OpenPLS

USER GUIDE

OpenPLS User Guide

Complete walkthrough for researchers and practitioners

Version 0.3.28

2026-08-04

Contents

1 What is PLS-SEM?

Partial Least Squares Structural Equation Modeling (PLS-SEM) is a method for estimating models with **latent constructs**: things we cannot measure directly, only through observable indicators. Customer satisfaction, brand personality, and organizational trust are all latent: nobody hands you a "satisfaction" reading; you infer it from survey items that stand in for it.

PLS-SEM answers three linked questions in one estimation:

1. **Measurement.** How well do the observable indicators (survey items, ratings, sensor readings) represent each latent construct?
2. **Structure.** How strongly does each construct affect the others? Does perceived quality drive satisfaction? Does trust mediate the effect of usefulness on adoption?
3. **Prediction.** How well does the model generalize? What would the outcome be for a new observation?

Unlike covariance-based SEM (LISREL, Mplus, `lavaan`), PLS-SEM does not require the data to be multivariate-normal, works with small samples, and cheerfully handles **formative** measurement (indicators that jointly define a construct, e.g. income + education + occupation → socioeconomic status). This makes it the default choice in business research, marketing, information systems, and increasingly in health and social sciences.

1.1 The building blocks

A PLS-SEM model has four kinds of pieces:

- **Latent variables (LVs)**: the constructs you care about. Drawn as circles.
- **Indicators / manifest variables (MVs)**: the observed columns of your dataset that measure each LV. Drawn as squares in traditional path diagrams; in OpenPLS they live in the sidebar and get assigned to LVs.
- **Measurement model / outer model**: the arrows between an LV and its indicators. Direction distinguishes **reflective** (Mode A: LV causes the indicators, e.g. satisfaction → sat1..sat4) from **formative** (Mode B: indicators cause the LV, e.g. income + edu → SES).
- **Structural model / inner model**: the arrows between LVs. These are the hypotheses under test.

1.2 The iteration

PLS-SEM alternates two least-squares regressions until convergence:

1. Estimate each LV as a weighted composite of its indicators using the current inner-model weights.
2. Estimate the inner-model path coefficients by regressing each endogenous LV on its predecessors.

Convergence is fast (usually < 10 iterations). Standard errors and p-values come from **boot-**

strapping: resampling the data with replacement several thousand times and re-estimating.

1.3 What OpenPLS gives you

OpenPLS is a modern, open, cloud-native tool for PLS-SEM. In one project you can:

- Upload a dataset (CSV, XLSX, SPSS `.sav`, Stata `.dta`), or **build a questionnaire, publish it as a public survey, and import the accumulated responses as a dataset with one click** (chapter 13).
- Draw the model visually (drag from LV to LV), type it in a form, or **generate a starter model from a research question with the AI Companion** (chapter 12).
- Compute the PLS estimation, with live re-compute on every change.
- Assess measurement quality: reliabilities, AVE, HTMT, VIF.
- Assess structural quality: R^2, adjusted R^2, f^2, Q^2, SRMR, dULS, Fornell-Larcker.
- Bootstrap paths for p-values and confidence intervals.
- Run the advanced methods: MGA, MICOM, moderation, mediation, IPMA, PLSpredict, FIMIX, HOC, PLSc, Gaussian copula, CVPAT, Nomological validity, common-method bias.
- **Get contextual AI hints on every tab, chat with an assistant that has read-only access to your project data, and draft publication-ready Methods / Results / Discussion sections plus anticipated reviewer defenses** - all grounded in a curated library of open-access PLS-SEM papers (chapter 12).
- Export the whole run as PDF, XLSX, LaTeX, or a self-contained JSON bundle for reproducibility.
- Invite collaborators, track activity, and keep an audit trail of every model version.

The rest of this guide walks each of these in the order you'd actually use them.

> This guide is written against OpenPLS version 0.3.27 running at `cloud.openpls.app`. Screens may differ slightly in later releases; the flows remain the same.

2 Getting started

You need an account to save projects. Head to `https://cloud.openpls.app`, sign up with an email, verify it, and log in. You'll land on the **Projects** page, your workspace.

2.1 The Projects page

The Projects page (Figure 1) is your home screen. It lists every project you have access to and points you to two starting actions in the header plus a duplicate pair inside the empty-state card.

Three ways to start (numbered in Figure 1):

1. **New project** (top right), a blank workspace. You upload your own dataset and build the model from scratch.

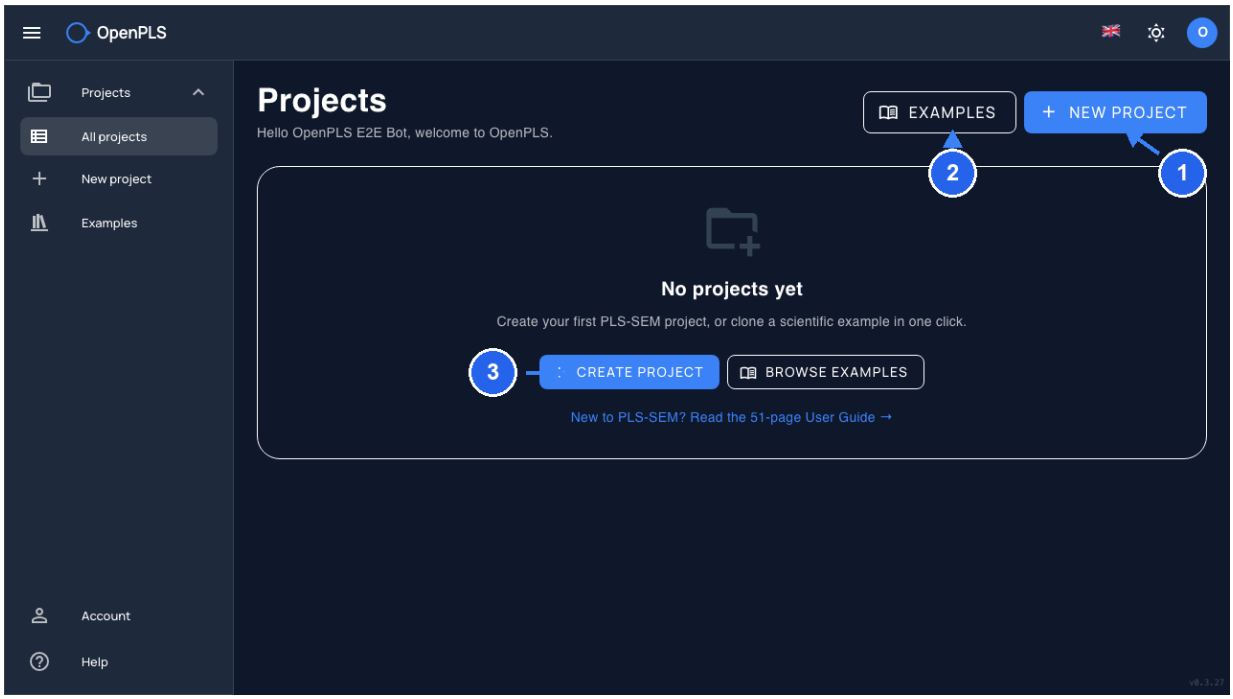

Figure 1 Projects landing page: the header offers **New project** and **Examples**; the empty-state card in the center offers the same two actions.

2. **Examples**: the sample-case library. One click clones a scientific PLS-SEM classic (ECSI, MOBI, Russett, several synthetic datasets) into your workspace, dataset + model pre-wired.
3. **Create project** in the empty-state card, shortcut to the same dialog as (1). Use this or the header button, whichever is closer to your cursor.

2.2 Creating a project

Clicking either **New project** opens the dialog shown in Figure 2.

Fill in (numbered in Figure 2):

1. **Project name**: required, ≥ 2 characters. Anything from “Customer satisfaction 2026” to “H1 experiment, country cohort”.
2. **Description**: optional, up to 500 characters. Where you jot the research question or the hypothesis you’re testing. Useful when you come back six months later.
3. Click **Create**. You land on the fresh project detail page. Because there’s no dataset attached yet, the app opens on the Dataset tab so you can upload one right away.

2.3 Cloning a scientific example

If you’re new to PLS-SEM, start with a known-good model. Click **Examples** in the header to open the library (Figure 3).

1. Pick a card that matches your domain (customer satisfaction, brand experience, technol-

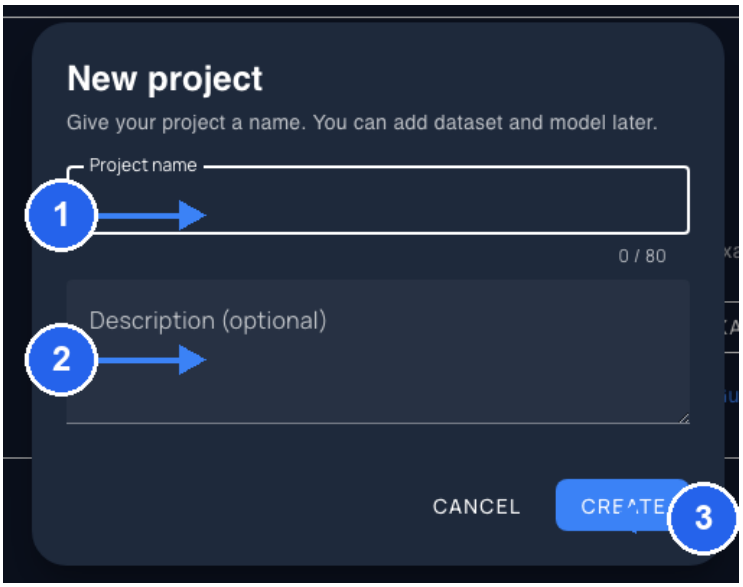

Figure 2 The **New project** dialog with name field, optional description, and the **Create** action in the bottom right.

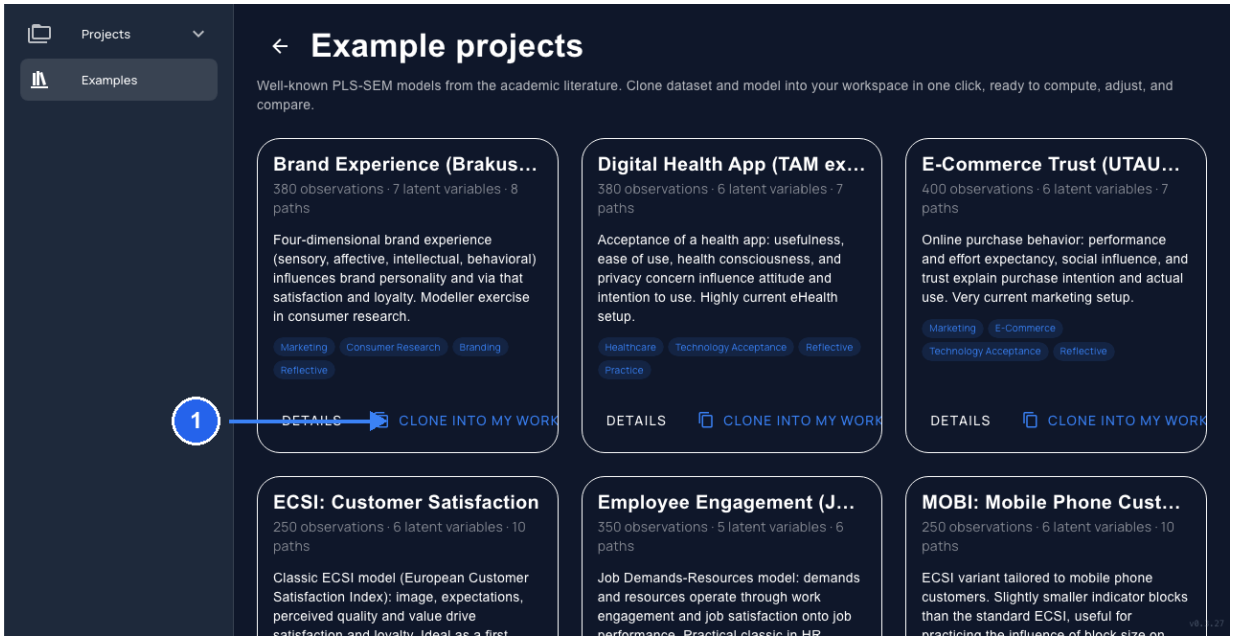

Figure 3 The Examples library. Each card shows the dataset stats and citation; click any card to open, then **Clone into my workspace** to make an editable copy.

ogy acceptance, job engagement, health apps, political science…). Each ships with a fully specified dataset + model.

2. Click **Clone into my workspace**. OpenPLS creates a private copy under your account. You can freely delete indicators, add LVs, change paths, the original stays intact for other users.

> The examples are open-licence teaching cases from the PLS-SEM literature (*Tenenhaus et al. 2005*, *Brakus & Schmitt 2009*, *Bakker & Demerouti 2017*, *Russett 1964*, *Venkatesh et al. 2012*, *Davis 1989*). Full citations are on each card. Synthetic datasets are clearly labelled and were simulated against the published path structure for teaching only.

2.4 Your first look at a project

Once you're inside a project, you see the tab strip at the top:

- **Dataset**: upload, inspect, edit missing-code conventions
- **Model**: build the model (Editor / Form / Versions)
- **Results**: read the compute output
- **Advanced**: the 13 advanced methods
- **Manage**: team, activity log, project settings

We'll walk each tab in turn. Next: uploading a dataset.

3 Datasets

Every project needs exactly one dataset before you can build a model. OpenPLS reads **CSV**, **XLSX**, **SPSS `.sav`**, and **Stata `.dta`**, up to 50 MB per file.

3.1 The empty dataset panel

A fresh project opens with the Dataset tab active. If no data is attached yet you'll see the upload zone shown in Figure 4.

Two steps to get data in:

1. Confirm you're on the **Dataset** tab. It's the first tab in every project.
2. Drag a file onto the dashed drop zone, or click it to open your OS file picker.

For CSV uploads OpenPLS opens a **CSV separator** dialog next. It previews the first few rows and auto-suggests the separator (`,`, `;`, tab, or `|`); confirm to proceed.

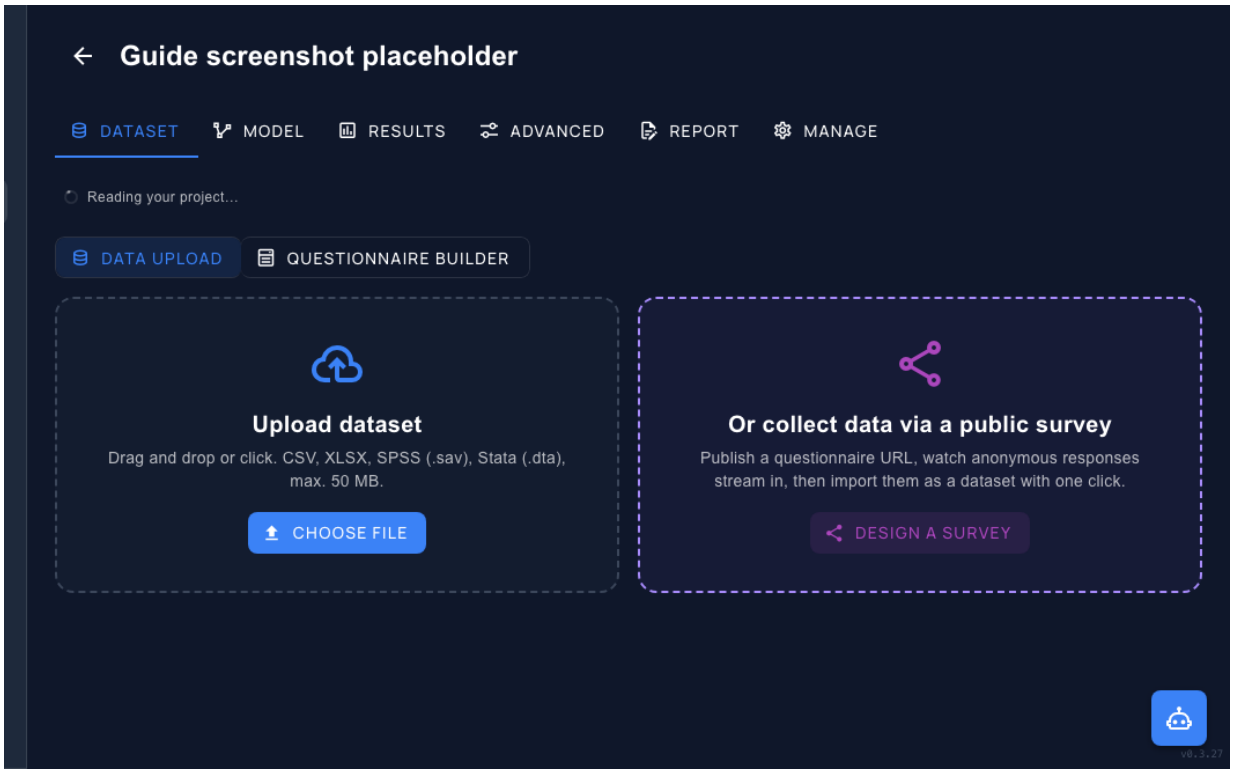

Figure 4 Empty Dataset tab. The tab strip along the top switches between Dataset, Model, Results, Advanced, Report, and Manage. Below it a sub-toggle picks between uploading a file and building a public survey.

CSVs from European Excel exports usually use ‘;‘ as the separator. OpenPLS auto-detects this, but confirm the preview looks right before clicking Upload, getting the separator wrong is the fastest way to produce a nonsense dataset.

3.2 Once the dataset is ready

Parsing is server-side and takes a second or two for typical files. When it finishes, the dataset card appears in the left column and its details render on the right (Figure 5). Two chips are important:

1. **Status chip**: turns green and reads “Ready” once parsing has completed. If it stays red (“Error”), click the dataset to see the error message.
2. **Missing codes chip**: shows which numeric values OpenPLS is treating as missing. The default `−99, −999` matches the SPSS convention. Click the chip to edit.

The right pane lists every column with its type (numeric, string, mixed), the number of missing cells, and (for numeric columns) min, max, and mean. Use this to sanity-check the parse: a column you expected to be numeric that came back as `string` usually means a stray non-numeric cell.

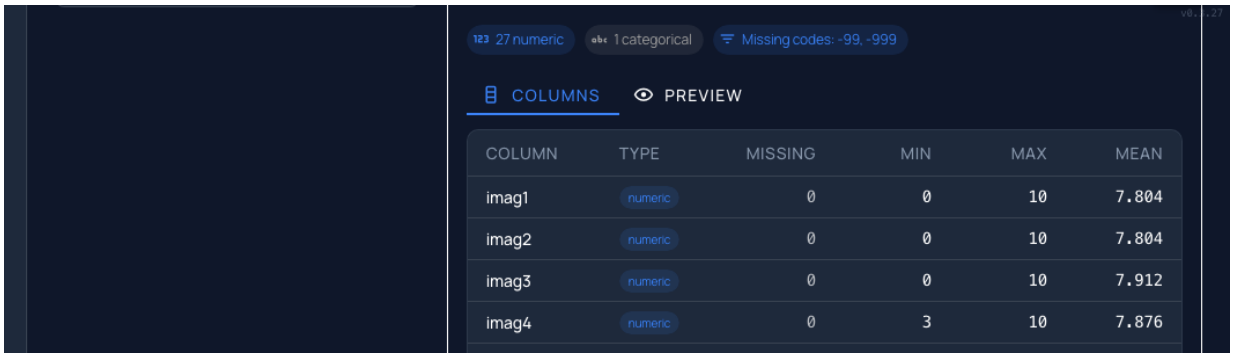

Figure 5 Dataset panel after upload. The dataset card shows the Ready-state chip and the current missing-code sentinels; the main pane on the right shows per-column stats.

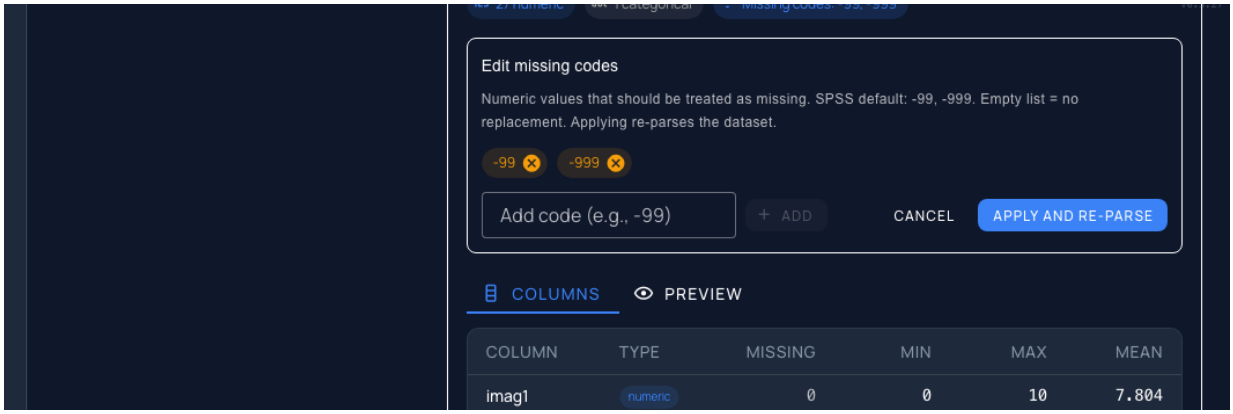

Figure 6 Missing-codes editor. Draft chips are removable via the ×; new sentinels are added via the input; **Apply and re-parse** commits the change and re-reads the file.

3.3 Editing missing codes

Some datasets code missingness as blank, others use −99 (SPSS default), yet others use 9999 or −1. Getting this wrong is the **single most common reason** for a "why is my R^2 so low?" bug report, the model treats sentinel values as real data, tanking the correlations.

Click the missing-codes chip to open the editor shown in Figure 6.

The three elements in Figure 6:

1. **Existing sentinels**: each is a chip. Click the × on a chip to drop it from the set.
2. **Add code**: type any integer (positive or negative, up to your dataset's convention) and click **Add**. It appears as a new chip.
3. **Apply and re-parse**: sends the new set to the server, which re-reads the file with the updated sentinels and refreshes the column stats. Wait for the button's spinner to stop before moving on.

> Applying missing codes re-parses the file from scratch. If you had already computed a model, the compute is invalidated, rerun it after the re-parse.

3.4 Managing multiple datasets

A single project can hold several datasets side by side (e.g. one per country, or a pilot vs. full sample). Add more by dragging or picking additional files. Each dataset gets its own card in the left column; click a card to make it active. Models bind to a specific dataset, so switching the active dataset from the card list only changes what the Dataset tab shows, the model still points at whatever it was created with.

3.5 Deleting a dataset

To delete a dataset, hover its card in the left column and click the trash icon. Deletion is instant and cannot be undone. If any model in the project still references the deleted dataset, its compute will error until you re-attach a different dataset.

> Datasets are deleted immediately with no way to undo. Any model in the project that still points at the deleted dataset will error on compute, either re-attach a different dataset in the Model tab or delete the model too.

4 Building a model

OpenPLS gives you three ways to shape the model:

- **Editor**: the visual drag-and-drop canvas. This is the primary workflow and OpenPLS's killer feature.
- **Form**: a plain form with a table of LVs and a table of paths. Faster when you know the model by heart or are copying from a paper.
- **Versions**: the audit log of every past compute. Each computation snapshots the model + result so you can restore any prior state.

All three sub-views edit the same underlying model. Switch freely.

4.1 The visual editor

Click **Model → Editor** to open the canvas shown in Figure 7. The toolbar sits above the canvas; the indicator sidebar on the right lists every numeric column of the dataset.

Reading Figure 7 from left to right, the toolbar exposes:

1. **Model selector**: a project can hold multiple models against the same dataset. Pick one from the dropdown.
2. **New model**: creates a fresh empty model. The dialog asks for a name; you can rename later in Settings.
3. **Dataset picker**: every model binds to one dataset. If you uploaded more than one CSV, choose here.

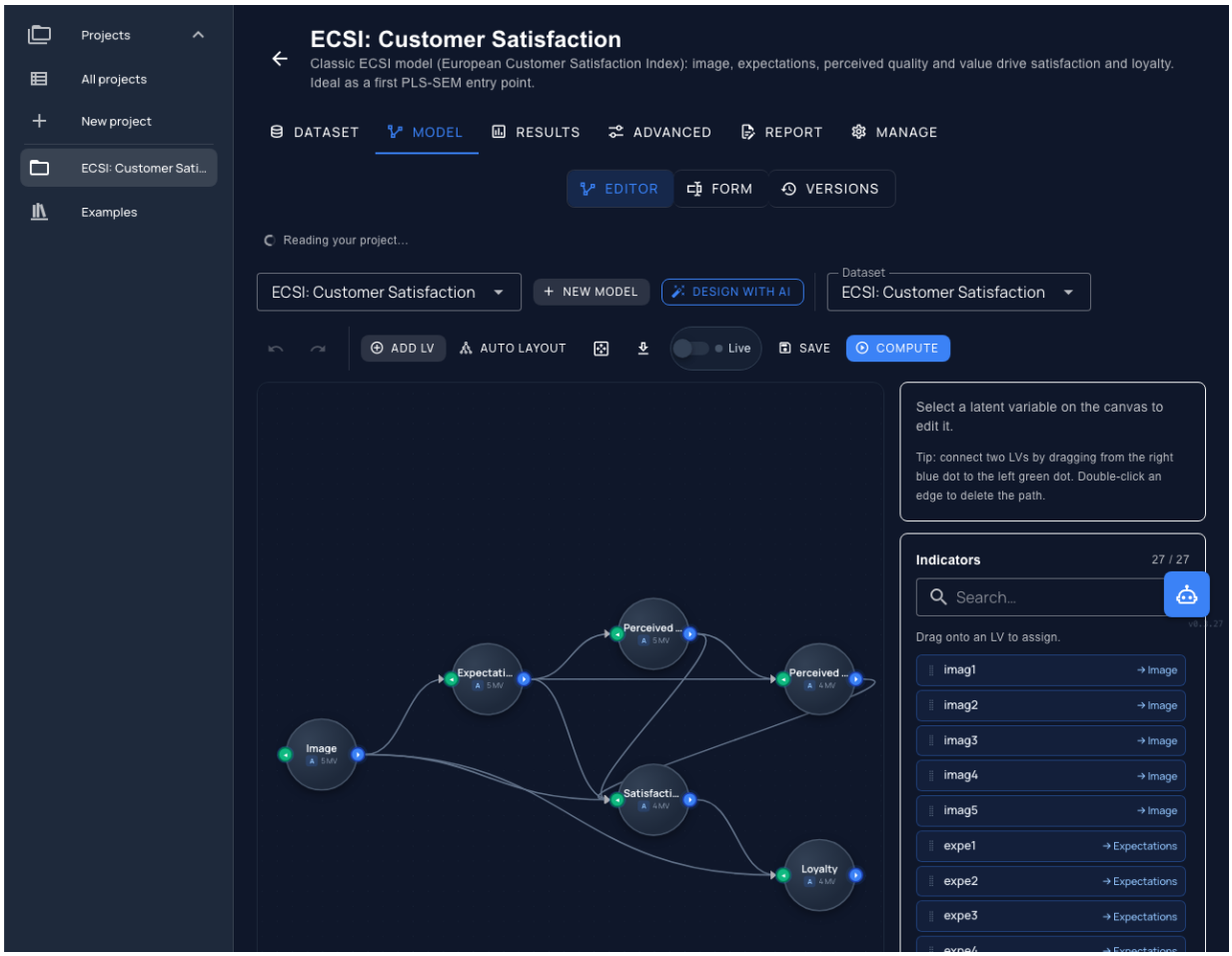

Figure 7 The visual editor with the toolbar (top), the Vue Flow canvas (center), the indicator sidebar (right).

4. **Undo / Redo**: Cmd+Z / Cmd+Shift+Z on macOS, Ctrl+Z / Ctrl+Y on Windows and Linux. Undo goes back through LV creates, indicator assignments, and path draws.
5. **Add LV**: drops a new latent-variable circle onto the canvas at a random position and auto-selects it.
6. **Auto layout**: runs dagre to arrange nodes left-to-right in topological order. Cheap way to get from a mess of overlapping circles to something publication-ready.
7. **Fit to window**: re-centers the canvas on all visible nodes.
8. **Export model**: download the model as `.openpls.json` (full OpenPLS format, includes positions) or `.splsm` (SmartPLS-compatible XML).
9. **Live toggle**: flip on, and OpenPLS re-computes the model 500 ms after every change. Path coefficients render on the edges, R^2 appears inside each endogenous LV. Fast feedback while you iterate.
10. **Save**: writes the current model to the database. There is no autosave; click Save (or Compute, which saves implicitly) to persist your edits.
11. **Compute**: runs the full estimation server-side (bootstrap + metrics), writes a Result document, and navigates to the Results tab.

4.2 The canvas

Each **latent variable** renders as a circle with:

- The LV name
- A pill showing measurement mode: **A** (reflective) or **B** (formative)
- A count of assigned indicators, e.g. “5 MV”
- Once results exist, the R^2 value inside the circle

Each **path** renders as an arrow between two LVs. On the source LV, the small blue dot on the right is the **source handle**; on the target LV, the small green dot on the left is the **target handle**. Drag from a source dot to a target dot to draw a new path. Double-click an existing path to delete it.

Right-clicking or selecting an LV opens the editor on the right (Figure 8).

1. **Name**: free text. Renaming propagates to every path referring to this LV.
2. **Mode A / Mode B**: reflective vs. formative. Reflective means the LV causes its indicators (change satisfaction, all sat items move together); formative means indicators define the LV (income, education, occupation together define SES). Pick wrong and reliability metrics either become meaningless (Mode B with reflective interpretation) or fail to converge (Mode A with a formative construct).
3. **Indicators**: a multi-select over every numeric column in the dataset. Adding an indicator here immediately shows up as an arrow inside the LV circle. Alternatively, drag chips from the right-hand **Indicators** sidebar directly onto the LV circle, OpenPLS drops the indicator on release.
4. **Delete LV**: the small trash icon in the header. Removes the LV and any paths pointing to

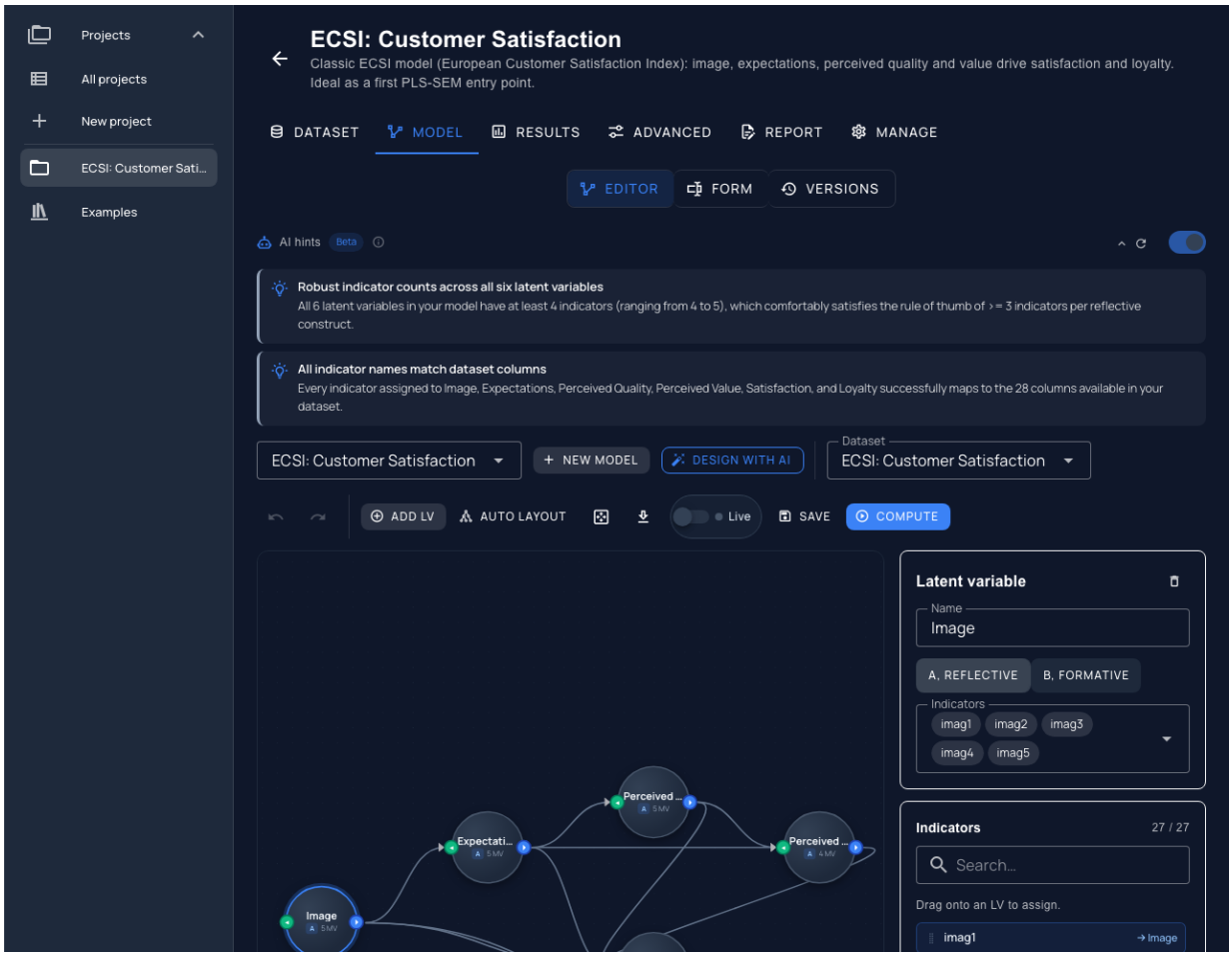

Figure 8 LV selected. The right-hand card lets you rename, change measurement mode, and pick indicators.

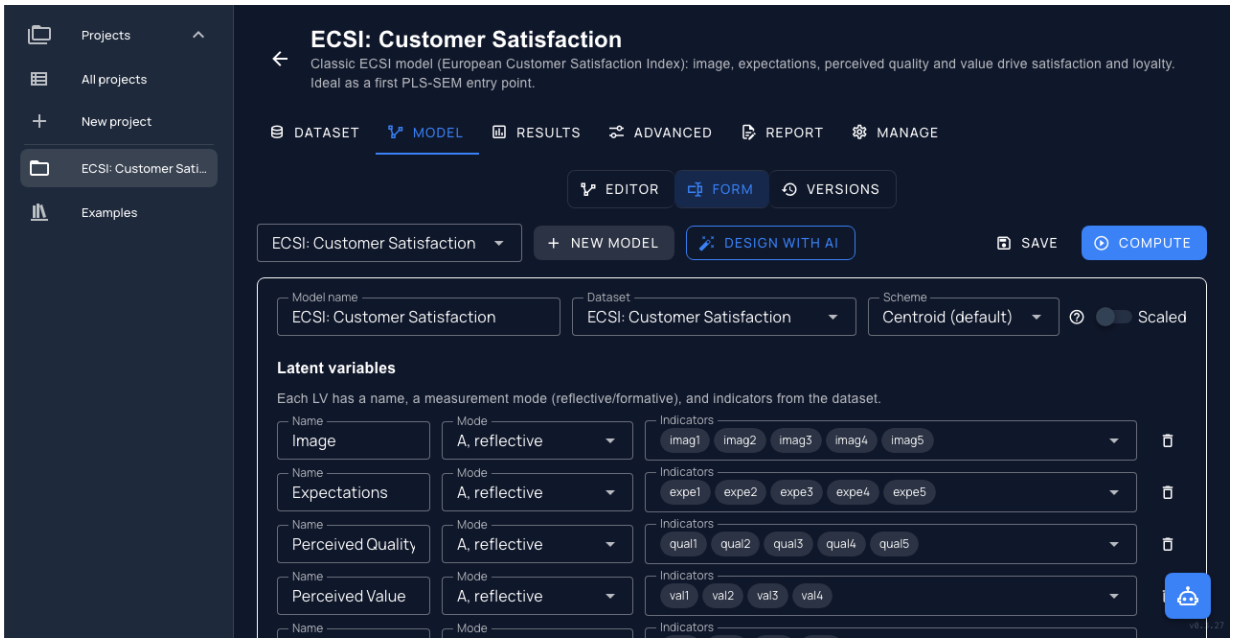

Figure 9 The Form sub-view: LV table on top (name, mode, indicators), paths table below, Save + Compute at the top right.

or from it.

> The right-hand **Indicators** sidebar lists every remaining numeric column, with a badge showing which LV (if any) already uses it. A single indicator can appear in multiple LVs, but that usually indicates a modeling mistake, investigate before continuing.

4.3 The form view

Not every user wants to click circles. Switch to **Form** to see the table-based editor (Figure 9).

The form is a plain HTML form. Add LVs with the button at the bottom of the LV table; pick indicators from a multi-select in each row. Add paths with the button below the paths table; source and target come from the LV dropdown. Everything you can do in the editor you can do here.

When to use the form: copy-pasting a model from a paper, keyboard users, or automated model construction.

4.4 Live compute

The **Live** toggle in the toolbar turns on incremental re-compute. The moment you change anything (add an LV, connect a path, tick an indicator, rename), OpenPLS debounces for 500 ms then runs a full PLS estimation server-side. The result flows back and paints:

- Path coefficients on each edge

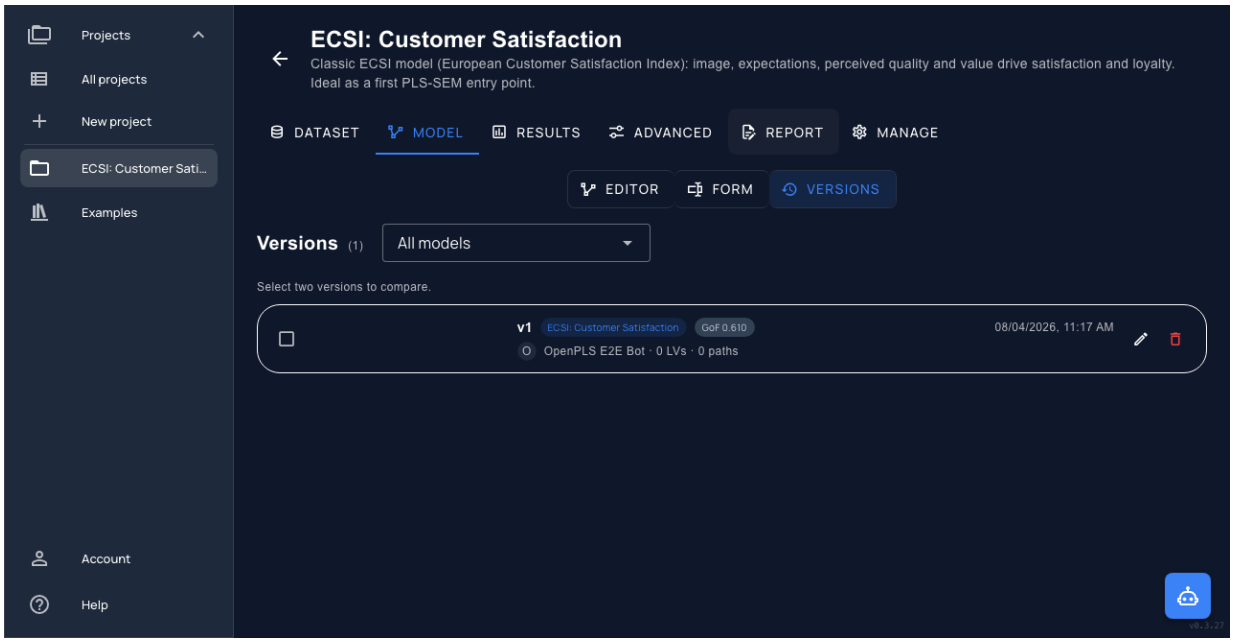

Figure 10 The Versions list. Every prior compute is a card with the LV/path counts, GoF, and creator avatar.

- R^2 and R^2 adjusted inside each endogenous LV
- p-values on the paths (from a bootstrap that runs on the last full Compute, not re-bootstrapped on every keystroke)

The `Live` chip in the toolbar cycles through **computing…**, **waiting…**, and **error** so you know what state the background job is in.

Live compute is the fastest way to explore what happens when you add a new path or reassign an indicator. Turn it off for large models where you want to make many edits before hitting Compute manually.

4.5 Versions

Every time you click **Compute**, OpenPLS snapshots the full model + result into a versioned entry. Switch to the **Versions** sub-view (Figure 10) to browse them.

Each version card shows:

- **v#**: sequential version number
- The model it belongs to (a chip)
- **GoF**: the goodness-of-fit for that version's result, if it succeeded
- The creator's avatar and a stats line (LV count, path count)
- **Rename** (pencil), label the version for future reference, e.g. "pre-invariance test" or "final submission"
- **Delete**: remove the version. The underlying model + result documents are also deleted.

Restoring a version replaces the current live model state with the snapshot. Path positions,

indicators, mode, everything.

Comparing two versions: tick the checkbox on two cards. A side-by-side diff opens on the right showing which paths were added, removed, or changed between the two.

> Restoring a version overwrites your live editor state. If you have unsaved edits, click Save first (or make a fresh Compute) so a snapshot exists to fall back to.

5 Computing and reading results

Once the model has at least one endogenous LV and every LV has ≥ 1 indicator, the **Compute** button in the Editor/Form toolbar turns solid blue. Click it. The compute runs server-side; typical ECSI-scale runs finish in 3-6 seconds. On success, OpenPLS navigates to the Results tab.

5.1 The Results header

The Results tab (Figure 11) opens with four KPI cards at the top, three export buttons on the right, and a tab strip below for the detailed tables.

The header shows four KPI cards:

1. **Goodness-of-Fit (GoF)**: $\sqrt{(\text{mean(communality)} \times \text{mean}(R^2))}$. Rough overall fit; below 0.10 weak, 0.25 moderate, 0.36+ strong.
2. **SRMR**: Standardized Root Mean Square Residual. Discrepancy between observed and model-implied correlations. Below 0.08 is the conventional cutoff for good fit; below 0.10 acceptable. The dULS value in the subtitle is another discrepancy measure.
3. **Paths**: the count of inner-model paths in the estimated model.
4. **LVs / Indicators**: the count of latent variables and total indicators (MVs). Useful as a sanity check when comparing runs.

Above the KPIs sits a result-selector dropdown listing every past Compute for the current model, timestamped. Switch between historical computes without leaving the tab.

To the right of the selector are the three **export** buttons:

- **PDF**: a print-ready report (path diagram, tables, references).
- **XLSX**: every result table in one workbook, one sheet per metric (Paths, Inner summary, Outer loadings, Reliability, HTMT, Fornell-Larcker, VIF, Effects, …).
- **Report JSON**: a self-contained JSON bundle containing the model, dataset schema, and every result table. Ideal for supplementary materials in a paper: re-run the exact same computation from the bundle later.

5.2 Inner summary (default)

The **Inner Summary** table lists each latent variable with:

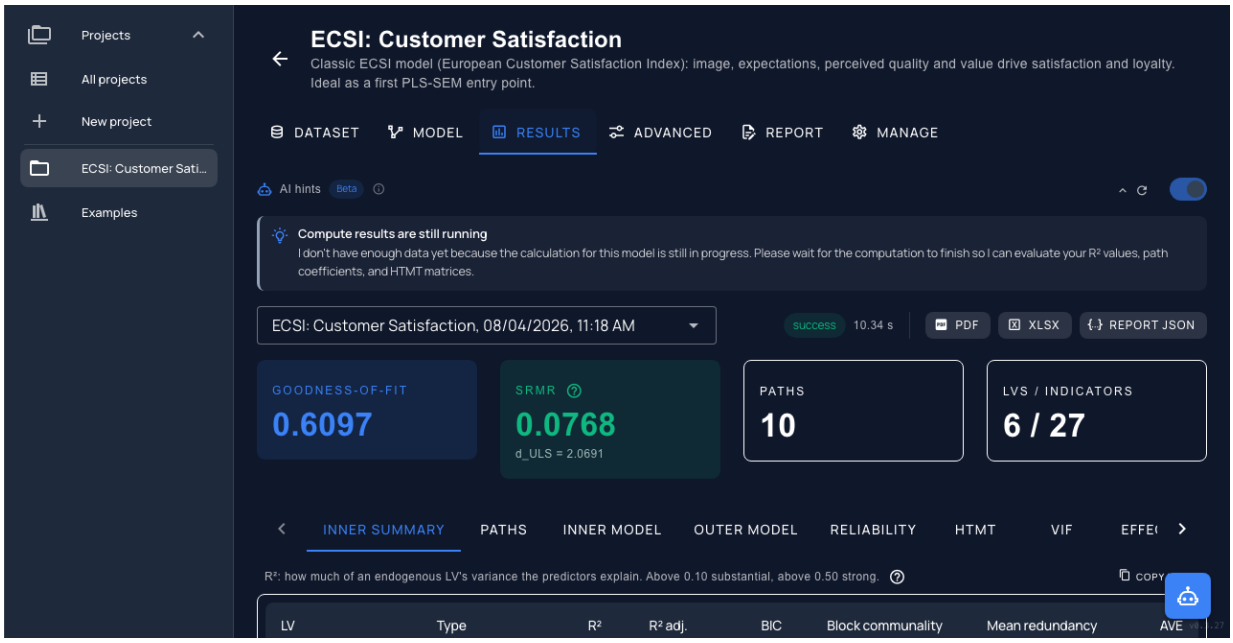

Figure 11 Results tab: KPI cards for goodness-of-fit, SRMR, path count, LV/MV count. Below them the tabbed detail: Inner Summary, Paths, Inner Model, Outer Model, Reliability, HTMT, VIF, Effect Sizes, IPMA.

- **R^2**: variance in the LV explained by its predecessors. 0.10 weak, 0.25 moderate, 0.50 strong. Exogenous LVs (no predecessors) show 0.
- **R^2 adj.**: adjusted for the number of predictors. Prefer this when comparing across models.
- **BIC**: Bayesian Information Criterion for the LV's outer model. Lower is better; only meaningful across nested specifications.
- **Block communality**: average squared loading of the LV's indicators. Higher = stronger measurement.
- **Mean redundancy**: how much variance in the block is "redundant" with the LV's predictors.
- **AVE**: Average Variance Extracted. Above 0.5 means the LV captures more of its indicators' variance than error does.

5.3 Paths

Switch to the **Paths** sub-tab (Figure 12) for the structural coefficients table.

For each path (source → target) the table lists:

- **Estimate**: the standardized path coefficient. Interpret like a beta from an OLS regression.
- **Std. err.**: from the bootstrap.
- **t**: estimate / std. err. Above 1.96 is significant at $\alpha = 0.05$ in a two-sided test.
- **p-value**: bootstrap p, corresponding to the t.
- **95% CI (lower, upper)**: percentile bootstrap CI. If it excludes zero the path is significant.

Copy the LaTeX-ready version via the small copy icon at the top right of the table.

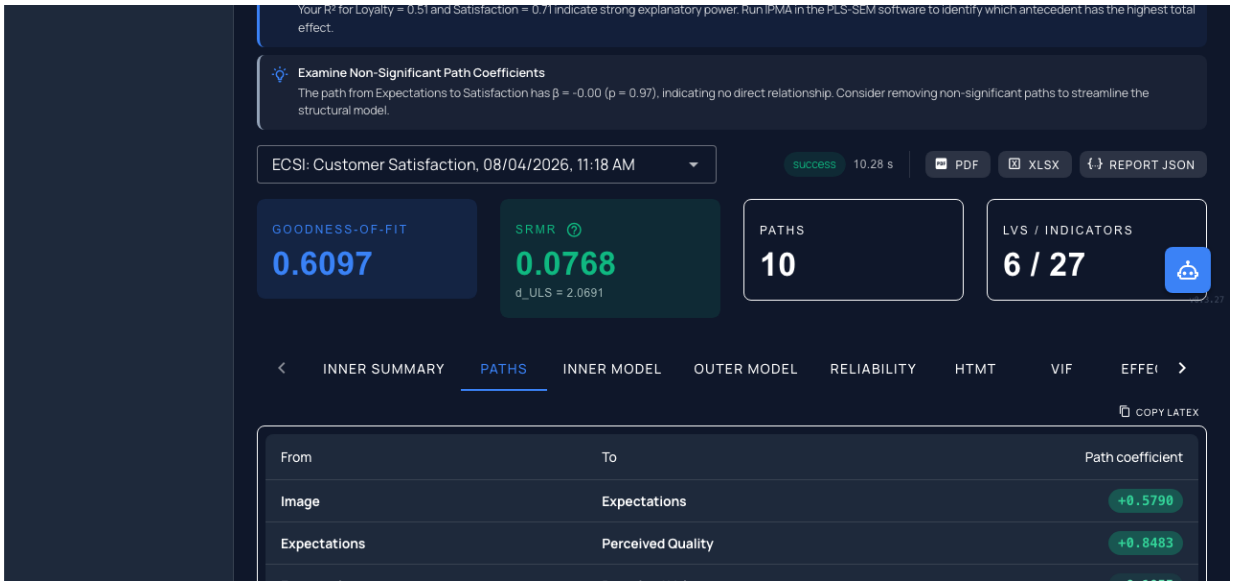

Figure 12 The Paths sub-tab: every inner-model path with its estimate, standard error, t-statistic, p-value, and 95% CI.

5.4 Inner model / outer model

The **Inner model** tab restates the paths in a wider table with the LV pairs on the rows. **Outer model** flips it: one row per indicator with its LV, loading (Mode A) or weight (Mode B), and t statistic.

Interpret loadings: above 0.708 ($\sqrt{0.5}$) means the LV explains $\geq$ 50 % of the indicator's variance. Loadings below 0.4 are candidates for removal; between 0.4 and 0.7 keep only if removing them tanks AVE.

5.5 Reliability

The Reliability tab shows two indices per LV plus two block-level diagnostics:

- **Cronbach's** α: the traditional index. Assumes tau-equivalent loadings; more conservative.
- **Composite reliability (Dillon-Goldstein** ρ**)**: the modern PLS-SEM default. 0.7-0.9 acceptable, $>$ 0.95 suggests redundant indicators.
- **1st / 2nd eigenvalue**: of the block's correlation matrix. A clear gap between them (1st much larger than 1, 2nd $\approx$ 1) supports unidimensionality of a reflective block.

Aim for $\alpha > 0.7$ and $\rho > 0.7$ on reflective constructs. For formative constructs, reliability is not meaningful, check VIF instead.

Dijkstra & Henseler's consistent reliability (ρ_A) is only reported when you run PLSc (see Section 6.6).

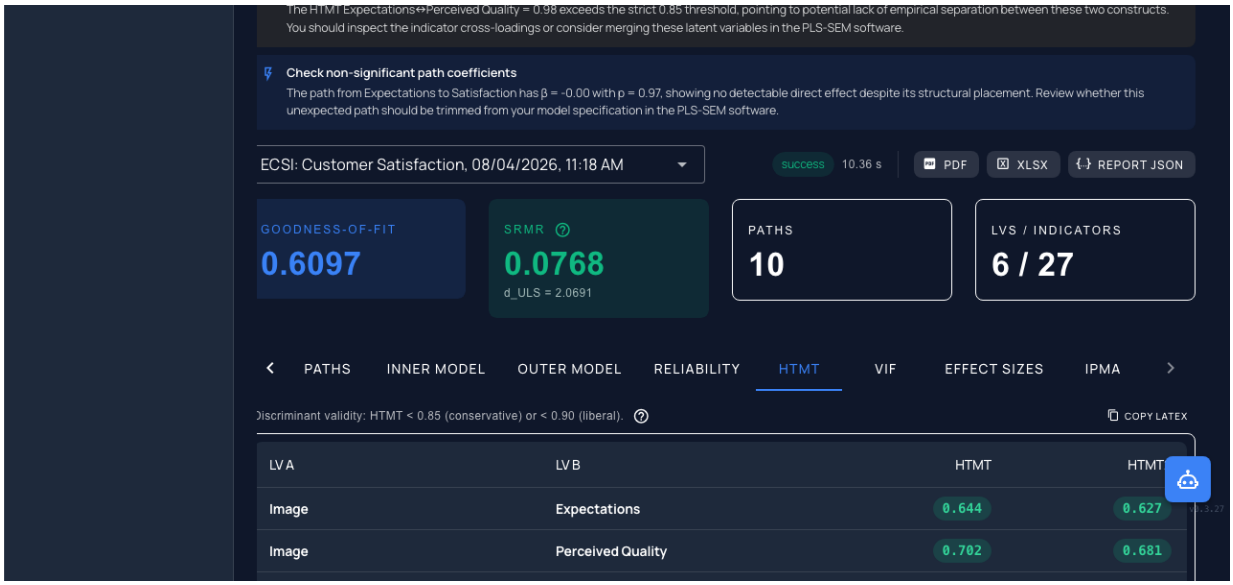

Figure 13 The HTMT sub-tab: heterotrait-monotrait ratios in a triangle matrix. Cells above the threshold (0.85 conservative, 0.90 liberal) are highlighted.

5.6 HTMT

The **HTMT** sub-tab (Figure 13) shows the heterotrait-monotrait ratio for every pair of constructs, with red cells highlighting values above the discriminant-validity threshold.

HTMT is the modern discriminant-validity test. For every pair of constructs it computes the ratio of "heterotrait" correlations (cross-construct) to "monotrait" correlations (within-construct). If this ratio approaches 1, the two constructs are indistinguishable.

Thresholds:

- **< 0.85** (Henseler, Ringle & Sarstedt 2015, conservative), safe
- **< 0.90** (Kline 2011, liberal), acceptable
- ≥ **0.90**: merge the constructs, drop one, or re-inspect the indicators

OpenPLS also runs a bootstrap version (HTMT-BST) that returns a confidence interval per pair. If the upper CI excludes 1.0, the constructs are demonstrably discriminant.

5.7 VIF

Variance Inflation Factor. Two flavours:

- **Outer VIF**: for indicators within a formative LV. Detects redundancy across formative items; keep < 5 (< 3 stricter).
- **Inner VIF**: for predictor LVs in each endogenous block. Detects collinearity between predictors; same thresholds.

5.8 Effects

Direct, indirect, and total effects for each source → target pair. Useful when reporting mediation: the direct effect is the arrow you drew, the indirect effect is the sum of all paths through intervening LVs, and the total is their sum.

5.9 Model fit

Below the KPI band OpenPLS also shows:

- **SRMR** and **dULS**: already covered above.
- **Fornell-Larcker matrix**: the classical alternative to HTMT. Off-diagonal squared correlations must be lower than the diagonal √AVE for discriminant validity.

Between HTMT and Fornell-Larcker, HTMT is the newer and more sensitive diagnostic. Report both if your reviewers expect either.

6 Advanced methods

The **Advanced** tab is where OpenPLS goes beyond a basic PLS run. Thirteen methods sit under three groups in the side-nav:

- **Inference**: Bootstrap, MGA, MICOM, Moderation, Nomological validity
- **Robustness**: PLSc, Gaussian copula, Common method bias
- **Model extensions**: IPMA, PLSpredict, CVPAT, FIMIX-PLS, Higher-order model

Every method follows the same shape: configure inputs at the top, click Run, wait for the status chip to turn green, inspect the result tables below. Every past run is preserved and selectable from the "Past runs" dropdown at the bottom (Figure 14).

6.1 Bootstrap

Purpose. Compute standard errors, t-statistics, p-values, and confidence intervals for every path coefficient and outer loading. Also computes bias-corrected accelerated (BCa) intervals for mediation effects (Figure 15).

Inputs.

1. **Iterations**: default 500, publication-grade ≥ 5000. Every iteration resamples with replacement and re-estimates the model, so runtime is roughly linear in this number.
2. **Seed**: optional, sets the RNG. Set a seed if you need perfectly reproducible standard errors.

Click **Start bootstrap** to launch. Runs on a Cloud Run service; the panel streams progress + ETA while it iterates.

Interpretation. For each path, the bootstrap gives you a distribution over B replicates. From

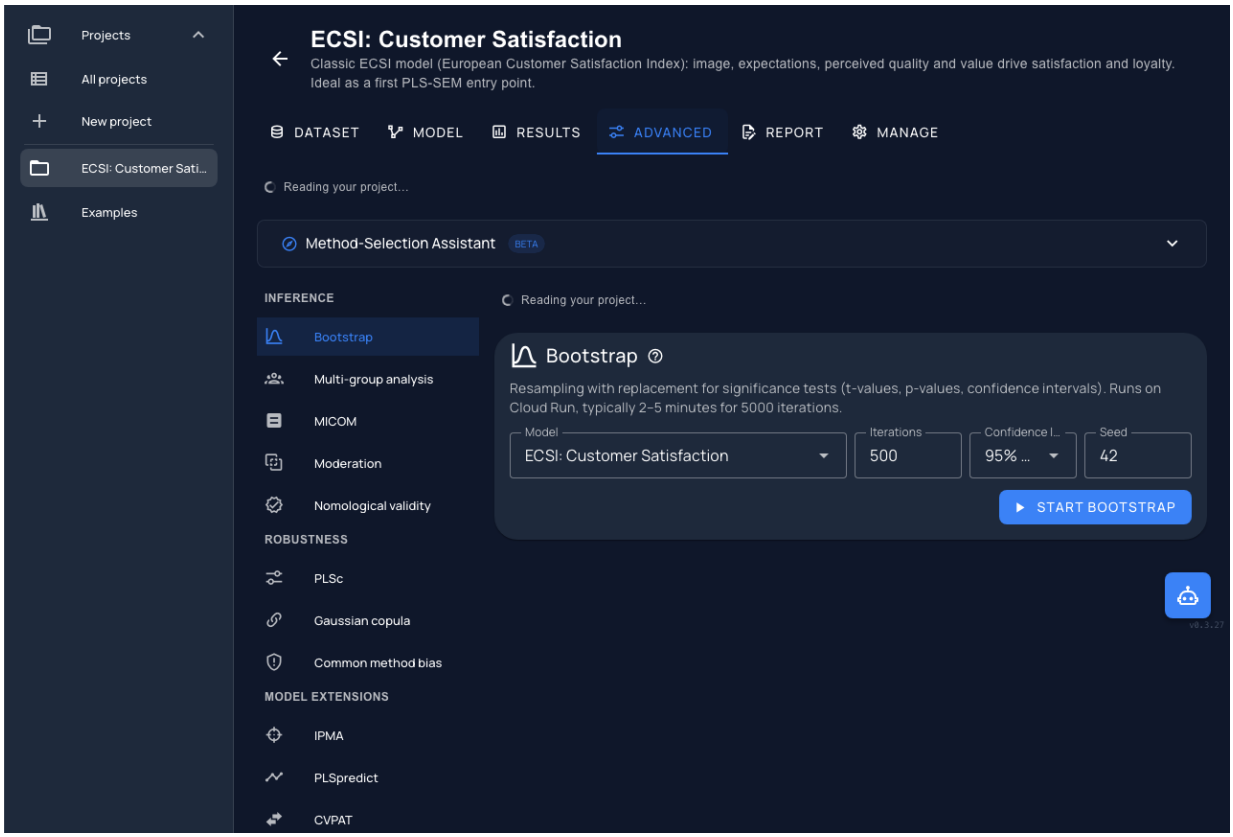

Figure 14 The Advanced tab: 13 methods in the side nav, current panel on the right.

that distribution OpenPLS computes the mean estimate, SE, t = mean/SE, p-value, and percentile CIs.

6.2 Multi-Group Analysis (MGA)

Purpose. Test whether path coefficients differ across groups (women vs. men, EU vs. US, treatment vs. control) (Figure 16).

Inputs.

1. **Grouping variable**: a column of the dataset that splits the sample. Numeric columns can be split by range; string columns by value.
2. **Groups**: one card per group. For string columns, tick the values that belong in each group (e.g. Group A = {Automotive}, Group B = {Fashion}). For numeric columns, set min/max.
3. **Permutations**: 200 quick, 1000 default, 5000+ for publication. MGA runs one permutation-based test per path.

Interpretation. For every path the table shows the estimate in each group and a p-value under the null "path is equal across groups". If $p < 0.05$ the effect differs meaningfully between the groups.

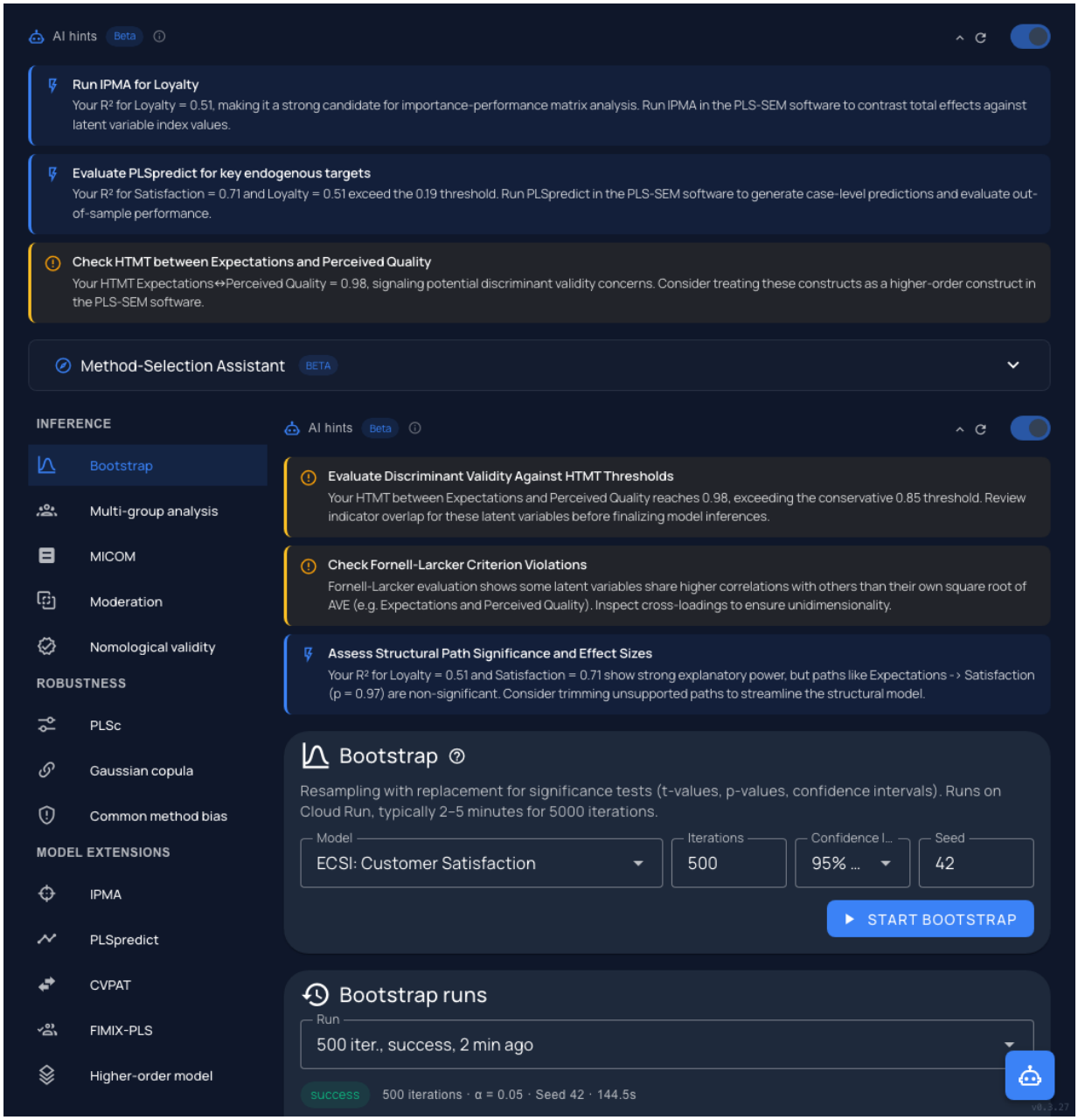

Figure 15 Bootstrap panel with iteration count, seed, and Run button.

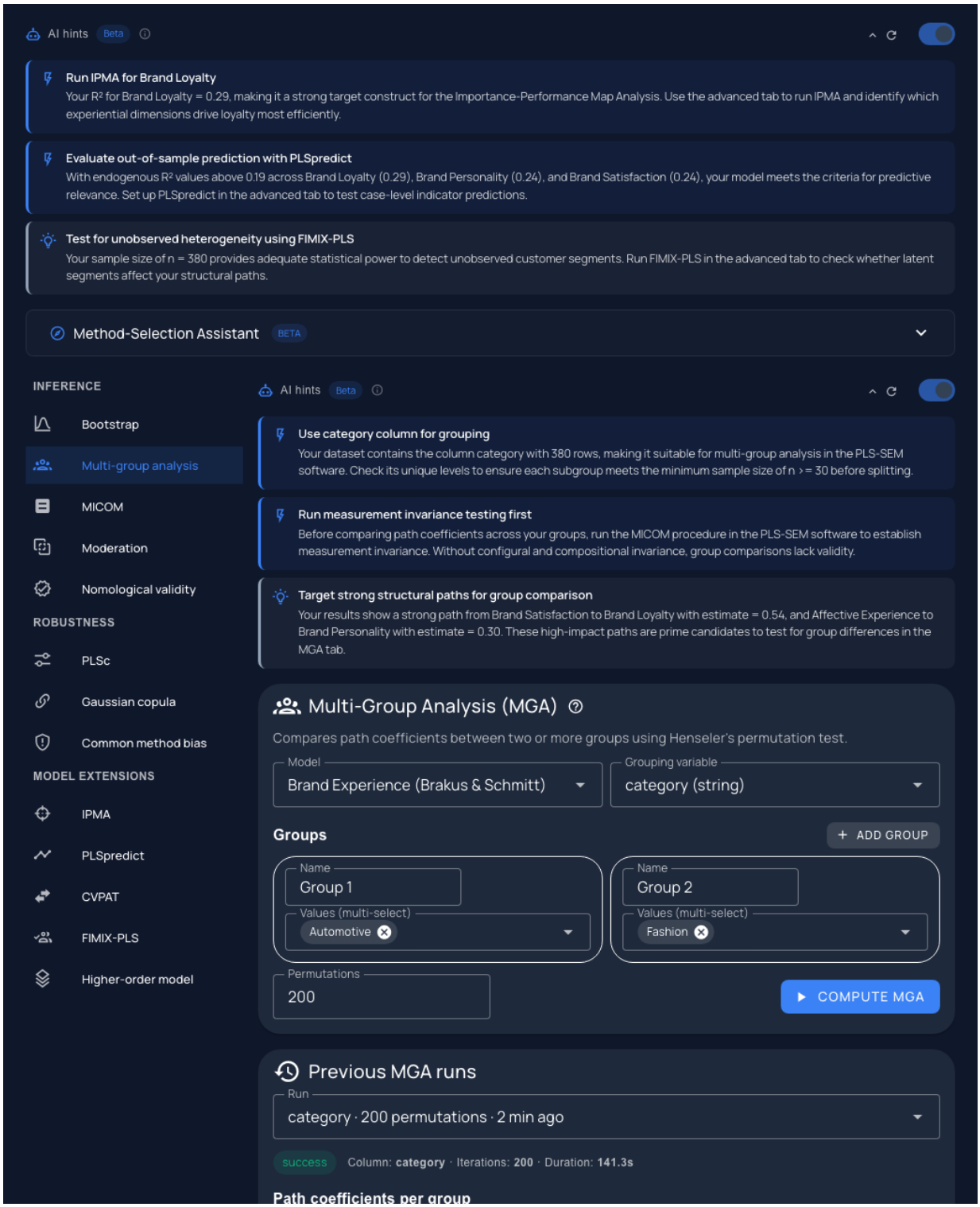

Figure 16 The MGA panel: pick a grouping column, define one card per group, set permutation count.

Always run **MICOM** first. If measurement invariance fails at Step 2 (compositional), the LVs mean different things in each group and comparing paths becomes meaningless.

6.3 MICOM

Purpose. Test **Measurement Invariance of Composite Models** across two groups. Prerequisite for MGA/moderation across groups (Figure 17).

Inputs. Identical to MGA, grouping variable + exactly two groups + permutation count.

Interpretation. Three-step verdict per construct:

1. **Configural invariance**: same indicators, same algorithm. Auto-satisfied by OpenPLS.
2. **Compositional invariance**: the composite scores of each LV correlate near 1 across groups. A p-value < 0.05 means the composites are meaningfully different, stop, MGA results are not interpretable.
3. **Scalar invariance**: composite means and variances are equal. If both Step 2 and Step 3 pass: **full invariance**. If only Step 2 passes: **partial invariance**: MGA is still valid for comparing paths, just not for comparing latent means.

6.4 Moderation

Purpose. Test whether the effect of X on Y depends on a moderator Z. Uses the two-stage approach (Henseler & Chin 2010) (Figure 18).

Inputs.

1. **Predictor (independent)**: the X in `Y = X + Z + X×Z`.
2. **Moderator**: the Z.
3. **Target (dependent)**: the Y.
4. **Interaction LV name**: optional, defaults to `X_x_Z`.

Interpretation. OpenPLS refits the model with an added interaction LV (product of predictor and moderator composite scores). The **interaction effect** row is the payoff: if its p-value < 0.05, the moderator changes the X → Y relationship.

The **simple slopes** plot below the tables visualises this: X → Y at low (M – 1 SD), mean, and high (M + 1 SD) moderator values. If the slopes fan out, the moderator amplifies X; if they converge, it attenuates.

6.5 Nomological validity

Purpose. Check that every hypothesised path came out with the expected sign and significance. Sanity check before you write up the results.

Inputs. Auto-populated from your model, every path becomes a row with an **expected sign**

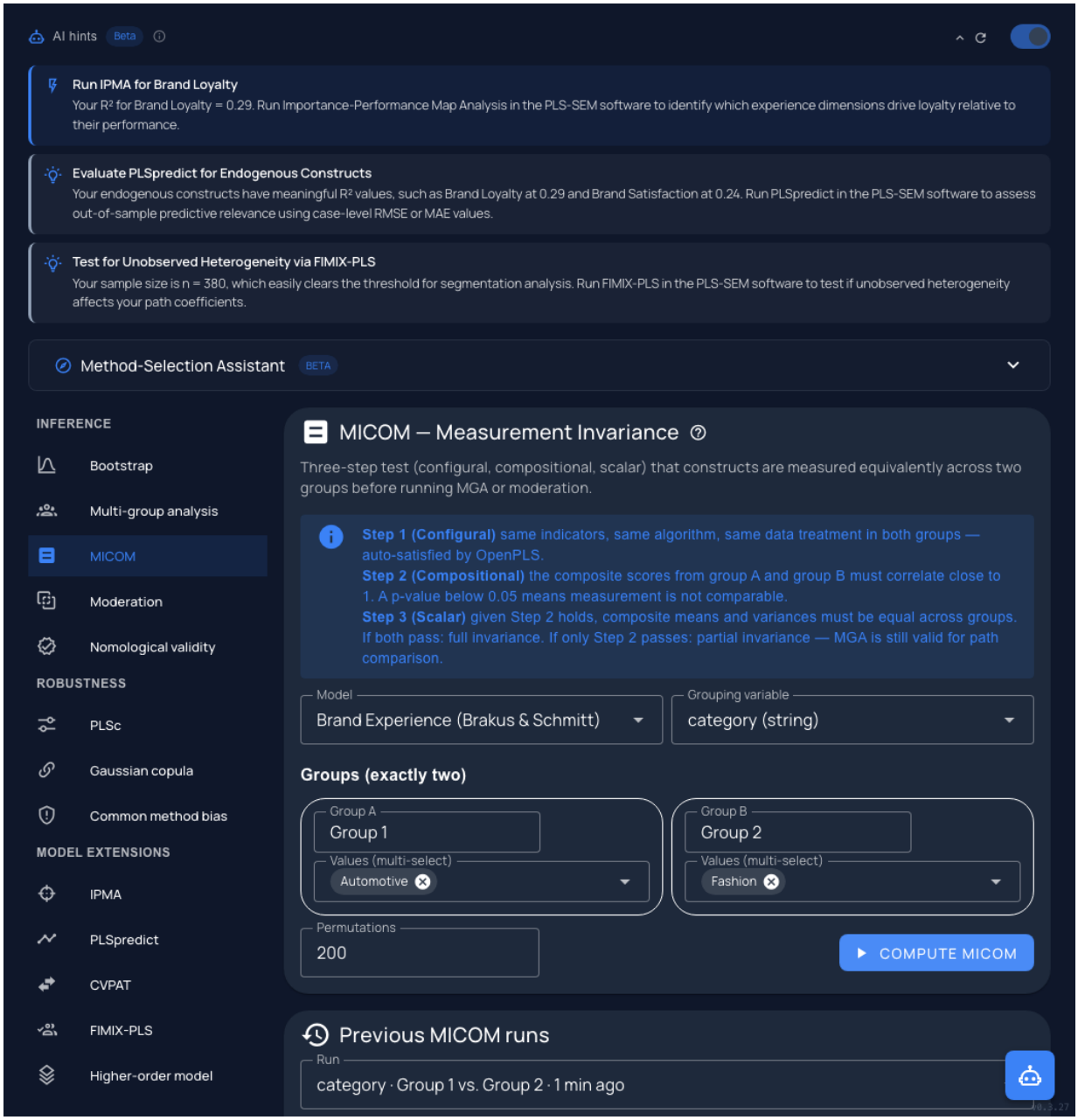

Figure 17 MICOM panel: same grouping-column and group-definition UI as MGA.

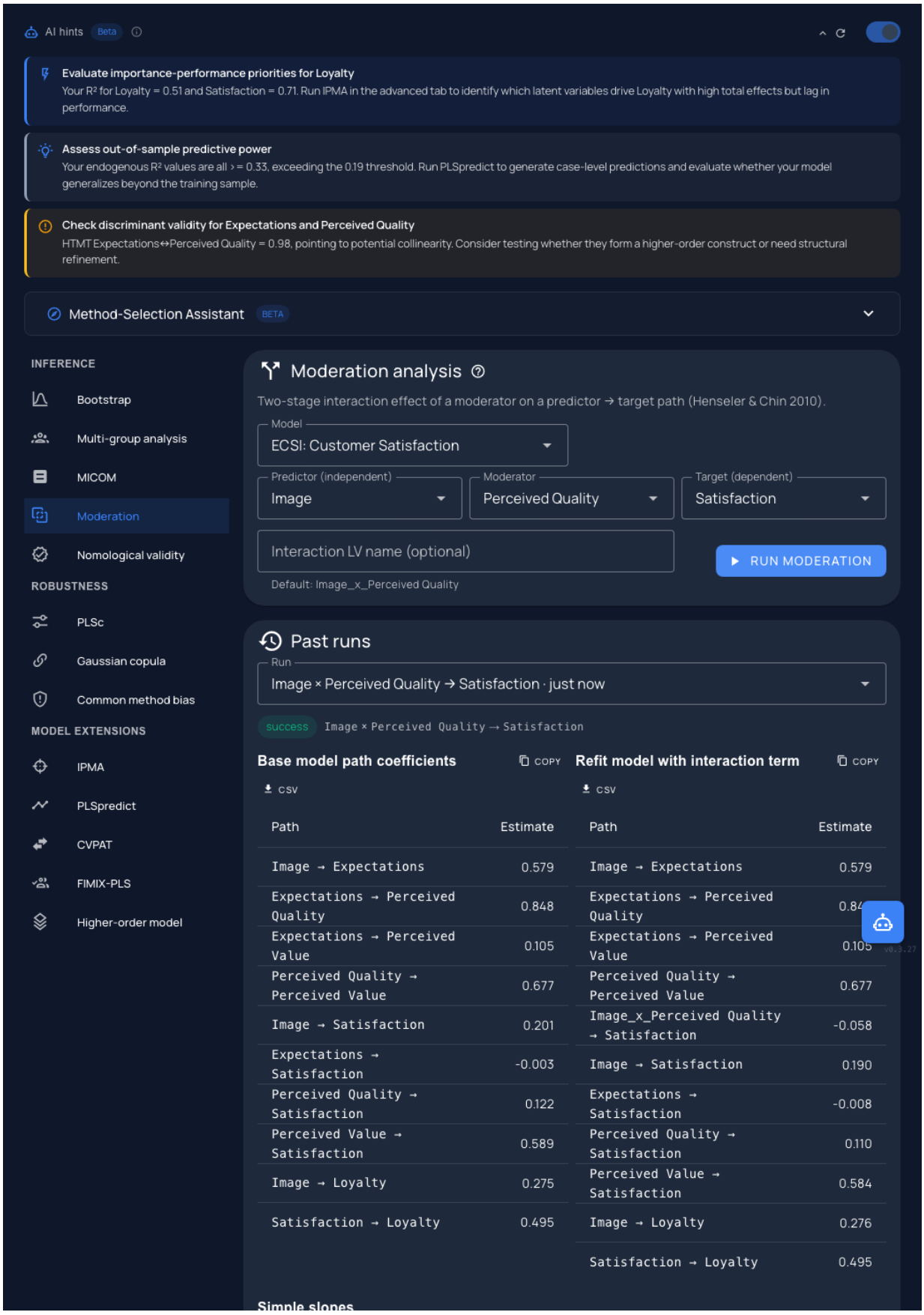

Figure 18 The Moderation panel: three LV dropdowns (Predictor / Moderator / Target).

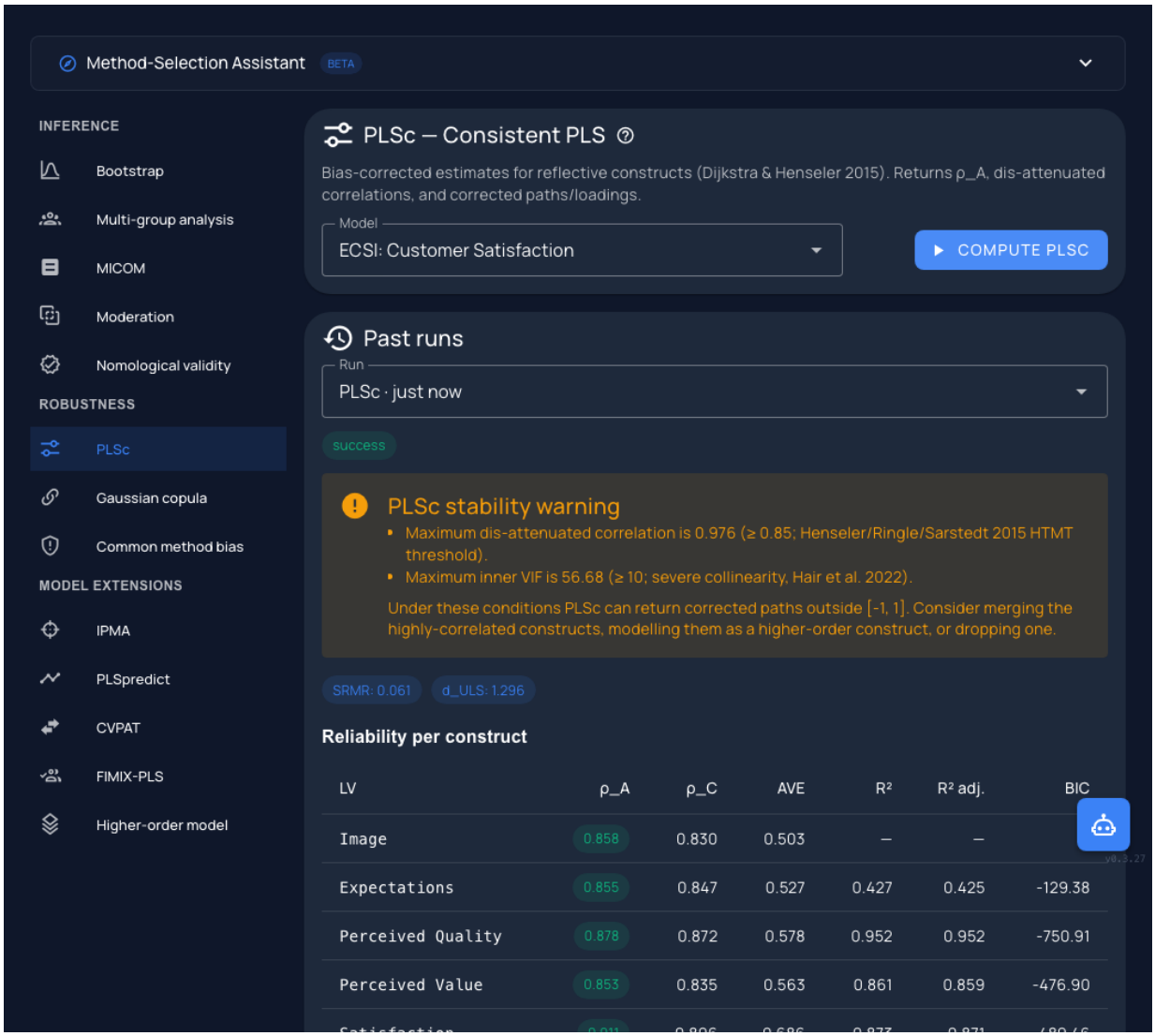

Figure 19 PLSc panel, one-click on models with only Mode-A LVs.

toggle: **+** for "hypothesised positive" or **–** for "hypothesised negative". Set the sign for each hypothesis, pick the significance level α (default 0.05), and click Run test.

Interpretation. For each hypothesis the table shows the estimated sign, the p-value from the last bootstrap, and a verdict (pass / fail) against your expected sign at α. Failures are candidates to re-examine or to report as null findings.

6.6 PLSc

Purpose. Consistent PLS estimation. Corrects the attenuation bias of standard PLS when constructs are truly common-factor rather than composite (Figure 19).

Inputs. None, click Run.

Interpretation. Rerun the exact same model with the PLSc correction applied. Path coefficients typically move slightly away from zero (the correction unbias-es the attenuation from finite reli-

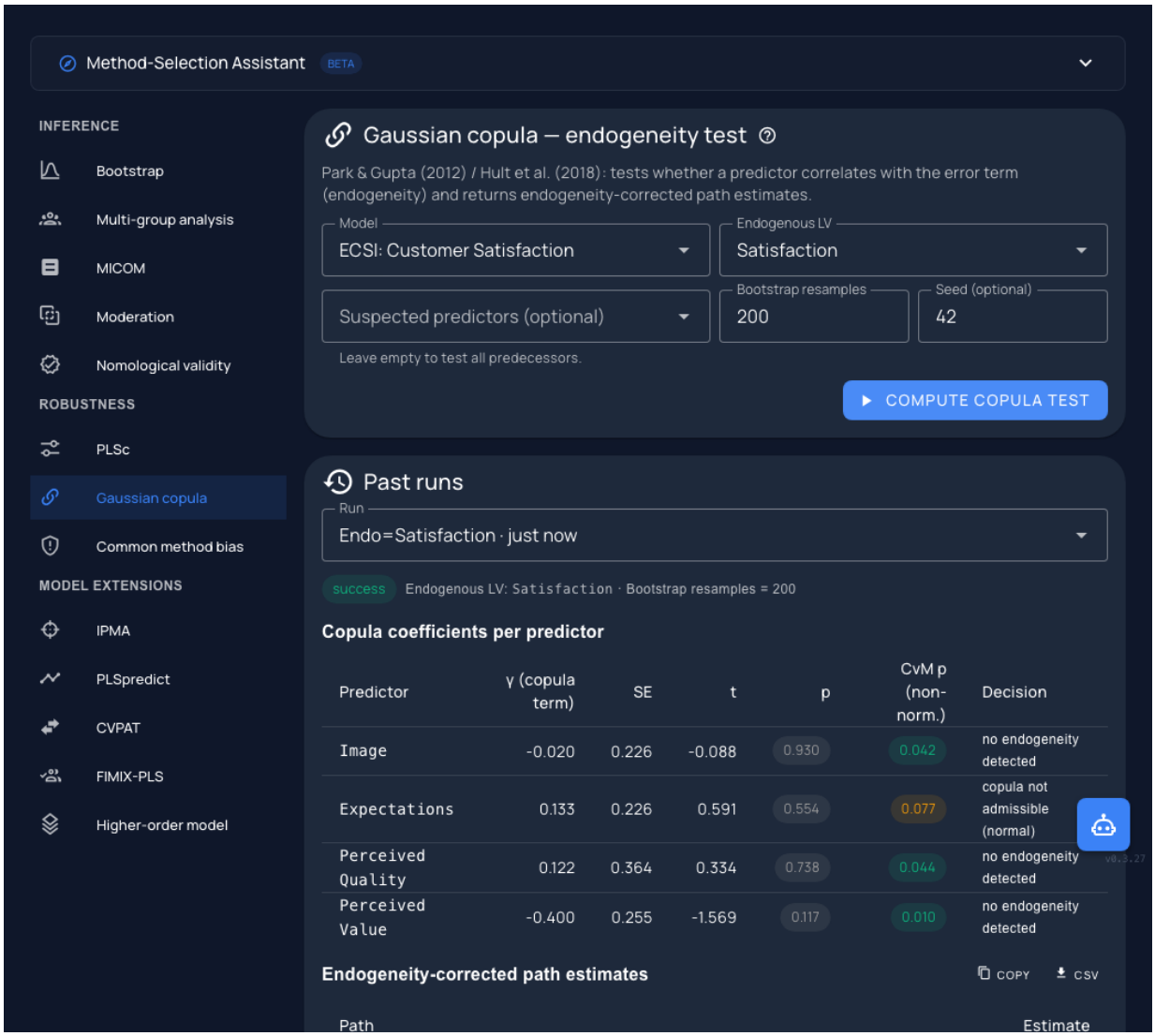

Figure 20 The Copula panel: endogenous LV picker, bootstrap resamples.

ability). Report both standard PLS and PLSc if reviewers expect factor-based interpretation.

Only valid when. All LVs are Mode A (reflective). Mode B LVs break the PLSc assumption.

6.7 Gaussian copula (endogeneity)

Purpose. Test whether a predictor is endogenous (correlated with the error term). If yes, return endogeneity-corrected estimates via the Park & Gupta (2012) / Hult et al. (2018) approach (Figure 20).

Inputs.

1. **Endogenous LV**: pick the LV whose predictors you suspect are correlated with the error term.
2. **Suspected predictors**: optional; leave empty to test all predecessors of the endogenous LV.

3. **Bootstrap resamples**: default 500, raise to 1000-5000 for publication-grade CIs.
4. **Seed**: optional.

Interpretation. For each suspected predictor OpenPLS reports a copula-term coefficient γ, its p-value, and a CvM (Cramér-von Mises) p for non-normality of the predictor (required by the method). The augmented-paths table shows the corrected coefficients if the copula term is significant.

Only valid when. Predictors are non-normal. If a predictor's CvM p is > 0.10 the copula correction can't be trusted.

6.8 Common method bias (CMB, Lindell-Whitney)

Purpose. Detect method bias via the Lindell & Whitney (2001) marker-variable procedure (Figure 21).

Inputs.

1. **Marker variable**: a theoretically unrelated numeric column that is NOT an indicator of any LV in the model. Examples: demographic like `tenure_years` or `age`.
2. **Significance** α: default 0.05.

Interpretation. OpenPLS partials out the smallest observed marker-LV correlation from every construct correlation and reports which paths lose significance. Paths that flip from significant to non-significant are suspect for common-method inflation.

Only valid when. Your dataset has a numeric column outside the model. If not, pick a different sample or run Harman's single-factor test manually.

6.9 IPMA

Purpose. Importance-Performance Map Analysis. Answers "which constructs matter most for the target, and which of them are under-performing?" (Figure 22)

Inputs.

1. **Target LV**: usually the outcome you care about (Loyalty, Adoption, …).

Interpretation. For each predecessor of the target, OpenPLS computes:

- **Importance** = total effect (direct + indirect) on the target
- **Performance** = mean of the LV's composite score, rescaled to 0-100

Plotted as a scatter (Performance on x, Importance on y). Top-left = high importance, low performance = the biggest improvement lever. Top-right = high both = strengths to preserve.

6.10 PLSpredict

Purpose. Assess **out-of-sample** predictive relevance. R^2 is in-sample; PLSpredict tells you if the model generalises (Figure 23).

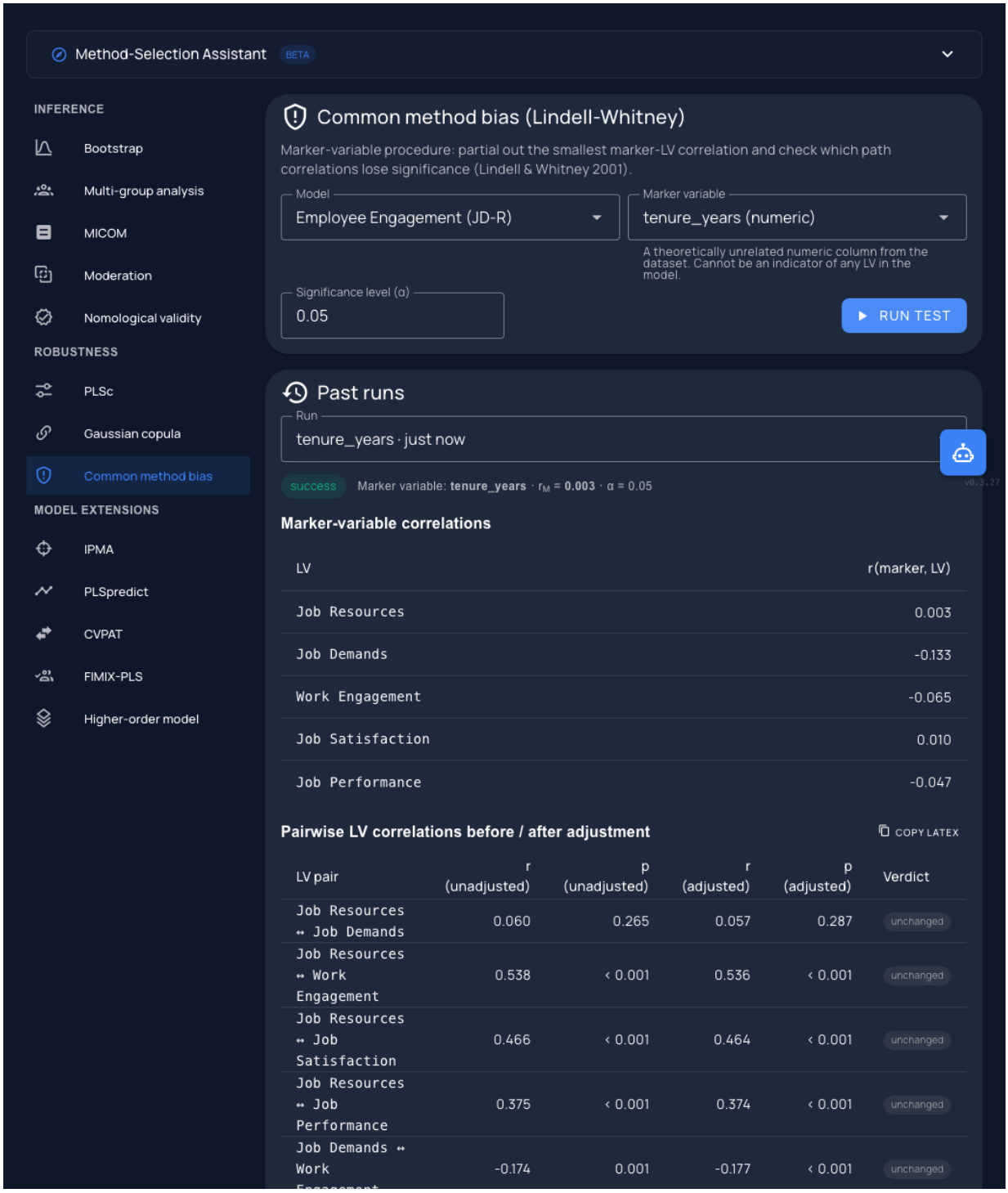

Figure 21 CMB panel: pick a marker column and run.

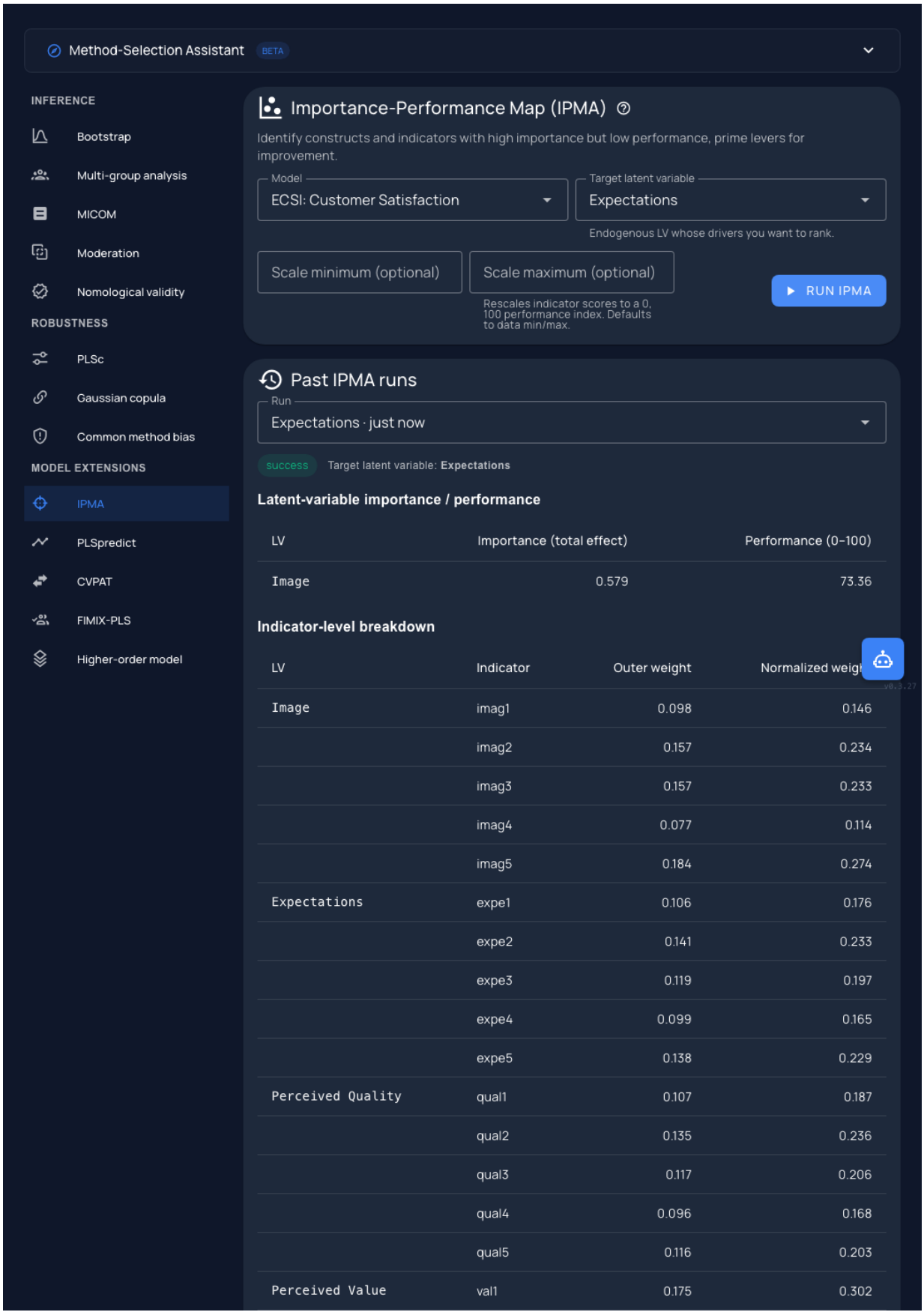

Figure 22 IPMA panel: pick the target LV, run.

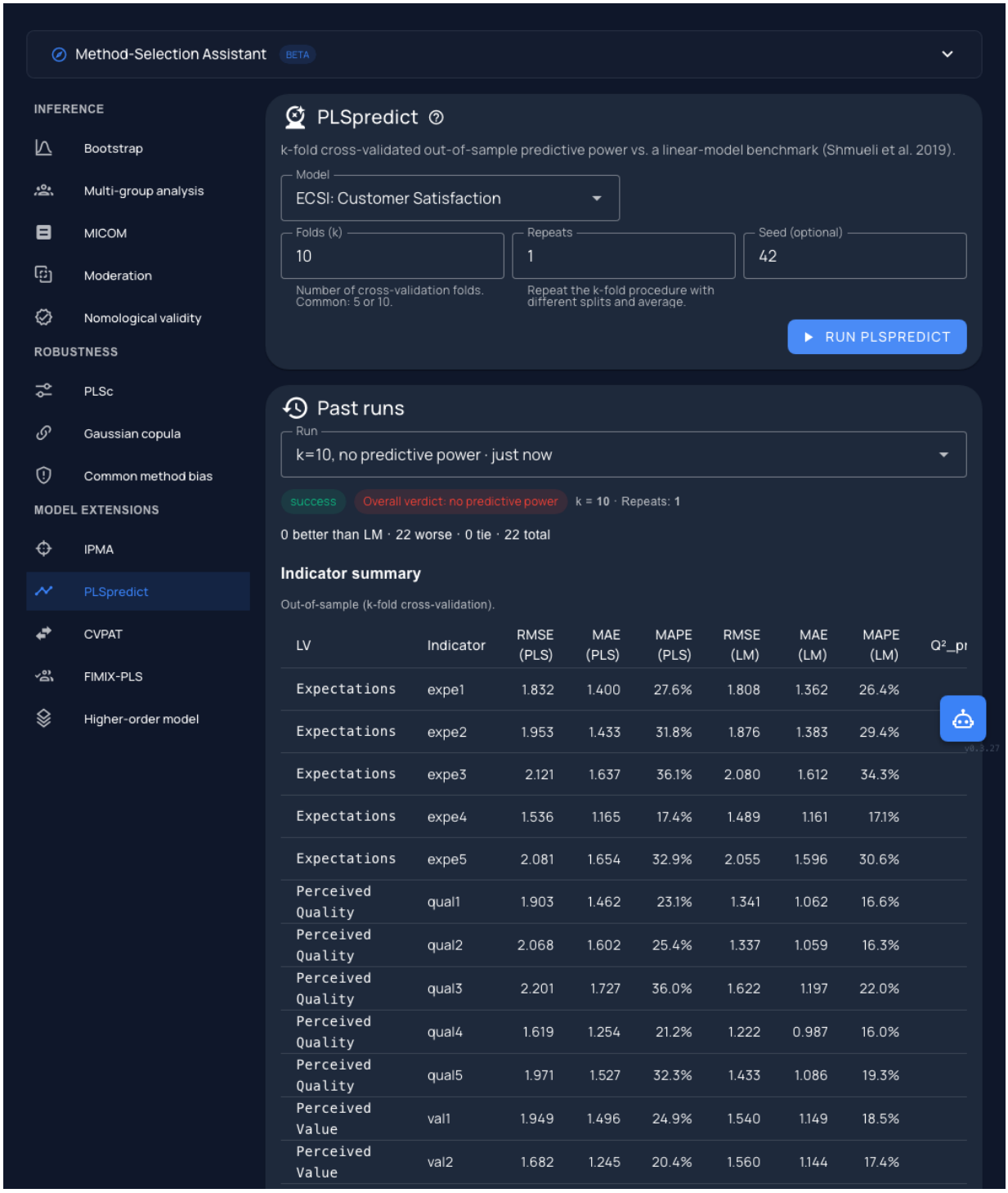

Figure 23 PLSpredict panel: k-fold, repeats, seed.

Inputs.

1. **k-fold**: default 10. Higher = smaller test folds.
2. **Repeats**: default 1. Increase to 10+ to average across different fold assignments and reduce variance.
3. **Seed**: for reproducibility.

Interpretation. For each indicator the table lists:

- **Q^2_predict**: cross-validated predictive R^2. Above 0 meaningful; higher is better.
- **RMSE_PLS** vs. **RMSE_LM**: PLS predictions vs. a linear benchmark. If PLS RMSE is lower for the majority of indicators, PLS has predictive value beyond the linear alternative.

6.11 CVPAT

Purpose. Cross-Validated Predictive Ability Test. Compares the PLS model against a benchmark (indicator average or linear model) via a t-test on out-of-sample loss (Figure 24).

Inputs. Benchmark + k-fold + repeats + seed. Defaults are safe.

Interpretation. For each endogenous LV OpenPLS reports the loss difference (PLS – benchmark) with a p-value. Negative loss difference + $p < 0.05$ = PLS predicts better than the benchmark, a strong claim.

6.12 FIMIX-PLS

Purpose. Finite-Mixture PLS. Discovers latent classes of respondents when you suspect heterogeneity but don't know the grouping (Figure 25).

Inputs.

1. **Number of classes (K)**: try 2..5, compare fit criteria.
2. **EM restarts**: best of n random starts. More = more robust.
3. **Max iterations, tolerance**: convergence controls; leave defaults.

Interpretation. For each K, OpenPLS estimates class-specific paths + a mixing proportion ρ per class. Compare **AIC**, **BIC**, **CAIC**, and **entropy (EN)** across K values, the elbow in BIC combined with EN > 0.6 identifies the "right" K.

Once K is decided, export the class assignments and re-run standard PLS per class for detailed inference. Or use them as a grouping in MGA.

6.13 Higher-order construct (HOC)

Purpose. Combine multiple first-order LVs into a single second-order construct via the disjoint two-stage approach (Sarstedt et al. 2019) (Figure 26).

Inputs.

1. **Name**: the HOC's LV name, e.g. `BrandExperience`.

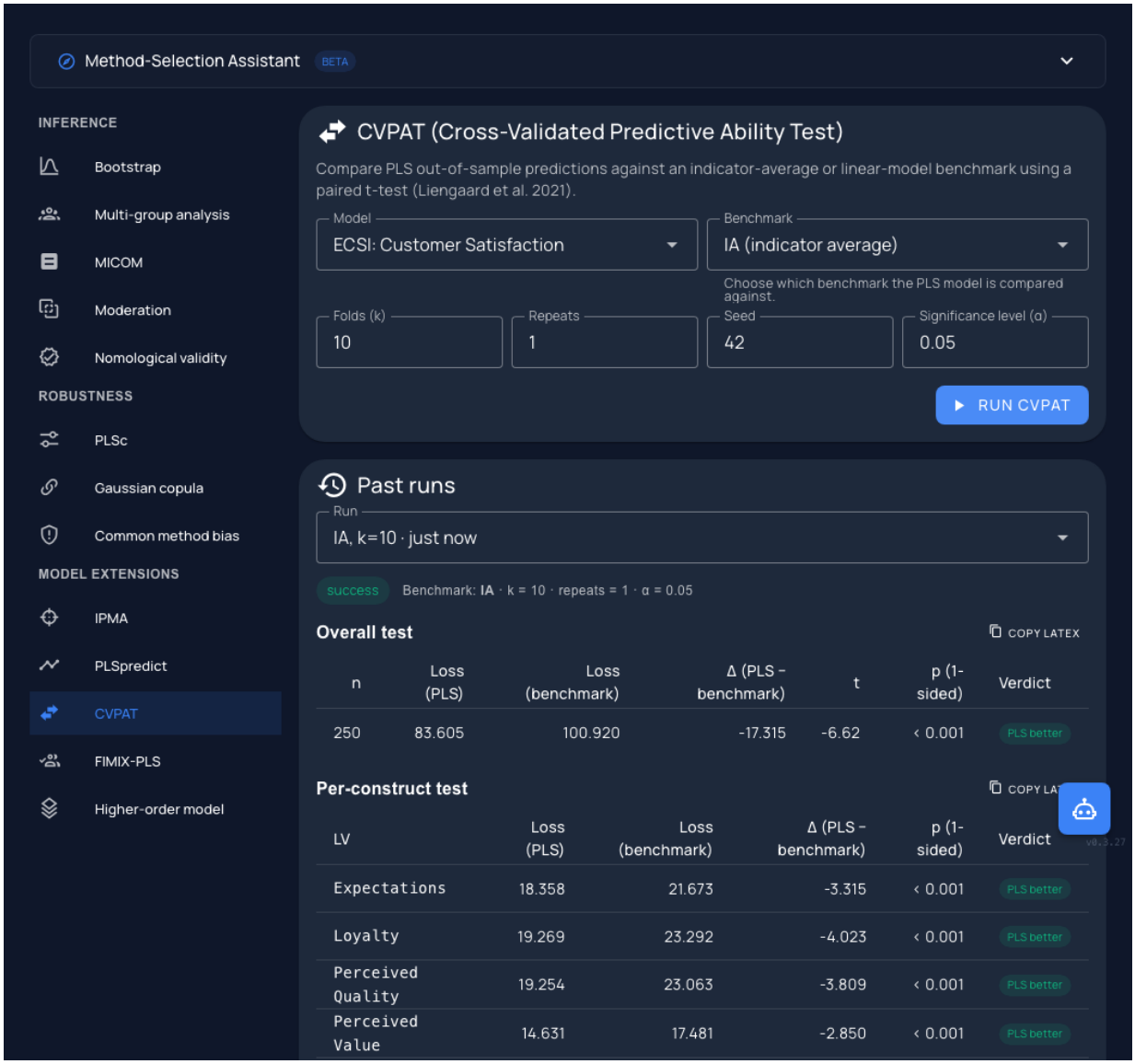

Figure 24 CVPAT panel: benchmark picker (IA = indicator average, LM = linear model), k-fold, repeats.

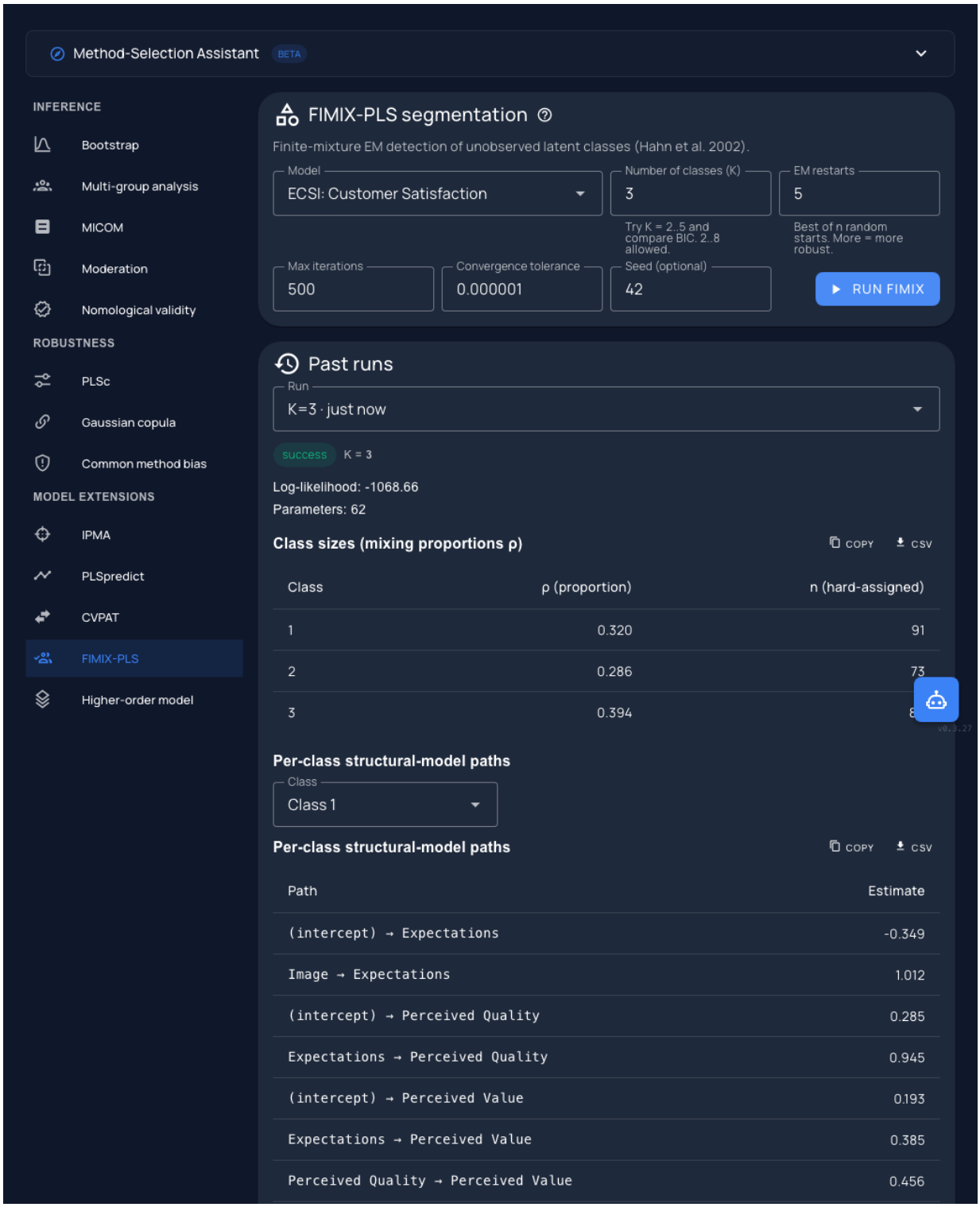

Figure 25 FIMIX panel: number of classes, EM restarts, tolerance.

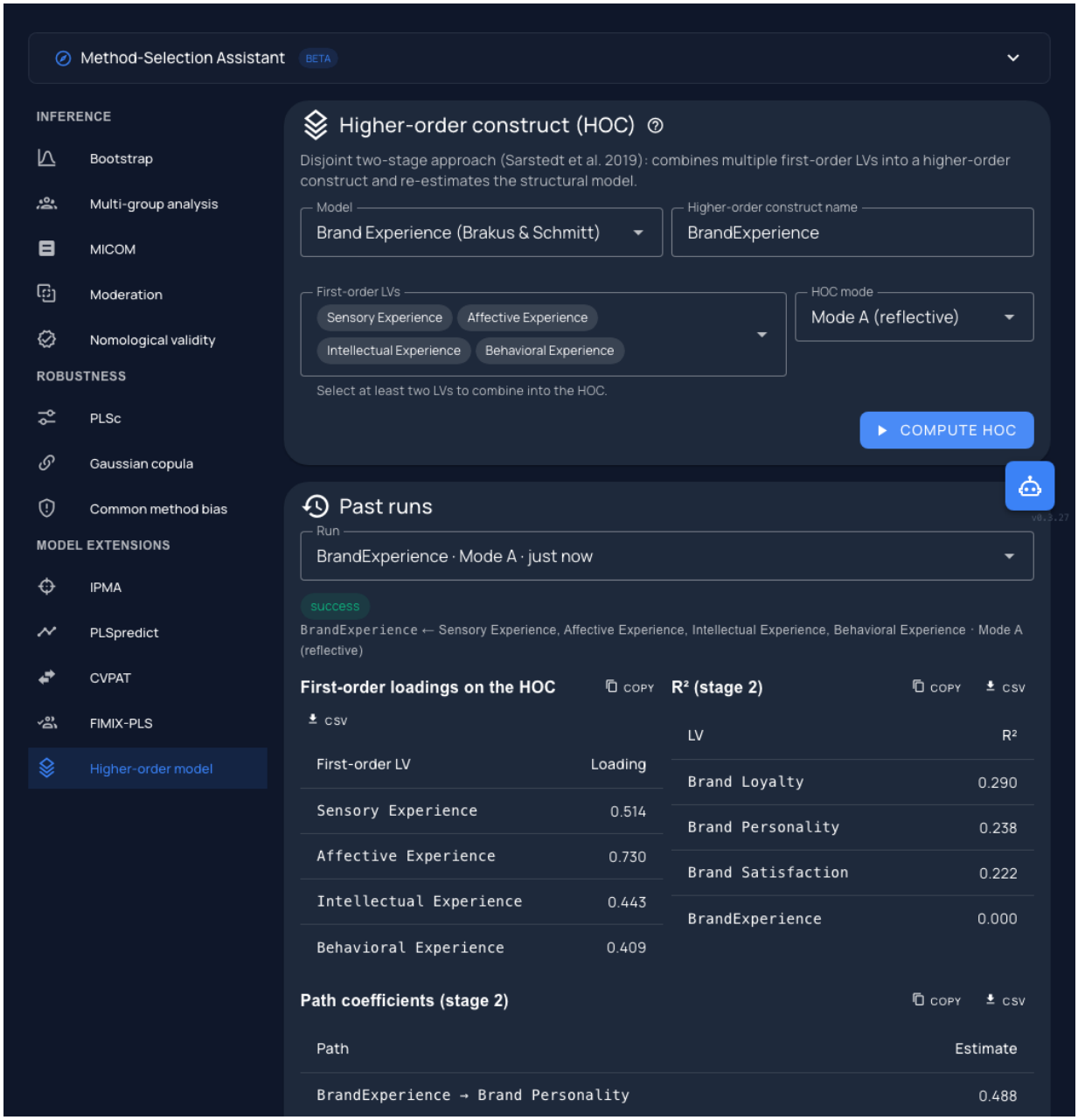

Figure 26 HOC panel: name the HOC, pick 2+ first-order LVs, choose Mode A or B.

2. **First-order LVs**: the components to combine (at least 2).
3. **HOC mode**: A (reflective) or B (formative).

Interpretation. OpenPLS runs stage 1 (normal PLS to get first-order LV scores), then stage 2 (a new PLS with the HOC in place of the four dimensions). Reports:

- **Loadings**: first-order LVs on the HOC (Mode A) or weights (Mode B).
- **R^2 (stage 2)**: variance of the HOC's downstream targets explained.
- **Paths (stage 2)**: structural coefficients in the re-estimated model.

Compare Mode A vs. Mode B by refitting with the other mode and checking which produces cleaner reliability and paths.

7 Exports

Every result in OpenPLS can leave the tool in one of four formats. The export buttons live in the Results-tab header, above the KPI cards, and are also available from the model-export dropdown in the Editor toolbar.

7.1 PDF report

What you get. A print-ready PDF with:

- Cover with model + dataset metadata
- Path diagram (rendered from the current Editor positions)
- Every result table from the Results tab, formatted for print
- References to the OpenPLS methods (each metric cites the paper that defined it)

When to use. Attach to a paper submission, share with a non-technical collaborator, include in a slide deck.

Filename. `{model_name}_report.pdf`, sanitised to strip spaces and slashes.

7.2 Excel workbook (XLSX)

What you get. One workbook with one sheet per result table. Sheet names as they appear in the file:

- `Overview`, run metadata (compute time, GoF, SRMR, LV/path count)
- `Inner Summary`, R^2, R^2 adjusted, AVE, communality per LV
- `Model Fit`, SRMR + d_ULS (only when computed)
- `Path Coefficients`, estimates for every inner-model path
- `Inner Model`, with bootstrap SE, t, p, CI when available
- `Outer Model`, loadings (Mode A) or weights (Mode B) per indicator
- `Effect Sizes` f^2, direct effect sizes for every predictor path
- `Predictive Relevance` Q^2, cross-validated redundancy per LV

- `VIF, Outer` and `VIF, Inner`, collinearity
- `HTMT`, heterotrait-monotrait matrix
- `Reliability`, Cronbach α, composite reliability, eigenvalues
- `Model, LVs` and `Model, Paths`, the model spec itself

Conditional sheets, added only when the corresponding advanced run exists:

- `Nomological Validity`, per-hypothesis sign + verdict
- `CMB (Lindell–Whitney)`, marker-corrected correlations
- `CVPAT`, cross-validated predictive-ability test

When to use. Reviewer wants raw numbers, or you want to re-format tables in your paper's house style.

Filename. `{model_name}_results.xlsx`.

7.3 Report JSON bundle

What you get. A self-contained JSON with:

- Full model specification (LVs, indicators, paths, positions, measurement mode)
- Dataset schema (column names, types, missing counts)
- Every result table verbatim
- Compute metadata (timestamp, duration, OpenPLS version)

When to use. As **supplementary material** for a paper. Anyone who imports the JSON can inspect exactly what you estimated without running OpenPLS.

Filename. `{model_name}.report.json`.

7.4 Model export (Editor toolbar)

The Editor toolbar has a **download** icon opening a menu with two model-only export formats:

- **JSON (`.openpls.json`)**: full OpenPLS model including positions. Round-trips: you can import this into another OpenPLS project to reuse the model with a different dataset.
- **SmartPLS (`.splsm`)**: an XML file compatible with SmartPLS 3 and 4. Lets you hand off the model to a collaborator on that tool.

These are model-only; they don't contain results. Use the Report JSON bundle for that.

7.5 LaTeX snippets

Every result table in the Results tab has a small **Copy LaTeX** icon at the top right. Click it, paste directly into your paper's `.tex` source. The generated LaTeX uses `booktabs` (`\toprule`, `\midrule`, `\bottomrule`) and matches conventional table style in JMS, MIS Quarterly, and similar journals.

The LaTeX snippets embed the model name, dataset name, and OpenPLS version in the table caption for traceability. Adjust the caption in your paper if you want a cleaner look.

7.6 Reproducibility

For a paper, the ideal reproducibility bundle is:

1. The **Report JSON** (results and model + dataset schema).
2. The **raw dataset** (a supplementary file, if licensing allows).
3. A short note pointing to `openpls.app` for the estimation.

Any reader can then re-run the exact same computation without depending on your local install.

The exported values are the same numbers you see in the Results tab. Comparisons against SmartPLS on published PLS-SEM datasets are ongoing; latest results are linked from `openpls.app/validation`.

8 Collaboration

OpenPLS treats projects as shareable workspaces. Invite co-authors, research assistants, or reviewers to view or edit the same model. Every action leaves an audit trail so you can see who did what and when.

Everything collaboration-related lives under **Manage** in the tab strip. Three sub-views: **Team**, **Activity**, **Settings**.

8.1 The team panel

Open **Manage** → **Team** to reach the panel shown in Figure 27.

Three sections stack:

1. **Team**: every current member with their name, email, avatar, and role pill. The owner (project creator) is marked. To the right of each row are the role change and remove actions. The remove action is only visible to the owner.
2. **Invite person**: email input + role dropdown + Invite button. Sends a signed invitation email via Resend. The recipient can accept via a link that opens `/invitations/{id}`.
3. **Pending invitations**: one card per outstanding invite with the invited email, timestamp, and a **Revoke** action.

8.2 Roles

Three roles:

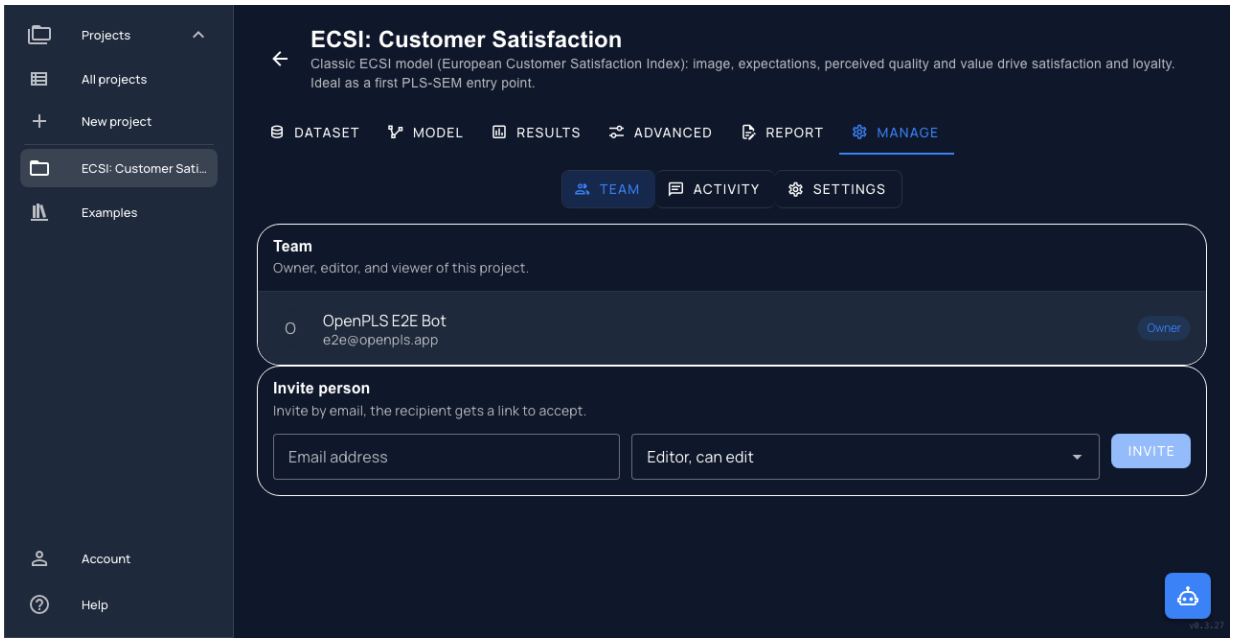

Figure 27 The Team sub-view: members list on top, invite form below, pending invitations at the bottom.

- **Owner**: creator, can do everything including delete the project.
- **Editor**: can edit dataset, model, run computes, run advanced methods, edit settings. Cannot delete the project or invite new members (only the owner can).
- **Viewer**: read-only. Sees every result and export, cannot change anything.

Role changes take effect immediately and appear in the activity log.

8.3 Sending an invite

1. Type the invitee's email in the **Email address** field.
2. Pick their role: **Editor, can edit** or **Viewer, read only**.
3. Click **Invite**.

OpenPLS creates an invitation entry in the database and sends the invitee an email with the accept link. If the email address already belongs to an OpenPLS user, the invitation lands under their own `/invitations` route directly.

Copy invitation link. Each pending-invitation card has a "Copy link" icon that puts the accept URL on the clipboard, useful when the recipient's email filters aggressively.

Revoking. Click **Revoke** on a pending card. The invitation becomes invalid; the accept link no longer works.

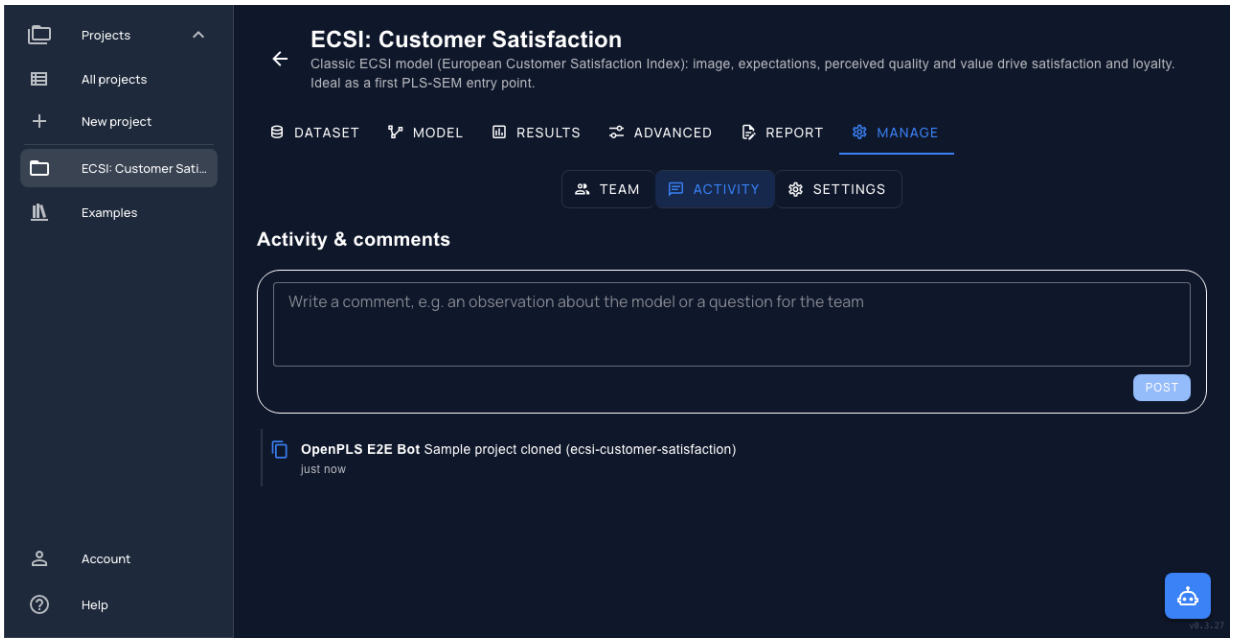

Figure 28 The Activity sub-view: chronological feed of every project event, with icons and actor.

Invited users who don't yet have an OpenPLS account can create one via the accept flow (their email is pre-filled). The invitation holds their spot until they accept, no need to sign up first and be re-invited.

8.4 The activity log

Switch to the **Activity** sub-view (Figure 28) for the audit trail.

The backend logs one entry per event. Events currently emitted:

- **Team events**: member invited, invitation accepted, invitation revoked, invitation mail failed, member role changed, member left, sample project cloned.
- **Model computed**: a full main compute finished.
- **Advanced-method runs**: MICOM, Moderation, PLSc, Copula, IPMA, PLSpredict, FIMIX, HOC, Nomological, CMB, CVPAT. One entry per successful completion.

Not (yet) logged: Bootstrap runs, MGA runs, individual model saves, dataset uploads, and dataset re-parses. Bootstrap and MGA run on Cloud Run services that write their run docs directly without touching the activity log, check the panel's own "Past runs" history instead.

Comments. The activity feed also includes a comment composer. Comments are stored as top-level entries and merged into the feed chronologically, they aren't threaded under a specific event. Use comments to annotate what a colleague should notice about a run.

8.5 Leaving a project

Editors and viewers see a **Leave** button on their own row in the team list. Clicking it drops the project from your workspace and frees the seat. The owner cannot leave, they must first delete the project or transfer ownership.

> Transferring ownership is not yet exposed in the UI. In the current release, the owner is the person who clicked ”New project” and stays the owner for the lifetime of the project. Ownership transfer is on the roadmap, until then, contact support if you need to hand a project over.

8.6 What collaborators see

Editors get the same UI as the owner minus the danger-zone actions. Viewers see:

- All tabs, Dataset, Model, Results, Advanced, Manage
- All read-only content
- Export buttons, viewers can download PDFs, XLSX, JSON
- Copy-LaTeX icons on tables

Viewers cannot:

- Edit the model or dataset
- Compute or trigger any advanced method
- Invite anyone or change roles
- Change settings

If a viewer tries to click an edit action, OpenPLS shows a subtle “permission denied” toast rather than silently failing.

9 Settings and language

The **Settings** sub-view under Manage handles project-level metadata and the danger zone. The **language switcher** in the topbar handles the UI locale for the current session and persists it to your user profile so it follows you across devices.

9.1 Project settings

Open **Manage** → **Settings** to reach the panel shown in Figure 29.

Two sections:

1. **Project details**: the same name + description fields from the New-project dialog, editable at any time. Click Save to persist. A success alert confirms the write.

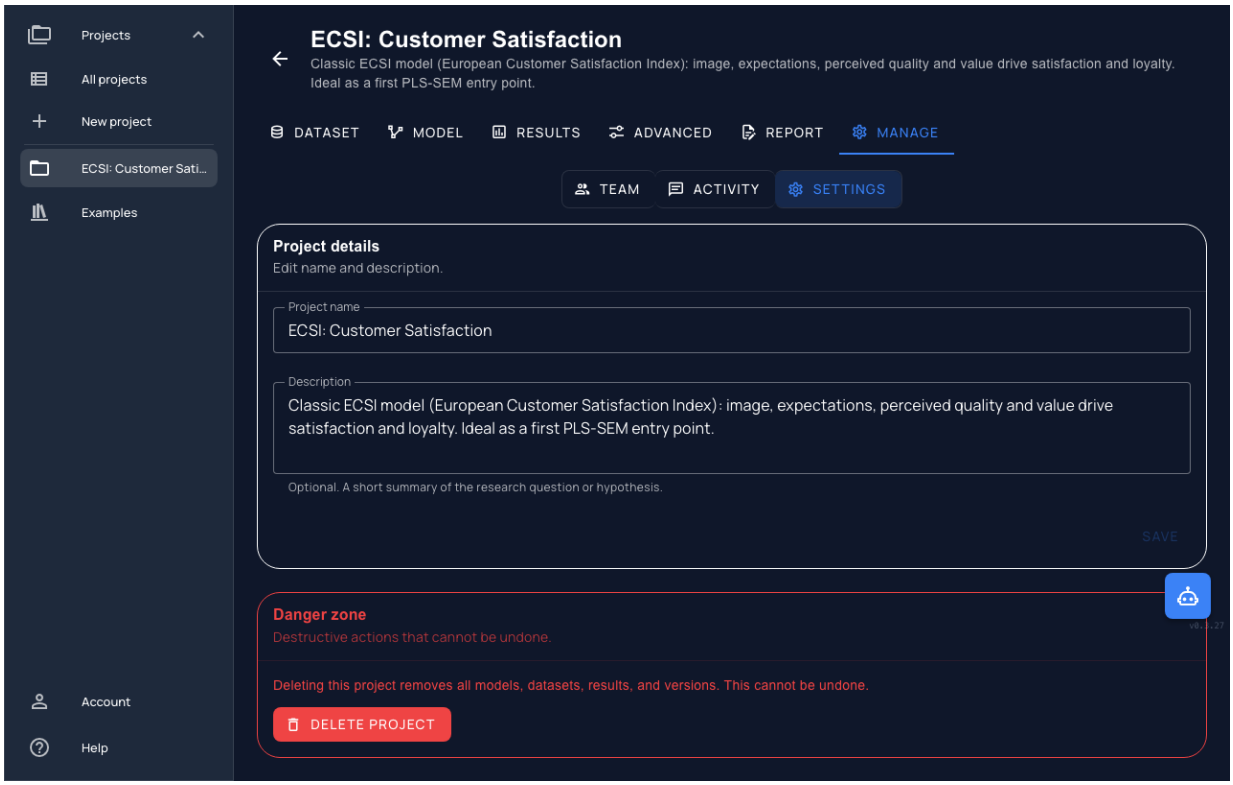

Figure 29 The Settings sub-view: name + description form on top, danger zone at the bottom.

2. **Danger zone**: the **Delete project** button. Clicking it opens a confirm dialog spelling the project name for double-check. The delete cascades: the project, its datasets, models, results, versions, and all advanced-method runs are removed. Cannot be undone.

Only the owner sees the danger-zone card. Editors and viewers see the project-details section only.

> Deleting a project also deletes the invitation history and the activity log. If you want an archive of the audit trail before deletion, export the Activity view first (available via the download icon in the Activity toolbar).

9.2 Language

OpenPLS ships with eight UI locales:

- English (default)
- Deutsch
- Español
- Français
- Português (Brasil)
- Simplified Chinese
- Italiano

- Japanese

Click the flag in the topbar to open the language menu (Figure 30).

Pick a locale. OpenPLS fetches the message bundle (small, ~30-60 KB) and re-renders. The choice is persisted to your user profile, next time you log in from any device, you'll land in the same locale.

What is translated.

- All UI labels, buttons, dialogs, toasts
- Help pages (`/help`)
- Report PDF captions and section headings
- Email templates for invitations, password reset, verification

What is not translated.

- Metric values and column names, statistics stay in the scientific standard (Latin-based numeric labels, English column headers). This is intentional so numbers can be copied cleanly into English-language papers.
- Sample-project citations and long descriptions, the science citations are in the original language of publication.

> Email templates (invitation, password reset, verification) are currently English-only, even when the recipient's profile locale is set to another language. Full email localisation is on the roadmap.

9.3 Account settings

Your own profile lives at `/account` (link in the sidebar). Editable:

- **Display name**: shown in the team panel and activity log.
- **Avatar**: click to upload; PNG or JPEG, 5 MB max, cropped to a circle.
- **Preferred locale**: same effect as the topbar switcher but persisted directly.
- **Password**: change here. Type the current password once and the new password twice; a confirmation email is not required.
- **Email preferences**: a single toggle "Receive product updates by email". Transactional emails (invitations, password reset, verification) are always sent regardless of this setting.

Sign out via the round avatar button at the top right of the topbar (next to the theme toggle and language switcher).

9.4 Theme

The topbar theme toggle (sun/moon icon) switches between light and dark mode for the current session. Dark mode is the default. The theme choice is stored in `localStorage`, so it doesn't

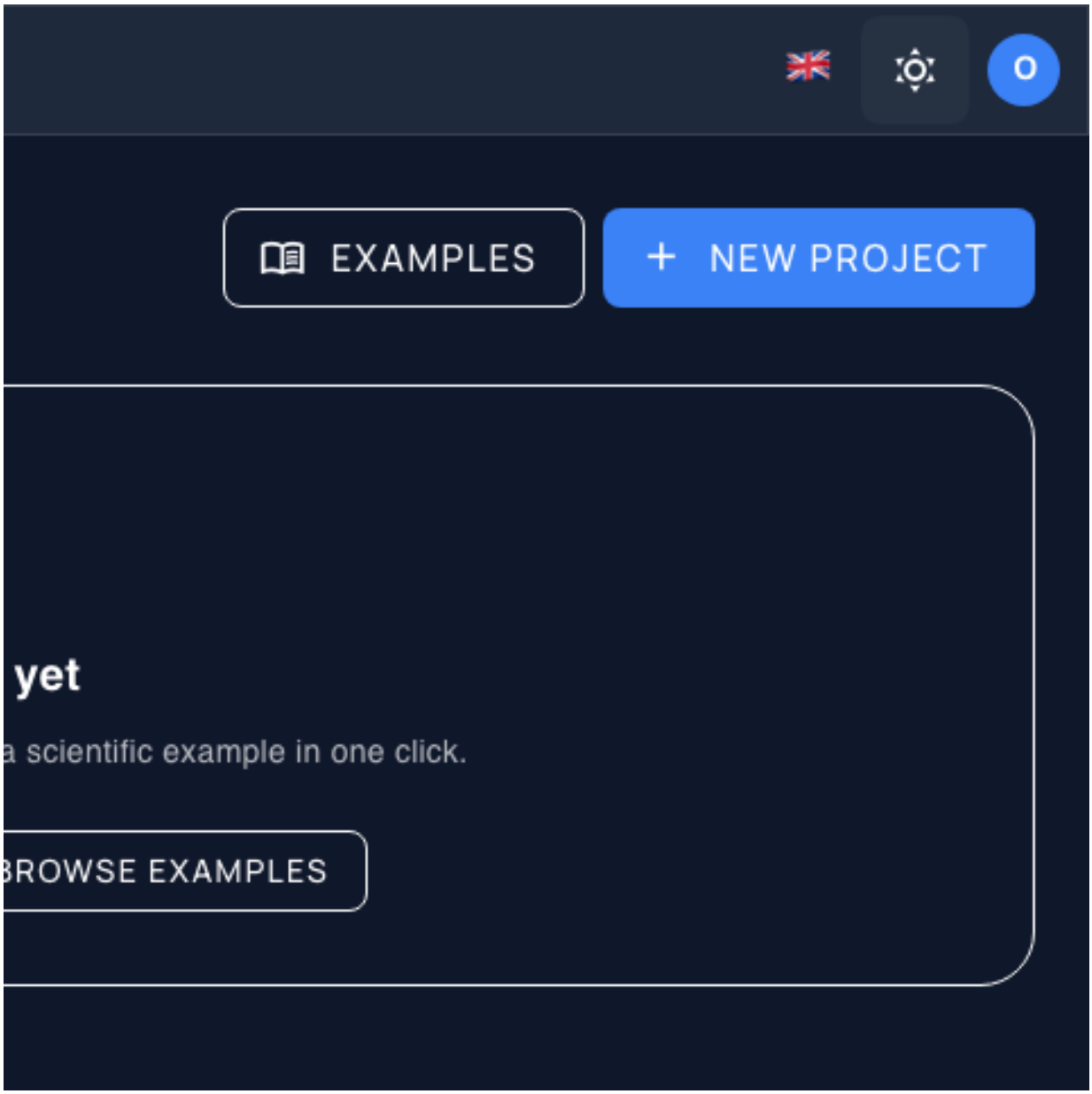

Figure 30 The language dropdown from the topbar. The current locale is highlighted.

sync across devices.

10 Help and further reading

10.1 In-app help

OpenPLS ships with an in-app help center at `/help`, accessible from the sidebar of every project or from `cloud.openpls.app/help`. Figure 31 shows the overview page.

Six sections:

- **How-to**: task-oriented step-by-step guides mirroring this guide's chapters, but bite-sized. Fastest way to look up "how do I …" without loading a full PDF.
- **Metrics**: every metric OpenPLS reports, with the formula, interpretation thresholds, and the paper that defined it. If you're unsure why R^2 adjusted is different from R^2, start here.
- **Choose**: decision trees for common questions: "when is my construct reflective vs. formative?", "which advanced method applies to my hypothesis?", "how big should the bootstrap be?".
- **Advanced**: a page per advanced method with the same content as Chapter 6 of this guide, plus per-method examples and interpretation walkthroughs. Deep-links from the app: clicking the small ? icon on any Advanced panel jumps to the matching help page anchor.
- **Pitfalls**: the recurring modeling mistakes we see in support tickets: sentinel values swallowing valid observations, wrong measurement mode, comparing incomparable groups. Read these before your first serious project.

The help pages are localised in all eight UI languages.

10.2 The OpenPLS engine (open-source)

The estimation code that powers OpenPLS is an open-source Python package called **openpls-engine**, released under GPL-3.0 at `https://github.com/jojacobsen/openpls-engine`. What you compute in the cloud can also be run locally with:

```
pip install openpls-engine
```

The package includes:

- The core PLS-SEM estimator (matches SmartPLS 4 outputs to 4 decimal places on validated cases)
- Every advanced method described in Chapter 6
- CLI + Python API for reproducible batch runs
- A Docker image for turn-key self-hosting

If you want to self-host the whole OpenPLS UI, the engine is the compute component; see the roadmap on the repo for the current state of Docker Compose orchestration.

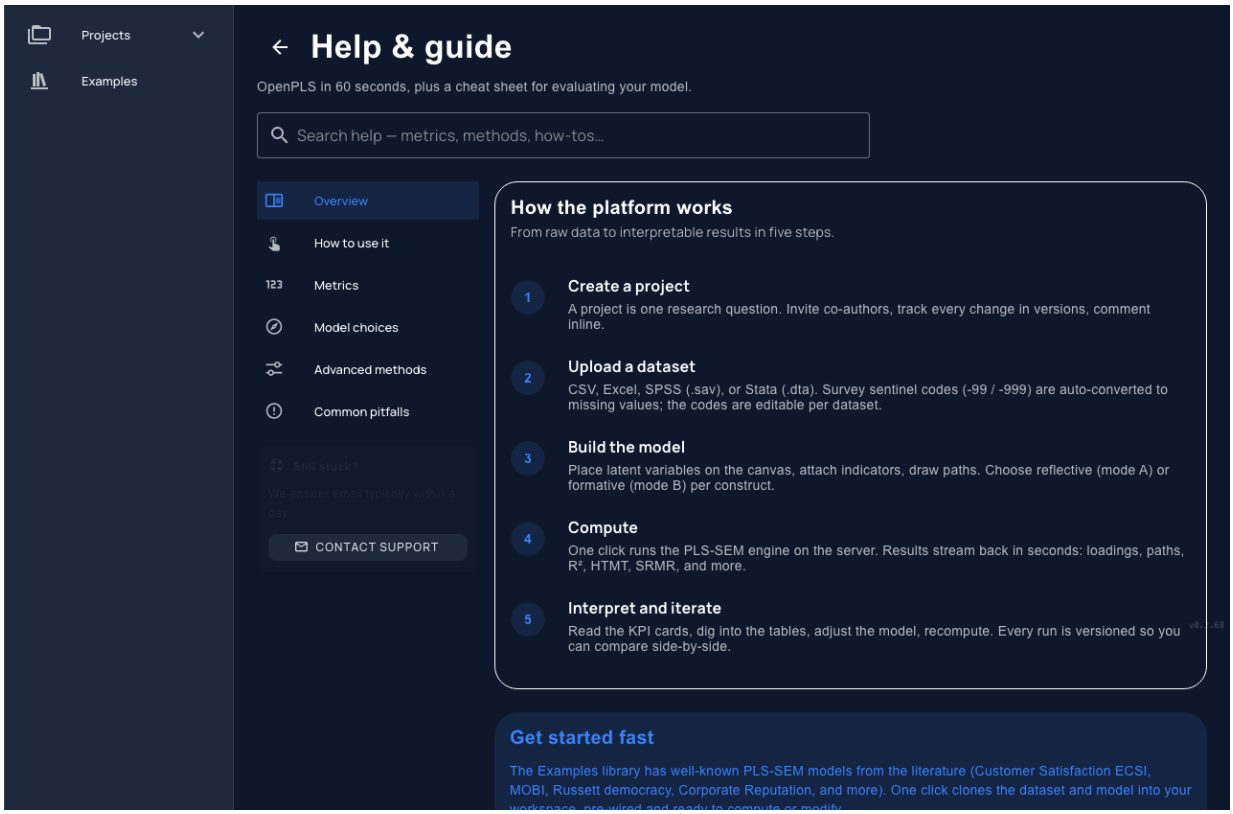

Figure 31 The Help overview: entry points to bedienung (how-to), metrics, choose-a-method, advanced, and common pitfalls.

10.3 Support, bugs, and feature requests

For anything related to the cloud app (`cloud.openpls.app`), write to `hello@openpls.app`. Include your project ID (visible in the URL) and, if reporting a bug, a short description of what you expected versus what happened.

Engine-specific bugs and feature requests can also be filed as GitHub issues at `https://github.com/jojacobsen/openpls-engine/issues`, or by email if you prefer.

10.4 Citing OpenPLS

If OpenPLS produced results in your paper, please cite:

Jacob, J. (2026). *OpenPLS Engine: An Open-Source Implementation of Partial Least Squares Structural Equation Modeling Validated Against SmartPLS 4 Across 14 Reference Cases.* SSRN Working Paper. `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=6869001`

This citation covers both the estimation engine and the cloud application, since they share a single validated implementation.

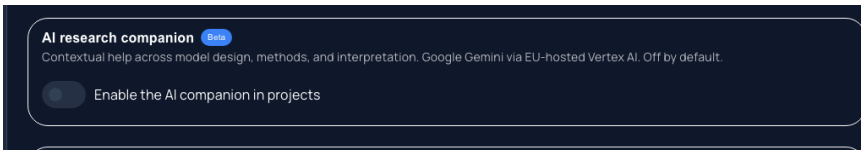

Figure 32 The AI Companion consent dialog. Read the four bullets, then click **Enable AI Companion**. Skip and it stays off; you can turn it on later in Account settings.

11 The AI Companion

OpenPLS ships with an **AI Companion** - an in-app assistant that reads your project state, cites peer-reviewed methodology papers, and helps with the parts of PLS-SEM that don't come from clicking buttons: choosing what to run next, drafting the write-up, anticipating what a reviewer will push back on.

The Companion is **opt-in**. Nothing is called until you accept the consent copy, and you can disable it any time in Account settings. This chapter walks each surface in the order you'll typically meet them.

> Every AI response is grounded in two sources: (1) the actual numbers in your project (dataset shape, model spec, latest compute result), which the assistant fetches through read-only tools; and (2) a curated library of open-access PLS-SEM papers whose abstracts are retrieved by semantic similarity to your question. The assistant does not have write access to your project - every applied suggestion is a click you make yourself.

11.1 Turning it on

First-time users see a one-step opt-in dialog after login. It summarizes what the Companion reads (your project data), what it sends onward for processing, and the usage budget (a soft cap of 100 turns per month per user prevents runaway usage).

If you dismissed the dialog earlier, open **Account** → **AI Companion** and toggle it there. The same page also carries a sub-toggle for **AI hints** (see below): with the Companion on but hints off, only proactive features stay silent - chat, model designer, and report drafting still work when you invoke them.

11.2 AI hints: the always-visible strip

Every tab that has a plausible next step displays a small **AI hints** strip at the top: one to three suggestions calibrated to what you have in this project right now. On the Dataset tab it might flag a column with 22 % missing values; on Results it might notice that your R^2 for Loyalty is 0.42 and suggest running IPMA; on Advanced it will point at whichever method your current state makes most compelling.

Hints fetch once per tab per session, then cache. Hit refresh to force a re-read when you've just

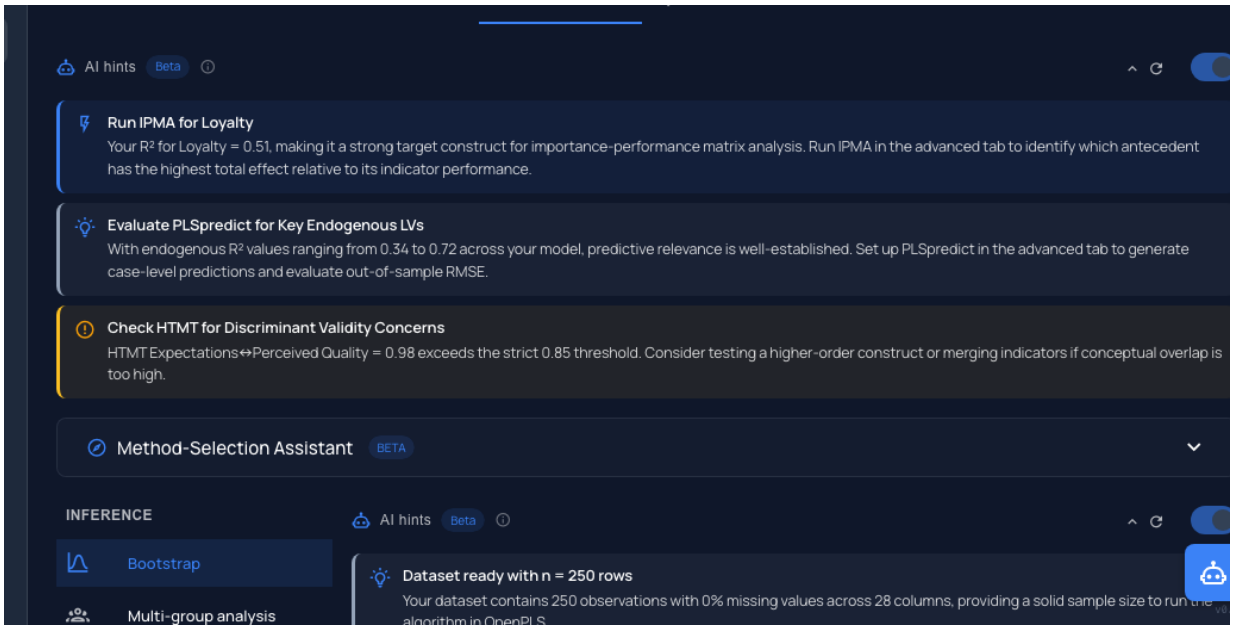

Figure 33 The AI hints strip on the Advanced tab. Header carries a **Beta** chip, an info tooltip, refresh, collapse, and the on/off toggle. Each hint has a severity icon: light-bulb for info, alert-circle for warning, flash for action.

imported a new dataset or edited the model.

Turning hints off individually. Not every user wants the strip in their face. The toggle at the right of the strip flips a per-user `aiHintsEnabled` flag: hints stop fetching, but the rest of the Companion (chat, designer, report) continues to work when you open it. Off → strip collapses to a muted placeholder that stays out of the way.

11.3 Chat: the floating drawer

A small robot button lives permanently in the bottom-right corner. Click it to open the chat drawer.

The empty state offers three ready-made prompts:

1. **Explain my results in plain language.** Reads the latest compute output and writes a one-paragraph summary calibrated to your specific numbers.
2. **Which advanced method should I run next?** Applies the same rules engine as the Method-Selection Assistant (see below) and returns a ranked shortlist inline.
3. **Are my constructs valid?** Walks the reliability + HTMT + AVE tables and flags any threshold violations by name.

Or type your own. Anything about the project - model design, dataset quality, method choice, path interpretation, reviewer pushback - is in scope. The Companion has four read-only tools:

- **getProject** - name, description, member count
- **getModel** - LVs with modes, indicators, structural paths
- **getDataset** - row and column count, per-column statistics (missingness, min/max/mean),

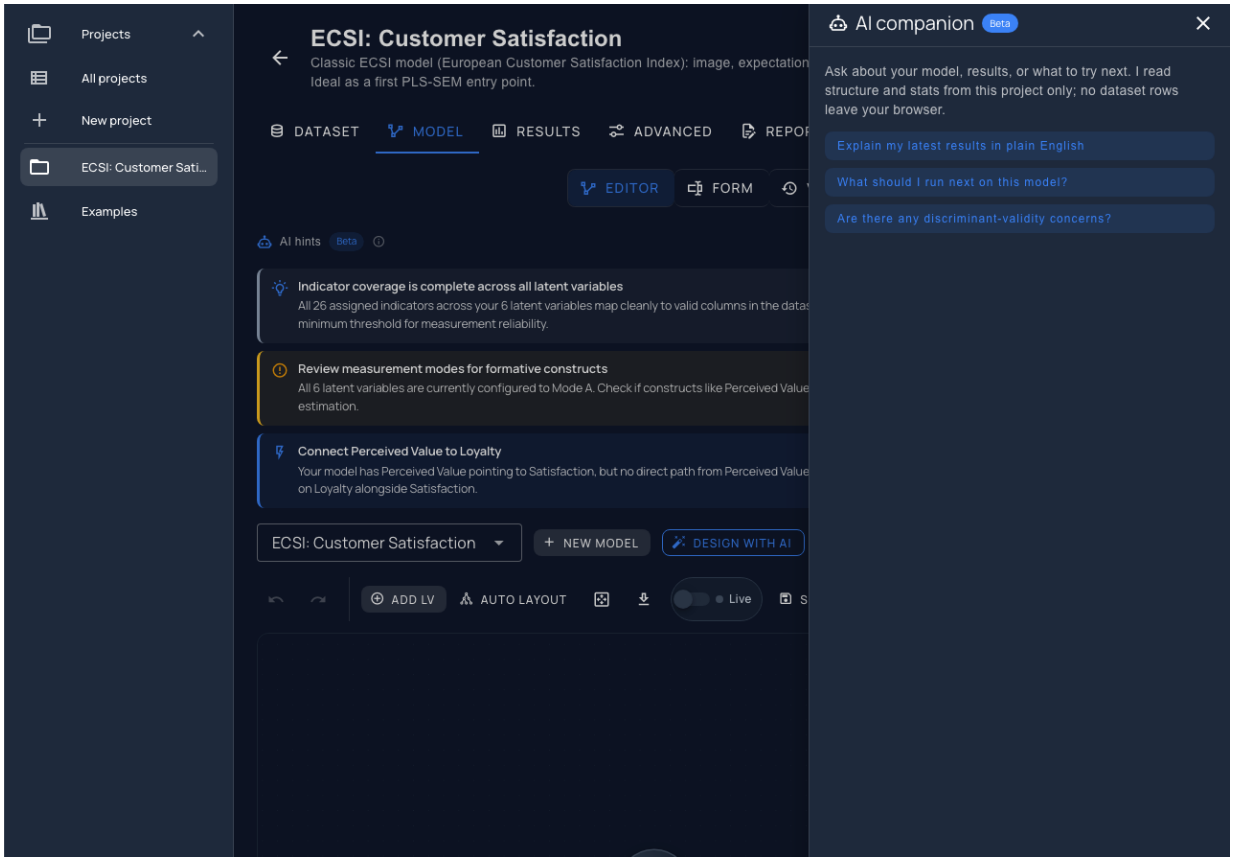

Figure 34 The AI Companion drawer, opened over the Results tab. Header shows the title and a beta chip, the suggestion tiles for first-time users offer three ready-made questions, and the composer at the bottom carries a live quota counter.

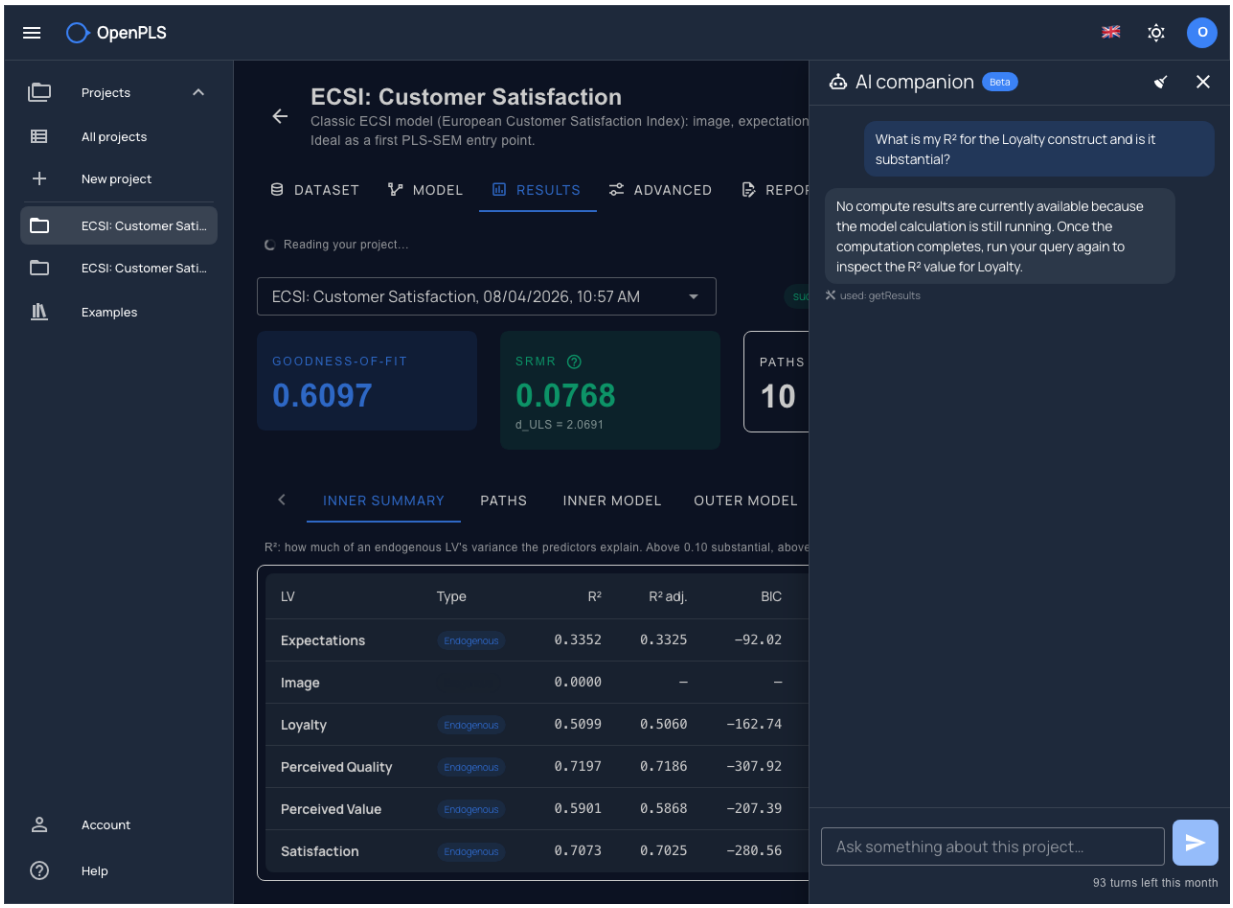

Figure 35 A chat turn with the used-tools line visible below the answer. The Companion called `getResults` to pull the actual R^2, β, and p-values before answering.

never the actual row values

- **getResults** - R^2 per LV, path coefficients with p-values, HTMT matrices, reliability, Fornell-Larcker verdicts, SRMR, GoF

The used-tools line beneath each answer tells you which of these were called. Nothing else leaves the project.

Citations. When a claim in the answer draws on the methodology literature, the Companion writes a bracketed role slug (e.g. `[henseler-2015-htmt]`). The frontend renders these as clickable pill chips showing the author and year (e.g. **Henseler+ 2015**). Hovering reveals the full title; clicking opens the DOI or publisher page in a new tab.

The paper library behind these chips is a curated set of ~100 open- access PLS-SEM references (see *The paper library* below).

History + persistence. Chat turns are saved per user in the account settings storage. Reload the page, open the drawer, your conversation is still there. The **Clear conversation** icon in the header wipes it.

Quota. 100 chat turns per user per month during beta. The counter at the bottom of the composer shows what's left. Model-design and report-drafting turns share the same bucket.

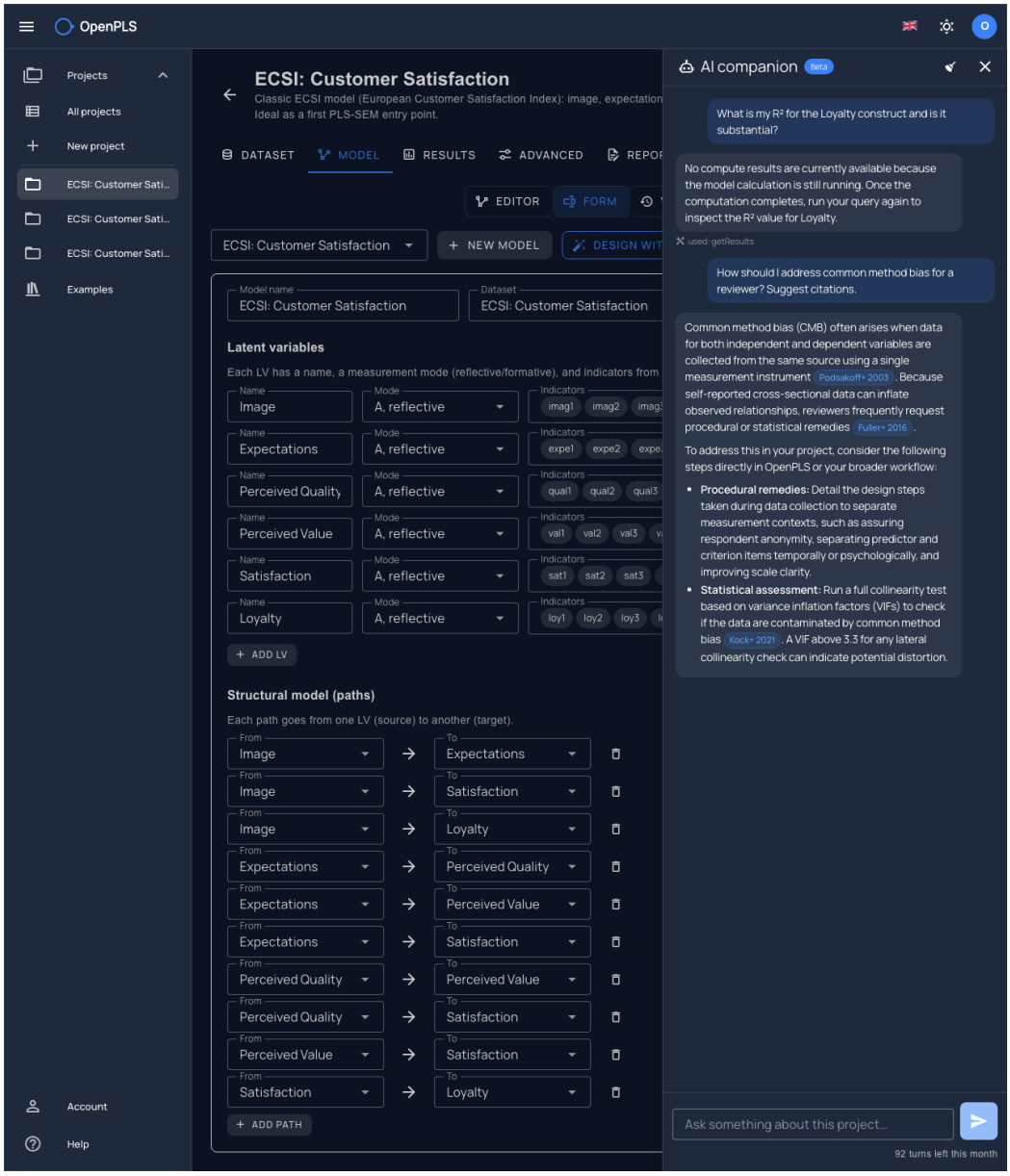

Figure 36 Chip citations rendered inline in a chat answer. Each pill links to the original paper's DOI page.

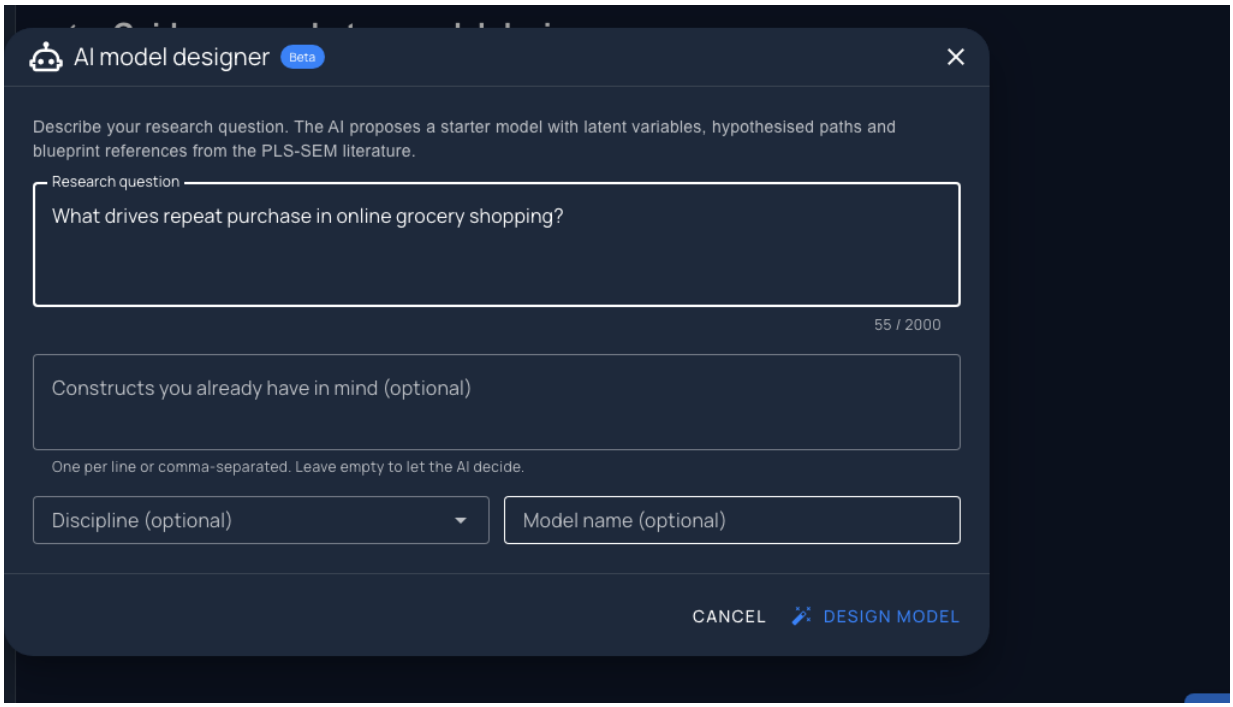

Figure 37 The AI model designer dialog. Type a research question (required), list constructs already in mind (optional), pick a discipline. Click **Submit** to receive a structured proposal.

11.4 Designing a model from a research question

If your model doesn't exist yet - a fresh project, no LVs drawn - open the **Editor** and click **Design with AI**. The dialog asks for your research question, and optionally the constructs you already have in mind and a discipline hint.

You get back a structured proposal:

- **3–6 latent variables** with their mode (A = reflective, B = formative), a short justification, and 3–5 indicator hints per LV (phrases describing what items would measure this construct - not full survey wording).
- **5–12 structural paths** each with a plain-English hypothesis and a short literature note.
- **2–3 blueprint models** from published PLS-SEM literature that share the structure.
- **Notes** on caveats or alternative specifications to consider.

The proposal is **not** written into your project until you click **Apply**. Preview it, refine your prompt if needed (**Refine** regenerates), or discard and start over.

Once applied, the model appears in the Editor as a normal draggable canvas - you can rename LVs, add paths, redistribute indicators. The proposal is a starting point, not a final artifact.

11.5 Designing a questionnaire

Once you have a model (LVs with modes assigned), OpenPLS can generate a full item battery: 4–6 survey items per LV with wording, Likert scale, reverse-coding flags, and citations to validated scales when the LV maps to a well-known construct in the literature (Trust, Satisfaction, Loyalty,

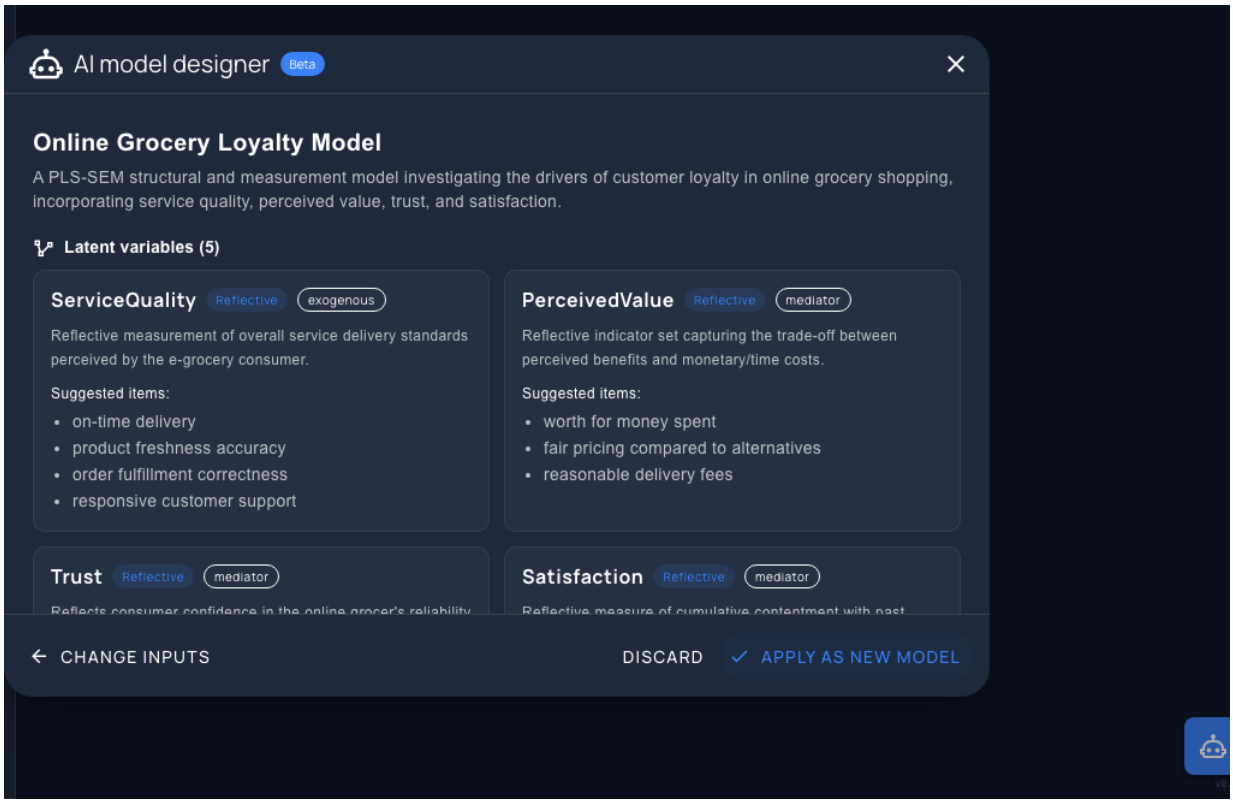

Figure 38 The proposal card. Each LV shows mode, indicator hints, and justification; paths list source → target with hypothesis. **Apply** writes the model into your project as a new version.

TAM's Perceived Usefulness, and so on).

Open the **Model** tab, then the **Form** sub-view. If any LV lacks items you'll see a **Design items with AI** button. Click it to open the questionnaire designer dialog.

The result is a table of items per LV: item ID, wording, reverse-coded flag, source citation. Skim, tweak the wording, remove or add items, then apply - the items land as indicators on each LV.

From here you have two options for actually collecting data: build a questionnaire that respondents fill out on a public URL (see the next chapter, *Public surveys*) or take the item battery to an external survey tool and import the responses as a CSV.

11.6 Method-Selection Assistant

Once you have compute results, the Advanced tab shows a **Method-Selection Assistant** panel at the top. Click to expand.

The assistant looks at your project state - sample size, grouping candidates in the dataset, R^2 per endogenous LV, HTMT violations, reliability thresholds - and returns a ranked shortlist of the advanced methods that make sense as your next step:

- **Bootstrap** (High) whenever you have a compute result but no confidence intervals yet.
- **MGA + MICOM** (High) when the dataset has a categorical column with 2–5 levels and enough per-group sample size.

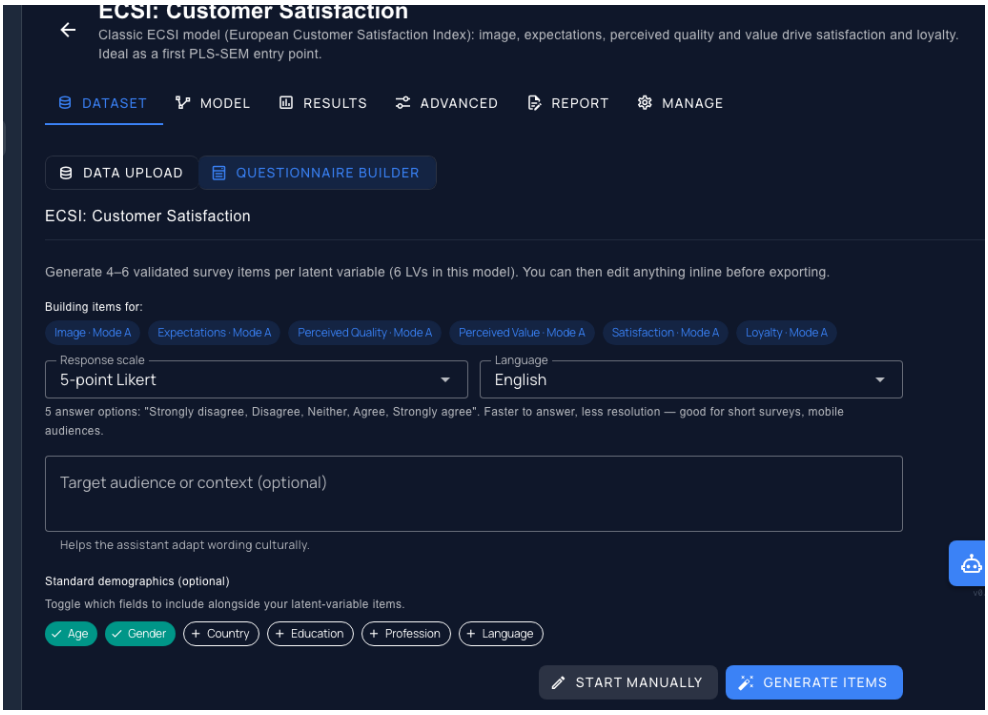

Figure 39 The AI questionnaire designer dialog. Pick a response scale (5- or 7-point Likert), audience language, and click **Submit**. The dialog shows which LVs it will build items for; ones that already have items get skipped unless you tick **Overwrite existing**.

- **IPMA** (High) when a target LV has $R^2 \geq 0.33$ - enough substantive variance for the antecedent ranking to be actionable.
- **PLSpredict** (Suggested) when at least one endogenous LV has $R^2 \geq 0.19$ and $n \geq 100$.
- **FIMIX-PLS** (Suggested) for $n \geq 200$ as a heterogeneity check.
- **HOC** (Suggested) when two LVs have HTMT close to 1 - they may be first-order facets of the same second-order construct.
- **PLSc** (Optional) when all LVs are reflective - a common-factor robustness check.
- **Copula / CMB / Nomological / CVPAT** (Optional) as standard reviewer-preemption diagnostics.

Each suggestion carries the reason grounded in your actual numbers (e.g. “Target LV Loyalty has $R^2 = 0.42$ - IPMA identifies which antecedents matter most for practical action”). The **Open in …** button (labeled with the specific method name) routes to the matching Advanced sub-tab so you can configure and run.

The panel is fetch-on-click; nothing runs unless you open it. Click **Regenerate suggestions** to re-read your project state.

11.7 Report drafting

A new tab - **Report** - sits between Advanced and Manage. Its four sub-buttons produce publication-ready text drawn from your actual data.

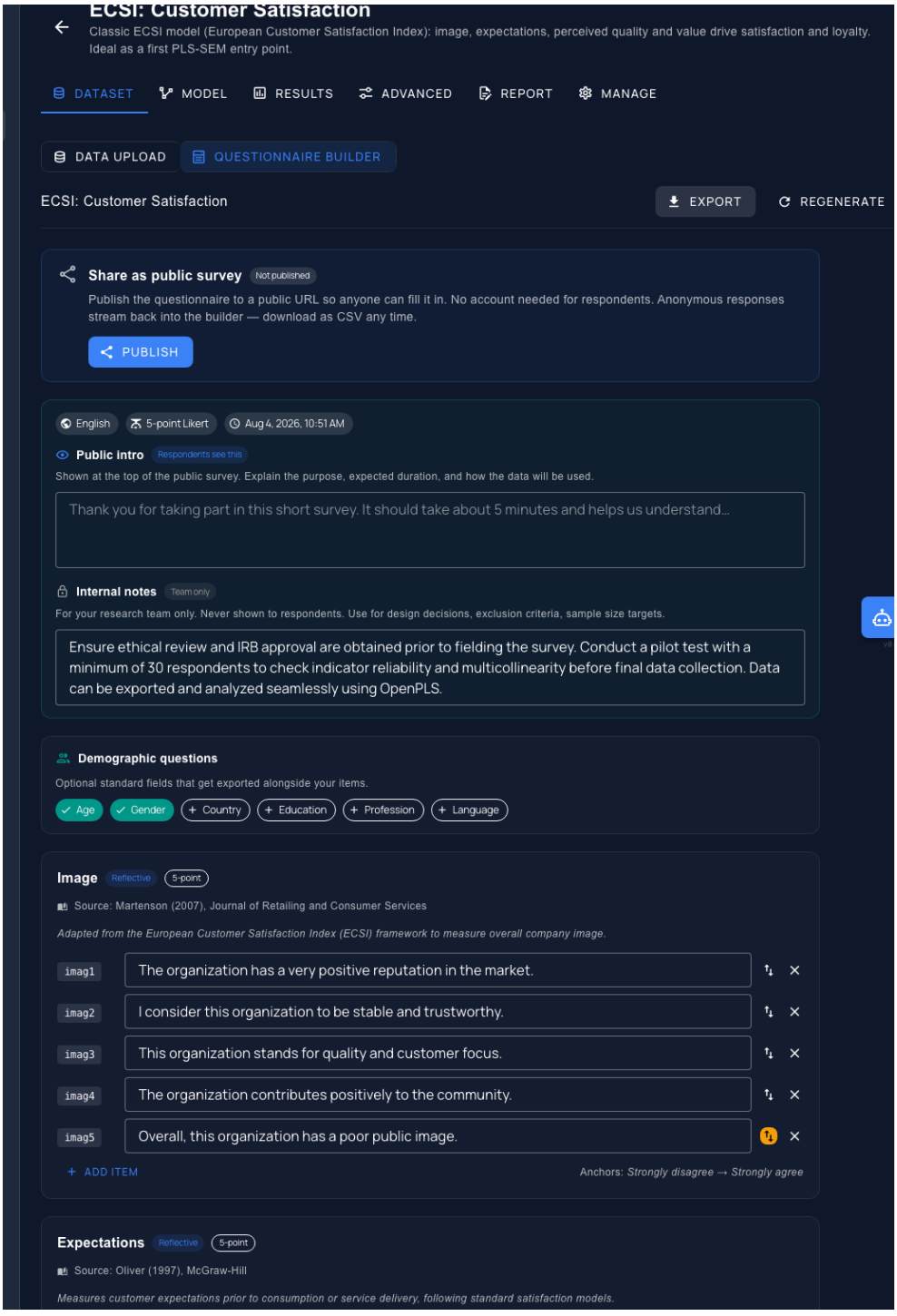

Figure 40 The generated questionnaire card. Each LV block lists items with wording + reverse chip + citation. Edit in place, then **Apply** to write onto your model.

Figure 41 The Method-Selection Assistant expanded. Priority chips (High / Suggested / Optional), a one-sentence rationale per method, a risk note, and an **“Open in …”** button that navigates to the corresponding Advanced sub-tab.

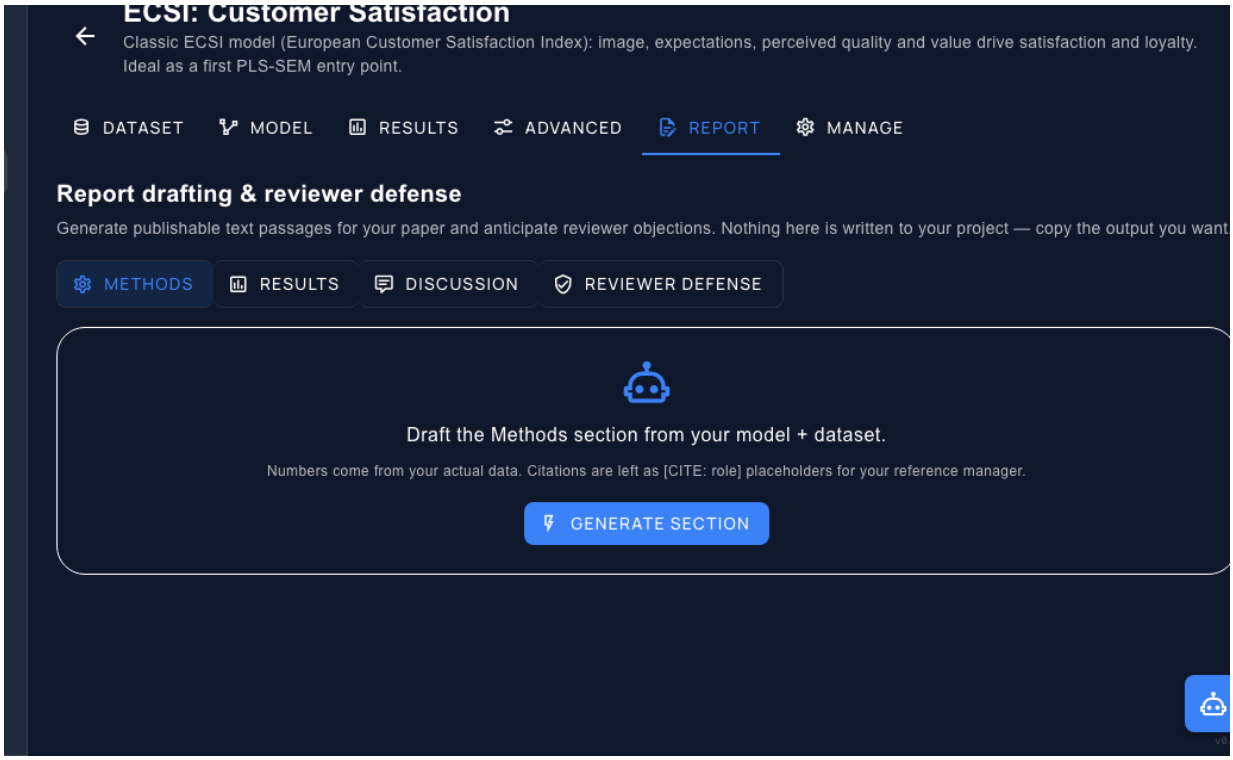

Figure 42 The Report tab overview. Four sub-buttons - Methods, Results, Discussion, Reviewer defense - and a large card that opens the corresponding drafter.

11.7.1 Methods / Results / Discussion

Pick a section and click **Generate section**. You get:

- **Markdown** on the left, ready to copy into a draft or blog post. Uses Unicode stats notation (R^2, β, χ^2, f^2) - no LaTeX inside the Markdown output.
- **LaTeX** on the right, ready for the paper's source. Uses `$R^2$`, `\beta`, `\cite{role}` conventions for a proper journal template.
- **Citation slots** on the far right: one row per bracketed role the section references, each linked to the DOI or publisher page.
- **Notes** - a one- or two-sentence caveat the drafter thinks the human author should know (e.g. "your SRMR of 0.11 is above the 0.08 threshold - consider commenting on fit").

The drafter follows Hair/Ringle/Sarstedt conventions:

- **Methods** reports sample, measurement model, estimation choices, reliability + validity criteria, model-fit indicators, and previews advanced procedures.
- **Results** reports measurement-model quality (α, ρc, AVE), discriminant validity (HTMT with any violations named), path coefficients with significance, R^2 per endogenous LV with the Chin 1998 verbal descriptor, and SRMR.
- **Discussion** interprets significant paths substantively, addresses non-significant ones, spells out practical implications for the target LV with the highest R^2, lists limitations, and proposes two concrete extensions.

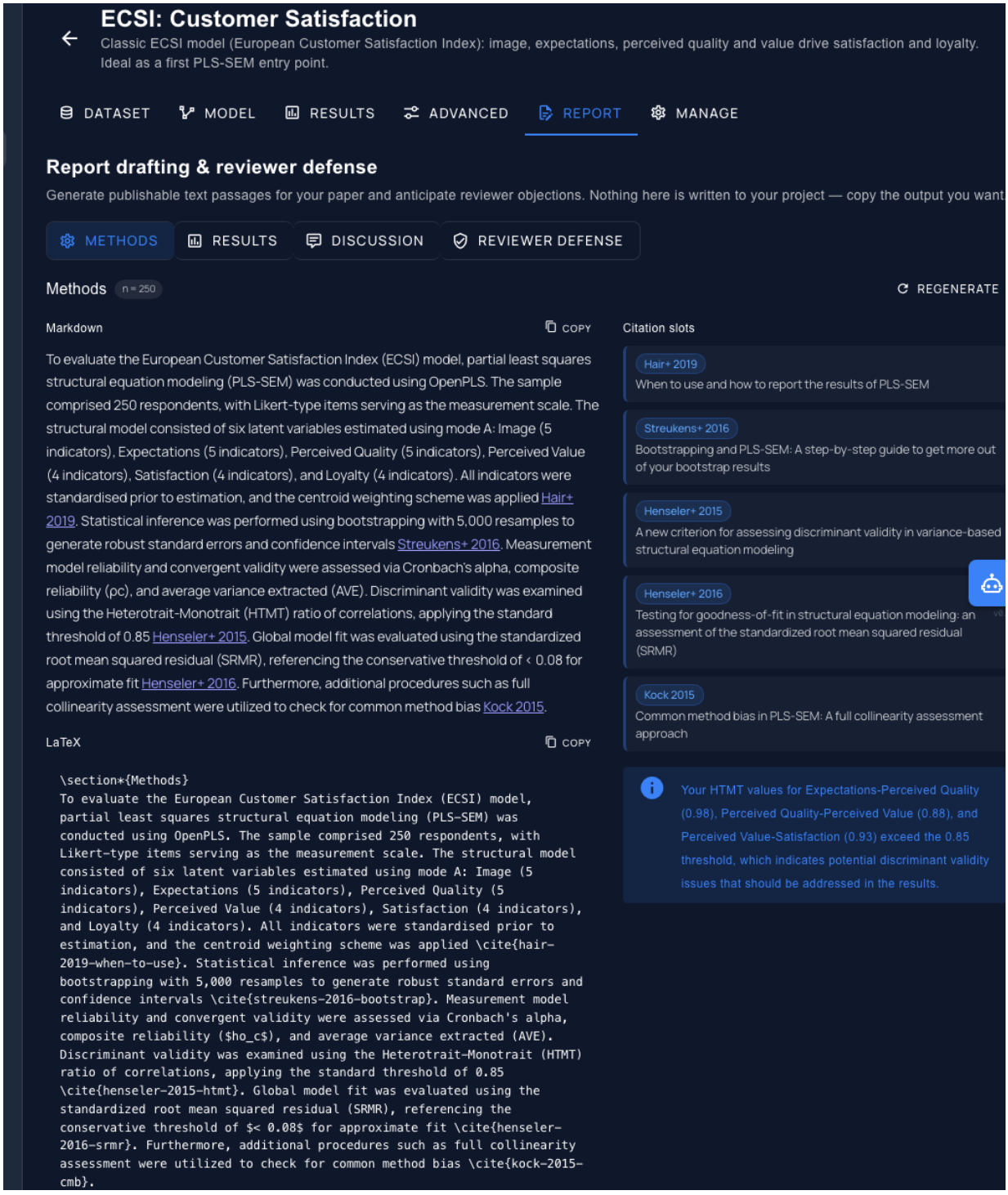

Figure 43 The Methods section generated. Markdown pane on the left, LaTeX pane below, citations rail on the right. Every number in the text traces back to the project.

Copy either pane with the copy-icon button in its header. Regenerate with the **Regenerate** button in the section header - the drafter re-reads the latest results, useful after a fresh bootstrap or a model tweak.

11.7.2 Reviewer defense

The fourth sub-button - **Reviewer defense** - is the drafter's adversarial mode. Click **Generate reviewer defense**.

You get four to seven anticipated objections a peer reviewer is likely to raise, each with:

- **Concern** in one or two sentences using the exact numbers from your snapshot.
- **Reviewer persona** - a short characterization ("CB-SEM traditionalist", "Statistical purist", "Senior methods editor") so you know which framing to expect.
- **Severity** - Critical, Moderate, or Nitpick.
- **Defense** - the actual counter-argument, referencing your numbers and citing the paper library inline.
- **Evidence** - the specific snapshot values that back the defense.
- **Mitigation** - 0–2 sentences on what to still run before submitting, if anything.
- **Suggested citations** - clickable chips for the papers the defender leans on.

A short **Overall strategy** paragraph at the top of the card lists the two or three moves that anchor your revision letter - usually "lead with X, concede Y as a limitation, commit to Z in a follow-up".

> The defender is a simulator, not an oracle. Real reviewers will bring context you can't preview - theoretical objections tied to your specific literature, methodological quirks of your target journal, personal hobby horses. Use the output as a rehearsal, not a guarantee.

11.8 The paper library

Every AI surface that produces citations draws from the same curated library - around 100 open-access PLS-SEM papers spanning methodology, critique, applications, and the modeling classics (TAM, UTAUT, TPB, SERVQUAL, ACSI, JD-R). Each entry is a title, author list, year, DOI, and the abstract as it appears on the publisher page. No full text - only the metadata reviewers need to find the paper themselves.

The library is stored in the same database as your projects and queried via **semantic vector search**: when you ask a question, the assistant embeds your question with a multilingual embedding model, finds the five closest paper abstracts by cosine similarity, and gives those to the LLM as reference context. The model then cites using the bracketed role slugs (e.g. `[henseler-2015-htmt]`), which the frontend replaces with pill chips.

If a chip appears grey/flat (not clickable), it means the assistant cited a role that isn't in the library

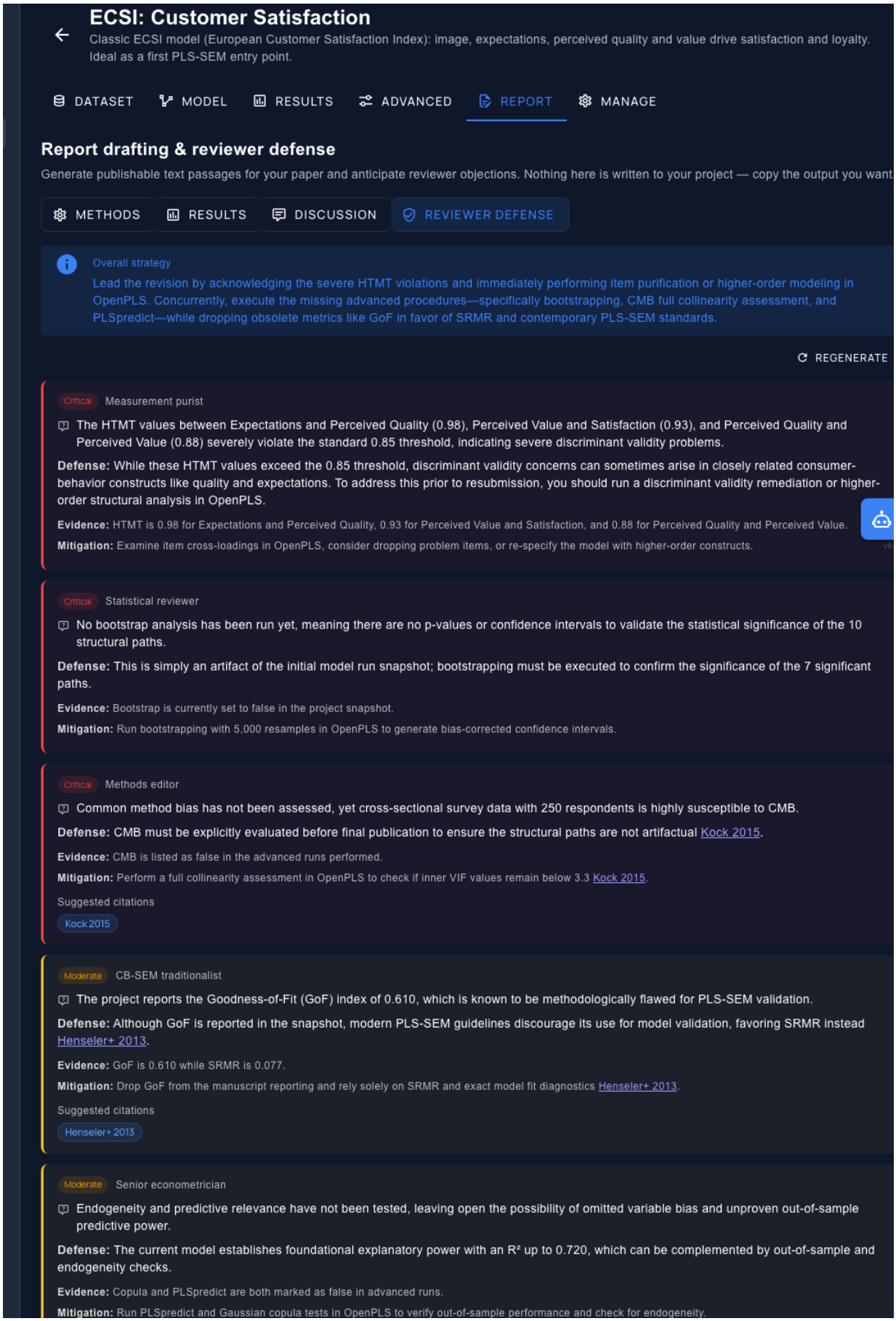

Figure 44 A reviewer-defense card. Severity chip (Critical / Moderate / Nitpick), reviewer persona, concern, defense, evidence, mitigation, and suggested citation chips.

yet - either an older paper outside our curation window or, occasionally, a hallucination. Grey chips are a signal to double-check the reference manually.

11.9 Quota and privacy

Quota. 100 turns per user per month during beta. Applies separately to interpretive turns (chat, model designer, report drafters) and to hints (100 hint fetches per month). Turns don't carry over.

Privacy. The Companion reads your project structure and aggregates (row counts, column stats, R^2, path coefficients), never the actual survey rows. All processing runs under contractual data-protection terms with the model provider, hosted in the EU, and your data is never used to train future models.

11.10 Turning it off

Account → AI Companion → toggle off disables every AI surface across the app: the drawer disappears, the hints strip collapses, the designer + report tabs still exist but their generate buttons become inactive with a "not enabled" hint.

The stored conversation is preserved (nothing is deleted); re-enable later and your chat history is exactly where you left it.

12 Public surveys

OpenPLS can host your questionnaire on a public URL, collect anonymous responses, and stitch them straight back into your project as a dataset. You don't need a third-party survey tool for the normal PLS-SEM workflow.

The typical loop:

1. **Build** the item battery in the Model tab (by hand, or with the AI questionnaire designer - see the previous chapter).
2. **Publish** the questionnaire to a public `/q/{slug}` URL.
3. **Share** the URL with respondents. New answers land in the builder in near real time.
4. **Import** the accumulated responses as a dataset with one click.
5. **Compute** the model on your fresh dataset.

This chapter walks each step.

12.1 Building the questionnaire

Open a project's **Model** tab, then the **Form** sub-view. If your LVs have no indicators yet, the panel starts empty; use the AI questionnaire designer (see chapter 12) or add items by hand. Each item is one column in the future response CSV.

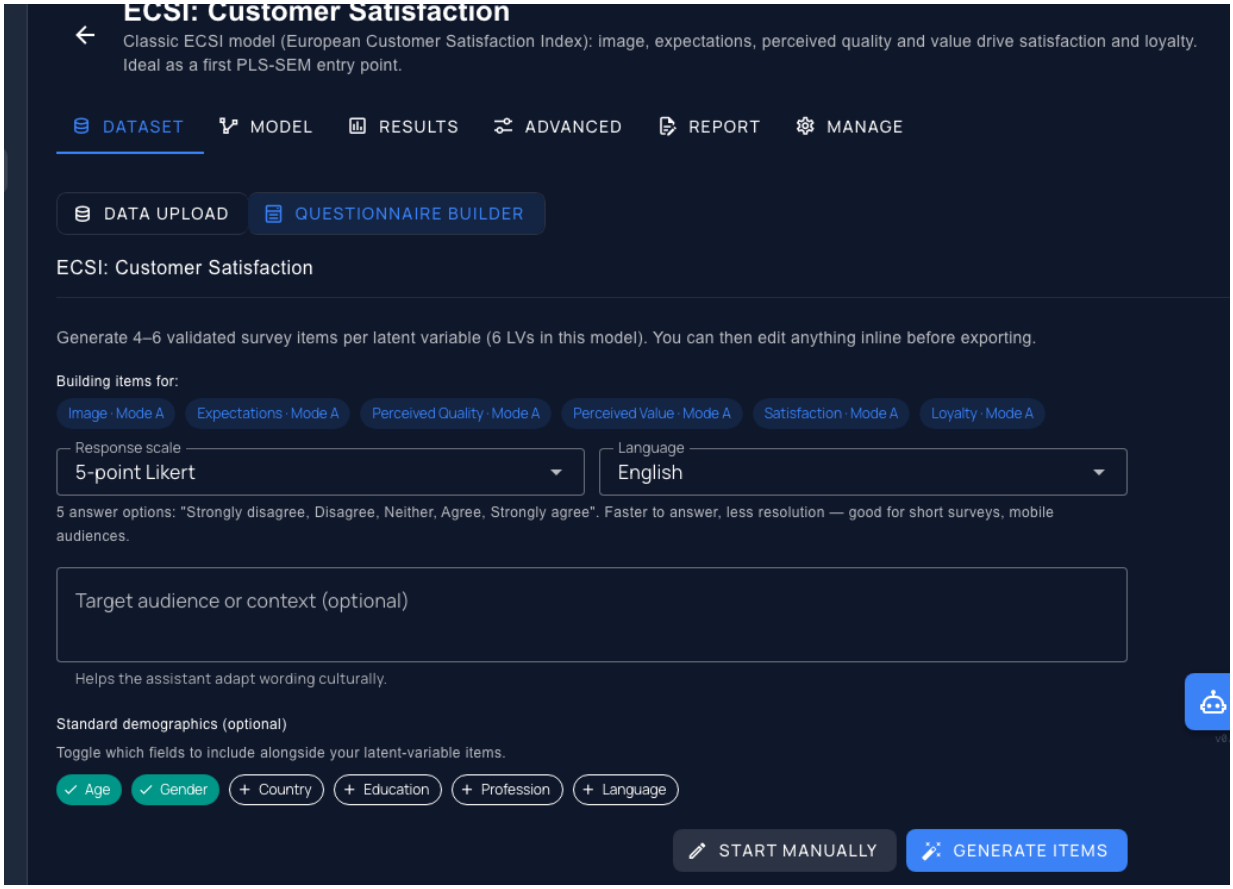

Figure 45 The questionnaire builder panel. Left: LVs with their items (drag to reorder, click × to remove, click a wording to edit). Right: the publish card with response scale, language, audience, and demographics toggles.

Per-item fields:

- **Wording** - the sentence respondents see. Reverse-coded items get a `R` chip.
- **Item ID** - auto-generated (e.g. `TRUST1`, `SAT2`), used as the column name in the exported dataset.
- **Reverse coded** - flip the toggle if the item's polarity is inverted (respondents mentally negate). Reverse-coding is applied automatically at import time.
- **LV binding** - inherited from the parent LV block; drag items between blocks to reassign.

Global fields (right column):

- **Response scale** - 5-point or 7-point Likert. Applies to all Likert-type items; open-ended and demographic items keep their own type.
- **Audience language** - the language shown to respondents (English, German, Spanish, French, Italian, Japanese, Portuguese, Chinese). Item wording stays in the language you wrote it; only the survey chrome (buttons, progress, thank-you page) translates.
- **Demographics** - pick zero or more from the six built-in optional fields (Age, Gender, Country, Education, Language, Profession). Each becomes a column in the exported dataset.
- **Public intro** - a paragraph respondents see above the first question. Split from your internal notes (which never appear on the public page).

12.2 Publishing

The **Publish** section of the panel toggles the survey live. Click **Publish**: OpenPLS mints a random slug, writes a public snapshot of the questionnaire to the responses collection, and hands you back the URL:

```
https://cloud.openpls.app/q/f3a8k29b
```

Anyone with the URL can respond. No account is needed - anonymity is the default. The public page shows only the public intro and the items themselves, formatted responsively for phones and desktops.

Preview opens the survey in a new tab so you can walk through it as a respondent (your responses in preview mode are not counted).

12.2.1 Version snapshots

Once responses start coming in, changing the questionnaire creates a data-integrity dilemma: if you rename an item or change its wording mid-way, the accumulated responses become inconsistent. OpenPLS handles this by asking you to pick a path when you save changes to a survey with existing responses:

- **Publish as new version** (recommended). Retires the current URL, mints a fresh slug (`/q/…`), and moves the counter to zero. Old responses stay archived under `v1` and remain accessible for download.

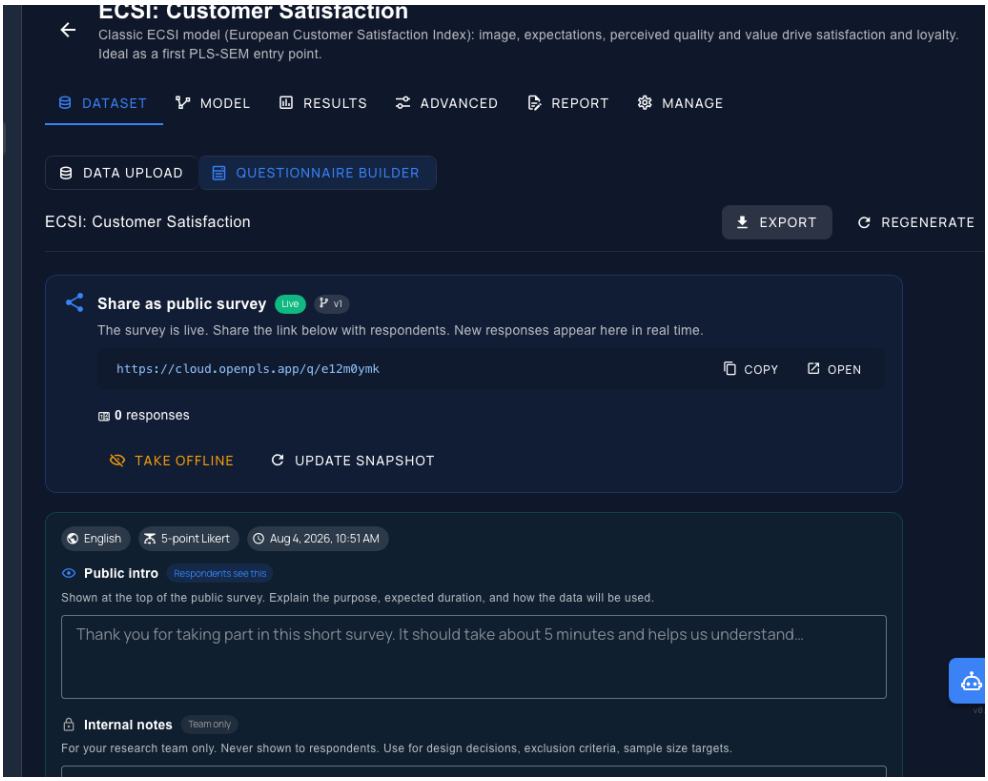

Figure 46 A published survey card. The URL is shown with copy-to-clipboard, live response counter, three actions (Preview / Unpublish / Publish as new version), and the version chip.

- **Download responses, then save in place.** Same URL, but you download a CSV archive of the pre-change batch first.
- **Save in place.** Same URL; new and old responses mix. Only use this for typo fixes.

Retired slugs stay online, but stop accepting new responses - respondents see a “no longer accepting responses” message. The publish card lists all prior versions with their final response count so nothing is lost.

12.3 The public respondent view

Respondents see a clean, single-column form with a progress bar. Items are grouped by LV so cognitive load stays manageable; the LV name itself is hidden (respondents don’t need the

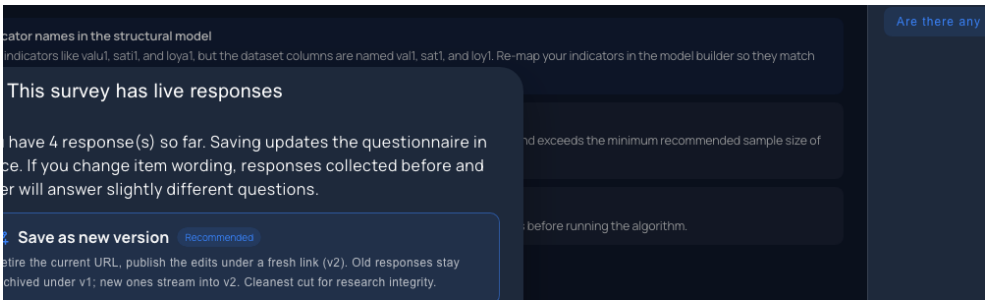

Figure 47 The save-warn dialog when a published survey has responses. Three paths, each with the concrete consequence spelled out.

theoretical label).

Design decisions worth knowing:

- **Language switcher.** Respondents can flip through the eight supported languages independently of the audience language you set - useful for international deployments.
- **One-response-per-browser guard.** Once submitted, a small flag in the browser's local storage prevents accidental resubmission from the same device. Not tamper-proof (any determined respondent can bypass it by clearing storage or using a different browser), but stops honest double-submits.
- **Thank-you page.** After Submit, respondents see a short acknowledgment. No account creation prompt, no marketing.
- **Mobile-first.** The form works down to ~320 px viewports; item cards stack, the language switcher becomes a compact menu.

12.4 Watching responses arrive

The publish card in the builder shows the current response count in near-real time (updates within a few seconds of each submission). Below the counter, a small **Recent** section lists the last five submissions with their timestamp and language.

12.5 Importing responses as a dataset

When you have enough responses to compute the model - for a typical PLS-SEM study, at least 100 for exploratory work; 200+ for publication-quality inference - click **Import as dataset** on the publish card.

OpenPLS creates a fresh dataset in your project. Column layout:

- **item_id columns** - one per survey item, using the `TRUST1`, `TRUST2`, … item IDs you defined. Cell value is the Likert response as integer (1..5 or 1..7). Reverse-coded items are reversed at import so downstream code doesn't need to remember.
- **demographic columns** - one per demographic field you enabled (`age`, `gender`, `country`, …). String or integer depending on the field type.
- **`submitted_at`** - ISO 8601 timestamp of when the response was submitted, useful for temporal filters or wave analysis.
- **`language`** - the language the respondent used, useful for MICOM/MGA cross-language comparisons.

Open the Data tab; the new dataset is Ready. Compute the model as usual. If you publish a new version later, importing again writes a **separate** dataset - the old one is preserved so you can compare estimates before and after the questionnaire change.

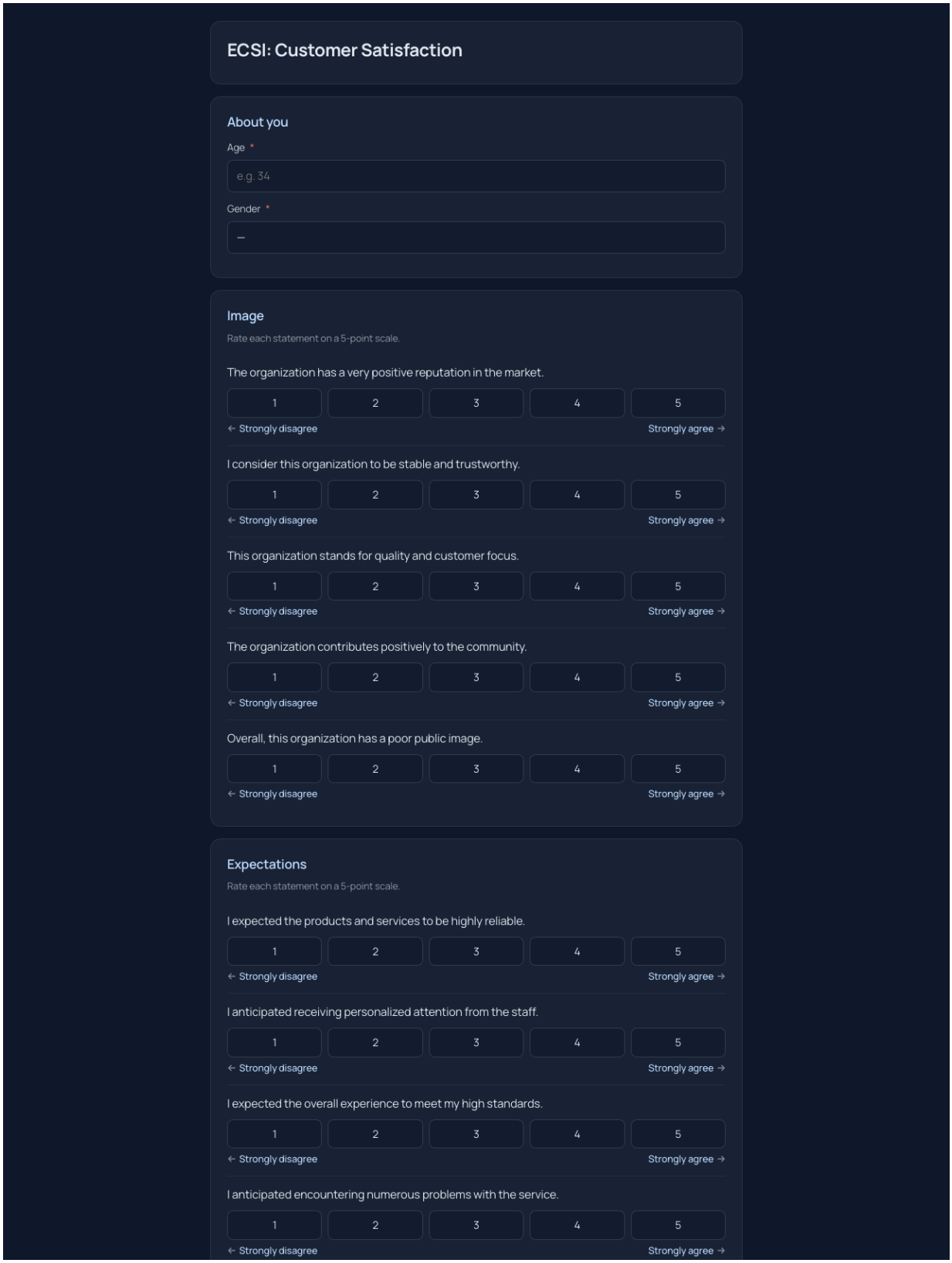

Figure 48 The public survey page rendered on desktop. Language switcher top-right, progress bar, item cards with radio buttons, Previous / Next navigation, and a final Submit action.

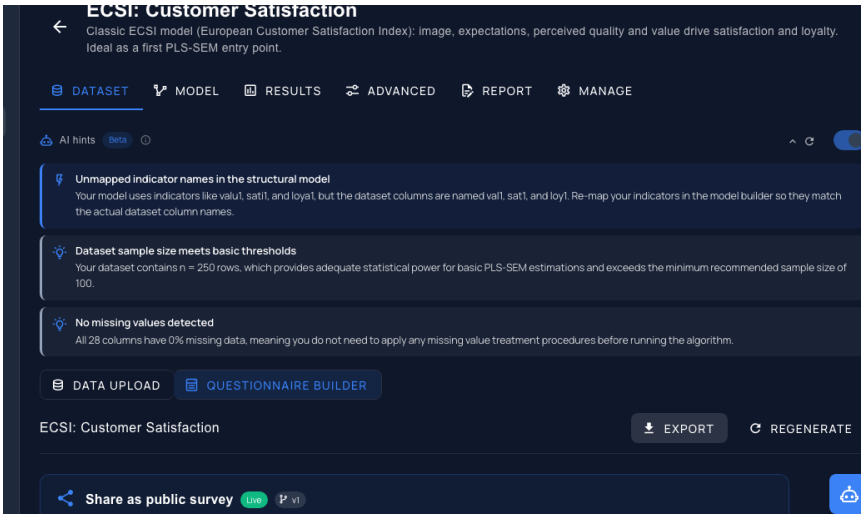

Figure 49 The response counter and recent list. Counter updates live; the recent list shows submitted_at plus language flag.

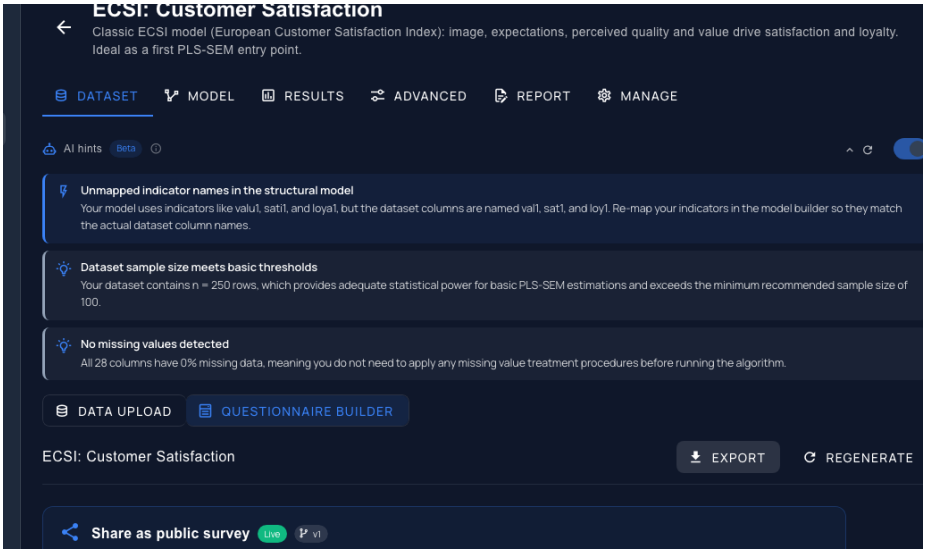

Figure 50 The import-as-dataset button on the publish card. Confirmation dialog shows: item columns, demographic columns, sample size, and version. Click **Create dataset** to write.

12.6 Downloading a raw CSV

If you'd rather bring the responses into an external tool (R, Stata, SPSS, Python), click **Download responses (CSV)** on the publish card. The CSV has the same columns as the imported dataset. Each row is one submission; empty cells mean the respondent skipped that optional field.

Downloads count only completed submissions. Partial responses (respondent closed the tab before Submit) don't reach the database.

12.7 Unpublishing

Unpublish takes the URL offline. Existing responses stay accessible in the builder and can still be downloaded or imported; new visitors see a "not accepting responses" page. The URL is not released - clicking Publish again on the same questionnaire re-uses the same slug where possible.

Take care: if you publish → collect → unpublish → change the questionnaire → republish → collect, you'll have mixed responses tied to the same URL slug. The save-warn dialog above prevents this in the normal edit flow, but unpublish followed by edit followed by republish is an escape hatch that bypasses it. Prefer **Publish as new version** whenever the questionnaire meaningfully changes.

12.8 Limits and conventions

- **Response limit per survey**: no hard cap during beta. Realistic soft ceiling is a few thousand per survey per project before the live counter noticeably lags. For 10 000+ responses, split into multiple surveys or use an external panel provider.
- **Response retention**: indefinite while the parent project exists. Deleting the project cascades the responses.
- **Anonymity**: only what respondents enter is stored. IP address, User-Agent, and any other identifying header are not persisted. If you want identifiable responses (e.g. for a panel study), add a respondent-ID field yourself as a survey item.

13 Appendix

13.1 Glossary

AVE: Average Variance Extracted. Mean of squared indicator loadings for a reflective LV. Above 0.5 = LV explains more variance in its indicators than error does.

BIC: Bayesian Information Criterion. Model-fit index that penalises additional parameters; lower is better; only comparable across nested specifications.

Bootstrap: resampling with replacement to derive standard errors, p-values, and confidence intervals for statistics that lack an analytical distribution.

Composite reliability (ρ_c, rho_c): the modern default for reliability in PLS-SEM. Above 0.7 acceptable, above 0.95 suggests redundant indicators.

CMB: Common Method Bias. Systematic error induced by the measurement instrument rather than the constructs being measured. Diagnosed via Lindell-Whitney marker or Harman's single-factor test.

CVPAT: Cross-Validated Predictive Ability Test. Compares PLS's out-of-sample loss against a benchmark (indicator average or linear model).

Endogeneity: a predictor is correlated with the error term. Violates OLS assumptions; corrected in PLS via the Gaussian copula approach.

Endogenous LV: a construct with at least one incoming path. Contrast with exogenous.

Exogenous LV: a construct with only outgoing paths. Its variance is not explained by the model.

FIMIX-PLS: Finite-Mixture PLS. Discovers latent classes when you suspect unobserved heterogeneity.

Formative (Mode B), measurement model where the indicators cause the LV (income + education + occupation → SES).

Fornell-Larcker: classical discriminant-validity test: √AVE(A) > correlation(A, B) for every pair A, B.

GoF: Goodness-of-Fit. `√(mean(communality) × mean(R²))`. Overall model quality proxy.

HTMT: Heterotrait-Monotrait ratio. Modern discriminant-validity test. Above 0.85 (conservative) or 0.90 (liberal) suggests the constructs are too similar.

HOC: Higher-Order Construct. A construct composed of multiple first-order LVs (e.g. Brand Experience = Sensory + Affective + Intellectual + Behavioral).

Indicator / Manifest Variable (MV): an observed dataset column that measures a latent variable.

Inner model: the structural paths between LVs. Contrast with outer model.

IPMA: Importance-Performance Map Analysis. Identifies which predecessors of a target LV combine high importance with low performance (biggest improvement lever).

Latent variable (LV): an unobserved construct measured indirectly via indicators.

Loading: the correlation between an indicator and its LV in a reflective measurement model.

Measurement model: the arrows between LVs and their indicators. Reflective (Mode A) or Formative (Mode B).

MGA: Multi-Group Analysis. Test whether path coefficients differ across groups.

MICOM: Measurement Invariance of Composite Models. Prerequisite for MGA/moderation across groups.

Mode A: reflective measurement (LV → indicators).

Mode B: formative measurement (indicators → LV).

Moderation: X's effect on Y depends on a third variable Z. Tested via an interaction term X × Z.

Outer model: the arrows between LVs and indicators. Contrast with inner model.

Path coefficient (β): the standardised strength of an inner-model arrow. Interpret like an OLS regression beta.

PLSc: Consistent PLS. Corrects the attenuation bias when constructs are truly common-factor rather than composite.

PLSpredict: out-of-sample predictive assessment via k-fold cross-validation.

Q^2: Stone-Geisser cross-validated redundancy. Above 0 = predictive relevance.

Q^2_predict: the PLSpredict variant of Q^2 using k-fold CV.

Reflective (Mode A), the LV causes its indicators. Change the construct and all indicators change together.

R^2: variance in an endogenous LV explained by its predecessors. In-sample.

SRMR: Standardized Root Mean Square Residual. Below 0.08 good fit, below 0.10 acceptable.

Structural model: same as inner model.

t-statistic: path estimate divided by its bootstrap standard error. Above ~1.96 significant at $\alpha = 0.05$.

VIF: Variance Inflation Factor. Below 5 acceptable collinearity, below 3 stricter. Applied both to formative indicators (outer VIF) and to predictor LVs in each endogenous block (inner VIF).

13.2 Keyboard shortcuts

Editor

- Cmd+Z / Ctrl+Z, undo
- Cmd+Shift+Z / Ctrl+Y, redo
- Delete or Backspace on a selected LV, remove the LV and its paths
- Double-click on an edge, remove the path

Editor drag operations

- Drag from LV source handle (blue dot, right side) to LV target handle (green dot, left side), draw a new path
- Drag an indicator chip from the right-hand sidebar onto an LV circle, assign the indicator to that LV

App-wide

- Cmd+K / Ctrl+K, future: command palette (roadmap)

13.3 References

Bakker, A. B., & Demerouti, E. (2017). Job demands-resources theory: Taking stock and looking forward. *Journal of Occupational Health Psychology*, 22(3), 273-285.

Brakus, J. J., Schmitt, B. H., & Zarantonello, L. (2009). Brand experience: What is it? How is it measured? Does it affect loyalty? *Journal of Marketing*, 73(3), 52-68.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3), 319-340.

Dijkstra, T. K., & Henseler, J. (2015). Consistent partial least squares path modeling. *MIS Quarterly*, 39(2), 297-316.

Hair, J. F., Hult, G. T. M., Ringle, C. M., & Sarstedt, M. (2022). *A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM).* 3rd ed. Sage.

Hair, J. F., Sarstedt, M., Ringle, C. M., & Gudergan, S. P. (2018). *Advanced Issues in Partial Least Squares Structural Equation Modeling.* Sage.

Henseler, J. (2021). *Composite-Based Structural Equation Modeling: Analyzing Latent and Emergent Variables.* Guilford Press.

Henseler, J., & Chin, W. W. (2010). A comparison of approaches for the analysis of interaction effects between latent variables using partial least squares path modeling. *Structural Equation Modeling*, 17(1), 82-109.

Henseler, J., Ringle, C. M., & Sarstedt, M. (2015). A new criterion for assessing discriminant validity in variance-based structural equation modeling. *Journal of the Academy of Marketing Science*, 43(1), 115-135.

Hult, G. T. M., Hair, J. F., Proksch, D., Sarstedt, M., Pinkwart, A., & Ringle, C. M. (2018). Addressing endogeneity in international marketing applications of partial least squares structural equation modeling. *Journal of International Marketing*, 26(3), 1-21.

Lindell, M. K., & Whitney, D. J. (2001). Accounting for common method variance in cross-sectional research designs. *Journal of Applied Psychology*, 86(1), 114-121.

Park, S., & Gupta, S. (2012). Handling endogenous regressors by joint estimation using copulas. *Marketing Science*, 31(4), 567-586.

Russett, B. M. (1964). Inequality and Instability: The Relation of Land Tenure to Politics. *World Politics*, 16(3), 442-454.

Sarstedt, M., Hair, J. F., Cheah, J.-H., Becker, J.-M., & Ringle, C. M. (2019). How to specify, estimate, and validate higher-order constructs in PLS-SEM. *Australasian Marketing Journal*, 27(3), 197-211.

Schuberth, F., Rademaker, M. E., & Henseler, J. (2023). Assessing the overall fit of composite models estimated by partial least squares path modeling. *European Journal of Marketing*, 57(6), 1678-1702.

Shmueli, G., Ray, S., Estrada, J. M. V., & Chatla, S. B. (2016). The elephant in the room: Predictive performance of PLS models. *Journal of Business Research*, 69(10), 4552-4564.

Tenenhaus, M., Vinzi, V. E., Chatelin, Y.-M., & Lauro, C. (2005). PLS path modeling. *Computational Statistics & Data Analysis*, 48(1), 159-205.

Venkatesh, V., Thong, J. Y., & Xu, X. (2012). Consumer acceptance and use of information technology: Extending the unified theory of acceptance and use of technology. *MIS Quarterly*, 36(1), 157-178.