mail-block

Variable-data PDF imposition

One piece of artwork, a spreadsheet of records, and as many copies as will fit on a sheet — imposed, marked and ready for the press.

User manual · version 0.1

What mail-block does

mail-block takes one piece of artwork, fills it in from a spreadsheet, and lays as many copies as will fit onto a printing sheet. Address labels, membership cards, tickets, shelf edging — anything printed many-up where every copy says something different.

This manual walks through a whole job, then covers each part of the app in turn. It describes version 0.1.

What the words mean

Five terms are used throughout, and the app is much easier to follow once they are clear.

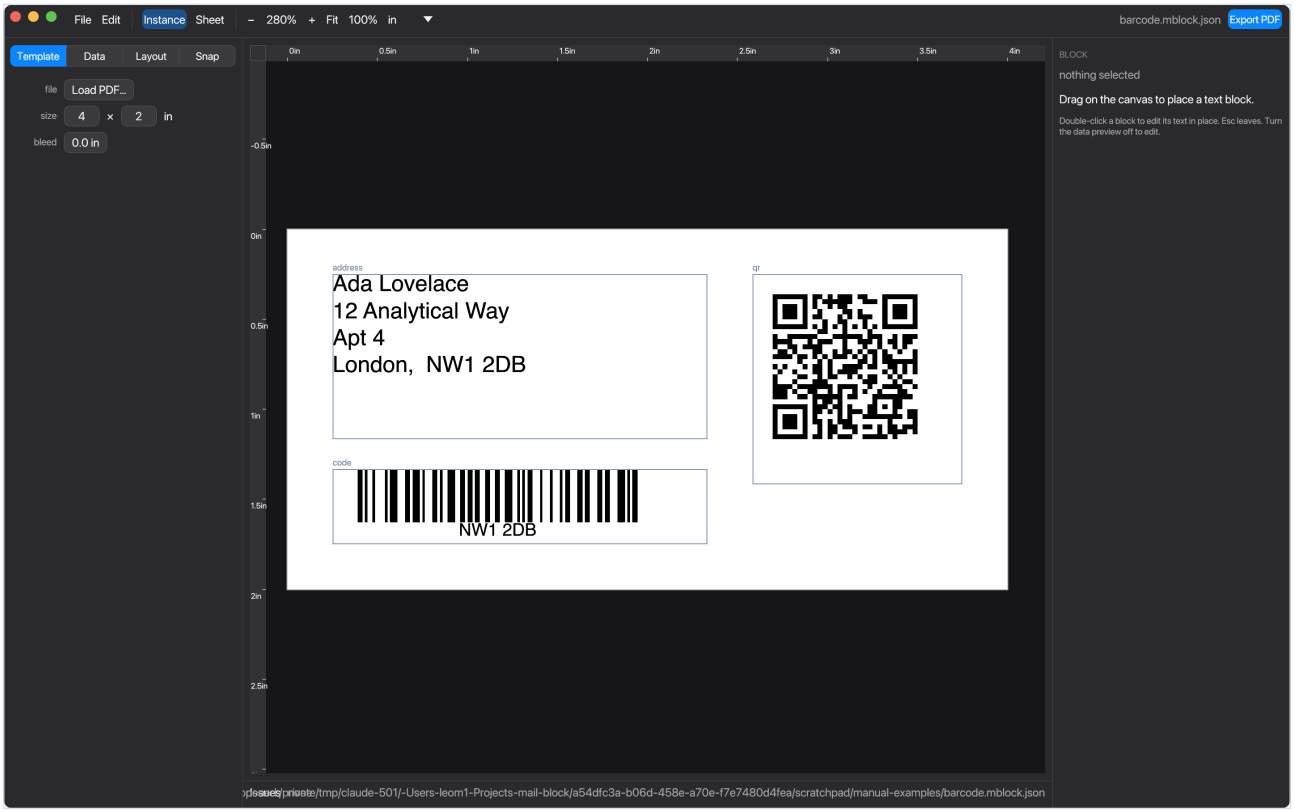
- **Template** — the artwork for one copy, as a PDF. One label, not a sheet of them.
- **Instance** — one copy on the sheet. Its size is the template's trim box.
- **Block** — a rectangle of text you place on the instance. Its text can contain placeholders, and it can be drawn as a barcode instead of as type.
- **Record** — one row of your CSV. Each record fills one instance.
- **Sheet** — the paper you print on, with as many instances imposed on it as fit.

The job, then, is: a template, a CSV, some blocks, and a sheet size. Everything else is detail.

Getting started

Open the disk image and drag **mail-block** to Applications. The first time you open it, macOS may ask you to confirm — it is not signed with an Apple developer certificate.

The app opens with an empty project: a 4 × 2 inch instance and one block. You can start there, or open a project you saved earlier. Projects are `.mblock.json` files and hold everything except the template and the CSV, which they reference by path.



The window: settings on the left, the instance in the middle, the selected block on the right.

The window has four parts:

- **The settings panel**, left, tabbed into **Template**, **Data**, **Layout** and **Snap**. Only the group you are working in is on screen.
- **The canvas**, middle, showing either the **Instance** — one copy, where you place blocks — or the **Sheet**, the imposition. The two are switched at the top.
- **The inspector**, right, for whichever block is selected. It has its own **Block** and **Barcode** tabs.
- **The issues bar**, along the bottom, which lists anything wrong with the job. Read it before you export.

A first job

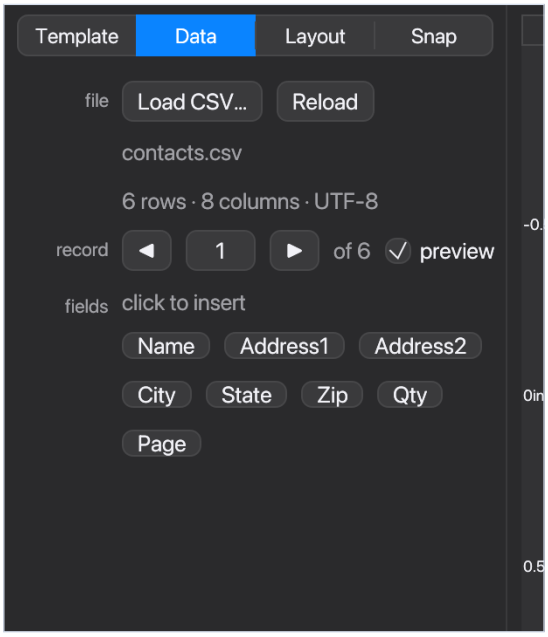
1. Load the template

On the **Template** tab, press **Load PDF…** and choose your artwork. The instance size comes from the page's trim box, falling back to the crop box and then the media box, so a PDF prepared for print is measured the way the printer measured it.

If you have no template yet, leave it empty and type an instance size instead. You will get a blank white instance to design against.

2. Load the data

On the **Data** tab, press **Load CSV…**. The file name appears under the button, with the number of rows, the number of columns and the encoding it detected.



The Data tab: the file it loaded, the record pager, and the columns as clickable fields.

The columns appear as chips. Clicking one inserts a placeholder into the block you are editing, which is quicker and safer than typing it.

The **record** pager below chooses which row the canvas shows. Turn **preview** off to see the placeholders themselves rather than the merged data.

3. Place a block

Drag a rectangle on the instance. Type into the inspector's **text** box, using `{{Column}}` wherever the data goes:

```
{{Name}}
{{Address1}}
{{Address2}}
{{City}}, {{State}} {{Zip}}
```

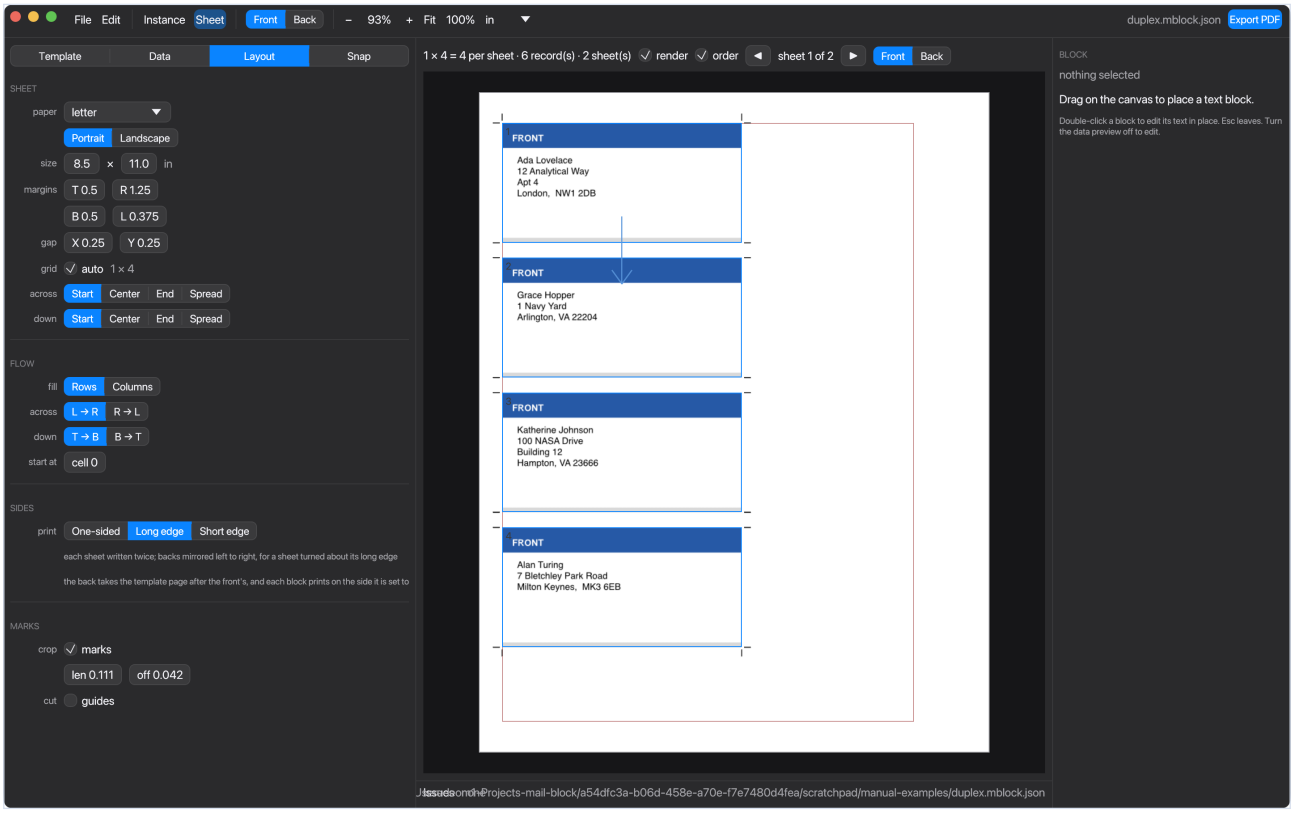
The inspector's Block tab: text, styling, geometry and type.

Two conveniences worth knowing from the start:

- `{{Address2|}}` gives a fallback for an empty cell — here, nothing. Lines that end up empty are removed, so a missing second address line does not leave a gap.
- Double-clicking a block edits its text **on the canvas**, in place, with the real fonts. Esc leaves. This only works with the data preview off, since merged text is not yours to edit.

4. Check the layout

On the **Layout** tab, set the sheet size, the margins around it, and the gap between instances. The **grid** row tells you what fits — `1 × 4` and so on — and the canvas switches to the **Sheet** view so you can see it.



The layout tab, with the sheet view showing what will print.

The sheet view draws the real imposition: the artwork as it will print, the cells over it, and the order they are filled in. **render** turns the artwork on and off; **order** turns the numbering and the flow arrow on and off.

5. Export

Press **Export PDF**, choose where to write it, and the job runs. The report tells you how many records went onto how many sheets.

Before you press it, read the issues bar. Errors stop the export; warnings do not, but they are worth a look — text that had to be shrunk to fit, a placeholder naming a column that is not in the data, a barcode payload that will not encode.

Blocks

Text and placeholders

`{{Column}}` is replaced by that column's value for the record being drawn. Whitespace inside the braces is ignored, so `{{ Name }}` works too. `\{{` prints a literal `{{`.

`{{Column|fallback}}` uses the fallback when the column is empty or missing. Leave the fallback out entirely — `{{Column|}}` — to substitute nothing at all.

Styling part of a line

Styling is written in the text as tags, so it survives being merged with data:

```
[b]{{Name}}[/]
[size=14][color=#B00]SALE[/][/]
[font=Georgia][i]Members only[/][/]
```

`[b]` bold, `[i]` italic, `[u]` underline, `[font=…]`, `[size=…]` (a number in points, or `+2` / `-1` relative to the block), `[color=…]` as `#RGB`, `#RRGGBB`, `gray(0.3)` or `cmyk(0,0.8,1,0)`. `[/]` closes the innermost tag.

You do not have to type them: select some text in the inspector and use the **B**, **I**, **U**, face and size controls, which write the tags for you.

Fitting

A block that cannot hold its text has to do something about it, and **overflow** says what:

- **Shrink** — reduce the size until it fits, down to the minimum you set. The usual choice for addresses, where one in five hundred is long.
- **Clip** — draw what fits and cut the rest off.
- **Overflow** — let it spill outside the block.
- **Error** — stop the export and name the record.

Anything shrunk or clipped is reported, with the record number, so you can look at the ones that needed it.

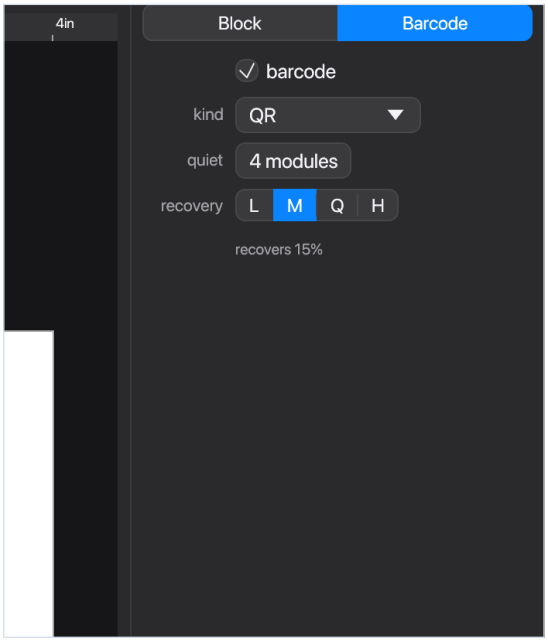
Position and alignment

geometry gives the block's exact rectangle in your display units; the handles on the canvas do the same thing by eye. Blocks snap to the instance edges, its centre and to each other — hold ■ to suspend that, or turn it off on the **Snap** tab.

align sets the text inside the block, horizontally and vertically. **rotate** turns the block about its top-left corner.

Barcodes

Any block can be a barcode instead of type. Select it, open the inspector's **Barcode** tab, and tick **barcode**. The block's text — merged with the record, the same as always — becomes the payload.



The Barcode tab. The symbol itself is drawn on the canvas as you change these.

kind chooses the symbology:

- **QR** — any text, and the only one that will hold a URL.
- **Code 128** — any printable ASCII, any length. The general-purpose linear code.
- **Code 39** — digits, capitals and `- . $ / + %`. Read by almost anything.
- **EAN-13** — twelve digits; the thirteenth, the check digit, is computed for you.
- **EAN-8** — seven digits, same idea.

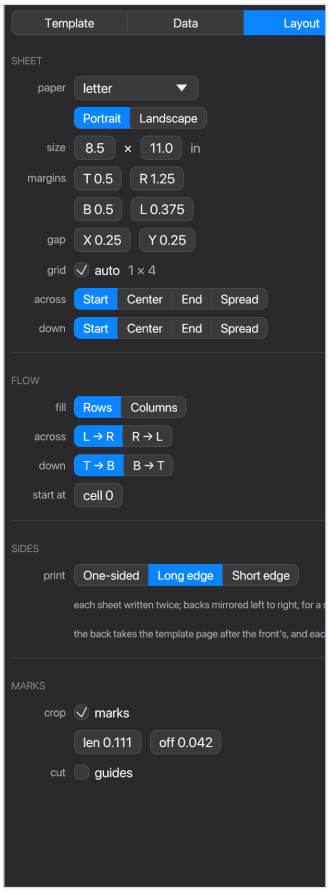
quiet is the clear margin around the symbol, in module widths. It is drawn inside the block, and it is not decoration: a symbol without it does not scan. Four modules is the standard for QR, ten for the linear codes.

recovery (QR only) is how much of the symbol can be lost and still read — L 7%, M 15%, Q 25%, H 30%. More recovery means less room for data, so a long URL at H may not fit.

print the payload (linear codes only) sets the payload under the bars in the block's own font, taking that height from the bars rather than growing past the block.

The symbol is scaled to fit the block without distortion — a QR stays square — and placed inside it by the block's own alignment. If a record's payload cannot be encoded, the reason appears under the controls and in the issues bar, naming the record, and the export stops rather than printing something that will not scan.

Layout



The Layout tab holds the sheet, the fill order, the sides and the marks.

The sheet

paper offers the common sizes, **size** is the actual measurement and accepts anything. **margins** are the unprintable or unusable border; **gap** is the space between instances, measured trim to trim.

How many fit

With **auto** ticked, the app fits as many as the sheet allows and tells you what it arrived at. Untick it to set the number across and down yourself; it starts from what was fitted, so nothing moves until you change a number. A grid that cannot fit is reported with the measurements, rather than silently dropping instances.

across and **down** place that grid in the usable area — start, centre, end, or spread, which distributes the leftover space between the instances and makes your gap a minimum rather than an exact figure.

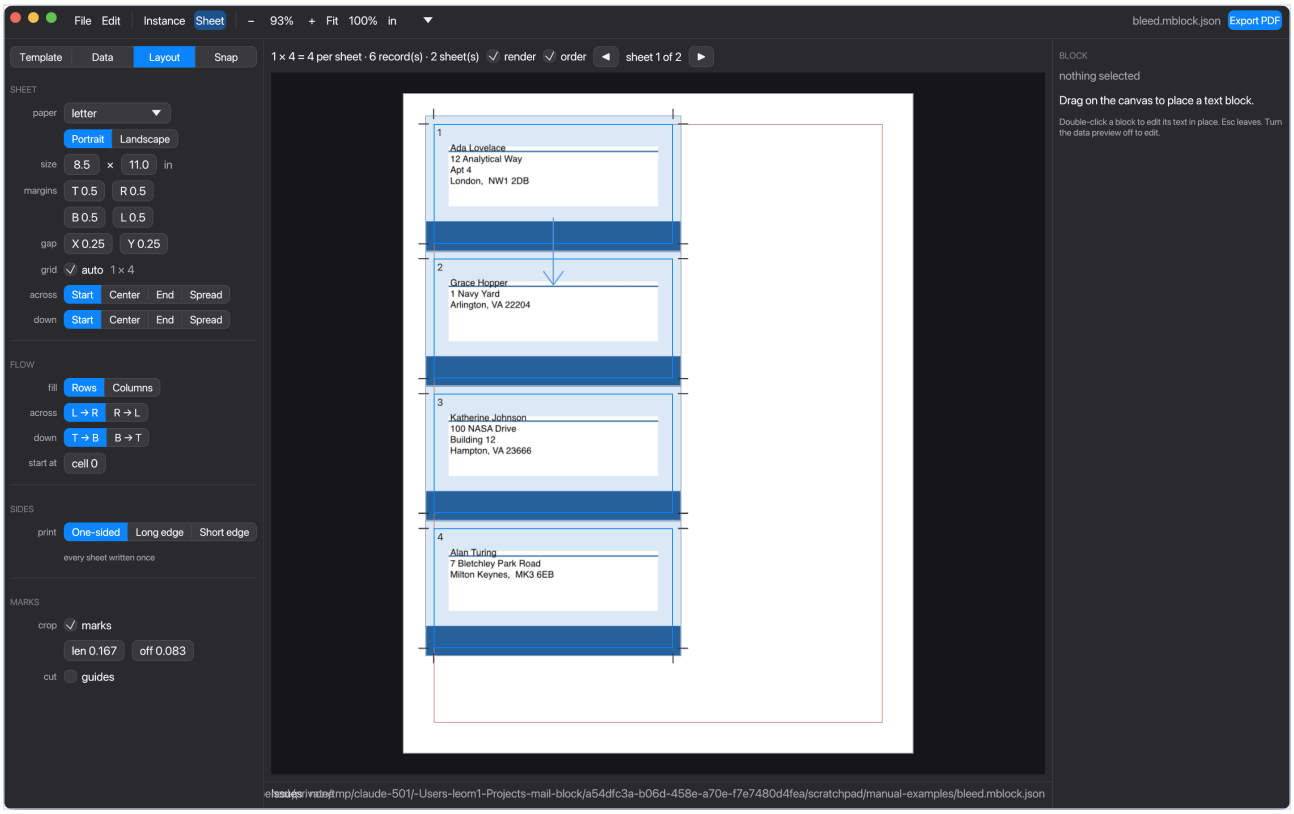
Fill order

flow decides which cell each record lands in: along the rows or down the columns, left to right or right to left, top to bottom or bottom to top. The arrow on the sheet view shows the direction; the numbers show the order.

start at skips cells on the first sheet, for part-used label stock.

Bleed and marks

Artwork that runs to the edge of the trim needs **bleed**: extra artwork beyond the trim so a slight misalignment on the guillotine does not leave a white sliver. Set it on the **Template** tab.



Bleed and crop marks on the sheet view.

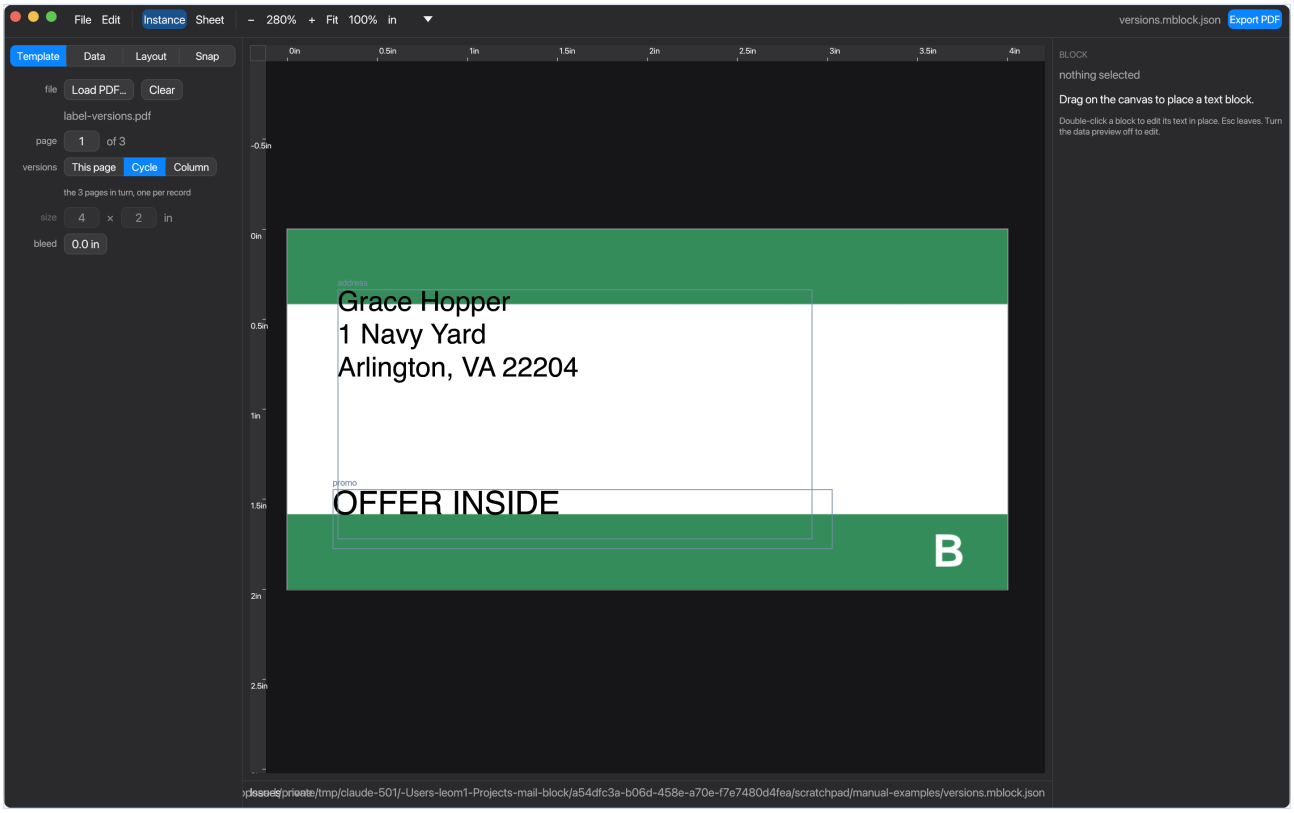
Bleed is not part of the cell. It reaches outside the trim into the gap, so adding it never re-lays the sheet, and neighbouring bleeds overlap in the gutter — which is exactly what lets one cut serve both. Ask for more bleed than the template page actually carries outside its trim and the app says so; the rest would come out as blank paper.

Marks, on the Layout tab, adds crop marks at each instance's trim corners, with a length and an offset measured out from the trim. Marks are drawn for every cell of the grid, including empty ones on a part-filled last sheet — the knife does not know the records ran out. A gutter between two instances is left unmarked, since both neighbours would mark the same cut line; that cut is marked at the edge of the sheet instead, on the same line.

cut guides draw a hairline around each instance. They are for proofing, not for the press.

Versions: multi-page templates

The pages of a template PDF are versions of the artwork — three designs to alternate, a different one per region — not a document to print through. Each instance takes one page.



A cycled template: this record is on version B, and the block bound to that version prints.

versions, on the Template tab, chooses how:

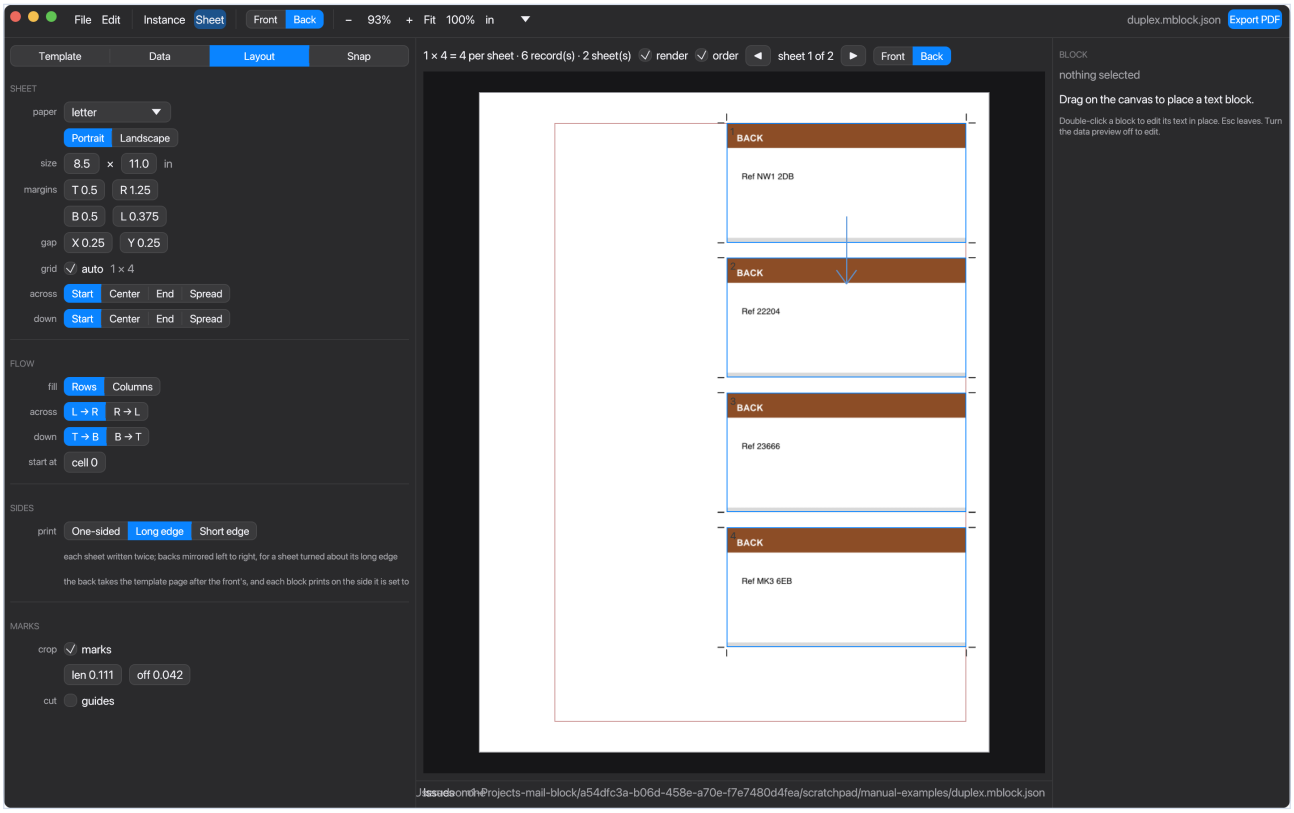
- **This page** — every instance uses the page you picked.
- **Cycle** — the pages in turn, one per record, coming round again at the end.
- **Column** — a column in your data holds the page number, counting from 1.

Only the pages a job actually uses are embedded, so a fifty-page template used one page at a time does not carry the other forty-nine into the output.

A block can belong to one version rather than all of them — the offer that only goes on version B. Untick **every page** in the block's inspector and give it a page. On the canvas, blocks belonging to another version keep a faint frame so you can still select and move them, but their text is not drawn: what you see is what that record prints.

Two-sided work

Turn **print** to **Long edge** or **Short edge** on the Layout tab and every sheet is written twice, front then back.



The back of the sheet: mirrored, with the back artwork and the blocks set to that side.

The backs are mirrored about the edge the sheet is turned on, so each cell lands behind its own front. Long edge is what an office duplexer does by default. Getting this backwards is the classic way to waste a run, which is why it is a setting and not a guess.

The back's artwork is the template page **after** the front's, so a two-sided job needs a template with at least two pages. Each block declares which side it prints on — front, back or both — and that setting is ignored entirely on a one-sided job.

The **Front / Back** switch at the top of the window chooses which pass you are working on. The sheet view mirrors with it, so a back shows the cells where they will actually print.

Exporting

Export PDF writes the job. The dialog offers:

- **crop marks** and **cut guides**, the same settings as the Layout tab.
- **split every N sheets**, which writes `out_001.pdf`, `out_002.pdf` and so on for presses that would rather not swallow a thousand-page file.
- **deterministic**, which drops the timestamp so two runs of the same job produce byte-identical files. Useful when a proof has to be compared against a reprint.

The report afterwards gives records read, records rendered, sheets, the grid, and every warning. Warnings are worth reading: they are the difference between a job that prints and a job that prints wrong.

What the issues bar tells you

It says	What to do
no column named "X"	The placeholder does not match a column heading. Check the spelling and the header row.
text was shrunk to N pt	A record was too long for its block. Look at it; widen the block or accept it.
text does not fit	The block's overflow is set to Error and a record overflowed.
the instance does not fit	The instance is larger than the sheet less its margins.
bleed is more than half the gap	Neighbouring artwork will overlap. Widen the gap or reduce the bleed.
asked for N pt of bleed	The template page has less artwork outside its trim than you asked for.
barcode: the payload must be …	A record's payload is not valid for the symbology.
bound to page N of an M-page template	A block is bound to a version that does not exist.

The command line

Everything the app does, `mail-block` does from a terminal, which is what you want for a job that runs every week:

```
mail-block render --project job.mblock.json
mail-block render --project job.mblock.json --csv new-data.csv --out october.pdf
mail-block plan --project job.mblock.json
```

`plan` reports what would happen without writing anything. `--strict` turns any warning into a failure, which is what you want in a script. `--report -` writes the whole report as JSON to standard output.

The project file

A project is a JSON file you can read, diff and keep in version control. It holds the blocks, the layout, the output settings, and the paths to the template and the CSV — relative to the project file where possible, so a job folder can be moved or copied without breaking.

Saving is ⌘S. The window title shows the file name, with "— edited" while there is unsaved work.