Penguins Go Parallel: a grammar of graphics framework for generalized parallel coordinate plots

Author 1[1*] Author 2[2] Author 3[3]

[1] University 1; [2] University 2; [3] University 3

October 28, 2022

Abstract

Parallel coordinate plots (PCP) are a valuable tool for exploratory data analysis of high-dimensional numerical data. The use of PCPs is limited when working with categorical variables or a mix of categorical and continuous variables. In this paper, we propose generalized parallel coordinate plots (GPCP) to extend the ability of PCPs from just numeric variables to dealing seamlessly with a mix of categorical and numeric variables in a single plot. In this process we find that existing solutions for categorical values only, such as hammock plots or parsets become edge cases in the new framework. By focusing on individual observations rather than a marginal frequency we gain additional flexibility. The resulting approach is implemented in the R package ggpcp.

XXX polyline or poly-line?

1 Introduction

Few approaches in data visualization exist that are truly high-dimensional. Most visualizations are projections of data into two or three dimensions enhanced by facetting or additional mappings to plot aesthetics, such as point size and color. Parallel coordinate plots are one of the exceptions: in parallel coordinate plots we can actually visualize an arbitrary number of variables to get a visual summary of a high-dimensional data set. In a parallel coordinate plot, each variable takes the role of a vertical (or parallel) axis; giving the visualization its name. Multivariate observations are then plotted by connecting their respective values on each axis across all axes using polylines (cf. Figure 1). For just two variables this switch from orthogonal axes to parallel axes is equivalent to a switch from the familiar Euclidean geometry to the projective space. In the projective space, points take the role of lines, while lines are replaced by points, i.e. points falling on a line in the Euclidean space correspond to lines crossing in a single point in the projective space. This duality provides a good basis for interpreting geometric features observed in a parallel coordinate plot [Inselberg, 1985].

The origins of parallel coordinate plots date back to the 19th century and are, depending on the source, either attributed to d'Ocagne [1885] or Gannett [1880]. Modern era parallel coordinate plots go back to Inselberg [1985] and Wegman [1990]. Parallel coordinate plots are used in an exploratory setting as a way of getting a high-level overview of the marginal distributions involved, identifying outliers in the data, and finding potential clusters of points. In the absence of those, Parallel Coordinate Plots are often criticized for the amount of clutter they produce, resembling a game of mikado (also known as pickup-sticks – if you are not familiar with the game, imagine spilling a box of spaghetti) rather than organized data. This clutter is sometimes mitigated by the use of α-blending [Miller and Wegman, 1991], density estimation [Heinrich and Weiskopf, 2009], or edge-bundling parallel coordinate plots [McDonnell and Mueller, 2008]. For a detailed overview of these and other techniques see Heinrich and Weiskopf [2013].

*Corresponding author: xxx

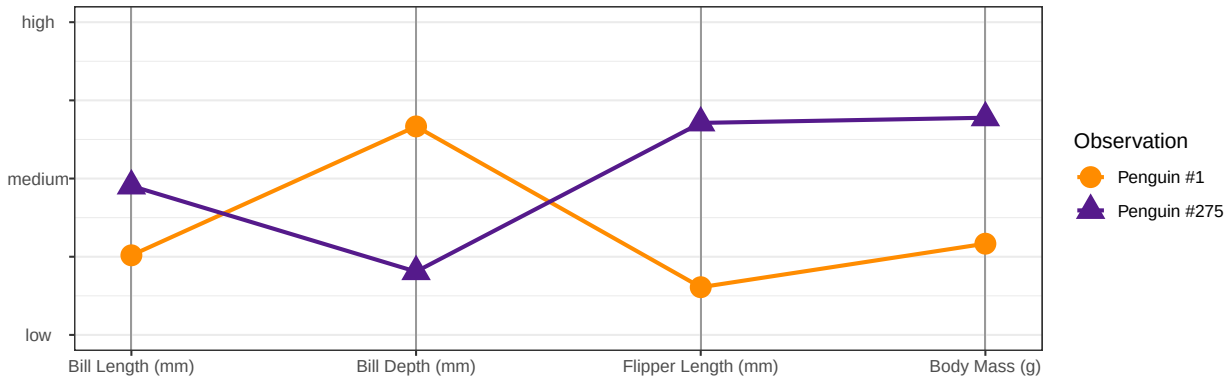

Figure 1: *Sketch of a parallel coordinate plot of two observations in four dimensions. Each dimension is shown as a vertical axis, observations are connected by polylines from one axis to the next. Two penguins from the Palmer Penguin data set (see section 5.1) were sampled for this example.*

While parallel coordinate plots are a powerful tool, using categorical variables alongside quantitative variables is a great challenge. In current solutions, levels of categorical variables are transformed to numbers and variables are then used as if they were numeric. This introduces ties into the data, and the resulting parallel coordinate plot becomes uninformative, as it only shows a mesh of lines from each level of one variable to each level of the next variable. Modifications of parallel coordinate plots have been specifically developed to deal with categorical data: parallel set plots [Kosara et al., 2006], Hammock plots [Schonlau, 2003], and common angle plots [Hofmann and Vendettuoli, 2013]; unfortunately, these solutions do not accommodate quantitative variables. Instead, they are intended for use with tabular data and show bands of observations from one categorical variable to the next. Hammock plots and common angle plots mitigate effects of the sine-illusion [Day and Stecher, 1991, VanderPlas and Hofmann, 2015] on parallel sets plots. An attempt to combine categorical and numeric variables in a parallel coordinate plot is introduced in the categorical parallel coordinate plots of Pilhöfer and Unwin [2013] by treating factor variables as numeric. Similar to parallel sets, this approach is also based on marginal frequencies for the categorical variables. Categorical parallel coordinate plots are the closest of these variations to our solution, but the `extracat` package has not been updated recently and is no longer on CRAN. In this paper, we describe a generalization of parallel coordinate plots to accommodate both categorical and quantitative variables, developed using the grammar of graphics, implemented in the R package `ggpcp`. The resulting plots can be used to gain additional insights into multivariate data compared to plots created using other available software.

The remainder of the paper is organized as follows: Section 2 introduces the `ggpcp` syntax and explains the improvements in `ggpcp` over other parallel coordinate plot software packages. Section 3 describes the data processing for parallel coordinate plots and how this wrangling is separated from the plot rendering in `ggpcp`. Section 4 discusses the rendering of parallel coordinate plots and factors such as plotting order and tie-breaking which are important for the design of PCPs. Section 5 provides three examples which highlight the use of generalized PCPs in exploratory settings.

2 Motivation and Package Usage

An important motivation for the `ggpcp` package is that other implementations of parallel coordinate plots for categorical variables make it difficult to follow a single observation across the chart. `ggpcp` alleviates this difficulty with two new developments: careful treatment of categorical variables to prevent line intersections at vertical axes, which maintains the visual ability to follow individual cases across the chart, and methods for ordering observations within categorical variables to reduce the amount of visual clutter.

Together, these features allow for easier perception of lines in generalized parallel coordinate plots: by reducing the number of intersecting lines at pivot points along the vertical axes, we allow our brains to leverage the gestalt principle of good continuation to follow one line across the plot. Reducing the number of line crossings at non-axis points simplifies the plot, reducing the overall cognitive load required to "untangle" (literally and metaphorically) the individual observations.

In addition, `ggpcp` leverages the full ggplot2 philosophy instead of using highly specific wrapper functions, allowing users to focus on the data, rather than the names of various parameters used for customization. `ggpcp` adopts tidy conventions for data wrangling, separating the necessary data manipulation to generate a parallel coordinate plot from the visual rendering. Scaling, ordering of cases, and the arrangement of the parallel axes are completed using `pcp_select`, `pcp_arrange`, and `pcp_scale`,respectively; the resulting data frame is then passed directly into the familiar `ggplot()` call. During the plotting state, the only modification from default `ggplot2` syntax is the use of `aes_pcp()` in place of `aes()`; this is necessary to handle the multiple axes in a parallel coordinate plot while maintaining the ability to map all other variables of the original data frame to aesthetics such as linetype and color. The user has complete control over layers such as PCP lines (`geom_pcp`), labels (`geom_pcp_labels`), and boxes around categorical variables (`geom_pcp_boxes`), but there are additional advantages to the use of `ggplot2`. Users can also augment their parallel coordinate plots with additional information, such as boxplots or violin plots, with standard ggplot2 syntax.

One of the strengths of `ggplot2` is its handling of small multiple plots with `facet_grid` and `facet_wrap`; these functions are fully supported in `ggpcp`. In addition, leveraging `ggpcp` on `ggplot2` expands the functionality available to users without much additional code, thanks to other packages such as `plotly` [Sievert, 2020] which leverage `ggplot2` to create interactive graphics for the web and `gridSVG` [Murrell and Potter, 2020] which exposes vectorized graphics to allow programmable changes, such as elementary interactions with graphical elements.

Listing 1: *A demonstration of ggpcp's data wrangling and plotting API.*

```
1  pcp <- penguins %>%                          # data management:
2    filter(!is.na(sex)) %>%                    #   seamless integration with tidy workflow
3    pcp_select(4,3,5:6, sex, species) %>%      #   variable selection, see section 3.1
4    pcp_scale(method="uniminmax") %>%          #   setting scaling method, see section 3.2
5    pcp_arrange() %>%                          #   arranging categorical data
6    ggplot(aes_pcp()) +                        # plotting the chart:
7      geom_pcp_axes() +                        #   vertical lines for axes
8      geom_pcp(aes(colour = species),          #   parallel coordinate line  segments
9               alpha = 0.8, overplot="none") +
10     geom_pcp_labels()                        #   labels for categorical variables
```

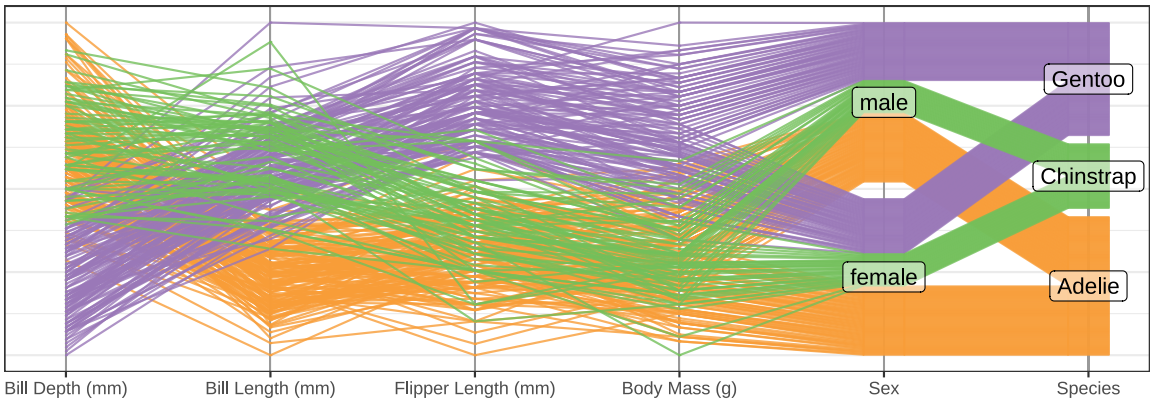

Figure 2: *The code in Listing 1 describes the data handling and basic structure of this parallel coordinate plot with both categorical and continuous data shown on vertical axes. Some minor modifications have been made to the plot for aesthetic purposes.*

In addition, ggpcp leverages the full ggplot2 philosophy instead of using highly specific wrapper functions, allowing users to focus on the data, rather than the names of various parameters used for customization. ggpcp adopts tidy conventions for data wrangling, separating the necessary data manipulation to generate a parallel coordinate plot from the visual rendering, as shown in Listing 1. Scaling, ordering of cases, and the arrangement of the parallel axes are completed using `pcp_select`, `pcp_arrange`, and `pcp_scale`,respectively; the resulting data frame is then passed directly into the familiar `ggplot()` call. During the plotting state, the only modification from default `ggplot2` syntax is the use of `aes_pcp()` in place of `aes()`; this is necessary to handle the multiple axes in a parallel coordinate plot while maintaining the ability to map all other variables of the original data frame to aesthetics such as linetype and color. The user has complete control over layers such as PCP lines (`geom_pcp`), labels (`geom_pcp_labels`), and boxes around categorical variables (`geom_pcp_boxes`), but there are additional advantages to the use of `ggplot2`. Users can also augment their parallel coordinate plots with additional information, such as boxplots or violin plots, with standard ggplot2 syntax.

An example parallel coordinate plot is shown in Figure 2, along with the ggpcp code to generate the plot in Listing 1. We can see that Gentoo penguins have smaller bill depth and larger flipper length and body mass than Chinstrap and Adelie penguins. Chinstrap penguins have longer bills than Adelie penguins, but are similar to Adelie penguins across most other measurements. Males tend to be larger than females across all three species. In addition, it is clear from Listing 1 that the data management process (lines 2-5) is entirely distinct from the plotting process in lines 6-9. This separation makes plots generated with ggpcp easy to prepare, use, and customize.

3 Data management

One of the ideas behind this re-implementation of parallel coordinate plots is to expose parallel coordinate plots at a functional level. Rather than using a single function with parameters controlling every aspect, we separate the data management from the visual rendering. In particular, we separate out the data management into three parts:

1. Variable selection and reshaping data,
2. Scaling of axes, both at the individual level and in the relationship of the axes to each other, and
3. Treatment of ties in categorical axes.

The code corresponding to each of these steps is shown in lines 3-5 of Listing 1.

The modularization of the data wrangling process has the additional advantage of laying out the necessary elements in successive steps. Some of these steps are optional: scaling variables might not be necessary if all variables are already on the same scale (i.e. method 'raw' in GGally); similarly, using pcp_arrange to break ties is only necessary if categorical variables are present and we want to spread these observations out so that individual lines are visible. In addition, by exposing these elements of the pcp data wrangling process, we allow users to create additional functions for handling these tasks.

The treatment of ties is an aspect not generally addressed in the original parallel coordinate plots of Inselberg [1985] and Wegman [1990]. We have found a need to deal with ties, because ties are visually the main obstacle of allowing the viewer to follow an observation from axis to axis through the high-dimensional space. If we can track a single observation through the high-dimensional space, we have the ability to look beyond the two-variable associations of adjacent axes. This allows users to more easily summarize main trends and identify observations which do not follow those trends. When ties cannot be separated and users cannot follow individual observations, higher-dimensional insights are next to impossible.

3.1 Variable Selection and Order of the Variables

One of the biggest strengths of the Grammar of Graphics is its mapping between data variables and visual aesthetics. In standard plots any mapping is a function between one data variable and one aesthetic. In a parallel coordinate plot, this one-to-one mapping between data and plot aesthetics is seemingly turned into a one-to-many mapping between arbitrarily many data variables to the x axis. By transforming the wide form of the data set into a long form [Wickham, 2014, 2021], we obtain a one-to-one mapping to a now discrete x axis consisting of the (names of the) original data variables.

From the user's perspective, this data reshaping is data selection; the data wrangling takes place behind the scenes in `pcp_select(data, ...)`, which selects the variables to be included in the parallel coordinate plot. Variables can be specified by any combination of the following methods:

- position, e.g. `1:4, 7, 5, 4`,
- name, e.g. `class, age, sex, aede1:aede3` or
- using pattern selectors, e.g. `starts_with("aede")`, see `?tidyselect::select_helpers`

Variables can be selected multiple times and will then be included in the data and the resulting plot multiple times. Note that the order in which variables are selected determines the order in which the corresponding axis is drawn in the parallel coordinate plots. `pcp_select` transforms the selected variables to long form and embellishes the data set with a number of additional variables. All of the newly created and added variables start with the prefix `pcp_`:

- `pcp_id`: integer variable identifying each observation in the original dataset. This variable is used as the grouping variable to identify which values should be connected by a line segment in the parallel coordinate plot.

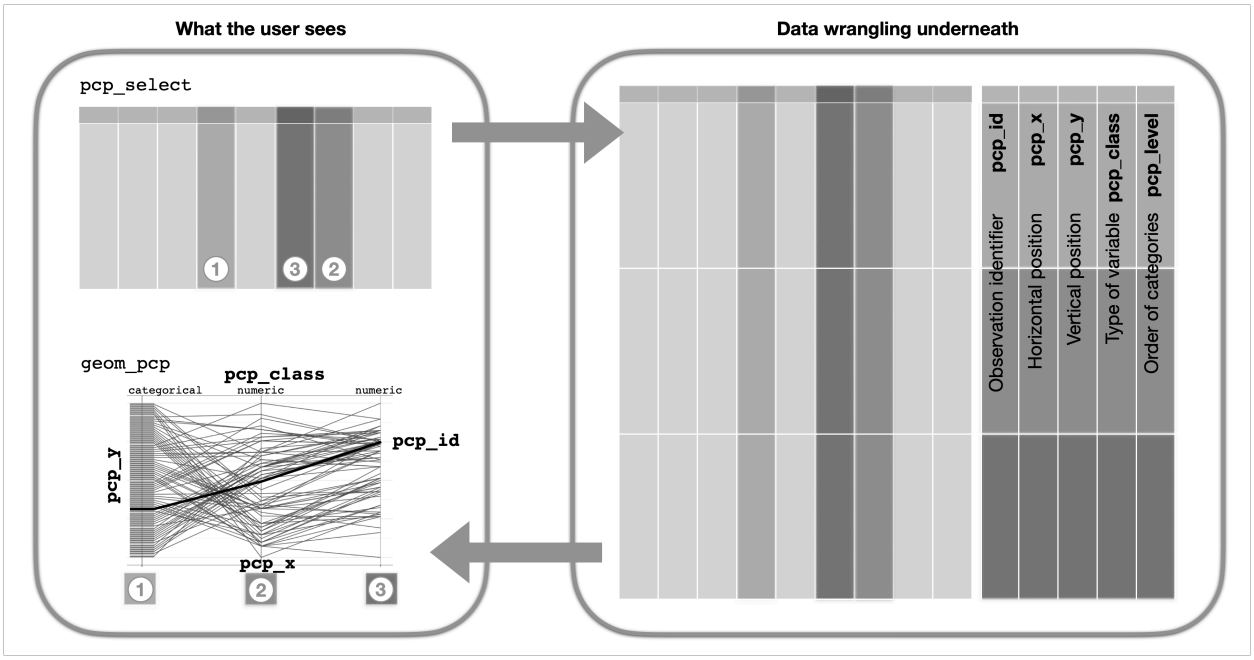

Figure 3: *The user selects a set of three variables (top left). On the right, an overview of the data wrangling step before a parallel coordinate plot can be drawn (bottom left). Note that the order in which variables are selected is reflected in the order in which variables are included in the parallel coordinate plot.*

- `pcp_x`: discrete variable consisting of the names of the selected variables in the order that they were selected - this is the order in which the variables will be included in the plot.
- `pcp_y`: numeric variable containing the values of all of the selected variables. In case a selected variable is not numeric, it is converted to a factor variable and the (numeric) factor levels are saved in `pcp_y`.
- `pcp_class`: character variable containing the class information of a selected variable.
- `pcp_level`: character variable containing the factor levels of selected data variables. In case of numeric variables, the data values are stored (in textual form). The ordering of factor variables will be discussed below but it is implemented using this added variable.

As a consequence of these design decisions, users have several ways of performing different tasks within the flow of generating data for a parallel coordinate plot. For instance, users can reorder variables using `pcp_select` or after variable selection using the `pcp_x` variable. Motivations for reordering factors in parallel coordinate plots are discussed in more detail in Section 4.1.

Similarly to previous implementations of parallel coordinate plots which attempted to accommodate categorical variables, we treat factor variables as variables with labels and an associated (numerical) ordering of those labels. Whenever we assign a numeric value to the ordering, we refer to the associated score, which is an integer value from one to the number of categories, if not specified explicitly otherwise. Ordered factors are plotted from the lowest level upwards. If a factor legend is included, it will need to be reversed to match this order by using `guides(color = guide_legend(reverse=TRUE))`, as shown in the example in Section 5.2. Where `ggpcp` differs from previous implementations of parallel coordinate plots is in the assignment of numerical values to individual observations within a factor level. This ordering is discussed further in Section 4.1.

3.2 Scaling

`pcp_scale(data, method)` scales the values on each axis and determines the relative relationship of the axes to each other. The `method` argument is a character string specifying the method to be used when

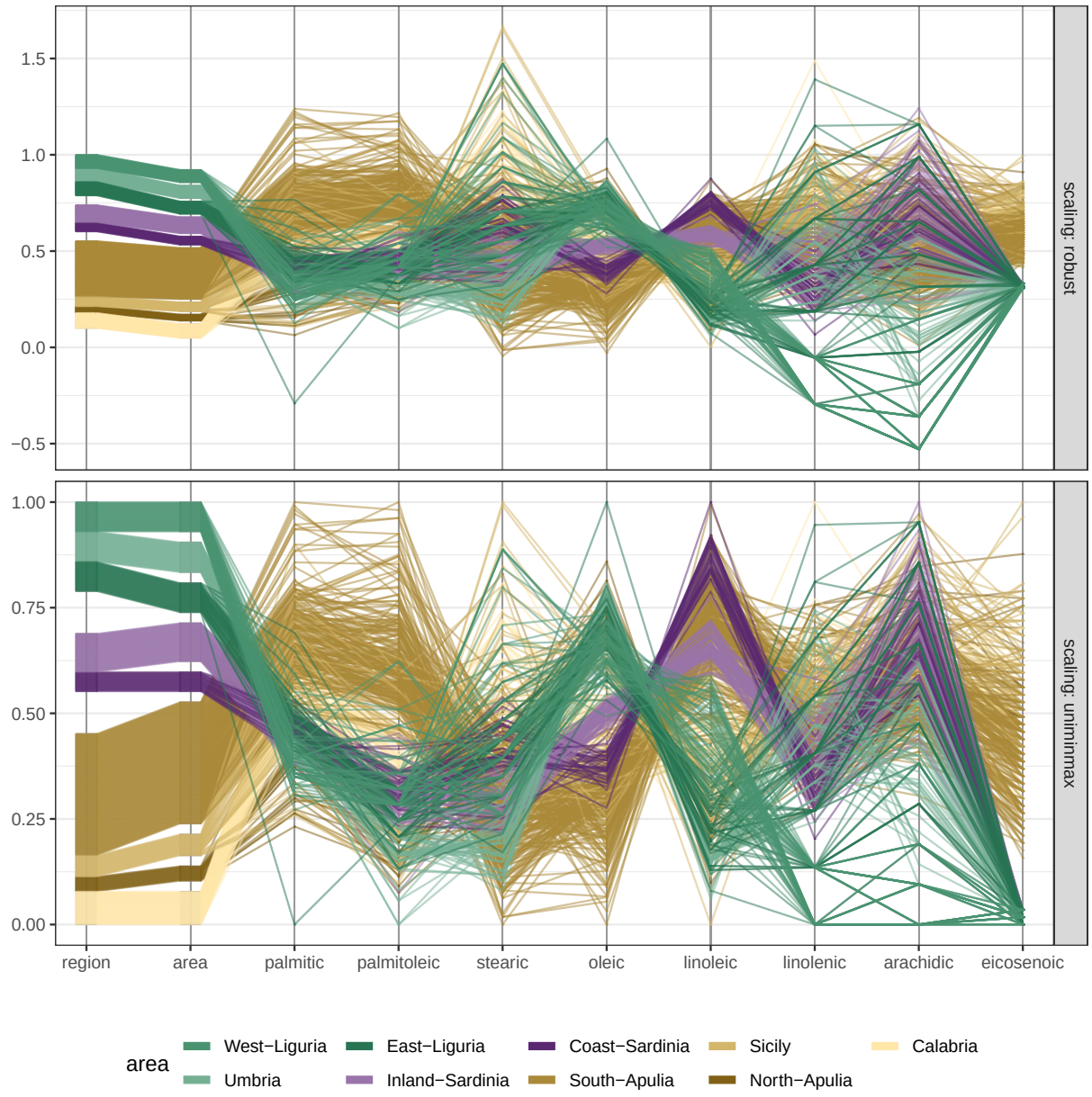

Figure 4: *Two scaling methods showing fatty acid compositions of olive oils from different regions in Italy, areas within each region are colored using similar hues within region (green for Northern Italy, purple for Sardinia, and tans for Southern Italy).robust scale: median to 0.5, see docu. Looks right but might be cryptic. check units XXX The two scaling methods roughly allow the same conclusions.*

transforming the values of each variable onto a common y axis. By default, the method `uniminmax` is chosen, which univariately scales each variable onto a range of [0,1] with the minimum at 0 and the maximum at 1. `globalminmax` maps the values across all axes onto a an interval of [0,1]. This method should only be used if the values across all variables are comparable. The method `robust` normalizes values univariately by mapping the median value to 0.5 and a robust 95% confidence interval (based on the median absolute deviation) to an interval of 0 to 1. Values outside this range Figure 4 indicate a variability in the measurements larger than that of a normal distribution, as can be seen for several acid measurements.

Figure 4 shows two of the scaling methods applied to the olive oil data [Forina et al., 1983, Wickham et al., 2011]: Measurements of fatty acids in 572 olive oils from three different regions in Italy are visualized as parallel coordinate plots. Similar to the findings by Cook and Swayne [2007], we see that eicosenoic acid is only found in increased quantities in olive oils from Southern Italy. Quantities of oleic and linoleic acids allow a separation between olive oils from Sardinia and Northern Italy. Both scaling methods enable us to find these conclusions. While uniminmax scaling uses the space allotted to the chart most efficiently, the robust normalization method emphasizes the heavy tails and skewness of some of the measurements, such as the percentages of stearic and arachidic fatty acids.

4 Visual Rendering

4.1 Breaking ties on categorical axes

One of the primary advantages of `ggpcp` over previous parallel coordinate plot software packages is that `ggpcp` handles categorical and continuous data in a way that allows users to trace a single observation through the projective space. This is accomplished through a tie-breaking algorithm: different categorical levels are grouped along the vertical axis in boxes proportional to the number of cases in each level. Within the box for a level, individual observations are arranged so that visual clutter is minimized and individual cases can be followed.

Figure 5 shows several approaches of dealing with categorical variables in parallel coordinate plots. The left-most panel shows two categorical variables and the typical net of lines that forms between them in an original parallel coordinate plot. The other three panels show three different approaches of breaking the ties resulting from the categorical variables, with our favored solution shown on the right: all observations are spaced out evenly. This results in a natural visualization of the marginal frequencies along each axis (additionally enhanced by the light gray boxes grouping observations in the same category). The ordering of the observations within the level is such that a minimal number of line crossings occurs between the axes. This method of dealing with categorical variables is the one we propose in the generalized parallel coordinate plot. While it is aesthetically pleasing, it also allows us, in the spirit of the original parallel coordinate plots, to follow an individual observation from left to right through the plot even for categorical variables. The other two solutions in the middle panels of Figure 5 show two intermediate solutions of breaking ties in categorical variables: jittering and equi-spaced (unordered) values.

When extended over multiple axes, the equispaced tie-breaking solution that reduces line crossings requires hierarchical sorting, which is implemented in the `ggpcp` function `pcp_arrange(data, method, space)`. The two implemented methods are `"from-left"` and `"from-right"`, meaning that ties are broken using a hierarchical ordering determined by variables' values from the left or the right, respectively. The parameter `space` specifies the amount of the y axis to use for space between levels of categorical variables. By default, 5% of the axis is used for spacing. While hierarchical sorting requires additional computations relative to the jittering or equally spaced solutions in Figure 5, this extra processing serves as "external cognition" [Scaife and Rogers, 1996] - the additional computer time reduces the cognitive load required to untangle the data as displayed in the chart.

Mart and Laguna [2003] discusses the NP hard problem of ordering categories to minimize line crossing. In PCPs, the category order is determined by the factor order (or the numerical scale in the continuous case); these line crossings are not avoidable through within-category sorting and are a function of the data

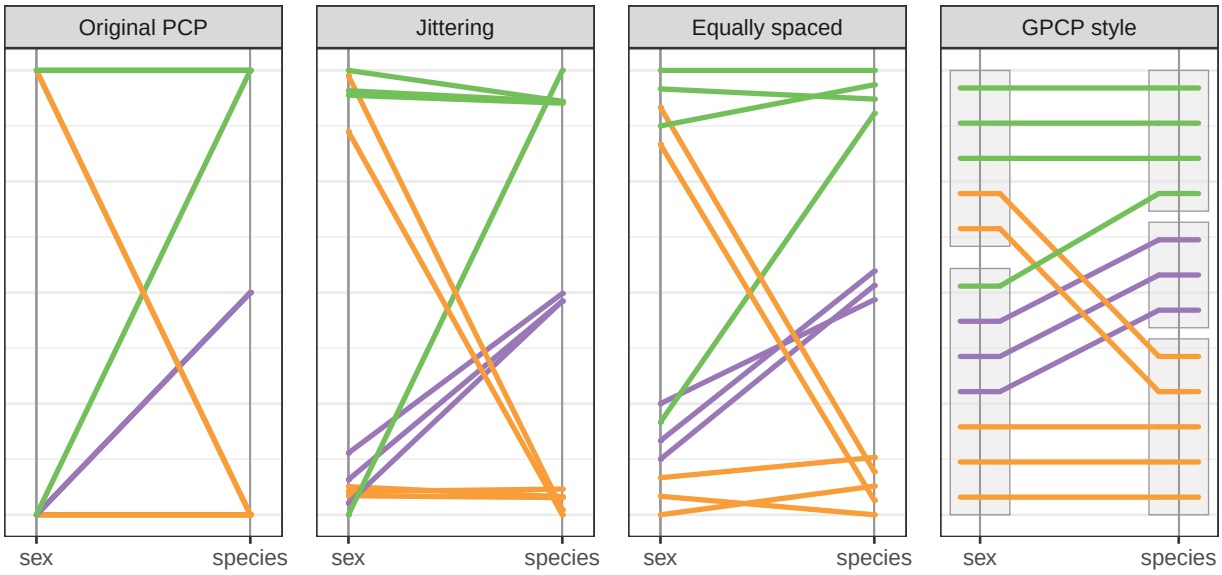

Figure 5: *Using 12 randomly sampled penguins from the Palmer penguin data, we show four different approaches of dealing with categorical variables: the panel on the left shows the typical net of lines resulting from categorical variables in regular parallel coordinate plots. In the other three panels, ties in categorical levels are broken using different approaches (from left to right): jittering, equi-spaced line segments and ordered equi-spaced line segments are shown.*

itself. Hierarchical sorting of individual observations minimizes extraneous crossings within these categories in cases where there are multiple similar observations, contributing to the Gestalt of 'common fate' among individuals with similar values across a number of PCP axes.

4.2 Variable Ordering and Transformations

There are many different goals one might have when drawing a PCP; these goals shape any effort the designer might put into optimization of visual appearance. For instance, the order of factor levels is an important consideration if the goal is to minimize line crossings and thus the visual complexity of the parallel coordinate plot. As previously discussed, some line crossings can be removed by sorting, however, others can only be removed through reordering of factor levels. While *automatic* sorting of factor levels is computationally difficult and statistically undesirable given that many factors have some implicit or explicit ordering that should not be automatically optimized, reordering factors can reduce the number of line crossings to produce a simpler and more comprehensible PCP. For example, in the last panel of Figure 5, a reordering of the second factor so that the dark (purple) lines are on the bottom could reduce the overall number of line crossings to just two crossings, once the hierarchical sorting is updated to accommodate the new factor order.

As briefly discussed in Section 3.1, users can transform individual variables, reordering factors or reversing an axis, using a mutate statement before variable selection. Univariate transformations like these may be useful to reduce the overall visual complexity of a parallel coordinate plot by reducing the number of negatively correlated axes and crossing lines which are hard to follow. An example showing the benefits of reordering and transforming variables for visual clarity is provided in Figure 8.

4.3 Line Segment Plotting Order

One of the primary advantages of the generalized approach to dealing with categorical variables is the ability to follow a single observation throughout the plot. As the number of observations increases, this

becomes less feasible because of overplotting of line segments, particularly for larger data sets. As more observations and line segments are drawn, more lines cross each other, increasing the effort required to follow a poly-line from one side of the plot to the other.

Coloring by groups and utilizing α-blending improves the readability of plots. However, the order of drawing the cases may affect what can be seen due to overplotting.

As a countermeasure, the order in which line segments are plotted should be carefully chosen. The parameter `overplot` defaults to option "small-on-top", where groups are plotted in order of size from largest to smallest so that the smallest is plotted last. An alternative setting, "none", is very flexible, but requires the user to specify the order they want before plotting–or just accept the current order of the dataset. The use and effect of `overplot` are demonstrated in Listing 2 and Figure 6, respectively.

```
1  pcp_df <- penguins %>%
2    arrange(sex) %>% # NA last = top of PCP axis
3    pcp_select(sex, species) %>%
4    pcp_scale(method="uniminmax") %>%
5    pcp_arrange()
```

```
6  ggplot(pcp_df, aes_pcp()) + geom_pcp_axes() + # draw lines in the provided order
7    geom_pcp(aes(colour = species), overplot = "none") +
8    geom_pcp_labels()
```

```
9  ggplot(pcp_df, aes_pcp()) + geom_pcp_axes() + # draw the smallest category last
10   geom_pcp(aes(colour = species), overplot = "small-on-top") +
11   geom_pcp_labels()
```

Listing 2: *The `overplot` parameter can be used to control the order in which lines are plotted, affecting the visual appearance and emphasis of parallel coordinate plots. The line 2 specifies the order of the dataset; lines 6-8 plot the data using the user-specified ordering while lines 9-11 plot the data using the default 'small-on-top' ordering.*

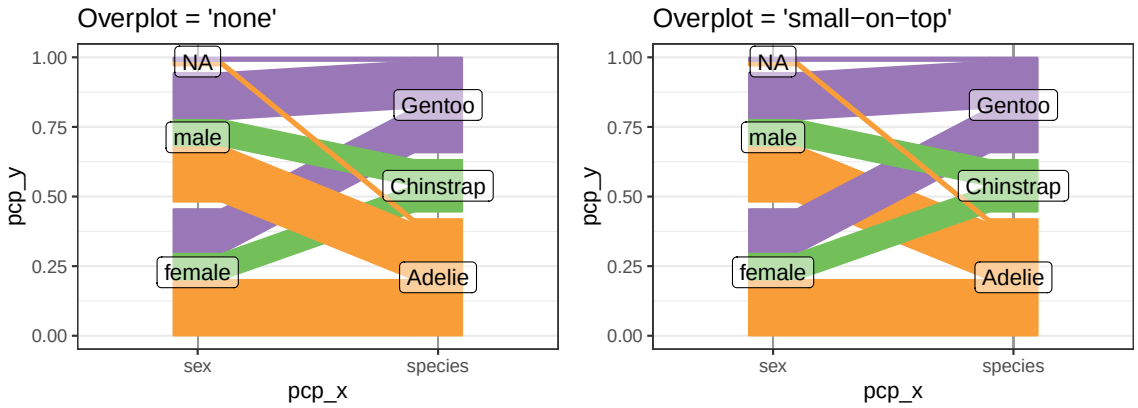

Figure 6: *The code in Listing 2 generates two parallel coordinate plots. The plot on the left uses the ordering of the dataset to determine line plotting order; as a result, lines with sex = NA are plotted last (on top). On the right, we use the "small-on-top" default; this ensures that the smallest category, Chinstrap, is plotted last.*

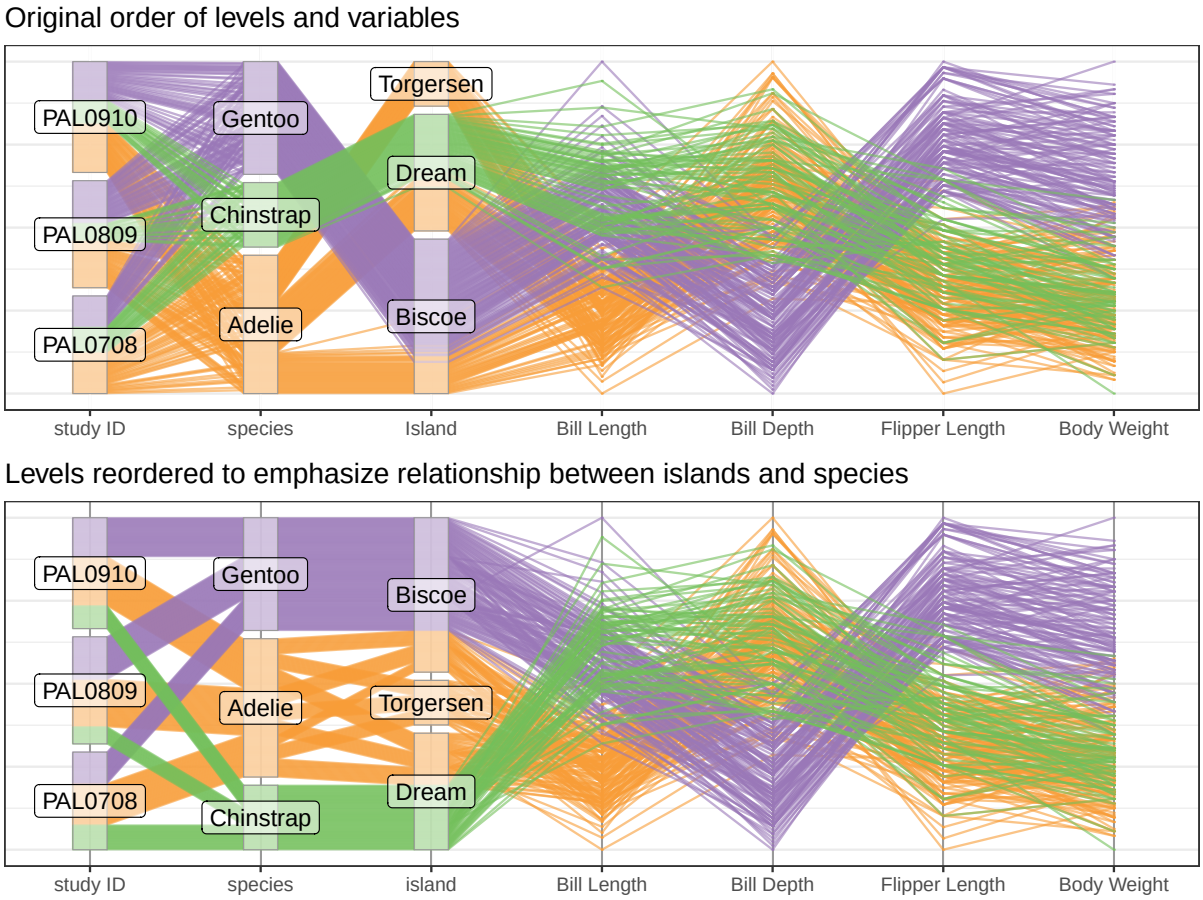

Figure 7: *Both of the levels of the island and the species variable re-ordered to reflect that two of the species are each only found on one island.*

5 Examples

5.1 Palmers Penguins

Several aspects in Parallel Coordinate Plots depend on orderings: the order of variables along the x axis, the order of levels in a categorical variable, the orderings of cases within categorical variable levels, and the order in which lines are drawn. Orderings should therefore (a) have good defaults, and (b) be easily changeable.

The top of Figure 7 shows a generalized parallel coordinate plot of the Palmer penguins data [Horst et al., 2020]. The numeric data consists of body measurements of three species of penguins: bill length, bill depth, flipper length, body weight. Adelie penguins generally have smaller bill lengths than the other two species, while Gentoo penguins can be distinguished by their relatively large flipper lengths. The bottom of Figure 7 shows the effect of re-ordering the levels of both the 'species' and the 'island' variables in the generalized parallel coordinate plots. This re-ordering of factor levels has the effect of emphasizing that Gentoo penguins and Chinstrap penguins are each found on only one island, while Adelie penguins are found on all three islands. In addition, only after levels of 'island' and 'species' are re-ordered can we see that for each species the numbers of penguins in the three years of the study (the study ID variable) were roughly the same.

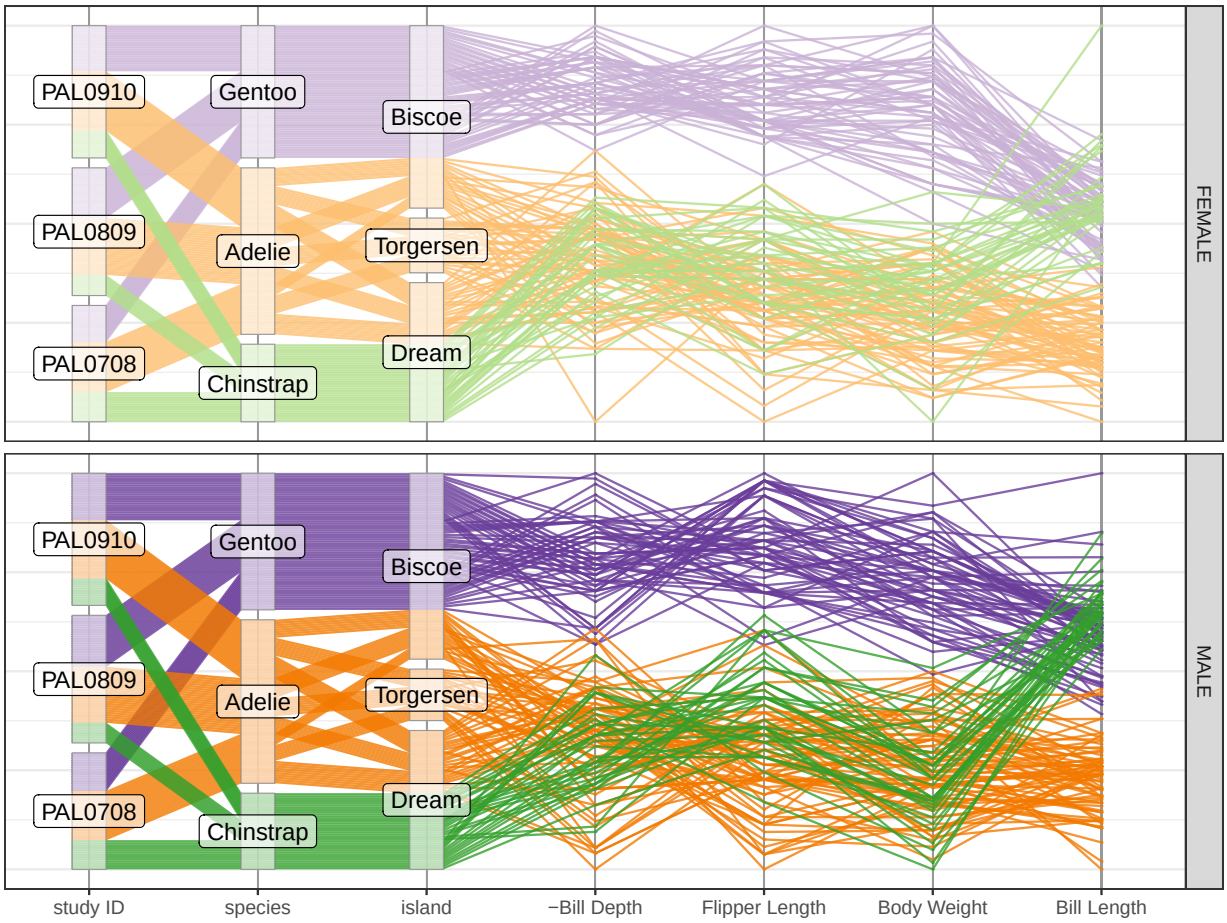

Figure 8: *Changing the order of the variables along the x-axis emphasizes the differences in body measurements between the species.*

Distinguishing species

Factor level ordering is but one consideration when constructing parallel coordinate plots. It is also important to carefully order the variables on the x-axis, as shown in Figure 8, where the variables have been re-ordered from Figure 7 to allow the viewer to identify which body measurements distinguish the species. In addition to the re-ordering, the axis for bill depth has been reversed. Both changes help to separate the species. Gentoo penguins have the lowest bill depth, while generally having the longest flippers and largest mass. Reversing the axis for bill depths aligns the smallest bill depths with the longest flippers, moving Gentoo penguins closer together as a group. The plot shows that the Gentoo penguins are bigger, that Gentoo and Chinstrap are both only found on single islands, and, finally, that Adelie and Chinstrap are distinguished by the lengths of their bills.

As ggpcp uses the ggplot2 API, faceting is fully supported. Figure 8 is faceted by gender: while the results are the same for the two sexes, any variability of body measures due to sex is removed from the plot by facetting. This makes the results stand out more. Interestingly, some potential outliers that were not visible previously now become visible. Note for example the two Gentoo males with particularly short flippers, and the Chinstrap female with an exceptionally long bill.

Determining sex

Figure 9 shows that within each species, the males tend to be larger in size and heavier than the females.

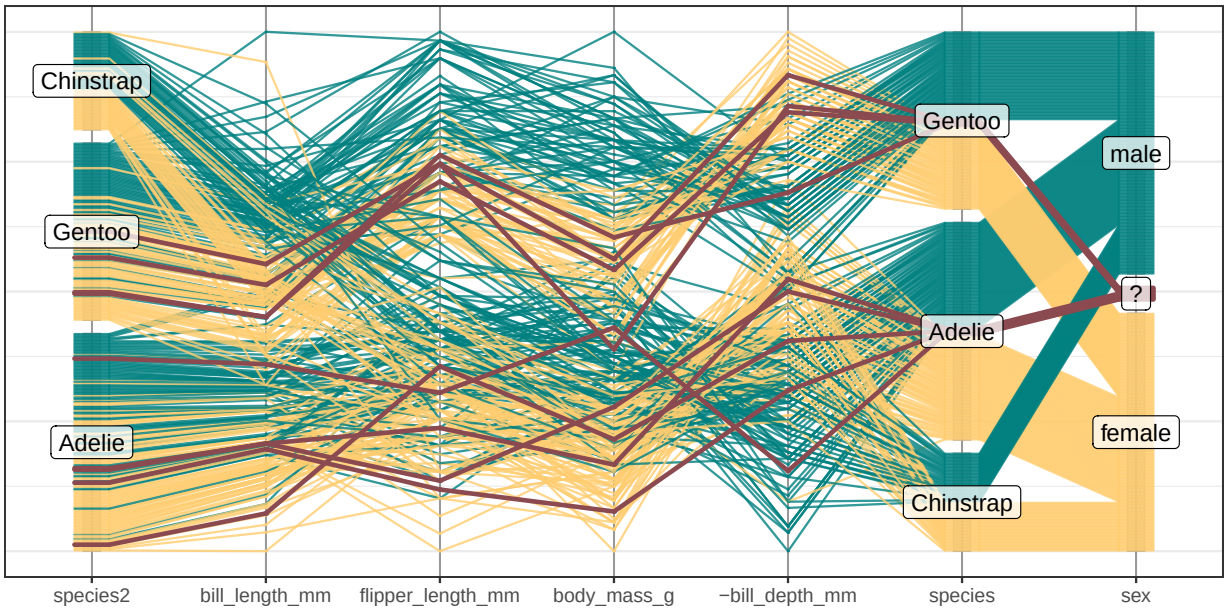

Figure 9: *Generalized Parallel Coordinate Plot of the Palmer penguins data with sex of penguin mapped to color. Dark lines represent penguins for which sex could not be determined. We see that researchers were able to sex all of the Chinstrap penguins. Note that species is included twice (with different order of the levels). XXX reason for that here or in the text?*

For several of the penguins, sex could not be determined because either the sexing primer did not amplify or no blood sample was obtained [Gorman et al., 2014]. These penguins are represented by dark lines. Comparing these penguins' body measurements to those of the other penguins, we can make suggestions regarding their sex.

In Figure 10 we explore this idea a bit further. This figure is based on the same data as Figure 9, however, we exclude Chinstrap penguins as researchers were able to sex all of those penguins. The body measurements of all sexed penguins are summarised by two ribbons for each sex and species. The inner ribbons are bounded by the 25% and the 75% percentile values on each axis. The lighter ribbon covers 95% of observations on each variable. We use these ribbons to reduce the noise introduced by individual lines. Body measurements of the unsexed animals are represented as line segments on top of the ribbons. This helps us to evaluate and assess the lines drawn for individual, unsexed penguins within the context of the marginal distributions (in this case their putative sex and species).

While we facet both by species and sex, note that the axes are re-scaled within each species to make use of the full range in y. However, we use the same scale between the two sexes of each species. This different treatment of faceting variables is achieved by the use of a `group_by` statement before `pcp_scale`. Listing 3 shows the code for prepping the data shown in Figure 10. By grouping on species but not on sex (line 9), data is being rescaled within species but the same scaling is used across males and females. Measurements for unsexed animals are shown as line segments on top of inter-quantile ribbons of both sexes. Viewers are encouraged to draw a conclusion about an animal's sex based on their values within the (2d density) context of their species and putative sex. Statistically, this comparison relates to a likelihood ratio test: the viewer is asked to make an assessment of the likelihood to observe the measurements of an animal under each of the two competing hypotheses of sex.

Listing 3: *Code to prepare data for Figure 9 by relabling penguins with NA sex as '?' and ordering sex so that penguins of unknown sex are between the male and female labels.*

```
1  penguins_pcp <- penguins %>%
2    filter(species != "Chinstrap") %>%                  # no unsexed animals in Chinstrap
3    mutate(
4      sex = ifelse(is.na(sex),"?", as.character(sex)),  # make assignment more readable
5      sex = factor(sex, levels = c("female", "?", "male"))
6    ) %>%
7    filter(!is.na(body_mass_g)) %>%
8    pcp_select(6:5, 3:4) %>%
9    group_by(species) %>%                               # re-scale by species
10   pcp_scale() %>%
11   pcp_arrange()
```

Chinstrap penguins are excluded (line 2) because all of their individuals in the data have a sex assigned. The general pattern of measurements of the Gentoo penguins suggests that three of the four individuals with missing sex information are female (the three with the lowest bill depth). The fourth animal has an exceptionally deep bill, however, all other measurements suggest that this animal, too, is female. For further evidence, we find from the original data that their nest partners are all sexed as male; this additional information is shown in Figure 10. While assuming that nest partners are male and female is not a perfect method, in particular, for penguins, which have been shown to live in same-sex partnerships, in all three of the studies considered for this data only nests with breeding successes have been considered. More details can be found in Gorman et al. [2014]. For Adelie penguins determining sex is not quite as clear-cut, but based on body mass and bill length measurements the three lightest penguins might be female, while the heaviest one could be male. The fifth penguin, marked '?', exhibits measurements that are neither typically male not typically female. We can assess these inferences using the additional information that four of the five unsexed Adelie penguins are nest partners. The un-partnered penguin is the lightest and has measurements which are more consistent with female penguins. The Adelie penguin indicated by ? is the partner of a female penguin (pair B) and might be assumed to be male. The remaining pair of unsexed Adelie penguins (pair A) consists of a putative male and female; this is consistent with the breeding pair assumption in the study.

5.2 Getting a second, third, ... and seventh opinion

Figure 11 shows data from Agresti [2002] published as part of the `poLCA` package [Linzer and Lewis, 2011]. Seven pathologists were asked to assess the same 118 slides for the presence or absence of carcinoma in the uterine cervix. Binary responses for each slide were recorded (yes/no). Pathologists all agreed on about 25% of slides, which they considered to be carcinoma free, and a further 12.5% of slides, which were considered to show carcinoma by all pathologists. For the remaining 62.5% of slides there was some disagreement and it is clear that this disagreement is not random. The pathologists have been ordered from left to right from the fewest number of overall carcinoma diagnoses made to the highest number. This shows a strong level of agreement between adjacent axes. Note, in this example we do not need to scale the variables. Aside from the actual scale the values are ordered in the same way.

Landis and Koch [1977, Table 1] allow us a closer look at this data. The pathologists evaluated the slides using five levels from 1 to 5, given as: (1) Negative, (2) Atypical Squamous Hyperplasia, (3) Carcinoma in Situ, (4) Squamous Carcinoma with Early Stromal Invasion, and (5) Invasive Carcinoma. Agresti [2002] classified levels 1 and 2 as "no" and levels 3 to 5 as "yes". Figure 12 gives an overview of this more detailed data. The different pathologists are drawn in the same order as in Figure 11. The results for each scan are colored by the overall average score (rounded to the closest integer). Compared to the previous figure, Figure 12 shows more variability between pathologists' evaluations, but only few scans have vastly different scores assigned to them. Pathologist C, in particular, rates two scans as negative, that all other

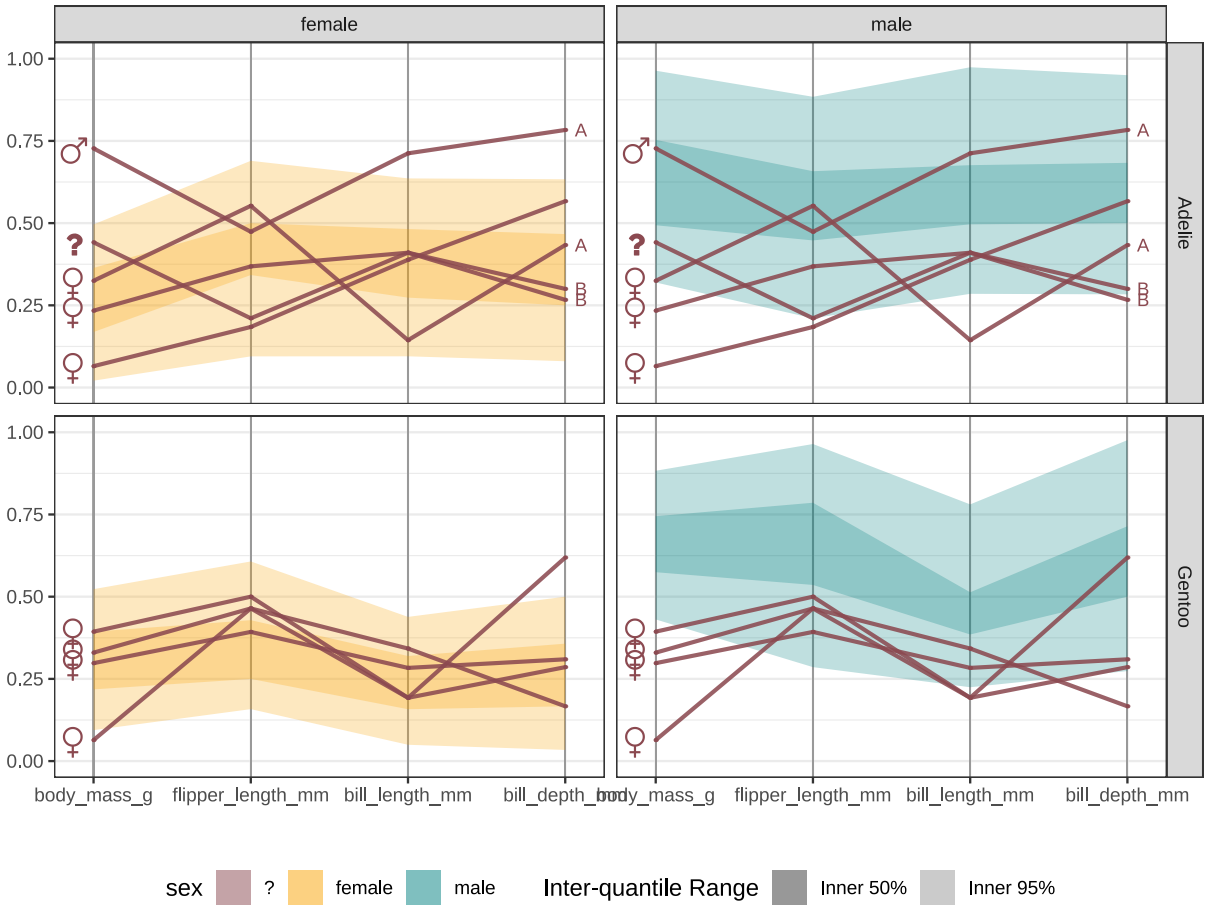

Figure 10: *Closer investigation of non-sexed Adelie and Gentoo penguins. The group_by call before pcp_scale is responsible for scaling by species while the same scale is kept across sex within species. Penguins without assigned sex (based on blood markers) are drawn on top of both sexes. The labels to the left of the ribbons are our best guess at a penguin's sex based on body measurements of other penguins of the same species. The letters on the right indicate nests – two penguins with the same letter share the same nest.*