Expert Elicitation

Seth Nathaniel Linga

2026-03-01

Contents

1 Load packages

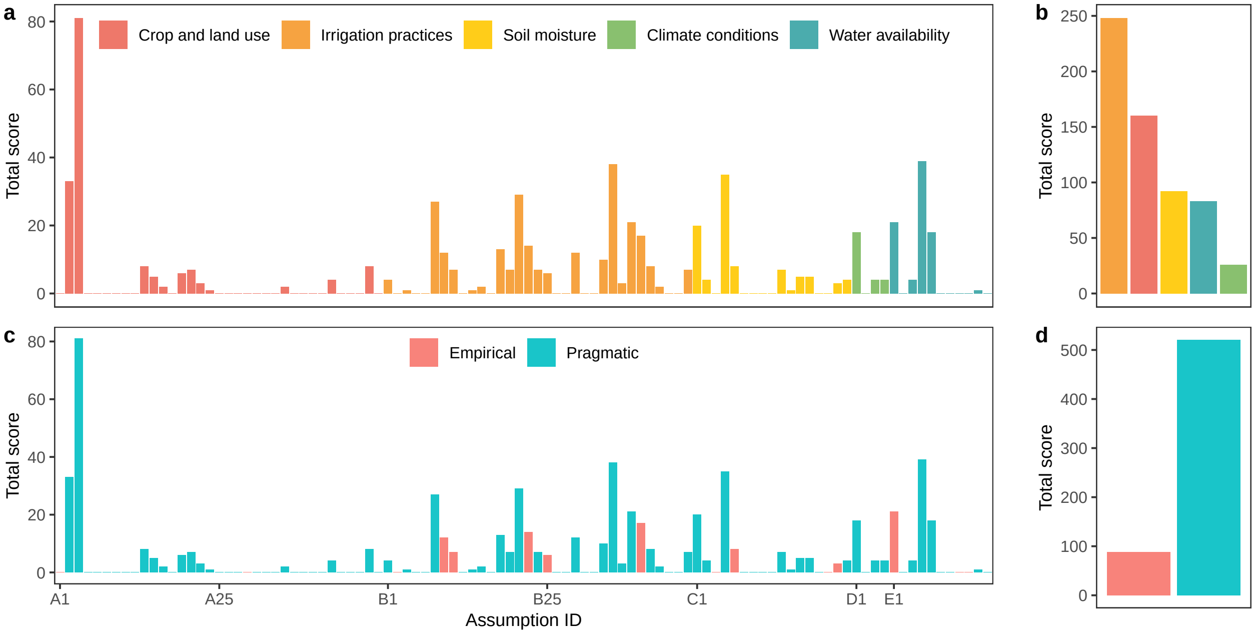
```r
library(tidyverse)
library(readxl)
library(ggh4x)
library(gridExtra)
library(grid)
library(networkD3)
library(htmlwidgets)
library(ggrepel)
library(data.table)
library(ggtext)
library(GGally)
library(sensobol)
```

2 Set directory and read data

```r
# Load data ####################################################################

setwd("~/Library/CloudStorage/OneDrive-UniversityofBirmingham/dawn/expert_elicitation/expert_elicitatio

score <- read_excel("data/workshop_data.xlsx", sheet = "model_assumptions")
pedigree <- read_excel("data/workshop_data.xlsx", sheet = "pedigree_mod")
pedigree_comb <- read_excel("data/workshop_data.xlsx", sheet = "pedigree_comb")
```

3 Score distribution

```r
# Calculate total score for each assumption ####################################
score <- score %>%
  rowwise() %>%
  mutate(total_score = sum(c_across(4:14), na.rm = TRUE)) %>%
  ungroup()

# Group assumptions by theme ###################################################
score <- score%>%
  mutate(facet = case_when(
    str_starts(code, "A") ~ "Crop and land use",
    str_starts(code, "B") ~ "Irrigation practices",
    str_starts(code, "C") ~ "Soil moisture",
    str_starts(code, "D") ~ "Climate conditions",
    str_starts(code, "E") ~ "Water availability",
    TRUE ~ "Other"),
    facet = factor(facet, levels = c("Crop and land use",
                                     "Irrigation practices",
                                     "Soil moisture",
                                     "Climate conditions",
                                     "Water availability")))

# Summarize total score per facet ##############################################
facet_summary <- score %>%
  group_by(facet) %>%
  summarise(total_score = sum(total_score, na.rm = TRUE)) %>%
  arrange(desc(total_score))

# Summarize total score per nature ############################################
nature_summary <- score %>%
  group_by(nature) %>%
  summarise(total_score = sum(total_score, na.rm = TRUE)) %>%
  arrange(desc(total_score))

# Plot A: Total score per assumption (facet) ##################################
plot_a <- ggplot(score, aes(x = code, y = total_score, fill = facet)) +
  geom_col(alpha = 0.9) +
  labs(x = NULL, y = "Total score") +
  scale_fill_manual(values = facet_colors) +
  theme_SNL +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())


# Plot B: Total score per facet ################################################
plot_b <- ggplot(facet_summary, aes(x = reorder(facet, -total_score),
                                     y = total_score, fill = facet)) +
  geom_col(alpha = 0.9) +
  labs(x = NULL, y = "Total score") +
  scale_fill_manual(values = facet_colors) +
  theme_SNL +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        legend.position = "none")

# Plot C: Total score per assumption (nature) #################################
plot_c <- ggplot(score, aes(x = code, y = total_score, fill = nature)) +
  geom_col(alpha = 0.9) +
  labs(x = "Assumption ID", y = "Total score") +
  scale_x_discrete(
    breaks = c("A1", "A25", "B1", "B25", "C1", "D1", "E1")  # <- show only selected codes
  ) +
  theme_SNL +
  theme(axis.text.x = element_text(size = 9))

# Plot D: Total score per nature ###############################################
plot_d <- ggplot(nature_summary, aes(x = nature, y = total_score, fill = nature)) +
  geom_col(alpha = 0.9) +
  labs(x = "", y = "Total score") +
  theme_SNL +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        legend.position = "none")

# Add labels ###################################################################
plot_label <- function(plot, label) {
  arrangeGrob(plot, left = textGrob(label,
                    x = unit(1.25, "lines"),
                    y = unit(0.97, "npc"),
                    just = c("left", "top"),
                    gp = gpar(fontsize = 12, fontface = "bold")))}

plot_a <- plot_label(plot_a, "a")
plot_b <- plot_label(plot_b, "b")
plot_c <- plot_label(plot_c, "c")
plot_d <- plot_label(plot_d, "d")

# Arrange subplot ##############################################################
row1 <- arrangeGrob(plot_a, plot_b, ncol = 2, widths = c(0.8, 0.2))
row2 <- arrangeGrob(plot_c, plot_d, ncol = 2, widths = c(0.8, 0.2))

assumptions <- grid.arrange(row1, row2, nrow = 2)
```



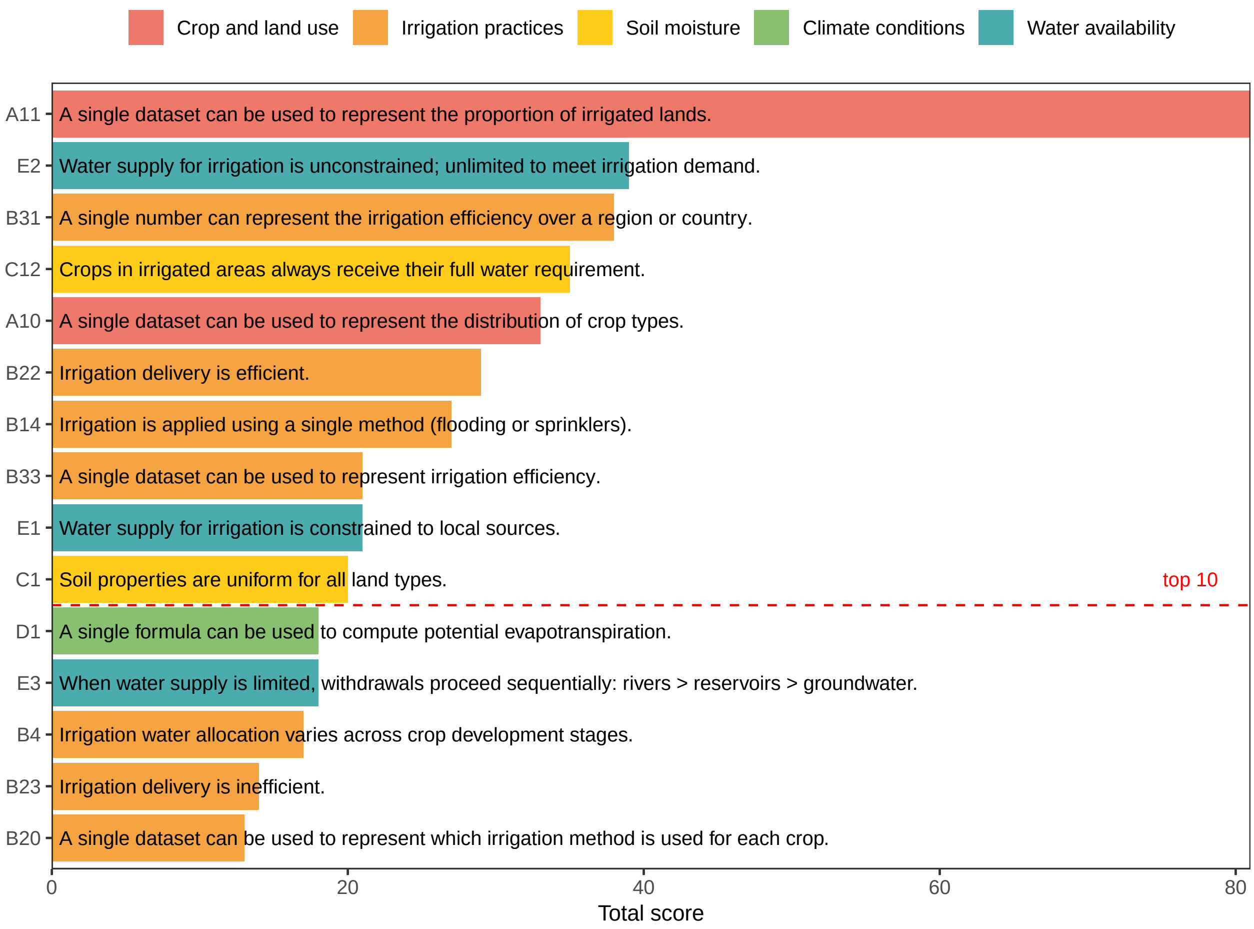
```
# Save and display #############################################################
ggsave("figures/1_assumptions.png",
       plot = assumptions, width = 10, height = 5, units = "in", dpi = 600)

assumptions
```

```
## TableGrob (2 x 1) "arrange": 2 grobs
##   z     cells    name            grob
## 1 1 (1-1,1-1) arrange gtable[arrange]
## 2 2 (2-2,1-1) arrange gtable[arrange]
```

4 Priority assumptions

```
# Filter top 15 scored assumptions #############################################
top15 <- score %>%
  arrange(desc(total_score), assumption) %>%
  slice_head(n = 15)
```

```r
# Fix factor levels to preserve order in the plot ##############################
top15$code <- factor(top15$code, levels = rev(top15$code))


# Plot: Top 15 assumptions by total score ######################################
top10 <- ggplot(top15, aes(x = reorder(code, total_score),
                          y = total_score,
                          fill = facet)) +
  geom_col(alpha = 0.9) +
  geom_text(aes(x = code, y = 0.5, label = assumption),
            hjust = 0,
            color = "black",
            size = 3.2) +

  # Add dashed line separating top 10 ##########################################
  annotate("segment",
           x = 5.5, xend = 5.5,
           y = 0, yend = max(top15$total_score) * 1.0,
           linetype = "dashed", color = "red") +

  # Add label above the dashed line ############################################
  annotate("text",
           x = 6.0,  # slight left shift
           y = max(top15$total_score) * 0.95,
           label = "top 10",
           size = 3.2, color = "red") +

  # Apply theme ################################################################
  labs(x = NULL,
       y = "Total score") +
  coord_flip() +
  scale_y_continuous(expand = c(0, 0)) +
  scale_fill_manual(values = facet_colors) +
  theme_SNL +
  theme(legend.position = "top")

# Save and display #############################################################
ggsave("figures/2_top10.png",
       plot = top10, width = 8, height = 6, units = "in", dpi = 600)

top10
```



#Bootstrap

```
set.seed(123)  # reproducibility

n_boot <- 1000  # number of bootstrap replicates
top_n <- 10     # top 10 assumptions in each replicate

# Function to do one bootstrap replicate ########################################
bootstrap_top10 <- function(data, top_n) {
  # Sample all rows with replacement
  sampled_data <- data %>% slice_sample(n = nrow(data), replace = TRUE)

  # Take the top N
  top_sample <- sampled_data %>% slice(1:top_n)

  # Count how many come from each facet
  top_sample %>%
    count(facet)
}

# Run the bootstrap with purrr ##################################################
boot_results <- map_dfr(1:n_boot, ~ bootstrap_top10(score, top_n), .id = "replicate")

# Fill missing facets with 0 (if a facet is not in top 10 for a replicate) #####
boot_results <- boot_results %>%
  complete(replicate, facet, fill = list(n = 0))

# Summarize across replicates ####################################################
facet_summary <- boot_results %>%
  group_by(facet) %>%
  summarise(
    mean_count = mean(n),
    sd_count = sd(n),
    median_count = median(n),
    .groups = "drop"
  )

facet_summary
```

```
## # A tibble: 5 x 4
##   facet                 mean_count sd_count median_count
##   <fct>                      <dbl>    <dbl>        <dbl>
## 1 Crop and land use          3.54     1.56             3
## 2 Irrigation practices       3.28     1.49             3
## 3 Soil moisture              1.68     1.17             2
## 4 Climate conditions         0.412    0.609            0
## 5 Water availability         1.08     0.972            1
```

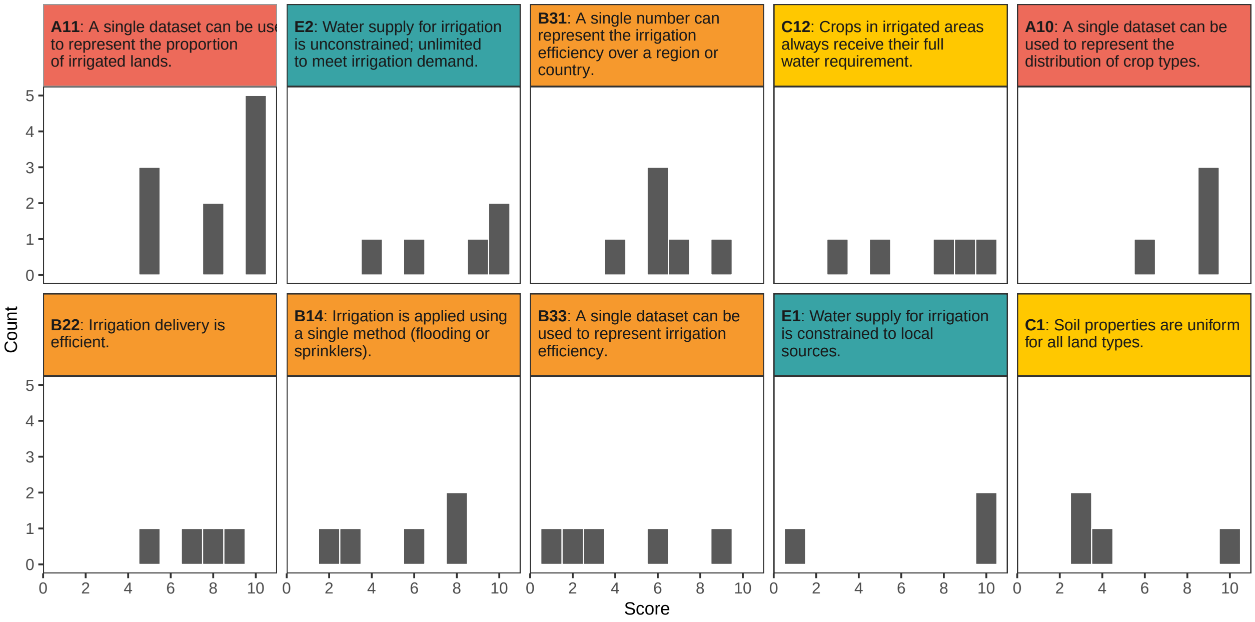
5 Statistics in scored assumptions

```
# Filter scored and top 10 assumptions ##########################################
scored <- score %>%
  filter(total_score > 0)
top10 <- scored %>%
  arrange(desc(total_score)) %>%
  slice_head(n = 10)

# Summarize statistics ##########################################################
summary_table <- score %>%
  count(facet, name = "total_assumptions") %>%
  # total in each facet
  left_join(scored %>%
              count(facet, name = "scored_assumptions"), by = "facet") %>%
  left_join(top10 %>%
              count(facet, name = "top10_assumptions"), by = "facet") %>%
  replace_na(list(scored_assumptions = 0, top10_assumptions = 0)) %>%
  mutate(
    total_pct = round(100 * total_assumptions / sum(total_assumptions), 1),
    scored_pct = round(100 * scored_assumptions / sum(scored_assumptions), 1),
    top10_pct = round(100 * top10_assumptions / sum(top10_assumptions), 1))

summary_table
```

```
## # A tibble: 5 x 7
##   facet         total_assumptions scored_assumptions top10_assumptions total_pct
##   <fct>                     <int>              <int>             <int>     <dbl>
## 1 Crop and lan~                35                 12                 2        35
## 2 Irrigation p~                33                 22                 4        33
## 3 Soil moisture                17                 10                 2        17
## 4 Climate cond~                 4                  3                 0         4
## 5 Water availa~                11                  5                 2        11
## # i 2 more variables: scored_pct <dbl>, top10_pct <dbl>
```

6 Histogram of scores

```
# Filter top 10 scored assumptions ##############################################
top10 <- score %>%
  arrange(desc(total_score)) %>%
  slice_head(n = 10) %>%
  pull(assumption)

# Convert to long format ########################################################
long_scores <- score %>%
  pivot_longer(
    cols = 4:(ncol(score) - 2),
    names_to = "expert",
    values_to = "score"
  ) %>%
  filter(assumption %in% top10, score != 0) %>%
  mutate(assumption = factor(assumption, levels = top10))

# Create assumption labels and facet colors ####################################
assumption_labels <- score %>%
  distinct(assumption, code, facet) %>%
  mutate(
    facet_color  = facet_colors[facet],
    wrapped_text = str_wrap(assumption, width = 28),
    label_text   = paste0("**", code, "**: ", gsub("\n", "<br>", wrapped_text))
  )

assumption_labels_named <- setNames(assumption_labels$label_text,
                                    assumption_labels$assumption)

# Strip colors in plotting order ###############################################
strip_colors <- assumption_labels %>%
  filter(assumption %in% levels(long_scores$assumption)) %>%
  arrange(match(assumption, levels(long_scores$assumption))) %>%
  pull(facet_color)

# Plot histogram of scores #####################################################
scores <- ggplot(long_scores, aes(x = score)) +
  geom_histogram(binwidth = 1, color = "white", linewidth = 0.3) +

  # Facet with colored strips (consistent with pedigree plot)
  ggh4x::facet_wrap2(
    ~ assumption,
    nrow = 2, ncol = 5, scales = "fixed",
    labeller = labeller(assumption = assumption_labels_named),
    strip = ggh4x::strip_themed(
      background_x = ggh4x::elem_list_rect(fill = strip_colors, color = "gray60")
    )
  ) +

  # Scales and labels ##########################################################
  scale_x_continuous(breaks = seq(0, max(long_scores$score, na.rm = TRUE), by = 2)) +
  labs(x = "Score", y = "Count") +

  # Apply consistent theme and style ###########################################
  theme_SNL +
  theme(
    legend.position = "none",
    strip.text = ggtext::element_markdown(
      size = 9, lineheight = 1.1, hjust = 0
    )
  )

# Save and display #############################################################
ggsave("figures/s1_scores.png",
       plot = scores, width = 10, height = 5, units = "in", dpi = 600)

scores
```



7 Models and assumptions

```
# Load data ####################################################################
dt <- read_excel("data/workshop_data.xlsx", sheet = "model")

# Create links and nodes #######################################################
links <- dt %>%
  count(source = model, target = assumption, name = "value")

all_nodes <- unique(c(links$source, links$target))
nodes <- data.frame(name = all_nodes)
```

```r
# Map source and target names to indices
links$source_id <- match(links$source, nodes$name) - 1
links$target_id <- match(links$target, nodes$name) - 1

links$group <- links$source

# Assign colors ################################################################
model_names <- unique(links$source)
assumption_names <- unique(links$target)

model_colors <- model_colors[model_names]
assumption_colors <- rep("#ffffff", length(assumption_names))
node_colors <- c(model_colors, assumption_colors)

color_scale <- paste0(
  "d3.scaleOrdinal().range([\"",
  paste(node_colors, collapse = "\",\""),
  "\"])")

# Create the Sankey diagram ###################################################
sankey <- sankeyNetwork(
  Links = links,
  Nodes = nodes,
  Source = "source_id",
  Target = "target_id",
  Value = "value",
  NodeID = "name",
  fontSize = 18,
  nodeWidth = 5,
  nodePadding = 5,
  sinksRight = FALSE,
  colourScale = color_scale,
  LinkGroup = "group",
  NodeGroup = "name"
)
```

```
## Links is a tbl_df. Converting to a plain data frame.
```

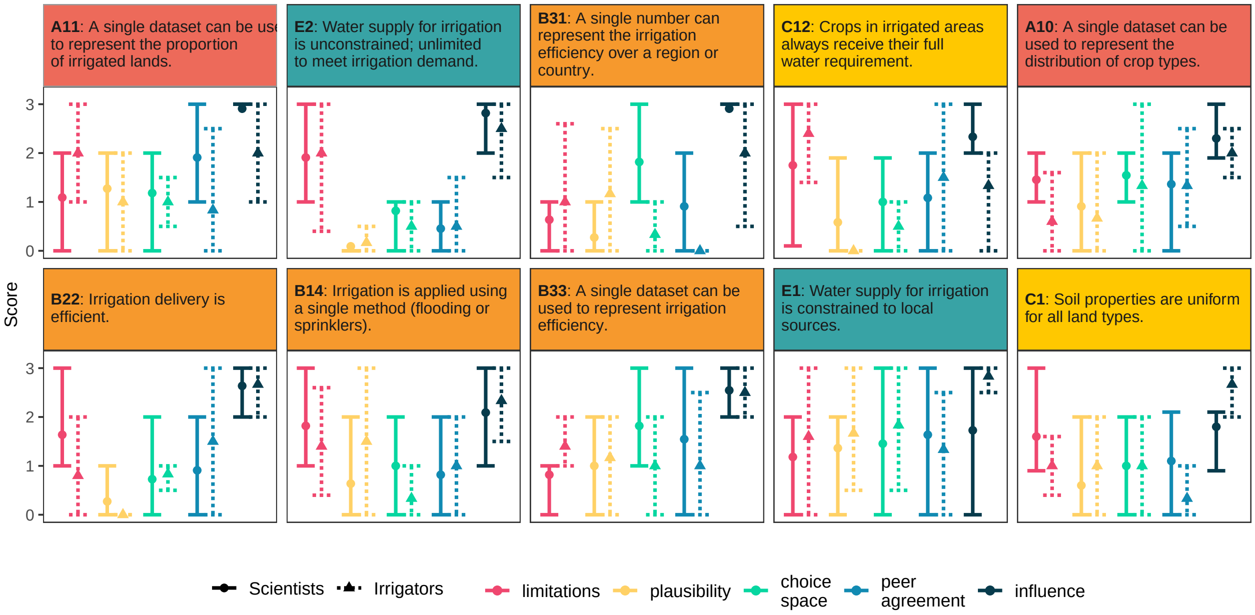
```r
# Adjust color transparency ####################################################
alpha <- 0.7

sankey <- onRender(sankey, sprintf("
  function(el) {
    d3.select(el).selectAll('.link')
      .style('stroke-opacity', %f);
  }
", alpha))

# Save the diagram as an interactive HTML ######################################
html_file <- "figures/s2_sankey_diagram.html"
saveWidget(sankey, file = html_file, selfcontained = TRUE)

# Convert to png (remove comment to save figure)################################
#png_file <- "figures/s2_sankey_diagram.png"
```

```
#webshot(url = html_file, file = png_file, vwidth = 1200, vheight = 800, zoom = 2)
```

8 Criteria scores

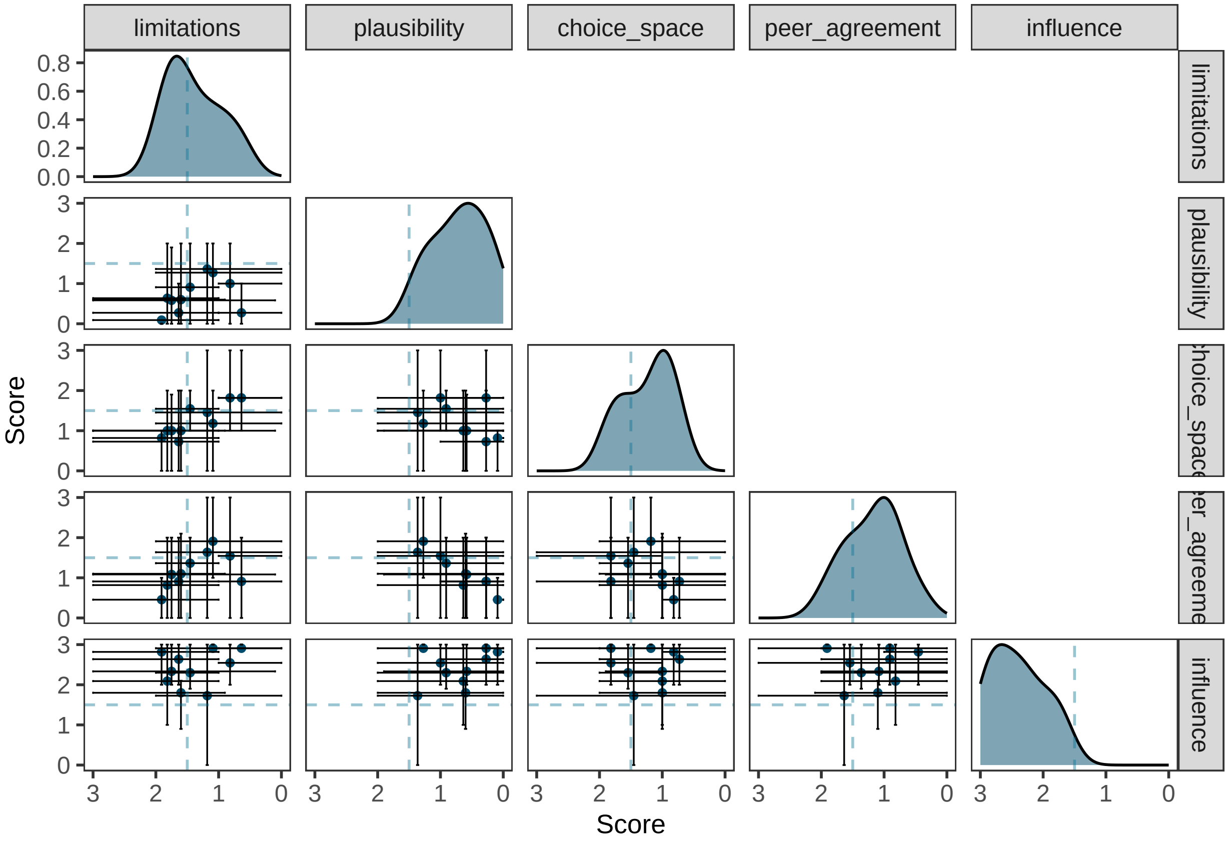
```
# Filter data and set factor levels ############################################
pedigree_comb <- pedigree_comb %>%
  filter(is.finite(score), score >= 0, score <= 3, assumption %in% top10) %>%
  mutate(
    group = ifelse(expert_id %in% 1:11, "Scientists", "Irrigators"),
    group = factor(group, levels = c("Scientists", "Irrigators")),  # set desired order
    assumption = factor(assumption, levels = top10)  # preserve top10 order
  )

# Set criterion order for plotting
ordered_criteria <- c("limitations", "plausibility", "choice\nspace", "peer\nagreement", "influence")

pedigree_comb <- pedigree_comb %>%
  mutate(
    criterion = fct_recode(criterion,
      "limitations"     = "limitations",
      "plausibility"    = "plausibility",
      "choice\nspace"   = "choice_space",
      "peer\nagreement" = "peer_agreement",
      "influence"       = "influence"
    ),
    criterion = factor(criterion, levels = ordered_criteria)  # set display order
  )

# Summarize scores #############################################################
ped_summary <- pedigree_comb %>%
  group_by(assumption, criterion, group) %>%
  summarise(
    mean = mean(score, na.rm = TRUE),
    q1   = quantile(score, 0.1, na.rm = TRUE),
    q2   = quantile(score, 0.9, na.rm = TRUE),
    .groups = "drop"
  )

# Join with assumption facet for coloring ######################################
assumption_facet <- score %>%
  distinct(assumption, facet, code)

ped_summary <- ped_summary %>%
  left_join(assumption_facet, by = "assumption") %>%
  mutate(assumption = factor(assumption, levels = top10))

# Create strip labels and colors ###############################################
assumption_labels <- score %>%
  distinct(assumption, code, facet) %>%
  mutate(
    facet_color  = facet_colors[facet],
    wrapped_text = str_wrap(assumption, width = 28),  # wrap at ~XX chars
    label_text   = paste0("**", code, "**: ", gsub("\n", "<br>", wrapped_text))
  )

assumption_labels_named <- setNames(assumption_labels$label_text,
                                    assumption_labels$assumption)

# Strip colors in plotting order ###############################################
strip_colors <- assumption_labels %>%
  filter(assumption %in% levels(ped_summary$assumption)) %>%
  arrange(match(assumption, levels(ped_summary$assumption))) %>%
  pull(facet_color)

# Plot pedigree scores per assumption ##########################################
dodge_width <- 0.7

ped_scores <- ggplot(ped_summary, aes(x = criterion, y = mean,
                                      color = criterion, group = group)) +

  # Add error bars and points
  geom_errorbar(aes(ymin = q1, ymax = q2, linetype = group),
                width = 0.8, linewidth = 0.95,
                position = position_dodge(width = dodge_width)) +
  geom_point(aes(shape = group),
             size = 2,
             position = position_dodge(width = dodge_width)) +

  # Facet by assumption with colored strips
  ggh4x::facet_wrap2(
    ~ assumption,
    nrow = 2, ncol = 5, scales = "free_x",
    labeller = labeller(assumption = assumption_labels_named),
    strip = ggh4x::strip_themed(
      background_x = ggh4x::elem_list_rect(fill = strip_colors, color = "gray60")
    )
  ) +

  # Scales
  scale_y_continuous(limits = c(0, 3.2), breaks = 0:3) +
  scale_color_manual(values = criteria_colors) +
  scale_shape_manual(name = "", values = c("Scientists" = 16, "Irrigators" = 17)) +
  scale_linetype_manual(name = "", values = c("Scientists" = "solid", "Irrigators" = "11")) +

  # Labels and theme
  labs(x = "", y = "Score", color = "Criterion") +
  theme_SNL +
  theme(
    legend.position = "bottom",
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    strip.text = ggtext::element_markdown(size = 9, lineheight = 1.1, hjust = 0),
    legend.text = element_text(size = 10)
  )
```

```
# Save and display #############################################################
ggsave("figures/3_ped_scores.png",
       plot = ped_scores, width = 10, height = 5, units = "in", dpi = 600)

ped_scores
```

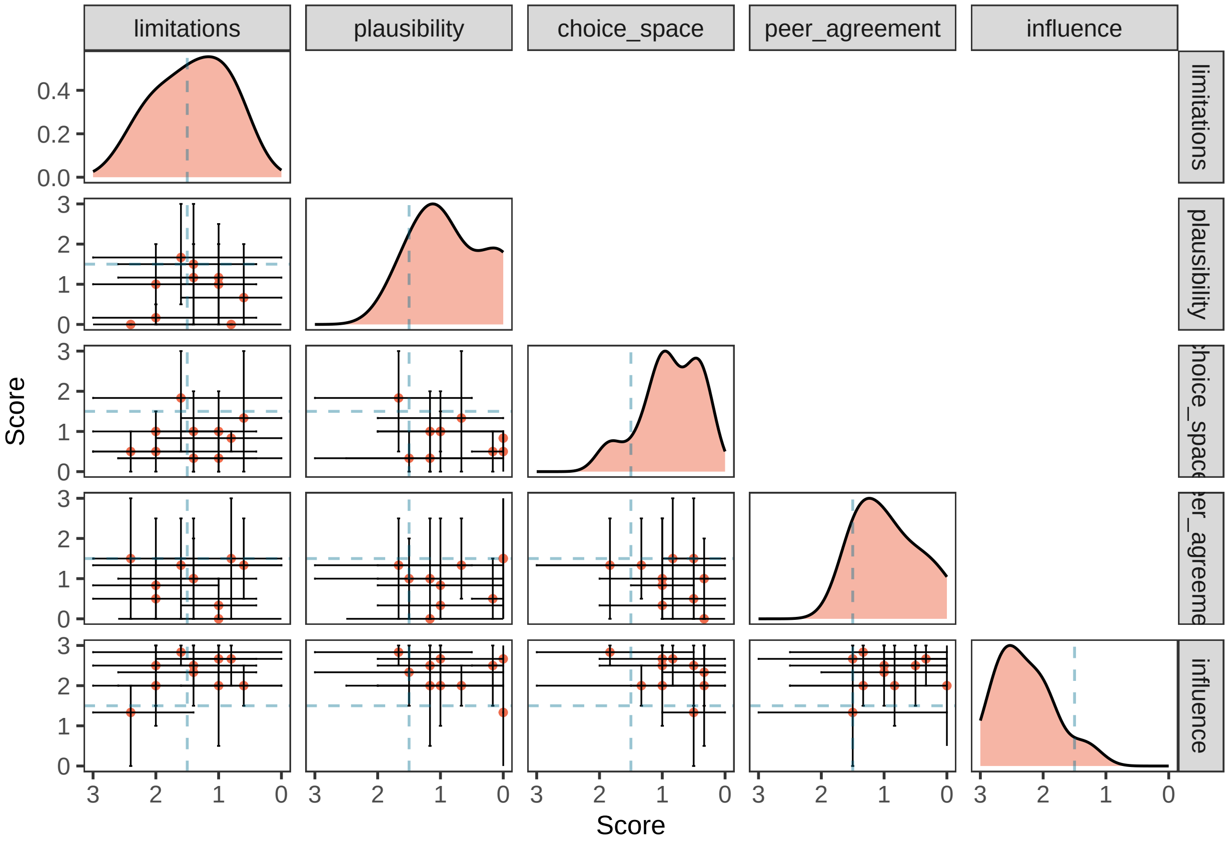


9 Pairwise diagnostic diagram

```
# Create dataset and compute quantiles #########################################
df <- pedigree %>%
  data.table() %>%
  .[, .(mean = mean(score, na.rm = TRUE),
        q1 = quantile(score, 0.1, na.rm = TRUE),
        q2 = quantile(score, 0.9, na.rm = TRUE)),
    .(assumption, criterion, code)]

# Scatter with error bars
error_point <- function(data, mapping, xvar, yvar, color_point = "#004C6D", ...) {
  x_mean <- paste0("mean_", xvar)
  x_low  <- paste0("q1_", xvar)
  x_high <- paste0("q2_", xvar)
  y_mean <- paste0("mean_", yvar)
  y_low  <- paste0("q1_", yvar)
  y_high <- paste0("q2_", yvar)

  ggplot(data, aes(x = .data[[x_mean]], y = .data[[y_mean]])) +
    geom_point(color = color_point, size = 1) +
    geom_errorbarh(aes(xmin = .data[[x_low]], xmax = .data[[x_high]]),
                   height = 0.05, size = 0.3) +
    geom_errorbar(aes(ymin = .data[[y_low]], ymax = .data[[y_high]]),
                  width = 0.05, size = 0.3) +
    geom_vline(xintercept = 1.5, linetype = "dashed", color = "#006E90", alpha = 0.4) +
    geom_hline(yintercept = 1.5, linetype = "dashed", color = "#006E90", alpha = 0.4) +
    scale_x_reverse(breaks = 3:0, limits = c(0, 3)) +
    scale_y_continuous(breaks = 0:3, limits = c(0, 3)) +
    theme_SNL +
    theme(legend.position = "top")
}

# Density for diagonal
custom_density <- function(data, mapping, fill_color = "#004C6D", ...) {
  ggplot(data, mapping) +
    geom_density(fill = fill_color, alpha = 0.5) +
    geom_vline(xintercept = 1.5, linetype = "dashed", color = "#006E90", alpha = 0.4) +
    scale_x_reverse(breaks = 3:0, limits = c(0, 3)) +
    theme_SNL
}

# Prepare data and create plot #################################################
create_ggpairs_plot <- function(sheet_name, filter_irrigators = FALSE, color_point = "#004C6D", save_fil

  pedigree <- read_excel("data/workshop_data.xlsx", sheet = sheet_name)

  # Filtering for irrigators
  if (filter_irrigators) {
    pedigree <- pedigree %>%
      filter(is.finite(score), score >= 0, score <= 3) %>%
      mutate(group = ifelse(expert_id %in% 1:11, "Scientists", "Irrigators")) %>%
      filter(group == "Irrigators")
  }

  # Compute mean and quantiles
  df <- pedigree %>%
    data.table() %>%
    .[, .(mean = mean(score, na.rm = TRUE),
          q1 = quantile(score, 0.1, na.rm = TRUE),
          q2 = quantile(score, 0.9, na.rm = TRUE)),
      .(assumption, criterion, code)]

  # Widen dataset
  df_wide <- df %>%
    pivot_wider(names_from = criterion, values_from = c(mean, q1, q2),
                names_sep = "_")

  # Prepare means-only data
  criteria <- unique(df$criterion)
  mean_cols <- paste0("mean_", criteria)
  df_means_only <- df_wide %>%
    select(all_of(mean_cols)) %>%
    rename_with(~ gsub("^mean_", "", .x))

  # Create ggpairs plot
  plot <- ggpairs(df_means_only,
                  lower = list(continuous = function(data, mapping, ...) {
                    xvar <- as_label(mapping$x)
                    yvar <- as_label(mapping$y)
                    error_point(df_wide, mapping, xvar, yvar, color_point = color_point, ...)
                  }),
                  diag = list(continuous = function(data, mapping, ...) {
                    custom_density(data, mapping, fill_color = color_point, ...)
                  }),
                  upper = NULL) +
    theme_SNL +
    labs(x = "Score", y = "Score")

  # Save plot
  ggsave(save_file, plot = plot, width = 10, height = 6, units = "in", dpi = 600)

  return(plot)
}

# Create plots #################################################################

# Scientists
pair_scientists <- create_ggpairs_plot(sheet_name = "pedigree_mod",
                                     filter_irrigators = FALSE,
                                     color_point = "#004C6D",
                                     save_file = "figures/s3_pair_scientists.png")

# Irrigators
pair_irrigators <- create_ggpairs_plot(sheet_name = "pedigree_comb",
                                       filter_irrigators = TRUE,
                                       color_point = "#EE6C4D",
                                       save_file = "figures/s3_pair_irrigators.png")

# Display plots ################################################################
pair_scientists
```



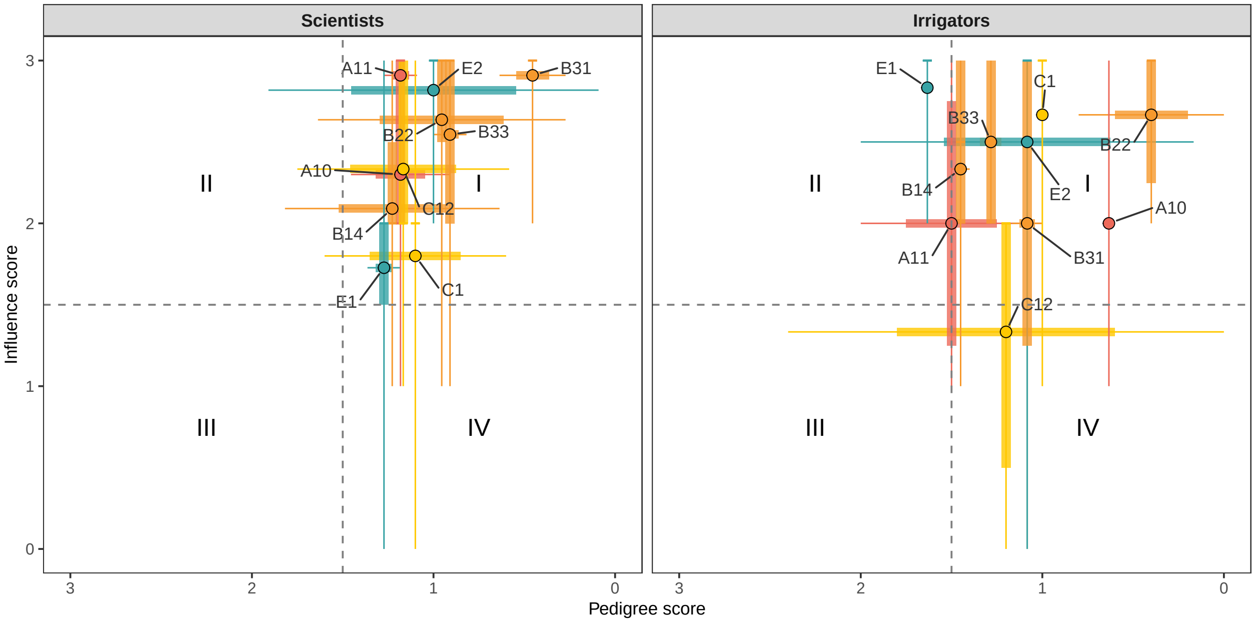
```
pair_irrigators
```



10 Assessment diagram

```
# Filter and classify by group ###################################################
pedigree_filtered <- pedigree_comb %>%
  filter(is.finite(score), score >= 0, score <= 3) %>%
  mutate(group = factor(
    ifelse(expert_id %in% 1:11, "Scientists", "Irrigators"),
    levels = c("Scientists", "Irrigators"))
  )

# Compute average scores for four pedigree criteria ############################
pedigree_avg <- pedigree_filtered %>%
  filter(criterion %in% c("choice_space",
                          "plausibility",
                          "limitations",
                          "peer_agreement")) %>%
  group_by(code, group, criterion) %>%
  summarise(mean_score = mean(score, na.rm = TRUE), .groups = "drop")

# Compute pedigree boxplot stats ################################################
box_stats <- pedigree_avg %>%
  group_by(code, group) %>%
  summarise(
    min_ped = min(mean_score),
    q1_ped = quantile(mean_score, 0.25),
    median_ped = median(mean_score),
```

```r
    q3_ped = quantile(mean_score, 0.75),
    max_ped = max(mean_score),
    avg_pedigree = mean(mean_score),
    .groups = "drop"
  )

# Compute influence boxplot stats ##############################################
influence_box <- pedigree_filtered %>%
  filter(criterion == "influence") %>%
  group_by(code, group) %>%
  summarise(
    min_inf = min(score),
    q1_inf = quantile(score, 0.25),
    median_inf = median(score),
    q3_inf = quantile(score, 0.75),
    max_inf = max(score),
    avg_influence = mean(score),
    .groups = "drop"
  )

# Merge pedigree, influence, and facet info ###################################
plot_data <- left_join(box_stats, influence_box, by = c("code", "group")) %>%
  left_join(
    score %>% distinct(code, facet),
    by = "code"
  ) %>%
  mutate(facet_color = facet_colors[facet])
box_height <- 0.025

# Create plot ##################################################################
diagnostic_diagram <- ggplot(plot_data) +

  # Pedigree whiskers
  geom_segment(aes(
    x = min_ped, xend = max_ped,
    y = avg_influence, yend = avg_influence,
    color = facet
  ), linewidth = 0.4) +

  # Pedigree box
  geom_rect(aes(
    xmin = q1_ped, xmax = q3_ped,
    ymin = avg_influence - box_height,
    ymax = avg_influence + box_height,
    fill = facet
  ), color = NA, alpha = 0.8) +

  # Pedigree median
  geom_segment(aes(
    x = median_ped, xend = median_ped,
    y = avg_influence - box_height,
    yend = avg_influence + box_height,
    color = facet
  ), linewidth = 0.6) +

  # Influence whiskers
  geom_segment(aes(
    x = avg_pedigree, xend = avg_pedigree,
    y = min_inf, yend = max_inf,
    color = facet
  ), linewidth = 0.4) +

  # Influence box
  geom_rect(aes(
    xmin = avg_pedigree - box_height,
    xmax = avg_pedigree + box_height,
    ymin = q1_inf, ymax = q3_inf,
    fill = facet
  ), color = NA, alpha = 0.8) +

  # Influence median
  geom_segment(aes(
    x = avg_pedigree - box_height,
    xend = avg_pedigree + box_height,
    y = median_inf, yend = median_inf,
    color = facet
  ), linewidth = 0.6) +

  # Mean point
  geom_point(aes(
    x = avg_pedigree,
    y = avg_influence,
    fill = facet
  ), shape = 21, size = 2.8, stroke = 0.4, color = "black") +

  # Labels
  ggrepel::geom_text_repel(
    aes(x = avg_pedigree, y = avg_influence,
        label = code), color = "grey20",
    size = 3.6,
    max.overlaps = 60,
    box.padding = 1.0,
    point.padding = 0.5
  ) +

  # Quadrant lines and annotations
  geom_vline(xintercept = 1.5, linetype = "dashed", color = "grey50") +
  geom_hline(yintercept = 1.5, linetype = "dashed", color = "grey50") +

  annotate("text", x = 0.75, y = 2.25, label = "I", size = 5) +
  annotate("text", x = 2.25, y = 2.25, label = "II", size = 5) +
  annotate("text", x = 2.25, y = 0.75, label = "III", size = 5) +
  annotate("text", x = 0.75, y = 0.75, label = "IV", size = 5) +

  # Scales
  scale_x_reverse(limits = c(3, 0), breaks = 0:3) +
  scale_y_continuous(limits = c(0, 3), breaks = 0:3) +
  scale_color_manual(values = facet_colors) +
  scale_fill_manual(values = facet_colors) +

  # Facet by group
  facet_wrap(~ group, ncol = 2) +

  # Labels & Theme
  labs(x = "Pedigree score", y = "Influence score",
       color = "Facet", fill = "Facet") +
  theme_SNL +
  theme(legend.position = "none",
        strip.text = element_text(size = 10, face = "bold"))

# Save and display ###############################################################
ggsave("figures/4_diagnostic_diagram.png",
       plot = diagnostic_diagram, width = 10, height = 5, units = "in", dpi = 600)

diagnostic_diagram
```


Scientists
Irrigators
Influence score
Pedigree score
I
II
III
IV
A11
E2
B31
B22
B33
A10
C12
B14
C1
E1
B33
B22
B14
E2
A10
A11
B31
C12