Simulating survey studies

Carlo Knotz

Table of contents

1 Introduction

This document shows you how you can simulate many, many different iterations of a fictional survey study – and how the different results are, collectively, normally distributed.

Important: Some of these simulations take a few minutes to run. Be patient.

1.1 Setup

These are the packages you'll need (use `install.packages()` to install anything that is missing):

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   4.0.0     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.2.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
theme_set(theme_classic())
library(bst290)
library(ggpubr)
```

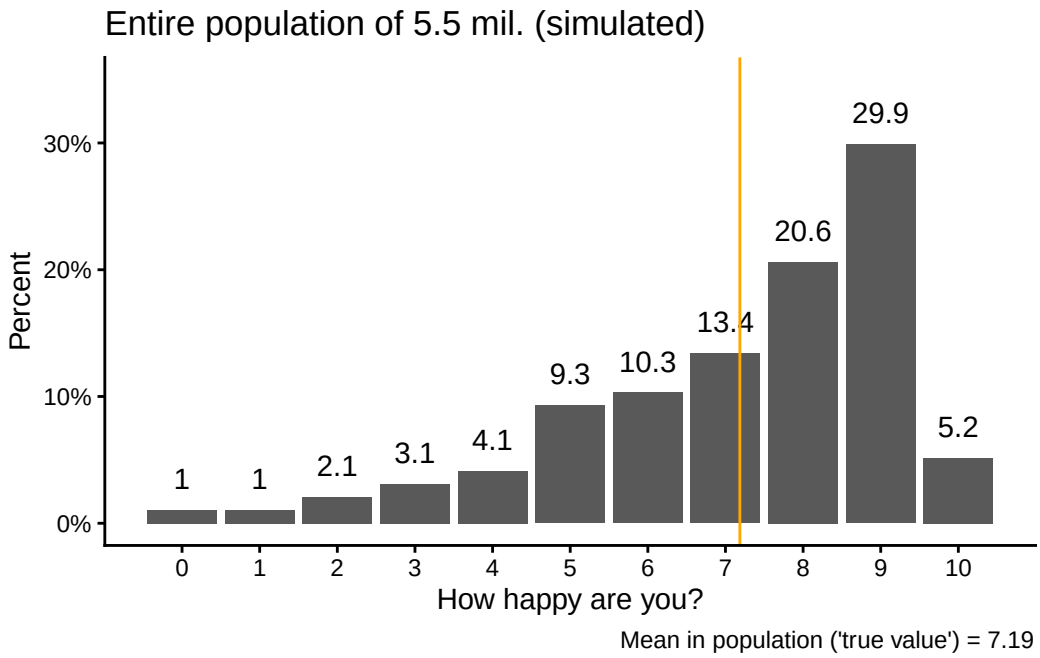
Next, we also record the start time with `Sys.time()` so that we can later check how long the entire simulation took:

```
start <- Sys.time()
```

1.2 Simulating our fictional population

Next, we simulate our fictional population: 5.5 mio. individuals and their levels of happiness on a 0-10 scale:

```
set.seed(42)


pop <- data.frame(idno = seq(1,5.5*10^6,1),
                  happy = sample(seq(0,10,1),
                                 size = 5.5*10^6,
                                 replace = T,
                                 prob = c(0.01,.01,.02,.03,.04,.09,.10,.13,.20,.29,0.05)))

popmean <- mean(pop$happy)

pop |>
  group_by(happy) |>
  summarize(obs = n()) |>
  mutate(share = obs/sum(obs)) |>
  ggplot(aes(x = happy, y = share)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = round(100*share, digits = 1)), vjust = -1) +
  geom_vline(xintercept = popmean, color = "orange") +
  scale_x_continuous(breaks = seq(0,10,1)) +
  scale_y_continuous(labels = scales::percent,
                     limits = c(0,.35)) +
  labs(y = "Percent", x = "How happy are you?",
       title = "Entire population of 5.5 mil. (simulated)",
       caption = paste0("Mean in population ('true value') = ",round(popmean, digits = 2))) +

popvis
```

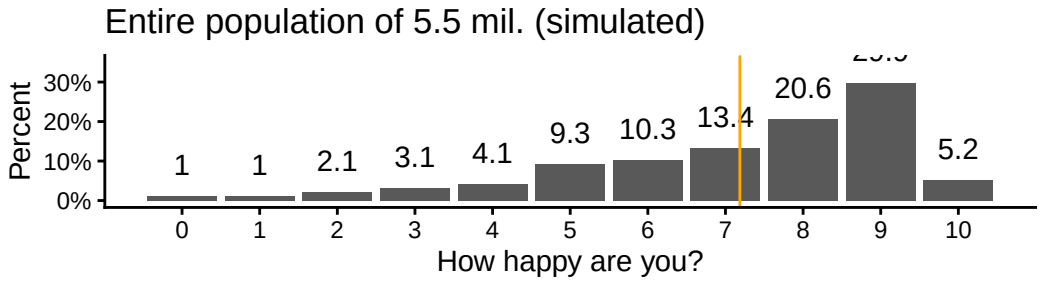


Note that we set a "seed" number so that we can always exactly reproduce the same results later.

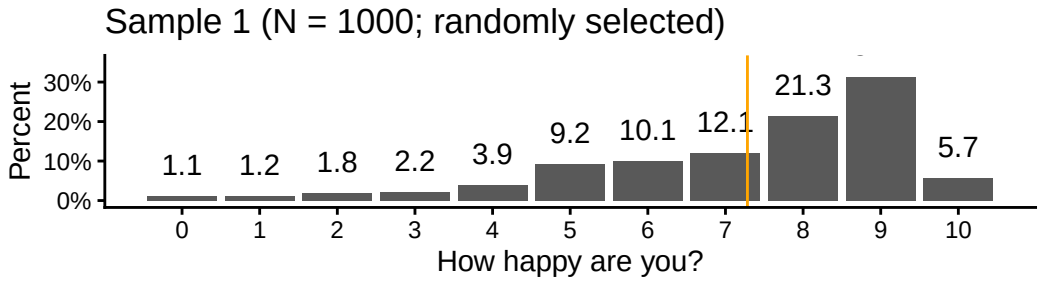
1.3 Taking a first sample

We pretend that we are doing a social science survey by taking a random sample from the population with `slice_sample()` and calculate the sample mean:

```
set.seed(42)
sam1 <- pop |>
  slice_sample(n = 1000)

sam1mean <- mean(sam1$happy)

sam1 |>
  group_by(happy) |>
  summarize(obs = n()) |>
  mutate(share = obs/sum(obs)) |>
  ggplot(aes(x = happy, y = share)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = round(100*share, digits = 1)), vjust = -1) +
  geom_vline(xintercept = sam1mean, color = "orange") +
  scale_x_continuous(breaks = seq(0,10,1)) +
  scale_y_continuous(labels = scales::percent,
                     limits = c(0,.35)) +
  labs(y = "Percent", x = "How happy are you?",
       title = "Sample 1 (N = 1000; randomly selected)",
       caption = paste0("Mean in sample ('estimate') = ",round(sam1mean, digits = 2))) -> sa

ggpubr::ggarrange(popvis,sam1vis, nrow = 2)
```



Mean in population ('true value') = 7.19

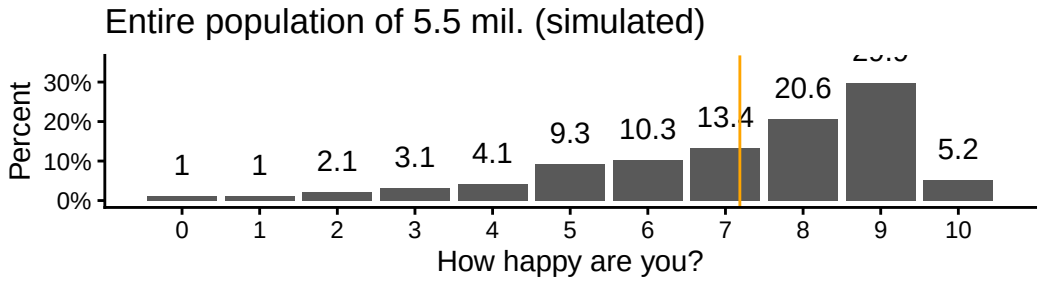


Mean in sample ('estimate') = 7.28

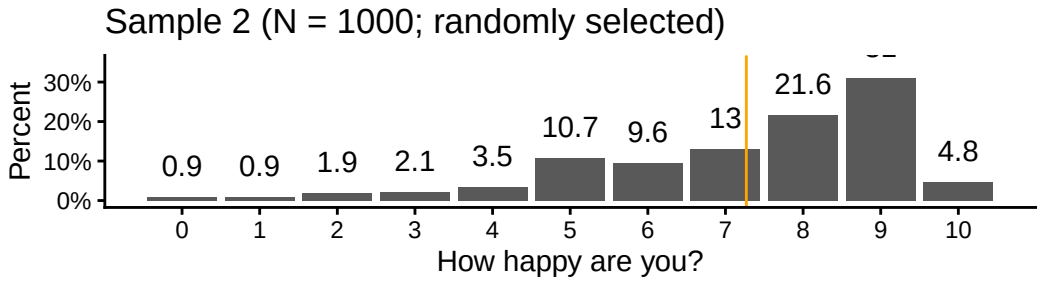
1.4 Taking a second sample

Then we take a second sample:

```
set.seed(17)
sam2 <- pop |>
  slice_sample(n = 1000)

sam2mean <- mean(sam2$happy)

sam2 |>
  group_by(happy) |>
  summarize(obs = n()) |>
  mutate(share = obs/sum(obs)) |>
  ggplot(aes(x = happy, y = share)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = round(100*share, digits = 1)), vjust = -1) +
  geom_vline(xintercept = sam2mean, color = "orange") +
  scale_x_continuous(breaks = seq(0,10,1)) +
  scale_y_continuous(labels = scales::percent,
                     limits = c(0,.35)) +
  labs(y = "Percent", x = "How happy are you?",
       title = "Sample 2 (N = 1000; randomly selected)",
       caption = paste0("Mean in sample ('estimate') = ",round(sam2mean, digits = 2))) -> sa

ggpubr::ggarrange(popvis,sam2vis, nrow = 2)
```



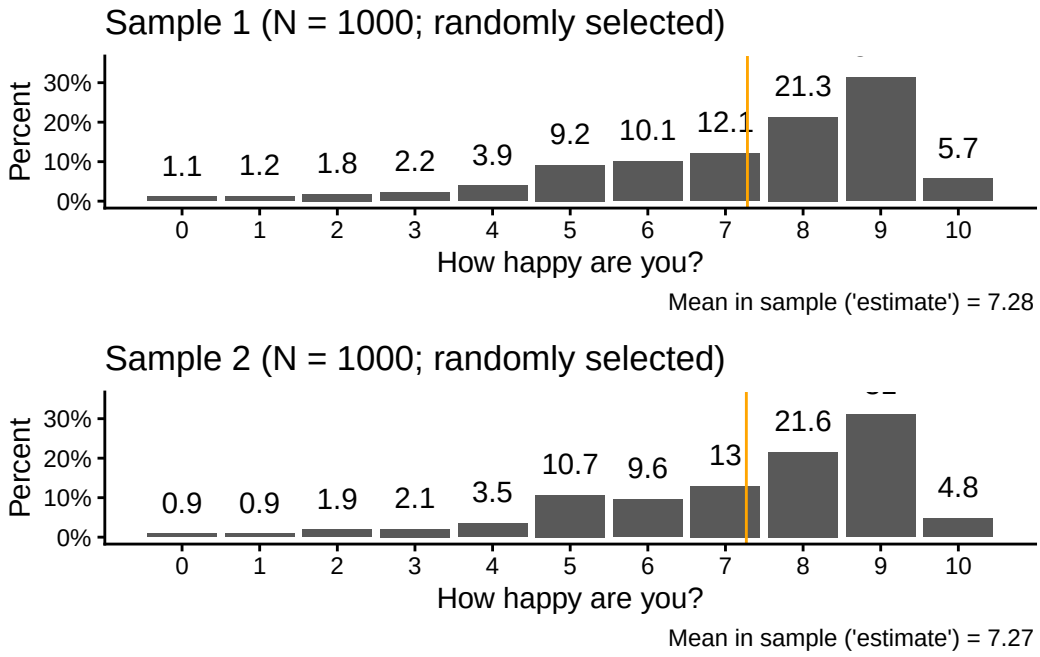
Mean in population ('true value') = 7.19



Mean in sample ('estimate') = 7.27

1.5 Comparing the samples

```
ggpubr::ggarrange(sam1vis,sam2vis, nrow = 2)
```

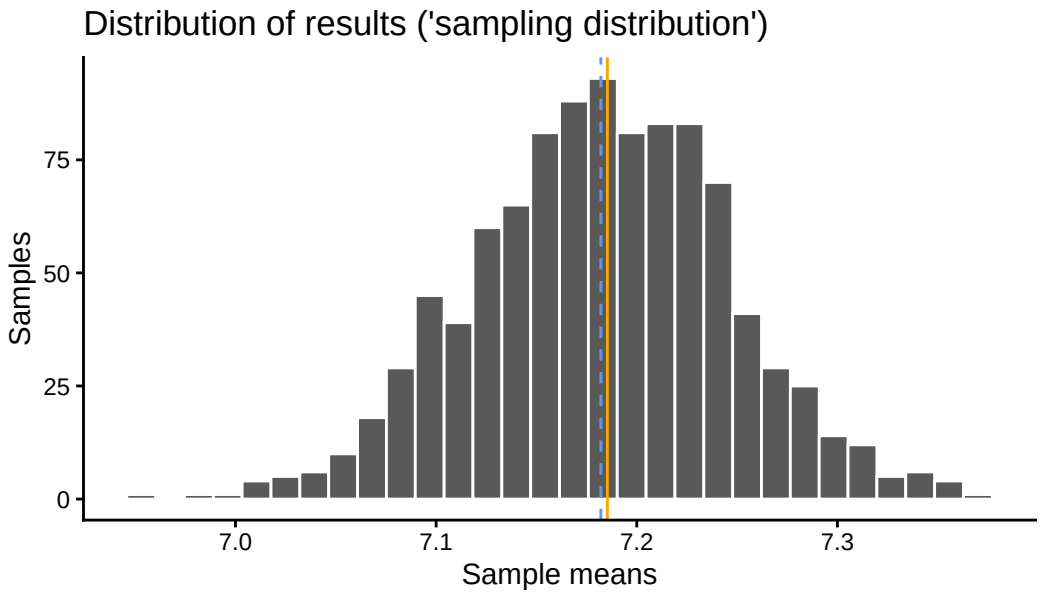


1.6 Scaling it up to 1000 samples

We collect 998 new samples so that we get to 1'000 in total:

```
sampdist <- data.frame(sample = seq(1,1000,1),
                       result = c(c(sam1mean,sam2mean),rep(NA,998)))

for(k in 3:1000) {

  loopsam <- pop |>
    slice_sample(n = 1000)

  sampdist[k,2] <- mean(loopsam$happy)
}

sampdistmean <- mean(sampdist$result)

sampdist |>
  ggplot(aes(x = result)) +
  geom_histogram(color = "white") +
  geom_vline(xintercept = sampdistmean, color = "cornflowerblue", linetype = "dashed") +
  geom_vline(xintercept = popmean, color = "orange") +
  labs(x = "Sample means", y = "Samples",
       title = "Distribution of results ('sampling distribution')",
       caption = paste0("Population mean ('true value'; orange) = ",round(popmean, digits = 2
                        "\n Mean over all results ('mean of means'; blue, dashed) = ",round(s
```

```
`stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```

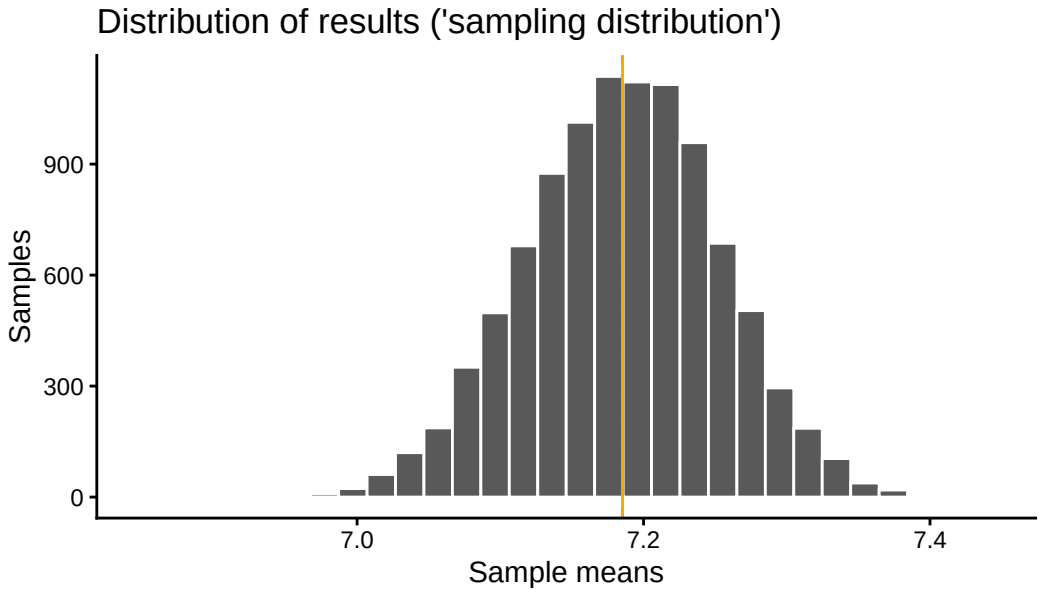

Distribution of results ('sampling distribution')
Samples
75
50
25
0
7.0
7.1
7.2
7.3
Sample means
Population mean ('true value'; orange) = 7.19
Mean over all results ('mean of means'; blue, dashed) = 7.18

1.7 Then we do 10'000 surveys

We're scaling it up once more to 10'000 surveys overall:

```
set.seed(42)
sampdist <- data.frame(sample = seq(1,10000,1),
                       result = c(c(sam1mean,sam2mean),rep(NA,9998)))

for(k in 3:10000) {

  loopsam <- pop |>
    slice_sample(n = 1000)

  sampdist[k,2] <- mean(loopsam$happy)
}

sampdistmean <- mean(sampdist$result)

sampdist |>
  ggplot(aes(x = result)) +
  geom_histogram(color = "white") +
  geom_vline(xintercept = sampdistmean, color = "cornflowerblue", linetype = "dashed") +
  geom_vline(xintercept = popmean, color = "orange") +
  labs(x = "Sample means", y = "Samples",
       title = "Distribution of results ('sampling distribution')",
       caption = paste0("Population mean ('true value'; orange) = ",round(popmean, digits = 2
                        "\n Mean over all results ('mean of means'; blue, dashed) = ",round(s
```

```
`stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



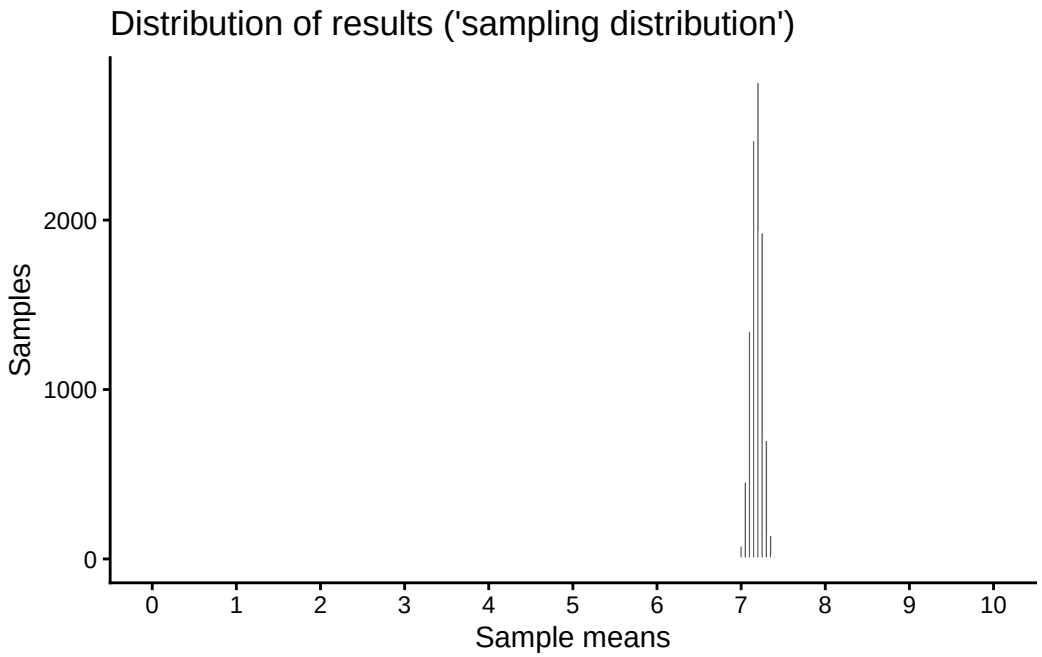
Population mean ('true value'; orange) = 7.19
Mean over all results ('mean of means'; blue, dashed) = 7.19

1.8 The results in context

To see how much "wiggling" there really is due to random sampling variation:

```
sampdist |>
  ggplot(aes(x = result)) +
  geom_histogram(color = "white", binwidth = 0.05) +
  scale_x_continuous(limits = c(0,10), breaks = seq(0,10,1)) +
  labs(x = "Sample means", y = "Samples",
       title = "Distribution of results ('sampling distribution')")
```

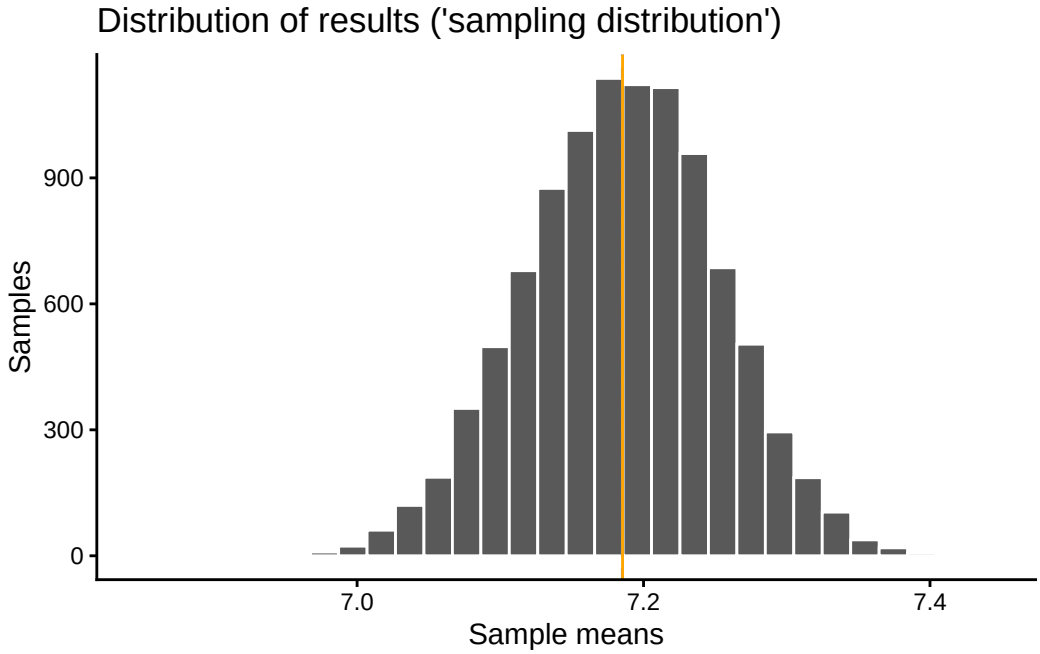
```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```



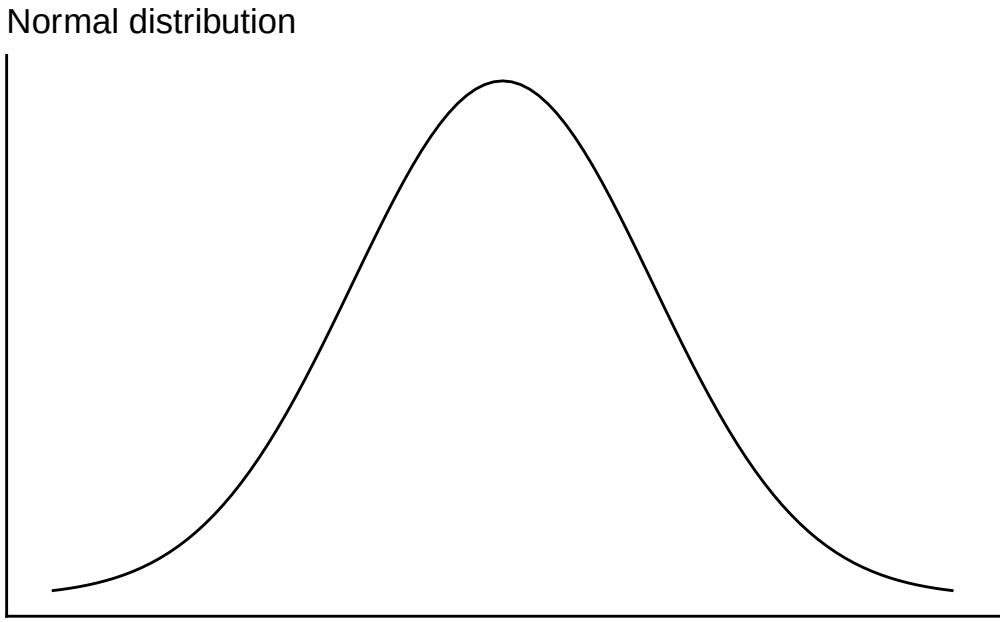
1.9 Comparing our results distribution ("sampling distribution") to the Normal distribution

```
sampdist |>
  ggplot(aes(x = result)) +
  geom_histogram(color = "white") +
  geom_vline(xintercept = sampdistmean, color = "cornflowerblue", linetype = "dashed") +
  geom_vline(xintercept = popmean, color = "orange") +
  labs(x = "Sample means", y = "Samples",
       title = "Distribution of results ('sampling distribution')") -> samdist10k
samdist10k
```

```
`stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```

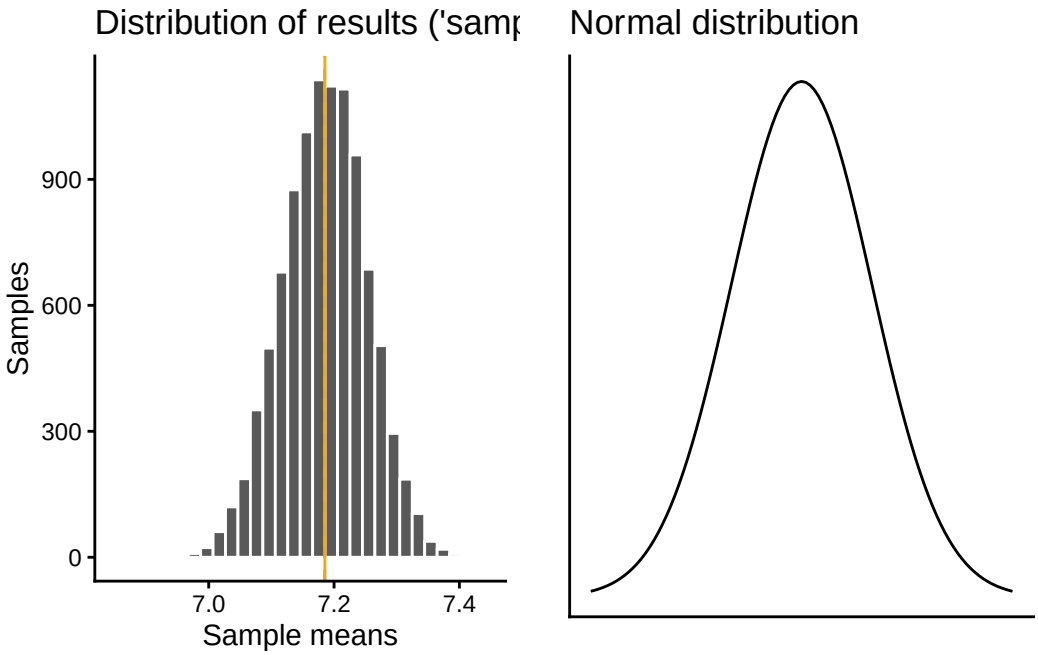



```
p1 <- ggplot(data = data.frame(x = c(-3, 3)), aes(x)) +
  stat_function(fun = dnorm, n = 101, args = list(mean = 0, sd = 1)) + ylab("") +
  scale_y_continuous(breaks = NULL) +
  scale_x_continuous(breaks = NULL) +
  labs(x = "", title = "Normal distribution")
p1
```




```
ggpubr::ggarrange(samdist10k,p1, ncol = 2)
```

```
`stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```

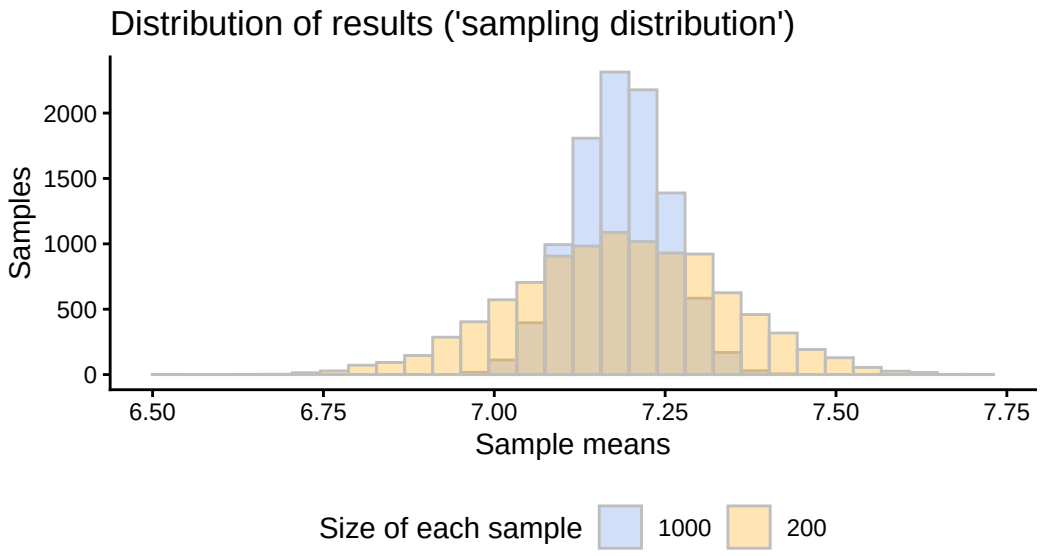


1.10 Now 10'000 new samples, but with a smaller sample size (200 instead of 1'000)

We reduce the sample size to 200 and take 10'000 new samples:

```
sampdist_200 <- data.frame(sample = seq(1,10000,1),
                           result = rep(NA,10000))

set.seed(42)
for(k in 1:10000) {

  loopsam <- pop |>
    slice_sample(n = 200)

  sampdist_200[k,2] <- mean(loopsam$happy)
}

sampdist_200mean <- mean(sampdist_200$result)

sampdist_SD <- sd(sampdist$result)
sampdist_200_SD <- sd(sampdist_200$result)

sampdist |>
  left_join(sampdist_200, by = "sample",
            suffix = c("1000","200")) |>
  pivot_longer(cols = -1,
               names_to = "size",
               values_to = "meanval") |>
  mutate(size = gsub("result","",size)) |>
  ggplot(aes(x = meanval,fill = size)) +
  geom_histogram(alpha=0.3, position="identity",
                 color = "grey") +
  scale_fill_manual(values = c("cornflowerblue","orange")) +
  labs(x = "Sample means", y = "Samples",
       fill = "Size of each sample",
       title = "Distribution of results ('sampling distribution')",
       caption = paste0("Population mean ('true value') = ",round(popmean, digits = 2),
                        "\n Mean over all results ('mean of means') for samples of 1000 = ",
                        round(sampdistmean, digits = 2)," (SD = ",round(sampdist_SD, digits =
                        "\n Mean over all results ('mean of means') for samples of 200 = ",
                        round(sampdist_200mean, digits = 2)," (SD = ",round(sampdist_200_SD,
  theme(legend.position = "bottom")
```

```
`stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



Population mean ('true value') = 7.19
Mean over all results ('mean of means') for samples of 1000 = 7.19 (SD = 0.07)
Mean over all results ('mean of means') for samples of 200 = 7.18 (SD = 0.15)

###… and taking time

How long did all of this take (on my rickety old Intel-based MacBook Pro):

```
# Taking time
end <- Sys.time()
duration <- end - start
duration
```

```
Time difference of 2.591034 mins
```