Solving Numerical Ties in Parallel Coordinate Plots

A Grammar of Graphics Approach Using ggpcp

Denise Bradford

Table of contents

0.1 Research Question

How can we effectively handle numerical ties in parallel coordinate plots so that we can still see individual observations across multiple axes without making the plot too busy and keeping it easy to understand when visualizing datasets with both categorical and continuous variables?

0.2 Background

When using traditional parallel coordinate plots (PCPs) with categorical variables or numerical ties, a major problem arises: multiple observations with the same value on an axis create overlapping lines that form an uninformative mesh. This makes it impossible to follow individual cases through the visualization. The early works of Inselberg (1985) and Wegman (1990) did not address this issue, and earlier attempts to combine categorical and numeric variables, like the categorical parallel coordinate plots of Pilhöfer and Unwin (2013), led to visualizations that didn't give any useful information.

The ggpcp package adds generalized parallel coordinate plots (GPCPs) that build on traditional PCPs by carefully handling ties on vertical axes so that they can easily handle both categorical and quantitative variables (VanderPlas et al. 2023).

0.3 Proposed Solutions

0.3.1 Tie-Breaking Methods

The research identifies and compares primary approaches to handling categorical variables and numerical ties:

Equi-spaced (Unordered): Spreads observations evenly within each categorical level without considering the ordering of adjacent axes. This provides better visual separation than jittering but does not optimize for readability.

Equi-spaced with Hierarchical Sorting (Recommended): The optimal solution that spaces observations evenly within categorical levels while ordering them hierarchically to minimize line crossings between axes. This method:

- Preserves the ability to follow individual observations across the plot
- Creates natural visualizations of marginal frequencies through proportional grouping
- Minimizes visual clutter through intelligent ordering
- Reduces cognitive load required to interpret the visualization

0.3.2 Technical Implementation: The `even_tie_spread` Algorithm

The even tie spread algorithm implements the equi-spaced hierarchical sorting approach through the following steps:

Step 1: Identify Tie Groups The algorithm first sorts observations by their value and a deterministic key (typically observation index), then identifies groups of observations sharing identical values. This double-sorting ensures consistent, reproducible spreading within tie groups.

Step 2: Calculate Spread Width The default spread width is set to 2% of the data range (configurable via the `frac` parameter):

```
width = frac × (max(x) - min(x))
```

This ensures that the spread is proportional to the scale of the data, making the visualization scale-invariant.

Step 3: Neighbor-Aware Capping (Optional) To prevent spread values from overlapping with neighboring distinct values, the algorithm can optionally cap the spread width based on the distance to nearest neighbors:

```
cap = min(gap_to_left/2, gap_to_right/2) × 1.999
actual_width = min(default_width, cap)
```

The factor of 1.999 (just under 2.0) ensures spread values approach but never quite reach the midpoint between neighboring values, maintaining clear separation.

Step 4: Generate Evenly Spaced Offsets For each tie group of size k, the algorithm generates k evenly spaced offsets centered at zero:

```
offsets = seq(-0.5, 0.5, length.out = k) × width
new_values = original_value + offsets
```

This creates symmetric spreading around the original value, with the range spanning from `original_value - width/2` to `original_value + width/2`.

Step 5: Deterministic Within-Group Ordering The use of a secondary key parameter ensures that observations within a tie group are spread in a consistent, deterministic order. This is critical for:

- Reproducibility across multiple runs
- Hierarchical sorting that considers adjacent axis values
- Maintaining visual continuity when the same data is plotted multiple times

Edge Case Handling: The algorithm includes robust handling for:

- **Missing values**: NA observations are preserved unchanged
- **Single observations**: Values with no ties remain at their original position
- **Degenerate ranges**: When data range is zero or infinite, no spreading occurs
- **Empty input**: Returns an empty vector immediately

0.3.3 3. Implementation Details

The hierarchical sorting approach is implemented in ggpcp through the `pcp_arrange(data, method, space)` function, which internally utilizes the `even_tie_spread` algorithm. Two primary methods are available:

- **"from-left"**: Tie breaks are determined hierarchically by variables' values from left to right
- **"from-right"**: Tie breaks are determined hierarchically by variables' values from right to left

The `space` parameter controls the amount of vertical axis used for spacing between categorical levels (default: 5%, equivalent to `frac = 0.05` in the spreading algorithm).

Algorithm Parameters:

- `x`: The numeric vector containing values to spread (including ties)
- `key`: A secondary sorting key (default: observation index) ensuring deterministic ordering within tie groups
- `width`: Explicit spread width (if NULL, calculated as `frac × range`)
- `frac`: Fraction of data range to use for spread width (default: 0.02 or 2%)
- `cap_to_neighbor`: Boolean flag to enable/disable neighbor-aware capping (default: TRUE)

0.3.4 Visual Design Principles

The solution leverages Gestalt principles of perception:

- **Good Continuation**: Minimizing line crossings allows the visual system to naturally follow lines across axes
- **Common Fate**: Observations with similar values “move together” through the vertical axes, creating perceptual grouping
- **External Cognition**: Additional computational processing reduces the cognitive burden on users

0.3.5 Algorithm Illustration

Consider a simple example with values `[1.0, 1.0, 1.0, 2.5, 2.5]` and a data range of 1.5:

Without Spreading:

```
Position:  1.0  1.0  1.0  2.5  2.5
           (3 overlapping) (2 overlapping)
```

With Even Tie Spreading (frac = 0.02):

```
width = 0.02 × 1.5 = 0.03

Tie group at 1.0 (k=3):
offsets = [-0.015, 0.0, 0.015]
new_values = [0.985, 1.000, 1.015]

Tie group at 2.5 (k=2):
offsets = [-0.015, 0.015]
new_values = [2.485, 2.515]

Position:  0.985  1.000  1.015  2.485  2.51
           (spread evenly)      (spread evenly)
```

With Neighbor-Aware Capping:

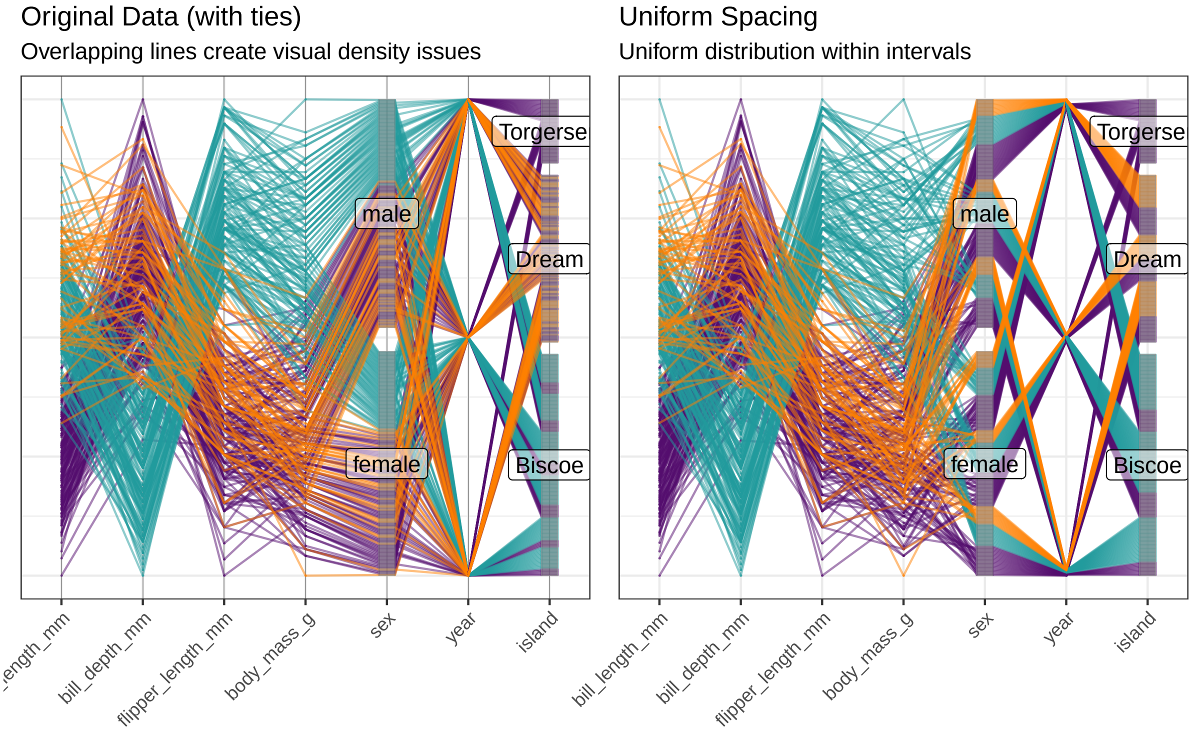
```
Gap between 1.0 and 2.5 = 1.5
Half-gap = 0.75
Cap = 0.75 × 1.999   1.499

Since default width (0.03) < cap (1.499), no adjustment needed
Spreading proceeds as above without crossing into neighbor territory
```

This illustration demonstrates how the algorithm maintains clear separation between distinct values while enabling individual observation tracking within tie groups.

0.4 Example Applications

0.4.1 Palmer Penguins Dataset

The Palmer Penguins dataset (Horst, Hill, and Gorman 2020) contains body measurements of three penguin species (Adelie, Chinstrap, and Gentoo) with both categorical variables (species, sex) and continuous measurements (bill length, bill depth, flipper length, body mass).


Original Data (with ties)

Overlapping lines create visual density issues

Uniform Spacing

Uniform distribution within intervals

Figure 1: ggpcp data using the Palmer Penguins Dataset

0.4.2 Iris Dataset

While not explicitly detailed in the provided materials, the Iris dataset represents a classic application where species (categorical) must be visualized alongside four continuous measurements (sepal length, sepal width, petal length, petal width).

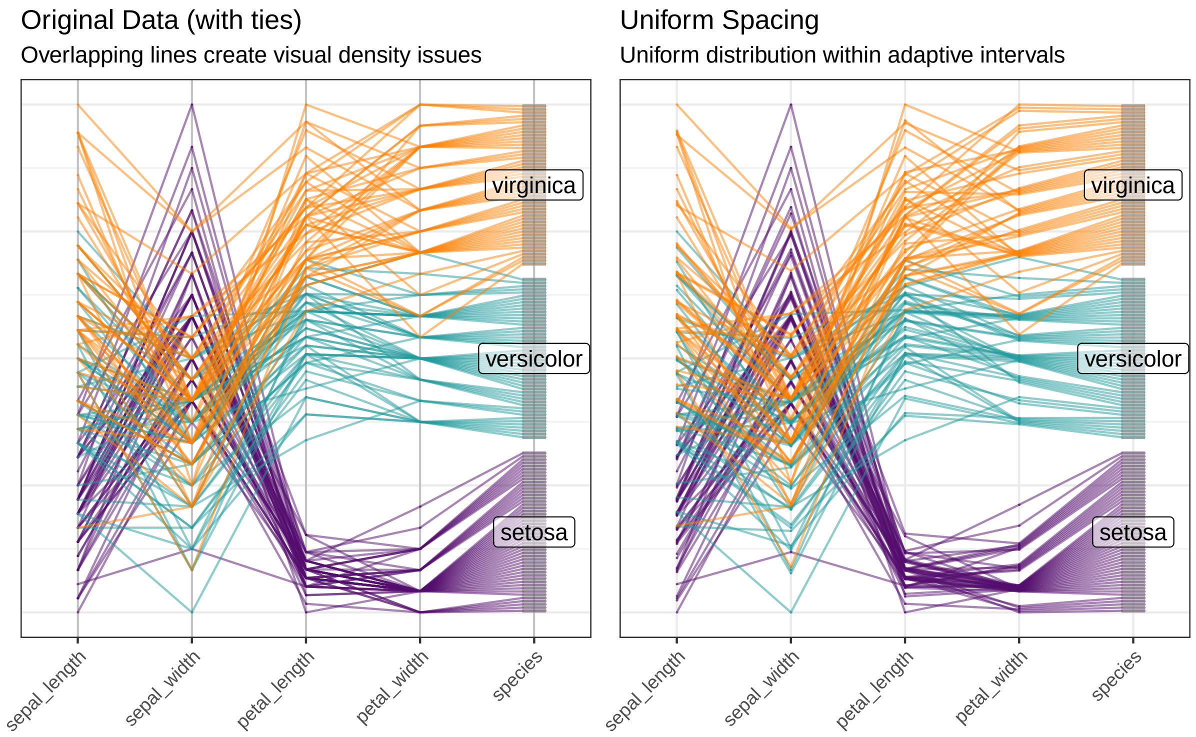


Figure 2: ggpcp data using the Iris Dataset

0.4.3 Asthma Dataset

The asthma dataset (Schonlau and Yang (2024)) demonstrates the power of GPCPs for health science applications with mixed variable types including:

- **Categorical variables**: Age group (child, adolescent, adult), gender (male, female)
- **Numerical variables**: Hospitalizations, comorbidities

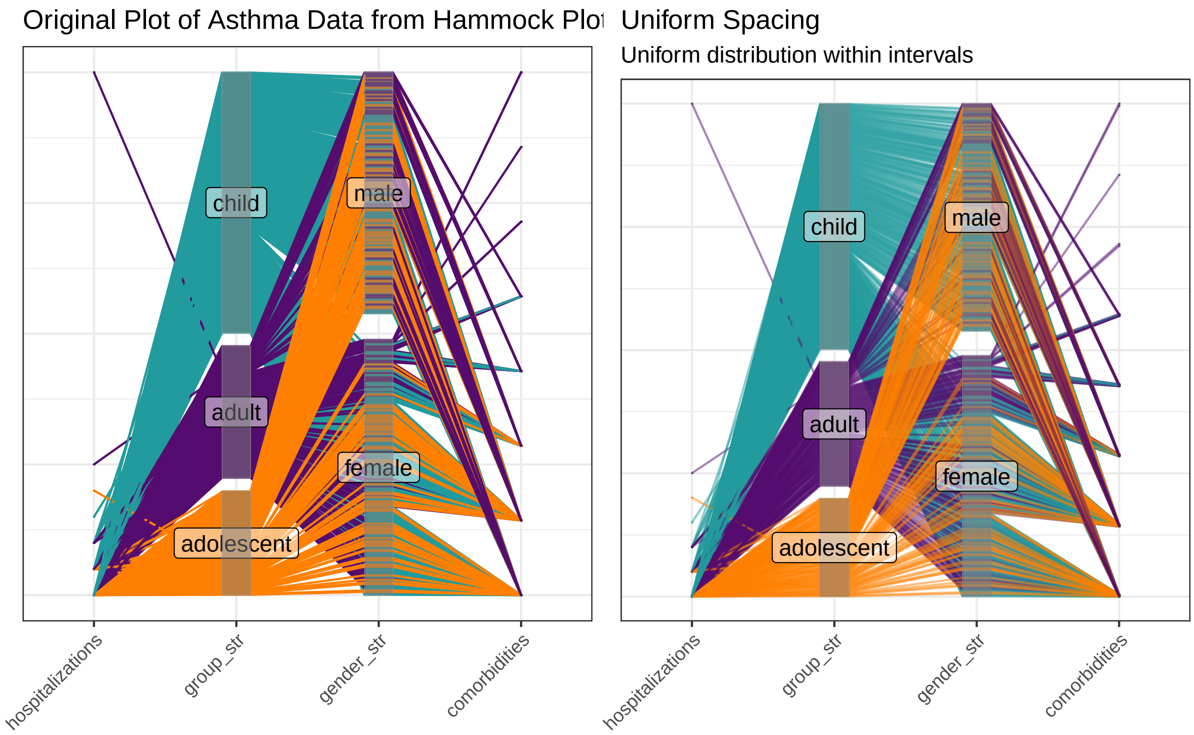


Figure 3: ggpcp data using the Asthma Data used in Schonlau and Yang (2024)

The visualization shows two approaches zoomed on numerical ties:

Left panel (Original Plot): Shows the traditional hammock plot representation where the dense convergence at numerical values creates triangular patterns that, while showing frequency, make individual observation tracking difficult.

Right panel (Uniform Spacing): Demonstrates the ggpcp approach with equi-spaced hierarchical sorting, where:

- Individual lines remain traceable across all axes
- Marginal frequencies are preserved through proportional box heights
- The transition from categorical age groups through numerical hospitalizations and comorbidities to categorical gender maintains visual continuity
- Patterns such as the concentration of observations at hospitalizations = 0 with varying comorbidities become more interpretable

0.5 Implementation Workflow

This function moves tied (identical) values in a numeric vector so that they can be seen in plots while keeping their approximate original position. It works by finding groups of values that are the same and then putting them evenly within a small range of the original value. The default spread width is 2% of the data range, but you can set a limit to keep it from overlapping with nearby distinct values. The function keeps the order of ties by using a key parameter for consistency, doesn't change NA values, and doesn't spread when there's only one observation at a value.

The ggpcp package implements a clear separation between data management and visualization:

```r
even_tie_spread <- function(x, key = seq_along(x), width = NULL, frac = 0.02) {
  stopifnot(is.numeric(x))

  n <- length(x)
  if (n == 0L) return(x)

  out <- x
  is_na <- is.na(x)

  # Calculate default width
  if (is.null(width)) {
    rng <- range(x[!is_na], na.rm = TRUE)
    width <- if (is.finite(diff(rng))) frac * diff(rng) else 0
  }

  if (width <= 0 || !is.finite(width)) return(out)

  # Sort by value, then key
  ord <- order(x, key, na.last = TRUE)

  # Identify tie groups
  run_vals <- x[ord]
  run_id <- cumsum(c(TRUE, diff(run_vals) != 0 | is.na(diff(run_vals))))

  # Spread each tie group
  for (group in unique(run_id)) {
    idx <- which(run_id == group)

    if (length(idx) <= 1 || is.na(run_vals[idx[1]])) next

    # Create evenly spaced offsets
    k <- length(idx)
    offsets <- seq(-0.5, 0.5, length.out = k) * width

    # Apply offsets
    out[ord[idx]] <- run_vals[idx[1]] + offsets
  }

  out
}

# Data management steps
data %>%
  dplyr::mutate(across(where(is.numeric),
                       ~ even_tie_spread(.x, width = 0.03,
                       key = dplyr::row_number()))) %>%
  pcp_select(variables) %>%
  pcp_scale(method = "uniminmax") %>%
  pcp_arrange(method = "from-left")

# Visualization
ggplot(pcp_data, aes_pcp()) +
  geom_pcp_axes() +
  geom_pcp(aes(colour = category)) +
  geom_pcp_labels() +
  theme_pcp()
```

This modular approach allows users to:

- Customize each step independently
- Reuse prepared data across multiple plot variations
- Extend functionality through custom functions
- Integrate with existing ggplot2 workflows

References

Horst, Allison Marie, Alison Presmanes Hill, and Kristen B. Gorman. 2020. *Palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data*. https://doi.org/10.32614/CRAN.package.palmerpenguins.

Inselberg, Alfred. 1985. "The Plane with Parallel Coordinates." *The Visual Computer* 1 (2): 69–91. https://doi.org/10.1007/BF01898350.

Pilhöfer, Alexander, and Antony Unwin. 2013. "New Approaches in Visualization of Categorical Data: R Package `extracat`." *Journal of Statistical Software* 53 (i07): 1–25. https://doi.org/10.18637/jss.v053.i07.

Schonlau, Matthias, and Rosie Yuyan Yang. 2024. "Hammock Plots: Visualizing Categorical Data Beyond Parallel Coordinates." *Journal of Computational and Graphical Statistics*.

VanderPlas, Susan, Yawei Ge, Antony Unwin, and Heike Hofmann. 2023. "Penguins Go Parallel: A Grammar of Graphics Framework for Generalized Parallel Coordinate Plots." *Journal of Computational and Graphical Statistics* 32 (4): 1405–20. https://doi.org/10.1080/10618600.2023.2181762.

Wegman, Edward J. 1990. "Hyperdimensional Data Analysis Using Parallel Coordinates." *Journal of the American Statistical Association* 85: 664–75.