq_along(df)
## [1] 1 2 3 4
```

20

Corpo do Loop

- Os cálculos que quer fazer
- Aquele que está repetido.
- Para `df`, `output[i] <- median(df[i])`
- Primeira vez: `output[1] <- median(df[1])`
- Segunda vez: `output[2] <- median(df[2])`
- Corpo marcado por {}

21

Loop de `df`

```
set.seed(42)
output <- numeric(length = ncol(df))

for(i in seq_along(output)) {
  output[i] <- median(df[[i]])

# dupla [[]] porque median veja df como lista
}

output

 ## [1]  0.3836984 -0.2060551 -0.2819540 -0.2864019
```

22

Alternativas aos Loops

- `apply()`: família das funçoes; sintaxe difícil
- `purrr::map()`: família de funções dentro de Tidyverse
- Quando aprende `purrr::map()`, permite mais flexibilidade e clareza na escrever loops
- Factoide que `for` loops são mais lentos que outras funções - **FAKE NEWS**
  - Faz anos que não é assim

23

`for` loop de `camis`

- Lugar para salvar resultados: `camis` (já existe)
- Sequência: (i in 1:anos) - Não pode usar `seq_along()` – usa colunas; queremos fileiras (anos) - se for `seq_along(camis)`: `i` teria 1, 2, 3, 4, 5, 6, 7
- Corpo: cálculos das variáveis fixas

24

Calcular Variáveis Fixas

- Variáveis `publico`, `custo`, `preco`
- `publico`
  - Quer aumentar o público por 250 em todos os anos depois do primeiro (2:5)
  - Pode multiplicar o `novo_publico` por (i - 1), ou seja 0, 1, 2, 3, 4
  - `camis$publico[i] <- publico_ano1 + (i - 1) * novo_publico`
- `custo`
  - `custo_ano1` ajustado pela inflação
  - `camis$custo[i] <- custo_ano1 * (1 + inflacao)^(i-1)`
- `preco`
  - `custo` multiplicado pela (1 + taxa de marcação) (`markup`)
  - `camis$preco[i] <- camis$custo[i] * (1 + markup)`

25

Exercício 9 – Parte B – Calcular as Variáveis Fixos

Parte B – Código

```
## # A tibble: 5 x 7
##   desenv publico custo preco compras renda lucro
##    <dbl>   <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>
## 1  20000    4000  32    64         0     0     0
## 2      0    4250  33.6  67.2       0     0     0
## 3      0    4500  35.3  70.6       0     0     0
## 4      0    4750  37.0  74.1       0     0     0
## 5      0    5000  38.9  77.8       0     0     0
```

Quantas Compras - Variável Estocástica (prop_compras)

- Qual proporção do público vai comprar a camiseta em cada ano?
- Entre mínimo de 15% e máximo de 40%
- Premissa: qualquer proporção neste intervalo igualmente provável - Distribuição Uniforme - Valores aleatórios uniformes em R: função `runif()` com os argumentos - *n* = número de valores que você quer - *min* = valor mínimo - *max* = valor máximo

```
prop_compras <- runif(5, min = min_prop, max = max_prop)
```

Variáveis que Dependem na prop_compras

- `compras`: `publico` em qualquer anos multiplicado por a `prop_compras`
- `renda`: `compras` multiplicado por `preco`
- `lucro`: `renda` menos `desenv` menos `custo` multiplicado por `compras`

Completar Este Cálculo

26

Exercício 9 – Parte B – Calcular as Variáveis Fixos

27

Parte B - Código

```
camis$desenv[1] <- desenv_ano1
for (i in 1:anos) {
  camis$publico[i] <- publico_ano1 + (i - 1) * novo_publico
  camis$custo[i] <- custo_anoI * (1 + inflacao)^(i-I)
  camis$preco[i] <- camis$custo[i] * (1 + markup)
}
camis
```

```
## # A tibble: 5 x 7
##   desenv publico custo preco compras renda lucro
##    <dbl>   <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>
## 1  20000    4000  32    64          0     0     0
## 2      0    4250  33.6  67.2        0     0     0
## 3      0    4500  35.3  70.6        0     0     0
## 4      0    4750  37.0  74.1        0     0     0
## 5      0    5000  38.9  77.8        0     0     0
```

28

Quantas Compras - Variável Estocástica (`prop_compras`)

- Qual proporção do público vai comprar a camiseta em cada ano?
- Entre mínimo de 15% e máximo de 40%
- Premissa: qualquer proporção neste intervalo igualmente provável
  - Distribuição Uniforme
  - Valores aleatórios uniformes em R: função `runif()` com os argumentos
    - *n* = número de valores que você quer
    - *min* = valor mínimo
    - *max* = valor máximo

```
prop_compras <- runif(5, min = min_prop, max = max_prop)
```

29

Variáveis que Dependem na `prop_compras`

- `compras -- publico` em qualquer anos multiplicado por a `prop_compras`
- `renda -- compras` multiplicado por `preco`
- `lucro -- renda` menos `desenv` menos `custo` multiplicado por `compras`

30

Completar Este Cálculo

```
set.seed(42)

prop_compras <- runif(5, min = min_prop,
                      max = max_prop)

camis <- camis %>%
  mutate(compras = publico * prop_compras,
         renda = compras * preco,
         lucro = renda - desenv - custo *
                 compras)
```

31

Exercício 9 – Parte C – Fazer os Resultados Finais

32

Terminar Uma Simulação - Fazer os Totais Interessantes

- Usar os comandos de Tidyverse
- Colocar os resultados num novo tibble

```
## # A tibble: 1 x 4
##   media_prop tot_compras tot_renda
tot_lucro
##        <dbl>       <dbl>     <dbl>
<dbl>
## 1      0.331        7396.   523635.
241817.
```

33

Poder de Simulação

- Simulação simples só mostra uma estimativa de lucro
- Só vivemos uma sequência das proporções de compra
- Se pudermos, repetir o experimento muitas vezes, o que podemos aprender?
- Aproveitando da Lei de Grande Números e Teorema de Tendência Central - Nossa estimativa das compras e lucro seria mais próximos a tendência verdadeira
- Com computação moderna, pode fazer 1.000 repetições do modelo rapidinho

34

Passos

- Pôr mais um loop em volta do modelo
- Gravar os resultados de cada experimento na base de dados `total_camis`
- Fazer uma análise simples dos valores finais - média do `compras`, `renda` e `lucro`
- Produzir um resultado mais firme que pode ajudar Sustentare tomar uma decisão sobre programa

35

Como Funcionará o Super-Loop

- Vai calcular os resultados de 5 anos 1.000 vezes (`n_sims`)
- Vai adicionar fileiras para `total_camis` com cada iteração
- Vai limpar `camis` para iniciar limpo cada simulação
- Vai calcular os resultados estatísticos utilizando `descr()`

36

Elementos do Super-Loop

- Lugar para salvar resultados: `total_camis` (reiniciar)
- Sequência: (*n* in número de simulações)
  - nova variável índice para não confundir com *i*
- Corpo: cálculos do modelo
- Depois do loop, análise estatística

37

Exercício 10 – Super-Loop

- Construir o novo loop
- Colocar os resultados no tibble `total_camis` - Com as mesmas variáveis que usamos antes
- Variável índice = *n*
- Corpo do loop = função `camis_modelo` que eu criei

38

Função `camis_modelo` – 1

```
camis_modelo <- function() {
  # colocar as variáveis num tibble
  camis <- tibble(desenv = numeric(length = anos),
                  publico =  numeric(length = anos),
                  custo = numeric(length = anos),
                  preco = numeric(length = anos),
                  compras = numeric(length = anos),
                  renda = numeric(length = anos),
                  lucro = numeric(length = anos))
  # Carregar camis com os valores
    camis$desenv[1] <- desenv_ano1
  ## Outras precisam loop
   for (i in 1:anos) {
    camis$publico[i] <- publico_ano1 + (i - 1) * novo_publico
    camis$custo[i] <- custo_ano1 * (1 + inflacao)^(i-1)
    camis$preco[i] <- camis$custo[i] * (1 + markup)
  }
```

39

Função `camis_modelo` – 2

```
camis <- camis %>%
    mutate(compras = publico * prop_compras,
           renda = compras * preco,
           lucro = renda - desenv - custo * compras)
  # Calcular os Totais
  sum_camis <- camis %>%
    summarise(media_prop = mean(prop_compras),
              tot_compras = sum(compras),
              tot_renda = sum(renda),
              tot_lucro = sum(lucro))
  # Salvar os resultados
  return(sum_camis)
}
```

40

Super Loop – Código

```
set.seed(42)
n_sims <- 1000L

for (n in 1:n_sims){
  sum_camis <- camis_modelo()

  # Carregar resultados

  total_camis$sim[n] = n
  total_camis$media_prop[n] = sum_camis$media_prop
  total_camis$tot_compras[n] = sum_camis$tot_compras
  total_camis$tot_renda[n] = sum_camis$tot_renda
  total_camis$tot_lucro[n] = sum_camis$tot_lucro
}
```

41

Super Loop – Resultado

```
> total_camis
# A tibble: 1,000 x 5
     sim media_prop tot_compras tot_renda tot_lucro
   <int>      <dbl>       <dbl>     <dbl>     <dbl>
 1     1      0.331       7396.   523635.   241817.
 2     2      0.288       6490.   462867.   211434.
 3     3      0.291       6529.   462260.   211130.
 4     4      0.304       6751.   475609.   217805.
 5     5      0.303       6766.   477564.   218782.
 6     6      0.305       6899.   493285.   226643.
 7     7      0.281       6230.   436588.   198294.
 8     8      0.278       6291.   449566.   204783.
 9     9      0.263       5955.   426012.   193006.
10    10      0.354       7922.   561073.   260537.
# ... with 990 more rows
```

42

Quanto tempo levou para 1.000 simulações

```
inicio_loop = Sys.time()
# <Código do Loop>
termino_loop = Sys.time()
print(glue::glue("Tempo de execução = ",
                 round(termino_loop - inicio_loop, 2),
                 " segundos"))
```

Tempo de execução = 9.03 segundos

43

Avaliação dos Resultados

```
summarytools::descr(total_camis[, 2:5], stats = c("mean", "sd", "min", "med",
                                                  "max", "iqr", "cv"))
```

```
Descriptive Statistics
total_camis
N: 1000

                media_prop   tot_compras   tot_lucro   tot_renda
-------------- ------------ ------------- ----------- -----------
          Mean         0.28       6205.87   200678.41   441356.82
       Std.Dev         0.03        727.20    26101.38    52202.75
           Min         0.19       4199.98   129022.25   298044.50
        Median         0.28       6222.14   200786.40   441572.80
           Max         0.37       8434.05   280320.51   600641.01
           IQR         0.04        999.80    35549.48    71098.97
            CV         0.12          0.12        0.13        0.12
```

44