Load in Data:

-What this does: Loads the R packages used for reading data (`readr`), wrangling (`dplyr`, `tidyr`), and plotting (`ggplot2`). - Output and data:No dataset is created here; this just makes functions like `read_csv()`, `%>%`, `mutate()`, `pivot_longer()`, and `ggplot()` available. - If something fails: An error here usually means a package isn't installed (you'd need to install it), or your environment can't load it.

```
attendance <- readr::read_csv(
  "https://raw.githubusercontent.com/rfordatascience/tidytuesday/main/data/2020/2020-02-04/attenda
)

standings <- readr::read_csv(
  "https://raw.githubusercontent.com/rfordatascience/tidytuesday/main/data/2020/2020-02-04/standi
)

games <- readr::read_csv(
  "https://raw.githubusercontent.com/rfordatascience/tidytuesday/main/data/2020/2020-02-04/games.
)
```

Load the NFL datasets (attendance, standings, games) - What this does: Reads three CSV files from the TidyTuesday NFL data release into three tibbles: `attendance`, `standings`, and `games`. - Output/data: After this runs, you should have three data frames in memory. A quick check is to print `glimpse(attendance)` / `glimpse(standings)` / `glimpse(games)` to confirm expected columns (team identifiers, year/week, game stats, etc.). - What to look for: Verify that `year` and `week` exist (sometimes as character), and that team name columns look consistent across tables before joining later.

```
# 2. Clean attendance: make full team name & integer year/week

attendance_clean <- attendance %>%
  mutate(
    year      = as.integer(year),
    week      = as.integer(week),
    full_team = paste(team, team_name)  # "Arizona Cardinals"
  )
```

Data Wrangling + Feature creation:

Clean attendance full team name + numeric year/week - What this does: Creates `attendance_clean` by standardizing team labels, combining city + team name into one string, and converting `year` and `week` into integers. - Output/data: `attendance_clean` should include a `full_team` column (e.g., "Arizona Cardinals") plus numeric `year` and `week` that can safely be used for joins. - Why it matters: Joins later depend on exact matching keys. If `year/week` are characters in one table and integers in another, joins will silently drop rows or create lots of `NA`s.

Build team-game level dataset `games_team` - What this does: Reshapes the raw `games` table into a team-game table: each NFL game becomes two rows (one for the home team and one for the away team). - Output/data: `games_team` should contain identifiers like `game_id`, `team`, `opponent`, `year`, `week`, and outcome variables like `win` (from the team's perspective). - What to check: `nrow(games_team)` should be roughly 2 × number of games Also check that `team` and `opponent` are full team names and look consistent.

```
# 4. Rolling stats per team/year


games_rolling <- games_team %>%
  group_by(team, year) %>%
  arrange(week, date, time, .by_group = TRUE) %>%
  mutate(
    games_played_prior = row_number() - 1L,

    cum_yds_for       = lag(cumsum(yds_for),       default = 0),
    cum_yds_against   = lag(cumsum(yds_against),   default = 0),

    cum_wins          = lag(cumsum(win),          default = 0),
    cum_losses        = lag(cumsum(loss),         default = 0),
    cum_ties          = lag(cumsum(tie_flag),     default = 0),

    cum_pts_for       = lag(cumsum(pts_for),       default = 0),
    cum_pts_against   = lag(cumsum(pts_against),   default = 0),

    cum_to_for        = lag(cumsum(turnovers_for),     default = 0),
    cum_to_against    = lag(cumsum(turnovers_against), default = 0),

    avg_yds_for       = if_else(games_played_prior > 0,
                                cum_yds_for / games_played_prior, NA_real_),
    avg_yds_against   = if_else(games_played_prior > 0,
                                cum_yds_against / games_played_prior, NA_real_),
    avg_yds_diff      = avg_yds_for - avg_yds_against,

    avg_pts_for       = if_else(games_played_prior > 0,
                                cum_pts_for / games_played_prior, NA_real_),
    avg_pts_against   = if_else(games_played_prior > 0,
                                cum_pts_against / games_played_prior, NA_real_),
    avg_pts_diff      = avg_pts_for - avg_pts_against,

    avg_to_for        = if_else(games_played_prior > 0,
                                cum_to_for / games_played_prior, NA_real_),
    avg_to_against    = if_else(games_played_prior > 0,
                                cum_to_against / games_played_prior, NA_real_),
    avg_to_diff       = avg_to_for - avg_to_against,

    win_pct           = if_else(games_played_prior > 0,
                                cum_wins / games_played_prior, NA_real_)
```

```
  ) %>%
  ungroup()
```

Compute rolling (past-only) team metrics - What this does: Within each `(team, year)`, it orders games by week/date and computes rolling averages using only prior games (via `lag(...)`). - Output/data: The resulting table should now include features like `win_pct`, `avg_pts_diff`, `avg_yds_diff`, and `avg_to_diff` that represent a team's form entering each game. - Interpretation: Early-season weeks will often have `NA` (or less stable values) because there aren't many prior games. That's expected and is part of why later models may drop `NA`s or treat them carefully.

```
# 5. Home / away stats & games_model


team_stats <- games_rolling %>%
  filter(games_played_prior >= 4) %>%
  select(
    game_id, year, week, team, home_away, win,
    win_pct,
    avg_pts_diff, avg_yds_diff, avg_to_diff
  )

home_stats <- team_stats %>%
  filter(home_away == "home_team") %>%
  rename_with(~ paste0(.x, "_home"),
              c(team, home_away, win, win_pct,
                avg_pts_diff, avg_yds_diff, avg_to_diff))

away_stats <- team_stats %>%
  filter(home_away == "away_team") %>%
  rename_with(~ paste0(.x, "_away"),
              c(team, home_away, win, win_pct,
                avg_pts_diff, avg_yds_diff, avg_to_diff))

games_model <- home_stats %>%
  inner_join(
    away_stats,
    by = c("game_id", "year", "week")
  ) %>%
  mutate(
    win_home       = win_home,
    delta_win_pct  = win_pct_home      - win_pct_away,
    delta_pts_diff = avg_pts_diff_home - avg_pts_diff_away,
    delta_yds_diff = avg_yds_diff_home - avg_yds_diff_away,
    delta_to_diff  = avg_to_diff_home  - avg_to_diff_away
  )

# ------------------------------------------------------------------
#
# ------------------------------------------------------------------
```

Create game-level model features (`games_model`) - What this does: Separates the team-game data into home and away views, then joins them back together to make one row per game. - Output/data: `games_model` should be a game-level dataset with: - `win_home` (1 if the home team won, 0 otherwise) - "delta" features like `delta_win_pct`, `delta_pts_diff`, `delta_yds_diff`, `delta_to_diff` (home metric minus away metric) - Interpretation: Positive deltas mean the home team entered the game with stronger recent performance than the away team on that metric. These deltas are set up to be predictive features.

```
attendance_home <- games_team %>%
  filter(home_away == "home_team") %>%
  left_join(
    attendance_clean,
    by = c("team" = "full_team", "year", "week")
  ) %>%
  transmute(
    game_id,
    team_home = team,          # full team name, e.g. "Arizona Cardinals"
    year,
    week,
    home_attendance = weekly_attendance
  )

#
games_att <- games_model %>%
  inner_join(
    attendance_home,
    by = c("game_id", "team_home", "year", "week")
  )
```

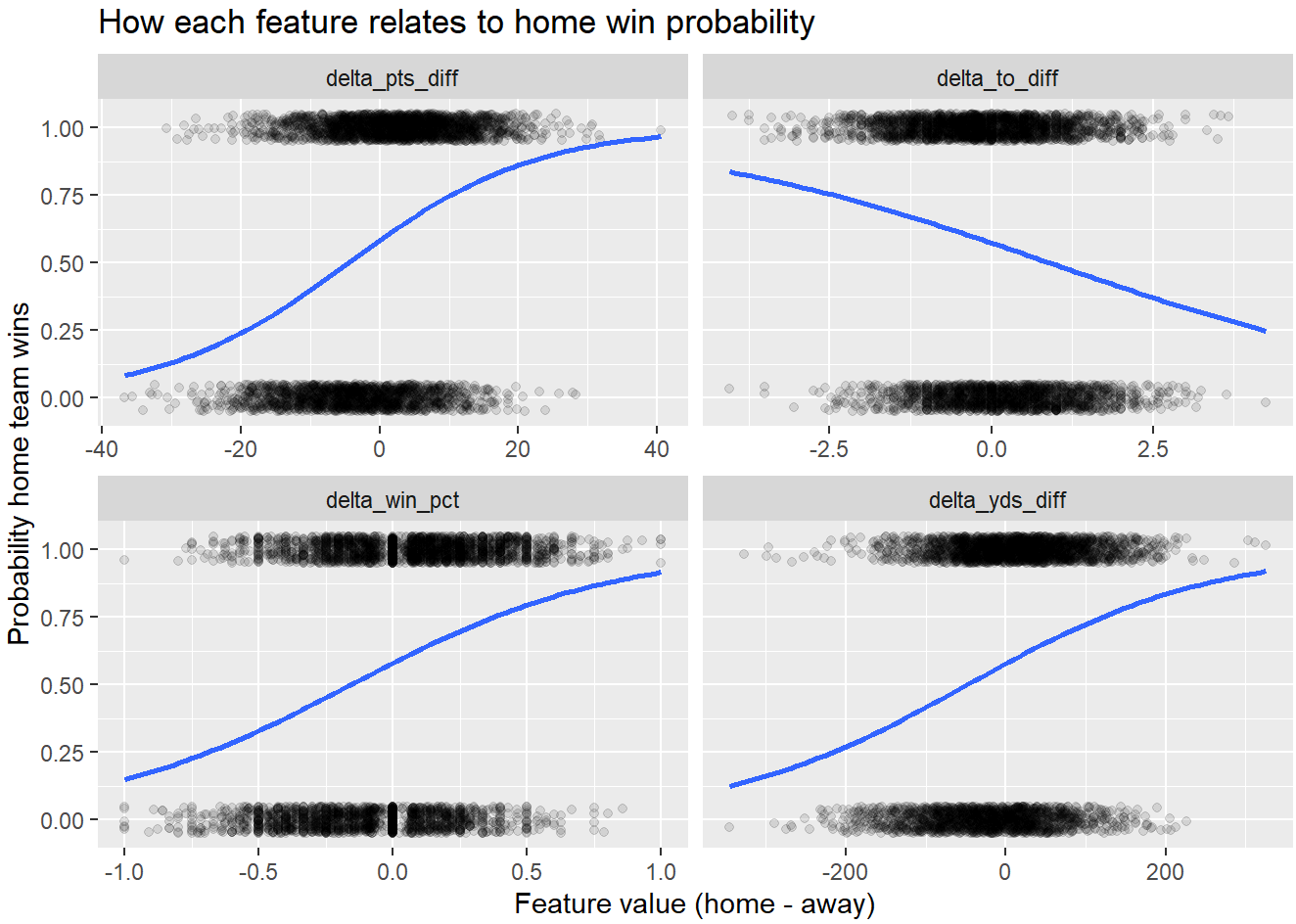
Attach home attendance to each game (`games_att`) - What this does: Filters to home-team rows and joins in weekly attendance from `attendance_clean`, producing `attendance_home`, then joins that onto `games_model`. - Output/data: `games_att` should contain everything in `games_model` plus `home_attendance` for the home team in that week. - What to watch for: If `home_attendance` has many `NA`s, it usually means the join keys don't match (team name formatting, or `year/week` types).

```
# one column for feature name, one for value
feature_long <- games_model %>%
  select(win_home, delta_win_pct, delta_pts_diff, delta_yds_diff, delta_to_diff) %>%
  pivot_longer(
    cols = c(delta_win_pct, delta_pts_diff, delta_yds_diff, delta_to_diff),
    names_to = "feature",
    values_to = "value"
  )

ggplot(feature_long, aes(x = value, y = win_home)) +
  geom_jitter(height = 0.05, width = 0, alpha = 0.1) +
  geom_smooth(
    method = "glm",
    method.args = list(family = "binomial"),
```

```
    se = FALSE
  ) +
  facet_wrap(~ feature, scales = "free_x") +
  labs(
    x = "Feature value (home - away)",
    y = "Probability home team wins",
    title = "How each feature relates to home win probability"
  )
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
coef_table_visuals <- feature_long %>%
  group_by(feature) %>%
  do({
    fit <- glm(win_home ~ value, data = ., family = binomial)
    as_tibble(summary(fit)$coefficients, rownames = "term") %>%
      rename(estimate = Estimate,
             std_error = `Std. Error`,
             stat = `z value`,
             p_value = `Pr(>|z|)`)
  }) %>%
  ungroup()
```

```
coef_table_visuals
```

```
# A tibble: 8 × 6
  feature        term        estimate std_error   stat  p_value
  <chr>          <chr>          <dbl>     <dbl>  <dbl>    <dbl>
1 delta_pts_diff (Intercept)  0.334    0.0339     9.85 6.80e-23
2 delta_pts_diff value        0.0744   0.00387   19.2  2.33e-82
3 delta_to_diff  (Intercept)  0.295    0.0324     9.10 9.13e-20
4 delta_to_diff  value       -0.330    0.0307   -10.8  5.72e-27
5 delta_win_pct  (Intercept)  0.326    0.0335     9.73 2.31e-22
6 delta_win_pct  value        2.06     0.118     17.5  1.04e-68
7 delta_yds_diff (Intercept)  0.317    0.0330     9.59 8.55e-22
8 delta_yds_diff value        0.00653  0.000441  14.8  1.17e-49
```

Visualize feature relationships with home win probability - What this code does: 1. Creates `feature_long` by pivoting the four "delta" features into a long format (one row per game-feature pair). 2. Plots `value` vs `win_home` with jitter and overlays a logistic regression smooth (`glm(..., family = binomial)`) to show the trend. - Data coming out: - `feature_long` has columns like `win_home`, `feature`, and `value`. - The plot should show whether larger feature values are associated with higher home-win probability. - Interpretation tip: Because the x-axis scale differs by feature (win pct vs yards vs turnovers), compare the direction of the trend more than the slope magnitude.

Predicting the Winner:

All four features show a strong and statistically significant relationship with home win probability, indicating that they meaningfully contribute to predicting the game winner rather than reflecting random noise.

The strongest predictor is delta_win_pct. Its large positive coefficient 2.0648 and high t-statistic 17.52 indicate that differences in team win percentage have the greatest impact on the chnace of a win. As the home team's win percentage advantage increases, the probability of a home win rises sharply, making this the most influential feature in the model.

Delta_pts_diff also shows a strong positive relationship with home win probability. The coefficient of 0.0744 and very large t-statistic 19.22 suggest that point differential is a reliable predictor. As the home team's point advantage grows, the probability of winning increases,

Delta_yds_diff has a smaller coefficient 0.00653 but remains highly statistically significant (t = 14.82). While the effect per yard is modest, yardage differences can be large in practice, and yardage advantage over a season can quickly increase leading to meaningful predictive signal.

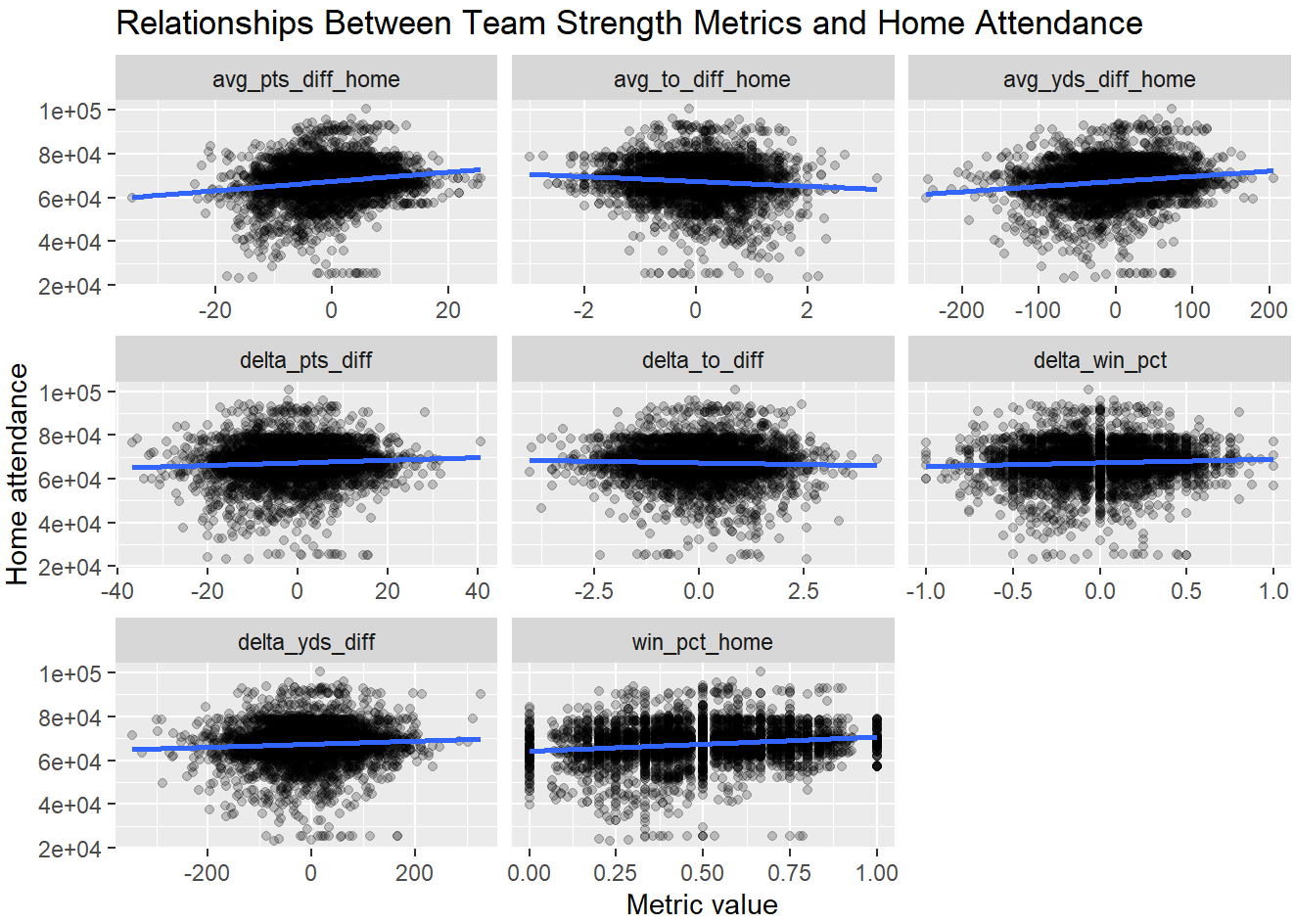
Delta_to_diff has a strong negative relationship with home win probability, with a coefficient of −0.3301 and a t-statistic of −10.75. This indicates that a more positive turn over differential negativity effects the teams chances of winning. Which makes sense you want to turn over the ball less then your opponents do

Overall, delta_win_pct and delta_pts_diff emerge as the most impactful predictors of the game winner, with delta_to_diff providing a strong negatiove relationship and delta_yds_diff offering additional predictive value.

```r
games_att_long <- games_att %>%
  select(
    home_attendance,
    win_pct_home,
    avg_pts_diff_home,
    avg_yds_diff_home,
    avg_to_diff_home,
    delta_win_pct,
    delta_pts_diff,
    delta_yds_diff,
    delta_to_diff
  ) %>%
  pivot_longer(
    -home_attendance,
    names_to = "metric",
    values_to = "value"
  )
games_att_long <- games_att_long |>
  dplyr::filter(is.finite(value), is.finite(home_attendance))

ggplot(games_att_long, aes(value, home_attendance)) +
  geom_point(alpha = 0.2) +
  geom_smooth(method = "lm", se = FALSE) +
  facet_wrap(~ metric, scales = "free_x") +
  labs(
    title = "Relationships Between Team Strength Metrics and Home Attendance",
    x = "Metric value",
    y = "Home attendance"
  )
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
coef_table_attendance <- games_att_long %>%
  group_by(metric) %>%
  do({
    fit <- lm(home_attendance ~ value, data = .)
    as_tibble(summary(fit)$coefficients, rownames = "term") %>%
      rename(estimate = Estimate,
             std_error = `Std. Error`,
             stat = `t value`,
             p_value = `Pr(>|t|)`)
  }) %>%
  ungroup()

coef_table_attendance
```

```
# A tibble: 16 × 6
   metric            term        estimate std_error   stat  p_value
   <chr>             <chr>          <dbl>     <dbl>  <dbl>    <dbl>
 1 avg_pts_diff_home (Intercept) 67442.      145.   465.   0
 2 avg_pts_diff_home value         213.       20.6   10.4  6.94e-25
 3 avg_to_diff_home  (Intercept) 67412.      146.   461.   0
 4 avg_to_diff_home  value       -1103.      189.    -5.85 5.26e- 9
 5 avg_yds_diff_home (Intercept) 67433.      145.   464.   0
 6 avg_yds_diff_home value          23.5       2.54   9.23 4.46e-20
```

```
 7 delta_pts_diff    (Intercept) 67430.      147.   459.   0
 8 delta_pts_diff    value          59.5      14.9    4.00 6.53e- 5
 9 delta_to_diff     (Intercept) 67412.      147.   459.   0
10 delta_to_diff     value        -292.      133.    -2.20 2.78e- 2
11 delta_win_pct     (Intercept) 67428.      147.   459.   0
12 delta_win_pct     value        1663.      469.     3.55 3.93e- 4
13 delta_yds_diff    (Intercept) 67426.      147.   459.   0
14 delta_yds_diff    value           7.04      1.81   3.90 9.77e- 5
15 win_pct_home      (Intercept) 64213.      351.   183.   0
16 win_pct_home      value        6471.      647.    10.0  2.86e-23
```

Relationship between attendance and team/"delta" metrics - What this code does: 1. Creates `games_att_long` by pivoting multiple metrics (home performance metrics and deltas) into long form while keeping `home_attendance`. 2. Produces a faceted scatter plot (attendance vs metric) to compare relationships across metrics. 3. Fits simple linear models `lm(home_attendance ~ value)` separately for each metric and returns a coefficient table. - Data coming out: - `games_att_long` has `home_attendance`, `metric`, and `value`. - `coef_table_attendance` summarizes how attendance changes with each metric (estimate, std error, t-stat, p-value). - Interpretation tip: Attendance is influenced by many factors not in this dataset (market size, stadium capacity, rivalries, weather). Treat these as associations, not causal effects.

Predicting Attendance:

Team strength metrics show statistically significant but relatively modest relationships with home attendance, and the scatterplots indicate substantial variability that is not explained by on-field performance alone. This suggests that attendance is influenced by many external factors beyond team quality.

Average home team strength metrics have the clearest association with attendance. Avg_pts_diff_home and avg_yds_diff_home both show positive and highly significant relationships, indicating that stronger season-long performance by the home team is associated with higher turnout. In contrast, avg_to_diff_home has a significant negative relationship showing that having less turn overs then your opponent contribute to game quality.

Game-level matchup differences are weaker predictors. Delta_pts_diff is positive but smaller in magnitude. This implies that fans respond more to the home team's overall performance across the season than to specific matchup advantages in a given game.

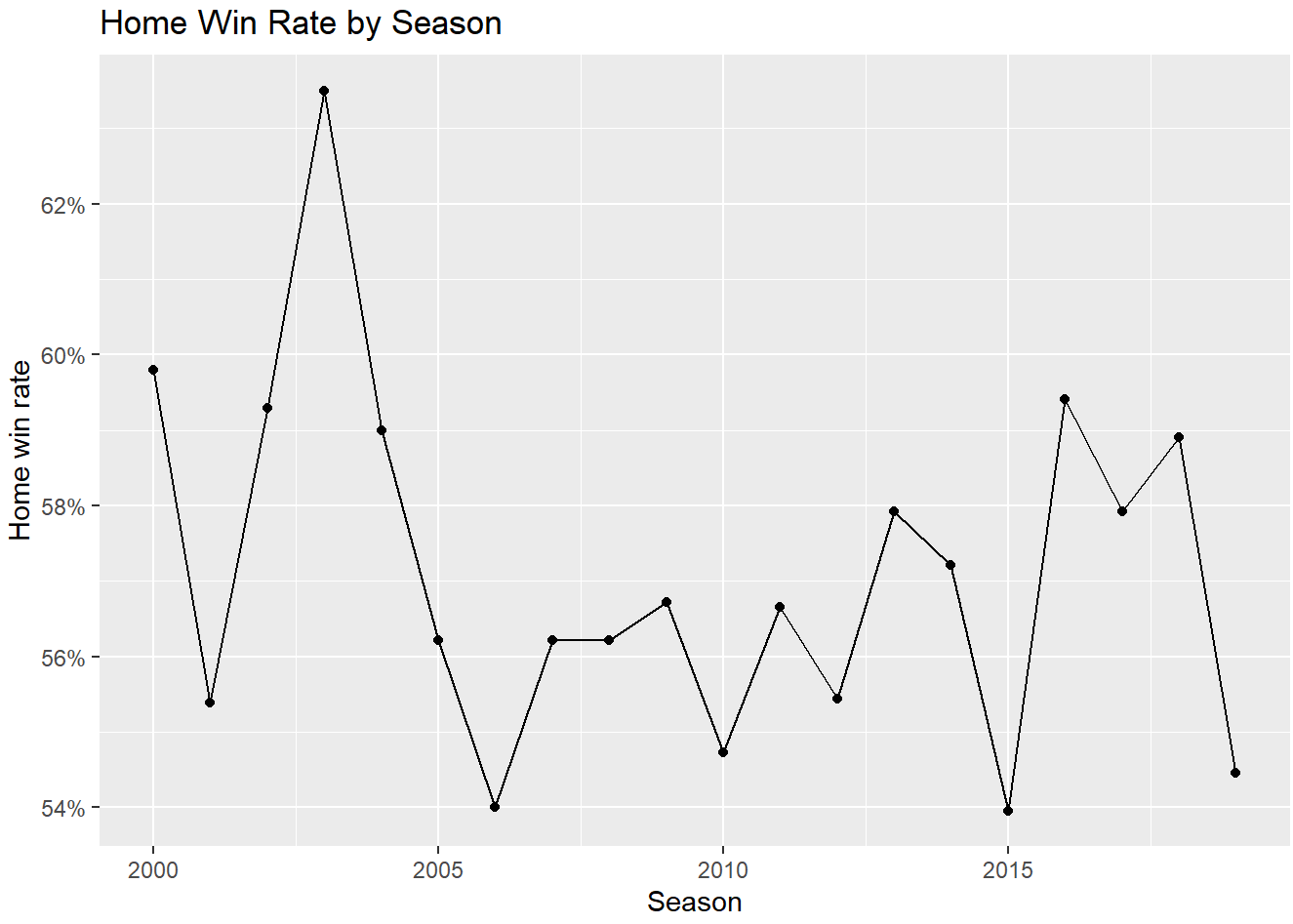
Overall, stronger home teams tend to draw larger crowds, but the wide spread in attendance values shows that team performance alone explains only part of attendance behavior.

```r
# Overall home win rate
games_model %>%
  summarise(
    n_games       = n(),
    home_win_rate = mean(win_home, na.rm = TRUE)
  )
```

```
# A tibble: 1 × 2
  n_games home_win_rate
    <int>         <dbl>
1    4011         0.571
```

```r
home_by_year <- games_model %>%
  group_by(year) %>%
  summarise(
    n_games       = n(),
    home_win_rate = mean(win_home, na.rm = TRUE)
  )


ggplot(home_by_year, aes(x = year, y = home_win_rate)) +
  geom_line() +
  geom_point() +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  labs(
    title = "Home Win Rate by Season",
    x = "Season",
    y = "Home win rate"
  )
```



Summarize and plot home-field advantage over time - What this code does: 1. Computes the overall home win rate across all games. 2. Computes `home_by_year`, a year-by-year table with number of games and mean home win rate. 3. Plots `home_win_rate` by season as a line chart. - Data coming out: - The printed `home_by_year` lets you see which seasons had stronger/weaker home advantage. - Interpretation tip: If you see dips in certain seasons, consider external context (rule changes, travel patterns, or unusual seasons) when writing your narrative.

Home team advantage:

If one were to blindly pick a winner in each game, the expected success rate would be 50%, equivalent to flipping a coin. However, as shown in the graph of home win rate by year, there is not a single season in which the home win rate falls below 54%. Across all 4,011 games in the dataset, the overall home win rate exceeds 57%. This consistency across a large sample size indicates that home-field advantage is a statistically meaningful factor. Therefore, using home status as a binary feature for predicting game outcomes is a statistically strong signal.