Inference and Modeling Intro

Spring R ’24

Dominic Bordelon, Research Data Librarian, ULS

Agenda

1. Inference and hypothesis testing
    1. Distribution functions
    2. Hypothesis tests
2. Modeling (simple linear regression)

About the trainer

Dominic Bordelon, Research Data Librarian
University Library System, University of Pittsburgh
dbordelon@pitt.edu

Services for the Pitt community:

- Consultations
- Training (on-request and via public workshops)
- Talks (on-request and publicly)
- Research collaboration

Support areas and interests:

- Computer programming fundamentals, esp. for data processing and analysis
- Open Science and Data Sharing
- Data stewardship/curation
- Research methods; science and technology studies

Today's packages

`stats` (part of base R, automatically attaches)

`infer`: tidy inference

`tidymodels`, particularly:

- `broom`: tidy model representation
- `parsnip`: standardized modeling interface

```r
install.packages(c("tidymodels", "infer"))
```

…and using penguins examples

```r
install.packages("palmerpenguins")
```

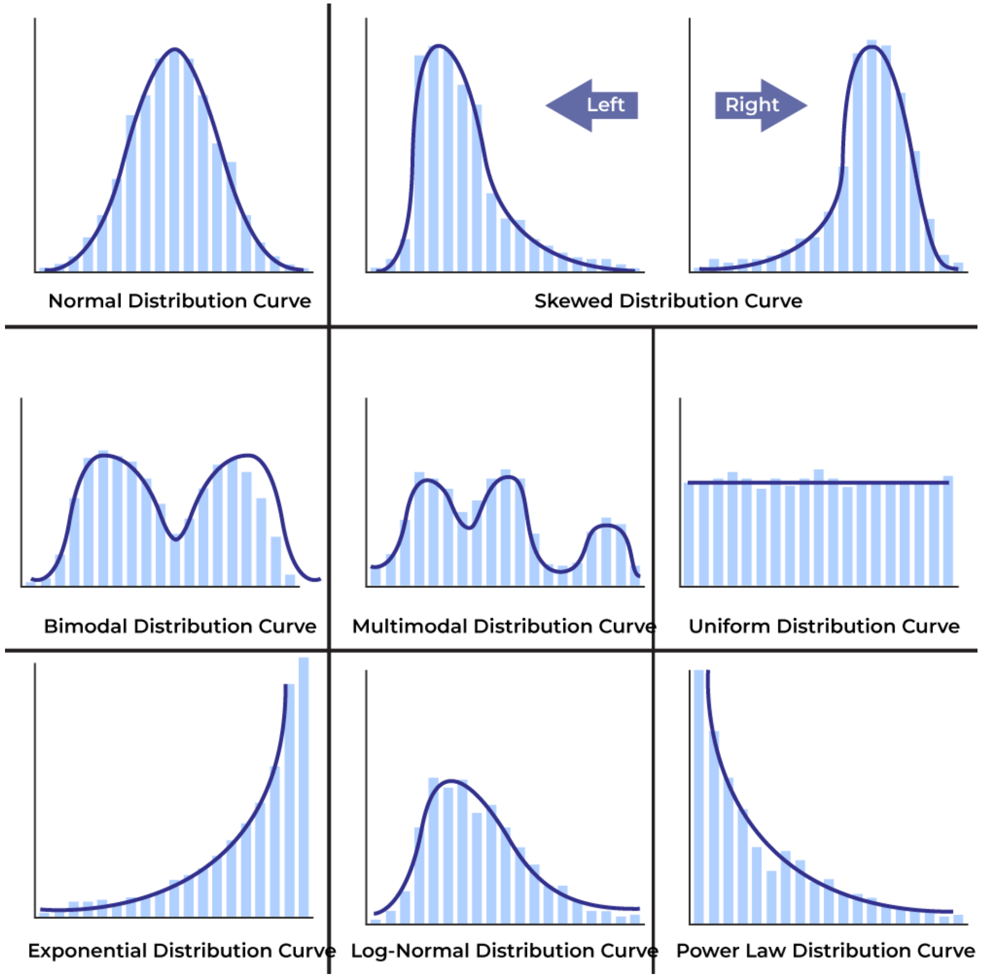
```r
library(palmerpenguins)

# load palmerpenguins' data into your environment:
data(penguins)
names(penguins)
```

```
[1] "species"           "island"            "bill_length_mm"
[4] "bill_depth_mm"     "flipper_length_mm" "body_mass_g"
[7] "sex"               "year"
```

Inference and hypothesis testing

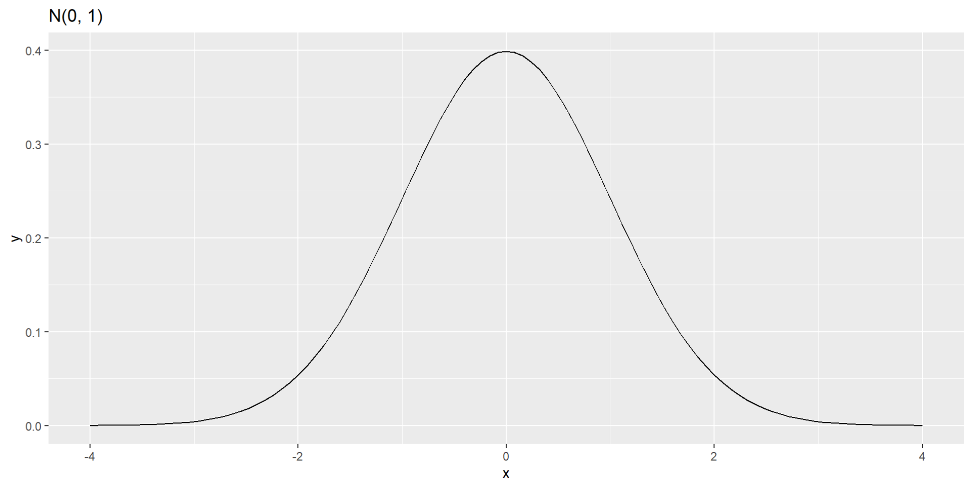
Frequency distributions

- In statistics we want to estimate parameters of the *population* using observed *samples*
- In frequentist or parametric statistics, we assume that the population distribution can be approximated with standard forms such as the Gaussian (normal) distribution



A variety of distributions. Image source: Geeks for Geeks

Distribution functions



Probability distributions in R are typically described by four functions. The examples here are for the Normal distribution.

All have `mean` and `sd` arguments.

- `dnorm(x)` for density (height of curve) at x
- `pnorm(q)` for getting a probability p value at some q, i.e., $P(X \leq q)$
- `qnorm(p)` for finding the quantile of some p
- `rnorm(n)` for generating a random sample of size n from $N(\mu, \sigma)$

Distributions

Commonly used distributions in the `stats` package

Distribution	Name in functions	Example function	Applications
Normal (Gaussian)	`norm`	`pnorm()`	Numerous (often assumed)
Student's t	`t`	`pt()`	Estimating parameters without knowing σ; t-tests
Binomial (Bernoulli)	`binom`	`pbinom()`	Number of successes in `size` trials; logistic regression
Chi-squared (χ^2)	`chisq`	`pchisq()`	Chi-squared tests (goodness of fit in a 2-way table)
F	`f`	`pf()`	ANOVA; F-tests (model goodness-of-fit)

Hypothesis tests

Hypothesis tests in base R tend to follow this format:

- a function call
- accepts vector or data frame inputs
- returns an object which can be assigned
- can be calculated “by hand” using other R functions
  - example: `t.test()` can also be found by calculating the t test statistic, followed by `pt()` to obtain a p

Student's t-test

One-sample t-test

"Is sample mean $\bar{x}$ different from population mean μ?"

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$$

```r
mu <- 4147
x_bar <- mean(penguins$body_mass_g, na.rm=TRUE)
s <- sd(penguins$body_mass_g, na.rm=TRUE)
n <- penguins[!is.na(penguins$body_mass_g), "body_
  pull() %>%
  length()

t <- (x_bar - mu) / (s / sqrt(n))

# find p for this t score, on t distribution with
pt(q = t,
   df = (n - 1),
   lower.tail = TRUE)

# or:
t.test(x = penguins$body_mass_g,
       mu = mu)
```

Two-sample t-test (independent)

"Is sample mean $\bar{X}_1$ different from sample mean $\bar{X}_2$?"

$$t = \frac{\bar{X}_1 - \bar{X}_2}{s_p \cdot \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

```r
adelie <- penguins %>%
  filter(species == "Adelie")
chinstrap <- penguins %>%
  filter(species == "Chinstrap")

t.test(x = adelie$body_mass_g,
       y = chinstrap$body_mass_g)
```

Chi-squared (χ^2) test

"Is the conditional distribution across two categorical variables equal, or varied?"

`chisq.test(x, y = NULL)` where `x` is a matrix, or `x` and `y` are vectors

`chisq_test(formula)` where `formula` is written in the format `response ~ explanatory`

```r
# how are penguins distributed across species and islands?
chisq_test(penguins, island ~ species)
```

⚠ This is not a good example of a chi-squared test, statistically speaking: our data do not meet the requirements for the test! But the example is left here because R will let us do it! (and it is syntactically correct)

Tidy inference: `infer`

```r
t <- penguins %>%
  specify(response = body_mass_g) %>%
  hypothesize(null = "point", mu = 4147) %>%
  calculate(stat = "t") %>%
  pull()

penguins %>%
  specify(response = body_mass_g) %>%
  assume(distribution = "t") %>%
  visualize() +
  shade_p_value(obs_stat = t, direction = "greater")

penguins %>%
  t_test(response = body_mass_g,
         mu = 4147)

penguins %>%
  filter(species %in% c("Adelie", "Chinstrap")) %>%
  t_test(formula = body_mass_g ~ species)
```

Modeling

Simple linear regression

"What is the relationship between X and Y?"

Considering

$$Y = \beta_0 + \beta_1 X + \epsilon$$

where β_0 is the y intercept, β_1 is the slope coefficient, X is the explanatory or predictor variable, ϵ is irreducible error, and Y is the response variable. We want to estimate (or "fit") $\hat{\beta}_0$ and $\hat{\beta}_1$ such that the sum of squared residuals is minimized as much as possible.

`lm(formula, data)` fits a linear model and returns a model object, where formula is written `response ~ explanatory` and `data` is a data frame.

```r
lm(flipper_length_mm ~ bill_length_mm,
   data = penguins) %>%
  summary()

lm_flipper_bill <- lm(flipper_length_mm ~ bill_length_mm, data = penguins)
```

`predict()` outputs from a model

"For some x, what $\hat{y}$ does the model predict?"

Models in R implement behavior for the `predict()` function, which supplies a data frame of input values X and returns a data frame containing corresponding Y values. The input data frame needs variable(s) of the same name(s) as the predictor(s). Or you can omit the data frame and get the fitted values.

💡 `predict()` functions are found in documentation under article names like `predict.lm`, even though the function call is `predict()`.

```r
# min to max values, incrementing by 0.5:
x_range <- data.frame(bill_length_mm = seq(from=min(penguins$bill_length_mm, na.rm=TRUE), to=max(penguins$bil

predict(lm_flipper_bill, newdata = x_range)

# fitted values
predict(lm_flipper_bill)
```

Tidy modeling: parsnip

```r
linear_reg() %>%
  set_engine("lm") %>%
  fit(flipper_length_mm ~ bill_length_mm,
      data = penguins)

linear_reg() %>%
  set_engine("lm") %>%
  fit(flipper_length_mm ~ bill_length_mm,
      data = penguins)
```

Model *responsibly*

- Consider the validity and reliability of your measures
- Correlation $\neq$ causation
- Always keep in mind the assumptions made by your chosen method/model
- Validate unexpected results by recomputing “by hand” and/or running a different code implementation (or even in a different software)

Wrap up

Session in review

Today we learned about:

- how distributions and inference work in R
- linear modeling in R
- old and new ways of doing these things

Join us next week for machine learning intro!