



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.

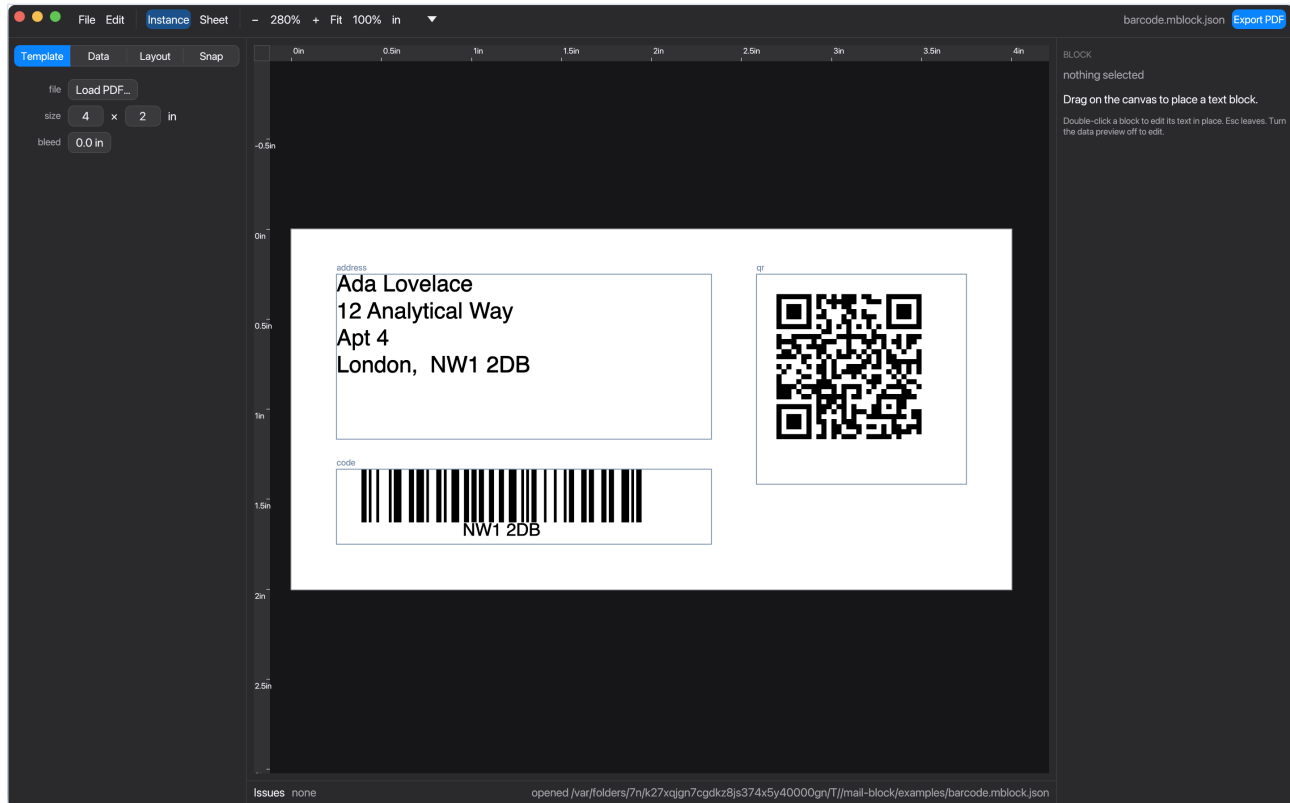
- **Template** — the artwork for one copy, as a PDF. One label, not a sheet of them.
- **Instance** — one copy on the sheet. Its size is the template's trim box.
- **Block** — a rectangle of text you place on the instance. Its text can contain placeholders, and it can be drawn as a barcode instead of 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

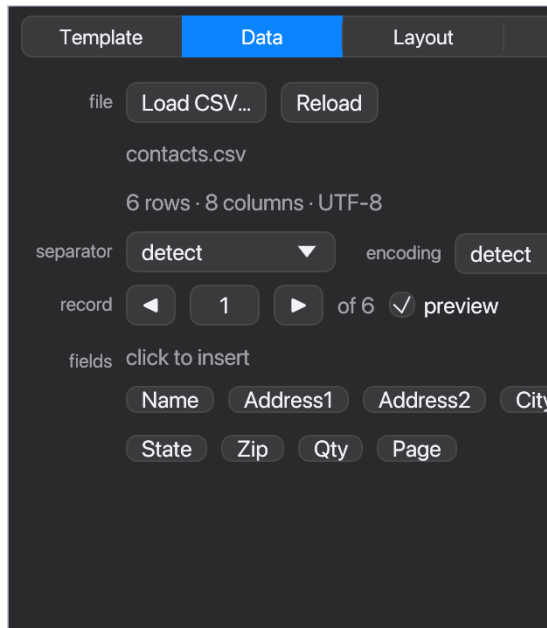
On the **Template** tab, press **Load PDF...** and choose your artwork. The instance size comes from the page's trim box, falling back to the crop box and then the media box, so a PDF prepared for print is measured the way the printer measured it.

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.

Artwork that runs past the trim — anything with bleed — is drawn where it really is, so it extends beyond the instance on the canvas and is shaded there. What sits inside the bright rectangle is what gets delivered; blocks are placed against that, which is what the imposition uses.

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.

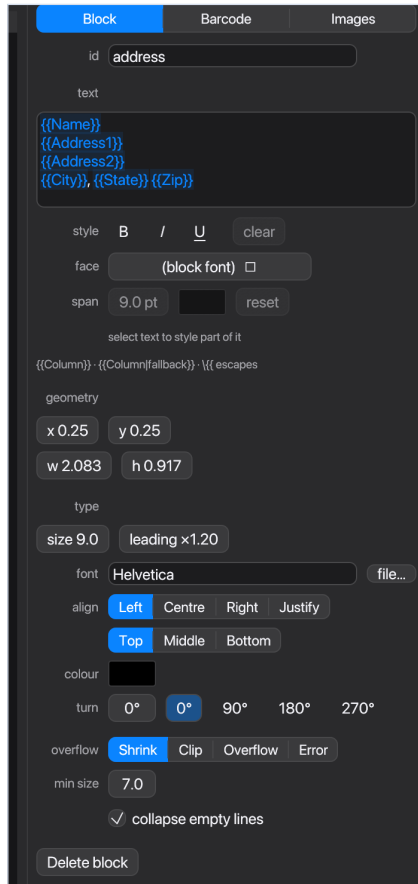
separator and **encoding** are both worked out from the file, and both can be set when the guess is wrong: a semicolon export from a European spreadsheet that reads as one column, or a Windows-1252 file that sniffs as UTF-8 until the first row with an umlaut in it. Setting either re-reads the file straight away.

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 and the block's own leading and letter spacing, so what you are typing into is spaced the way it will print. **Esc** leaves, and so does clicking anywhere else on the canvas — that click still does its job, so clicking another block finishes this one and selects that one in a single gesture. Clicking the inspector does *not* leave, since its style controls act on the text you have selected. Nothing needs saving: every keystroke is already in the project. This only works with the data preview off, since merged text is not yours to edit.

What the canvas is showing

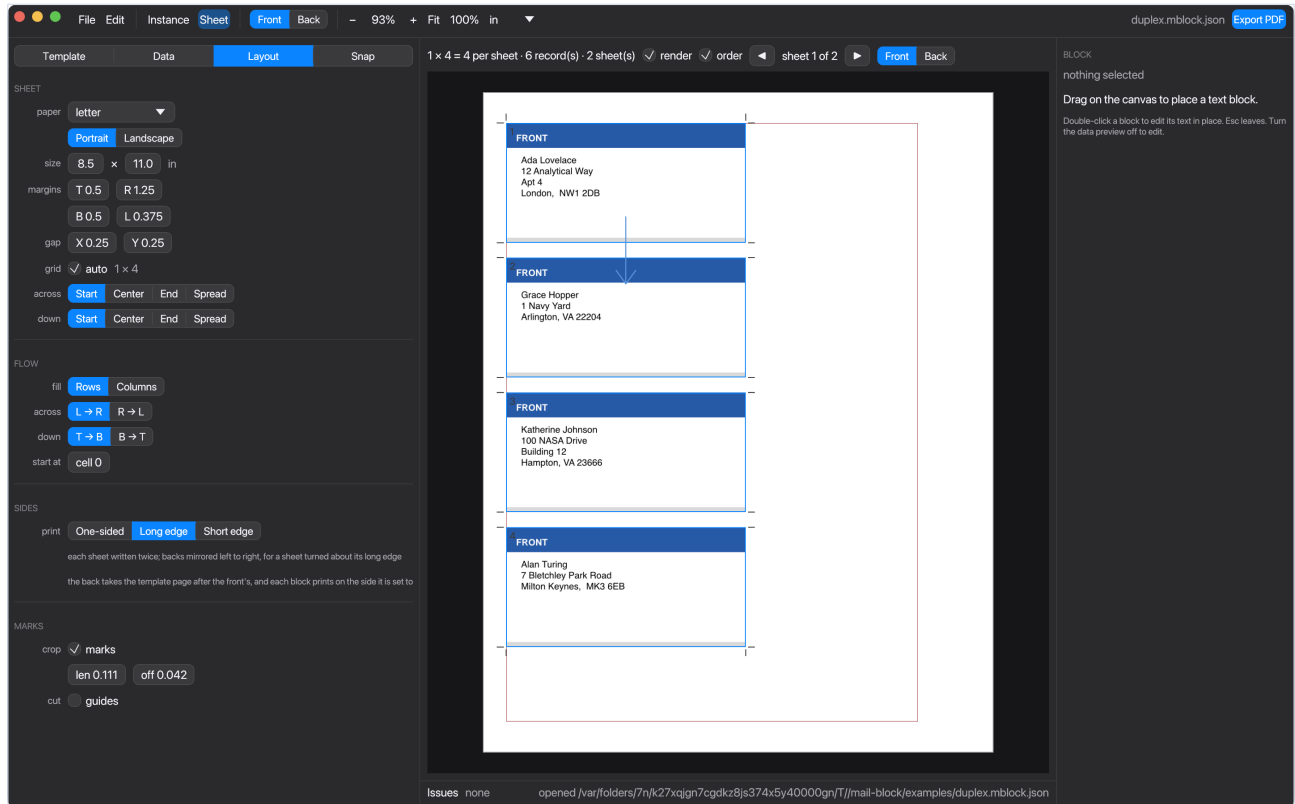
The instance canvas is not a drawing of your job — it is your job. The artwork and the type under the block frames are written to a PDF by the same code that exports, and rasterised; only the part on screen is drawn, so zooming in shows more detail rather than bigger pixels. Anything that will print, prints there: shrunk text, a barcode's quiet zone, the hairline you were not sure about.

Two things are drawn by the app rather than the writer, because they have to keep up with the mouse: the block you are dragging, resizing or turning, and the block you are editing in place. Both catch up with the

proof the moment you let go.

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.

What it draws is the sheet's own PDF, made by the same code the export uses, so it cannot drift from what you get. Only the part on screen is drawn, at the resolution of your screen: zoom in on a barcode or a hairline rule and you see the rule, not a larger version of the same few pixels.

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.

Turning the template

Artwork sometimes arrives on its side — landscape when the job is portrait — and the person who could re-export it has gone home. **turn** on the Template tab rotates the artwork by a quarter, a half or three quarters, on top of whatever rotation the page already carries.

A quarter turn puts the instance on its side too: a 4 × 2 inch label becomes 2 × 4, the grid refits, and the blocks keep their places on the turned instance. Like the page boxes, the turn is kept in the project — your PDF is not written to.

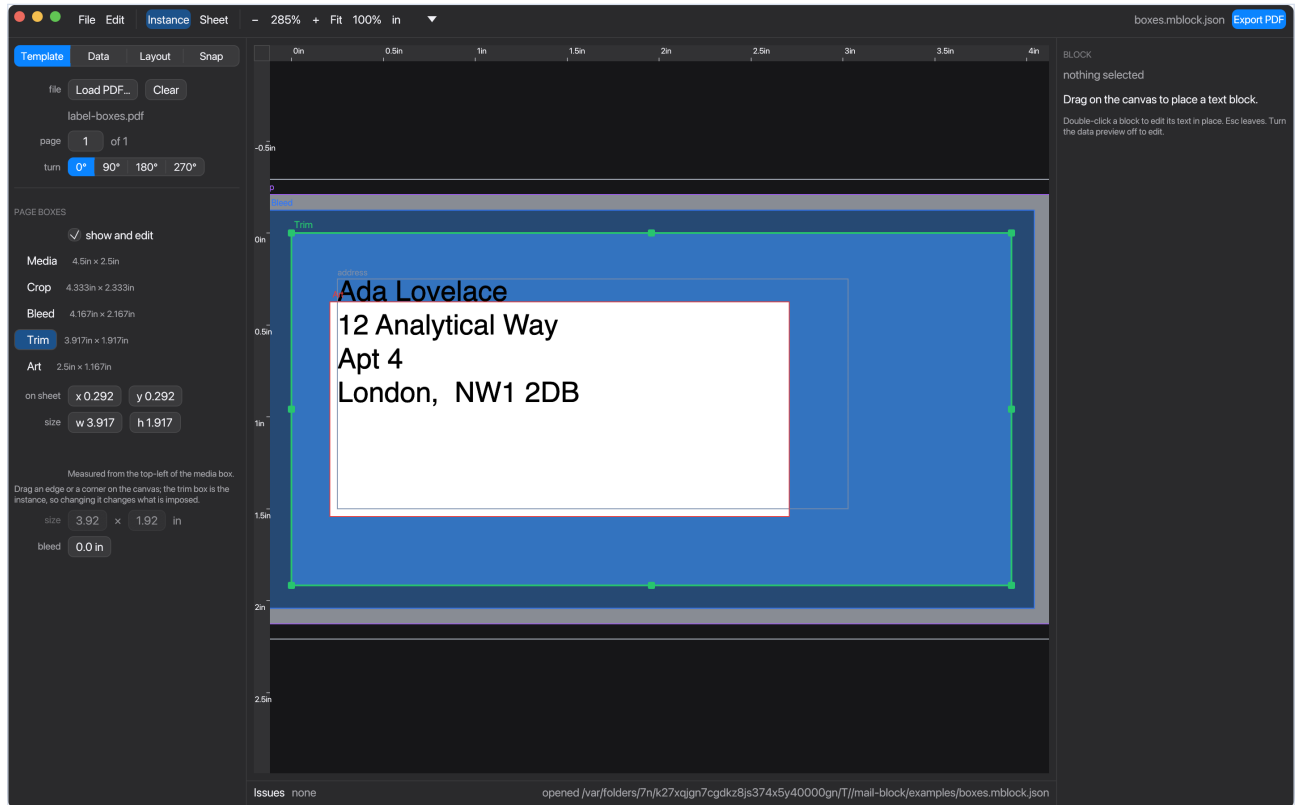
Only right angles. Anything else would have to turn the trim box with it, and a trim box is an axis-aligned rectangle by definition: the guillotine does not cut at 37 degrees. Blocks turn freely; the sheet they sit on turns in right angles.

Page boxes

A PDF page carries up to five rectangles, and they are what a printer works from:

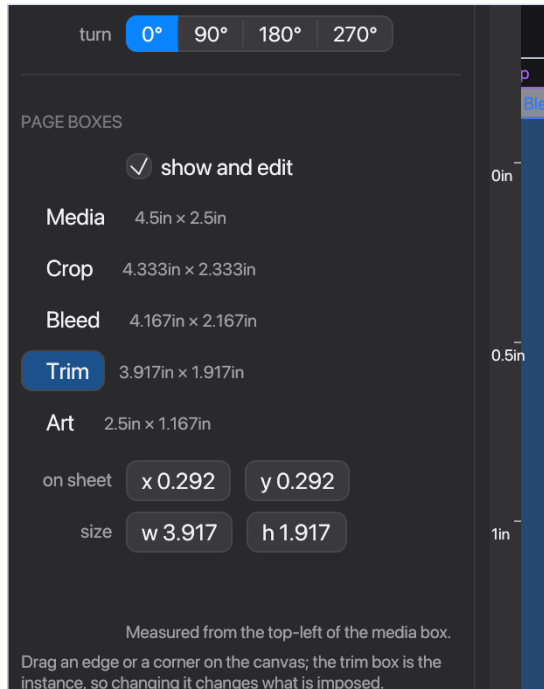
- **Media** — the sheet the page is drawn on. Everything else sits inside it.
- **Crop** — what a viewer shows.
- **Bleed** — how far the artwork runs past the trim, for the guillotine to cut through.
- **Trim** — the finished piece. This is the instance, and the imposition is measured by it.
- **Art** — the meaningful artwork. Carried and shown here, but nothing in the imposition reads it.

Artwork that imposes badly usually has a trim box that is wrong, or has no trim box at all so the crop or the media box stands in for it — and the person imposing is rarely the person who can re-export the file. Tick **show and edit** on the Template tab and all five are drawn on the canvas, each in its own colour.



The five boxes on the canvas, with the trim box selected and its handles showing.

Click a box's edge to select it, then drag an edge or a corner to change it; drag from inside the edge to move the whole box. The numbers beside the list do the same thing exactly: **on sheet** is where the box's top-left corner sits on the media box, and **size** is how big it is. The list shows each box's size with a dot against the ones you have changed.



Every box, its size, and the numbers for the one selected.

The list and the canvas both show what the imposition is using, so a box you change reads back changed; the trim box, being the instance, reports as the whole of it.

Changes are kept **in the project, not in the PDF**. Your artwork file is never written to, **Reset** puts a box back to what the file says, and two projects can trim the same artwork differently. Since the trim box is the instance, changing it changes what gets imposed — the grid refits as you let go.

The app expects the boxes to nest — trim inside bleed inside media — and says so in the issues bar when they do not. PDF does not require it, so this is a warning rather than a refusal.

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), and [color=...]. [/] closes the innermost tag.

A colour in a tag is written in whichever space it belongs to:

Written	Means
#B00, #BB0000	red, green, blue
black, white, grey, red, green, blue	the obvious
cmk(0, 91, 100, 0) or cmk(0, 0.91, 1, 0)	four inks — per cent, or fractions
k(40)	40% black ink
gray(0.3)	grey by lightness, where 0 is black

The **span colour** control writes these for you, in whatever space the block is in: set a CMYK block's selection and the tag it writes carries the four inks, not a hex triplet somebody converted.

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.

Fonts installed while the app is open

The face list is every font on the machine, read when the app starts. Install a font while mail-block is running and it appears by itself: the app looks again whenever you come back to its window, and every few seconds while you are in it — so install the font, switch back, and it is in the list. **Rescan**, at the top of the face list, does the same on demand.

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.

Colour

colour is written in one of three spaces, and the choice reaches the PDF:

- **RGB** — three channels, 0 to 255. What screen work and office printers expect.

- **CMYK** — four inks, 0 to 100%. What a press expects: the four numbers you type are the four numbers in the file, so a separation comes out as specified rather than as something a converter guessed. C 0 M 91 Y 100 K 0 stays that, exactly.
- **Grey** — one ink, 0 to 100%. K 100 is solid black.

Type the numbers, or pick by eye with the swatch beside them; picking keeps whatever space the block is in. Switching space **converts** — the colour that looked right still looks right — so a CMYK block turned to RGB shows the RGB that matches it, not the same four numbers reinterpreted.

The canvas shows CMYK the way a viewer does, without a profile. What prints is the numbers.

Swatches

The row under the numbers keeps colours for later. + saves the one you are looking at; clicking a saved swatch applies it, in the space it was saved in, so a customer's CMYK red stays four inks. Right-click a swatch to remove it. The library is kept with the app's own settings, not with the project, because a palette outlives the job that introduced it.

Load... and **Save...** read and write **Adobe Swatch Exchange** (.ase) — the file design tools and print shops pass palettes around in. Colours keep their space through the file, so a CMYK palette arrives as CMYK. Loading merges: what you already have is kept, and re-loading your own palette adds nothing.

The eyedropper

■, beside the colour numbers, picks a colour off the canvas. Click it, then click anywhere on the instance: the colour comes from the **rendered page**, so what you pick is the ink that will print — the artwork's own red, a photograph's background, the grey of a rule — and not an approximation of it. The picked colour arrives in the space the block is written in. Click ■ again to put the dropper down.

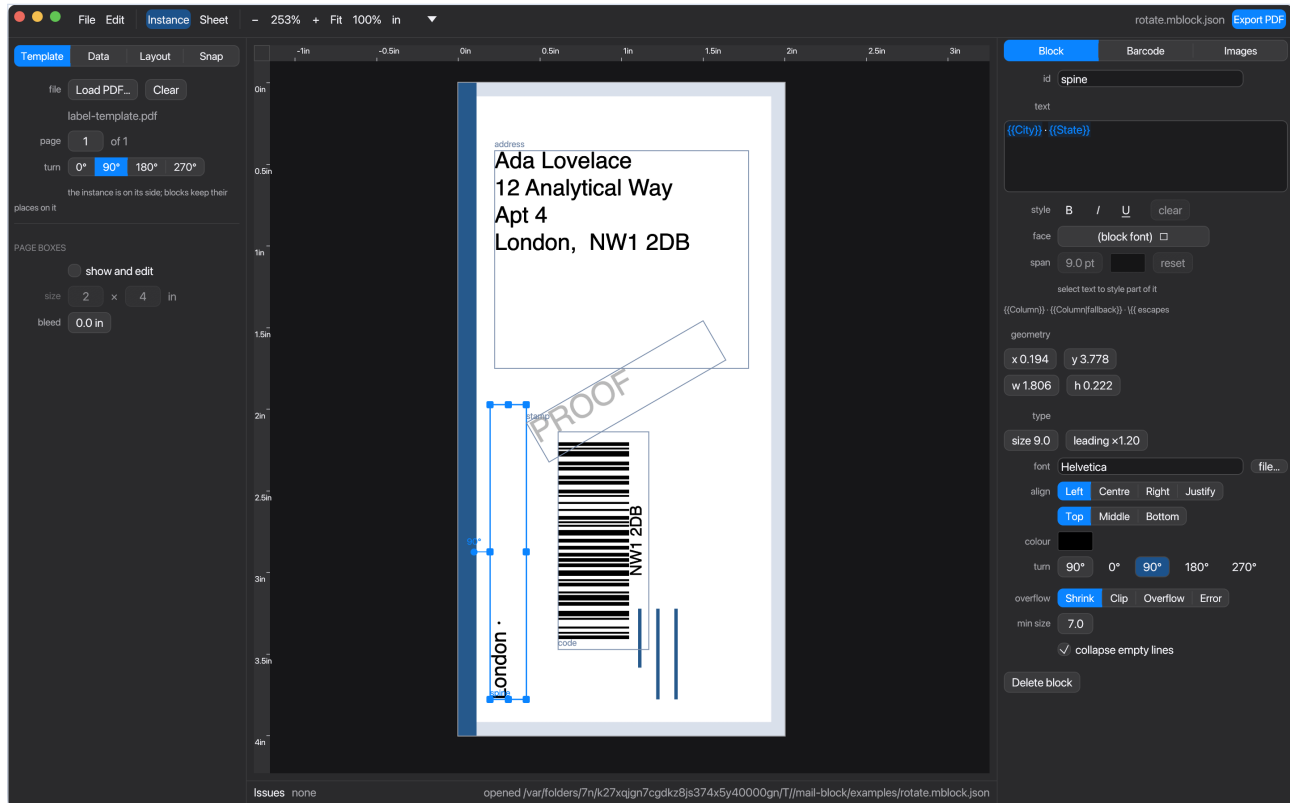
Position and alignment

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.

Turning a block

turn rotates the block anticlockwise about its **top-left corner** — the corner stays where it is and the rest swings around it. Type an angle, click one of 0°, 90°, 180° or 270°, or grab the round knob above the selected block on the canvas and drag. A dragged turn settles on 15° steps; hold ■ for any angle in between.



A turned template with a block on its side, a barcode turned with it, and one at an angle.

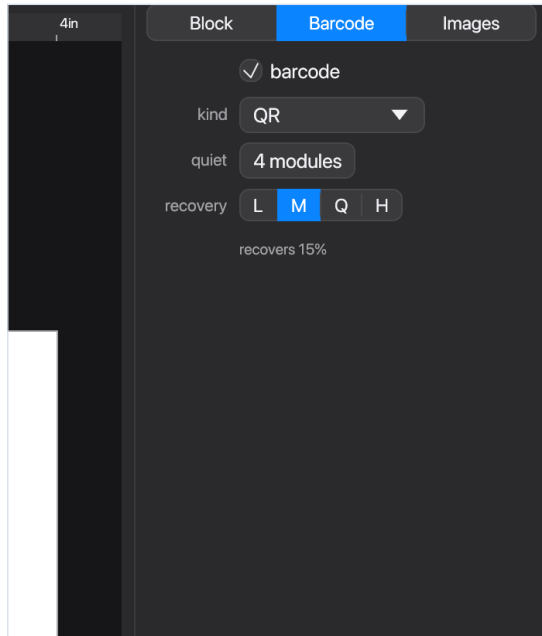
Everything in the block turns with it: the type, the underlines, and a barcode with its caption. The canvas draws it where it will print, and clicking, dragging and the resize handles all follow the block round, so a turned block is worked on exactly like a straight one.

Two things stay upright by design. The block's name floats above its corner as a label on the drawing rather than part of the artwork. And a block being **edited in place** straightens up while the caret is in it, because the caret, the selection and the text are drawn square; press Esc and it lies back down.

Snapping is suspended while a block is turned: the guides are the instance's edges and centre, and a turned block's sides do not run the way they do.

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.

Pictures

A block can draw a **picture** instead of type: a photograph on a badge, a product shot on a shelf label, a signature on a certificate. Select the block, open the inspector's **Images** tab and tick **image**.

The Images tab, and the picture the record on screen resolves to.

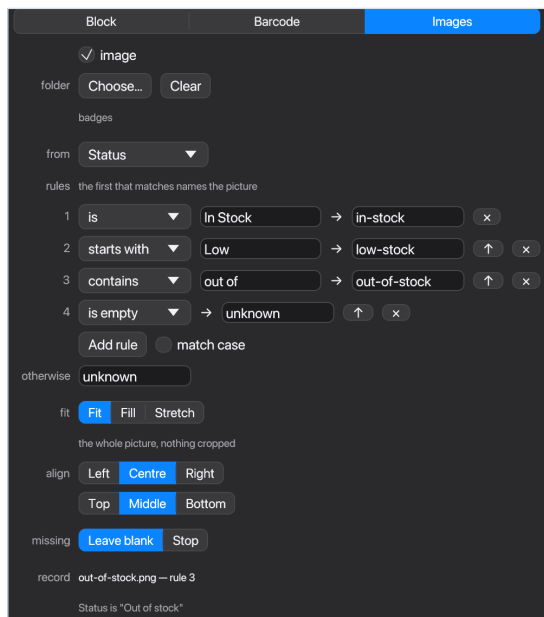
Two things make the picture: a **folder**, chosen once with **Choose...**, and a **name**, which is the block's own text merged with the record — exactly like every other block. So a column of file names, or of product codes, drives it:

The block's text	The record	What prints
{{Photo}}	Photo = ada	photos/ada.png
{{Sku}}.jpg	Sku = 4711	photos/4711.jpg
{{Line}}/{{Sku}}	Line = spring	photos/spring/4711.jpg

If the name has no extension, .jpg, .jpeg and .png are tried in turn, so a column of bare product codes works without editing the spreadsheet. The **fields** buttons put a column into the block's text for you. A name from the data cannot climb out of the folder you chose: . . / and absolute paths are refused, because the folder is your decision and the spreadsheet is not.

Conditional pictures

Often the picture is not named in the data at all: what the data has is a **status**, a **grade**, a **quantity**, and the picture follows from it. Set **from** to that column and the block is driven by **rules** instead of by its text.



Rules on a Status column, and what the record on screen made of them.

Each rule is a test and a picture. They are tried **in order** and the first that matches names the file, so put the specific ones above the general ones — the ↑ arrow moves a rule up. **otherwise** names the picture for a record that matched nothing; leave it blank and such a record simply gets none.

The tests:

Test	Matches when the value...
is / is not	equals, or does not equal, what you type
contains	has it anywhere inside
starts with / ends with	begins or finishes with it
is above / is below	sorts after or before it
is between	falls in a range, ends included, either way round
is empty / is not empty	the cell is blank, or is not
matches	fits a regular expression

Values are trimmed, so trailing spaces in a spreadsheet cell make no difference, and capitals are ignored unless you tick **match case** — a rule for **gold** also matches **gold**. **is above**, **is below** and **is between** compare as *numbers* when both sides are numbers and as *text* when they are not, so the same three tests serve a quantity column and a grade column: 9 is below 10, and A is below B.

Under the controls, the **record** line says which file the record on screen resolved to and which rule chose it, along with the value it read. Rules that name a picture the folder does not have, and patterns that are not patterns, are listed in the issues bar while you type rather than in the middle of a run.

JPEG and PNG. A JPEG is carried into the PDF exactly as it is on disk — not decoded, not recompressed, nothing lost. A PNG is decoded and written as samples, and its transparency comes through as a soft mask, so a cut-out photograph stays cut out. Each file is read once and written into the PDF once, however many records show it: a run of ten thousand badges over twelve photographs carries twelve.

fit decides how the picture sits in the block:

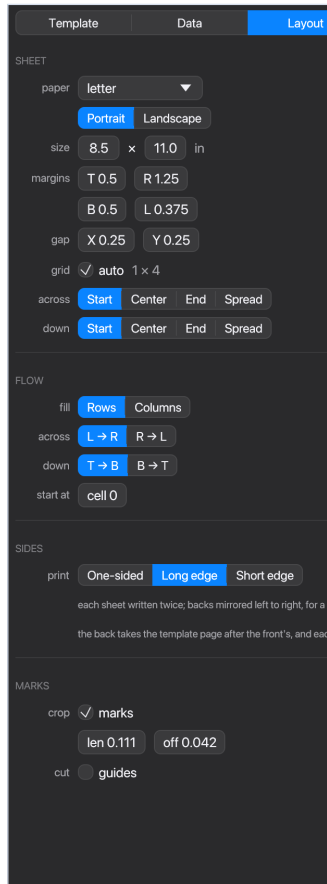
- **Fit** — the whole picture, as large as it goes, proportions kept. Nothing is cropped.
- **Fill** — fills the block, proportions kept, and whatever hangs over the edge is cut off.
- **Stretch** — fills the block exactly, proportions and all.

align places what is left over when fitting, and does nothing when filling or stretching, since neither leaves any.

missing decides what a record with no picture means. **Leave blank** prints the block empty and lists it in the report, once per name however many records ask for it — a catalogue with a few photographs still to come should still print. **Stop** ends the export instead, naming the block, the record and the file.

Turning a picture block turns its picture with it, and the canvas shows exactly what will print, since it is drawn by the same code.

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

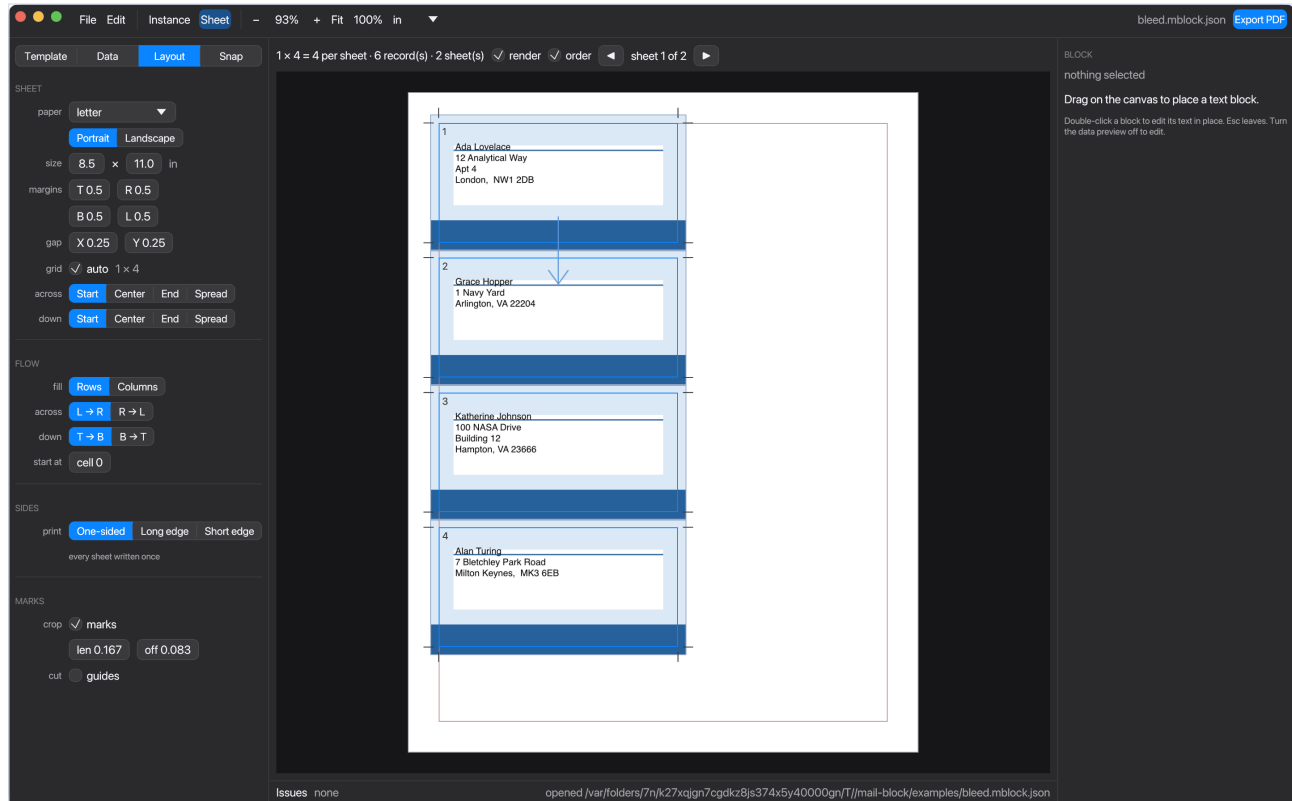
flow decides which cell each record lands in: along the rows or down the columns, left to right or right to left, top to bottom or bottom to top. The arrow on the sheet view shows the direction; the numbers show the order.

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.

A new project starts at **0.125 in** — an eighth of an inch, which is what a printer will ask for if you do not offer it first, and exactly half the default gutter, so neighbouring bleeds meet rather than overlap. Set it to zero for work that does not print to the edge. A project **file** that says nothing about bleed still means none: the default is for new work, and opening an old job never changes it.



Bleed and crop marks on the sheet view.

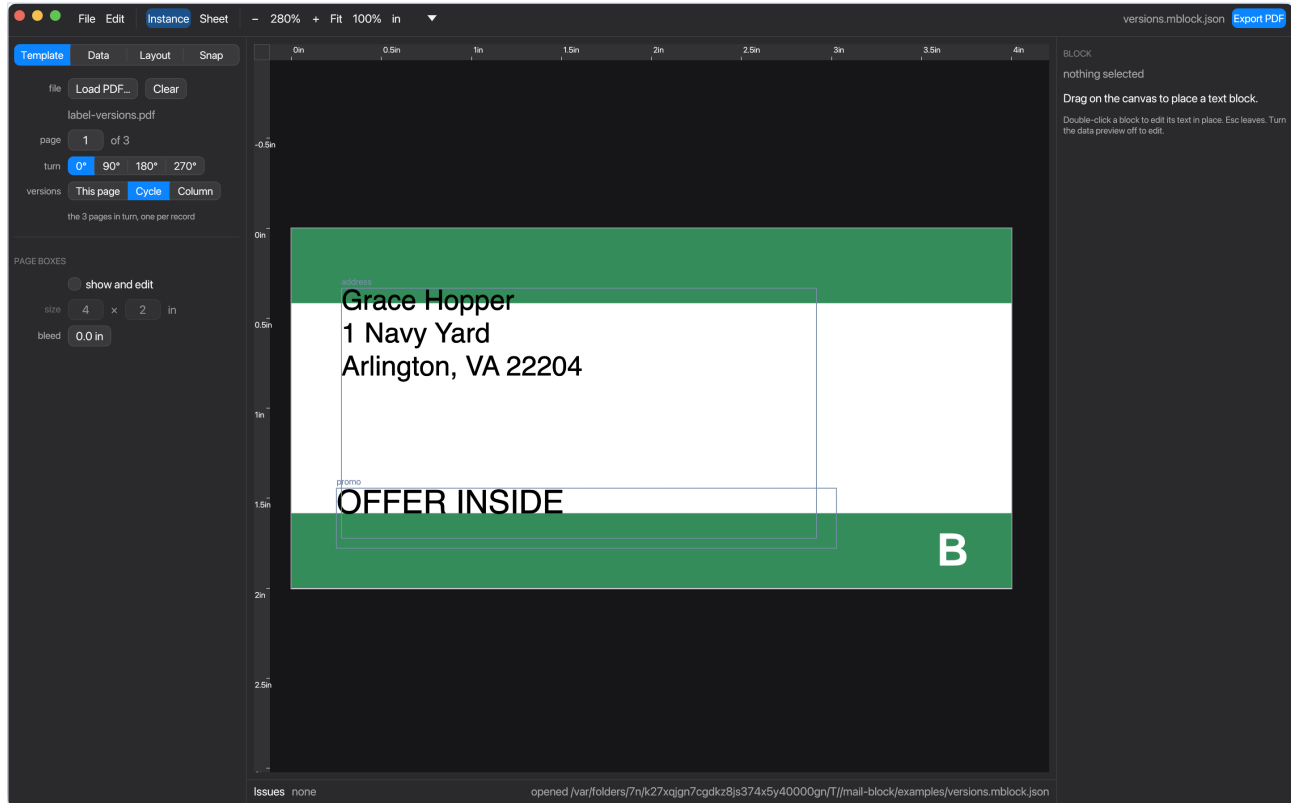
Bleed is not part of the cell. It reaches outside the trim into the gap, so adding it never re-lays the sheet, and neighbouring bleeds overlap in the gutter — which is exactly what lets one cut serve both. Ask for more bleed than the template page actually carries outside its trim and the app says so;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

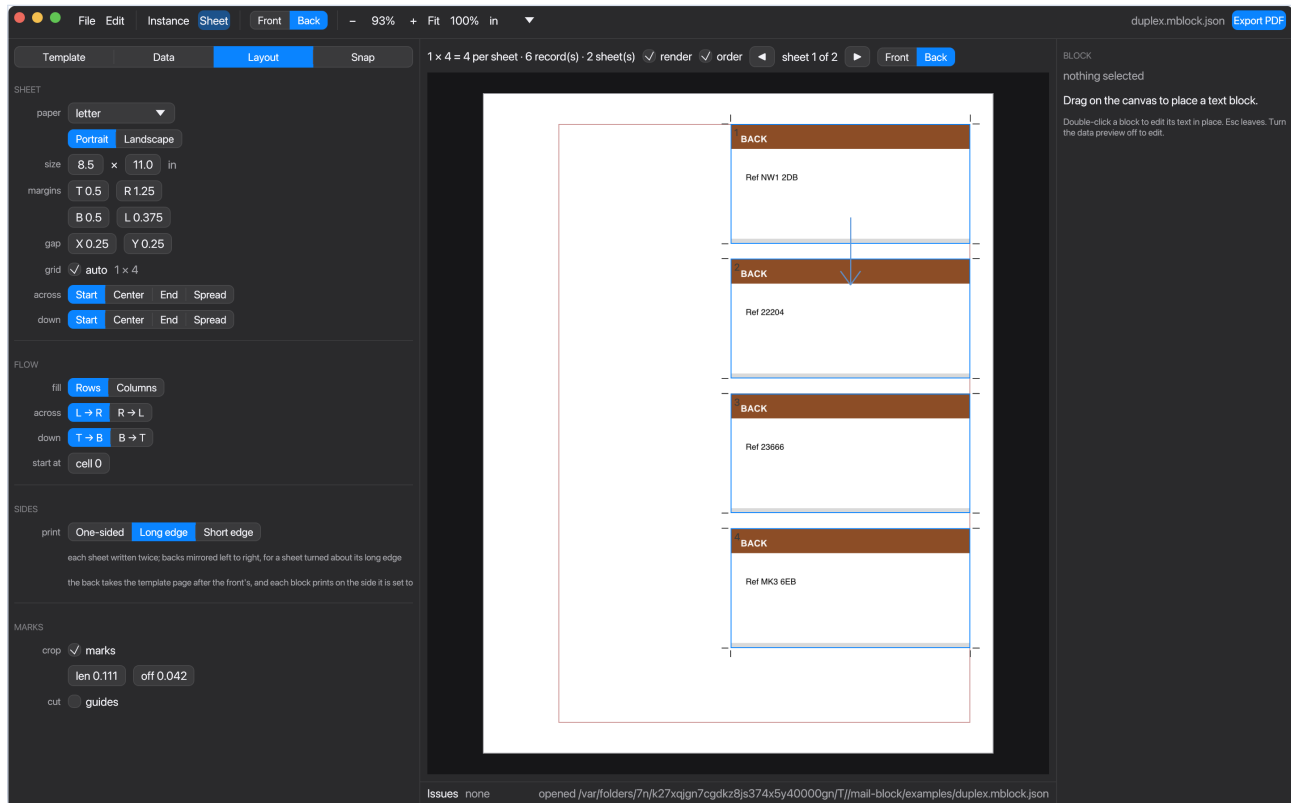
- **This page** — every instance uses the page you picked.
- **Cycle** — the pages in turn, one per record, coming round again at the end.
- **Column** — a column in your data holds the page number, counting from 1.

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, asking where to put it. What it writes is set beforehand, in **Output** at the foot of the Layout tab:

- **file** — where it goes. Choose it once and Export uses it; the folder is shown underneath so you can see where the run will land.
- **batch** — split the run into files of a fixed quantity, for a press that would rather not swallow a thousand-page file, or a job that goes out in boxes. See below.

- **reproducible** — drop the timestamp and fix the document id, so two runs of the same job come out byte for byte identical. Useful when a proof has to be compared against a reprint.

Crop marks and **cut guides** are on the same tab, under Marks.

Batches

Tick **batch** and say how many go in each file. **records** counts the pieces: every file holds exactly that many — five hundred to a box, five hundred to a file — which means a batch starts on a fresh sheet, and the sheet that ends one may be part empty. **sheets** counts pages instead: the sheets are filled as usual and the run is cut into files afterwards, so nothing is part empty but the pieces per file follow from what fits.

The line underneath says what your setting comes to before you run it — how many files, how many in the last one, and how many sheets it costs. The files are written beside the one you chose, numbered out_001.pdf, out_002.pdf and so on; an unbatched job keeps the plain name.

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.