PSYC278: Analysis of Behavioural Data

Lab 06 - Sampling Distributions & Z-Test

Raymond MacNeil & Ke Zhang

March 17th, 2023

Information on Final Lab Assignment

- Take home
- **Due: Friday, April 10th, 11:59 PM (PDT)**
- Part 1: an array of different questions encompassing the broad range of content we covered in labs
- Part 2: will focus on a larger dataset and will likely involve an ANOVA analysis
- Assignment for week of March 31st substituted with in-lab activity to be completed on Canvas

Goals of Lab

- By the end of this lab, you will learn how you can use R to:
  1. Generate and plot sampling distributions from a defined population
  2. Test hypotheses using the normal deviate (z) test
  3. Compute the power of the z-test given a real effect

Required Libraries

```r
library(psych)
library(pracma)
library(ggplot2)
library(latex2exp)
```

Review: Null Hypothesis Significance Testing

	State of Reality	
Decision	**H_0 is true**	**H_0 is false**
Reject H_0	Type I Error (α)	Power ($1-\beta$)
Retain H_0	Correct Retention ($1-\alpha$)	Type II Error (β)

Sampling Distributions & Null-Hypothesis Population

A statistic's **sampling distribution** provides all possible values that the statistic can take on, as well as the probability of obtaining each value under the assumption that it resulted from chance alone.

As stated by Pagano (2013), "*the* **null-hypothesis population** *is an actual or theoretical set of population scores that would result if the experiment were done on the entire population and the independent variable had no effect* (p. 300; emphasis original)."

Generating Hypothetical Populations and Samples: Binomial

The `rbinom()` function allows you to generate random samples of scores with a binary outcome. This could be heads/tails, win/loss, etc.

The arguments taken are `n` (e.g., number of coins per trial), `size` (e.g., number of trials or tosses), and `prob`, which is the probability associated with a ‘success’ – shorthand for, “outcome x when an outcome of either x or y is possible.”

Generating Hypothetical Populations and Samples: Binomial

By way of example, let's simulate the results of tossing four coins four times. The coins are all fair (`prob = 0.50`). We will consider a head to be a success.

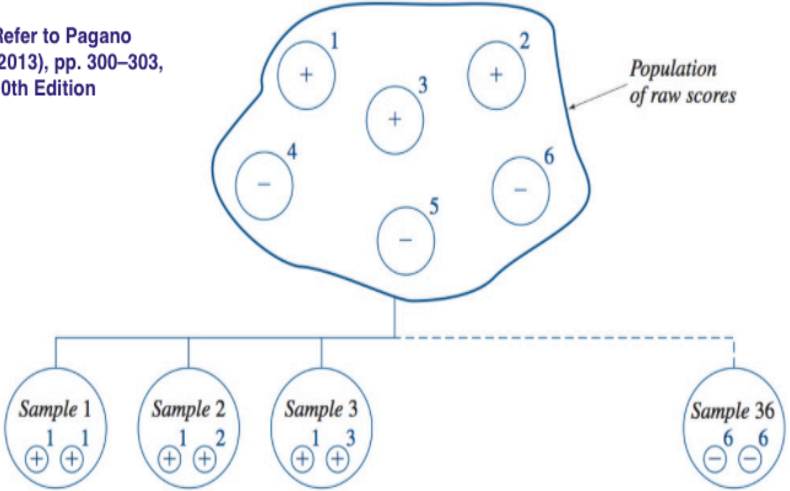
```
set.seed(333) # for reproduceability
rbinom(n = 4, size = 4, prob = .50)
```

```
## [1] 2 1 4 2
```

In this example, the output of the code can be translated to, "of the four coin tosses, we observed two heads for coin 1, one head for coin 2, four heads for coin 3, and two heads for coin 4."

Sampling Distributions: Pagano's Binomial Example

Refer to Pagano (2013), pp. 300–303, 10th Edition



Sampling Distributions: Pagano's Binomial Example

In several lines of code, we can programatically replicate Pagano's empirical demonstration of generating a sign-test H_0 sampling distribution (with $N = 2$) given a population comprised of three pluses (IDs 1–3) and three minuses (IDs 4–6).

```
ID <- c(1,2,3,4,5,6) # Identifier for each score
# 1 = plus, 0 = minus, for our purposes
scores <- c(1,1,1,0,0,0)

# Create N lists of the IDs to prepare for
# getting permuations, we use the rep() function
# introduced in lab one.
print(ID <- rep(list(ID), 2)) # NOTE. 2, because N = 2
```

```
## [[1]]
## [1] 1 2 3 4 5 6
##
## [[2]]
## [1] 1 2 3 4 5 6
```

Sampling Distributions: Pagano's Binomial Example

```r
# Do the same for the scores assocaited with
# the element IDs
print(scores <- rep(list(scores), 2))
```

```
## [[1]]
## [1] 1 1 1 0 0 0
##
## [[2]]
## [1] 1 1 1 0 0 0
```

Sampling Distributions: Pagano's Binomial Example

The `expand.grid()` function can be used to generate permutations (with replacement) of size N when the desired object to permute is a list of size N.

```
head(expand.grid(ID), 10)
```

```
##    Var1 Var2
## 1     1    1
## 2     2    1
## 3     3    1
## 4     4    1
## 5     5    1
## 6     6    1
## 7     1    2
## 8     2    2
## 9     3    2
## 10    4    2
```

Sampling Distributions: Pagano's Binomial Example

For additional information on the `expand.grid()` function for getting all permutations of size n, refer to this post at R-bloggers.

```
# expand.grid, permutations with replacement
samp.dist <- data.frame(expand.grid(ID),
               expand.grid(scores))
#
# Swap columns one and two for visualization purposes
samp.dist <- samp.dist[,c(2,1,4,3)]
colnames(samp.dist) <- c('IDn1', 'IDn2', 'IDn1.Score', 'IDn2.Score')
```

Sampling Distributions: Pagano's Binomial Example

So, what do we have so far?

```
head(samp.dist)
```

```
##   IDn1 IDn2 IDn1.Score IDn2.Score
## 1    1    1          1          1
## 2    1    2          1          1
## 3    1    3          1          1
## 4    1    4          1          0
## 5    1    5          1          0
## 6    1    6          1          0
```

```
tail(samp.dist)
```

```
##    IDn1 IDn2 IDn1.Score IDn2.Score
## 31    6    1          0          1
## 32    6    2          0          1
## 33    6    3          0          1
## 34    6    4          0          0
## 35    6    5          0          0
## 36    6    6          0          0
```

Sampling Distributions: Pagano's Binomial Example

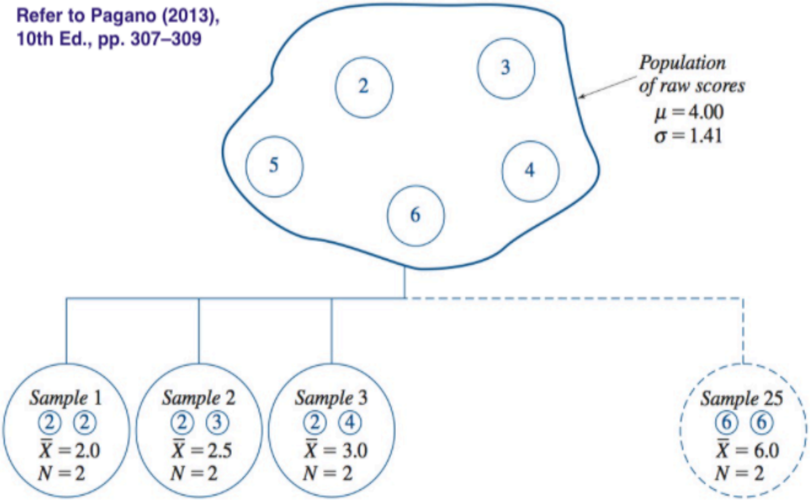
Now we must create a variable representing the number of 'pluses' in each sample – the statistic in Pagano's example. The `rowSums()` function sums across rows in a matrix or dataframe. Note that here I am indexing into columns three and four because we don't care about including the IDs in the result.

```
samp.dist$num.pluses <- rowSums(samp.dist[,c(3,4)])
# Number of 0, 1, and 2 pluses
t <- table(samp.dist$num.pluses)
# NOTE. Output is proportion of 0,1, and 2 pluses
rbind(t, t / sum(t))
```

```
##      0    1    2
## t 9.00 18.0 9.00
##   0.25  0.5 0.25
```

Sampling Distribution of the Mean: Pagano Example

Refer to Pagano (2013), 10th Ed., pp. 307–309



Sampling Distribution of the Mean: Pagano Example

Note the use of the `rowMeans()` function, which behaves like `rowSums()` though instead of returning the sum(s) of the dataframe's rows (or specified subset), it returns the mean(s).

```
scores <- 2:6
# expand.grid, permutations with replacement
samp.dist.m <- data.frame(expand.grid(rep(list(scores), 2)))
# Swap columns one and two for visualization purposes
samp.dist.m <- samp.dist.m[,c(2,1)]
colnames(samp.dist.m) <- c('score.1', 'score.2')
samp.dist.m$mean <- rowMeans(samp.dist.m)
print(t <- table(samp.dist.m$mean))
```

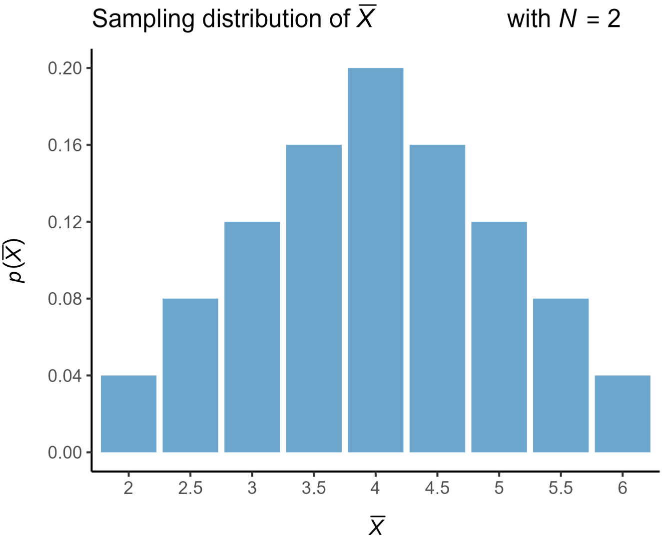
```
##
##   2 2.5   3 3.5   4 4.5   5 5.5   6
##   1   2   3   4   5   4   3   2   1
```

Sampling Distribution of the Mean: Pagano Example

```
require(latex2exp) # You are NOT expected to know LaTeX
pct <- as.numeric(t) / sum(t)
xlab <- names(t)
df <- data.frame(xlab, pct)
p <- ggplot(data = df, aes(x = xlab, y = pct)) +
  geom_bar(stat = "identity", fill = 'skyblue3') +
  labs(x = TeX("$\\bar{\\textit{X}$"), # This weird stuff is LaTeX
       y = TeX("$\\textit{p}(\\bar{\\textit{X}})$"),
       title = TeX("Sampling distribution of $\\bar{\\textit{X}}$
                   with $\\textit{N} = 2$")) +
  theme_classic() +
  scale_y_continuous(breaks = seq(0, 0.24, .04)) +
  theme(axis.title.x = element_text(margin = unit(c(3.5,0,0,0),
                                           "mm"), size = 11),
        axis.title.y = element_text(margin = unit(c(0,3.5,0,0),
                                           "mm"), size = 11),
        axis.text = element_text(size = 9.5))
p


# ggsave(filename = "xdist.png", width = 5, height = 4,
#        units = "in", dpi = 400)
```

Sampling Distribution of the Mean: Pagano Example



Hypothesis Testing with the Z-Test: Formulae

$$\sigma_{\bar{X}} = \frac{\sigma}{\sqrt{N}}$$

$$z_{\text{crit}} = \frac{\bar{X}_{\text{crit}} - \mu_{\text{null}}}{\sigma_{\bar{X}}}$$

$$\bar{X}_{\text{crit}} = \mu_{\text{null}} + \sigma_{\bar{X}}(z_{\text{crit}})$$

$$z_{\text{obt}} = \frac{\bar{X}_{\text{obt}} - \mu}{\sigma_{\bar{X}}}$$

Hypothesis Testing with the Z-Test: Formulae

Formula for determining required N given a specified level of power $(1-\beta)$.

$$\text{Note that,} \quad N_{\text{need}} = \left[\frac{\sigma\left(|z_{\text{crit}}| + |z_{\text{obt.need}}|\right)}{\mu_{\text{real}} - \mu_{\text{null}}}\right]^2$$

Hypothesis Testing with the Z-Test

Pagano, practice problem 12.1 (p. 315): A university president believes that, over the past few years, the average age of students attending his university has changed. To test this hypothesis, an experiment is conducted in which the age of 150 students who have been randomly sampled from the student body is measured. The mean age is 23.5 years. A complete census taken at the university a few years before the experiment showed a mean age of 22.4 years, with a standard deviation of 7.6.

Hypothesis Testing with the Z-Test

Skipping Parts A & B... **Part C:** Using $\alpha = 0.05_{\text{2-tail}}$, what is the conclusion?

```
N <- 150; Mu <- 22.4; Sigma <- 7.6
Xbar.Obt <- 23.5; alpha <- 0.05
Xbar.Obt.SE <- Sigma / sqrt(N)
Zobt <- (Xbar.Obt - Mu) / Xbar.Obt.SE
Zcrit <- qnorm(alpha/2)
# What does the output from the below line tell us?
# Why MIGHT it be necessary to compare the absolute values?
abs(Zobt) >= abs(Zcrit)
```

```
## [1] FALSE
```

```
# Can we just compare this result to alpha as is?
pnorm(Zobt, lower.tail = F)
```

```
## [1] 0.03814278
```

Hypothesis Testing with the Z-Test

Going right to the money...

```
round(p <- 2*pnorm(q = Xbar.Obt, mean = Mu,
                   sd = Xbar.Obt.SE, lower.tail = FALSE), digits = 3)
```

```
## [1] 0.076
```

```
# Can be useful to check your computations by working backwards...
# Verify it is correct by comparing quantile to Zobt
qnorm(p/2, lower.tail = FALSE)
```

```
## [1] 1.772657
```

```
Zobt
```

```
## [1] 1.772657
```

Conclusion: Retain H_0. The data indicates that the average age of students attending fictional university x has not changed over the last few years.

Sidebar: Notes on using pnorm() for calculating p-values

- If required to calculate p with `pnorm()`, pay careful attention to your input (**T** or **F**) into the `lower.tail` argument and whether you are performing a one or two-tailed test.
- If `lower.tail = T` (default), you need to subtract the `pnorm()` output *from* one to get the proper value for comparison with alpha
- Further, if it's a two-tailed test, you need to either: (a) compare the output from above to $\alpha/2$; or (b) multiply the output by two before comparing with $alpha$.

Hypothesis Testing with the Z-Test

Solution to Pagano, Chapter 12, Question 20: **Part A.** A set of sample scores from an experiment has an $N = 30$ and an $\bar{X}_{\text{obt}} = 19$. Can we reject the null hypothesis that the sample is a random sample from a normal population with $\mu = 22$ and $\sigma = 8$? Use $\alpha = 0.01_{\text{1-tail}}$ and assume the sample mean is in the correct direction.

```
n <- 30
xobt <- 19
mu <- 22
sigma <- 8
sem <- sigma / sqrt(n)
print(zcrit <- qnorm(p = .01, lower.tail = TRUE))
## [1] -2.326348
print(zobt <- (xobt - mu) / sem)
## [1] -2.05396
print(zobt <= zcrit)
## [1] FALSE
```

Hypothesis Testing with the Z-Test: Power

Part B. What is the power of the experiment to detect a real effect such that $\mu_{\text{real}} = 20$?

```
mu_real <- 20
print(xcrit <- mu + zcrit*sem)
## [1] 18.60215
# Compute result by first calculating z-score
zobt_real <- (xcrit - mu_real) / sem
# pnorm defaults: mean = 0, sd  = 1, lower.tail = TRUE
round(pnorm(zobt_real, lower.tail = T), digits = 4)
## [1] 0.1693
# Compute result directly with pnorm()
round(pnorm(q = xcrit, mean = mu_real, sd = sem,
      lower.tail = T), digits = 4)
## [1] 0.1693
```

N.B. Pagano gives the answer as 0.1685, but this is assuming that one is computing the answer using the textbook's z-table, which requires rounding z_{obt} to two decimal places, i.e. 0.96.

Hypothesis Testing with the Z-Test: Power

Part C. What is the power of the experiment to detect $\mu_{\text{real}} = 20$ if N is increased to 100?

```
n.new <- 100
sem.new <- sigma / sqrt(n.new)
xcrit.new <- mu + zcrit*sem.new
zobt_real <- (xcrit.new - mu_real) / sem.new
round(pnorm(zobt_real), digits = 4)
```

```
## [1] 0.5689
```

Hypothesis Testing with the Z-Test: Power

Part D. What does N have to equal to achieve a power of 0.8000 to detect $\mu_{\text{real}} = 20$?

```
# Compute z-obtained that would yield .80 power
zcrit <- qnorm(p = 0.99, lower.tail = F)
zobt.need <- qnorm(p = 0.80, lower.tail = F)
# Plug in values to formula
numer <- sigma * ( abs(zcrit) +  abs(zobt.need) )
denom <- mu_real - mu
N <- (numer / denom)^2
# Round with digits = 0
print(round(N, digits = 0))
```

```
## [1] 161
```

Generating Normally Distributed Random Samples

- The `rnorm()` function can be used to generate random samples from a theoretical population of normally distributed scores.
- It takes arguments `n` (the number of scores to be sampled), `mean` (the *population* mean), and `sd` (the *population* standard deviation).
- Type `?rnorm` into the console for more information on this function.

- The `sample()` function can be used to simulate the process of randomly sampling from another sample or a **defined** population.
- It takes arguments `x` (the set of scores you want to sample from, i.e. your 'population') and `n` (the size of the sample you wish to generate).

Example: Simulation of Type-I Errors

Note: I encourage you to work through this example, and attempt to understand what is happening, but you are not responsible for knowing how to run statistical simulations in this course.
Setup...

```
NSamples <- 1000 # Number random samples to draw
Mu <- 100 # Population Mean
Sigma <- 15 # Population SD
N <- 25 # Size of samples
SEM <- Sigma / sqrt(N) # Standard error of the mean
alpha <- 0.05 # Type-I Error Rate
Zcrit <- qnorm(alpha/2) # Z-critical, two-tailed
# Pre-allocate memory for speed
type1.Z <- matrix(data = NaN, nrow = NSamples, ncol = 2,
                  dimnames = list(NULL, c("Z", "p<=.05")))
```

Example: Simulation of Type-I Errors

Run the simulation and get results...

```r
set.seed(123) # Set the seed of the random number generator
for (ii in 1:NSamples) {
  samp.ii <- rnorm(n = N, mean = Mu, sd = Sigma)
  Xobt <- mean(samp.ii)
  Zobt <- (Xobt - Mu) / SEM
  type1.Z[ii,"Z"] <- Zobt
  type1.Z[ii,"p<=.05"] <- abs(Zobt) >= abs(Zcrit)
}
print(NumTypeIError <- sum(type1.Z[,"p<=.05"]))
```

```
## [1] 45
```

```r
# Should approximate alpha
print(RateTypeIError <- NumTypeIError / NSamples)
```

```
## [1] 0.045
```

References

Pagano, R. R. (2013). *Understanding Statistics in the Behavioral Sciences* (10th ed.). Belmont, CA: Cengage Learning.