STATS-506 HW3

Zekai Xu

2025-09-29

Github Repo

Question 1

Part a

```r
# Load Dataset
aux <- as.data.frame(read_xpt("./AUX_I.xpt"))
demo <- as.data.frame(read_xpt("./DEMO_I.xpt"))

# Merge Dataset
df <- inner_join(aux, demo, by = "SEQN")

# Print dimension of the merged dataframe
print(dim(df))
```

```
[1] 4582  119
```

Part b

```r
# Select columns used in this question
var2keep <- c(
  "SEQN", #Identifier
  "RIAGENDR", #Gender
  "DMDCITZN", #Citizenship
  "DMDHHSZA", #Number of children 5 years or younger in the household
  "INDHHIN2", #Annual household income
  "AUXTWIDR", #Tympanometric width, right ear
  "AUXTWIDL"  #Tympanometric width, left ear
)
df_partb <- df[var2keep]

# Rename columns
names(df_partb) <- c("id", "gender", "citizenship", "numOfKids", "income", "tw_re", "tw_le")

# Convert "unknown" data points to NA values
df_partb$citizenship[df_partb$citizenship %in% c(77, 99)] <- NA
df_partb$income[df_partb$income %in% c(77, 99)] <- NA

# Convert categorical column into factor
df_partb$gender <- factor(
  df_partb$gender,
  levels = c(1, 2),
  labels = c("Male", "Female")
)
df_partb$citizenship <- factor(
  df_partb$citizenship,
  levels = c(1, 2),
  labels = c("USA", "Other")
)
#df_partb$numOfKids <- factor(df_partb$numOfKids)
#df_partb$income <- factor(df_partb$income)

# Overview of processed data
glimpse(df_partb)
```

```
Rows: 4,582
Columns: 7
$ id          <dbl> 83732, 83733, 83735, 83736, 83741, 83742, 83744, 83747, 83~
$ gender      <fct> Male, Male, Female, Female, Male, Female, Male, Male, Male~
$ citizenship <fct> USA, Other, USA, USA, USA, Other, USA, USA, USA, USA, USA,~
$ numOfKids   <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0~
$ income      <dbl> 10, 4, 10, 7, 7, 6, 3, 3, 10, 15, 6, 5, 5, 1, 8, 14, 10, 2~
$ tw_re       <dbl> 49, 83, 110, 99, 73, 99, 122, 69, 63, 78, 44, 133, 144, NA~
$ tw_le       <dbl> 76, 51, 49, 116, 73, 70, 111, 73, 60, 88, 39, 112, NA, NA,~
```

Part c

```r
# Create formulas for 4 Poisson Regression model
preds_1r <- c("gender")
preds_2r <- c("gender", "citizenship", "numOfKids", "income")
preds_1l <- c("gender")
preds_2l <- c("gender", "citizenship", "numOfKids", "income")
fm_1r <- reformulate(termlabels = preds_1r, response = "tw_re", intercept = TRUE)
fm_2r <- reformulate(termlabels = preds_2r, response = "tw_re", intercept = TRUE)
fm_1l <- reformulate(termlabels = preds_1l, response = "tw_le", intercept = TRUE)
fm_2l <- reformulate(termlabels = preds_2l, response = "tw_le", intercept = TRUE)

# Model
m_1r <- glm(formula = fm_1r, data = df_partb, family = poisson(link = "log"))
m_2r <- glm(formula = fm_2r, data = df_partb, family = poisson(link = "log"))
m_1l <- glm(formula = fm_1l, data = df_partb, family = poisson(link = "log"))
m_2l <- glm(formula = fm_2l, data = df_partb, family = poisson(link = "log"))
```

Table 1: Incidence Rate Ratios (IRR) with 95 percent CI and p-values

		Effect Size	
Model	Term	IRR [95 percent CI]	p-value
1R (Right: gender)	genderFemale	1.01 [1.00, 1.02]	0.006
2R (Right: gender + citizenship + numOfKids + income)	genderFemale	1.01 [1.01, 1.02]	<0.001
2R (Right: gender + citizenship + numOfKids + income)	citizenshipOther	1.07 [1.07, 1.08]	<0.001
2R (Right: gender + citizenship + numOfKids + income)	numOfKids	1.00 [1.00, 1.01]	0.860
2R (Right: gender + citizenship + numOfKids + income)	income	0.99 [0.99, 1.00]	<0.001
1L (Left: gender)	genderFemale	1.01 [1.01, 1.02]	<0.001
2L (Left: gender + citizenship + numOfKids + income)	genderFemale	1.02 [1.01, 1.02]	<0.001
2L (Left: gender + citizenship + numOfKids + income)	citizenshipOther	1.04 [1.03, 1.05]	<0.001
2L (Left: gender + citizenship + numOfKids + income)	numOfKids	0.98 [0.98, 0.99]	<0.001
2L (Left: gender + citizenship + numOfKids + income)	income	1.00 [1.00, 1.00]	<0.001

Part d

```r
# ---------- (A) Likelihood Ratio Test: does gender improve fit? ----------
m_2l_nogender <- update(m_2l, . ~ . - gender)
anova(m_2l_nogender, m_2l, test = "LRT")  # reports Chi-sq, df, p-value
```

Table 2: Model Fit Statistics: Sample Size, McFadden Pseudo-R2, and AIC

Model	N	Pseudo-R2 (McFadden)	AIC
1R	4,149	0.000	96,618
2R	3,886	0.067	90,169
1L	4,103	0.000	98,685
2L	3,847	0.070	91,835

```
Analysis of Deviance Table

Model 1: tw_le ~ citizenship + numOfKids + income
Model 2: tw_le ~ gender + citizenship + numOfKids + income
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1      3843      68040
2      3842      68021  1   19.318 1.107e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# ---------- (B) IRR difference (Wald/contrast): Female vs Male ----------
# emmeans on link scale, then contrast on response scale -> ratio = IRR
emm <- emmeans(m_2l, ~ gender)                 # marginal means (on link scale)
pairs(emm) %>% summary(type = "response")      # ratio, 95% CI, p-value (IRR Female/Male)
```

```
 contrast      ratio      SE  df null z.ratio p.value
 Male / Female 0.985 0.00346 Inf    1  -4.394  <.0001

Results are averaged over the levels of: citizenship
Tests are performed on the log scale
```

```r
# ---------- (C) Predicted mean TW by gender (adjusted) ----------
# Response-scale marginal means (i.e., predicted means for each gender)
summary(emm, type = "response")                # mean, SE, 95% CI for each gender
```

```
 gender rate    SE  df asymp.LCL asymp.UCL
 Male   85.7 0.246 Inf      85.2      86.1
 Female 87.0 0.243 Inf      86.5      87.5

Results are averaged over the levels of: citizenship
Confidence level used: 0.95
Intervals are back-transformed from the log scale
```

Evidence of Gender Difference (Model 2L)

Task. Assess whether males and females differ in (i) their **incidence rate ratio (IRR)** and (ii) the **predicted value** of left-ear tympanometric width (TW), adjusting for citizenship, number of children, and income.

1) Likelihood-Ratio Test (model-level evidence)

- Compare Model 1 (without gender) vs Model 2L (with gender).
- ΔDeviance = **19.318** on 1 df, **p = 1.11×10** .
- **Conclusion:** Reject *H : no gender effect.* Adding gender significantly improves model fit.

2) IRR Difference (contrast/Wald test)

- **IRR (Male / Female) = 0.985**, z = -4.394, **p < 0.0001**.
- Equivalently, **IRR (Female / Male) 1.015** $\rightarrow$ females have about **1.5% higher** expected left-ear TW than males, holding covariates fixed.
- Interpretation: IRR differs significantly from 1, indicating a statistically significant gender effect on the rate (mean) of TW.

3) Adjusted Predicted Means (response scale)

- **Male:** 85.7 (95% CI: **85.2–86.1**)
- **Female:** 87.0 (95% CI: **86.5–87.5**)
- **Conclusion:** Predicted means differ in the same direction as the IRR result; females have a slightly higher adjusted TW.

Overall Interpretation

Both the model-level LRT and the IRR/mean comparisons show a **statistically significant** gender difference in left-ear TW after adjustment. The **effect size is small** (**1.5%** higher for females;   **1.3 units** difference in predicted means), so practical significance should be considered alongside statistical significance.

Question 2

Part a

```
# Connect Database
file_path <- "./sakila_master.db"
con <- dbConnect(SQLite(), dbname = file_path)

## ==== Method 1: extract -> compute in R ====
method1 <- function() {
  stores <- dbGetQuery(con, "SELECT store_id FROM store")

  customers <- dbGetQuery(con, "
    SELECT store_id, active
    FROM customer
  ") %>%
    mutate(                          # normalize 'active' to 0/1
      active = as.integer(tolower(trimws(as.character(active))) %in%
                          c("1","t","true","y","yes"))
    )

  per_store <- customers %>%
    group_by(store_id) %>%
    summarise(
      n_customers = n(),
      n_active    = sum(active),
      .groups = "drop"
    ) %>%
    mutate(pct_active = ifelse(n_customers > 0, 100 * n_active / n_customers, NA_real_))

  stores %>%
    left_join(per_store, by = "store_id") %>%
    select(store_id, n_customers, pct_active) %>%
    arrange(store_id)
}

## ==== Method 2: single SQL ====
method2 <- function() {
  dbGetQuery(con, "
    SELECT
      s.store_id,
      COUNT(c.customer_id) AS n_customers,
      CASE
        WHEN COUNT(c.customer_id) = 0 THEN NULL
        ELSE 100.0 * SUM(
          CASE
            WHEN lower(c.active) IN ('1','t','true','y','yes') THEN 1
            WHEN lower(c.active) IN ('0','f','false','n','no','') THEN 0
            WHEN c.active = 1 THEN 1
            WHEN c.active = 0 THEN 0
            ELSE 0
          END
        ) * 1.0 / COUNT(c.customer_id)
      END AS pct_active
    FROM store AS s
    LEFT JOIN customer AS c
      ON c.store_id = s.store_id
    GROUP BY s.store_id
    ORDER BY s.store_id
  ")
}

## ==== Run & print ====
res_r   <- method1() %>% mutate(pct_active = round(pct_active, 2))
res_sql <- method2()  %>% mutate(pct_active = round(pct_active, 2))

print(res_r)
```

```
  store_id n_customers pct_active
1        1         326      97.55
2        2         273      97.44
```

```
print(res_sql)
```

```
  store_id n_customers pct_active
1        1         326      97.55
2        2         273      97.44
```

```
## ==== Microbenchmark ====
microbenchmark(
  two_step_R = method1(),
  single_SQL = method2(),
  times = 20L
)
```

```
Warning in microbenchmark(two_step_R = method1(), single_SQL = method2(), :
less accurate nanosecond times to avoid potential integer overflows
```

```
Unit: microseconds
       expr      min       lq     mean   median        uq       max neval
 two_step_R 2795.708 2846.896 2966.491 2942.365 2979.9620  3716.568    20
 single_SQL  212.503  224.147 2819.402  231.035  246.4305 51309.778    20
```

Directly using SQL query is way faster than SQl + R approach.

Part b

```
## =============== Method 1: Extract tables, then join in R =================
method1 <- function() {
  staff   <- dbGetQuery(con, "SELECT staff_id, first_name, last_name, address_id FROM staff"
  address <- dbGetQuery(con, "SELECT address_id, city_id FROM address")
  city    <- dbGetQuery(con, "SELECT city_id, country_id FROM city")
  country <- dbGetQuery(con, "SELECT country_id, country FROM country")

  staff %>%
    left_join(address, by = "address_id") %>%
    left_join(city,    by = "city_id") %>%
    left_join(country, by = "country_id") %>%
    mutate(full_name = paste(first_name, last_name)) %>%
    select(staff_id, first_name, last_name, full_name, country) %>%
    arrange(staff_id)
}

## =============== Method 2: Single SQL query (one-shot) ====================
method2 <- function() {
  dbGetQuery(con, "
    SELECT
      s.staff_id,
      s.first_name,
      s.last_name,
      (s.first_name || ' ' || s.last_name) AS full_name,
      co.country
    FROM staff AS s
    JOIN address AS a ON s.address_id = a.address_id
    JOIN city    AS ci ON a.city_id    = ci.city_id
    JOIN country AS co ON ci.country_id = co.country_id
    ORDER BY s.staff_id
  ")
}

## =============== Run both & show a sample ==================================
res_r   <- method1()
res_sql <- method2()

print(head(res_r, 10))
```

```
  staff_id first_name last_name    full_name   country
1        1       Mike   Hillyer Mike Hillyer    Canada
2        2        Jon  Stephens Jon Stephens Australia
```

```
print(head(res_sql, 10))
```

```
  staff_id first_name last_name    full_name   country
1        1       Mike   Hillyer Mike Hillyer    Canada
2        2        Jon  Stephens Jon Stephens Australia
```

```
## =============== Microbenchmark ============================================
mb <- microbenchmark(
  two_step_R = method1(),
  single_SQL = method2(),
  times = 50L
)
print(mb)
```

```
Unit: microseconds
       expr      min       lq      mean   median       uq      max neval
 two_step_R 2632.897 2685.869 2954.3280 2780.251 2941.791 6246.678    50
 single_SQL   96.678  106.149  127.9561  113.283  120.868  777.565    50
```

Directly using SQL query is way faster than SQl + R approach.

Part c

```r
# ---------------- Method 1: extract tables -> compute in R ----------------
method1 <- function() {
  payment  <- dbGetQuery(con, "SELECT rental_id, amount FROM payment")
  rental   <- dbGetQuery(con, "SELECT rental_id, inventory_id FROM rental")
  inventory<- dbGetQuery(con, "SELECT inventory_id, film_id FROM inventory")
  film     <- dbGetQuery(con, "SELECT film_id, title FROM film")

  # total $ per rental, then map rental -> film -> title
  per_rental <- payment %>%
    group_by(rental_id) %>%
    summarise(rental_value = sum(amount), .groups = "drop") %>%
    inner_join(rental,    by = "rental_id") %>%
    inner_join(inventory, by = "inventory_id") %>%
    inner_join(film,      by = "film_id")

  max_val <- max(per_rental$rental_value, na.rm = TRUE)

  per_rental %>%
    filter(rental_value == max_val) %>%
    distinct(title, rental_value) %>%
    arrange(title)
}

# ---------------- Method 2: single SQL query ----------------
method2 <- function() {
  dbGetQuery(con, "
    WITH per_rental AS (
      SELECT rental_id, SUM(amount) AS rental_value
      FROM payment
      GROUP BY rental_id
    ),
    max_rental AS (
      SELECT MAX(rental_value) AS max_val FROM per_rental
    )
    SELECT f.title, pr.rental_value
    FROM per_rental pr
    JOIN rental r     ON pr.rental_id = r.rental_id
    JOIN inventory i  ON r.inventory_id = i.inventory_id
    JOIN film f       ON i.film_id = f.film_id
    WHERE pr.rental_value = (SELECT max_val FROM max_rental)
    ORDER BY f.title
  ")
}

# ---------------- Run & show ----------------
res_r   <- method1()
res_sql <- method2()

print(res_r)
```

```
# A tibble: 9 x 2
  title                  rental_value
  <chr>                         <dbl>
1 FLINTSTONES HAPPINESS          12.0
2 MIDSUMMER GROUNDHOG            12.0
3 MINE TITANS                    12.0
4 SCORPION APOLLO                12.0
5 SHOW LORD                      12.0
6 STING PERSONAL                 12.0
7 TIES HUNGER                    12.0
8 TRAP GUYS                      12.0
9 VIRTUAL SPOILERS               12.0
```

```
print(res_sql)
```

```
                   title rental_value
1  FLINTSTONES HAPPINESS        11.99
2    MIDSUMMER GROUNDHOG        11.99
3            MINE TITANS        11.99
4        SCORPION APOLLO        11.99
5        SCORPION APOLLO        11.99
6              SHOW LORD        11.99
7         STING PERSONAL        11.99
8            TIES HUNGER        11.99
9              TRAP GUYS        11.99
10      VIRTUAL SPOILERS        11.99
```

```
# ---------------- Microbenchmark ----------------
microbenchmark(
  two_step_R = method1(),
  single_SQL = method2(),
  times = 20L
)
```

```
Unit: milliseconds
       expr       min       lq      mean    median       uq       max neval
 two_step_R 24.879169 25.24604 26.326065 25.761981 27.76930 28.457116    20
 single_SQL  5.394903  5.46163  5.868439  6.002052  6.15859  6.744705    20
```

Directly using SQL query is way faster than SQl + R approach.

Question 3

Part a

```
# Load Dataset
df <- read.csv("./au-500.csv", sep = ",")

# Calculate percentage
pct <- sum(str_detect(df$web, regex("\\.com$"))) / dim(df)[1] * 100
cat(sprintf("Percentage of websites that end with `.com`: %.2f%%\n", pct))
```

```
Percentage of websites that end with `.com`: 0.00%
```

Part b

```
dom <- str_extract(df$email, regex("(?<=@)(?:[A-Za-z0-9-]+\\.)+[A-Za-z]{2,}"))
sort(table(dom), decreasing = TRUE)[1]
```

```
hotmail.com
        114
```

Part c

```
p_excl_comma_ws <- mean(str_detect(df$company_name, "[^\\p{L},\\s]"), na.rm = TRUE)
p_excl_comma_ws_amp <- mean(str_detect(df$company_name, "[^\\p{L},\\s&]"), na.rm = TRUE)

cat(sprintf("Proportion with non-alphabetic (excluding comma/whitespace): %.2f%%\n",
            100 * p_excl_comma_ws))
```

```
Proportion with non-alphabetic (excluding comma/whitespace): 9.00%
```

```
cat(sprintf("Proportion with non-alphabetic (excluding comma/whitespace/&): %.2f%%\n",
            100 * p_excl_comma_ws_amp))
```

```
Proportion with non-alphabetic (excluding comma/whitespace/&): 0.80%
```

Part d

```
phone_cols <- names(df)[str_detect(tolower(names(df)), "(phone|mobile)")]

fmt_au_mobile <- function(x) {
  d <- str_replace_all(x, "[^0-9]", "") # strip non-digits
  ifelse(nchar(d) == 10,
         paste0(substr(d, 1, 4), "-", substr(d, 5, 7), "-", substr(d, 8, 10)),
         NA_character_)  # not 10 digits -> NA
}


cat("Before:\n")
```

```
Before:
```

```
print(utils::head(df[phone_cols], 10))
```

```
         phone1       phone2
1  03-8174-9123 0458-665-290
2  07-9997-3366 0497-622-620
3  08-5558-9019 0427-885-282
4  02-6044-4682 0443-795-912
5  02-1455-6085 0453-666-885
```

```
6  08-7868-1355 0451-966-921
7  08-6522-8931 0427-991-688
8  02-5226-9402 0415-961-606
9  07-3184-9989 0411-732-965
10 08-6890-4661 0461-862-457
```

```r
df <- df %>% mutate(across(all_of(phone_cols), fmt_au_mobile))

cat("\nAfter (cell format 1234-567-890):\n")
```

```
After (cell format 1234-567-890):
```

```r
print(head(df[phone_cols], 10))
```

```
         phone1       phone2
1  0381-749-123 0458-665-290
2  0799-973-366 0497-622-620
3  0855-589-019 0427-885-282
4  0260-444-682 0443-795-912
5  0214-556-085 0453-666-885
6  0878-681-355 0451-966-921
7  0865-228-931 0427-991-688
8  0252-269-402 0415-961-606
9  0731-849-989 0411-732-965
10 0868-904-661 0461-862-457
```

Part e

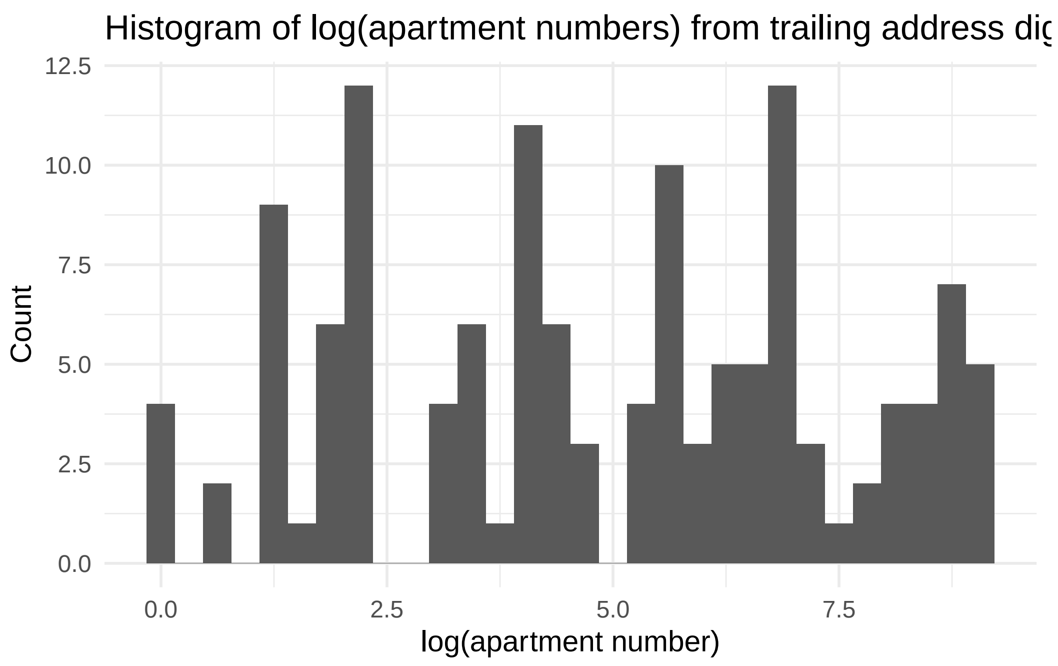
```r
addr_col <- names(df)[str_detect(tolower(names(df)), "address")][1]

# Extract trailing digits as apartment number; keep positive integers only
apt_num <- df[[addr_col]] %>%
  as.character() %>%
  str_extract("\\d+\\s*$") %>%   # trailing number at end of string
  as.numeric() %>%
  { .[!is.na(.) & . > 0] }

# Log-transform (natural log)
```

```r
log_apt <- log(apt_num)

# Plot histogram (ggplot2)
ggplot(data.frame(log_apt = log_apt), aes(x = log_apt)) +
  geom_histogram(bins = 30, linewidth = 0.2) +
  labs(
    title = "Histogram of log(apartment numbers) from trailing address digits",
    x = "log(apartment number)",
    y = "Count"
  ) +
  theme_minimal()
```



Part f

```r
# Leading digit via logs (no string ops needed)
ld <- floor(apt_num / (10 ^ floor(log10(apt_num))))
ld <- ld[ld %in% 1:9]

obs_counts <- table(factor(ld, levels = 1:9))
n          <- sum(obs_counts)
obs_prop   <- as.numeric(obs_counts) / n
```

```r
exp_prop   <- log10(1 + 1 / (1:9))

# Chi-squared goodness-of-fit vs Benford
chisq.test(as.numeric(obs_counts), p = exp_prop, rescale.p = TRUE)
```

```

	Chi-squared test for given probabilities

data:  as.numeric(obs_counts)
X-squared = 62.268, df = 8, p-value = 1.67e-10
```

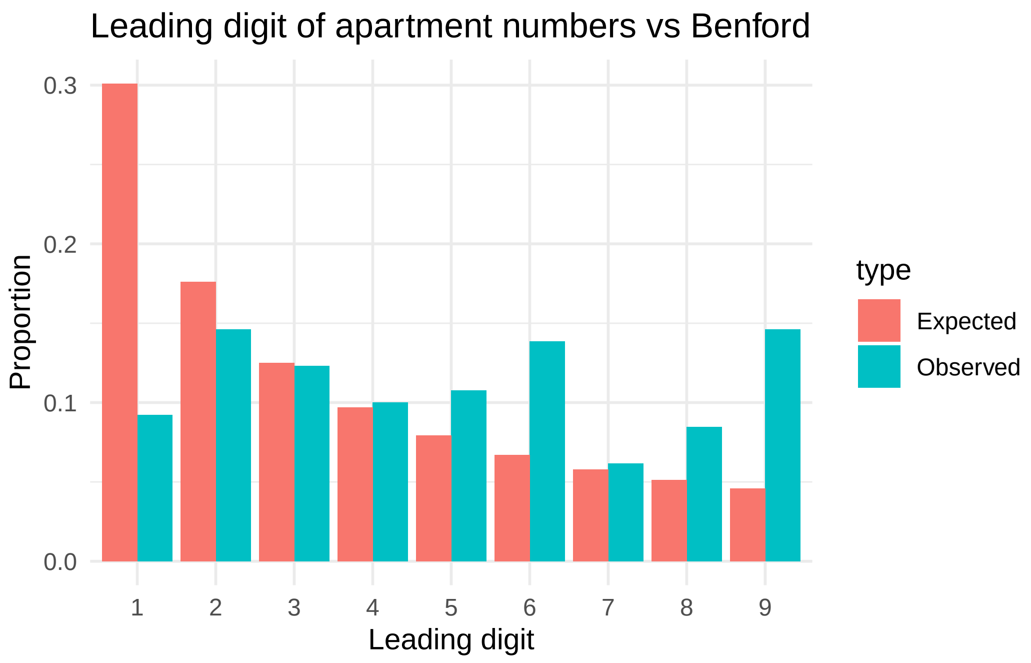
```r
# (optional) quick summary
print(data.frame(
  digit    = 1:9,
  observed = as.numeric(obs_counts),
  expected = round(n * exp_prop, 1),
  obs_prop = round(obs_prop, 4),
  exp_prop = round(exp_prop, 4)
))
```

```
  digit observed expected obs_prop exp_prop
1     1       12     39.1   0.0923   0.3010
2     2       19     22.9   0.1462   0.1761
3     3       16     16.2   0.1231   0.1249
4     4       13     12.6   0.1000   0.0969
5     5       14     10.3   0.1077   0.0792
6     6       18      8.7   0.1385   0.0669
7     7        8      7.5   0.0615   0.0580
8     8       11      6.6   0.0846   0.0512
9     9       19      5.9   0.1462   0.0458
```

```r
# (optional) simple plot
library(ggplot2)
library(tidyr)

plot_df <- data.frame(digit = factor(1:9), Observed = obs_prop, Expected = exp_prop) |>
  pivot_longer(c(Observed, Expected), names_to = "type", values_to = "prop")

ggplot(plot_df, aes(digit, prop, fill = type)) +
  geom_col(position = "dodge") +
  labs(title = "Leading digit of apartment numbers vs Benford",
       x = "Leading digit", y = "Proportion") +
  theme_minimal()
```



The apartment numbers do not follow Benford's law. The leading digit "1" occurs far less than Benford's 30% expectation, while higher digits (especially 6–9) occur more often, and the distribution lacks the characteristic monotonic decline. These systematic deviations indicate the data would likely fail a Benford goodness-of-fit test. This is consistent with apartment numbers being administratively assigned sequences rather than naturally generated measurements.