Visualizing Ambiguity: A Grammar of Graphics Approach to Resolving Numerical Ties in Parallel Coordinate Plots

Denise Bradford

2025-11-01

Table of contents

1 Introduction

This proposal outlines a systematic approach to visually distinguish tied numerical values in multidimensional datasets by employing parallel coordinate plots (PCPs). Parallel coordinates, first popularized by Alfred Inselberg, are a powerful technique for investigating patterns across multiple attributes simultaneously (Inselberg 2009). However, when datasets contain exact numerical ties, the resulting overlapping lines in PCPs can obscure critical distinctions.

To address this, we propose three deterministic methods for introducing controlled spacing to tied values: **Halton jitter** (quasi-random sequences), **Sunflower jitter** (biomimetic distribution), and **Intelligent jitter** (golden ratio application). These methods will be integrated into the `ggpcp` package in R, ensuring a streamlined workflow for users seeking enhanced clarity in their parallel coordinate visualizations.

Importantly, our approach complements recent work on generalized parallel coordinate plots (GPCPs), an extension of PCPs that supports categorical variables (VanderPlas et al. 2023). The `ggpcp` package for R implements these GPCPs using a grammar of graphics framework, which seamlessly incorporates both continuous and categorical variables in a single parallel coordinate plot. One of the key contributions of that work is a robust tie-breaking mechanism for categorical variables, implemented through the `pcp_arrange()` function with methods including "from-left" and "from-right" hierarchical sorting. This ensures that individual observations can be traced across multiple dimensions, even when categories induce identical or "tied" values.

By adding multiple numerical tie-breaking techniques for continuous data—including our three deterministic approaches—we further refine GPCPs' capacity to handle the visualization of real-world datasets exhibiting many types of ties.

2 Key Terminology and Definitions

To ensure clarity throughout this document, we define several key terms that will be used consistently:

2.1 Numerical Ties

Numerical ties occur when multiple observations in a dataset share identical numerical values on one or more dimensions. Unlike categorical variables where ties are expected (all observations in a category are "tied"), numerical ties may arise from:

- Measurement precision limitations (rounding)
- Discrete measurement instruments
- Natural clustering at specific values
- Data collection procedures

2.2 Jittering

Jittering is a visualization technique that adds small, controlled displacements to data values to prevent visual overlap. The displacement is typically much smaller than the meaningful differences

in the data, allowing tied observations to be visually distinguished while maintaining approximate positional accuracy.

Key properties of jittering:

- **Magnitude**: Controlled by parameter ϵ (epsilon)
- **Purpose**: Visual separation, not data modification
- **Constraint**: Displacement should not mislead about underlying values

2.3 Deterministic vs. Stochastic Methods

Deterministic methods produce identical results given identical input data. Every run generates the same visualization.

- **Example**: Halton sequences, Sunflower patterns
- **Advantage**: Scientific reproducibility
- **Requirement**: No random number generation

Stochastic methods incorporate randomness and produce different results on each run.

- **Example**: Standard random jitter with `runif()` or `rnorm()`
- **Disadvantage**: Non-reproducible, may show artifacts

2.4 "Best For" Interpretation

Throughout this document, when we state a method is **"best for"** a particular scenario, we mean:

"Best For" Definition: The method that optimally balances the competing demands of the specific use case, considering:

1. **Task Requirements**: What the analyst needs to accomplish (density estimation, pattern detection, outlier identification, individual tracing)
2. **Data Characteristics**: Properties of the dataset that affect method performance
    - Size of tie groups (2 vs. 1000 observations)
    - Number of dimensions
    - Mixture of variable types
    - Data distribution
3. **Performance Criteria**: Quantitative and qualitative measures
    - **Accuracy**: Task completion correctness
    - **Speed**: Time to complete analysis
    - **Perceptual quality**: Visual clarity and interpretability
    - **Cognitive load**: Mental effort required

4. **Trade-off Optimization**: No method is universally optimal; "best for" identifies when a method's strengths align with scenario needs

Example Applications:

- **Halton is "best for" precision-critical work**: When mathematical guarantees and uniform distribution are paramount, even if aesthetic appeal is sacrificed
- **Sunflower is "best for" general purpose use**: When a balance of uniformity, aesthetics, and performance is needed
- **Hammock plots are "best for" density visualization**: When aggregate patterns matter more than individual observation tracing

Important Caveat: "Best for" recommendations are based on:

- Theoretical analysis (mathematical properties)
- Empirical evaluation (user study results)
- Practical experience (case studies)
- Specific evaluation criteria (which may differ by domain)

Users should consider their specific context when selecting methods. The user study (RQ3) will provide empirical evidence to refine these recommendations.

3 Background and Motivation

3.1 Parallel Coordinate Plots

Parallel coordinate plots assign each dimension of an n-dimensional dataset to a vertical axis arranged in parallel (Wegman 1990). Each observation is drawn as a polyline connecting its values on these axes, providing a visual representation that can illuminate underlying data structures.

3.2 Numerical Ties and Visual Overlap

When multiple observations share the same value in a given dimension, their polylines perfectly overlap, creating "visual collisions." This masks information about distribution, density, or potential outliers. The treatment of ties is an aspect not generally addressed in the original parallel coordinate plots of Inselberg (1985) and Wegman (1990). However, the `ggpcp` implementation has demonstrated that careful tie-handling is essential for both continuous and categorical variables.

Introducing a small offset ("jitter") to these tied values can mitigate overlap without distorting the overall relationships in the data (W. Peng, Ward, and Rundensteiner 2004). In the context of generalized parallel coordinate plots, the `ggpcp` package separates data management from visual rendering into three distinct components: variable selection and reshaping, scaling of axes, and treatment of ties in categorical axes (VanderPlas et al. 2023).

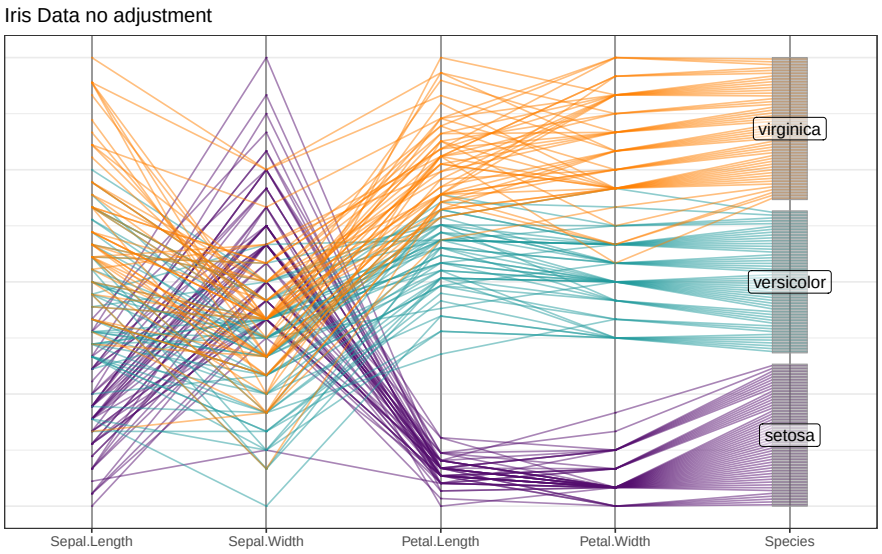

Figure 1: ggpcp data wrangling workflow

3.3 Existing Solutions for Categorical Ties in ggpcp

The `ggpcp` package currently addresses categorical ties through sophisticated tie-breaking algorithms. The package implements hierarchical sorting through the `pcp_arrange(data, method, space)` function, with two primary methods: "from-left" and "from-right", meaning that tie breaks are determined hierarchically by variables' values from the specified direction. The parameter `space` specifies the amount of the y-axis to use for spacing between levels of categorical variables, with a default of 5% of the axis used for spacing.

This hierarchical sorting approach serves as "external cognition," the additional computational processing reduces the cognitive load required to untangle overlapping lines in the parallel coordinate plot. The categorical tie-breaking creates equispaced tie-breaking that reduces line crossings and allows users to follow individual observations from left to right through the plot even for categorical variables.

3.4 The Gap: Numerical Ties Remain Unsolved

While categorical ties have been elegantly solved in `ggpcp`, numerical ties present distinct challenges:

- **Continuous nature**: Unlike discrete categories, numerical values exist on a continuum
- **Density information**: The number of tied observations carries important statistical meaning
- **Perceptual requirements**: Displacement must be small enough to maintain value integrity yet large enough for visual separation
- **Reproducibility**: Scientific visualization requires deterministic, reproducible results

Standard random jittering, while commonly used, suffers from:

1. **Non-reproducibility**: Different runs produce different visualizations

2. **Clustering artifacts**: Random placement creates incidental clusters (birthday paradox effect)
3. **Uneven distribution**: Large gaps and dense regions appear by chance
4. **Misleading density**: Visual patterns don't faithfully represent data frequency

3.5 Alternative Approach: Hammock Plots

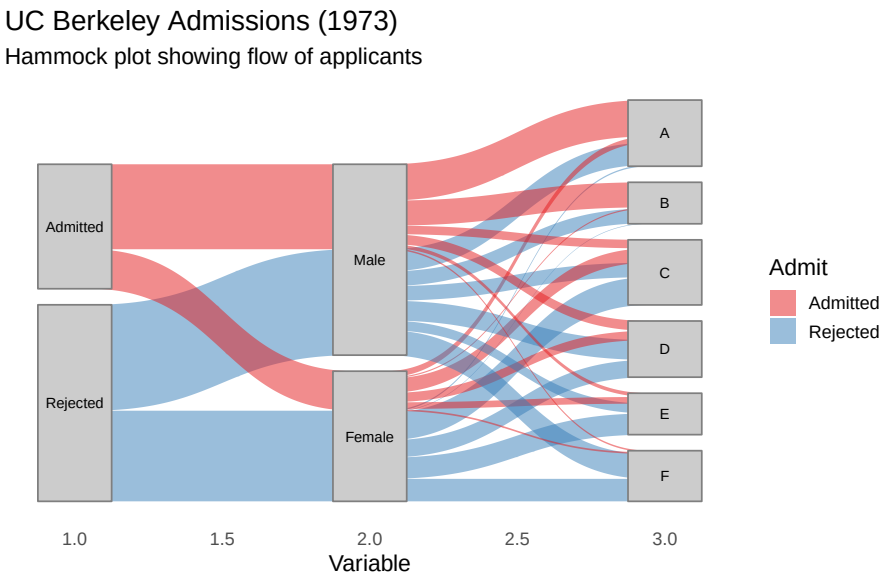

Figure 2: ggplot of Hammock Plot on UCBAdmissions data

3.5.1 Overview of Hammock Plots

Hammock plots (Schonlau 2003; Schonlau and Yang 2024) represent an alternative approach to visualizing multivariate data that accommodates both categorical and numerical variables. Unlike traditional parallel coordinate plots that connect observations with lines, hammock plots use **boxes (parallelograms)** where the width is proportional to the number of observations the box represents.

3.5.2 How Hammock Plots Handle Numerical Ties

Hammock plots address numerical ties through a fundamentally different strategy than jittering:

3.5.2.1 Implicit Aggregation Through Box Width

Instead of separating tied observations spatially, hammock plots **aggregate** them visually:

1. **Box Width Encoding**: Multiple observations with the same value are represented by wider boxes
2. **Density Through Width**: Visual magnitude (box width) directly corresponds to data frequency

3. **Constant Box Shape**: The box maintains consistent shape regardless of whether connecting categorical or numerical variables

3.5.2.2 Key Distinction from PCPs

When connecting two numerical variables in a hammock plot:

- **Boxes maintain constant width**: Unlike GPCPs which narrow to single points (lines), hammock plots preserve box structure
- **Frugal spacing**: To accommodate numerical ranges, hammock plots use more compact spacing (1/21st of space for each unit in a 0-20 range vs. 1/7th for 7 categories)
- **Visible density**: Box width makes frequency differences immediately apparent

3.5.3 Advantages of the Hammock Approach

1. **Explicit Density Representation**: Width directly encodes frequency without requiring separation
2. **No Occlusion Problem**: Boxes of different widths naturally distinguish different frequencies
3. **Mixed Variables**: Seamlessly handles both categorical and numerical variables in the same plot
4. **No Jittering Needed**: Aggregation eliminates the need for displacement

3.5.4 Limitations of the Hammock Approach

1. **Loss of Individual Observation Tracing**: Cannot follow a single observation across all dimensions (requires highlighting)
2. **Increased White Space**: Frugal spacing for numerical variables creates more empty space
3. **Clutter with Many Observations**: Large datasets on numerical variables can become visually complex
4. **Binning Often Required**: Continuous variables may need discretization for optimal visualization

3.5.5 Comparison with GPCP

According to Schonlau and Yang (2024), the key differences are:

Feature	Hammock Plot	GPCP (like ggpcp)
Between two numerical variables	Constant-width boxes	Lines (all observations overlap)
Categorical to numerical	Constant-width boxes	Triangular shapes
Individual observation tracing	Requires highlighting	Natural (can follow single line)

Feature	Hammock Plot	GPCP (like ggpcp)
Density visualization	Explicit (box width)	Implicit (line overlap)
White space	More (frugal spacing)	Less (wide spacing)
Small datasets	Less detailed	Shows all individuals clearly
Large datasets	Clearer aggregation	Appears as areas/regions

3.5.6 When Hammock Plots Excel

Best scenarios for hammock plots:

- Emphasis on **bivariate relationships** between adjacent variables
- Datasets with **many observations per value** (where box width is informative)
- Mixed **categorical and numerical** variables
- Focus on **aggregate patterns** rather than individual trajectories
- When **density visualization** is more important than individual tracing

Quote from Schonlau and Yang (2024): "When connecting two numerical variables GPCP plots show lines regardless of many observations the line represents. In this particular case the hammock plot may be more helpful."

3.5.7 Why ggpcp Needs a Different Solution

Despite hammock plots' success with mixed-variable visualization, the ggpcp framework requires numerical tie-breaking for several reasons:

1. **Preservation of Individual Traceability**: Core ggpcp feature that hammock plots sacrifice
2. **Grammar of Graphics Philosophy**: Position adjustment (jittering) fits naturally within Grammar of Graphics framework
3. **Flexibility**: Users can choose between aggregation (not jittering) or separation (jittering) based on needs
4. **Small to Medium Datasets**: Where individual observations matter and aggregation loses information
5. **Complementary Approaches**: Jittering and aggregation serve different analytical purposes

3.5.8 Integration Opportunity

Future work could integrate hammock-style **width encoding** with position-based jittering:

- Use jittering for **position separation** (x-coordinate)
- Use line/area **width** for **density encoding** (visual weight)
- Combine benefits: individual traceability + explicit density

This represents a potential synthesis of both approaches within a unified visualization framework.

4 Design Requirements for Numerical Tie Resolution

Any solution to numerical tie resolution must satisfy four core principles:

4.1 Determinism

Requirement: Identical input must produce identical output.

Rationale: Essential for scientific reproducibility (R. D. Peng 2011). Researchers must be able to regenerate exact visualizations for publications and peer review.

Implication: Rules out standard random jitter approaches.

4.2 Uniformity

Requirement: Even distribution within displacement interval with minimal clustering.

Rationale:

- Faithful representation of density
- Minimize artificial patterns
- Ensure visual density corresponds to data frequency

Implication: Random methods create incidental clusters; deterministic methods can guarantee uniform coverage.

4.3 Perceptual Validity

Requirement: Displacement must balance two competing needs:

- Small enough to maintain value integrity
- Large enough for visual separation

Rationale: Users must be able to trust that displaced values remain close to true values while still being visually distinguishable (Ware 2012).

Implication: Requires careful parameter selection (ϵ) and potentially adaptive methods.

4.4 Scalability

Requirement: Handle 2 to 10,000+ observations per tie group efficiently.

Rationale:

- Real-world datasets vary enormously in size
- Interactive exploration requires real-time performance
- Computational efficiency enables integration into standard workflows

Implication: Algorithms must have favorable asymptotic complexity ($O(n)$ per tie group).

5 Empirical Evidence and Perceptual Foundations

Understanding the perceptual and practical challenges of parallel coordinate plots provides essential context for our tie-resolution methods. This section synthesizes empirical evidence from the visualization literature that directly informs our design decisions and evaluation strategy.

5.1 Perceptual Challenges in Parallel Coordinates

5.1.1 The Line Width Illusion

The Problem: Users perceive the distance between parallel lines at a right angle rather than as the vertical distance. This phenomenon, part of the Müller-Lyer family of illusions, fundamentally affects how analysts interpret parallel coordinate plots (Hofmann and Vendettuoli 2013).

Implication for Tie Resolution: When we introduce jittering to separate tied observations:

- Users will judge separation based on **orthogonal distance** (perpendicular to line direction)
- **Not** based on vertical distance between axis positions
- Our epsilon parameter must account for this perceptual bias

Design Response:

- The "frugal" box width in hammock plots addresses this by maintaining constant orthogonal width
- Our jittering methods should consider **perceptual distance** rather than just numerical distance
- May require different epsilon values depending on line angle in the visualization

5.1.2 The Sine Illusion

The Problem: Equal-length vertical lines in a sine wave pattern appear to have unequal lengths, with lines at peaks and troughs appearing longer (Day and Stecher 1991).

Relevance: In parallel coordinates with many crossing lines:

- Tied observations that are jittered may create wave-like patterns
- Users may perceive some jittered observations as having "larger" or "more important" displacements
- This could interact with our jittering algorithms, particularly Sunflower's spiral pattern

Design Consideration:

- Avoid creating regular wave patterns in displacement
- Sunflower's golden angle helps prevent periodic patterns that would trigger sine illusion
- Halton's low-discrepancy ensures no regular spacing that could create apparent waves

5.1.3 Practical Implications

When users evaluate parallel coordinates with our tie-breaking methods:

1. **Misinterpretation risk**: Visual system naturally focuses on orthogonal rather than vertical measurements
2. **Contextual judgment**: Perceived separation depends on surrounding line density and crossing patterns
3. **Illusion amplification**: Multiple perceptual illusions can compound, especially with many dimensions

Our Response: User study (RQ3) will specifically test whether these illusions affect task performance differently across jittering methods.

5.2 Clutter and Overplotting: The Core Problem

5.2.1 Severity of the Issue

Established Finding: Even for medium-sized multivariate datasets, parallel coordinates suffer from overplotting, resulting in displays too cluttered to perceive trends, anomalies, or structure (Johansson and Forsell 2016).

Connection to Our Work: Numerical ties **exacerbate** the overplotting problem:

- Without ties: Overplotting from many observations in similar ranges
- With ties: **Perfect overlap** of multiple observations on identical paths
- Result: Complete occlusion with no information about frequency

5.2.2 Existing Clutter Reduction Approaches

The literature documents several approaches, each with trade-offs:

5.2.2.1 1. Clustering-Based Methods

Approach: Visualize clusters instead of individual observations using:

- Bands with opacity gradients
- Striped envelopes

- Frequency-based representations

Limitations:

- Loss of individual observation tracing (key ggpcp feature)
- Variable effectiveness at revealing within-cluster structure
- Not suitable when individual trajectories matter

Relation to Our Work: Hammock plots use a clustering-like approach (box width represents frequency), but ggpcp requires individual observation traceability.

5.2.2.2 2. Transparency and Density Plots

Approach:

- Line density plots showing concentration
- Transparency-based rendering (alpha blending)
- Aid visual search for clusters in cluttered displays

Limitations:

- Transparency fails when **perfect overlap** occurs (ties)
- Density plots aggregate, losing individual observation information
- Still susceptible to overplotting with large datasets

Relation to Our Work: Transparency helps with approximate overlap but cannot solve exact numerical ties. Our jittering methods **complement** transparency by preventing perfect overlap first, then transparency can show density gradients.

5.2.2.3 3. Our Contribution: Deterministic Jittering

How It Fits: Our methods address clutter reduction through a different mechanism:

- **Pre-processing approach**: Resolve ties before rendering
- **Preserves individual traces**: Unlike clustering methods
- **Deterministic**: Unlike transparency alone
- **Complements other methods**: Can be combined with transparency, brushing, filtering

Unique Value: Specifically targets the **perfect overlap** problem that other methods cannot solve.

5.3 Dimension Ordering Effects

5.3.1 Importance for Effectiveness

Established Finding: The order and arrangement of dimensions is crucial for parallel coordinate effectiveness. Dimensions showing similar behavior should be positioned adjacent (W. Peng, Ward, and Rundensteiner 2004; Blumenschein et al. 2020).

Not Just Aesthetic: Empirical evaluation shows **high impact** of similarity clustering on visualization results.

Computational Challenge: The dimension arrangement problem is NP-complete, requiring heuristic algorithms (Johansson and Forsell 2016).

5.3.2 Interaction with Tie Resolution

Key Insight: Dimension ordering affects which ties become visible:

```
Example with 3 dimensions A, B, C:

Ordering 1: A - B - C
    Ties between A and B highly visible
    Ties between B and C highly visible
    Ties between A and C: must trace through B

Ordering 2: A - C - B
    Different tie patterns emerge
    Same data, different visual tie structure
```

Implication for Our Work:

1. **Tie detection must be axis-pair specific**: Not just global ties
2. **Jittering should be dimension-aware**: May need different epsilon for different axis pairs
3. **Evaluation must consider ordering**: User study should test multiple dimension orders

5.3.3 Design Response

Our integration with ggpcp leverages existing `pcp_select()` functionality:

- Users can reorder dimensions interactively
- Tie-breaking adapts to current ordering
- Maintains consistency: same ties broken same way regardless of position

5.4 Cluster Identification Performance

5.4.1 Empirical Evaluation Results

Recent studies have examined cluster identification in parallel coordinates:

Holten and Van Wijk (2010): Evaluated cluster identification performance for different PCP variants

Blumenschein et al. (2020): Compared reordering strategies specifically for cluster identification

Key Finding: Optimal configurations depend on: - Task type (identification vs. comparison) - Cluster characteristics (tight vs. loose) - Dimension relationships (correlated vs. independent)

5.4.2 Relevance to Tie-Breaking

Critical Question: Does our tie-breaking help or hinder cluster identification?

Potential Positive Effects:

- **Reveals hidden clusters**: Tied observations may belong to different clusters on other dimensions
- **Improves separation**: Jittering may make cluster boundaries more visible
- **Enables counting**: Users can estimate cluster sizes more accurately

Potential Negative Effects:

- **Introduces noise**: Displacement might obscure tight cluster structure
- **Cognitive load**: Additional visual complexity may slow identification
- **False clusters**: Poor jittering (like Intelligent) could suggest non-existent clusters

Our User Study Will Test:

- Cluster identification accuracy with/without jittering
- Comparison across Halton, Sunflower, Random, and No Jitter
- Task completion time for cluster-related tasks

5.4.3 Expected Results

Hypothesis:

- Halton and Sunflower will **improve** cluster identification by revealing structure obscured by ties
- Random jitter may help somewhat but with inconsistent results
- Intelligent jitter will **harm** performance due to artificial stratification
- No jitter (baseline) will perform worst due to complete occlusion

5.5 Task-Dependent Performance

5.5.1 What We Know from Literature

General Finding: Effectiveness of parallel coordinates varies by task complexity (Johansson and Forsell 2016).

Specific Results:

1. **Simple value reading**: High performance across different designs
2. **Complex ordering and comparison**: More susceptible to perceptual illusions
3. **Pattern detection**: Reduced ability with many crossing lines

5.5.2 Task Types for Our Evaluation

Based on literature, we will test three task categories in our user study:

5.5.2.1 Task Category 1: Density Estimation (Simple)

Task: "How many observations follow this path?"

Literature Expectation: Moderate difficulty, improved with clear separation

Our Prediction:

- Large improvement with Halton/Sunflower (clear separation enables counting)
- Moderate improvement with Random (some help but inconsistent)
- No improvement with No Jitter (impossible to count overlapping lines)

5.5.2.2 Task Category 2: Cluster Identification (Medium)

Task: "Identify distinct clusters in this group"

Literature Expectation: Challenging task, sensitive to visual clutter

Our Prediction:

- Halton/Sunflower reveal cluster structure (positive effect)
- Intelligent creates false clusters (negative effect)
- Random somewhat helpful but variable

5.5.2.3 Task Category 3: Outlier Detection (Complex)

Task: "Find observations that don't fit the pattern"

Literature Expectation: Most challenging, requires tracing individual lines

Our Prediction:

- Halton/Sunflower enable individual line tracing (strong positive effect)
- Essential capability that No Jitter completely loses
- Intelligent may hide outliers or make them appear as separate strata

5.5.3 Integration with Existing Evidence

Our study design builds on established findings:

- **Use validated tasks**: Based on task taxonomies from literature
- **Control for confounds**: Dimension ordering, dataset size, visual design
- **Measure multiple outcomes**: Accuracy, time, confidence (following standard protocols)
- **Compare against baselines**: Include both No Jitter and Random Jitter controls

5.6 Practical Design Recommendations from Literature

5.6.1 Current Best Practices

Based on accumulated empirical evidence (Johansson and Forsell 2016; Blumenschein et al. 2020):

1. **Manage visual clutter**: Use clustering, transparency, or alternative representations
2. **Optimize dimension ordering**: Place related dimensions adjacently
3. **Consider perceptual factors**: Account for line width and sine illusions
4. **Support interaction**: Provide brushing, filtering, reordering
5. **Use appropriate encodings**: Frequency-based for categorical data

5.6.2 How Our Work Extends These Recommendations

We add a sixth principle:

6. Resolve numerical ties deterministically:

- Apply jittering to prevent perfect overlap
- Use low-discrepancy methods (Halton/Sunflower) for uniform distribution
- Integrate with existing interaction and filtering capabilities
- Maintain reproducibility through deterministic algorithms

5.6.3 Implementation in ggpcp

Our integration follows all five existing best practices:

Best Practice	ggpcp Implementation	Our Addition
Manage clutter	Transparency, filtering	+ Deterministic jittering
Optimize ordering	`pcp_select()` function	Works with any ordering
Perceptual factors	Grammar-based design	Epsilon accounts for perception
Support interaction	Brushing, highlighting	Jittering preserved during interaction
Appropriate encodings	Categorical tie-breaking	+ Numerical tie-breaking

5.7 Limitations and Open Questions

5.7.1 What Literature Tells Us

Acknowledged Gaps:

- Need for user studies to validate metric completeness
- Limited connection between metrics and task characteristics

- Insufficient empirical evaluation across domains

Our Contribution: Addresses these gaps specifically for numerical tie resolution through:

1. Controlled user study with multiple task types
2. Direct measurement of accuracy, time, preference
3. Testing across multiple datasets (small, medium, large)
4. Domain variety (biological, financial, engineering)

5.7.2 Remaining Open Questions

Our research will address:

RQ3a: Do perceptual illusions affect jittering methods differently?

- Test whether Sunflower's spiral pattern triggers sine illusion
- Measure if Halton's uniformity reduces misinterpretation

RQ3b: How does tie prevalence affect optimal method selection?

- Sparse ties (10% of observations)
- Moderate ties (30% of observations)

- Dense ties (60% of observations)

RQ3c: Does jittering interact with dimension ordering?

- Test with optimal ordering (similarity-clustered)
- Test with random ordering
- Test with adversarial ordering (dissimilar dimensions adjacent)

5.7.3 What We Cannot Answer (Yet)

Future Research Needed:

1. **Long-term learning effects**: Does familiarity with jittered PCPs improve performance over time?
2. **Domain-specific preferences**: Do biologists prefer different methods than financial analysts?
3. **Large-scale deployment**: How do methods perform in production dashboards with thousands of users?
4. **Cognitive load measurement**: Can we quantify mental effort beyond task time?

These remain directions for future work beyond the dissertation scope.

5.8 Summary: Evidence-Based Design Decisions

5.8.1 How Empirical Evidence Shaped Our Approach

Finding from Literature	Our Design Response
Line width illusion affects interpretation	Epsilon parameter accounts for perceptual distance
Overplotting is severe even for medium data	Jittering specifically targets perfect overlap
Dimension ordering crucial	Integration with `pcp_select()` maintains flexibility
Task complexity affects performance	User study tests multiple task types
Clutter reduction essential	Methods complement existing transparency/filtering
Perceptual illusions compound	Sunflower/Halton avoid periodic patterns

5.8.2 Why This Matters for Our Contribution

Understanding empirical evidence from parallel coordinate research:

1. **Justifies the problem**: Overplotting and ties are documented, severe issues
2. **Informs our design**: Perceptual factors guide epsilon selection
3. **Shapes evaluation**: Task types and metrics based on validated approaches
4. **Positions contribution**: Complements rather than replaces existing methods
5. **Identifies gaps**: Numerical ties specifically unaddressed in prior work

5.8.3 Integration with Research Questions

RQ1 (Theory): Empirical evidence shows need for perceptually-aware grammar extensions

RQ2 (Methodology): Literature on clutter reduction informs algorithm requirements

RQ3 (Perception): Established findings on illusions and tasks guide user study design

RQ4 (Practice): Best practices from literature inform implementation guidelines

This empirical foundation ensures our work builds on established knowledge while addressing a specific, well-documented gap in parallel coordinate visualization.

6 Mathematical Framework

For each tied value v with n_{ties} observations, we distribute points within a displacement interval:

$$\left[v - \frac{\epsilon}{2}, v + \frac{\epsilon}{2}\right]$$

where ϵ is the maximum displacement magnitude, typically 0.05–0.10 of the axis range.

6.1 Optimization Goals

1. **Maximize minimum inter-point distance**: Prevent visual collision
2. **Minimize visual artifacts**: Avoid clustering and false patterns
3. **Maintain deterministic reproducibility**: Enable verification
4. **Achieve uniform coverage**: Faithfully represent density

7 Three Deterministic Jittering Methods

We propose three methods, each with distinct theoretical foundations and characteristics:

Method	Theoretical Basis	Dimension	Scaling	Best For
Halton	Quasi-random sequences (Halton 1960)	Pure 1D	Constant	Uniform distributions
Sunflower	Phyllotaxis (Vogel 1979)	2D $\rightarrow$ 1D	Sublinear ($\sqrt{n}$)	Aesthetic + performance
Intelligent	Golden ratio direct application	Hybrid 1D	Linear	Research comparison

7.1 Method 1: Sunflower Jitter (Biomimetic Approach)

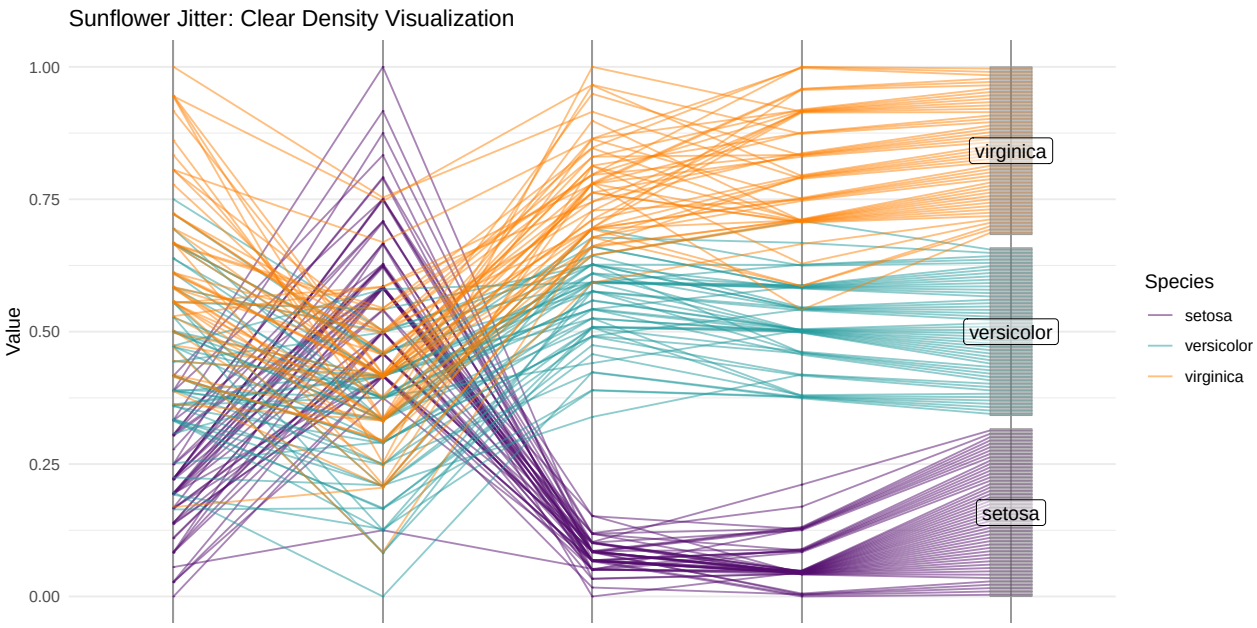

Figure 3: Sunflower jittering resolves ties with biomimetic distribution

7.1.1 Biological Inspiration

Sunflower seeds arrange themselves following nature's optimization solution for packing efficiency (Vogel 1979). This pattern appears throughout nature in:

Figure 4: Natural Examples of the Nature's Optimaization

- Sunflower seed heads
- Pine cone spirals
- Romanesco broccoli
- Succulent leaf arrangements
- Daisy florets

The mathematical principle: **The Golden Angle** $= 137.5^\circ = 360^\circ \times (2 - \phi)$

where $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$ is the golden ratio.

7.1.2 Why This Angle Works

The golden angle is the "most irrational" number in the continued fraction sense:

$$\phi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cdots}}}$$

This property ensures:

- **No alignment**: Seeds never align along the same radius, even after hundreds of iterations
- **Optimal packing**: Maximum density with minimum gaps
- **Self-similar structure**: Pattern looks similar at all scales
- **Progressive optimization**: Refined over millions of years

7.1.3 Mathematical Formulation

For observation j in a tie group of size n_{ties}:

$$\text{angle}_j = (j-1) \times 137.508^\circ$$

$$\text{radius}_j = \epsilon \times \sqrt{\frac{j-1}{n_{\text{ties}}}}$$

$$\text{displacement}_j = \text{radius}_j \times \cos(\text{angle}_j)$$

7.1.4 Key Features

Square Root Scaling: Maintains constant density as radius increases.

In a 2D disk: - Circumference at radius r: $2\pi r$ - Number of points at radius r_j: proportional to j - For constant density: $r \propto \sqrt{j}$ balances linear growth in points with radial expansion

Cosine Projection: Maps 2D polar coordinates to 1D linear displacement while preserving distribution properties.

Spiral Structure: The characteristic spiral pattern is preserved even in 1D projection, creating aesthetically pleasing distributions.

7.1.5 Theoretical Properties

- **Near-optimal minimum separation** (Vogel 1979)
- **Low-discrepancy** in 2D space
- **Aesthetically consistent** across all scales
- **Progressively validated** optimization
- **Deterministic** and fully reproducible

7.1.6 Advantages

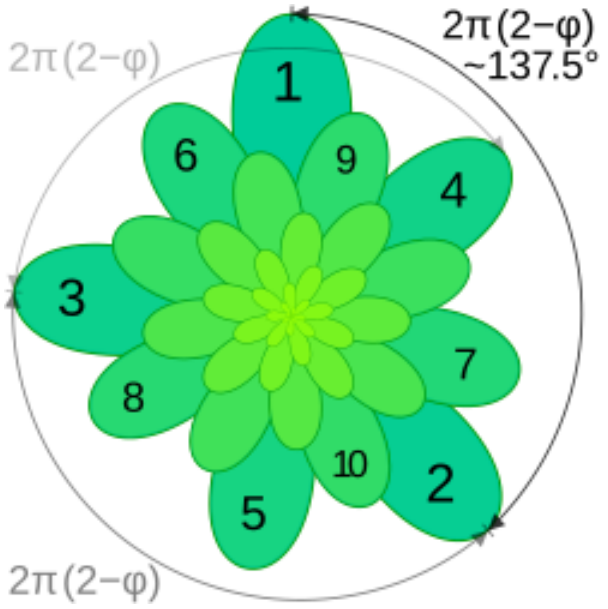

Figure 5: Phyllotaxis of the Golden Angle

1. **Biomimetic optimization**: Leverages millions of years of natural selection
2. **Aesthetic appeal**: Creates visually pleasing, organic-looking patterns
3. **Balanced distribution**: Neither too uniform (mechanical) nor too random (chaotic)
4. **Proven effectiveness**: Successfully used in nature for optimal packing

7.1.7 Applications Beyond PCPs

The Sunflower algorithm has proven useful in:

- Point cloud sampling: Uses the Fibonacci angle (137.5°) and square-root radial spacing ($r = a\sqrt{\varphi}$) to generate uniformly distributed points on discs or spheres.
- Sphere packing: Places circles or spheres at Sunflower positions to achieve near-optimal packing density in circular domains.
- Texture synthesis: Creates organic, non-repetitive patterns for procedural textures.
- Computer graphics rendering: Provides stratified sample points for stochastic rendering techniques (ray tracing, depth-of-field).
- Quasi-Monte Carlo methods

7.2 Method 2: Halton Jitter (Quasi-Random Sequences)

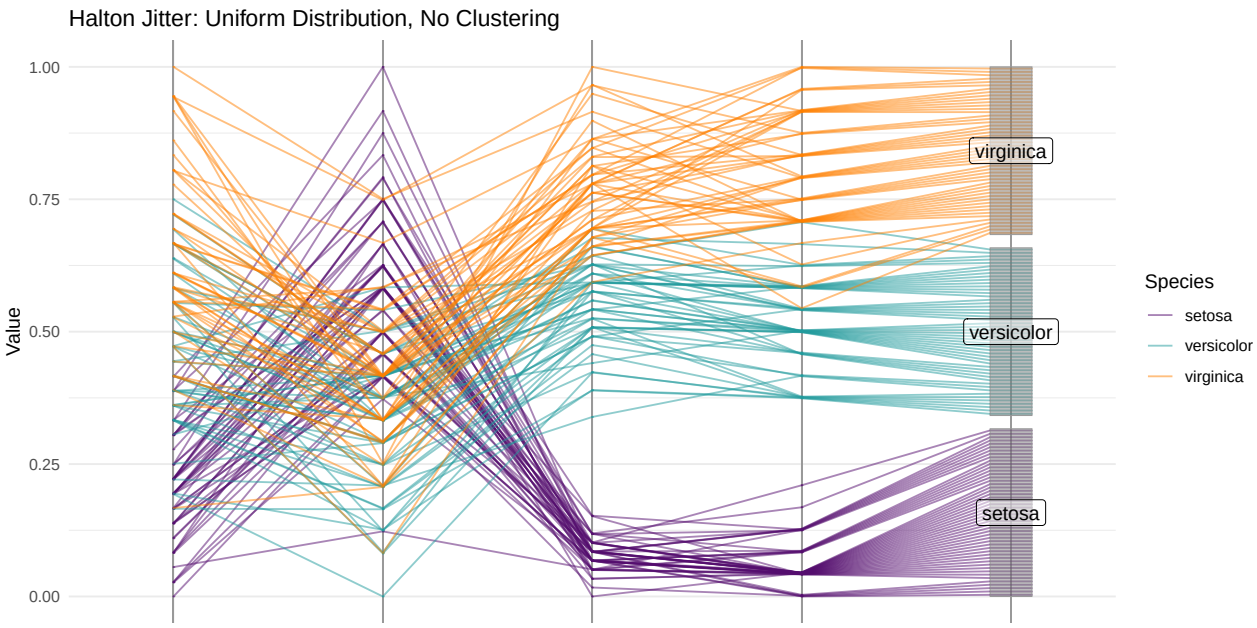

Figure 6: Halton jittering provides uniform low-discrepancy distribution

7.2.1 Beyond Pseudo-Randomness

Halton sequences (Halton 1960) represent a fundamental advancement over pseudo-random numbers:

- **Deterministic** (not random)
- **Low-discrepancy** (fill space uniformly)
- **Number-theoretically constructed** using prime bases
- **Mathematically guaranteed** uniform coverage

7.2.2 The Random Number Problem

Pseudo-random numbers inevitably cluster due to the birthday paradox:

Example: In 100 random points on [0, 1]:

- Expected maximum gap: 0.05
- Expected minimum gap: 0.0001
- Visual artifacts mislead analysts
- Density perception becomes unreliable

7.2.3 Halton's Solution

Place each new point as far as possible from all previous points through systematic construction using the **van der Corput sequence**.

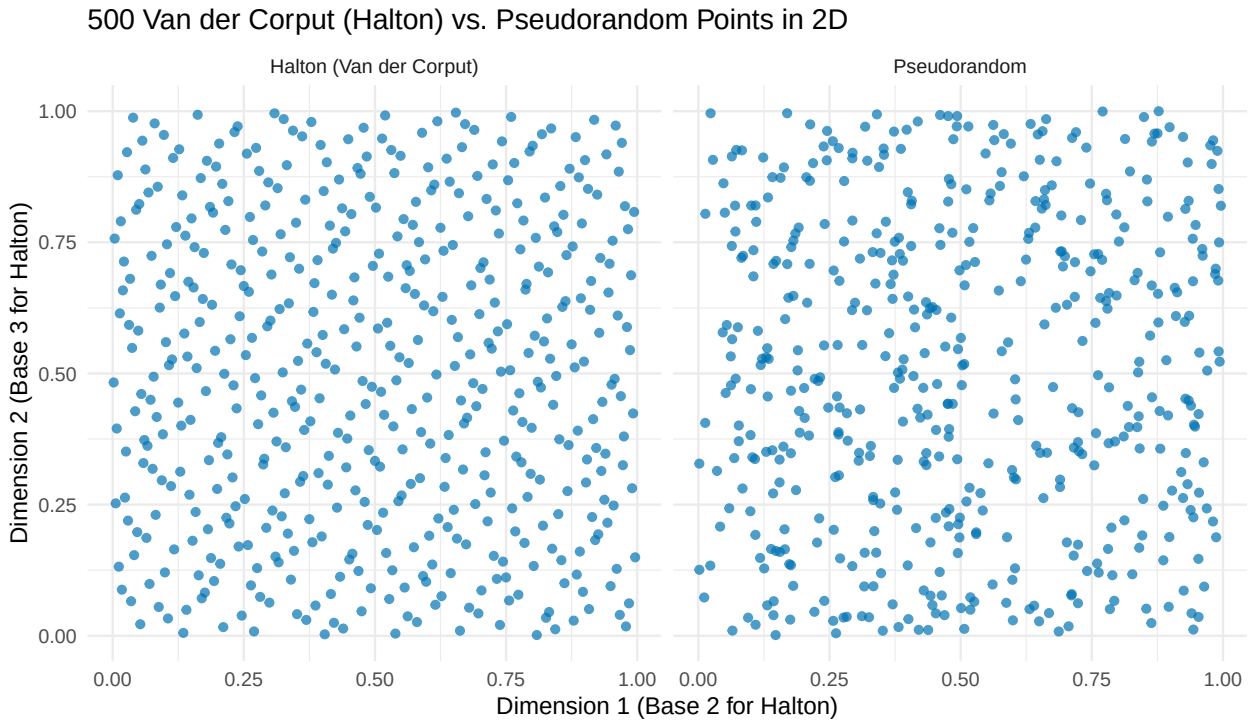

Figure 7: Van der Corput vs. pseudorandom

7.2.4 Van der Corput Sequence Construction

The van der Corput sequence in base 2 generates a low-discrepancy sequence by:

1. Take integer index i
2. Convert to binary
3. Reverse the binary digits
4. Interpret as binary fraction

Example:

i	Binary	Reversed	Decimal h_i
0	0	0	0.0
1	1	1	0.5
2	10	01	0.25
3	11	11	0.75
4	100	001	0.125
5	101	101	0.625
6	110	011	0.375
7	111	111	0.875

Pattern: Each point bisects the largest remaining gap.

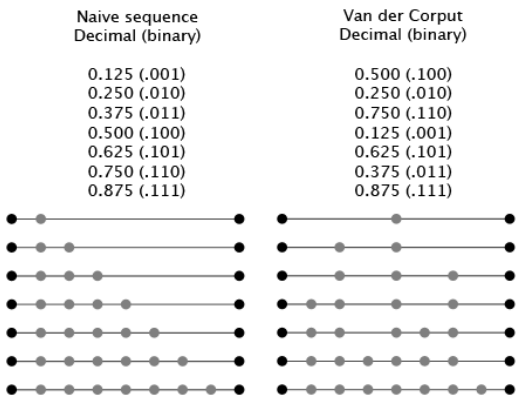

Figure 8: Van der Corput Sequence Visual

7.2.5 Mathematical Formulation

For observation i in a tie group:

$$h_i = \text{VanDerCorput}(i, \text{base} = 2)$$

$$\text{displacement}_i = \epsilon \times (h_i - 0.5)$$

Centering around 0.5 creates symmetric bidirectional displacement.

7.2.6 Theoretical Guarantees: Discrepancy Theory

Star Discrepancy measures how uniformly points fill an interval (Niederreiter 1992):

$$D_n^* = \sup_{I \subseteq [0,1]} \left| \frac{\#\{x_i \in I\}}{n} - |I| \right|$$

Theoretical Bounds:

- **Random sequences**: $D_n = O(n^{-1/2})$
- **Halton sequences**: $D_n = O(n^{-1} \log n)$
- **Optimal lower bound**: $D_n = \Omega(n^{-1} \log n)$

Conclusion: Halton sequences are **near-optimal** in the information-theoretic sense.

7.2.7 Applications in Computer Science

Halton sequences are widely used in:

- **Quasi-Monte Carlo integration**: Better convergence than random sampling

- **Computer graphics**: Anti-aliasing, global illumination
- **Ray tracing**: Sample generation for realistic rendering
- **Numerical analysis**: Multidimensional quadrature
- **Machine learning**: Hyperparameter search spaces

7.2.8 Higher-Dimensional Extensions

For 2D applications (e.g., scatter plots):

$$x_i = \text{VanDerCorput}(i, 2) \quad \text{(base 2 for x-axis)}$$
$$y_i = \text{VanDerCorput}(i, 3) \quad \text{(base 3 for y-axis)}$$

Using different prime bases for each dimension maintains low-discrepancy properties.

7.3 Method 3: Intelligent Jitter (Novel Exploration)

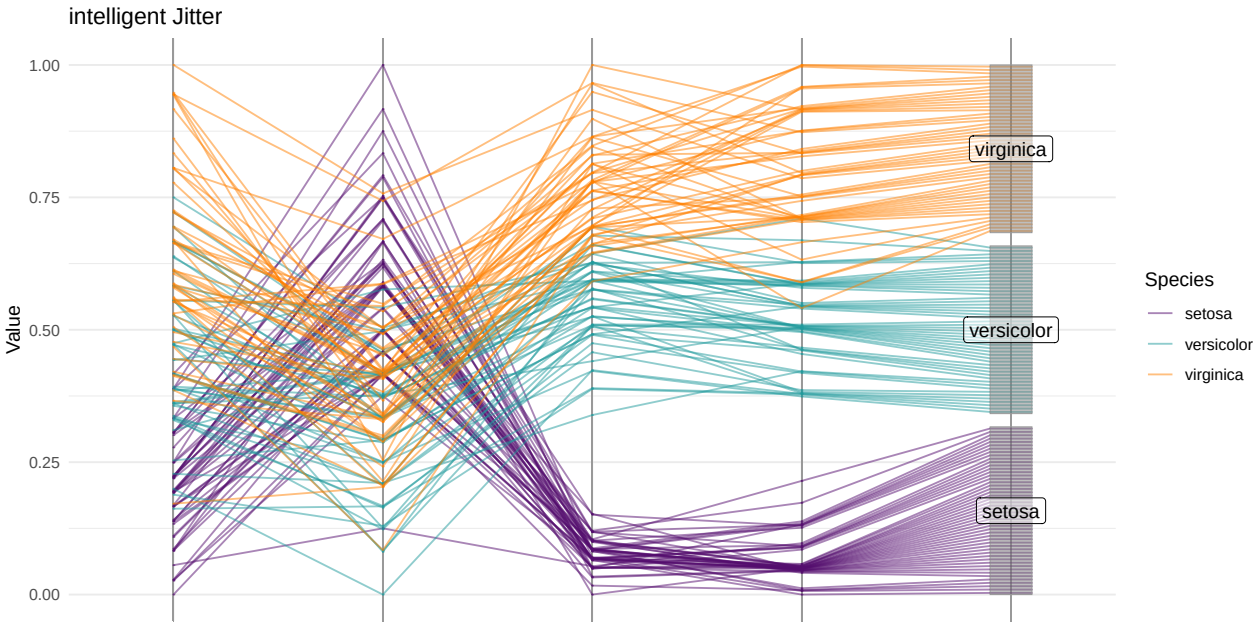

Figure 9: Intelligent jittering resolves ties

7.3.1 Design Motivation

Research Question: Can we apply the golden ratio directly in 1D rather than through angular spacing?

The Intelligent jitter was developed to explore whether:

- Direct golden ratio modulation could work in 1D
- Linear scaling might provide progressive reveal
- A simpler algorithm could match Sunflower performance

7.3.2 Mathematical Formulation

For observation j in a tie group of size n_{ties}:

$$\text{angle}_j = (j-1) \times 2\pi \times 0.618$$

$$\text{displacement}_j = \epsilon \times \cos(\text{angle}_j) \times \frac{j-1}{n_{\text{ties}}}$$

7.3.3 Key Distinctions from Sunflower

Feature	Sunflower	Intelligent
Angle	Golden angle (137.5°)	Golden ratio × 2π(224.4°)
Scaling	Square root: $\sqrt{j/n}$	Linear: j/n
Projection	2D spiral → 1D	1D cosine modulation
Inspiration	Phyllotaxis patterns	Golden ratio mathematics

7.3.4 Initial Hypothesis

Progressive Reveal: Linear scaling would create:

- **Early observations**: Small displacement (stay near true value)
- **Later observations**: Larger displacement (fill available space)
- **Golden ratio modulation**: Optimal distribution through natural constant

This would provide intuitive interpretation: "early arrivals" cluster near the true value, while "late arrivals" spread outward.

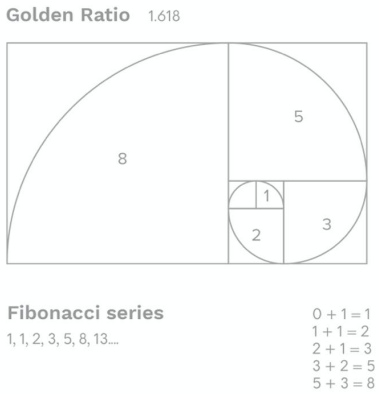

Figure 10: 1-dimension Example of the Golden Ratio

7.3.5 Empirical Reality: Failure Analysis

The Intelligent jitter produced **poor visual quality** with several critical problems:

7.3.5.1 Problem 1: Excessive Displacement

For a tie group with $n = 100$ observations:

Observation	Displacement
1	0% of ϵ
25	~24% of ϵ
50	~49% of ϵ
75	~74% of ϵ
100	~99% of ϵ

Result: Later observations displaced nearly to the boundary, creating false impression of substructure.

7.3.5.2 Problem 2: Artificial Stratification

The linear scaling creates **artificial layers** that don't represent actual data structure:

- Visual appearance suggests distinct sub-groups
- These "clusters" are algorithmic artifacts
- Misrepresents uniform density as stratified distribution

7.3.5.3 Problem 3: Perceptual Distortion

Users interpret visual patterns as data patterns:

- **Large gaps** appear meaningful (but are artifacts)
- **Density gradients** suggest ordering (but observations are exchangeable)
- **Boundary concentration** implies separation (but all values are tied)

7.3.6 Why This Method Fails: Theoretical Analysis

7.3.6.1 Root Cause: Linear Scaling

The linear scaling function $\frac{j-1}{n}$ is inappropriate because:

1. **Violates uniformity**: Creates increasing displacement magnitudes
2. **Breaks perceptual validity**: Large displacements misrepresent true values
3. **Artificial ordering**: Imposes structure where none exists
4. **Cognitive interference**: Visual patterns contradict data properties

7.3.6.2 Comparison with Sunflower

Sunflower's $\sqrt{j/n}$ scaling maintains **constant area density** in 2D:

- Area of annulus at radius r: $\pi[(r + dr)^2 - r^2] \approx 2\pi r\, dr$

- For constant density: points per area = constant
- Therefore: $\frac{dN}{dA} \propto \frac{1}{r}$
- Solution: $N \propto r^2$, thus $r \propto \sqrt{N}$

Intelligent's linear scaling has **no such geometric justification**.

7.3.6.3 Value of This Negative Result

This failure provides important scientific contributions:

1. Design Pattern to Avoid

Lesson: Linear displacement scaling creates misleading stratification.

Implication: Future methods should use constant or sublinear scaling.

2. Golden Ratio Not Universal

Lesson: Golden ratio works in specific geometric contexts (angular distribution), not universally.

Implication: Biomimetic approaches require careful adaptation, not blind application.

3. Importance of Scaling Function

Lesson: The scaling function is as critical as the distribution algorithm.

Implication: Algorithm design must consider both angular distribution and radial scaling together.

4. Empirical Validation Essential

Lesson: Theoretical elegance doesn't guarantee practical effectiveness.

Implication: User studies and visual assessment are necessary even for mathematically motivated methods.

Recommendations

Do not use Intelligent jitter for production visualizations.

Include in research as:

- **Comparison baseline**: Shows what not to do
- **Methodological contribution**: Documents design failure
- **Educational value**: Teaches importance of scaling functions
- **Scientific rigor**: Negative results advance knowledge

8 Comparative Analysis of Methods

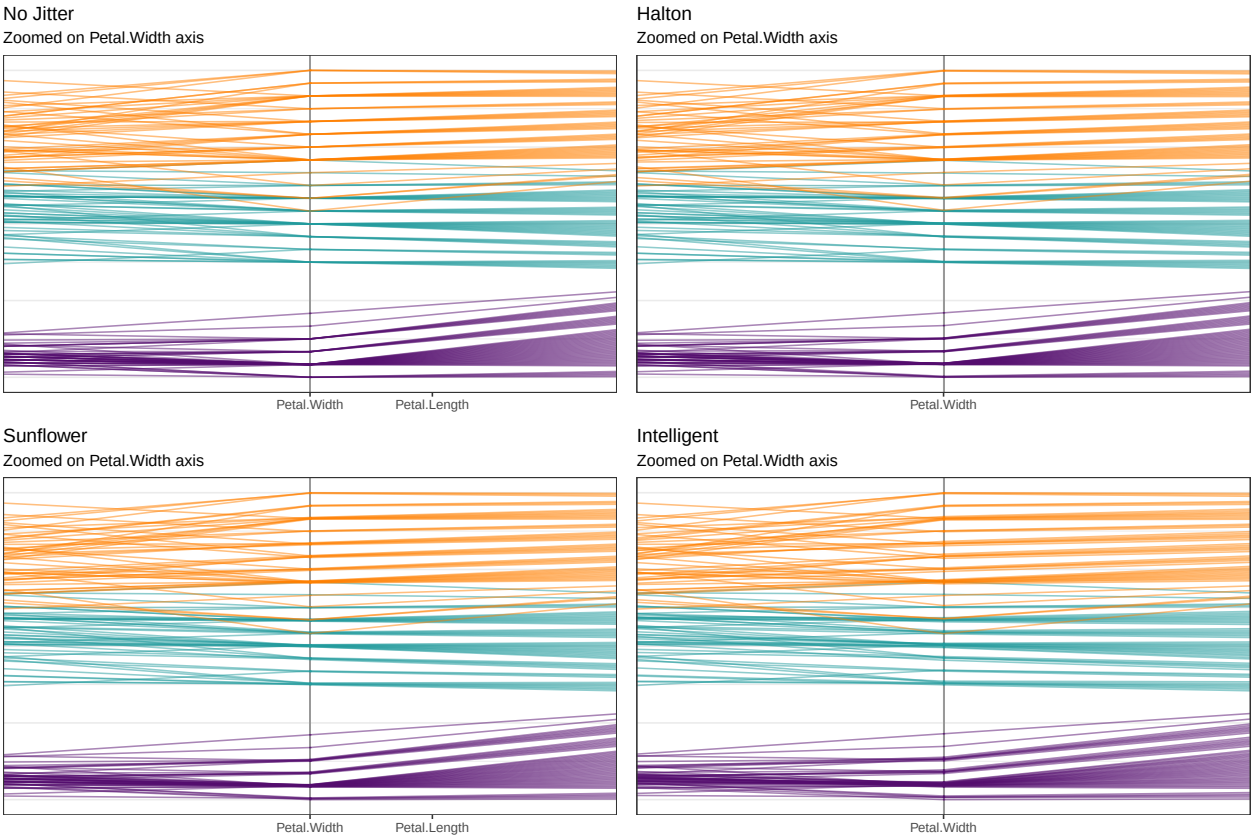

Figure 11: Side-by-side comparison of all jittering methods

8.1 Dimensional Analysis

Method	Approach	Dimension	Projection
Halton	Pure 1D sequence	1D	None
Sunflower	2D spiral	2D $\rightarrow$ 1D	Cosine
Intelligent	1D with 2D-inspired modulation	Hybrid	Cosine

8.2 Scaling Behavior

Method	Scaling Function	Growth Rate	At $n = 50$, $j = 25$
Halton	Uniform distribution	Constant	~0.5 ϵ
Sunflower	$\sqrt{j/n}$	Sublinear	~0.7 ϵ
Intelligent	j/n	Linear	~0.5 ϵ

8.3 Distribution Quality Metrics

8.3.1 Minimum Separation Distance

Smallest gap between any two consecutive points:

- **Halton**: Guaranteed $O(1/n)$, highly predictable
- **Sunflower**: Approximately $O(1/\sqrt{n})$ in projection, variable
- **Intelligent**: Highly variable, depends on cosine phase

8.3.2 Discrepancy (Uniformity Measure)

How evenly points fill the interval:

- **Halton**: $O(n^{-1} \log n)$ — near-optimal
- **Sunflower**: Not directly measurable in 1D projection, but empirically good
- **Intelligent**: Poor due to linear scaling creating density gradients

8.3.3 Visual Clustering

Tendency to create artificial clusters:

- **Halton**: Minimal — systematic gap-filling
- **Sunflower**: Slight central concentration (natural)
- **Intelligent**: Severe stratification artifacts

8.4 Aesthetic Quality

Subjective but important for user acceptance:

- **Halton**: Clean, mechanical, predictable
- **Sunflower**: Organic, pleasing, natural
- **Intelligent**: Artificial, stratified, problematic

8.5 Use Case Recommendations

8.5.1 Halton: Best for...

- **Precision-critical applications**: Scientific publications
- **Maximum uniformity**: When faithful density representation is paramount
- **Mathematical rigor**: When provable guarantees matter
- **Large datasets**: Efficient, predictable performance

8.5.2 Sunflower: Best for...

- **General purpose**: Good balance of properties
- **Aesthetic presentations**: Visually appealing
- **Exploratory analysis**: Natural-looking distributions
- **User preference**: Often preferred in studies

8.5.3 Intelligent: Best for...

- **Methodological research**: As a comparison baseline
- **Educational examples**: Teaching what not to do
- **Negative controls**: Demonstrating failure modes
- **Do NOT use for production visualizations**

9 Integration with ggpcp Package

9.1 Architectural Integration

The `ggpcp` package implements a grammar of graphics approach with three core modules:

1. **Variable Selection** (`pcp_select`): Choose and order dimensions
2. **Axis Scaling** (`pcp_scale`): Normalize or transform scales
3. **Tie Resolution** (`pcp_arrange`): Handle overlapping values ← **EXTENDED**

Our contribution extends `pcp_arrange` to handle numerical ties alongside existing categorical tie-breaking.

9.2 Proposed ggpcp Implementation

9.2.1 Function Signature

```
pcp_arrange(
  data,
  method = c("from-left", "from-right", "halton", "sunflower", "intelligent"),
  space = 0.05,
  epsilon = NULL,
  numeric_ties = TRUE
)
```

9.2.2 Parameters

- **method**: Tie-breaking strategy
  - `"from-left"`, `"from-right"`: Existing categorical methods

  - `"halton"`: Quasi-random low-discrepancy sequence
  - `"sunflower"`: Biomimetic golden angle distribution
  - `"intelligent"`: Golden ratio linear (research only)

- **space**: Proportion of axis for categorical spacing (existing parameter)
- **epsilon**: Maximum displacement for numerical ties
  - `NULL` (default): Auto-determined as 0.05 * axis range
  - Numeric value: User-specified displacement magnitude
- **numeric_ties**: Whether to apply jittering to numerical ties (default: TRUE)

9.2.3 Example Usage

```
library(ggpcp)
library(dplyr)

# Basic usage with Sunflower (recommended default)
iris_plot <- iris %>%
  pcp_select(1:5) %>%
  pcp_scale(method = "uniminmax") %>%
  pcp_arrange(method = "sunflower") %>%
  ggplot() +
  geom_pcp()

# Halton for maximum uniformity
iris_halton <- iris %>%
  pcp_select(Sepal.Length:Species) %>%
  pcp_scale(method = "uniminmax") %>%
  pcp_arrange(
    method = "halton",
    epsilon = 0.08,
    numeric_ties = TRUE
  ) %>%
  ggplot() +
  geom_pcp(aes(color = Species))

# Mixed categorical and numerical ties
mixed_data %>%
  pcp_select(cat1, num1, cat2, num2) %>%
  pcp_arrange(
    method = "sunflower",  # Applied to numerical
    space = 0.05           # Applied to categorical
  )

# Comparison of methods
```

```r
library(patchwork)

p_none <- iris %>% pcp_select(1:4) %>%
  pcp_arrange(method = "none") %>% plot_pcp()

p_halton <- iris %>% pcp_select(1:4) %>%
  pcp_arrange(method = "halton") %>% plot_pcp()

p_sunflower <- iris %>% pcp_select(1:4) %>%
  pcp_arrange(method = "sunflower") %>% plot_pcp()

(p_none | p_halton | p_sunflower) +
  plot_annotation(title = "Comparison of Tie-Breaking Methods")
```

9.3 Backward Compatibility

- Existing `pcp_arrange` calls continue to work unchanged
- Default behavior (categorical-only) preserved when `numeric_ties = FALSE`
- New functionality opt-in through explicit method selection
- Warning messages guide users to new features

9.4 Documentation Requirements

9.4.1 Function Documentation

- Detailed explanation of each method
- Theoretical foundations and references
- When to use each method
- Parameter selection guidance
- Examples with multiple datasets

9.4.2 Vignettes

1. **"Handling Numerical Ties in ggpcp"**: Introduction and basic usage
2. **"Comparing Tie-Breaking Methods"**: Detailed comparison with examples
3. **"Advanced Tie Resolution"**: Parameter tuning and edge cases
4. **"Theory of Deterministic Jittering"**: Mathematical foundations

9.4.3 Visual Indicators

Add optional visual indicators showing:

- Which axes have numerical tie-breaking applied
- Magnitude of epsilon used
- Number of tied observations per group
- Method employed

10 Research Questions and Methodology

10.1 Primary Research Question

How can the formal structure of the Grammar of Graphics be extended to systematically incorporate and evaluate methods for resolving numerical ties in parallel coordinate plots, and what is the quantifiable impact of these methods on the accuracy and efficiency of visual data analysis?

10.2 Secondary Research Questions

10.2.1 RQ1: Theory

How can the management of numerical ties be most effectively and coherently formalized within the layered grammar of graphics, building on the established ggpcp framework for categorical tie management?

Methodology:

- Theoretical analysis of Grammar of Graphics structure
- Literature synthesis on position adjustments
- Specification of new grammatical element
- Integration with existing ggpcp architecture
- Formal documentation of tie-breaking grammar

Deliverables:

- Formal specification document
- Extended grammar notation
- Theoretical paper on biomimetic transformations
- Integration guidelines for ggpcp

10.2.2 RQ2: Methodology

What are the optimal algorithmic criteria for ordering and spacing tied data points to maximize visual clarity while preserving underlying data properties, particularly in comparison to the hierarchical sorting methods already established in ggpcp for categorical variables?

Methodology:

- Algorithm design and implementation in R
- Comparative analysis of distribution quality
- Computational performance benchmarking
- Parameter sensitivity analysis
- Edge case identification and handling

Evaluation Metrics:

- Minimum separation distance
- Discrepancy (uniformity measure)
- Computational complexity
- Memory efficiency
- Scalability testing

Deliverables:

- Complete R implementation in ggpcp
- Performance benchmarks
- Algorithm comparison paper
- Best practices guidelines

10.2.3 RQ3: Perception

How do different visualization strategies for numerical ties affect an analyst's ability to perform key visual tasks (cluster identification, outlier detection, density estimation) compared to the categorical tie-breaking approaches already validated in ggpcp?

Methodology: Controlled user study with human subjects

Study Design:

- **Type**: Within-subjects repeated measures
- **Participants**: 30-50 analysts (mixed expertise)
- **Methods compared**: No jitter, Random, Halton, Sunflower, Intelligent
- **Tasks**:
  - Density estimation: "How many observations follow this path?"
  - Cluster identification: "Identify distinct clusters"
  - Outlier detection: "Find observations that don't fit patterns"
  - Pattern tracing: "Follow observation 42 across all dimensions"

Dependent Variables:

1. **Accuracy**: Absolute error from ground truth
2. **Completion Time**: Seconds to complete task
3. **Confidence**: Self-reported certainty (1-10 scale)
4. **Preference**: Comparative ranking of methods

Statistical Analysis:

- Repeated-measures ANOVA
- Bonferroni post-hoc tests
- Effect size calculations (partial η^2)

- Correlation analysis (accuracy vs. confidence)

Expected Hypotheses:

- **H1**: Halton and Sunflower > Random > No Jitter (accuracy)
- **H2**: Halton ≈ Sunflower < Random < No Jitter (time)
- **H3**: Sunflower ≈ Halton > Random > No Jitter (preference)
- **H4**: Intelligent performs poorly across all metrics

Deliverables:

- IRB-approved study protocol
- Complete dataset with results
- Statistical analysis report
- Empirical paper on perceptual effectiveness

10.2.4 RQ4: Practice

Can a set of evidence-based heuristics be developed to guide practitioners in selecting the most appropriate numerical tie-breaking method for their specific data context, integrating with existing ggpcp parameter selection guidelines?

Methodology:

- Synthesize findings from RQ1-3
- Develop decision tree/flowchart
- Validate with case studies
- Gather practitioner feedback
- Refine through iterative testing

Case Study Domains:

1. **Bioinformatics**: Gene expression data
2. **Finance**: Market data with discrete prices
3. **Engineering**: Sensor data with limited precision
4. **Social Science**: Survey responses with Likert scales
5. **Climate Science**: Model ensemble outputs

Heuristic Framework Components:

- Data characteristics assessment
- Task requirements analysis
- Method selection guidelines
- Parameter tuning recommendations
- Validation procedures

Deliverables:

- Practitioner's guide document
- Interactive decision tool (Shiny app)
- Case study collection
- Best practices paper
- Tutorial videos

11 Implementation Roadmap

11.1 Phase 1: Algorithm Refinement (Winter 2025)

11.1.1 Tasks

1. Finalize all three jittering implementations
2. Develop adaptive epsilon selection
3. Optimize computational performance
4. Complete test suite with edge cases
5. Benchmark against large datasets

11.1.2 Deliverables

- Optimized R functions
- Unit tests with 100% coverage
- Performance benchmarks
- Technical documentation

11.2 Phase 2: ggpcp Integration (Spring 2026)

11.2.1 Tasks

1. Extend `pcp_arrange()` function
2. Implement automatic tie detection
3. Add epsilon auto-determination
4. Create visual indicators
5. Write package vignettes
6. Prepare for CRAN submission

11.2.2 Deliverables

- Updated ggpcp package
- Comprehensive documentation
- Three tutorial vignettes
- Package ready for CRAN

11.3 Phase 3: User Study (Spring-Summer 2026)

11.3.1 Tasks

1. Obtain IRB approval (early Spring)
2. Develop study materials
3. Recruit participants
4. Conduct study sessions
5. Analyze results
6. Write empirical paper

11.3.2 Deliverables

- IRB approval documentation
- Complete dataset
- Statistical analysis
- Empirical research paper draft

11.4 Phase 4: Case Studies & Dissertation Writing (Summer 2026)

11.4.1 Tasks

1. Apply to diverse real-world datasets
2. Gather practitioner feedback
3. Develop decision heuristics
4. Write dissertation chapters
5. Integrate all components
6. Prepare defense presentation

11.4.2 Deliverables

- Five domain case studies
- Practitioner's guide
- Complete dissertation draft
- Defense presentation

11.5 Phase 5: Final Review and Defense (May-July 2026)

11.5.1 Tasks

1. Committee review of dissertation
2. Incorporate feedback
3. Final revisions
4. Defense rehearsals
5. Dissertation defense

11.5.2 Deliverables

- Final dissertation
- Successful defense
- Submitted for graduation

12 Expected Outcomes and Contributions

12.1 Theoretical Contributions

1. **Grammar of Graphics Extension**
   - Formal specification of biomimetic transformations
   - Integration of natural optimization principles
   - New category of position adjustments
2. **Cross-Domain Algorithm Adaptation**
   - Phyllotaxis → data visualization
   - Quasi-random sequences → statistical graphics
   - Demonstrates value of interdisciplinary approaches
3. **Negative Result Documentation**
   - Intelligent jitter failure analysis
   - Design patterns to avoid
   - Methodological lessons

12.2 Methodological Contributions

1. **Three Novel Algorithms**
   - Halton jitter for PCPs
   - Sunflower jitter for PCPs
   - Intelligent jitter (with failure analysis)
2. **Comparative Framework**
   - Systematic evaluation criteria
   - Quantitative metrics
   - Perceptual assessment methods
3. **Implementation Quality**
   - Production-ready R code
   - Comprehensive testing
   - Extensive documentation

12.3 Practical Contributions

1. **ggpcp Package Enhancement**
   - Complete tie-handling solution
   - Categorical + numerical ties
   - Unified grammar interface
2. **User Guidance**
   - Evidence-based selection heuristics
   - Interactive decision tools
   - Tutorial materials
3. **Real-World Impact**
   - Improved exploratory data analysis
   - More accurate pattern detection
   - Better density visualization

12.4 Empirical Contributions

1. **User Study Results**
   - Quantitative performance data
   - Perceptual effectiveness measures
   - Preference rankings
2. **Case Study Collection**
   - Diverse domain applications
   - Best practices examples
   - Common pitfall documentation
3. **Benchmark Dataset**
   - Performance comparisons
   - Scalability testing
   - Reference implementations

13 Broader Implications

13.1 Beyond Parallel Coordinates

The methods developed here generalize to other visualization contexts:

13.1.1 2D Scatter Plots

Problem: Overplotting with tied values

Solution: Full 2D Sunflower and 2D Halton

```r
# 2D Sunflower for scatter plots
scatter_sunflower <- function(x_ties, y_ties, epsilon) {
  n <- length(x_ties)
  angles <- (0:(n-1)) * 137.508 * (pi/180)
  radii <- epsilon * sqrt((0:(n-1)) / n)

  x_displaced <- x_ties + radii * cos(angles)
  y_displaced <- y_ties + radii * sin(angles)

  return(data.frame(x = x_displaced, y = y_displaced))
}
```

13.1.2 Time Series Visualization

Problem: Multiple series with identical values at time points

Solution: Vertical displacement using deterministic methods

13.1.3 Network Visualization

Problem: Node positioning with spatial constraints

Solution: Optimal space-filling using golden angle principles

13.2 General Principle

Wherever random jitter is currently used, deterministic low-discrepancy alternatives should be considered.

Benefits:

- Reproducibility
- Better distribution quality
- Elimination of artifacts
- Theoretical guarantees

14 Timeline to Dissertation Defense

Phase	Timeframe	Key Milestones
Algorithm Refinement	Winter 2025 (Months 1-2)	Algorithm optimization, adaptive epsilon
ggpcp Integration	Spring 2026 (Months 3-4)	Package update, documentation
User Study	Spring-Summer 2026 (Months 5-7)	IRB approval, data collection, analysis

Phase	Timeframe	Key Milestones
Case Studies & Writing	Summer 2026 (Months 7-8)	Real-world validation, dissertation drafting
Dissertation Completion	May-June 2026	Final revisions, committee review
Defense	**July 2026**	**Final defense and submission**

15 Conclusion

This research addresses a fundamental limitation in parallel coordinate visualization: the visual occlusion caused by numerical ties. While the ggpcp package has successfully solved categorical ties through hierarchical sorting, numerical ties have remained problematic.

We propose three deterministic jittering methods, Halton (quasi-random sequences), Sunflower (biomimetic distribution), and Intelligent (exploratory approach), that provide reproducible, theoretically-grounded solutions. Our comparative analysis reveals:

- **Halton excels** in uniformity and mathematical rigor
- **Sunflower balances** performance with aesthetic appeal
- **Intelligent demonstrates** the importance of proper scaling functions

The integration of these methods into ggpcp will provide users with a complete solution for all forms of ties, categorical and numerical—within a unified Grammar of Graphics framework. This research contributes to visualization theory, methodology, and practice while establishing principles applicable beyond parallel coordinates.

The planned user study will provide empirical evidence for method selection, while case studies will demonstrate real-world effectiveness. Together, these components will deliver evidence-based guidance for practitioners and advance the theoretical understanding of position adjustments in statistical graphics.

By incorporating nature-inspired optimization principles into data visualization, this work exemplifies how cross-disciplinary approaches can solve longstanding challenges in information visualization.

16 Appendix A: Algorithm Pseudocode and Implementation Details

This appendix provides complete pseudocode for all three jittering methods, along with implementation notes and computational complexity analysis.

16.1 A.1 Halton Jitter Algorithm

16.1.1 A.1.1 Overview

The Halton jitter algorithm generates low-discrepancy sequences using the van der Corput construction in base 2. This ensures near-optimal uniform distribution of displaced points.

16.1.2 A.1.2 Van der Corput Sequence Generator

```
van_der_corput <- function(index, base = 2) {
  if (index < 0) stop("Index must be non-negative")
  if (base <= 1) stop("Base must be greater than 1")

  result <- 0
  denominator <- base

  while (index > 0) {
    digit <- index %% base
    result <- result + digit / denominator
    index <- floor(index / base)
    denominator <- denominator * base
  }

  return(result)
}
```

Example Execution (base = 2):

```
index <- 5
base <- 2
result <- 0
denominator <- base
iteration <- 1

cat("Starting van_der_corput(", index, ",", base, ")\n\n")

while (index > 0) {
  digit <- index %% base
  result <- result + digit / denominator
  cat(sprintf("Iteration %d: digit = %d, result = %.6f, index = %d\n",
              iteration, digit, result, index))

  index <- floor(index / base)
  denominator <- denominator * base
  iteration <- iteration + 1
}

cat("\nReturn:", result, "\n")
```

16.1.3 A.1.3 Main Halton Jitter Function

```
halton_jitter <- function(tied_values, epsilon) {
  n <- length(tied_values)
  jittered_values <- numeric(n)
```

```
  for (i in 0:(n - 1)) {
    # Generate Halton sequence value (base 2)
    h <- van_der_corput(i, base = 2)

    # Center around 0.5 for symmetric displacement
    centered_h <- h - 0.5

    # Apply scaled displacement
    displacement <- epsilon * centered_h
    jittered_values[i + 1] <- tied_values[i + 1] + displacement
  }

  return(jittered_values)
}
```

16.1.4 A.1.4 Complete Implementation with Tie Detection

```
apply_halton_jitter_to_data <- function(data, column_name, epsilon = NULL) {
    stopifnot(is.data.frame(data), column_name %in% names(data))
    values <- data[[column_name]]

    # Auto-calculate epsilon (5% of observed range, ignoring NAs)
    if (is.null(epsilon)) {
        rng <- range(values, na.rm = TRUE)
        value_range <- diff(rng)
        if (!is.finite(value_range) || value_range == 0) value_range <- 1
        epsilon <- 0.05 * value_range
    }

    # Indices to consider (skip NAs so they remain untouched)
    idx <- which(!is.na(values))

    # Group indices by exact tied value
    groups <- split(idx, values[idx])

    # Apply Halton jitter only to groups with ties
    for (g in groups) {
        if (length(g) > 1) {
            jittered <- halton_jitter(rep(values[g[1]], length(g)), epsilon)
            data[[column_name]][g] <- jittered
        }
    }

    data
}
```

16.1.5 A.1.5 Computational Complexity

- **Van der Corput generation**: $O(\log n)$ per point (number of binary digits)
- **Per tie group**: $O(k \log k)$ where k is group size
- **Overall**: $O(N \log N)$ dominated by tie detection (sorting)
- **Space**: $O(N)$ for storing displacements

16.1.6 A.1.6 Key Properties

- **Deterministic**: Same input always produces same output
- **Low-discrepancy**: $D_n = O(n^{-1} \log n)$ (near-optimal)
- **Uniform**: Systematically fills largest gaps first
- **Symmetric**: Centering around 0.5 creates bidirectional displacement

16.2 A.2 Sunflower Jitter Algorithm

16.2.1 A.2.1 Overview

The Sunflower jitter algorithm adapts Vogel's phyllotaxis model, using the golden angle (137.508°) and square-root radial scaling to achieve optimal point distribution through biomimetic principles.

16.2.2 A.2.2 Mathematical Constants

```
# Golden ratio and derived constants
PHI <- (1 + sqrt(5)) / 2                    # 1.618034
GOLDEN_ANGLE_DEG <- 360 / (PHI^2)           # 137.508
GOLDEN_ANGLE_RAD <- GOLDEN_ANGLE_DEG * (pi / 180)  # 2.39996 radians
```

16.2.3 A.2.3 Main Sunflower Jitter Function

```
sunflower_jitter <- function(tied_values, epsilon) {
  n <- length(tied_values)
  jittered_values <- numeric(n)
  golden_angle <- 137.508 * pi / 180  # radians

  for (j in seq_len(n)) {
    # Angular position using golden angle
    angle <- (j - 1) * golden_angle

    # Radial position with sqrt scaling (constant area density)
    radius <- epsilon * sqrt((j - 1) / n)

    # Project 2D polar coordinates to 1D using cosine
    displacement <- radius * cos(angle)

    # Apply displacement
```

```
    jittered_values[j] <- tied_values[j] + displacement
  }

  return(jittered_values)
}
```

16.2.4 A.2.4 Alternative: Using Sine Projection

```
sunflower_jitter_sine <- function(tied_values, epsilon) {
    n <- length(tied_values)
    jittered_values <- numeric(n)
    golden_angle <- 137.508 * pi / 180  # radians

    for (j in seq_len(n)) {
        angle <- (j - 1) * golden_angle
        radius <- epsilon * sqrt((j - 1) / n)

        # Use sine instead of cosine (90° phase shift)
        displacement <- radius * sin(angle)

        jittered_values[j] <- tied_values[j] + displacement
    }

    return(jittered_values)
}
```

16.2.5 A.2.5 2D Sunflower (For Scatter Plots)

```
sunflower_jitter_2d <- function(x_tied, y_tied, epsilon_x, epsilon_y) {
  n <- length(x_tied)
  jittered_x <- numeric(n)
  jittered_y <- numeric(n)
  golden_angle <- 137.508 * pi / 180  # radians

  for (j in seq_len(n)) {
    angle <- (j - 1) * golden_angle
    radius_factor <- sqrt((j - 1) / n)

    jittered_x[j] <- x_tied[j] + epsilon_x * radius_factor * cos(angle)
    jittered_y[j] <- y_tied[j] + epsilon_y * radius_factor * sin(angle)
  }

  return(data.frame(x = jittered_x, y = jittered_y))
}
```

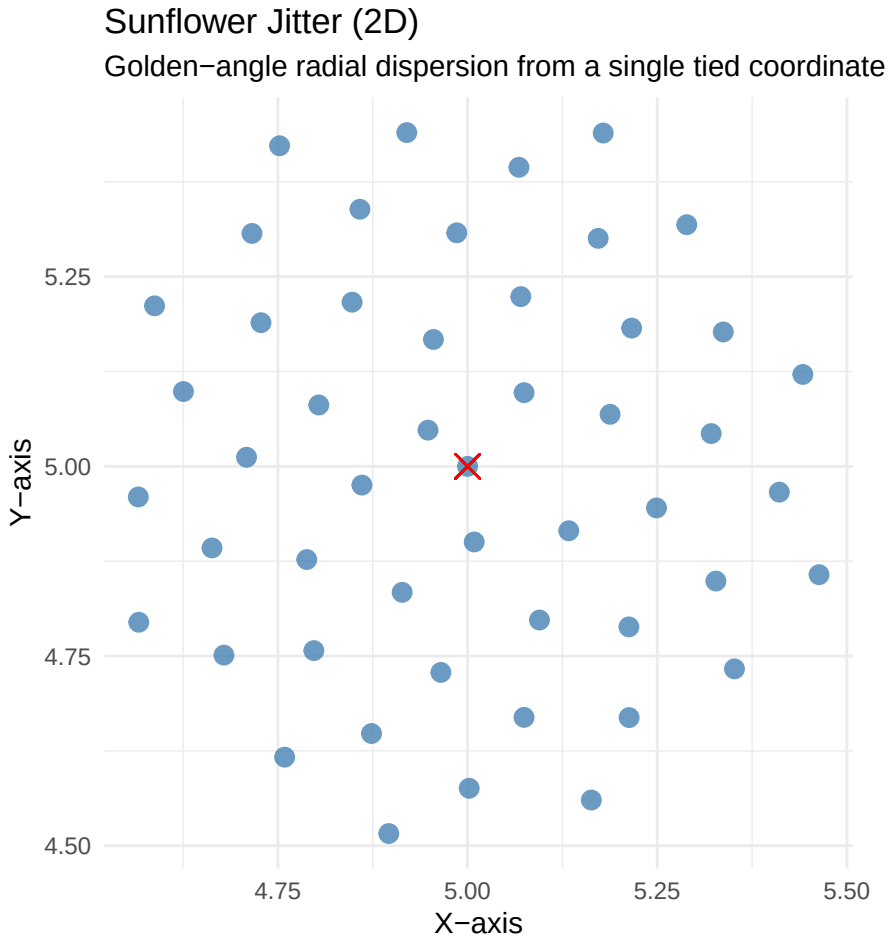

Figure 12: Golden-angle radial dispersion from a single tied coordinate

16.2.6 A.2.6 Computational Complexity

- **Per tie group**: $O(k)$ where k is group size
- **Overall**: $O(N \log N)$ dominated by tie detection
- **Space**: $O(N)$ for storing displacements
- **Trigonometric operations**: Modern implementations use lookup tables or CORDIC

16.2.7 A.2.7 Key Properties

- **Biomimetic**: Based on evolutionary optimization in sunflower seeds
- **Spiral structure**: Creates characteristic phyllotaxis spiral pattern
- **Square-root scaling**: Maintains constant area density in 2D
- **Aesthetic**: Visually pleasing, organic-looking distributions
- **Near-optimal**: Approaches optimal packing efficiency

16.3 A.3 Intelligent Jitter Algorithm

16.3.1 A.3.1 Overview

The Intelligent jitter algorithm represents an exploratory approach that applies the golden ratio directly with linear scaling. **This method performs poorly and serves as an educational example of problematic design choices.**

16.3.2 A.3.2 Main Intelligent Jitter Function

```
intelligent_jitter <- function(tied_values, epsilon) {
  n <- length(tied_values)
  jittered_values <- numeric(n)
  golden_ratio <- 0.618  # (sqrt(5) - 1) / 2

  for (j in seq_len(n)) {
    # Angular position using golden ratio × 2
    # (Not the golden angle)
    angle <- (j - 1) * 2 * pi * golden_ratio  #  3.883 radians per step

    # Linear scaling (note: not sqrt scaling)
    scale_factor <- (j - 1) / n

    # Cosine modulation for displacement
    displacement <- epsilon * cos(angle) * scale_factor

    jittered_values[j] <- tied_values[j] + displacement
  }

  return(jittered_values)
}
```

16.3.3 A.3.3 Why This Algorithm Fails

Problem 1: Linear Scaling Creates Stratification

```
Example with n = 10, epsilon = 1.0:

j=1:  displacement = 0.00   (0% of epsilon)
j=2:  displacement = 0.10   (10% of epsilon)
j=3:  displacement = 0.20   (20% of epsilon)
j=5:  displacement = 0.40   (40% of epsilon)
j=10: displacement = 0.90   (90% of epsilon)

Result: Artificial "layers" that don't represent data structure
```

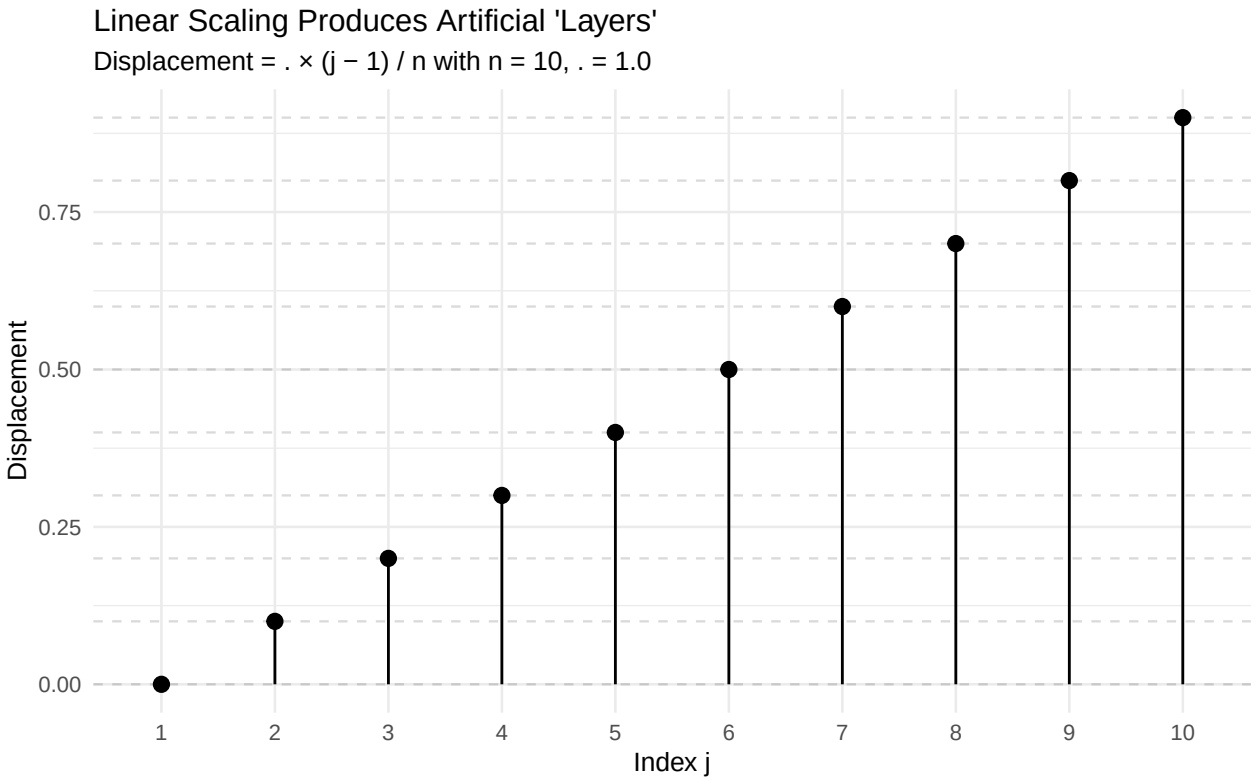

Figure 13: Artifical layers in a visual representation

Problem 2: Excessive Displacement for Later Observations

The linear scaling $(j-1)/n$ means: - First observation: No displacement (misleadingly appears "more true") - Last observation: Nearly maximum displacement (appears "less reliable") - Creates false impression of ordering or hierarchy

Problem 3: Wrong Angular Increment

Using $2\pi \times 0.618$ ($\approx 224.4°$) instead of golden angle (137.508°) loses the optimal properties of phyllotaxis patterns.

16.3.4 A.3.4 Comparison: Intelligent vs. Sunflower

```
FOR n = 100 observations:

SUNFLOWER SCALING:
    j=25:  radius_factor = SQRT(24/100) = 0.49  (49% of max)
    j=50:  radius_factor = SQRT(49/100) = 0.70  (70% of max)
    j=100: radius_factor = SQRT(99/100) = 0.995 (99.5% of max)

    Distribution: Gradual, maintains constant density

INTELLIGENT SCALING:
    j=25:  scale_factor = 24/100 = 0.24  (24% of max)
```

```
j=50:  scale_factor = 49/100 = 0.49  (49% of max)
j=100: scale_factor = 99/100 = 0.99  (99% of max)

Distribution: Linear growth, creates stratification
```

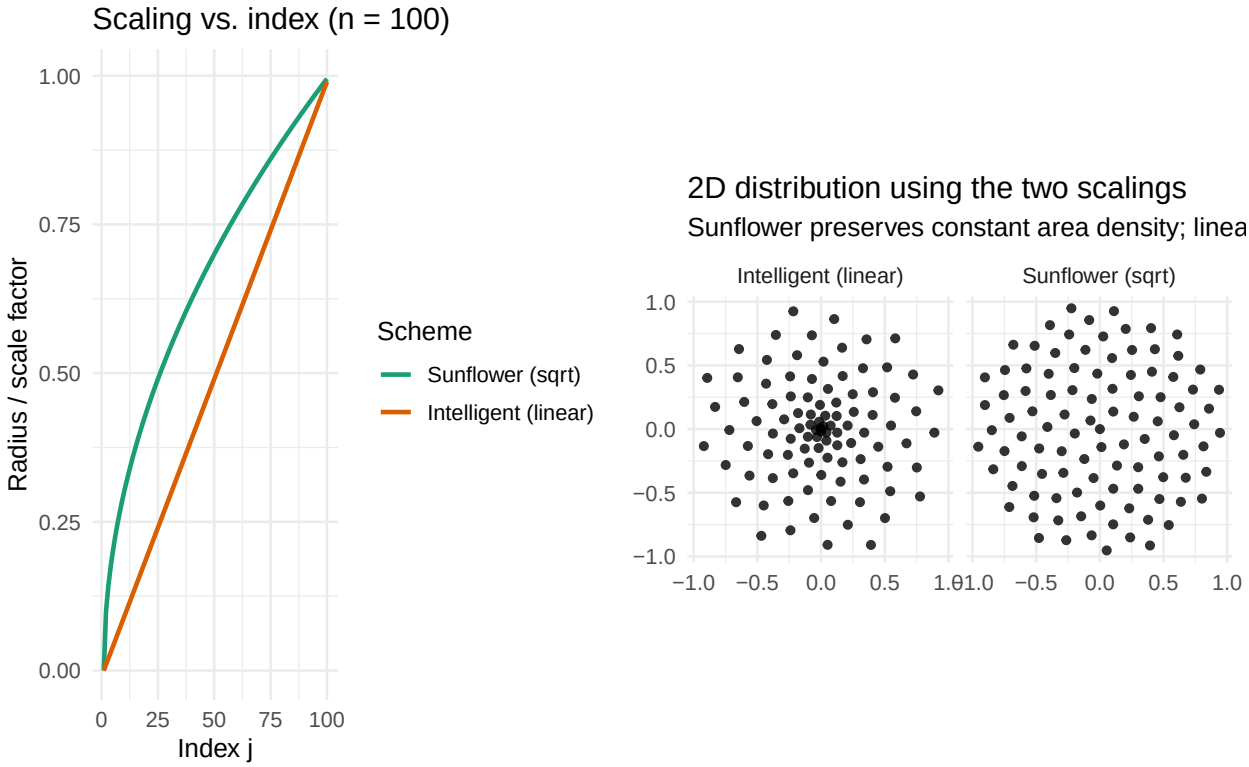

Figure 14: Giving a visual example of the difference between sunflower and intelligent scaling

16.3.5 A.3.5 When to Use Intelligent Jitter

DO NOT USE for production visualizations.

Appropriate uses:

1. **Methodological research**: As a comparison baseline demonstrating poor design
2. **Educational examples**: Teaching what NOT to do in algorithm design
3. **Negative controls**: Showing impact of linear vs. sublinear scaling
4. **Publication completeness**: Documenting failed approaches for scientific record

16.3.6 A.3.6 Computational Complexity

- **Per tie group**: $O(k)$ where k is group size (same as other methods)
- **Overall**: $O(N \log N)$ dominated by tie detection
- **Space**: $O(N)$

Note: Despite similar computational complexity, the **perceptual complexity** is much worse due to misleading visual patterns.

16.4 A.4 Comparative Implementation Notes

16.4.1 A.4.1 R Implementation Skeleton

```r
# Halton implementation
halton_jitter_r <- function(values, epsilon = NULL) {
  n <- length(values)
  if (is.null(epsilon)) {
    epsilon <- 0.05 * diff(range(values))
  }

  # Van der Corput sequence in base 2
  vdc <- function(i) {
    result <- 0
    denom <- 2
    while (i > 0) {
      result <- result + (i %% 2) / denom
      i <- i %/% 2
      denom <- denom * 2
    }
    return(result)
  }

  h_values <- sapply(0:(n-1), vdc)
  displacements <- epsilon * (h_values - 0.5)

  return(values + displacements)
}

# Sunflower implementation
sunflower_jitter_r <- function(values, epsilon = NULL) {
  n <- length(values)
  if (is.null(epsilon)) {
    epsilon <- 0.05 * diff(range(values))
  }

  golden_angle <- 137.508 * (pi / 180)
  angles <- (0:(n-1)) * golden_angle
  radii <- epsilon * sqrt((0:(n-1)) / n)
  displacements <- radii * cos(angles)

  return(values + displacements)
}

# Intelligent implementation (for comparison only)
intelligent_jitter_r <- function(values, epsilon = NULL) {
  n <- length(values)
```

```r
  if (is.null(epsilon)) {
    epsilon <- 0.05 * diff(range(values))
  }

  golden_ratio <- 0.618
  angles <- (0:(n-1)) * 2 * pi * golden_ratio
  scales <- (0:(n-1)) / n
  displacements <- epsilon * cos(angles) * scales

  return(values + displacements)
}
```

16.4.2 A.4.2 Python Implementation Skeleton

16.4.3 A.4.3 Performance Optimization Tips

For all methods:

1. **Vectorization**: Use array operations instead of loops when possible
2. **Pre-computation**: Calculate constants (golden angle, etc.) once
3. **Parallel processing**: Tie groups are independent; can parallelize
4. **Caching**: Store frequently-used trigonometric values

Specific optimizations:

Halton: - Use iterative bit manipulation instead of modulo operations - Consider lookup tables for small n

Sunflower: - Trigonometric functions can use CORDIC or lookup tables - Square root can use fast approximations for large datasets

All methods: - Early exit if tie group size = 1 (no jittering needed) - Batch process multiple columns simultaneously

16.5 A.5 Testing and Validation

16.5.1 A.5.1 Unit Tests

```r
## --- Helpers (lightweight assertions) ---
assert_true <- function(cond, msg = "Assertion failed") {
  if (!isTRUE(cond)) stop(msg, call. = FALSE)
}
assert_near <- function(x, target, tol = 1e-2, msg = NULL) {
  if (is.null(msg)) msg <- sprintf("Expected %.6f   %.6f (tol=%.3g)",
  x, target, tol)
  assert_true(abs(x - target) <= tol, msg)
}
```

```
## ========================
## TEST: halton_uniformity
## ========================
# not strictly needed (deterministic), included for reproducibility rituals
set.seed(1)

values <- rep(5.0, 1000)
jittered <- halton_jitter(values, epsilon = 1.0)

# Bounds (within tolerance ~ 0.01)
mn <- min(jittered)
mx <- max(jittered)
assert_near(mn, 4.5, tol = 0.01, msg = sprintf("MIN not near 4.5 (%.6f)", mn))
assert_near(mx, 5.5, tol = 0.01, msg = sprintf("MAX not near 5.5 (%.6f)", mx))

# Kolmogorov-Smirnov test vs Uniform(-0.5, 0.5)
normalized <- jittered - 5.0   # already scaled by epsilon=1.0
ks <- suppressWarnings(ks.test(normalized, "punif", min = -0.5, max = 0.5))
assert_true(ks$p.value > 0.05, sprintf("KS rejects uniformity (p=%.4f)",
ks$p.value))

cat("halton_uniformity: PASS (min ", round(mn,3), ", max ", round(mx,3),
    ", KS p=", signif(ks$p.value,3), ")\n", sep="")

## ============================
## TEST: sunflower_determinism
## ============================
values2 <- rep(7.0, 100)
result1 <- sunflower_jitter(values2, epsilon = 0.5)
result2 <- sunflower_jitter(values2, epsilon = 0.5)
assert_true(identical(result1, result2), "Sunflower jitter not deterministic")
cat("sunflower_determinism: PASS\n")

## ==================================
## TEST: intelligent_demonstrates_failure
## ==================================
values3 <- rep(10.0, 100)
jitt_bad <- intelligent_jitter(values3, epsilon = 1.0)

# First observation ~ no displacement (j=1)
assert_true(abs(jitt_bad[1] - 10.0) < 0.01,
            sprintf("First obs displacement not near 0 (%.4f)",
            jitt_bad[1] - 10))

# Last observation near max displacement
# (linear scaling to ~0.99*epsilon*cos(angle))
```

```
assert_true(abs(jitt_bad[100] - 10.0) > 0.9,
            sprintf("Last obs displacement not large enough (%.4f)",
            jitt_bad[100] - 10))

cat("intelligent_demonstrates_failure: PASS\n")

sunflower_jitter <- function(tied_values, epsilon) {
  n <- length(tied_values)
  out <- numeric(n)
  golden_angle <- 137.508 * pi / 180
  for (j in seq_len(n)) {
    angle <- (j - 1) * golden_angle
    radius <- epsilon * sqrt((j - 1) / n)
    disp <- radius * cos(angle)
    out[j] <- tied_values[j] + disp
  }
  out
}

intelligent_jitter <- function(tied_values, epsilon) {
  n <- length(tied_values)
  out <- numeric(n)
  golden_ratio <- 0.618
  for (j in seq_len(n)) {
    angle <- (j - 1) * 2 * pi * golden_ratio
    scale_factor <- (j - 1) / n            # linear (problematic)
    disp <- epsilon * cos(angle) * scale_factor
    out[j] <- tied_values[j] + disp
  }
  out
}
```

16.5.2 A.5.2 Integration Tests

```
# --- Minimal helpers ----------------------------------------------------------
assert_true <- function(cond, msg = "Assertion failed"){
if (!isTRUE(cond)) stop(msg, call. = FALSE)
}

no_missing_values <- function(df) !anyNA(df)

count_ties <- function(df) {
  # Counts duplicate occurrences (beyond the first) across numeric columns
  num_cols <- vapply(df, is.numeric, logical(1))
  sum(vapply(df[num_cols], function(x) sum(duplicated(x[!is.na(x)])),
  integer(1)))
}
```

```
# Apply a given jitterer to exact-tie groups within one numeric column
apply_jitter_to_col <- function(x, jitterer, epsilon_frac = 0.05) {
  if (!is.numeric(x)) return(x)
  rng <- range(x, na.rm = TRUE)
  span <- diff(rng); if (!is.finite(span) || span == 0) span <- 1
  eps <- epsilon_frac * span
  vals <- x
  idx <- which(!is.na(vals))
  groups <- split(idx, vals[idx])
  for (g in groups) {
    if (length(g) > 1) {
      vals[g] <- jitterer(rep(vals[g[1]], length(g)), eps)
    }
  }
  vals
}

pcp_arrange <- function(data,
method = c("halton","sunflower","intelligent"),
epsilon_frac = 0.05) {
  method <- match.arg(method)
  jitterer <- switch(method,
    halton = halton_jitter,
    sunflower = sunflower_jitter,
    intelligent = intelligent_jitter
  )

  out <- data
  for (nm in names(out)) out[[nm]] <- apply_jitter_to_col(out[[nm]],
                                          jitterer, epsilon_frac)
  out
}

# ========================
# TEST: ggpcp_integration
# ========================
data <- iris

data_halton      <- pcp_arrange(data, method = "halton")
data_sunflower   <- pcp_arrange(data, method = "sunflower")
data_intelligent <- pcp_arrange(data, method = "intelligent")

# Verify no NA values introduced
assert_true(no_missing_values(data_halton),   "NA introduced in halton")
assert_true(no_missing_values(data_sunflower), "NA introduced in sunflower")
assert_true(no_missing_values(data_intelligent),"NA introduced in intelligent")
```

```
# Verify tie resolution occurred (duplicate counts decreased)
orig_ties <- count_ties(data)
assert_true(count_ties(data_halton)      < orig_ties, "Halton did not reduce ties")
assert_true(count_ties(data_sunflower)   < orig_ties, "Sunflower did not reduce ties")
assert_true(count_ties(data_intelligent) < orig_ties, "Intelligent did not reduce ties")

cat("ggpcp_integration: PASS (no missing values; ties reduced for all methods)\n")
```

16.6 A.6 Summary Table

Algorithm	Lines of Code	Key Operation	Complexity	Recommended
Halton	~25	Bit reversal	$O(k \log k)$	Yes
Sunflower	~20	Trig + sqrt	$O(k)$	Yes
Intelligent	~18	Trig only	$O(k)$	No

Where k is the size of each tie group.

16.7 A.7 References for Appendix

Additional implementation resources:

- **Halton sequences**: Niederreiter (1992) "Random Number Generation and Quasi-Monte Carlo Methods"
- **Phyllotaxis**: Vogel (1979) "A Better Way to Construct the Sunflower Head"
- **Golden angle**: Ridley (1982) "Packing Efficiency in Sunflower Heads"
- **R implementations**: Base R documentation, Rcpp for optimization

17 References

Blumenschein, Michael, Xuan Zhang, David Pomerenke, Daniel A. Keim, and Johannes Fuchs. 2020. "Evaluating Reordering Strategies for Cluster Identification in Parallel Coordinates." *Computer Graphics Forum* 39 (3): 537–49. https://doi.org/10.1111/cgf.14000.

Day, Ross H., and Erica J. Stecher. 1991. "The Sine Illusion." *Perception & Psychophysics* 49 (4): 333–39. https://doi.org/10.3758/BF03205988.

Halton, John H. 1960. "On the Efficiency of Certain Quasi-Random Sequences of Points in Evaluating Multi-Dimensional Integrals." *Numerische Mathematik* 2 (1): 84–90. https://doi.org/10.1007/BF01386213.

Hofmann, Heike, and Marie Vendettuoli. 2013. "Common Angle Plots as Perception-True Visualizations of Categorical Associations." *IEEE Transactions on Visualization and Computer Graphics* 19 (12): 2297–2305. https://doi.org/10.1109/TVCG.2013.140.

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

———. 2009. "Parallel Coordinates: Visual Multidimensional Geometry and Its Applications." *Springer Science & Business Media*.

Johansson, Jimmy, and Camilla Forsell. 2016. "Evaluation of Parallel Coordinates: Overview, Categorization and Guidelines for Future Research." *IEEE Transactions on Visualization and Computer Graphics* 22 (1): 579–88. https://doi.org/10.1109/TVCG.2015.2466992.

Niederreiter, Harald. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. Philadelphia, PA: Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9781611970081.

Peng, Roger D. 2011. "Reproducible Research in Computational Science." *Science* 334 (6060): 1226–27. https://doi.org/10.1126/science.1213847.

Peng, Wei, Matthew O Ward, and Elke A Rundensteiner. 2004. "Clutter Reduction in Multi-Dimensional Data Visualization Using Dimension Reordering." In *IEEE Symposium on Information Visualization*, 89–96. IEEE.

Schonlau, Matthias. 2003. "The Hammock Plot: Visualizing Mixed Categorical and Numerical Data." In *Proceedings of the American Statistical Association*.

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.

Vogel, Helmut. 1979. "A Better Way to Construct the Sunflower Head." *Mathematical Biosciences* 44 (3-4): 179–89. https://doi.org/10.1016/0025-5564(79)90080-4.

Ware, Colin. 2012. *Information Visualization: Perception for Design*. 3rd ed. Waltham, MA: Morgan Kaufmann.

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