COGS 536 Recitation Worksheet – 21 October

Today's goals

1. Understand and use `dnorm()`, `pnorm()`, and `qnorm()`.
2. Inspect a dataset with `str()` and `head()`.
3. Convert between numeric and categorical (character/factor) types with `dplyr::mutate()`.
4. Reorder factor levels for better plots and summaries.
5. Make essential `ggplot2` charts: scatterplot, bar (counts), histogram, density, boxplot, violin.

Data Source: The CSV file we're going to use if from UNICEF "Learning and Skills" datasets (https://data.unicef.org/resources/dataset/learning-and-skills/). Download it from ODTÜClass and save it locally to adjust the path in your code.

0) Setup

```
# Install once if needed:
# install.packages(c("tidyverse"))

library(tidyverse)  # dplyr, ggplot2, readr, etc.
```

Tip: Put this `.Rmd` and your CSV in a project folder. Create a `data/` subfolder for cleanliness.

1) Normal distribution helpers: `dnorm()`, `pnorm()`, `qnorm()`

- `dnorm(x, mean, sd)`: density (height of the curve) at point `x`.
- `pnorm(q, mean, sd)`: cumulative probability $P(X \leq q)$
- `qnorm(p, mean, sd)`: the quantile q such that $P(X \leq q) = p$

```
# Standard normal examples (mean = 0, sd = 1)
dnorm(0) # density at x = 0
pnorm(1.96) # ~0.975
qnorm(0.975) # ~1.96

# Non-standard normal: mean = 100, sd = 15 (e.g., test scores)
dnorm(100, mean = 100, sd = 15)
pnorm(120, mean = 100, sd = 15)    # P(X <= 120)
qnorm(0.90, mean = 100, sd = 15)   # 90th percentiles
```

Mini-exercise

1. What is P(X > 130) if X is distributed as N(100, 15^2)?
   (Hint: `1 - pnorm(130, 100, 15)`).
2. What score is the median (50th percentile)?
   (Hint: Use `qnorm()`.)
3. Compute the z-score threshold that leaves 2.5% in the upper tail.

Optional check: Replace numbers and re-run to see how mean/sd change the distribution.

2) Load and take a quick look at the UNICEF dataset

```
# Adjust the path if your file has a different name/location
csv_path <- "data/unicef_learning_skills.csv"

# Read CSV
unicef <- read_csv(csv_path, show_col_types = FALSE))

# Peek at structure and first rows
str(unicef)
head(unicef, 10)
```

If you get an error: Confirm the file path and that your CSV delimiter is a comma. If it's TSV, use `read_tsv()`.

Mini-exercise

1. How many rows and columns does your dataset have? (`dim(unicef)`)
2. Identify two **numeric** columns and one **categorical** column from `str()`.
3. Use `summary()` on a numeric column to get min/median/mean/quantiles.

3) Changing data types with `mutate()`

Convert numeric → character (or factor), and character → numeric.

```
# 1. Convert numeric (Subject label) to character/factor
# Suppose 'Subject' is numeric but represents a categorical
label: make it character/factor

unicef <- unicef %>%
  mutate(
         # Convert 'Subject' (numeric) to character
         Subject_chr = as.character(Subject),
         # Convert 'Subject' (numeric) to character
         Subject_fct = as.factor(Subject)
)
```

```
# 2. Convert character (Grade) to factor (A common character-to-factor change)
# 'Grade' is a character column, but it's clearly a set of categories: make it a factor

unicef <- unicef %>%
      mutate(
             Grade_fct = as.factor(Grade)
)
```

Note: `as.numeric()` on a non-numeric string (e.g., "N/A") becomes `NA`. Use `readr::parse_number()` for messy strings.

Mini-exercise

1. Pick one column that should be **categorical** and convert it to a **factor**.
2. Pick one column that should be **numeric** and convert it to **numeric**.
3. Count how many NAs appear after conversion (`sum(is.na(col))`). Why?

4) Reordering factor levels

Reorder factors to control plot and table ordering.

```
# Strategy: Put 'Age 7 to 14' (the broader/older group) before 'Grade 2/3'.
# Note: You can use the original 'Grade' if you converted it to factor *in place* in a previous step, but it's safer to use the new 'Grade_fct' column created above.

unicef <- unicef %>%
     mutate(
            Grade_fct_reordered = factor(Grade_fct, levels = c("Age 7 to 14", "Grade 2/3")) )
```

Strategy: Put your most important or most frequent level first, or order by a statistic.

```
# Inspect the changes
# Use table() to see the count of observations in the old and new factor columns

cat("Original Grade levels/counts:\n")
print(table(unicef$Grade_fct, useNA = "ifany"))

cat("\nReordered Grade levels/counts:\n")
print(table(unicef$Grade_fct_reordered, useNA = "ifany"))

# Use str() to confirm the data type and level order
str(df)
```

Mini-exercise

1. Choose a factor column and set a meaningful **manual order** of levels.
2. Choose a numeric indicator and **reorder** a categorical column by its mean using `factor levels.`

5) Core `ggplot2` charts

We'll assume you have a numeric indicator column (e.g., `indicator_value`) and a categorical column (e.g., `region`), **rename to columns that exist in your CSV**.

5.1 Scatterplot

```
# Let's visualise the relationship between scores for girls
and boys:

unicef %>%
  ggplot(aes(x = `Girls score`, y = `Boys score`)) +
  geom_point(alpha = 0.7) +
  labs(title = "Relationship Between Girls' and Boys' Test
Scores", x = "Girls' Score (%)", y = "Boys' Score (%)"
  )
```

Think: Try more examples and check if you see linear/nonlinear patterns? Outliers?

5.2 Bar chart for categorical counts (`geom_bar`)

```
# Let's create a chart visualizes the count of records for
each level within the Category column.

unicef %>%
  ggplot(aes(x = Category)) +
  geom_bar() +
  labs(
       title = "Count of Records by Subject Category",
       x = NULL, # Remove x-axis label as the labels
themselves are descriptive
       y = "Count of Observations" ) +
# Rotate text slightly if the labels were longer
theme(axis.text.x = element_text(angle = 30, hjust = 1))
```

5.3 Histogram

```
# Use a numeric column, like 'Girls score'.
# Remember to use backticks (`) for column names with spaces!
unicef %>%
  ggplot(aes(x = `Girls score`)) +
  geom_histogram(bins = 30) +
  labs(title = "Distribution of Girls' Test Scores", x =
"Girls' Score (%)", y = "Frequency")
```

5.4 Density plot

```
unicef %>%
     ggplot(aes(x = `Boys score`)) + geom_density() + labs(
title = "Density Plot of Boys' Test Scores", x = "Boys' Score
(%)", y = "Density" )

# Example of using a categorical variable to split the density
unicef %>%
  ggplot(aes(x = `Boys score`, color = Category)) +
  geom_density(alpha = 0.5) + # Use alpha for transparency
when overlaying
  labs(
    title = "Density of Boys' Scores by Subject Category",
    x = "Boys' Score (%)",
    y = "Density",
    color = "Subject"
  )
```

5.5 Boxplot

```
# Map the categorical variable 'Grade' to the x-axis
# and the numeric variable 'Girls score' to the y-axis.
unicef %>%
  ggplot(aes(x = Grade, y = `Girls score`)) +
  geom_boxplot() +
  labs(
    title = "Girls' Score Distribution by Grade Level",
    x = NULL, # Remove x-axis label as the labels themselves
are descriptive
    y = "Girls' Score (%)"
  ) +
  # Rotate text slightly for better label fit
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```

5.6 Violin plot

```
# Map the categorical variable 'Grade' to the x-axis
# and the numeric variable 'Boys score' to the y-axis.
unicef %>%
  ggplot(aes(x = Grade, y = `Boys score`)) +
  geom_violin(trim = FALSE) +
  labs(
    title = "Boys' Score Distribution by Grade Level (Violin
Plot)",
    x = NULL,
    y = "Boys' Score (%)"
  ) +
  # Rotate text for label fit
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```

Mini-exercise

1. Make a **scatterplot** of two numeric indicators. Add a smooth trend with `geom_smooth(se = FALSE)`.
2. Create a **bar chart** of a categorical variable; then reorder its levels by descending count.
3. Compare the **histogram** and **density** of the same numeric variable. Which communicates the shape better here?
4. Plot **boxplot vs violin** for the same grouping. What extra info does the violin show?

Appendix

- `dnorm(x, m, s)`, `pnorm(q, m, s)`, `qnorm(p, m, s)`
- Inspect data: `str(df)`, `head(df)`, `summary(df)`
- Type conversions: `mutate(new = as.numeric(old))`, `as.character()`, `as.factor()`
- Factor ordering: `factor(x, levels = ...)`, `forcats::fct_reorder(x, y)`
- Core plots: `geom_point()`, `geom_bar()`, `geom_histogram()`, `geom_density()`, `geom_boxplot()`, `geom_violin()`

Next week (preview): filtering (filter()), finding NAs, grouped summaries (group_by() + summarise()), and creating new columns (mutate()).