

## **BFF-piRNA-seq reveals conserved processing hotspots in flies and mice**

### **Supplementary file information**

Parthena Konstantinidou<sup>1</sup>, Franziska Ahrend<sup>1,2</sup>, Brian J. Galletta<sup>3</sup>, Angel Jaimes<sup>1</sup>, Dimitrios G. Anastasakis<sup>4</sup>, Nasser M. Rusan<sup>3</sup>, Markus Hafner<sup>5</sup>, Astrid D. Haase<sup>1,\*</sup>

### **Supplementary Information:**

#### **Supplementary Figures 1–7**

**Supplementary Methods:** BFF-piRNA-seq step-by-step protocol and troubleshooting guide.

**Supplementary Software:** Custom computational code used for processing and analysis of BFF-piRNA-seq data.

**Supplementary Data 1:** Piwi protein quantification in *Drosophila* ovarian somatic sheath cells (OSCs): raw data and calculations related to Fig. 3b.

### **Supplementary Tables:**

#### **Supplementary Data 2**

Top 200 pachytene piRNA clusters in adult mouse testes (mm10 coordinates), ranked by piRNA production according to BFF-MIWI-piRNA-seq. Related to Fig. 2 and Supplementary Fig. 2.

#### **Supplementary Data 3**

Unique-mapping intervals (“seeds”) within piRNA clusters in *Drosophila* ovarian somatic sheath cells (OSCs), defined using the piRNA Cluster Builder (PICB) and used for the systematic analysis of piRNA processing patterns shown in Fig. 6 (dm6; see Methods).

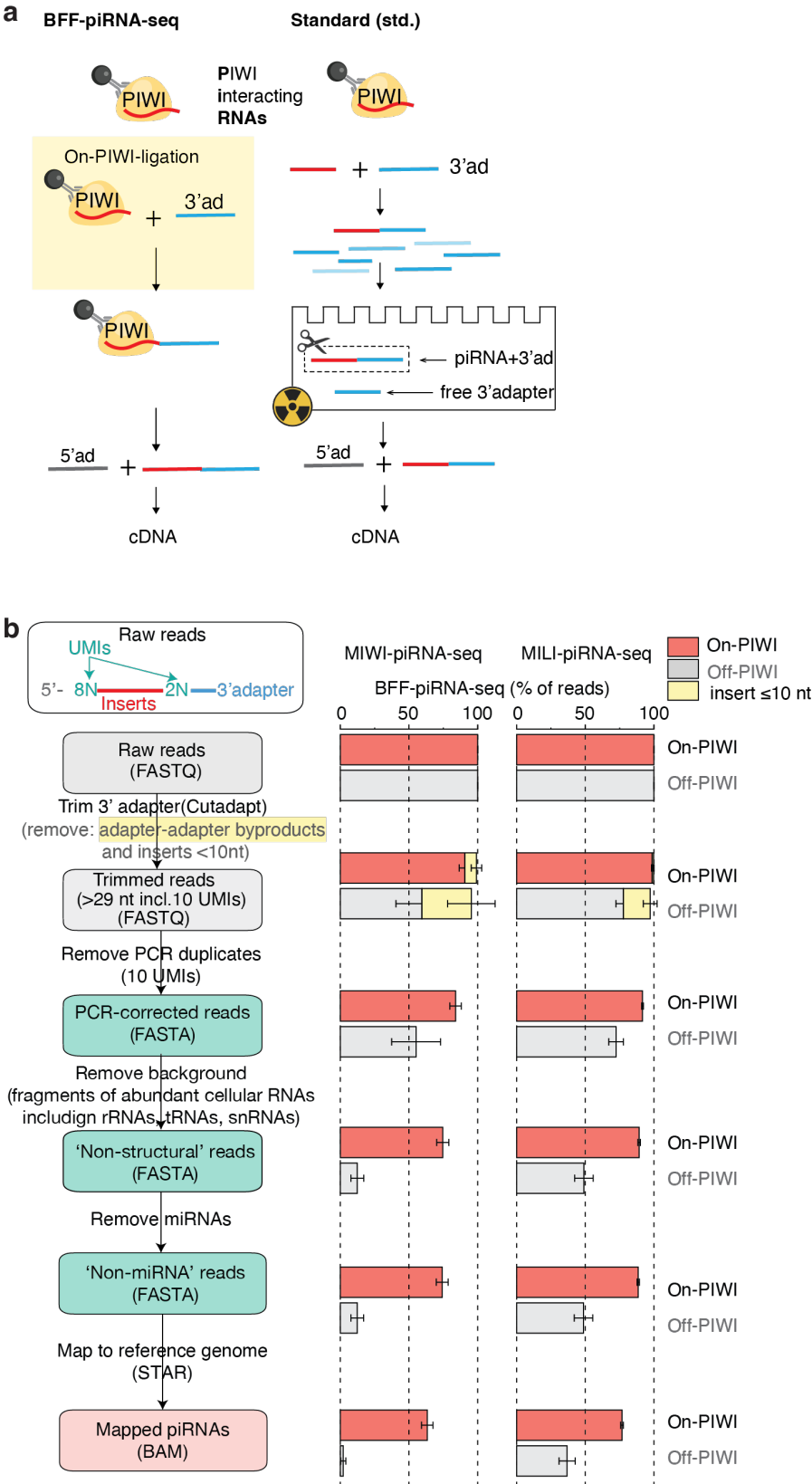
#### **Supplementary Data 4**

Unique-mapping intervals (“seeds”) within piRNA clusters in *Drosophila* ovarian follicle cells, defined from BFF-piRNA-seq data of purified tj-GAL4-driven FLAG–HA-Piwi piRNAs using the piRNA Cluster Builder (PICB). These intervals were used for the systematic analysis of piRNA processing patterns shown in Supplementary Fig. 6 (dm6; see Methods).

#### **Supplementary Data 5**

Unique-mapping intervals (“seeds”) within piRNA clusters in *Drosophila* ovarian germ cells, defined from BFF-piRNA-seq data of purified nos-GAL4-driven FLAG–HA-Piwi piRNAs using the piRNA Cluster Builder (PICB). These intervals were used for the systematic analysis of piRNA processing patterns shown in Supplementary Fig. 6 (dm6; see Methods).

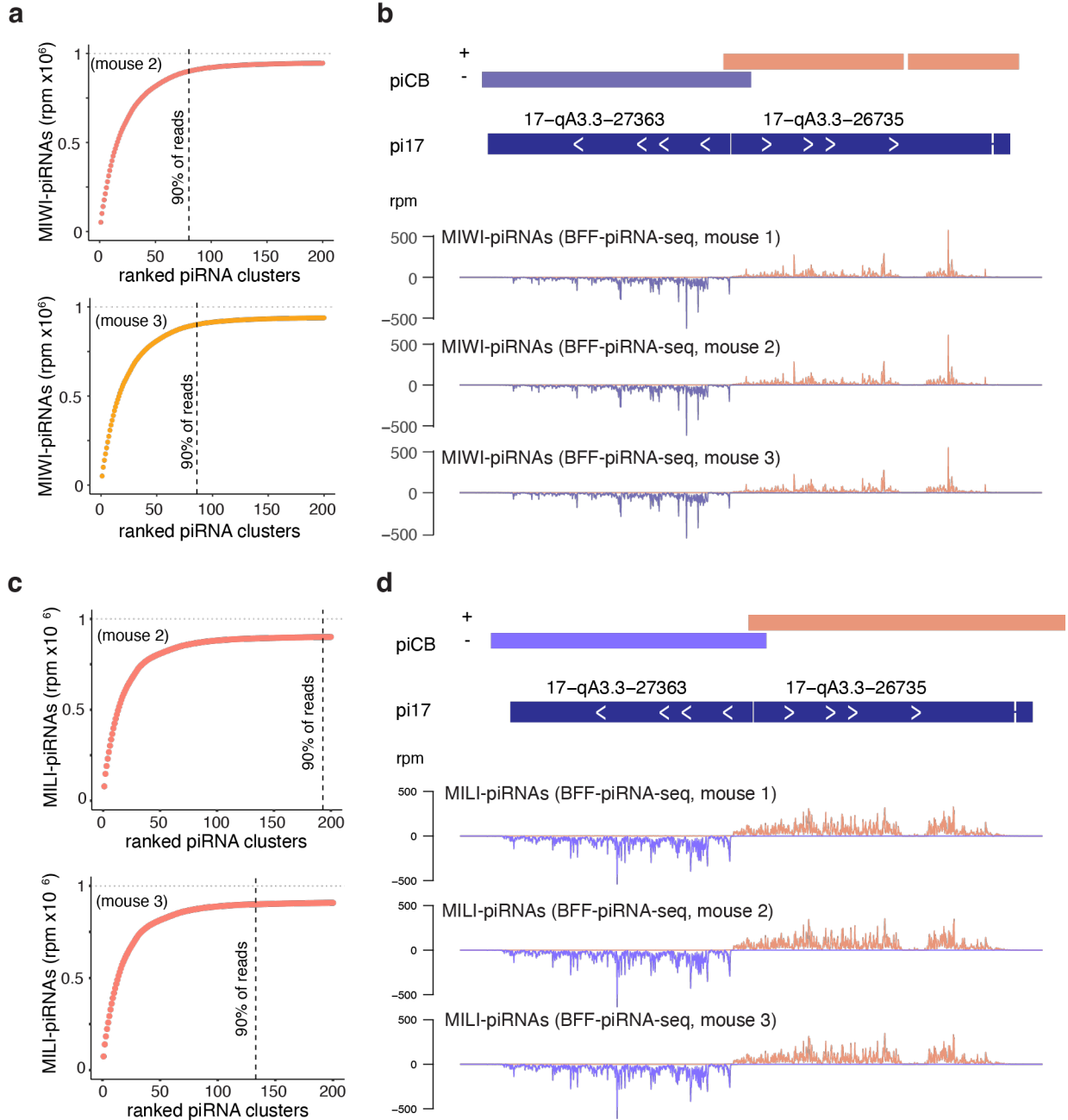
1    **Supplementary Figurures**



**Supplementary Fig. 1 | BFF-piRNA-seq experimental and computational workflows.**

(a) Schematic of BFF-piRNA-seq versus a standard (std.) PAGE-based cloning protocol.

(b) Read loss throughout data-analysis steps for MIWI- and MILI-piRNAs prepared with on-PIWI (red bar) versus off-PIWI (gray bar) 3' adapter ligation (mean  $\pm$  s.e.m., n = 3 individuals).



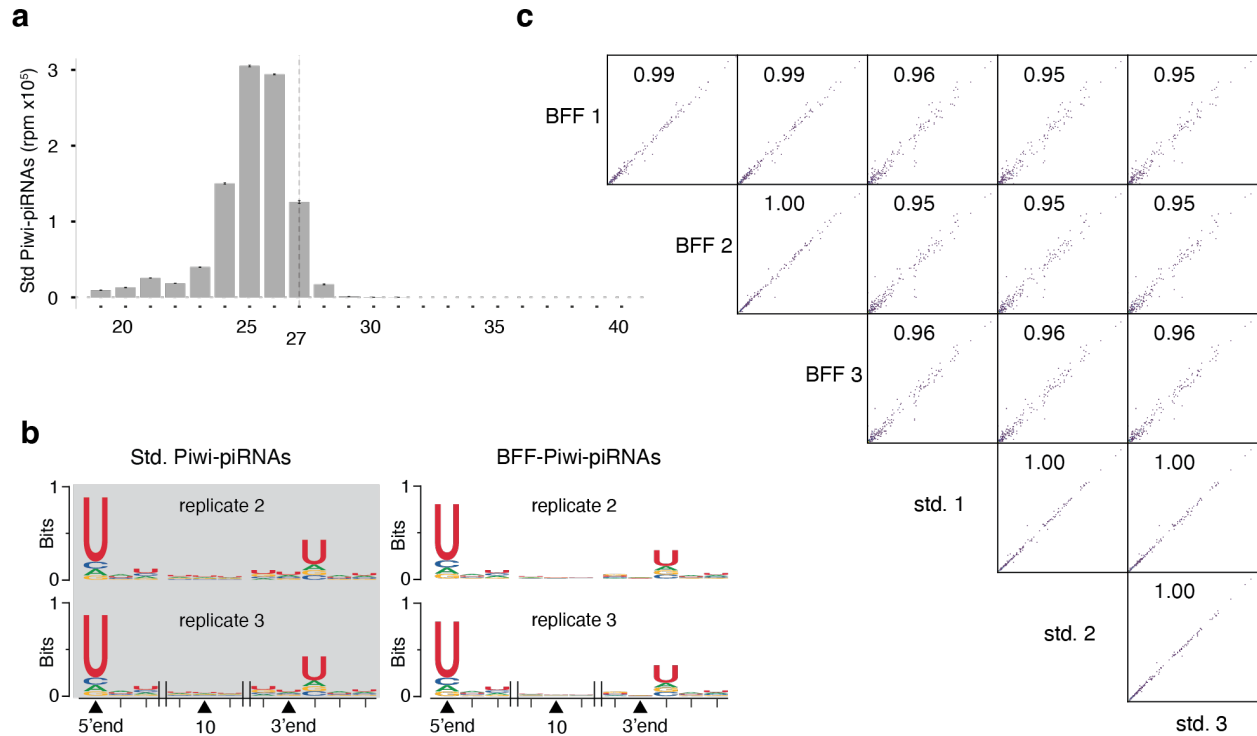
### Supplementary Fig. 2 | BFF-piRNA-seq captures pachytene piRNA clusters

(a) Cumulative BFF-piRNA abundance across previously published MIWI clusters ranked by productivity (mouse 2 and 3).

(b) BFF-MIWI-piRNAs originating from the bidirectional pachytene piRNA cluster, pi17 (chr17:27,284,000–27,372,300) (resolution: 50 nt bins; n = 3).

(c) Cumulative BFF-piRNA abundance across previously published MILI-piRNA clusters ranked by productivity (mouse 2 and 3).

(d) BFF-MILI-piRNAs originating from the bidirectional pachytene piRNA cluster, pi17 (chr17:27,284,000–27,372,300) (resolution: 50 nt bins; n = 3).

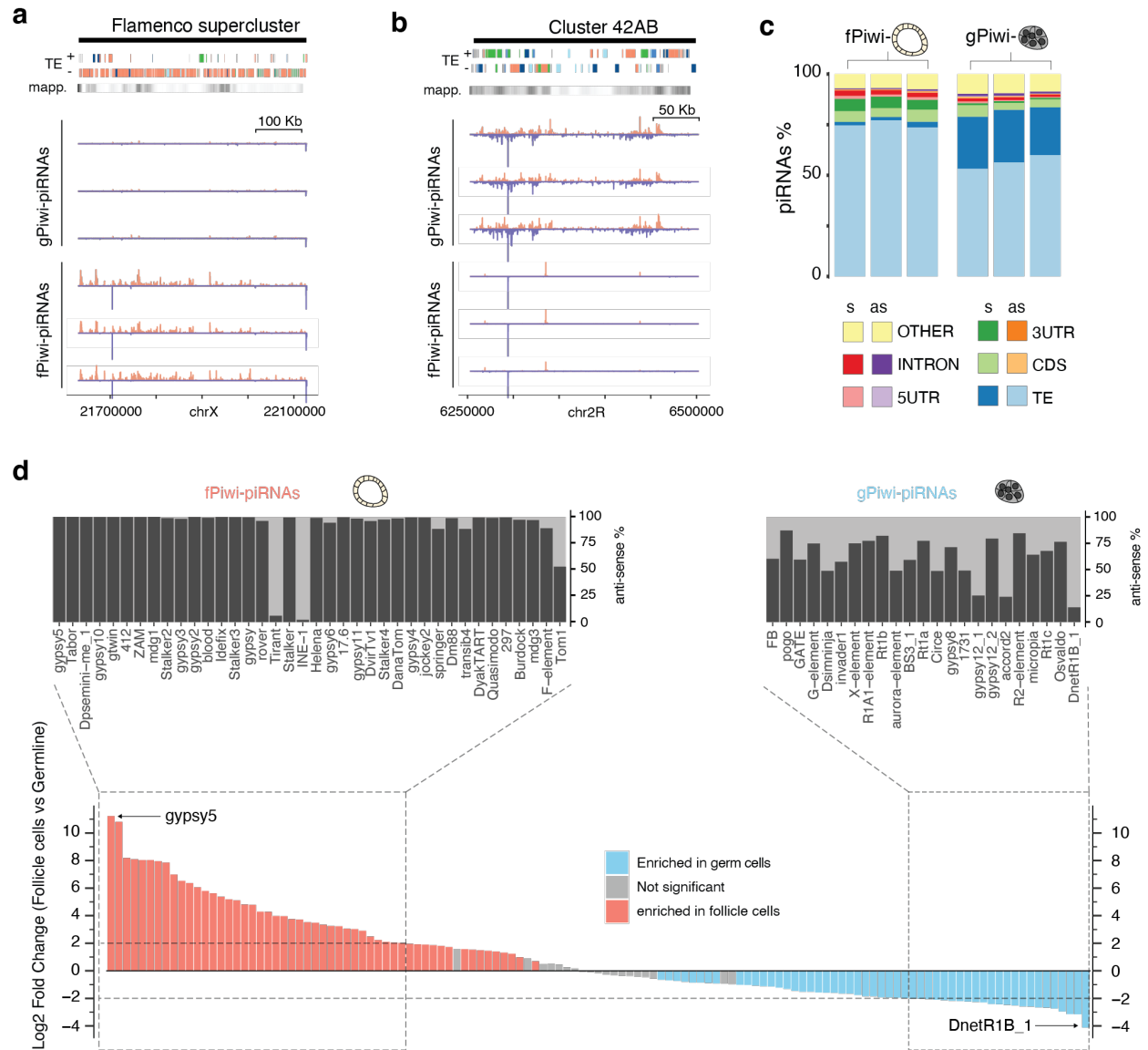


**Supplementary Fig. 3 | BFF-piRNA-seq provides an unrestricted view of piRNAs in OSC.**

(a) Size distribution of standard Piwi-piRNAs from *Drosophila* OSCs (mean  $\pm$  s.e.m.,  $n = 3$  biological replicates) (SRP277087).

(b) Sequence logos of Piwi-piRNAs and downstream genomic sequences (biological replicates 2 and 3).

(c) Relative Piwi-piRNA abundance across the most productive piRNA clusters (90th percentile). Spearman correlation coefficient ( $r_s$ );  $n = 3$ .

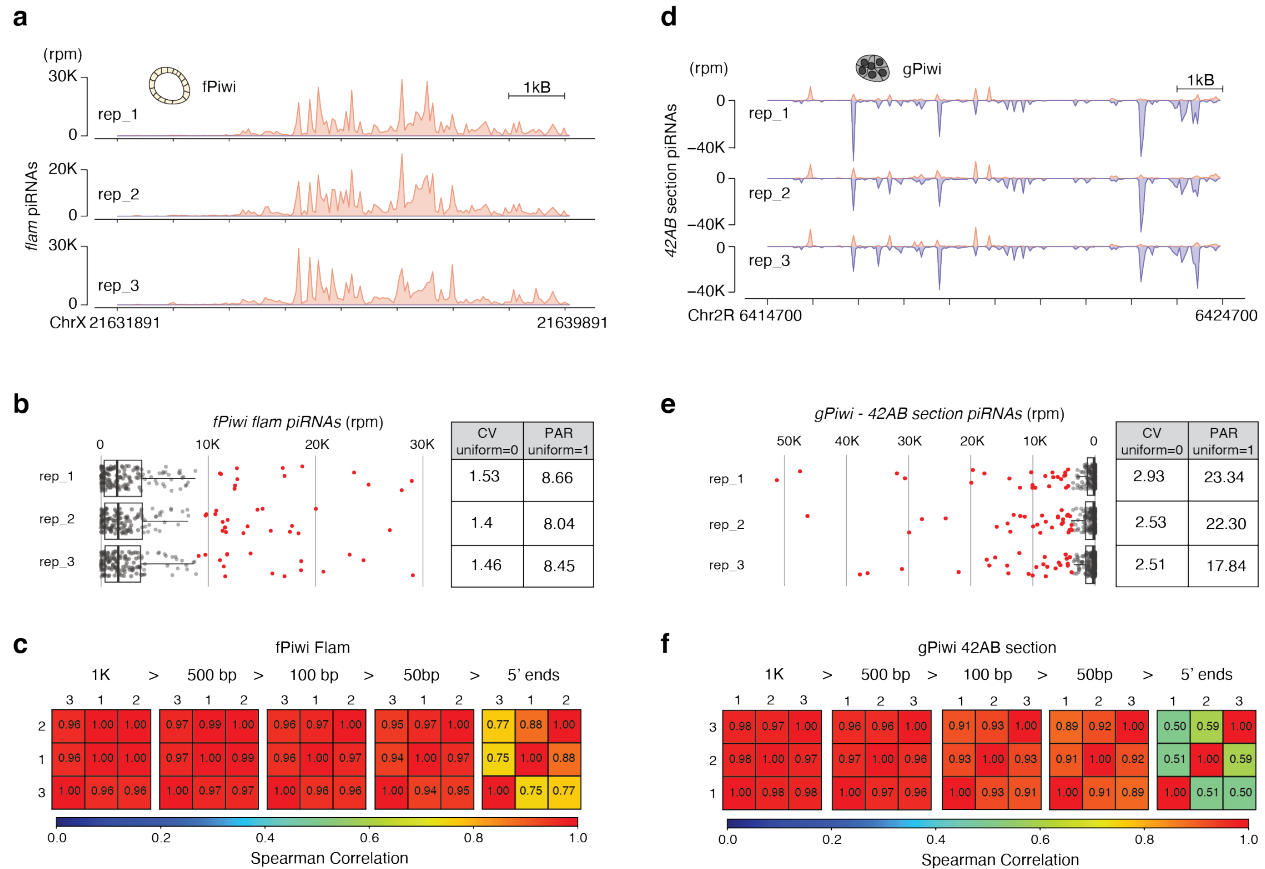


**Supplementary Fig. 4 | BFF-piRNA-seq provides an unrestricted view of piRNAs from the ovarian soma and germ cells.**

(a) Follicle versus germline Piwi-piRNAs mapping to representative somatic (flamenco) and (b) germline (42AB) clusters.

(c) Genomic annotation of follicle versus germline Piwi-piRNAs (n = 3 each).

(d) Individual transposon enrichment in follicle versus germline Piwi-piRNAs. piRNA orientation (sense or antisense) is indicated for transposons with  $\geq 4$ -fold enrichment ( $\log_2$  fold-change  $> 2$  or  $< -2$ ).



**Supplementary Fig. 5 | BFF-piRNA-seq reveals robust piRNA patterns with solitary processing hotspots in flies.**

(a) Follicular Piwi- (fPiwi-) piRNA coverage across *flamenco* 5' region (chrX:21,631,891–21,640,020, n = 3) at 50-nucleotide resolution.

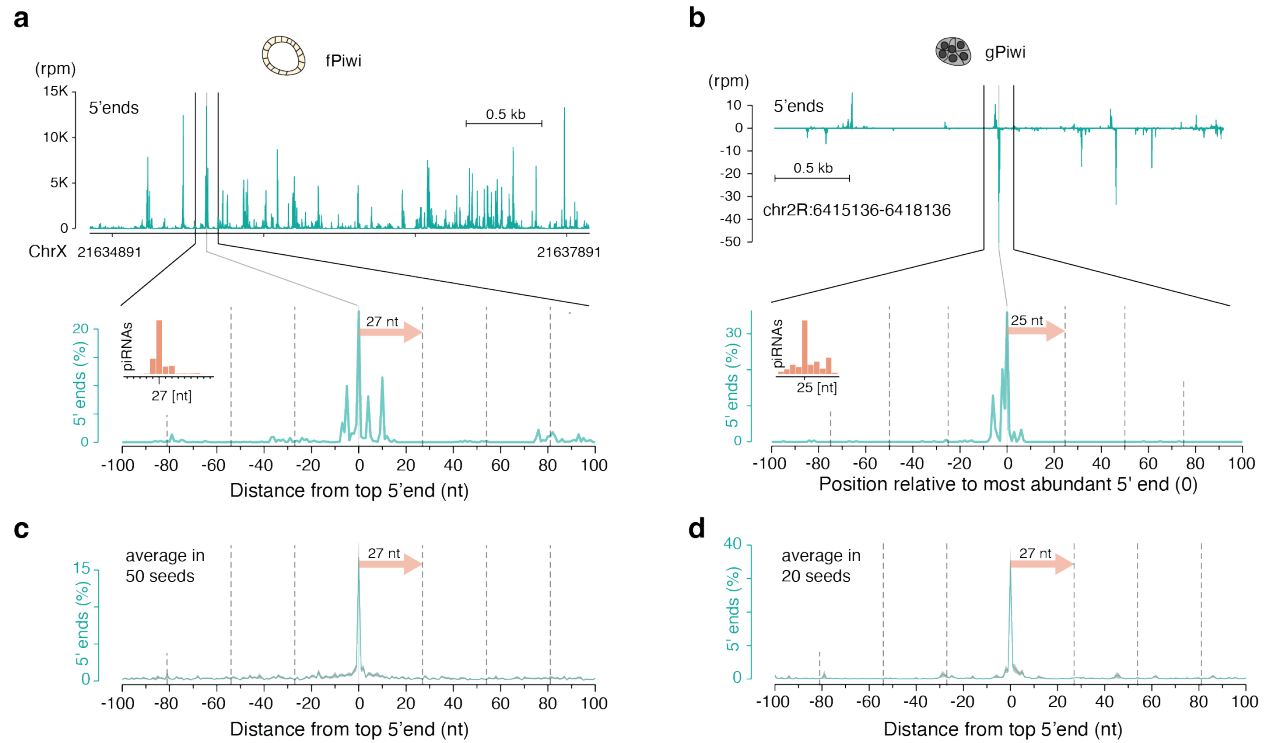
(b) Dispersion of fPiwi-piRNA coverage within *flamenco* 5' region (50-nucleotide bins), reported as coefficient of variation (CV; 0 = uniform coverage) and peak-to-average ratio (PAR; 1 = uniform coverage).

(c) Consistency of fPiwi-piRNA coverage across biological replicates (n = 3) reveals biological robustness and technical reproducibility. Spearman correlation ( $r_s$ ) at multiple resolutions (1 kb to single piRNA 5' ends).

(d) Germ cell (g)Piwi-piRNA coverage across representative *42AB* section (chr2R:6,414,700–6,424,700) at 50-nucleotide resolution.

(e) gPiwi-piRNA abundance within *42AB* 50-nucleotide bins. Coefficient of variation (CV; 0 = uniform coverage) and peak-to-average ratio (PAR; 1 = uniform coverage).

(f) Consistency of gPiwi-piRNA coverage across biological replicates (n = 3). Spearman correlation ( $r_s$ ) at multiple resolutions (1 kb to single piRNA 5' ends).



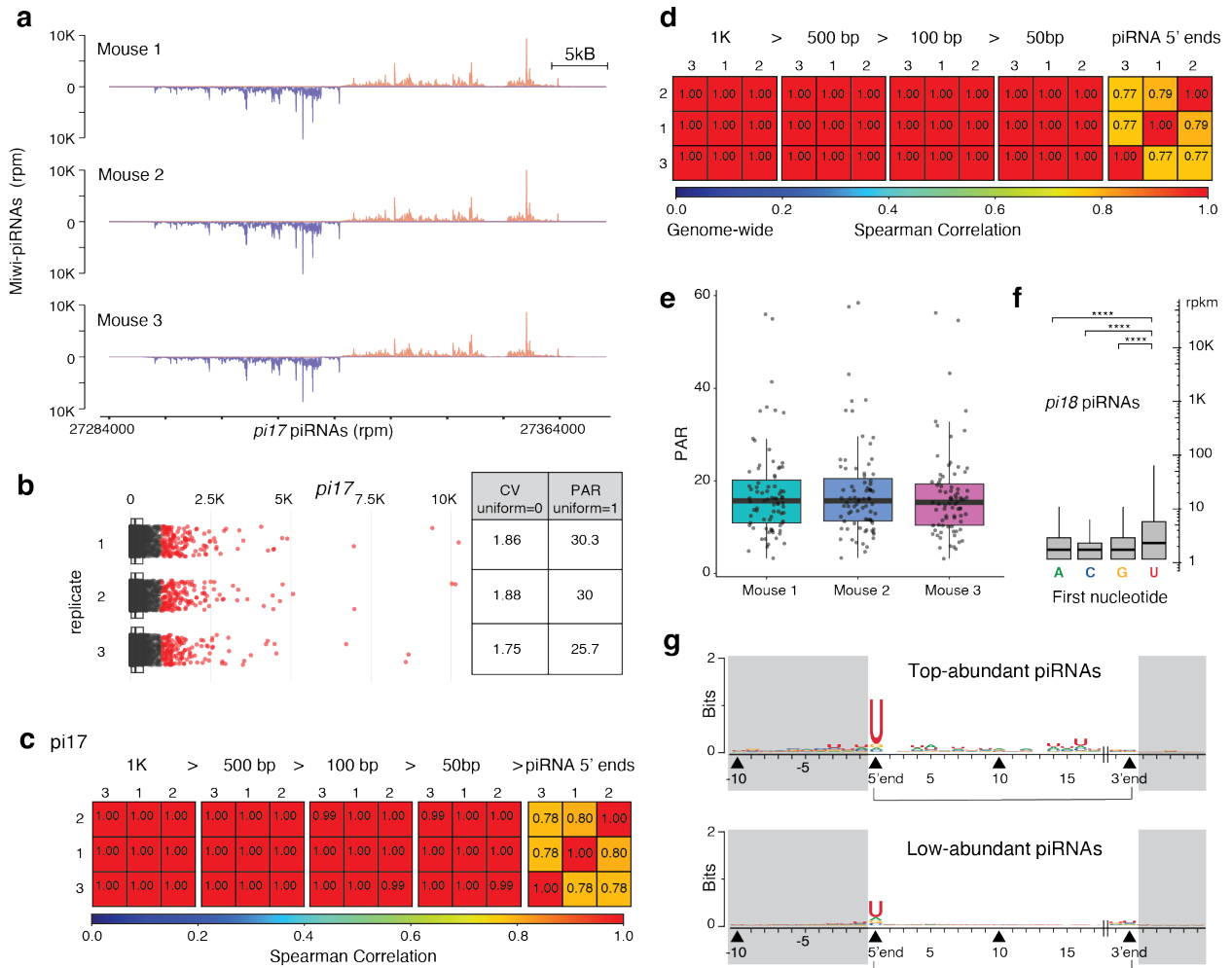
**Supplementary Fig. 6 | BFF-piRNA-seq reveals robust piRNA patterns with solitary processing hotspots in flies.**

(a) Distribution of fPiwi-piRNA 5' ends centered around the most productive processing site within the flamenco 5' region. Size distribution of piRNAs in the displayed area (inset). Orange arrow indicates most frequent piRNA length.

(b) Distribution of gPiwi-piRNAs and 5' ends across a representative 10-kb section of germline cluster 42AB (chr2R:6,414,700–6,424,700) at 50-nucleotide resolution ( $n = 3$ ). Below: gPiwi-piRNA 5' start sites centered around the most productive processing hotspot within 42AB. Size distribution of piRNAs in the displayed area (inset). Orange arrow indicates most frequent piRNA length.

(c) Average density of fPiwi-piRNA 5'-ends upstream and downstream of the 50 most productive processing sites within well-mappable piRNA-cluster regions (see Methods and Supplementary Data 4).

(d) Average density of gPiwi-piRNA 5' ends around the most productive processing sites across 20 well-mapping piRNA-cluster regions (see also Methods and Supplementary Data 5).



## Supplementary Fig. 7 | BFF-piRNA-seq reveals processing hotspots in mice.

(a) BFF-MIWI-piRNAs originating from the bidirectional pachytene piRNA cluster, pi17 (chr17:27,284,000–27,372,300) (resolution: 50 nt bins; n = 3 individuals).

(b) Dispersion of MIWI-piRNA coverage across pi17 (50-nucleotide bins), reported as coefficient of variation (CV; 0 = uniform coverage) and peak-to-average ratio (PAR; 1 = uniform coverage).

(c) Consistency of coverage within pi17 across three individual mice at multiple resolutions (1 kb to individual piRNA 5' start sites) (Spearman correlation  $r_s$ ; n = 3).

(d) Genome-wide reproducibility of MIWI-piRNA coverage across three individual mice at multiple resolutions (1 kb to individual piRNA 5' start sites; Spearman correlation  $r_s$ ; n = 3; top 200 pachytene piRNA clusters (Supplementary Data 2)).

(e) peak-to-average ratio (PAR) of MIWI-piRNA coverage (50-nt bins) within top-productive (90th percentile) pachytene piRNA clusters (n = 3).

(f) Individual piRNA abundance (>1 read) per 5' end nucleotide normalized to total piRNAs in pi18.

(g) Sequence logos for high-abundance (top 20%) versus low-abundance (bottom 20%) MIWI-piRNAs from the most productive pachytene piRNA cluster, pi18.

## Supplementary Protocol

### BFF-piRNA-seq: Step-by-Step Protocol

#### Materials

##### Enzymes and reagents

1. cOmplete, Mini, EDTA-free Protease Inhibitor Cocktail (Sigma-Aldrich, 04693159001)
2. T4 RNA ligase 2, truncated K227Q (NEB, M0351)
3. T4 RNA ligase 1 (NEB, M0437M; recommended). M0204 may also be used.
4. 10× RNA ligase buffer (NEB). The supplied buffer does not contain ATP, which is appropriate for 3' adapter ligation; ATP is added separately for 5' adapter ligation (see Section F).
5. SUPERase•In RNase Inhibitor (Invitrogen, AM2694), or another RNase inhibitor
6. SuperScript IV Reverse Transcriptase (Invitrogen, 18090010)
7. dNTP mix, 10 mM each (NEB, N0447)
8. Platinum Taq DNA polymerase (Invitrogen, 15966005)
9. Tris-HCl, pH 7.5
10. MgCl<sub>2</sub>
11. NaCl
12. EDTA
13. NP-40
14. DTT
15. PBS
16. Tween 20
17. TRIzol reagent

##### Magnetic beads

- M2 anti-FLAG magnetic beads (Millipore Sigma, M8823), for FLAG-tagged PIWI proteins.
- Protein A-coupled magnetic beads, when the anti-PIWI antibody is rabbit.
- Protein G-coupled magnetic beads, when the anti-PIWI antibody is mouse.

**IMPORTANT:** Use RNase- and DNase-free water and plastics (tips and microcentrifuge tubes) throughout the protocol.

##### Equipment

- Magnetic stand
- Tube rotator
- Thermocycler
- Heat block
- TapeStation or Bioanalyzer (or agarose gel electrophoresis apparatus)
- Pippin Prep (Sage Science) or another automated gel apparatus; a custom agarose gel can also be used (see Section I)

##### Kits and gels

1. Direct-zol RNA Miniprep kit (Zymo Research, R2051) plus TRIzol reagent if not included
2. Oligo Clean & Concentrator (Zymo Research, D4060)
3. DNA Clean & Concentrator-5 (Zymo Research, D4004)
4. 3% agarose Pippin Prep cassettes (Sage Science, CDP3010). Marker P accommodates five samples; marker C accommodates four and may perform slightly better for short fragments.
5. TapeStation D1000 tapes and reagents (or Bioanalyzer equivalent)

## Oligonucleotides and primers

### 1. Barcoded 3' adapter\*: /5rApp/NNTGACTGTGGAATTCTCGGGTGCCAAGG/3AmMO/

The 3' adapter is pre-adenylated at the 5' end and blocked at the 3' end with an amino modification to prevent self-ligation. It contains two UMI nucleotides at its 5' end; the remaining eight UMI nucleotides are supplied by the 5' adapter. The underlined sequence represents the barcode. 3' adapters harboring different in line barcodes can be used to pool different samples at the initial 3' ligation step.

### 2. 5' hybrid adapter (8N): /5AmMC6/GTTCAGAGTTCTACAGTCCGACGArUrCrNrNrNrNrNrNrN

The adapter is DNA for the first 24 nt and RNA for the final 10 nt and carries eight UMI nucleotides at its 3' end. Blocking the 5' end is optional because RNA-DNA ligation is inefficient; the RNA segment can be shortened to the final nucleotide.

### 3. RT primer: GCCTTGGCACCCGAGAATTCCA

The RT primer is also used in the first low-cycle PCR.

### 4. 5' short PCR primer: CTTCAGAGTTCTACAGTCCGACGA

The first C creates a single 5' mismatch that reduces use of the long-primer sequence as a template during the second low-cycle PCR (barcoding).

### 5. 5' long PCR primer: AATGATACGGCGACCACCGAGATCTACACGTTTCAGAGTTCTACAGTCCGA

The engineered mismatch is retained because the long-primer sequence is used by Illumina during sequencing and should not otherwise be altered.

### 6. 3' index primer\* (example RPI74-G2; 8-nt barcode):

CAAGCAGAAGACGGCATACTGAGATGCAATTCGGTGACTGGAGTTCCTTGGCACCCGAGAATTCCA

The underlined sequence represents the barcode.

\* Different 3' adapters and 3' index primers may be used with alternative barcodes.

| Oligo           | Identifier      | Sequence (how to order)                             |
|-----------------|-----------------|-----------------------------------------------------|
| 3' adapter      | 29.01           | /5rApp/NNTGACTGTGGAATTCTCGGGTGCCAAGG/3AmMO/         |
|                 | 29.02           | /5rApp/NNACACTCTGGAATTCTCGGGTGCCAAGG/3AmMO/         |
|                 | 29.03           | /5rApp/NNACAGAGTGGAATTCTCGGGTGCCAAGG/3AmMO/         |
|                 | 29.04           | /5rApp/NNGCGATATGGAATTCTCGGGTGCCAAGG/3AmMO/         |
| 5' adapter      | 5' adapter      | /5AmMC6/GTTCAGAGTTCTACAGTCCGACGArUrCrNrNrNrNrNrNrN  |
| RT primer       | RT primer       | GCCTTGGCACCCGAGAATTCCA                              |
| 5' short primer | 5' short primer | CTTCAGAGTTCTACAGTCCGACGA                            |
| 5' long primer  | 5' long primer  | AATGATACGGCGACCACCGAGATCTACACGTTTCAGAGTTCTACAGTCCGA |

| 3' index primer | Full-length sequence                                               |
|-----------------|--------------------------------------------------------------------|
| P1-A1           | CAAGCAGAAGACGGCATACTGAGATGTCGGTAAGTGACTGGAGTTCCTTGGCACCCGAGAATTCCA |

|         |                                                                        |
|---------|------------------------------------------------------------------------|
| P2-A2   | CAAGCAGAAGACGGCATAACGAGATAGGTCACTGTGACTGGAGTTCCTTGGCACCCGAGA<br>ATTCCA |
| P3-A3   | CAAGCAGAAGACGGCATAACGAGATGAATCCGAGTGACTGGAGTTCCTTGGCACCCGAGA<br>ATTCCA |
| P4-A4   | CAAGCAGAAGACGGCATAACGAGATGTACCTTGGTGACTGGAGTTCCTTGGCACCCGAGAA<br>TTCCA |
| P5-A5   | CAAGCAGAAGACGGCATAACGAGATCATGAGGAGTGACTGGAGTTCCTTGGCACCCGAGA<br>ATTCCA |
| P6-A6   | CAAGCAGAAGACGGCATAACGAGATTGACTGACGTGACTGGAGTTCCTTGGCACCCGAGA<br>ATTCCA |
| P7-A7   | CAAGCAGAAGACGGCATAACGAGATCGTATTCGGTGACTGGAGTTCCTTGGCACCCGAGAA<br>TTCCA |
| P8-A8   | CAAGCAGAAGACGGCATAACGAGATCTCCTAGAGTGACTGGAGTTCCTTGGCACCCGAGA<br>ATTCCA |
| P9-A9   | CAAGCAGAAGACGGCATAACGAGATTAGTTGCGGTGACTGGAGTTCCTTGGCACCCGAGA<br>ATTCCA |
| P10-A10 | CAAGCAGAAGACGGCATAACGAGATGAGATACGGTGACTGGAGTTCCTTGGCACCCGAGA<br>ATTCCA |

*O/N* = overnight.

### Buffers

IP buffer: 20 mM Tris, pH 7.5, 150 mM NaCl, 2 mM EDTA, 1% (v/v) NP-40.

Lysis buffer: IP buffer + 0.5 mM DTT (add fresh) + cOmplete protease inhibitor cocktail (add fresh); prepare immediately before use and keep on ice.

Ligation buffer: 50 mM Tris-HCl, pH 7.5, 10 mM MgCl<sub>2</sub>.

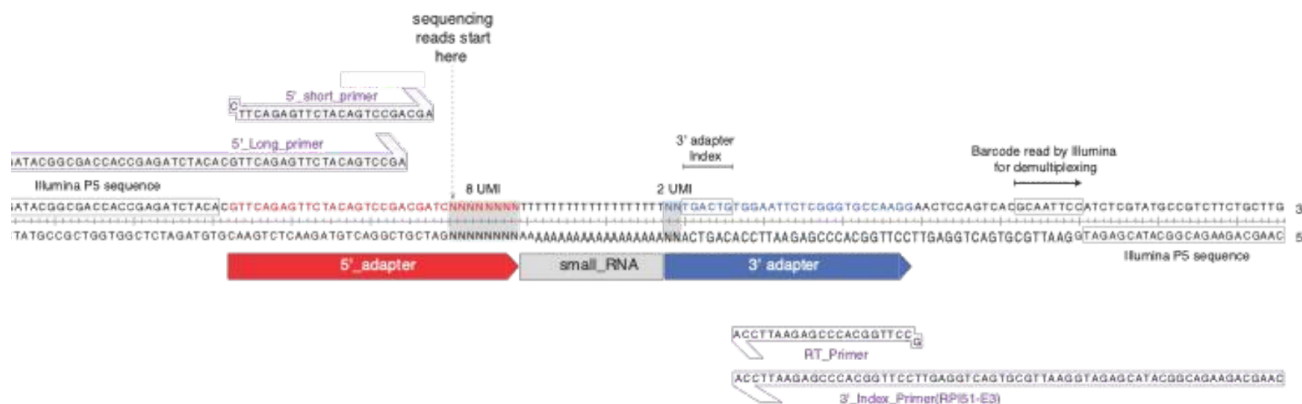


**Figure 1. Final BFF adapter and amplification design.** *The schematic shows the 5' and 3' adapters, primers used for the first low-cycle PCR and barcoding PCR, UMIs, and internal barcodes. A 19-nt poly-U insert is shown as a stand-in for piRNAs.*

### Step-by-step protocol

**WORKFLOW:** Use either Section A (cells) or Section B (gonad tissue), then continue with Section C.

### A. Cell lysis

- Resuspend the cell pellet in 200  $\mu$ L lysis buffer (IP buffer + DTT + protease inhibitor) and incubate on ice for 10 min.
- Add IP buffer to a final volume of 1 mL. For smaller IP volumes, use appropriately sized tubes to maximize bead-lysate contact and improve capture efficiency.
- Centrifuge at  $15,000 \times g$  for 15 min at  $4^{\circ}\text{C}$  and transfer the supernatant to a clean tube.

**NOTE:** For very low input (few cells or a very small pellet), centrifugation can be omitted; proceed directly to immunoprecipitation after lysis.

### B. Gonad tissue lysis (ovaries or decapsulated testes)

- Add 200  $\mu$ L, or at least three tissue volumes, of lysis buffer (IP buffer + freshly added 0.5 mM DTT + fresh protease inhibitors). Homogenize with a pestle or handheld homogenizer, returning the sample to ice intermittently, until the tissue is thoroughly disrupted.
- Add IP buffer to a final volume of 1 mL.
- Clarify the lysate by centrifugation at  $15,000 \times g$  for 15 min at  $4^{\circ}\text{C}$ . Recover the supernatant while avoiding any floating fat layer.

### C. Immunoprecipitation (IP)

For endogenous PIWI proteins, bind the antibody to Protein A/G magnetic beads according to the manufacturer's instructions, then wash the antibody-coupled beads twice with IP buffer before immunoprecipitation.

#### Example conditions used for mouse piRNAs with rabbit anti-MIWI antibody (Cell Signalling, 2079):

- Use 40  $\mu$ L Protein A Dynabeads per lysate (120  $\mu$ L total for three replicates).
- Incubate the beads with 300  $\mu$ L antibody solution in PBS-T (6  $\mu$ L anti-MIWI in 300  $\mu$ L PBS-T; 1:50) with rotation for at least 10 min at room temperature.
- Immobilize the beads on a magnet, remove the supernatant, and wash once with 1 mL PBS-T.
- Wash twice with 1 mL IP buffer to equilibrate the beads.
- Resuspend the beads in one bead volume of IP buffer (120  $\mu$ L total).
- Add 40  $\mu$ L resuspended beads to each lysate.
- Incubate overnight at 4°C with rotation.

**NOTE:** For FLAG-tagged PIWI, wash anti-FLAG beads twice with 1 mL IP buffer, add them directly to the lysate, and immunoprecipitate for 2 h at 4°C.

- Wash the beads three times with 1 mL IP buffer.
- Wash the beads three times with 1 mL wash buffer. Do not omit these washes.
- During the final wash, transfer the bead-bound PIWI-piRNA complexes to a clean tube for ligation.

### D. On-PIWI 3' adapter ligation

Prepare the following 50- $\mu$ L on-PIWI 3' adapter ligation mixture:

| Component                             | Volume      |
|---------------------------------------|-------------|
| 3' adapter, 10 $\mu$ M                | 2 $\mu$ L   |
| 10 $\times$ RNA ligase buffer, no ATP | 5 $\mu$ L   |
| 50% aqueous PEG-8000                  | 15 $\mu$ L  |
| T4 Rnl2 truncated K227Q ligase        | 2.5 $\mu$ L |
| SUPERase•In, 40 U/ $\mu$ L            | 1.5 $\mu$ L |
| H <sub>2</sub> O                      | 24 $\mu$ L  |

**IMPORTANT:** T4 Rnl2 truncated K227Q ligates the pre-adenylated 5' end of the adapter to the 3'-OH of RNA in the absence of ATP. Do not add ATP at this step; this minimizes ATP-dependent circularization and multimerization of 5'-phosphorylated RNAs.

- Capture the beads on a magnet and remove all wash buffer.
- Briefly spin the tube at low speed, return it to the magnet, and remove any residual liquid.
- Add 50  $\mu$ L ligation mixture to the beads and gently resuspend.
- Incubate overnight at 4°C with gentle rotation or shaking sufficient to keep the beads suspended.
- Wash the beads three times with 1 mL wash buffer.
- Transfer the beads to a clean tube during the final wash.

### E. Recovery of 3'-ligated RNA with TRIzol

- Capture the beads on a magnet and remove all wash buffer; a brief low-speed spin can help remove residual liquid.

- Add 400  $\mu\text{L}$  TRIzol and rotate at room temperature for 10 min.
- Place the tube on a magnetic rack and transfer the supernatant containing the ligation products to a clean 1.5-mL tube.
- Extract the ligation products with the Direct-zol RNA Miniprep kit according to the manufacturer's instructions and elute in 50  $\mu\text{L}$   $\text{H}_2\text{O}$ .
- Concentrate with the Oligo Clean & Concentrator kit and elute in 6  $\mu\text{L}$   $\text{H}_2\text{O}$ .

## F. 5' adapter ligation

Prepare the following 5' adapter ligation mixture:

| Component                             | Volume          |
|---------------------------------------|-----------------|
| 5' adapter (8N), 10 $\mu\text{M}$     | 1 $\mu\text{L}$ |
| 10 $\times$ RNA ligase buffer, no ATP | 2 $\mu\text{L}$ |
| 50% aqueous PEG-8000                  | 6 $\mu\text{L}$ |
| DMSO                                  | 2 $\mu\text{L}$ |
| ATP, 20 mM                            | 1 $\mu\text{L}$ |

**IMPORTANT:** ATP is required by T4 RNA ligase 1 for the 5' adapter ligation step.

- Add the ligation mixture to the RNA sample and mix gently.
- Denature at 90°C for 1 min.
- Immediately cool on ice for 2 min.
- Add 2  $\mu\text{L}$  T4 Rnl1 and 0.5  $\mu\text{L}$  SUPERase•In; mix gently.
- Incubate for 2 h at 25°C or 16 h at 16°C.
- Add 30  $\mu\text{L}$   $\text{H}_2\text{O}$ .

## Purify with the Oligo Clean & Concentrator kit using the following modification:

- Add 100  $\mu\text{L}$  Oligo Binding Buffer to the sample.
- Add 200  $\mu\text{L}$  ethanol and mix thoroughly by pipetting.
- Transfer to a Zymo-Spin IC column and centrifuge; discard the flow-through.
- Add 750  $\mu\text{L}$  DNA Wash Buffer and centrifuge at  $16,000 \times g$  for 1 min.
- Transfer the column to a clean microcentrifuge tube and wait 1 min. Add 11.5  $\mu\text{L}$   $\text{H}_2\text{O}$  directly to the membrane and centrifuge at  $12,000 \times g$  for 30 s to elute.

**IMPORTANT:** The manufacturer's standard protocol uses 400  $\mu\text{L}$  ethanol. The reduced 200- $\mu\text{L}$  volume enriches against shorter nonspecific products while retaining ligated piRNAs; the manufacturer recommends this strategy when retaining products  $\geq 80$  nt.

## G. Reverse transcription (cDNA synthesis)

Transfer the ligated RNA to a PCR tube or strip and anneal the RT primer:

| Component                   | Volume          |
|-----------------------------|-----------------|
| RT primer, 50 $\mu\text{M}$ | 1 $\mu\text{L}$ |
| dNTP mix, 10 mM each        | 1 $\mu\text{L}$ |

- Mix and briefly centrifuge.
- Heat at 65°C for 5 min, then place on ice or at 4°C for at least 1 min.

**During the annealing step, prepare the RT reaction mix:**

| Component                                      | Volume |
|------------------------------------------------|--------|
| 5× SSIV buffer                                 | 4 µL   |
| 100 mM DTT                                     | 1 µL   |
| SUPERase•In                                    | 1 µL   |
| SuperScript IV Reverse Transcriptase, 200 U/µL | 1 µL   |

- Mix and briefly centrifuge the RT reaction mix.
- Add the RT reaction mix to the annealed RNA and incubate at 56°C for 30 min.
- Heat-inactivate at 80°C for 10 min.
- Add 60 µL H<sub>2</sub>O.

#### **H. First low-cycle PCR (second-strand synthesis and short-primer amplification)**

Prepare the following 400-µL PCR master mix and aliquot into PCR tubes (preferably 50 µL per tube):

| Component                      | Volume   |
|--------------------------------|----------|
| dNTP mix, 10 mM each           | 10 µL    |
| MgCl <sub>2</sub> , 50 mM      | 12 µL    |
| 10× PCR buffer                 | 40 µL    |
| 5' short PCR primer, 100 µM    | 2 µL     |
| RT primer, 100 µM              | 2 µL     |
| Platinum Taq DNA polymerase    | 1.6 µL   |
| cDNA (half of the RT reaction) | 40 µL    |
| H <sub>2</sub> O               | 292.4 µL |

*MgCl<sub>2</sub> and PCR buffer are supplied with Platinum Taq DNA polymerase.*

| Step                 | Temperature | Time  | Cycles |
|----------------------|-------------|-------|--------|
| Initial denaturation | 94°C        | 2 min | 1      |
| Denaturation         | 94°C        | 30 s  | 6      |
| Annealing            | 60°C        | 30 s  | 6      |
| Extension            | 72°C        | 15 s  | 6      |

|                 |      |       |   |
|-----------------|------|-------|---|
| Final extension | 72°C | 1 min | 1 |
|-----------------|------|-------|---|

- Clean and concentrate the PCR product with DNA Clean & Concentrator-5 (or equivalent) and elute in 31  $\mu\text{L}$   $\text{H}_2\text{O}$ .

**NOTE:** Short-primer second-strand synthesis and limited amplification improve separation of ligated piRNA products from adapter-adapter byproducts on agarose gels, facilitating byproduct removal before barcoding.

## I. Removal of adapter-adapter byproducts

- Run a 3% agarose Pippin Prep and select fragments from 75-150 nt according to the manufacturer's instructions.
- Recover the full elution volume (~40  $\mu\text{L}$ ) and, if necessary, adjust the recovered volume to at least 40  $\mu\text{L}$ .

**NOTE:** A custom agarose gel may be used instead, provided the product and adapter-adapter byproducts are clearly separated. We have observed lower recovery after manual gel extraction.

*Optional: use DNA Clean & Concentrator-5 (Zymo) to exchange Pippin Prep buffer for  $\text{H}_2\text{O}$  and elute in 40  $\mu\text{L}$   $\text{H}_2\text{O}$ .*

## J. Pilot PCR

**NOTE:** Perform a pilot PCR before library barcoding to determine the minimum cycle number required. This minimizes overamplification and increases the fraction of sequencing reads that represent independent ligation events.

| Component                                     | Volume             |
|-----------------------------------------------|--------------------|
| dNTP mix, 10 mM each                          | 2.5 $\mu\text{L}$  |
| $\text{MgCl}_2$ , 50 mM                       | 3 $\mu\text{L}$    |
| 10 $\times$ PCR buffer                        | 10 $\mu\text{L}$   |
| 5' long PCR primer, 100 $\mu\text{M}$         | 0.5 $\mu\text{L}$  |
| 3' index primer, 100 $\mu\text{M}$            | 0.5 $\mu\text{L}$  |
| Platinum Taq DNA polymerase                   | 0.4 $\mu\text{L}$  |
| First PCR product (1/4 of Pippin Prep output) | 10 $\mu\text{L}$   |
| $\text{H}_2\text{O}$                          | 73.1 $\mu\text{L}$ |

| Step                 | Temperature | Time  | Cycles |
|----------------------|-------------|-------|--------|
| Initial denaturation | 94°C        | 2 min | 1      |
| Denaturation         | 94°C        | 30 s  | 14     |
| Annealing            | 60°C        | 30 s  | 14     |

|                 |      |       |    |
|-----------------|------|-------|----|
| Extension       | 72°C | 15 s  | 14 |
| Final extension | 72°C | 1 min | 1  |

- Remove 5 µL after cycles 6, 8, 10, 12, and 14.
- Analyze the aliquots on a TapeStation D1000 or Bioanalyzer. Alternatively, remove 15 µL and run on a 2.5% agarose gel; the optimal cycle is the first cycle at which the desired library product becomes visible.

**NOTE:** Check the sequencing core's required submission concentration (the source protocol uses ~3 nM). To choose the barcoding PCR cycle number, identify the pilot-PCR cycle at which the library reaches ~3-5 nM and subtract three cycles (minimum five cycles). Example: ~3.5 nM after 10 pilot-PCR cycles → use 7 cycles for barcoding PCR.

### K. Second low-cycle PCR: barcoding

**ATTENTION:** Use a different 3' index primer for each sample to enable multiplexing and pooling before sequencing.

| Component                                     | Volume   |
|-----------------------------------------------|----------|
| dNTP mix, 10 mM each                          | 7.5 µL   |
| MgCl <sub>2</sub> , 50 mM                     | 9 µL     |
| 10× PCR buffer                                | 30 µL    |
| 5' long PCR primer, 100 µM                    | 1.5 µL   |
| 3' index primer, 100 µM                       | 1.5 µL   |
| Platinum Taq DNA polymerase                   | 1.2 µL   |
| First PCR product (3/4 of Pippin Prep output) | 30 µL    |
| H <sub>2</sub> O                              | 219.3 µL |

| Step                 | Temperature | Time  | Cycles                                 |
|----------------------|-------------|-------|----------------------------------------|
| Initial denaturation | 94°C        | 2 min | 1                                      |
| Denaturation         | 94°C        | 30 s  | Optimal number determined by pilot PCR |
| Annealing            | 60°C        | 30 s  | Same                                   |
| Extension            | 72°C        | 15 s  | Same                                   |
| Final extension      | 72°C        | 1 min | 1                                      |

- Clean and concentrate with DNA Clean & Concentrator-5; elute in ~30 µL H<sub>2</sub>O (approximately 10-fold concentration).

- Analyze and quantify the library with a TapeStation or Bioanalyzer.
- If adapter-adapter byproducts remain, perform size selection with the ProNex Size-Selective Purification System (Promega, NG2001). For abundant byproducts, use a 1.7× ProNex chemistry:sample ratio. Adjust the ratio to the library length distribution according to the manufacturer's instructions, and consider eluting in a smaller volume to offset concentration loss.

**ATTENTION:** Perform the bead cleanup only when substantial adapter-adapter (linker-linker) byproducts and/or residual primers remain.

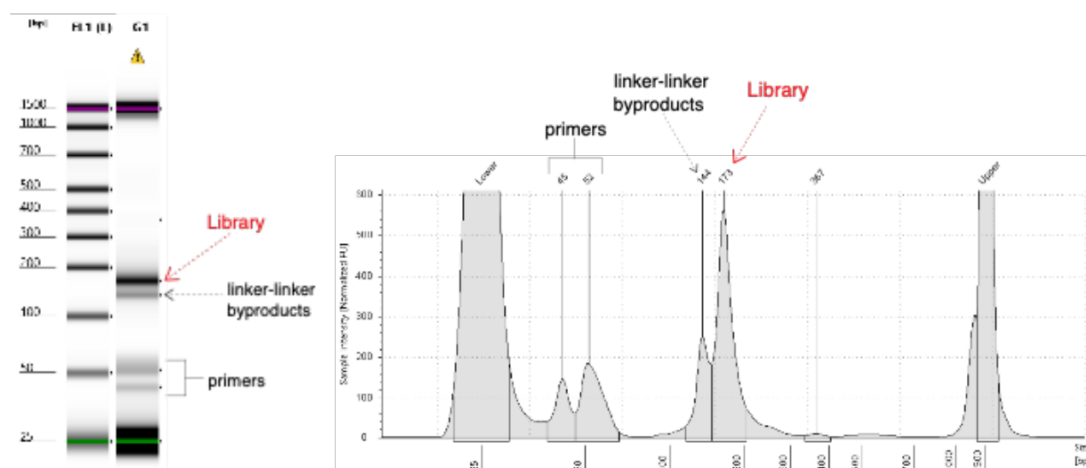


**Figure 2. TapeStation analysis before bead-based cleanup.** Left, electronic agarose-gel image. Right, TapeStation electropherogram. “Linker” denotes adapter-derived products; “library” denotes successfully ligated piRNA library molecules.

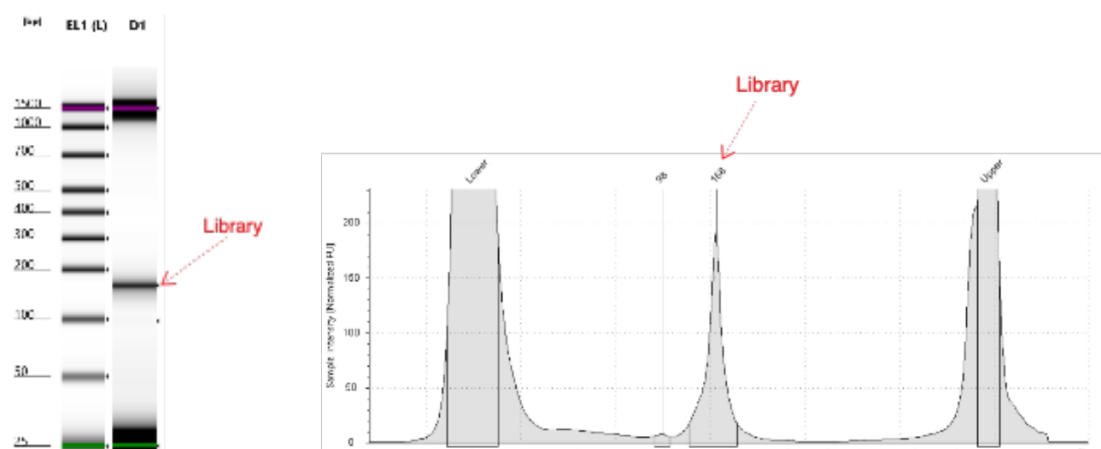


**Figure 3. TapeStation analysis after bead-based cleanup.** Left, electronic agarose-gel image. Right, TapeStation electropherogram. “Library” denotes the successfully ligated piRNA library.

# 'BFF-piRNA-seq reveals conserved features of primary piRNA processing'

Parthena Konstantinidou, Franziska Ahrend, Brian G. Galletta, Angel Jaimes, Dimitrios G. Anastasakis, Nasser M. Rusan, Markus Hafner, Astrid D. Haase

This vignette describes the computational methods associated with this manuscript. The analysis in this vignette does not include all of the figures created in the manuscript, but contains representative data to demonstrate the materials and methods. Files used in this analysis were preprocessed according to the methods described in the manuscript.

Code by Parthena Konstantinidou and Franziska Ahrend

## BFF supplementary code

### load R packages

```
# Load R packages (install the packages first if you don't have them):

library(data.table)
library(ggplot2)
library(ggpubr)
library(patchwork)
library(rtracklayer)
library(GenomeInfoDb)
library(GenomicRanges)
library(Biostrings)
```

### 3' adapter ligation controls (adult Miwi-piRNAs)

#### Import data

```
# Import fasta files of mapped reads with their UMIs

library(Biostrings)

# On-PIWI samples
On1_withUMIs<-readDNAStringSet(On1_withUMIs_path)
On2_withUMIs<-readDNAStringSet(On2_withUMIs_path)
On3_withUMIs<-readDNAStringSet(On3_withUMIs_path)
# In-solution controls
Off1_withUMIs<-readDNAStringSet(Off1_withUMIs_path)
Off2_withUMIs<-readDNAStringSet(Off2_withUMIs_path)
Off3_withUMIs<-readDNAStringSet(Off3_withUMIs_path)
```

### piRNA length distributions on-PIWI VS In-solution 3'ligation (Fig.S1b)

```
seq_list<-list("On1_withUMIs"=On1_withUMIs,
               "On2_withUMIs"=On2_withUMIs,
               "On3_withUMIs"=On3_withUMIs,
               "Off1_withUMIs"=Off1_withUMIs,
```

```

      "Off2_withUMIs"=Off2_withUMIs,
      "Off3_withUMIs"=Off3_withUMIs)

ligation_widths<-rbindlist(lapply(names(seq_list), function(i){
  DT<-data.table("sample"=i, "width"=width(seq_list[[i]]))
  # Remove UMIs from width
  DT$real_width<-DT$width-10
  return(DT)
}))

ligation_widths[, experiment := fcase(
  grepl("^On", sample), "On-PIWI",
  default = "In-solution"
)]

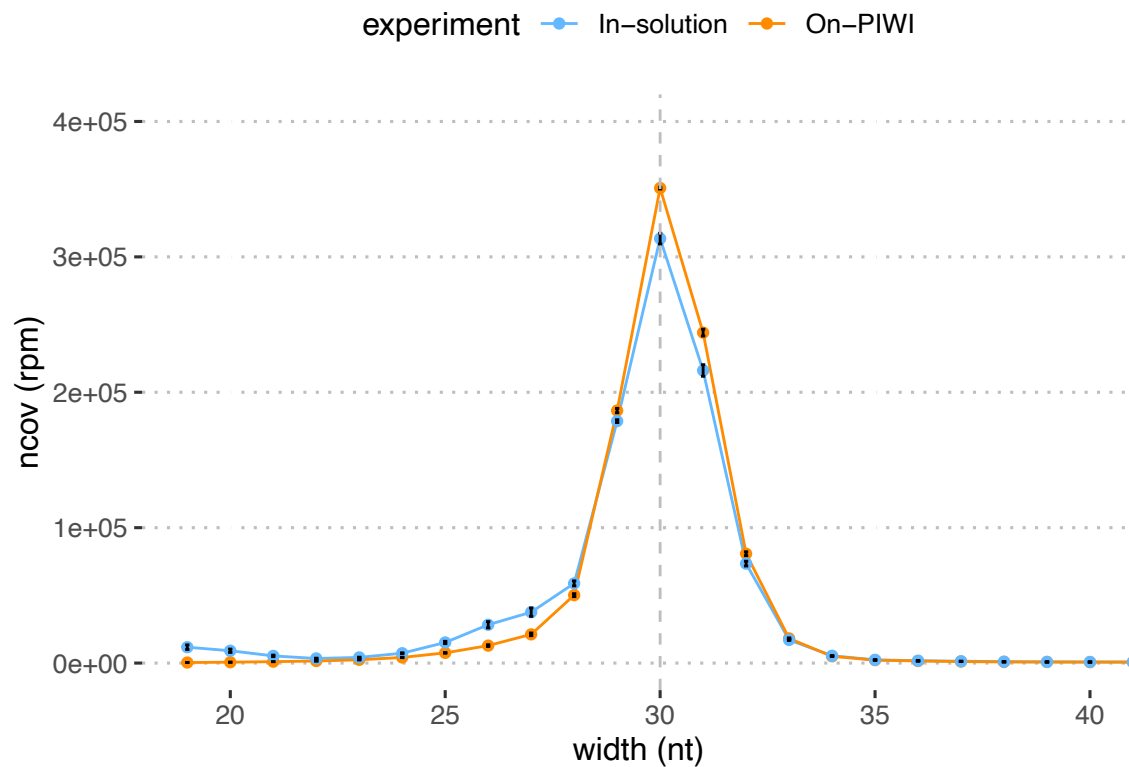
ligation_widths[, "reads" := .N, by=c("real_width", "sample")]
ligation_widths[, "total" := .N, by="sample"]
ligation_widths<-unique(ligation_widths[,c("experiment", "sample", "real_width",
      "reads", "total")])
ligation_widths$rpm<-(ligation_widths$reads/ligation_widths$total)*1000000

# Average piRNA reads per length with error bars
ligation_widths_2.dt<-ligation_widths
ligation_widths_2.dt[, "mean" := mean(rpm), by=c("experiment", "real_width")]
ligation_widths_2.dt[, "SD" := sd(rpm), by=c("experiment", "real_width")]
ligation_widths_2.dt[, "SE" := SD/length(rpm), by=c("experiment", "real_width")]
ligation_widths_3.dt<-unique(ligation_widths_2.dt[,c("experiment", "real_width",
      "mean", "SD", "SE")])

colors <- c("On-PIWI" = "darkorange", "In-solution" = "steelblue1")

## Plot average piRNA lengths with error bars
ggplot() + theme_pubclean() +
  geom_line(data = ligation_widths_3.dt,
    aes(x = real_width, y = mean, color=experiment)) +
  geom_point(data = ligation_widths_3.dt,
    aes(x = real_width, y = mean, color=experiment), size=1.5) +
  geom_errorbar(data = ligation_widths_3.dt,
    aes(x = real_width, ymin=mean-SE, ymax=mean+SE), width=.1) +
  geom_vline(xintercept =30, color="grey", linetype=2)+
  scale_x_continuous(breaks = c(20, 25, 30, 35, 40))+
  theme(aspect.ratio = 0.6,
    plot.title = element_text(hjust=0.5)) +
  scale_color_manual("experiment", values = colors)+
  labs(colour= "experiment") +
  xlab("width (nt)") +
  ylab("ncov (rpm)") +
  coord_cartesian(xlim = c(19, 40), ylim = c(0, 400000), expand = TRUE)

```



End nucleotide preferences during on-PIWI VS In-solution 3'ligation (Fig.1b and Fig.S1c)

```
seq_list2<-lapply(names(seq_list), function(i){
  clean<-DNASTringSet(gsub("--$", "", as.character(seq_list[[i]])))
  clean.df<-as.data.frame(clean)
  # Replace T with U (turn to RNA)
  clean.df$x<-gsub(pattern = "T", replacement = "U", x = clean.df$x)
  return(clean.df)
})

names(seq_list2)<-names(seq_list)
```

```

# Retrieve 3' end (last 2 nt of piRNA and the 2 UMIs at 3'end)
last4.df<-lapply(names(seq_list2), function(i){
  nucleotides<-4
  df<-seq_list2[[i]]
  last4.df<-substr(df$x, width(df$x)-(nucleotides-1), width(df$x))
  return(last4.df)
})

names(last4.df)<-names(seq_list2)

# Retrieve 5' end (last 2 UMIs of the 5'adapter and the first 2 nt of piRNA)
nt_7to10.df<-lapply(names(seq_list2), function(i){
  df<-seq_list2[[i]]
  df_7to10<- substr(df$x, 7, 10)
  return(df_7to10)
})

names(nt_7to10.df)<-names(seq_list2)

# Plot sequence logos:
library(ggseqlogo)

p1_on1 = ggplot() + geom_logo(last4.df$On1_withUMIs, seq_type='RNA') +
  theme_logo() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),
    axis.text.x = element_text(size = 10)) +
  scale_x_continuous(breaks = 1:4, labels = c("y", "z", "+1", "+2"))

p2_on1 = ggplot() + geom_logo(nt_7to10.df$On1_withUMIs, seq_type='RNA') +
  theme_logo() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),
    axis.text.x = element_text(size = 10)) +
  scale_x_continuous(breaks = 1:4, labels = c("-2", "-1", "1", "2"))

p1_on2 = ggplot() + geom_logo(last4.df$On2_withUMIs, seq_type='RNA') +
  theme_logo() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),
    axis.text.x = element_text(size = 10)) +
  scale_x_continuous(breaks = 1:4, labels = c("y", "z", "+1", "+2"))

p2_on2 = ggplot() + geom_logo(nt_7to10.df$On2_withUMIs, seq_type='RNA') +
  theme_logo() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),
    axis.text.x = element_text(size = 10)) +
  scale_x_continuous(breaks = 1:4, labels = c("-2", "-1", "1", "2"))

```

```

p1_on3 = ggplot() + geom_log(last4.df$On3_withUMIs, seq_type='RNA') +
  theme_logo() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),
    axis.text.x = element_text(size = 10)) +
  scale_x_continuous(breaks = 1:4, labels = c("y", "z", "+1", "+2"))

p2_on3 = ggplot() + geom_log(nt_7to10.df$On3_withUMIs, seq_type='RNA') +
  theme_logo() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),
    axis.text.x = element_text(size = 10)) +
  scale_x_continuous(breaks = 1:4, labels = c("-2", "-1", "1", "2"))

p1_off1 = ggplot() + geom_log(last4.df$Off1_withUMIs, seq_type='RNA') +
  theme_logo() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),
    axis.text.x = element_text(size = 10)) +
  scale_x_continuous(breaks = 1:4, labels = c("y", "z", "+1", "+2"))

p2_off1 = ggplot() + geom_log(nt_7to10.df$Off1_withUMIs, seq_type='RNA') +
  theme_logo() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),
    axis.text.x = element_text(size = 10)) +
  scale_x_continuous(breaks = 1:4, labels = c("-2", "-1", "1", "2"))

p1_off2 = ggplot() + geom_log(last4.df$Off2_withUMIs, seq_type='RNA') +
  theme_logo() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),
    axis.text.x = element_text(size = 10)) +
  scale_x_continuous(breaks = 1:4, labels = c("y", "z", "+1", "+2"))

p2_off2 = ggplot() + geom_log(nt_7to10.df$Off2_withUMIs, seq_type='RNA') +
  theme_logo() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),
    axis.text.x = element_text(size = 10)) +
  scale_x_continuous(breaks = 1:4, labels = c("-2", "-1", "1", "2"))

p1_off3 = ggplot() + geom_log(last4.df$Off3_withUMIs, seq_type='RNA') +
  theme_logo() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),

```

```

    axis.text.x = element_text(size = 10)) +
    scale_x_continuous(breaks = 1:4, labels = c("y", "z", "+1", "+2"))

p2_off3 = ggplot() + geom_logos(nt_7to10.df$Off3_withUMIs, seq_type='RNA') +
  theme_logos() +
  ylim(0,2) +
  theme(
    axis.text.y = element_text(size = 8),
    axis.text.x = element_text(size = 10)) +
    scale_x_continuous(breaks = 1:4, labels = c("-2", "-1", "1", "2"))

library(patchwork)

Ligation_ctrl_Logos<-wrap_plots(
  (p2_off1 + ggtitle('In-solution')) / p2_off2 / p2_off3,
  p1_off1 / p1_off2 / p1_off3,
  (p2_on1 + ggtitle('On-PIWI')) / p2_on2 / p2_on3,
  p1_on1 / p1_on2 / p1_on3,
  ncol = 4
)

# Get frequencies of 3' most nucleotide for different 5'most UMI in the 3' adaptor
frequency_of3end_by_UMI<-lapply(names(last4.df), function(i){
  last4<- data.table("seq"=last4.df[[i]])

  pos3_U <- last4[substr(last4$seq, 3, 3) == "U", ]
  pos3_C <- last4[substr(last4$seq, 3, 3) == "C", ]
  pos3_G <- last4[substr(last4$seq, 3, 3) == "G", ]
  pos3_A <- last4[substr(last4$seq, 3, 3) == "A", ]

  get_pos2_freq <- function(subset) {
    pos2 <- substr(subset$seq, 2, 2)
    # Filter out N's and other non-standard nucleotides
    pos2_valid <- pos2[pos2 %in% c("A", "U", "C", "G")]
    freq <- table(pos2_valid) / length(pos2_valid)
    return(freq)
  }

  # Calculate frequencies for each group
  freq_when_U <- get_pos2_freq(pos3_U)
  freq_when_C <- get_pos2_freq(pos3_C)
  freq_when_G <- get_pos2_freq(pos3_G)
  freq_when_A <- get_pos2_freq(pos3_A)

  # Combine results into data frame
  results <- data.frame(
    pos3 = c("U", "C", "G", "A"),
    freq_A = c(freq_when_U["A"], freq_when_C["A"], freq_when_G["A"], freq_when_A["A"]),

```

```

    freq_U = c(freq_when_U["U"], freq_when_C["U"], freq_when_G["U"], freq_when_A["U"]),
    freq_C = c(freq_when_U["C"], freq_when_C["C"], freq_when_G["C"], freq_when_A["C"]),
    freq_G = c(freq_when_U["G"], freq_when_C["G"], freq_when_G["G"], freq_when_A["G"])
  )

  return(results)
})

names(frequency_of3end_by_UMI)<-names(last4.df)

# Reshape the data for ggplot
library(tidyr)
library(dplyr)

plot_data3.L<-lapply(names(frequency_of3end_by_UMI), function(i){
  plot_data <- frequency_of3end_by_UMI[[i]] %>%
    gather(key = "nucleotide", value = "frequency", -pos3) %>%
    mutate(nucleotide = substr(nucleotide, 6, 6),
           nucleotide = factor(nucleotide, levels = c("A", "G", "C", "U")))
  return(plot_data)
})

names(plot_data3.L)<-names(frequency_of3end_by_UMI)

# Plot frequencies of nt in 3' end for different 3' adaptor starts (1st UMI of 3' ad)

plot_nucleotide_freq <- function(data) {

  # Define nucleotide colors
  nuc_colors <- c("A" = "green4", "G" = "goldenrod1",
                  "C" = "dodgerblue3", "U" = "firebrick3")

  ggplot(data, aes(x = frequency, y = pos3, fill = nucleotide)) +
    geom_bar(stat = "identity", position = "stack") +
    scale_fill_manual(values = nuc_colors) +
    theme_minimal() +
    labs(x= element_blank(),
         y = "nt +1",
         fill = "Nucleotide in Position 2") +
    theme(
      axis.text.y = element_text(color = nuc_colors, size = 8),
      aspect.ratio = 0.8,
      legend.position = "none"
    ) +
    scale_y_discrete(position = "right") +
    scale_x_continuous(breaks = c(0, 1, 0.5))
}

Off1_3end_freq<-plot_nucleotide_freq(plot_data3.L$Off1_withUMIs)
Off2_3end_freq<-plot_nucleotide_freq(plot_data3.L$Off2_withUMIs)
Off3_3end_freq<-plot_nucleotide_freq(plot_data3.L$Off3_withUMIs)
On1_3end_freq<-plot_nucleotide_freq(plot_data3.L$On1_withUMIs)
On2_3end_freq<-plot_nucleotide_freq(plot_data3.L$On2_withUMIs)

```

```

On3_3end_freq<-plot_nucleotide_freq(plot_data3.L$On3_withUMIs)

# Get frequencies of 5' most nucleotide for different 3'most UMI in the 5' adaptor
frequency_of5end_by_UMI<-lapply(names(nt_7to10.df), function(i){

first4<- data.table("seq"=nt_7to10.df[[i]])

pos2_U <- first4[substr(first4$seq, 2, 2) == "U", ]
pos2_C <- first4[substr(first4$seq, 2, 2) == "C", ]
pos2_G <- first4[substr(first4$seq, 2, 2) == "G", ]
pos2_A <- first4[substr(first4$seq, 2, 2) == "A", ]

get_pos3_freq <- function(subset) {
  pos3 <- substr(subset$seq, 3, 3)
  # Filter out N's and other non-standard nucleotides
  pos3_valid <- pos3[pos3 %in% c("A", "U", "C", "G")]
  freq <- table(pos3_valid) / length(pos3_valid)
  return(freq)
}

# Calculate frequencies for each group
freq_when_U <- get_pos3_freq(pos2_U)
freq_when_C <- get_pos3_freq(pos2_C)
freq_when_G <- get_pos3_freq(pos2_G)
freq_when_A <- get_pos3_freq(pos2_A)

# Combine results into a data frame
results <- data.frame(
  pos2 = c("U", "C", "G", "A"),
  freq_A = c(freq_when_U["A"], freq_when_C["A"], freq_when_G["A"], freq_when_A["A"]),
  freq_U = c(freq_when_U["U"], freq_when_C["U"], freq_when_G["U"], freq_when_A["U"]),
  freq_C = c(freq_when_U["C"], freq_when_C["C"], freq_when_G["C"], freq_when_A["C"]),
  freq_G = c(freq_when_U["G"], freq_when_C["G"], freq_when_G["G"], freq_when_A["G"])
)
return(results)
})

names(frequency_of5end_by_UMI)<-names(nt_7to10.df)

# Reshape the data for ggplot
plot_data5.L<-lapply(names(frequency_of5end_by_UMI), function(i){
  plot_data <- frequency_of5end_by_UMI[[i]] %>%
    gather(key = "nucleotide", value = "frequency", -pos2) %>%
    mutate(nucleotide = substr(nucleotide, 6, 6),
           nucleotide = factor(nucleotide, levels = c("A", "G", "C", "U")))
  return(plot_data)
})

names(plot_data5.L)<-names(frequency_of5end_by_UMI)

```

```

plot_5nucleotide_freq <- function(data) {

  # Define nucleotide colors
  nuc_colors <- c("A" = "green4", "G" = "goldenrod1",
                  "C" = "dodgerblue3", "U" = "firebrick3")

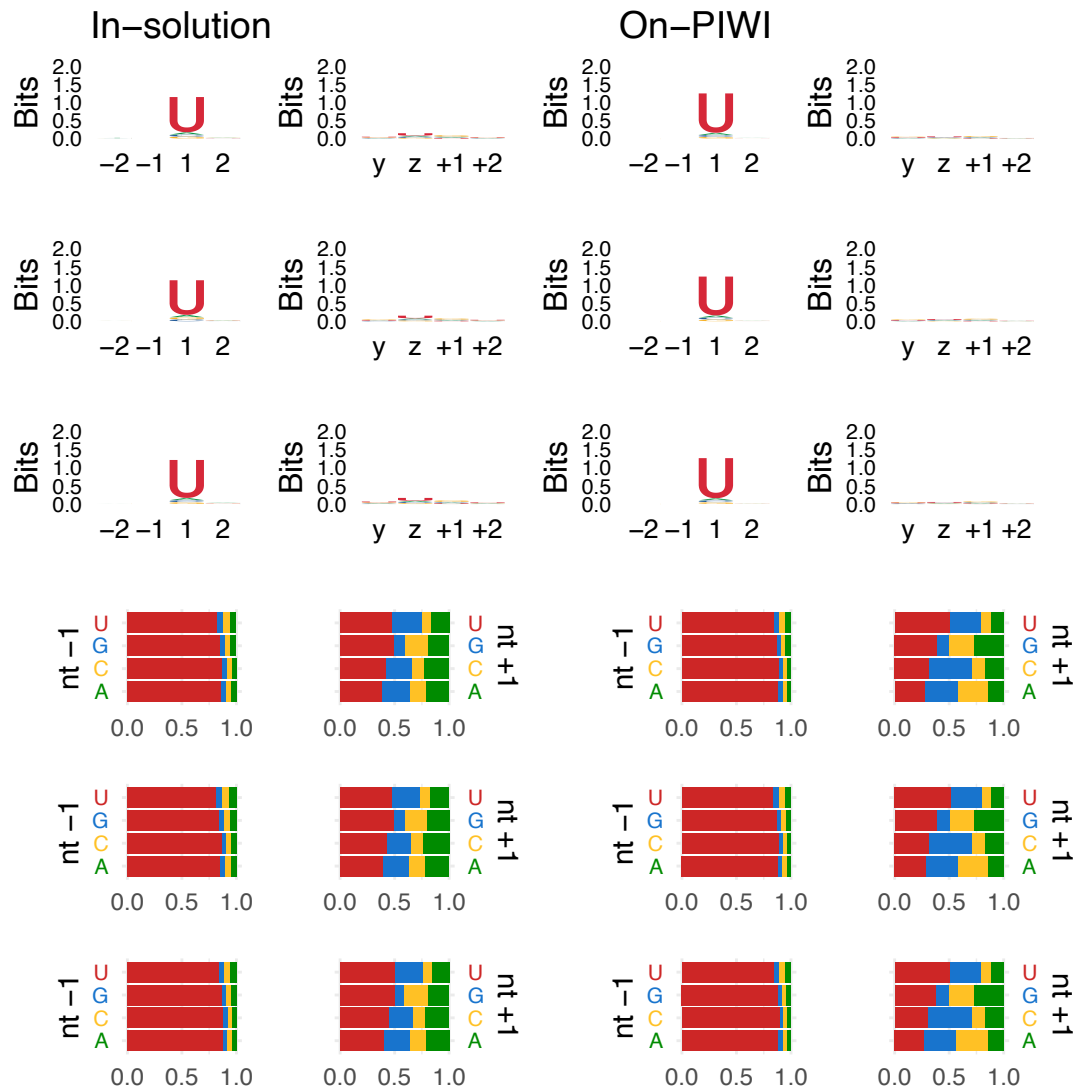
  ggplot(data, aes(x = frequency, y = pos2, fill = nucleotide)) +
    geom_bar(stat = "identity", position = "stack") +
    scale_fill_manual(values = nuc_colors) +
    theme_minimal() +
    labs(x= element_blank(),
         y = "nt -1",
         fill = "Nucleotide in Position 2") +
    theme(
      axis.text.y = element_text(color = nuc_colors, size = 8),
      aspect.ratio = 0.8,
      legend.position = "none"
    ) +
    scale_x_continuous(breaks = c(0, 1, 0.5))
}

On1_5end_freq<-plot_5nucleotide_freq(plot_data5.L$On1_withUMIs)
On2_5end_freq<-plot_5nucleotide_freq(plot_data5.L$On2_withUMIs)
On3_5end_freq<-plot_5nucleotide_freq(plot_data5.L$On3_withUMIs)
Off1_5end_freq<-plot_5nucleotide_freq(plot_data5.L$Off1_withUMIs)
Off2_5end_freq<-plot_5nucleotide_freq(plot_data5.L$Off2_withUMIs)
Off3_5end_freq<-plot_5nucleotide_freq(plot_data5.L$Off3_withUMIs)

Ligation_ctrl_bars<-(Off1_5end_freq / Off2_5end_freq / Off3_5end_freq |
                     Off1_3end_freq / Off2_3end_freq / Off3_3end_freq |
                     On1_5end_freq / On2_5end_freq / On3_5end_freq |
                     On1_3end_freq / On2_3end_freq / On3_3end_freq)

Ligation_ctrl_Logos / Ligation_ctrl_bars

```



BFF VS std. *Drosophila* Piwi-piRNAs length

Import mapped data

```
# Import primary alignments using PICBload

if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

BiocManager::install("PICB")

library(PICB)
```

```

library(BSgenome.Dmelanogaster.UCSC.dm6)
myGenome <- "BSgenome.Dmelanogaster.UCSC.dm6"

# load bam files

BFF_1<- unlist(GRangesList(PICBload(
  BAMFILE = BFF1_bam_path,REFERENCE.GENOME = myGenome,
  PERFECT.MATCH.ONLY=FALSE, IS.SECONDARY.ALIGNMENT=FALSE,
  STANDARD.CONTIGS.ONLY=FALSE, FILTER.BY.FLAG=FALSE,
  READ.SIZE.RANGE = c(19-56), USE.SIZE.FILTER=FALSE,
  GET.ORIGINAL.SEQUENCE = TRUE
)))

BFF_2<- unlist(GRangesList(PICBload(
  BAMFILE = BFF2_bam_path,REFERENCE.GENOME = myGenome,
  PERFECT.MATCH.ONLY=FALSE, IS.SECONDARY.ALIGNMENT=FALSE,
  STANDARD.CONTIGS.ONLY=FALSE, FILTER.BY.FLAG=FALSE,
  READ.SIZE.RANGE = c(19-56), USE.SIZE.FILTER=FALSE,
  GET.ORIGINAL.SEQUENCE = TRUE
)))

BFF_3<- unlist(GRangesList(PICBload(
  BAMFILE = BFF3_bam_path,REFERENCE.GENOME = myGenome,
  PERFECT.MATCH.ONLY=FALSE, IS.SECONDARY.ALIGNMENT=FALSE,
  STANDARD.CONTIGS.ONLY=FALSE, FILTER.BY.FLAG=FALSE,
  READ.SIZE.RANGE = c(19-56), USE.SIZE.FILTER=FALSE,
  GET.ORIGINAL.SEQUENCE = TRUE
)))

# Combine in list of GRanges
BFF_list<-GRangesList(BFF_1, BFF_2, BFF_3)

names(BFF_list)<-c("BFF_1", "BFF_2", "BFF_3")

OSS5_BFF_list<-lapply(names(BFF_list), function(i){
  GR<-BFF_list[[i]]
  GR$sample<-i
  GR$MULT<-NULL
  return(GR)
})

names(OSS5_BFF_list)<-names(BFF_list)

std_1<- unlist(GRangesList(PICBload(
  BAMFILE = std1_bam_path,REFERENCE.GENOME = myGenome,
  PERFECT.MATCH.ONLY=FALSE, IS.SECONDARY.ALIGNMENT=FALSE,
  STANDARD.CONTIGS.ONLY=FALSE, FILTER.BY.FLAG=FALSE,
  READ.SIZE.RANGE = c(19-56), USE.SIZE.FILTER=FALSE,
  GET.ORIGINAL.SEQUENCE = TRUE
)))

std_2<- unlist(GRangesList(PICBload(
  BAMFILE = std2_bam_path,REFERENCE.GENOME = myGenome,

```

```

    PERFECT.MATCH.ONLY=FALSE, IS.SECONDARY.ALIGNMENT=FALSE,
    STANDARD.CONTIGS.ONLY=FALSE, FILTER.BY.FLAG=FALSE,
    READ.SIZE.RANGE = c(19-56), USE.SIZE.FILTER=FALSE,
    GET.ORIGINAL.SEQUENCE = TRUE
  )))

std_3<- unlist(GRangesList(PICBload(
  BAMFILE = std3_bam_path,REFERENCE.GENOME = myGenome,
  PERFECT.MATCH.ONLY=FALSE, IS.SECONDARY.ALIGNMENT=FALSE,
  STANDARD.CONTIGS.ONLY=FALSE, FILTER.BY.FLAG=FALSE,
  READ.SIZE.RANGE = c(19-56), USE.SIZE.FILTER=FALSE,
  GET.ORIGINAL.SEQUENCE = TRUE
  )))

# Combine in list of GRanges
std_list<-GRangesList(std_1, std_2, std_3)

names(std_list)<-c("std_1", "std_2", "std_3")

OSS5_std_list<-lapply(names(std_list), function(i){
  GR<-std_list[[i]]
  GR$sample<-i
  GR$MULT<-NULL
  return(GR)
})

names(OSS5_std_list)<-names(std_list)

```

## Plot size distributions (Fig.2a)

```

library(GenomicRanges)

# BFF piRNA-seq
BFF_lengths.dt<-rbindlist(lapply(names(OSS5_BFF_list), function(i){
  GR<-OSS5_BFF_list[[i]]
  DT<-data.table("experiment"="BFF", "sample"=GR$sample, "width"=width(GR))
  return(DT)
})))

BFF_lengths.dt[, "reads" := .N, by=c("width", "sample")]
BFF_lengths.dt[, "total" := .N, by="sample"]
BFF_lengths.dt<-unique(BFF_lengths.dt[,c("experiment",
                                           "sample", "width", "reads", "total")])

BFF_lengths.dt$rpm<-(BFF_lengths.dt$reads/BFF_lengths.dt$total)*1000000

BFF_lengths2.dt<-BFF_lengths.dt
BFF_lengths2.dt[, "mean" := mean(rpm), by=width]
BFF_lengths2.dt[, "SD" := sd(rpm), by=width]
BFF_lengths2.dt[, "SE" := SD/length(rpm), by=width]
BFF_lengths3.dt<-unique(BFF_lengths2.dt[,c("experiment",
                                             "width", "mean", "SD", "SE")])

```

```

# Plot average BFF piRNA lengths with error bars
colors2 <- c("std" = "gray68", "BFF" = "darkorange")

BFF_distribution<-ggplot() + theme_pubclean() +
  geom_bar(data = BFF_lengths3.dt,
    aes(x = width, y = mean/100000, fill="BFF"), stat="identity") +
  geom_errorbar(data = BFF_lengths3.dt,
    aes(x = width, ymin=(mean-SE)/100000, ymax=(mean+SE)/100000), width=.1) +
  geom_vline(xintercept =27, color="grey", linetype=2)+
  scale_x_continuous(breaks = c(20, 25, 30, 35, 40))+
  theme(aspect.ratio = 0.6,
    legend.position = c(0.8, 0.8)) +
  scale_fill_manual("experiment", values = colors2)+
  labs(fill= "experiment") +
  xlab("width (nt)") +
  ylab(expression("ncov (rpm x 10"5*)")) +
  coord_cartesian(xlim = c(19, 40), ylim = c(0, 2), expand = TRUE)

# Standard method
std_lengths.dt<-rbindlist(lapply(names(OSS5_std_list), function(i){
  GR<-OSS5_std_list[[i]]
  DT<-data.table("experiment"="std", "sample"=GR$sample, "width"=width(GR))
  return(DT)
})))

std_lengths.dt[, "reads" := .N, by=c("width", "sample")]
std_lengths.dt[, "total" := .N, by="sample"]
std_lengths.dt<-unique(std_lengths.dt[, c("experiment",
  "sample", "width", "reads", "total")])
std_lengths.dt$rpm<-(std_lengths.dt$reads/std_lengths.dt$total)*1000000

# normalize reads to 24nt long flam read count}

# Find 24nt long unique-mappers within the Flam super-cluster
flam_sc<-GRanges(seqnames = "chrX",
  ranges = IRanges(21631611:22127300), strand = "+")

# std. method
std_lengths.dt$flam24nt<-0
std_lengths.dt[std_lengths.dt$sample %in% "std_1"]$flam24nt<-
  length(subsetByOverlaps(x = OSS5_std_list$std_1[
    OSS5_std_list$std_1$NH == 1 & width(OSS5_std_list$std_1) == 24],
    ranges = flam_sc, type = "within", ignore.strand=FALSE))

std_lengths.dt[std_lengths.dt$sample %in% "std_2"]$flam24nt<-
  length(subsetByOverlaps(x = OSS5_std_list$std_2[
    OSS5_std_list$std_2$NH == 1 & width(OSS5_std_list$std_2) == 24],
    ranges = flam_sc, type = "within", ignore.strand=FALSE))

std_lengths.dt[std_lengths.dt$sample %in% "std_3"]$flam24nt<-

```

```

length(subsetByOverlaps(x = OSS5_std_list$std_3[
  OSS5_std_list$std_3$NH == 1 & width(OSS5_std_list$std_3) == 24],
  ranges = flam_sc, type = "within", ignore.strand=FALSE))

# BFF
BFF_lengths.dt$flam24nt<-0
BFF_lengths.dt[BFF_lengths.dt$sample %in% "BFF_1"]$flam24nt<-
  length(subsetByOverlaps(x = OSS5_BFF_list$BFF_1[
    OSS5_BFF_list$BFF_1$NH == 1 & width(OSS5_BFF_list$BFF_1) == 24],
    ranges = flam_sc, type = "within", ignore.strand=FALSE))

BFF_lengths.dt[BFF_lengths.dt$sample %in% "BFF_2"]$flam24nt<-
  length(subsetByOverlaps(x = OSS5_BFF_list$BFF_2[
    OSS5_BFF_list$BFF_2$NH == 1 & width(OSS5_BFF_list$BFF_2) == 24],
    ranges = flam_sc, type = "within", ignore.strand=FALSE))

BFF_lengths.dt[BFF_lengths.dt$sample %in% "BFF_3"]$flam24nt<-
  length(subsetByOverlaps(x = OSS5_BFF_list$BFF_3[
    OSS5_BFF_list$BFF_3$NH == 1 & width(OSS5_BFF_list$BFF_3) == 24],
    ranges = flam_sc, type = "within", ignore.strand=FALSE))

# Make new data tables
std_lengths_flam_norm.dt<-std_lengths.dt[, .SD, .SDcols = c(
  "experiment", "sample", "width", "reads", "flam24nt")]
BFF_lengths_flam_norm.dt<-BFF_lengths.dt[, .SD, .SDcols = c(
  "experiment", "sample", "width", "reads", "flam24nt")]

# Normalize reads per length to the flam24nt (for each sample)

# std method
std_lengths_flam_norm.dt$flamNorm_reads<-
  std_lengths_flam_norm.dt$reads/std_lengths_flam_norm.dt$flam24nt
std_lengths_flam_norm.dt$mean<-0
std_lengths_flam_norm.dt[, "mean":=mean(flamNorm_reads), by="width"]
std_lengths_flam_norm.dt[, "SD":= sd(flamNorm_reads), by="width"]
std_lengths_flam_norm.dt[, "SE":= SD/length(flamNorm_reads), by="width"]

# BFF-beads
BFF_lengths_flam_norm.dt$flamNorm_reads<-
  BFF_lengths_flam_norm.dt$reads/BFF_lengths_flam_norm.dt$flam24nt
BFF_lengths_flam_norm.dt$mean<-0
BFF_lengths_flam_norm.dt[, "mean":=mean(flamNorm_reads), by="width"]
BFF_lengths_flam_norm.dt[, "SD":= sd(flamNorm_reads), by="width"]
BFF_lengths_flam_norm.dt[, "SE":= SD/length(flamNorm_reads), by="width"]

## Plot average std VS BFF piRNA lengths normalized to 24nt flam reads with error bars
normalized_distributions<-ggplot() + theme_pubclean() +
  geom_line(data = BFF_lengths_flam_norm.dt,
    aes(x = width, y = mean, color=experiment)) +
  geom_errorbar(data = BFF_lengths_flam_norm.dt,

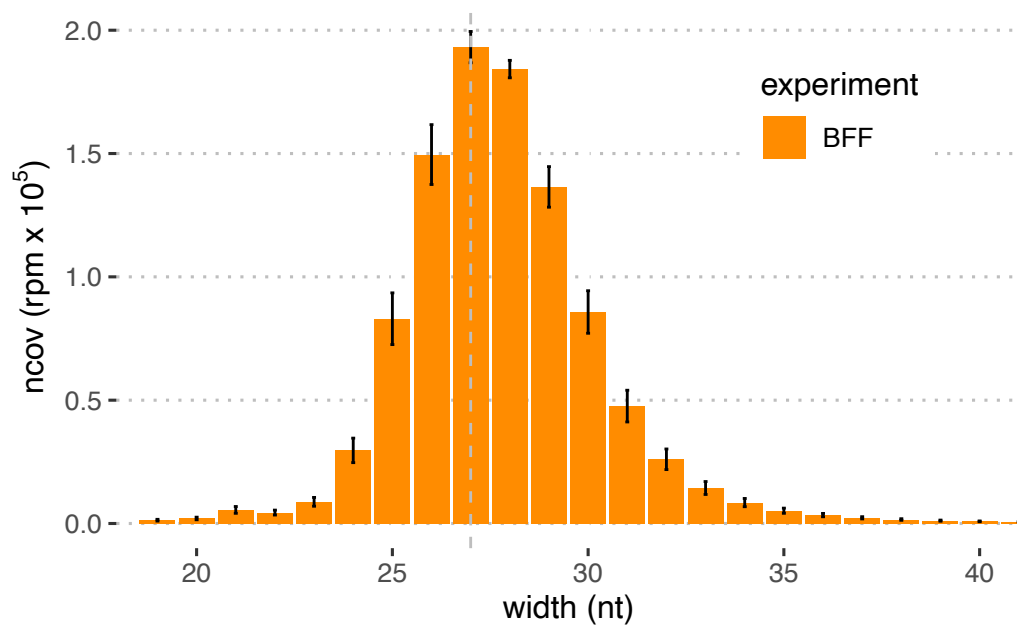
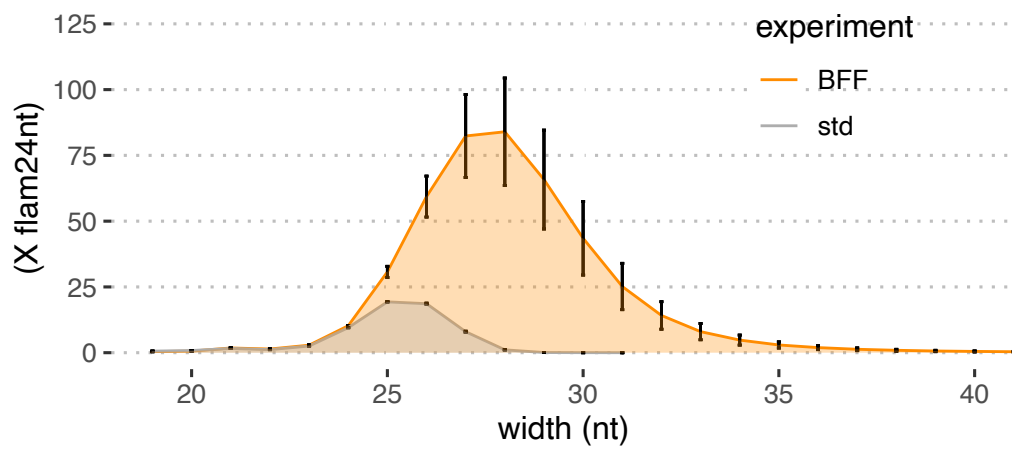
```

```

        aes(x = width, ymin=mean-SE, ymax=mean+SE), width=.1) +
geom_line(data = std_lengths_flam_norm.dt,
        aes(x = width, y = mean, color=experiment)) +
geom_errorbar(data = std_lengths_flam_norm.dt,
        aes(x = width, ymin=mean-SE, ymax=mean+SE), width=.1) +
  # Filled areas
geom_area(data = BFF_lengths_flam_norm.dt,
        aes(x = width, y = mean),
        fill = colors2["BFF"],
        alpha = 0.3) +
geom_area(data = std_lengths_flam_norm.dt,
        aes(x = width, y = mean),
        fill = colors2["std"],
        alpha = 0.3) +
scale_color_manual("experiment", values = colors2)+
scale_x_continuous(breaks = c(20, 25, 30, 35, 40))+
scale_y_continuous(breaks = c(0, 25, 50, 75, 100, 125))+
theme(aspect.ratio = 0.4,
      plot.title = element_text(hjust=0.4),
      legend.background = element_blank(),
      legend.position = c(0.8, 0.8),
      plot.caption = element_text(hjust = 1)) +
xlab("width (nt)")+
ylab("(X flam24nt)")+
coord_cartesian(xlim = c(19, 40), ylim = c(0, 125), expand = TRUE)

```

BFF\_distribution / normalized\_distributions



# Investigation of processing piRNA patterns

```
suppressPackageStartupMessages({
  library(PICB)
  library(BSgenome.Dmelanogaster.UCSC.dm6)
  library(ggplot2)
  library(Biostrings)
  library(GenomicRanges)
  library(tidyr)
  library(dplyr)
})
myGenome_dm6 <- "BSgenome.Dmelanogaster.UCSC.dm6"
```

## Fly Piwi - OSS

### Load alignments

```
# PIWI OSC on dm6 - R1
bam_PIWI_oss_dm_R1_dir <-
"STAROutput_PIWIoss/NonStructural_OnVSoft_beads_replicate_1_S1_trimmed_On_trimmed_collapsed_UMI_removed_miRNAout_Dm6Aligned-STAR_sortedByCoord.bam"

align_PIWI_oss_dm_R1 <- PICBload(
  BAMFILE = bam_PIWI_oss_dm_R1_dir,
  REFERENCE.GENOME = myGenome_dm6,
  WHAT = c("flag", "cigar"),
  GET.ORIGINAL.SEQUENCE = TRUE,
  VERBOSE = FALSE
)
length(align_PIWI_oss_dm_R1$unique) + length(align_PIWI_oss_dm_R1$multi.primary)
align_PIWI_oss_dm_R1_prim <- c(align_PIWI_oss_dm_R1$unique,
align_PIWI_oss_dm_R1$multi.primary)
```

4588059

```
# PIWI OSC on dm6 - R2
bam_PIWI_oss_dm_R2_dir <-
"STAROutput_PIWIoss/NonStructural_OnVSoft_beads_replicate_2_S2_trimmed_On_trimmed_collapsed_UMI_removed_miRNAout_Dm6Aligned-STAR_sortedByCoord.bam"

align_PIWI_oss_dm_R2 <- PICBload(
  BAMFILE = bam_PIWI_oss_dm_R2_dir,
  REFERENCE.GENOME = myGenome_dm6,
  WHAT = c("flag", "cigar"),
  GET.ORIGINAL.SEQUENCE = TRUE,
  VERBOSE = FALSE
)
length(align_PIWI_oss_dm_R2$unique) + length(align_PIWI_oss_dm_R2$multi.primary)
align_PIWI_oss_dm_R2_prim <- c(align_PIWI_oss_dm_R2$unique,
align_PIWI_oss_dm_R2$multi.primary)
```

2832718

```
# PIWI OSC on dm6 - R3
bam_PIWI_oss_dm_R3_dir <-
"STAROutput_PiWIoss/NonStructural_OnVSoft_beads_replicate_3_S3_trimmed_On_trimmed_collapsed_UMI_removed_miRNAout_Dm6Aligned-STAR_sortedByCoord.bam"

align_PIWI_oss_dm_R3 <- PICBload(
  BAMFILE = bam_PIWI_oss_dm_R3_dir,
  REFERENCE.GENOME = myGenome_dm6,
  WHAT = c("flag", "cigar"),
  GET.ORIGINAL.SEQUENCE = TRUE,
  VERBOSE = FALSE
)
length(align_PIWI_oss_dm_R3$unique) + length(align_PIWI_oss_dm_R3$multi.primary)
align_PIWI_oss_dm_R3_prim <- c(align_PIWI_oss_dm_R3$unique,
align_PIWI_oss_dm_R3$multi.primary)
```

3719247

```
message("PIWI OSC on dm6 - R1, # of unique alignments: ",
length(align_PIWI_oss_dm_R1$unique), " <- most unique alignments")
message("PIWI OSC on dm6 - R2, # of unique alignments: ",
length(align_PIWI_oss_dm_R2$unique))
message("PIWI OSC on dm6 - R3, # of unique alignments: ",
length(align_PIWI_oss_dm_R3$unique))
```

PIWI OSC on dm6 - R1, # of unique alignments: 1362764 <- most unique alignments

PIWI OSC on dm6 - R2, # of unique alignments: 879183

PIWI OSC on dm6 - R3, # of unique alignments: 1129911

## Retrieve genomic regions enriched in unique alignments

```
# PIWI OSC on dm6 - R1
PICB_PIWI_oss_dm_R1 <- PICBbuild(
  IN.ALIGNMENTS = align_PIWI_oss_dm_R1,
  REFERENCE.GENOME = myGenome_dm6,
  VERBOSITY = 0
)

# define regions enriched in unique alignments
seeds_PIWI_oss_dm_R1 <-
PICB_PIWI_oss_dm_R1$seeds[order(PICB_PIWI_oss_dm_R1$seeds$fraction_of_width_covered_by_unique_alignments, decreasing = TRUE),]
seeds_PIWI_oss_dm_R1_highFrac <-
seeds_PIWI_oss_dm_R1[seeds_PIWI_oss_dm_R1$fraction_of_width_covered_by_unique_alignments
>= 0.50]
seeds_PIWI_oss_dm_R1_highFrac_noMM <-
seeds_PIWI_oss_dm_R1_highFrac[seeds_PIWI_oss_dm_R1_highFrac$multimapping_reads_primary_alignments_FPM == 0,]
seeds_PIWI_oss_dm_R1_highFrac_noMM <-
seeds_PIWI_oss_dm_R1_highFrac_noMM[order(seeds_PIWI_oss_dm_R1_highFrac_noMM$uniq_reads_FPM, decreasing = TRUE),]
```

```

# PIWI OSC on dm6 - R2
PICB_PIWI_oss_dm_R2 <- PICBbuild(
  IN.ALIGNMENTS = align_PIWI_oss_dm_R2,
  REFERENCE.GENOME = myGenome_dm6,
  VERBOSITY = 0
)

# define regions enriched in unique alignments
seeds_PIWI_oss_dm_R2 <-
PICB_PIWI_oss_dm_R2$seeds[order(PICB_PIWI_oss_dm_R2$seeds$fraction_of_width_covered_by_u
nique_alignments, decreasing = TRUE),]
seeds_PIWI_oss_dm_R2_highFrac <-
seeds_PIWI_oss_dm_R2[seeds_PIWI_oss_dm_R2$fraction_of_width_covered_by_unique_alignments
>= 0.50]
seeds_PIWI_oss_dm_R2_highFrac_noMM <-
seeds_PIWI_oss_dm_R2_highFrac[seeds_PIWI_oss_dm_R2_highFrac$multimapping_reads_primary_a
lignments_FPM == 0,]
seeds_PIWI_oss_dm_R2_highFrac_noMM <-
seeds_PIWI_oss_dm_R2_highFrac_noMM[order(seeds_PIWI_oss_dm_R2_highFrac_noMM$uniq_reads_F
PM, decreasing = TRUE),]

```

```

# PIWI OSC on dm6 - R3
PICB_PIWI_oss_dm_R3 <- PICBbuild(
  IN.ALIGNMENTS = align_PIWI_oss_dm_R3,
  REFERENCE.GENOME = myGenome_dm6,
  COMPUTE.1U.10A.FRACTIONS = TRUE,
  VERBOSITY = 0
)

# define regions enriched in unique alignments
seeds_PIWI_oss_dm_R3 <-
PICB_PIWI_oss_dm_R3$seeds[order(PICB_PIWI_oss_dm_R3$seeds$fraction_of_width_covered_by_u
nique_alignments, decreasing = TRUE),]
seeds_PIWI_oss_dm_R3_highFrac <-
seeds_PIWI_oss_dm_R3[seeds_PIWI_oss_dm_R3$fraction_of_width_covered_by_unique_alignments
>= 0.50]
seeds_PIWI_oss_dm_R3_highFrac_noMM <-
seeds_PIWI_oss_dm_R3_highFrac[seeds_PIWI_oss_dm_R3_highFrac$multimapping_reads_primary_a
lignments_FPM == 0,]
seeds_PIWI_oss_dm_R3_highFrac_noMM <-
seeds_PIWI_oss_dm_R3_highFrac_noMM[order(seeds_PIWI_oss_dm_R3_highFrac_noMM$uniq_reads_F
PM, decreasing = TRUE),]

```

```

# Keep only those seeds in R1 (most unique alignments) that overlap with both R2 and R3
hits_R2 <- findOverlaps(seeds_PIWI_oss_dm_R1_highFrac_noMM,
seeds_PIWI_oss_dm_R2_highFrac_noMM, ignore.strand = FALSE)
R1_in_R2 <- unique(queryHits(hits_R2))

hits_R3 <- findOverlaps(seeds_PIWI_oss_dm_R1_highFrac_noMM,
seeds_PIWI_oss_dm_R3_highFrac_noMM, ignore.strand = FALSE)
R1_in_R3 <- unique(queryHits(hits_R3))

R1_keep <- intersect(R1_in_R2, R1_in_R3)
seeds_PIWI_oss_dm_R1_highFrac_noMM_inAllR <- seeds_PIWI_oss_dm_R1_highFrac_noMM[R1_keep]

```

```
paste0("Number of seeds for OSC sample: ",
length(seeds_PIWI_oss_dm_R1_highFrac_noMM_inAllR))
message("Length of all filtered seeds in the OSC sample combined: ",
sum(width(seeds_PIWI_oss_dm_R1_highFrac_noMM_inAllR)), " nt.")
```

'Number of seeds for OSC sample: 83'

Length of all filtered seeds in the OSC sample combined: 450555 nt.

## Fly Piwi - Follicle

### Load alignments

```
# PIWI follicle on dm6 - R1
bam_PIWI_fol_dm_R1_dir <-
"STAROutput_PIWIfollicle/NonStructural_Ovaries_1_S3_R1_miRNA_Unmapped_dm6_Aligned.sorted
ByCoord.out.bam"

align_PIWI_fol_dm_R1 <- PICBload(
  BAMFILE = bam_PIWI_fol_dm_R1_dir,
  REFERENCE.GENOME = myGenome_dm6,
  WHAT = c("flag", "cigar"),
  GET.ORIGINAL.SEQUENCE = TRUE,
  VERBOSE = FALSE
)
length(align_PIWI_fol_dm_R1$unique) + length(align_PIWI_fol_dm_R1$multi.primary)
align_PIWI_fol_dm_R1_prim <- c(align_PIWI_fol_dm_R1$unique,
align_PIWI_fol_dm_R1$multi.primary)
```

9145475

```
# PIWI follicle on dm6 - R2
bam_PIWI_fol_dm_R2_dir <-
"STAROutput_PIWIfollicle/NonStructural_rep2_0v1_S1_R1_miRNA_Unmapped_dm6_Aligned.sortedB
yCoord.out.bam"

align_PIWI_fol_dm_R2 <- PICBload(
  BAMFILE = bam_PIWI_fol_dm_R2_dir,
  REFERENCE.GENOME = myGenome_dm6,
  WHAT = c("flag", "cigar"),
  GET.ORIGINAL.SEQUENCE = TRUE,
  VERBOSE = FALSE
)
length(align_PIWI_fol_dm_R2$unique) + length(align_PIWI_fol_dm_R2$multi.primary)
align_PIWI_fol_dm_R2_prim <- c(align_PIWI_fol_dm_R2$unique,
align_PIWI_fol_dm_R2$multi.primary)
```

14586920

```
# PIWI follicle on dm6 - R3
bam_PIWI_fol_dm_R3_dir <-
"STAROutput_PIWIfollicle/NonStructural_rep3_0v1_S7_R1_miRNA_Unmapped_dm6_Aligned.sortedB
yCoord.out.bam"

align_PIWI_fol_dm_R3 <- PICBload(
```

```

BAMFILE = bam_PIWI_fol_dm_R3_dir,
REFERENCE.GENOME = myGenome_dm6,
WHAT = c("flag", "cigar"),
GET.ORIGINAL.SEQUENCE = TRUE,
VERBOSE = FALSE
)
length(align_PIWI_fol_dm_R3$unique) + length(align_PIWI_fol_dm_R3$multi.primary)
align_PIWI_fol_dm_R3_prim <- c(align_PIWI_fol_dm_R3$unique,
align_PIWI_fol_dm_R3$multi.primary)

```

1813270

```

message("PIWI follicle on dm6 - R1, # of unique alignments: ",
length(align_PIWI_fol_dm_R1$unique))
message("PIWI follicle on dm6 - R2, # of unique alignments: ",
length(align_PIWI_fol_dm_R2$unique), " <- most unique alignments")
message("PIWI follicle on dm6 - R3, # of unique alignments: ",
length(align_PIWI_fol_dm_R3$unique))

```

PIWI follicle on dm6 - R1, # of unique alignments: 2521307

PIWI follicle on dm6 - R2, # of unique alignments: 3710503 <- most unique alignments

PIWI follicle on dm6 - R3, # of unique alignments: 487014

## Retrieve genomic regions enriched in unique alignments

```

# PIWI follicle on dm6 - R1
PICB_PIWI_fol_dm_R1 <- PICBbuild(
  IN.ALIGNMENTS = align_PIWI_fol_dm_R1,
  REFERENCE.GENOME = myGenome_dm6,
  VERBOSITY = 0
)

# define regions enriched in unique alignments
seeds_PIWI_fol_dm_R1 <-
PICB_PIWI_fol_dm_R1$seeds[order(PICB_PIWI_fol_dm_R1$seeds$fraction_of_width_covered_by_u
nique_alignments, decreasing = TRUE),]
seeds_PIWI_fol_dm_R1_highFrac <-
seeds_PIWI_fol_dm_R1[seeds_PIWI_fol_dm_R1$fraction_of_width_covered_by_unique_alignments
>= 0.50]
seeds_PIWI_fol_dm_R1_highFrac_noMM <-
seeds_PIWI_fol_dm_R1_highFrac[seeds_PIWI_fol_dm_R1_highFrac$multimapping_reads_primary_a
lignments_FPM == 0,]
seeds_PIWI_fol_dm_R1_highFrac_noMM <-
seeds_PIWI_fol_dm_R1_highFrac_noMM[order(seeds_PIWI_fol_dm_R1_highFrac_noMM$uniq_reads_F
PM, decreasing = TRUE),]

```

```

# PIWI follicle on dm6 - R2
PICB_PIWI_fol_dm_R2 <- PICBbuild(
  IN.ALIGNMENTS = align_PIWI_fol_dm_R2,
  REFERENCE.GENOME = myGenome_dm6,
  VERBOSITY = 0
)

# define regions enriched in unique alignments

```

```

seeds_PIWI_fol_dm_R2 <-
PICB_PIWI_fol_dm_R2$seeds[order(PICB_PIWI_fol_dm_R2$seeds$fraction_of_width_covered_by_u
nique_alignments, decreasing = TRUE),]
seeds_PIWI_fol_dm_R2_highFrac <-
seeds_PIWI_fol_dm_R2[seeds_PIWI_fol_dm_R2$fraction_of_width_covered_by_unique_alignments
>= 0.50]
seeds_PIWI_fol_dm_R2_highFrac_noMM <-
seeds_PIWI_fol_dm_R2_highFrac[seeds_PIWI_fol_dm_R2_highFrac$multimapping_reads_primary_a
lignments_FPM == 0,]
seeds_PIWI_fol_dm_R2_highFrac_noMM <-
seeds_PIWI_fol_dm_R2_highFrac_noMM[order(seeds_PIWI_fol_dm_R2_highFrac_noMM$uniq_reads_F
PM, decreasing = TRUE),]

```

```

PICB_PIWI_fol_dm_R3 <- PICBbuild(
  IN.ALIGNMENTS = align_PIWI_fol_dm_R3,
  REFERENCE.GENOME = myGenome_dm6,
  VERBOSITY = 0
)

# define regions enriched in unique alignments
seeds_PIWI_fol_dm_R3 <-
PICB_PIWI_fol_dm_R3$seeds[order(PICB_PIWI_fol_dm_R3$seeds$fraction_of_width_covered_by_u
nique_alignments, decreasing = TRUE),]
seeds_PIWI_fol_dm_R3_highFrac <-
seeds_PIWI_fol_dm_R3[seeds_PIWI_fol_dm_R3$fraction_of_width_covered_by_unique_alignments
>= 0.50]
seeds_PIWI_fol_dm_R3_highFrac_noMM <-
seeds_PIWI_fol_dm_R3_highFrac[seeds_PIWI_fol_dm_R3_highFrac$multimapping_reads_primary_a
lignments_FPM == 0,]
seeds_PIWI_fol_dm_R3_highFrac_noMM <-
seeds_PIWI_fol_dm_R3_highFrac_noMM[order(seeds_PIWI_fol_dm_R3_highFrac_noMM$uniq_reads_F
PM, decreasing = TRUE),]

```

```

# Keep only those seeds in R2 that overlap with both R1 and R3
hits_R1 <- findOverlaps(seeds_PIWI_fol_dm_R2_highFrac_noMM,
seeds_PIWI_fol_dm_R1_highFrac_noMM, ignore.strand = FALSE)
R2_in_R1 <- unique(queryHits(hits_R1))

hits_R3 <- findOverlaps(seeds_PIWI_fol_dm_R2_highFrac_noMM,
seeds_PIWI_fol_dm_R3_highFrac_noMM, ignore.strand = FALSE)
R2_in_R3 <- unique(queryHits(hits_R3))

R2_keep <- intersect(R2_in_R1, R2_in_R3)
seeds_PIWI_fol_dm_R2_highFrac_noMM_inAllR <- seeds_PIWI_fol_dm_R2_highFrac_noMM[R2_keep]

```

```

paste0("Number of seeds for follicle sample: ",
length(seeds_PIWI_fol_dm_R2_highFrac_noMM_inAllR))
message("Length of all filtered seeds in the follicle sample combined: ",
sum(width(seeds_PIWI_fol_dm_R2_highFrac_noMM_inAllR)), " nt.")

```

'Number of seeds for follicle sample: 50'

Length of all filtered seeds in the follicle sample combined: 168175 nt.

## Load alignments

```
# PIWI germline on dm6 - R1
bam_PIWI_germ_dm_R1_dir <-
"STAROutput_PIWIgermline/NonStructural_Ovaries_8_S2_R1_miRNA_Unmapped_dm6_Aligned.sorted
ByCoord.out.bam"

align_PIWI_germ_dm_R1 <- PICBload(
  BAMFILE = bam_PIWI_germ_dm_R1_dir,
  REFERENCE.GENOME = myGenome_dm6,
  WHAT = c("flag", "cigar"),
  GET.ORIGINAL.SEQUENCE = TRUE,
  VERBOSE = FALSE
)
length(align_PIWI_germ_dm_R1$unique) + length(align_PIWI_germ_dm_R1$multi.primary)
align_PIWI_germ_dm_R1_prim <- c(align_PIWI_germ_dm_R1$unique,
align_PIWI_germ_dm_R1$multi.primary)
```

1053572

```
# PIWI germline on dm6 - R2
bam_PIWI_germ_dm_R2_dir <-
"STAROutput_PIWIgermline/NonStructural_rep2_0v3_S3_R1_miRNA_Unmapped_dm6_Aligned.sortedB
yCoord.out.bam"

align_PIWI_germ_dm_R2 <- PICBload(
  BAMFILE = bam_PIWI_germ_dm_R2_dir,
  REFERENCE.GENOME = myGenome_dm6,
  WHAT = c("flag", "cigar"),
  GET.ORIGINAL.SEQUENCE = TRUE,
  VERBOSE = FALSE
)
length(align_PIWI_germ_dm_R2$unique) + length(align_PIWI_germ_dm_R2$multi.primary)
align_PIWI_germ_dm_R2_prim <- c(align_PIWI_germ_dm_R2$unique,
align_PIWI_germ_dm_R2$multi.primary)
```

3334392

```
# PIWI germline on dm6 - R3
bam_PIWI_germ_dm_R3_dir <-
"STAROutput_PIWIgermline/NonStructural_rep3_0v3_S9_R1_miRNA_Unmapped_dm6_Aligned.sortedB
yCoord.out.bam"

align_PIWI_germ_dm_R3 <- PICBload(
  BAMFILE = bam_PIWI_germ_dm_R3_dir,
  REFERENCE.GENOME = myGenome_dm6,
  WHAT = c("flag", "cigar"),
  GET.ORIGINAL.SEQUENCE = TRUE,
  VERBOSE = FALSE
)
length(align_PIWI_germ_dm_R3$unique) + length(align_PIWI_germ_dm_R3$multi.primary)
align_PIWI_germ_dm_R3_prim <- c(align_PIWI_germ_dm_R3$unique,
align_PIWI_germ_dm_R3$multi.primary)
```

3202961

```

message("PIWI germline on dm6 - R1, # of unique alignments: ",
length(align_PIWI_germ_dm_R1$unique))
message("PIWI germline on dm6 - R2, # of unique alignments: ",
length(align_PIWI_germ_dm_R2$unique), " <- most unique alignments")
message("PIWI germline on dm6 - R3, # of unique alignments: ",
length(align_PIWI_germ_dm_R3$unique))

```

PIWI germline on dm6 - R1, # of unique alignments: 298601

PIWI germline on dm6 - R2, # of unique alignments: 867913 <- most unique alignments

PIWI germline on dm6 - R3, # of unique alignments: 805143

## Retrieve genomic regions enriched in unique alignments

```

# PIWI germline on dm6 - R1
PICB_PIWI_germ_dm_R1 <- PICBbuild(
  IN.ALIGNMENTS = align_PIWI_germ_dm_R1,
  REFERENCE.GENOME = myGenome_dm6,
  VERBOSITY = 0
)

# define regions enriched in unique alignments
seeds_PIWI_germ_dm_R1 <-
PICB_PIWI_germ_dm_R1$seeds[order(PICB_PIWI_germ_dm_R1$seeds$fraction_of_width_covered_by
_unique_alignments, decreasing = TRUE),]

```

```

# PIWI germline on dm6 - R2
PICB_PIWI_germ_dm_R2 <- PICBbuild(
  IN.ALIGNMENTS = align_PIWI_germ_dm_R2,
  REFERENCE.GENOME = myGenome_dm6,
  VERBOSITY = 0
)

# define regions enriched in unique alignments
seeds_PIWI_germ_dm_R2 <-
PICB_PIWI_germ_dm_R2$seeds[order(PICB_PIWI_germ_dm_R2$seeds$fraction_of_width_covered_by
_unique_alignments, decreasing = TRUE),]

```

```

# PIWI germline on dm6 - R3
PICB_PIWI_germ_dm_R3 <- PICBbuild(
  IN.ALIGNMENTS = align_PIWI_germ_dm_R3,
  REFERENCE.GENOME = myGenome_dm6,
  VERBOSITY = 0
)

# define regions enriched in unique alignments
seeds_PIWI_germ_dm_R3 <-
PICB_PIWI_germ_dm_R3$seeds[order(PICB_PIWI_germ_dm_R3$seeds$fraction_of_width_covered_by
_unique_alignments, decreasing = TRUE),]

```

Seeds for germline Piwi are filtered at a later stage.

# Testing processing patterns

## Functions

```
# -----  
# Find the most abundant 5' end in each piC and show surrounding 5'ends (+/- 100bp)  
# -----  
find_most_abundant_5end <- function(piC_ranges, piRNA_data, n_top = 1, sampleName = "")  
{  
  results_df <- data.frame(  
    id = character(),  
    most_abundant_5end = numeric(),  
    rank = numeric(),  
    stringsAsFactors = FALSE  
  )  
  
  # Store the most abundant positions as GRanges  
  abundant_positions <- GRanges()  
  
  # iterate through given GRanges object piC_ranges  
  for (i in seq_along(piC_ranges)) {  
    current_piC <- piC_ranges[i]  
    id <- i  
    is_plus_strand <- as.character(strand(current_piC)) == "+"  
  
    # Get piRNAs that overlap with the current piC  
    overlapping_piRNAs <- subsetByOverlaps(piRNA_data, current_piC)  
  
    if (length(overlapping_piRNAs) > 0) {  
      # Get all 5' ends of the overlapping piRNAs (considering the strand  
orientation)  
      five_prime_ends <- if (is_plus_strand) start(overlapping_piRNAs) else  
end(overlapping_piRNAs)  
  
      # Count occurrences of each 5' end position  
      end_counts <- table(five_prime_ends)  
  
      # Sort by count (descending) and get top n_top positions  
      sorted_counts <- sort(end_counts, decreasing = TRUE)  
      n_to_take <- min(n_top, length(sorted_counts))  
      top_positions <- sorted_counts[1:n_to_take]  
  
      # Process each of the top positions  
      for (rank in 1:length(top_positions)) {  
        most_abundant_pos <- as.numeric(names(top_positions)[rank])  
  
        results_df <- rbind(results_df, data.frame(  
          id = id,  
          most_abundant_5end = most_abundant_pos,  
          rank = rank  
        ))  
  
        # Add to GRanges  
        abundant_positions <- c(abundant_positions,  
          GRanges(seqnames = seqnames(current_piC),  
            ranges = IRanges(start = most_abundant_pos,
```

```

        end = most_abundant_pos),
strand = strand(current_piC),
id = as.character(paste0(id, ".", rank)),
rank = rank))
    }
  }
}

return(abundant_positions)
}

# Helper function to find local maxima
find_local_maxima <- function(data, position_col = "position", value_col =
"mean_percent") {
  # Sort by position to ensure proper order
  data <- data[order(data[[position_col]]), ]

  n <- nrow(data)
  local_maxima <- c()

  # Check each point (excluding first and last)
  for (i in 2:(n-1)) {
    if (data[[value_col]][i] > data[[value_col]][i-1] &&
        data[[value_col]][i] > data[[value_col]][i+1]) {
      local_maxima <- c(local_maxima, data[[position_col]][i])
    }
  }

  return(local_maxima)
}

analyze_5end_distribution <- function(abundant_positions, piRNA_data, window_size = 100,
sampleName = "", piRNAlength = 27) {
  region_counts_df <- data.frame(
    id = character(),
    position = numeric(),
    count = numeric(),
    stringsAsFactors = FALSE
  )

  # iterate through each abundant position (from function find_most_abundant_5end)
  for (i in seq_along(abundant_positions)) {
    current_pos <- abundant_positions[i]
    id <- current_pos$id
    is_plus_strand <- as.character(strand(current_pos)) == "+"

    # Create region +/- window_size around most abundant 5' end
    region_start <- start(current_pos) - window_size
    region_end <- start(current_pos) + window_size
    region_range <- GRanges(
      seqnames = seqnames(current_pos),
      ranges = IRanges(start = region_start, end = region_end),
      strand = strand(current_pos)
    )

    # Get piRNAs overlapping the region
    region_piRNAs <- subsetByOverlaps(piRNA_data, region_range)

    # Get 5' ends based on strand

```

```

region_5ends <- if (is_plus_strand) start(region_piRNAs) else end(region_piRNAs)

# Count 5' ends per position
positions <- region_start:region_end
counts <- sapply(positions, function(pos) sum(region_5ends == pos))

# Adjust positions relative to most abundant 5' end
relative_positions <- if (is_plus_strand) {
  -window_size:window_size
} else {
  window_size:(-window_size)
}

region_counts_df <- rbind(
  region_counts_df,
  data.frame(
    id = id,
    position = relative_positions,
    count = counts
  )
)

}

# Calculate total counts per id and add percentage column
region_counts_percent <- region_counts_df %>%
  group_by(id) %>%
  mutate(
    total_counts = sum(count),
    percentInWindow = (count / total_counts) * 100
  ) %>%
  ungroup()

region_counts_per_df <- region_counts_percent %>%
  group_by(position) %>%
  summarise(
    mean_percent = mean(percentInWindow, na.rm = TRUE),
    se_percent = ifelse(n() > 1, sd(percentInWindow, na.rm = TRUE)/sqrt(n()), 0)
  )

# Plot mean with error bands using percentage per window
p3 <- ggplot() +
  geom_ribbon(data = region_counts_per_df,
    aes(x = position,
      ymin = mean_percent - se_percent,
      ymax = mean_percent + se_percent),
    alpha = 0.7,
    fill = "darkgrey") +
  geom_line(data = region_counts_per_df,
    aes(x = position, y = mean_percent),
    color = "#00a89e", linewidth = 1) +
  geom_vline(xintercept = c(piRNAlength, piRNAlength*2, piRNAlength*3,
    -piRNAlength, -piRNAlength*2, -piRNAlength*3),
    linetype = "dashed", color = "gray60", size = 0.3) +
  labs(
    x = "Position relative to most abundant 5' end",
    y = "Mean percent 5' ends in window-region",
    title = "Mean distribution of 5' ends with standard error"
  ) +
  scale_x_continuous(limits = c(-window_size, window_size),

```

```

        breaks = seq(-window_size, window_size, by = 10)) +
theme(
  axis.text.y = element_text(size = 8),
  panel.grid.major.x = element_line(color = "gray90", linetype = "solid"),
  panel.grid.minor.x = element_blank()
) +
theme_classic()

print(p3)

# Table with local maxima plot
local_maxima_positions <- find_local_maxima(region_counts_per_df)
maxima_data <- region_counts_per_df[region_counts_per_df$position %in%
local_maxima_positions, ]
maxima_data <- maxima_data[order(maxima_data$mean_percent, decreasing = TRUE), ]
print("Local maxima with their values sorted by mean_percent:")
print(maxima_data[, c("position", "mean_percent")])

return(region_counts_percent)
}
# -----

# Plot piRNA length distributions from cigar values
# Needs to be run with alignments with simple cigars
# (e.g. through PICBload w/ WHAT = c("flag", "cigar") and otherwise default parameters)
# -----

piRNALengthsFromCigarOneSample <- function(alignmentsWCigar,
                                             sampleName,
                                             selected_GRanges = NULL,
                                             barColor = "#523896",
                                             comprehensiveOutput = FALSE,
                                             range = c(20, 35),
                                             yRange = NULL) {
  if (!is.null(selected_GRanges)) {
    alignmentsWCigar <- subsetByOverlaps(piRNALengths_sample, selected_GRanges)
    print("Subset by overlaps with your selected GRanges.")
  }

  #get frequency table for cigars
  alignmentsWCigarFT <- as.data.frame(table(alignmentsWCigar$cigar))
  alignmentsWCigarFT$Percent <- alignmentsWCigarFT$Freq / length(alignmentsWCigar)

  #Remove 'M' from frequency table (allow later conversion to numerical number)
  alignmentsWCigarFT$Var1 <- gsub("M", "", alignmentsWCigarFT$Var1)

  # Convert Var1 to numeric
  alignmentsWCigarFT$Var1 <- as.numeric(as.character(alignmentsWCigarFT$Var1))

  # Create a complete data frame with all lengths from 20 to 35 or range
  all_lengths <- data.frame(Var1 = range[1]:range[2])

  # Merge with existing data, filling NAs with 0
  alignmentsWCigarFT <- merge(all_lengths, alignmentsWCigarFT, by = "Var1", all.x =
TRUE)
  alignmentsWCigarFT$Freq[is.na(alignmentsWCigarFT$Freq)] <- 0
  alignmentsWCigarFT$Percent[is.na(alignmentsWCigarFT$Percent)] <- 0

  # Convert Var1 to factor with levels in the correct order

```

```

alignmentsWCigarFT$Var1 <- factor(alignmentsWCigarFT$Var1, levels =
range[1]:range[2])

plot_piRNALengths <- ggplot(alignmentsWCigarFT, aes(x = Var1, y = Percent)) +
  geom_bar(stat = "identity", position = position_dodge(), fill = barColor) +
  labs(x = "alignment length", y = "piRNAs [%]", title = paste0("piRNA length
distribution in \n", sampleName)) +
  theme_classic() +
  scale_x_discrete(limits = as.character(range[1]:range[2])) + # Ensure all x
values from 20-35 are shown
  theme(axis.text.x = element_text(angle = 0))

if (!is.null(yRange)) {
  plot_piRNALengths <- plot_piRNALengths + ylim(yRange)
}

if (comprehensiveOutput) {
  return(list(plot = plot_piRNALengths, freqTable = alignmentsWCigarFT))
} else {
  return(plot_piRNALengths)
}
}

```

## OSS Piwi piRNA processing profiles

Fig. 3f (bottom).

```

# ChrX: 21631891-21640020: beginning of Flam
beginningFlam <- GRanges(seqnames = c("chrX"),
  ranges = IRanges(
    start = 21631891,
    end = 21640020),
  strand = "+",
  gene_name = "Beginning of Flam: chrX:21,631,891-21,640,020")

```

```

# Find the most abundant 5' end
options(repr.plot.width = 10, repr.plot.height = 5)
respBeginningFlam <- find_most_abundant_5end(piC_ranges=beginningFlam, piRNA_data =
align_PIWI_oss_dm_R1$unique, sampleName = "ossR1startFlam")

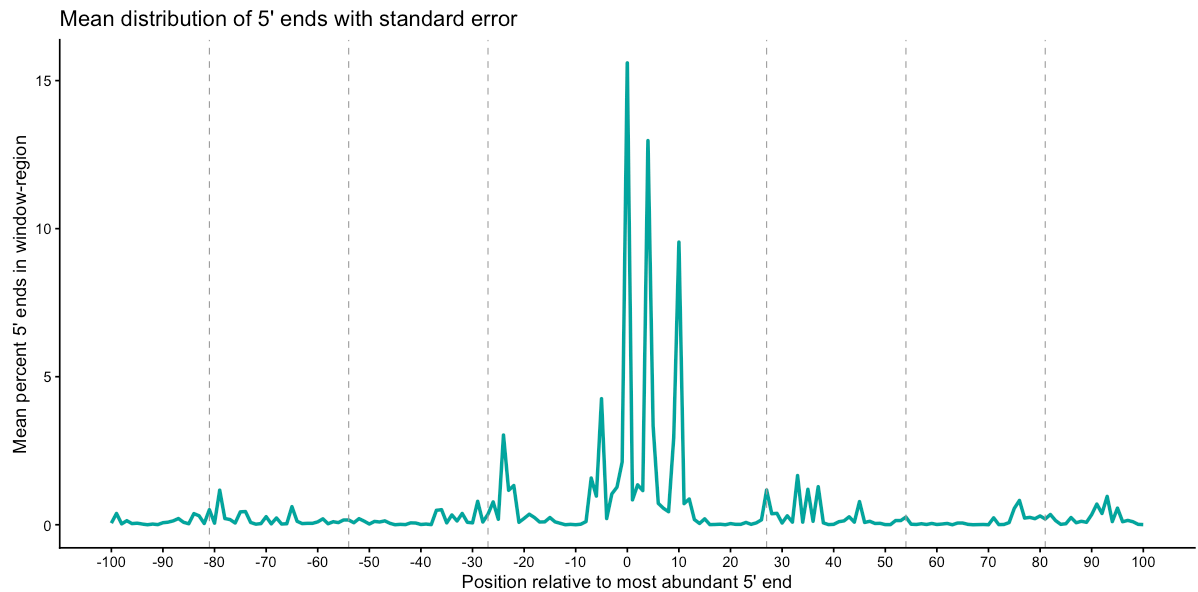
# Analyze the distribution of 5'ends around the most abundant 5' end
distribution_data <- analyze_5end_distribution(abundant_positions = respBeginningFlam,
piRNA_data=align_PIWI_oss_dm_R1$unique, sampleName = "ossR1startFlam")

```

```

[1] "Local maxima with their values sorted by mean_percent:"
# A tibble: 67 x 2
  position mean_percent
  <int>      <dbl>
1      0      15.6
2      4      13.0
3     10       9.55
4     -5       4.26
5    -24       3.03
6     33       1.66
7     -7       1.58
8      2       1.35
9    -22       1.32
10    37       1.29
# i 57 more rows

```



```
# Length of piRNAs with most abundant 5' end
```

```
# Retrieve all piRNAs with their 5' end at the most abundant 5' end
```

```
flamUnique <- subsetByOverlaps(aligned_PIWI_oss_dm_R1$unique, beginningFlam)
flamUnique5abundant <- flamUnique[start(flamUnique) == start(respBeginningFlam)]
```

```
options(repr.plot.width = 4, repr.plot.height = 4)
ex5pAbLengths <- piRNAlengthsFromCigarOneSample(alignmentsWCigar = flamUnique5abundant,
  sampleName = "Beginning of Flam in OSC, most abundant 5' end",
  range = c(22, 35),
  barColor = "#e9977a")
```

```
ex5pAbLengths
```

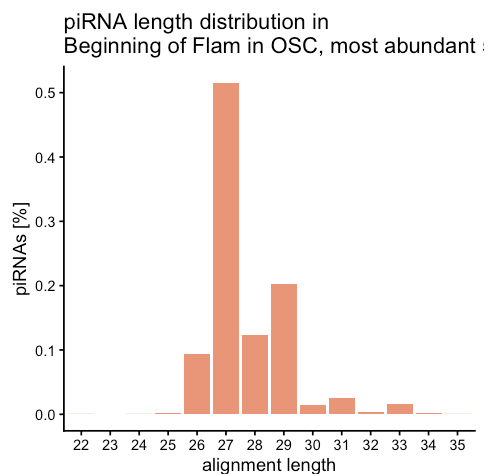


Fig. 3g

```
# Find the top most abundant 5' ends per filtered seedrespOSSSeeds <-
find_most_abundant_5end(piC_ranges=seeds_PIWI_oss_dm_R1_highFrac_noMM_inAllR, piRNA_data
= aligned_PIWI_oss_dm_R1$unique, n_top=1, sampleName = "OSSR1seeds")
options(repr.plot.width = 10, repr.plot.height = 5)
respOSSSeeds <-
find_most_abundant_5end(piC_ranges=seeds_PIWI_oss_dm_R1_highFrac_noMM_inAllR, piRNA_data
= aligned_PIWI_oss_dm_R1$unique, n_top=1, sampleName = "OSSR1seeds")
```

```
# Analyze the distribution of 5' ends around these most abundant 5' ends
distr_oss_data <- analyze_5end_distribution(abundant_positions = resp0SSSeeds,
piRNA_data=align_PIWI_oss_dm_R1$unique, sampleName = "OSSR1seeds")
```

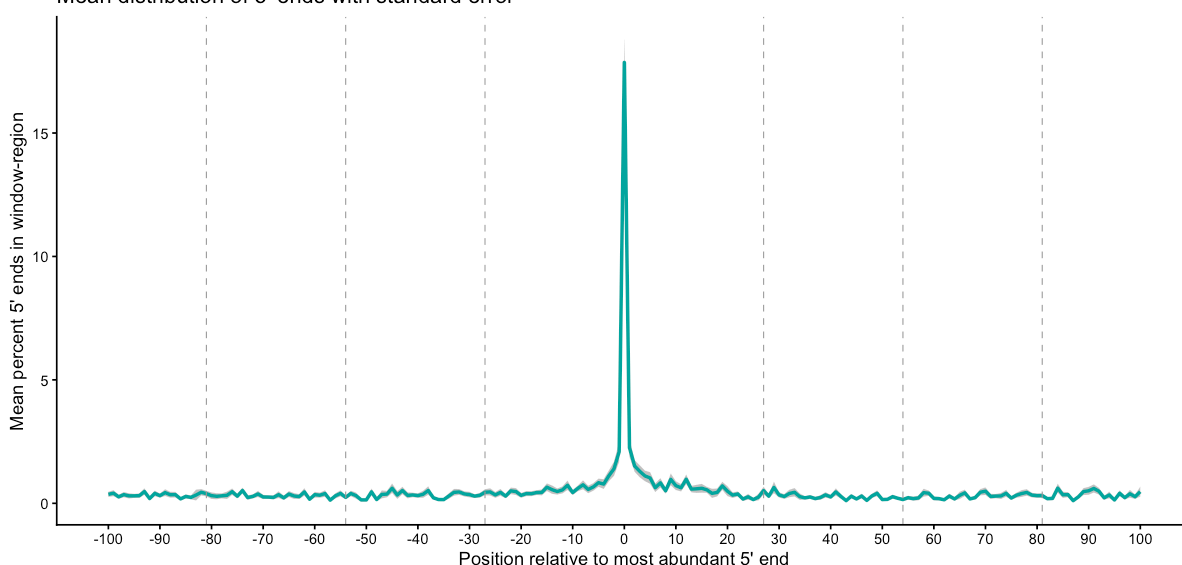
```
[1] "Local maxima with their values sorted by mean_percent:"
```

```
# A tibble: 64 x 2
  position mean_percent
  <int>     <dbl>
```

|    |     |       |
|----|-----|-------|
| 1  | 0   | 17.9  |
| 2  | 12  | 0.974 |
| 3  | 9   | 0.969 |
| 4  | -5  | 0.835 |
| 5  | 7   | 0.830 |
| 6  | -8  | 0.749 |
| 7  | -11 | 0.730 |
| 8  | 19  | 0.706 |
| 9  | -15 | 0.653 |
| 10 | 29  | 0.642 |

```
# i 54 more rows
```

Mean distribution of 5' ends with standard error



## Follicle Piwi (fPiwi) piRNA processing profiles

Extended Data Fig. 3g\_(bottom).

```
# Find the most abundant 5' end
options(repr.plot.width = 10, repr.plot.height = 5)
respBeginningFlamFol <- find_most_abundant_5end(piC_ranges=beginningFlam, piRNA_data =
align_PIWI_fol_dm_R2$unique, sampleName = "FolR2startFlam")
```

```
# Analyze the distribution of 5' ends around the most abundant 5' end
distribution_data <- analyze_5end_distribution(abundant_positions =
respBeginningFlamFol, piRNA_data=align_PIWI_fol_dm_R2$unique, sampleName =
"FolR2startFlam")
```

```
[1] "Local maxima with their values sorted by mean_percent:"
```

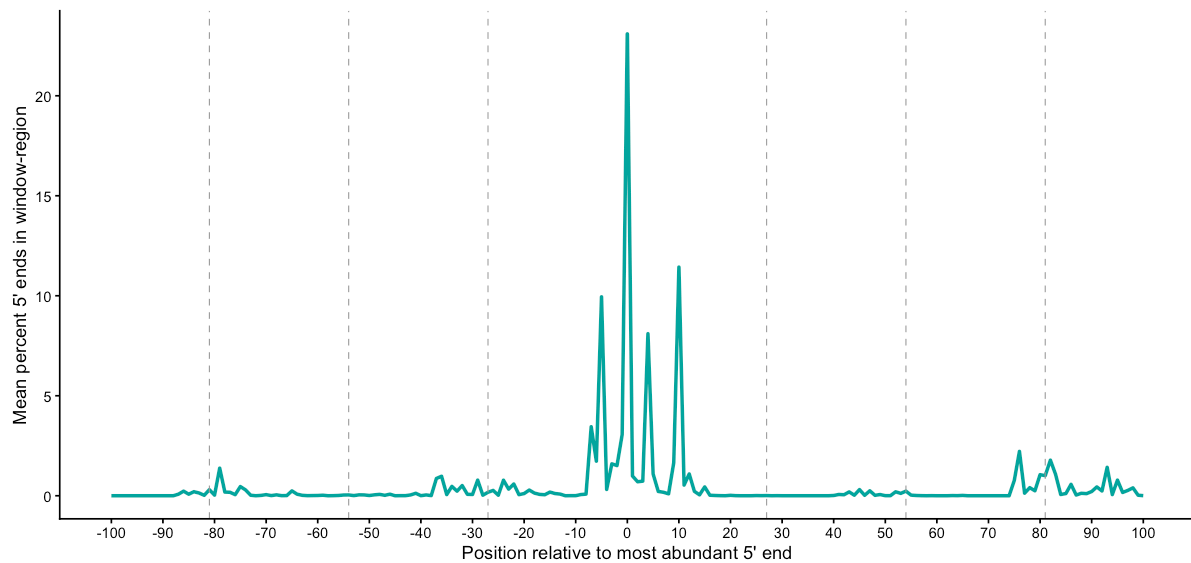
```
# A tibble: 56 x 2
  position mean_percent
  <int>     <dbl>
```

|   |    |      |
|---|----|------|
| 1 | 0  | 23.1 |
| 2 | 10 | 11.4 |
| 3 | -5 | 9.95 |
| 4 | 4  | 8.10 |
| 5 | -7 | 3.45 |

|    |     |      |
|----|-----|------|
| 6  | 76  | 2.22 |
| 7  | 82  | 1.78 |
| 8  | -3  | 1.60 |
| 9  | 93  | 1.42 |
| 10 | -79 | 1.38 |

# i 46 more rows

Mean distribution of 5' ends with standard error

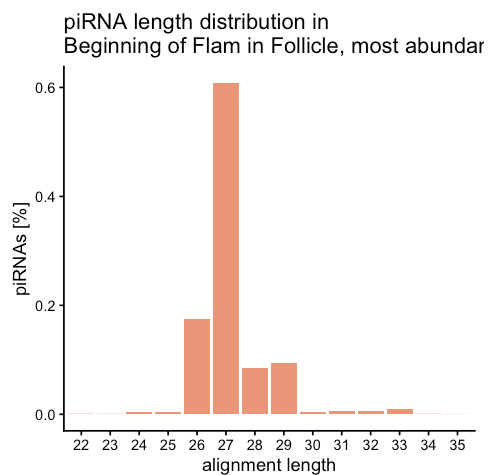


```
# Length of piRNAs with most abundant 5' end
```

```
# Retrieve all piRNAs with their 5' end at the most abundant 5' end
```

```
flamUnique <- subsetByOverlaps(aligned_PIWI_fol_dm_R2$unique, beginningFlam)
flamUnique5abundant <- flamUnique[start(flamUnique) == start(respBeginningFlamFol)]
```

```
options(repr.plot.width = 4, repr.plot.height = 4)
piRNALengthsFromCigarOneSample(aligned_WCigar = flamUnique5abundant,
  sampleName = "Beginning of Flam in Follicle, most abundant 5' end",
  range = c(22, 35),
  barColor = "#e9977a")
```



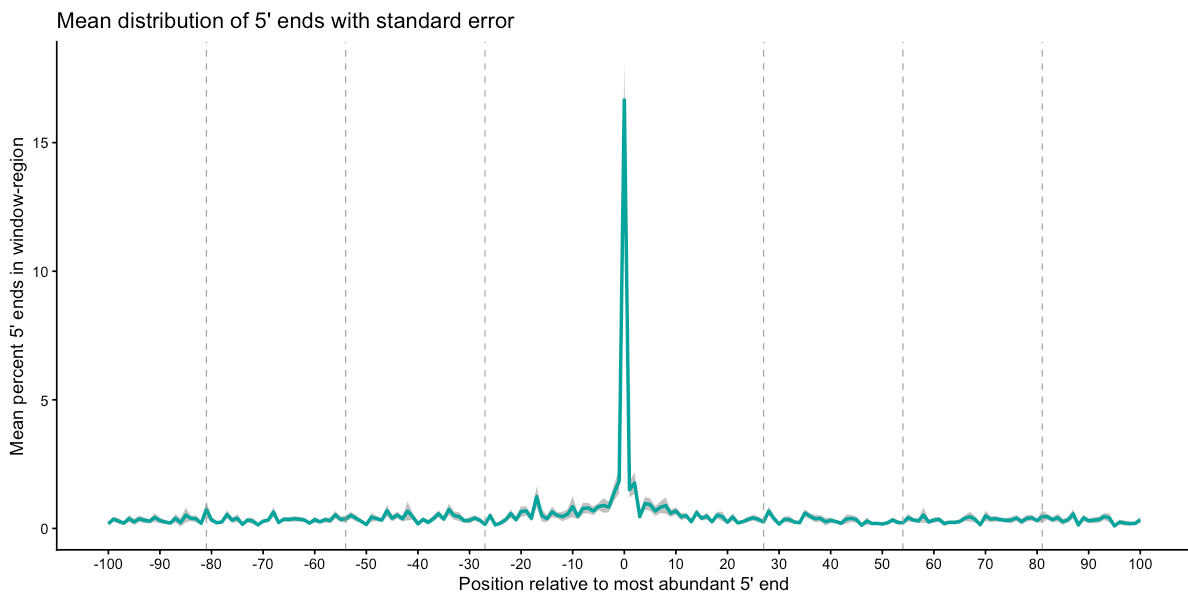
Extended Data Fig. 3i

```
# Find the top most abundant 5' ends per filtered seed
```

```
respFolSeeds <-
  find_most_abundant_5end(piC_ranges=seeds_PIWI_fol_dm_R2_highFrac_noMM_inAllR, piRNA_data
    = aligned_PIWI_fol_dm_R2$unique, n_top=1, sampleName = "FolR2seeds")
```

```
options(repr.plot.width = 10, repr.plot.height = 5)
# Analyze the distribution of 5' ends around these most abundant 5' ends
distr_fol_data <- analyze_5end_distribution(abundant_positions = respFolSeeds,
piRNA_data=align_PIWI_fol_dm_R2$unique, sampleName = "FolR2seeds")
```

```
[1] "Local maxima with their values sorted by mean_percent:"
# A tibble: 68 x 2
  position mean_percent
  <int>     <dbl>
1      0      16.7
2      2       1.77
3     -17       1.24
4      4       0.974
5     -4       0.893
6      8       0.889
7    -10       0.847
8     -7       0.794
9    -81       0.745
10   -34       0.740
# i 58 more rows
```



## Germline Piwi (gPiwi) piRNA processing profiles

Extended Data Fig. 3h (bottom).

```
# Germline example region chr2R 6414700 - 6424700 (-)
germ_ex_42ABm <- GRanges(seqnames = c("chr2R"),
  ranges = IRanges(
    start = 6414700,
    end = 6424700),
  strand = "-",
  gene_name = "Beginning of 42AB: chr2R:6414700-6424700 -")
```

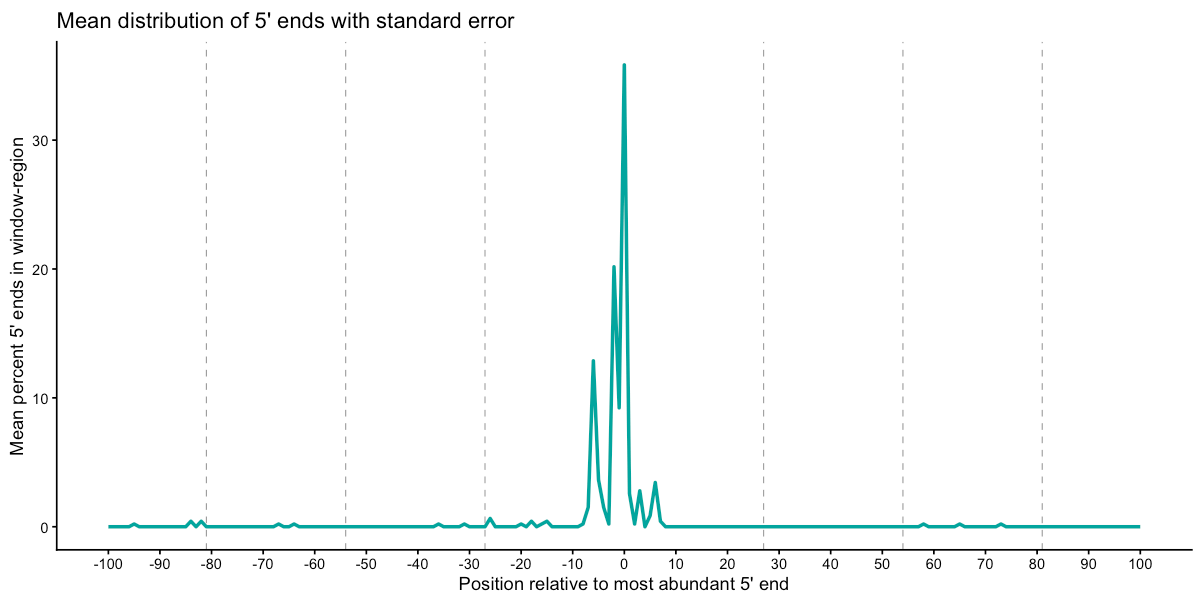
```
# Find the most abundant 5' end
options(repr.plot.width = 10, repr.plot.height = 5)
respGerm_ex_42ABm <- find_most_abundant_5end(piC_ranges=germ_ex_42ABm, piRNA_data =
align_PIWI_germ_dm_R2_prim, sampleName = "germR1startFlam")

# Analyze the distribution of 5' ends around the most abundant 5' end
distribution_data <- analyze_5end_distribution(abundant_positions = respGerm_ex_42ABm,
piRNA_data=align_PIWI_germ_dm_R2_prim, sampleName = "germR1startFlam")
```

```
[1] "Local maxima with their values sorted by mean_percent:"
```

```
# A tibble: 19 x 2
```

|    | position<br><int> | mean_percent<br><dbl> |
|----|-------------------|-----------------------|
| 1  | 0                 | 35.8                  |
| 2  | -2                | 20.2                  |
| 3  | -6                | 12.9                  |
| 4  | 6                 | 3.43                  |
| 5  | 3                 | 2.79                  |
| 6  | -26               | 0.644                 |
| 7  | -84               | 0.429                 |
| 8  | -82               | 0.429                 |
| 9  | -18               | 0.429                 |
| 10 | -15               | 0.429                 |
| 11 | -95               | 0.215                 |
| 12 | -67               | 0.215                 |
| 13 | -64               | 0.215                 |
| 14 | -36               | 0.215                 |
| 15 | -31               | 0.215                 |
| 16 | -20               | 0.215                 |
| 17 | 58                | 0.215                 |
| 18 | 65                | 0.215                 |
| 19 | 73                | 0.215                 |



```
# Length of piRNAs with most abundant 5' end
```

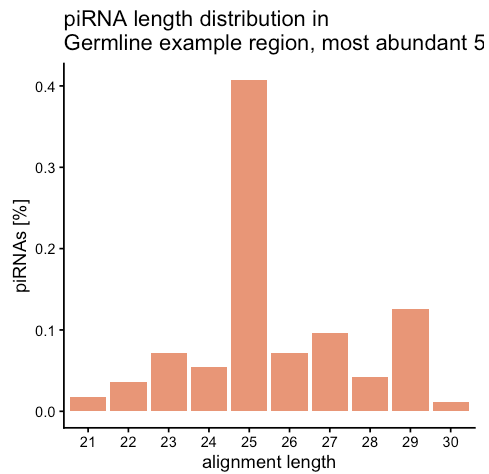
```
# Retrieve all piRNAs with their 5' end at the most abundant 5' end
```

```
germExUnique <- subsetByOverlaps(aligned_PIWI_germ_dm_R2_prim, germ_ex_42ABm)
germExUnique5abundant <- germExUnique[end(germExUnique) == start(respGerm_ex_42ABm)]
```

```
options(repr.plot.width = 4, repr.plot.height = 4)
```

```
ex5pAbLengths <- piRNAlengthsFromCigarOneSample(alignmentSWCigar =
  germExUnique5abundant,
  sampleName = "Germline example region, most abundant 5' end",
  range = c(21, 30),
  barColor = "#e9977a")
```

```
ex5pAbLengths
```



Extended Data Fig. 3j

```
# get filtered seeds that are in the germline full 42AB region
# full 42AB region: chr2R:6256844-6499214 -
germ_exFull_42AB <- GRanges(seqnames = c("chr2R"),
  ranges = IRanges(
    start = 6256844,
    end = 6499214),
  strand = "-",
  gene_name = "Full 42AB: chr2R:6256844-6499214 -")

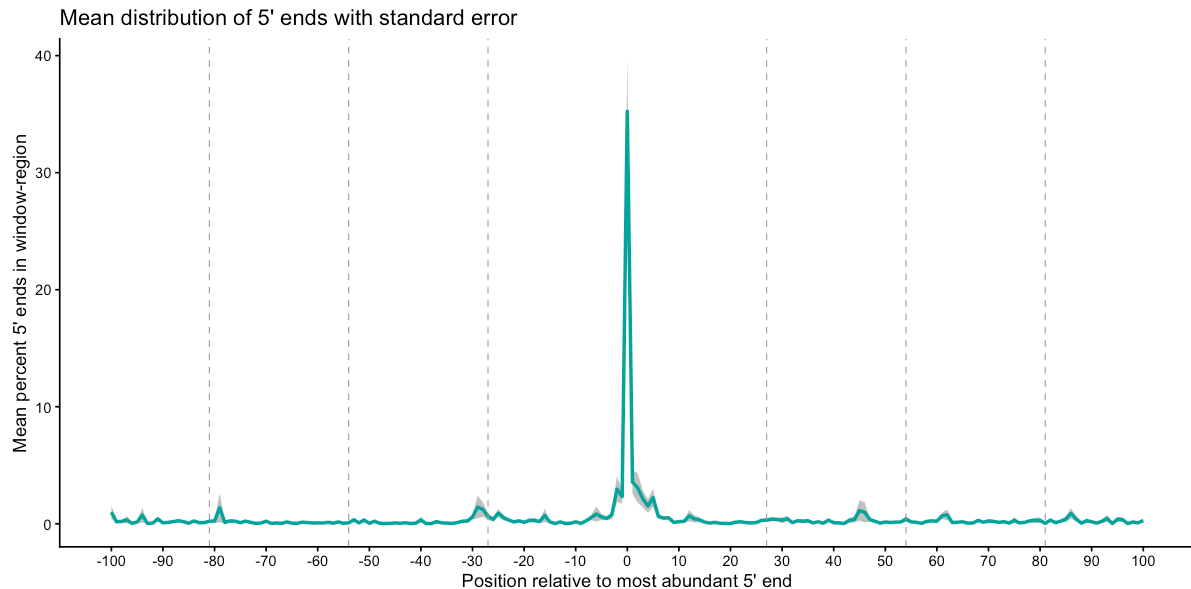
seeds_PIWI_germ_dm_R2_full42AB <- subsetByOverlaps(seeds_PIWI_germ_dm_R2,
germ_exFull_42AB, ignore.strand = TRUE)
# Keep only those seeds in R2 that overlap with both R1 and R3
seeds_PIWI_germ_dm_R2_full42AB <- subsetByOverlaps(seeds_PIWI_germ_dm_R2_full42AB,
seeds_PIWI_germ_dm_R1, ignore.strand = FALSE)
seeds_PIWI_germ_dm_R2_full42AB <- subsetByOverlaps(seeds_PIWI_germ_dm_R2_full42AB,
seeds_PIWI_germ_dm_R3, ignore.strand = FALSE)
seeds_PIWI_germ_dm_R2_full42AB_highFrac <-
seeds_PIWI_germ_dm_R2_full42AB[seeds_PIWI_germ_dm_R2_full42AB$fraction_of_width_covered_
by_unique_alignments >= 0.40]
message("Number of seeds in R2 that overlap with both R1 and R3 and have a fraction of
width covered by unique alignments >= 0.40: ",
length(seeds_PIWI_germ_dm_R2_full42AB_highFrac))
message("Length of all filtered seeds combined: ",
sum(width(seeds_PIWI_germ_dm_R2_full42AB_highFrac)), " nt.")
```

Number of seeds in R2 that overlap with both R1 and R3 and have a fraction of width covered by unique alignments >= 0.40: 20

Length of all filtered seeds combined: 221130 nt.

```
# Find the top most abundant 5' ends per filtered seed
respGermSeeds <-
find_most_abundant_5end(piC_ranges=seeds_PIWI_germ_dm_R2_full42AB_highFrac, piRNA_data =
align_PIWI_germ_dm_R2_prim, n_top=1, sampleName = "Germline_42AB_allFilteredSeeds")
options(repr.plot.width = 10, repr.plot.height = 5)
# Analyze the distribution of 5' ends around these most abundant 5' ends
distr_germ_data <- analyze_5end_distribution(abundant_positions = respGermSeeds,
piRNA_data=align_PIWI_germ_dm_R2_prim, sampleName = "Germline_42AB_allFilteredSeeds")
```

```
[1] "Local maxima with their values sorted by mean_percent:"
# A tibble: 60 x 2
  position mean_percent
  <int>     <dbl>
1      0      35.2
2     -2       2.96
3      5       2.23
4    -29       1.43
5    -79       1.37
6     45       1.12
7    -25       0.917
8     86       0.888
9     -6       0.835
10    62       0.770
# i 50 more rows
```



## OSS Piwi 5' and 3' end distributions around all 5' starts

Fig. 3h

```
# -----
# -----
# Testing for processing patterns of piRNAs (5' to 5' and 5' to 3' spacing)
# -----
# -----

get_piC_spacing_profileMem <- function(piC_regions, piRNAs, max_distance = 15, filter =
"all") {

  if (max_distance >= min(width(piRNAs))) {
    warning("max_distance should be less than the length of the piRNAs. Otherwise
you risk counting the 5'end of the same piRNA with the 3'end. ")
  }
  if (filter != "all" & !("seq" %in% names(mcols(piRNAs)))){
    stop("piRNAs should have a 'seq' column. Otherwise the filter won't work.")
  }

  # define all distances to consider & number of bins
  ds <- seq.int(-max_distance, max_distance)
  nbins <- length(ds)

  # helper function
  accumulate_counts <- function(anchors, events_sorted, maxd, nbins, reverse = FALSE,
```

```

subtract_self = FALSE) {
  m <- length(events_sorted)

  if (!length(anchors) || !m) return(integer(nbins))

  # Change from integer to numeric to handle larger numbers
  counts <- numeric(nbins)
  l <- 1; r <- 0
  # for each anchor get all events within their window of -maxd:maxd
  for (a in anchors) {
    low <- a - maxd
    high <- a + maxd
    while (l <= m && events_sorted[l] < low) l <- l + 1
    while (r < m && events_sorted[r + 1] <= high) r <- r + 1
    if (l <= r) {
      if (!reverse) { # + strand, reverse = FALSE
        slice <- events_sorted[l:r] - a
      } else { # - strand, reverse = TRUE
        slice <- a - events_sorted[l:r]
      }
      # convert to bin indices
      idx <- as.integer(slice + maxd + 1)
      # count how many events fell into each distance bin, add to counts
      counts <- counts + as.numeric(tabulate(idx, nbins = nbins))
      # subtract 1 from the 0-distance (for 5' vs 5' end to remove self-
pairs)
      if (subtract_self) {
        counts[maxd + 1] <- counts[maxd + 1] - 1
      }
    }
  }
  return(counts)
}

# for each region (seed) get overlapping piRNAs on same strand
profiles <- lapply(seq_along(piC_regions), function(i) {
  region <- piC_regions[i]
  sel <- subsetByOverlaps(piRNAs, region, ignore.strand = FALSE)
  # if fewer than 2 piRNAs: spacing profile = 0 (no pairs)
  if (length(sel) < 2) {
    counts5 <- rep(0, nbins)
    counts3 <- rep(0, nbins)
    return(c(counts5, counts3))
  }

  # get genomic coordinates of 5' and 3' ends for each piRNA considering the
strand
  p5 <- ifelse(strand(sel) == "+", start(sel), end(sel))
  p3 <- ifelse(strand(sel) == "+", end(sel), start(sel))

  # apply filter
  if (identical(filter, "all")) {
    anchors <- sort(p5)
  } else if (identical(filter, "onlyU")) {
    selF <- sel[startsWith(as.character(sel$seq), "T")]
    p5_F <- ifelse(strand(selF) == "+", start(selF), end(selF))
    anchors <- sort(p5_F)
  } else if (identical(filter, "nonU")) {
    selF <- sel[!startsWith(as.character(sel$seq), "T")]
    p5_F <- ifelse(strand(selF) == "+", start(selF), end(selF))
  }
})

```

```

    anchors <- sort(p5_F)
  } else { # unknown filter -> all
    message("Filter options are 'all', 'onlyU', 'nonU'. Your selected filter is
not known, continued with all piRNAs")
    anchors <- sort(p5)
  }

  p5s_sorted <- sort(p5)
  p3s_sorted <- sort(p3)

  # strand specific accumulation
  if (as.character(strand(region)) == "+") { # event - anchor
    counts3 <- accumulate_counts(anchors, p3s_sorted, max_distance, nbins,
reverse = FALSE, subtract_self = FALSE)
    counts5 <- accumulate_counts(anchors, p5s_sorted, max_distance, nbins,
reverse = FALSE, subtract_self = TRUE)
  } else if (as.character(strand(region)) == "-") { # anchor - event
    counts3 <- accumulate_counts(anchors, p3s_sorted, max_distance, nbins,
reverse = TRUE, subtract_self = FALSE)
    counts5 <- accumulate_counts(anchors, p5s_sorted, max_distance, nbins,
reverse = TRUE, subtract_self = TRUE)
  }

  # Normalisation within each region
  total5 <- sum(counts5, na.rm = TRUE)
  total3 <- sum(counts3, na.rm = TRUE)
  counts5 <- if (total5 > 0) counts5 / total5 else counts5
  counts3 <- if (total3 > 0) counts3 / total3 else counts3
  # return a concatenated vector of [bins for 5'ends, bins for 3'ends]
  return(c(counts5, counts3))
})

# run per region -> matrix, split into 5' and 3' parts
prof_mat <- do.call(rbind, profiles)
n_bins <- nbins
prof_mat5 <- prof_mat[, 1:n_bins, drop = FALSE]
prof_mat3 <- prof_mat[, (n_bins + 1):(2 * n_bins), drop = FALSE]
# name columns d-15_5 ... d0_5 ... d15_5 and the same for 3' (*_3)
colnames(prof_mat5) <- paste0("d", ds, "_5")
colnames(prof_mat3) <- paste0("d", ds, "_3")

region_df <- data.frame(
  seqnames = as.character(seqnames(piC_regions)),
  start = start(piC_regions),
  end = end(piC_regions),
  strand = as.character(strand(piC_regions))
)

cbind(region_df, as.data.frame(prof_mat5), as.data.frame(prof_mat3))
}

# Corrected function to create the spacing profile plot
plot_piC_spacing_profile <- function(spacing_df, max_distance = ((ncol(spacing_df)-4)/2-
1)/2, plot_title = "", y_limits = c(NA, NA)) {
  # Get all column names
  all_cols <- colnames(spacing_df)

  # Find 5' and 3' end columns (they should contain "_5" or "_3")
  col_names_5 <- all_cols[grepl("_5$", all_cols)]

```

```

col_names_3 <- all_cols[grepl("_3$", all_cols)]

# Check if we found the expected columns
if (length(col_names_5) == 0 || length(col_names_3) == 0) {
  stop("Could not find expected column names. Please check the dataframe
structure.")
}

# Extract distance values from column names
extract_distance <- function(col_names) {
  # Remove the "d" prefix and "_5" or "_3" suffix, handle negative signs
  distances <- gsub("^d", "", col_names)
  distances <- gsub("_[53]$", "", distances)
  return(as.numeric(distances))
}

distances_5 <- extract_distance(col_names_5)
distances_3 <- extract_distance(col_names_3)

# Calculate aggregate percentages across all regions
agg_5 <- colMeans(spacing_df[, col_names_5], na.rm = TRUE) * 100
agg_3 <- colMeans(spacing_df[, col_names_3], na.rm = TRUE) * 100

# Create data frame for plotting
plot_data <- data.frame(
  distance = c(distances_5, distances_3),
  percentage = c(agg_5, agg_3),
  end_type = rep(c("5' ends", "3' ends"), c(length(agg_5), length(agg_3)))
)

# Sort by distance for proper plotting
plot_data <- plot_data[order(plot_data$distance), ]

# Make 5' ends negative for bottom plotting
plot_data$percentage[plot_data$end_type == "5' ends"] <- -
plot_data$percentage[plot_data$end_type == "5' ends"]

# Create the plot
p <- ggplot(plot_data, aes(x = distance, y = percentage, fill = end_type)) +
  geom_bar(stat = "identity", width = 0.8) +
  scale_fill_manual(values = c("3' ends" = "#C41E3A", "5' ends" = "#1E3A8A")) +
  scale_x_continuous(
    breaks = seq(-max_distance, max_distance, by = 1),
    limits = c(-max_distance - 0.5, max_distance + 0.5)
  ) +
  geom_hline(yintercept = 0, color = "black", linewidth = 0.5) +
  labs(
    x = "Distance from 5' end (nt)",
    y = "Mean percentages per position (5' shown negative)",
    fill = "End type",
    title = paste("5' to 5' and 5' to 3' spacing profile - ", plot_title)
  ) +
  theme_classic() +
  theme(
    legend.position = "right",
    legend.title = element_text(size = 12),
    axis.text = element_text(size = 10),
    axis.title = element_text(size = 12)
  )
# apply y_limits, if given

```

```

    if (!is.na(y_limits[1]) & !is.na(y_limits[2])) {
      p <- p + ylim(y_limits[1], y_limits[2])
    }

    return(p)
  }

```

```
options(repr.plot.width = 5, repr.plot.height = 5)
```

```

sp_OSC_seedsAll <- get_piC_spacing_profileMem(piC_regions =
seeds_PIWI_oss_dm_R1_highFrac_noMM_inAllR, piRNAs = align_PIWI_oss_dm_R1$unique,
max_distance = 12, filter = "all")
sp_OSC_seedsAll_p <- plot_piC_spacing_profile(sp_OSC_seedsAll, plot_title = "OSC seeds
(all)", y_limits = c(-20, 11))
sp_OSC_seedsAll_p

```

```

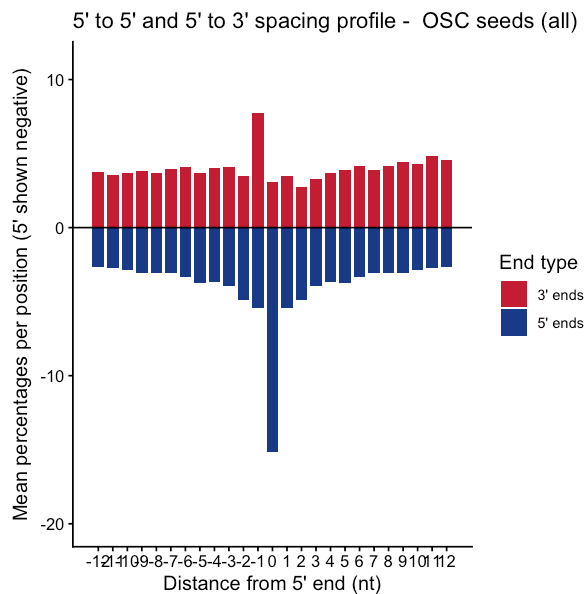
sp_OSC_seedsOnlyU <- get_piC_spacing_profileMem(piC_regions =
seeds_PIWI_oss_dm_R1_highFrac_noMM_inAllR, piRNAs = align_PIWI_oss_dm_R1$unique,
max_distance = 12, filter = "onlyU")
sp_OSC_seedsOnlyU_p <- plot_piC_spacing_profile(sp_OSC_seedsOnlyU, plot_title = "OSC
seeds (only U)", y_limits = c(-20, 11))
sp_OSC_seedsOnlyU_p

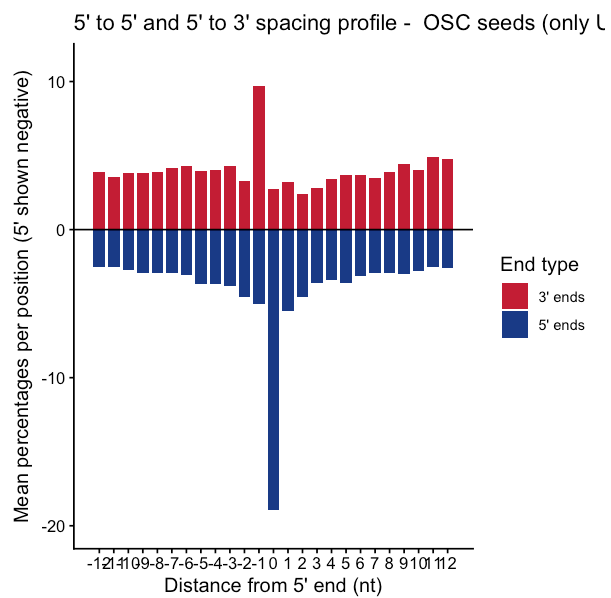
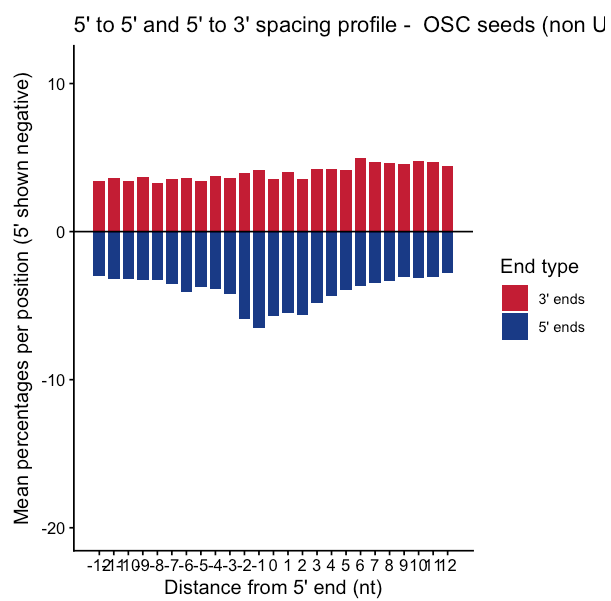
```

```

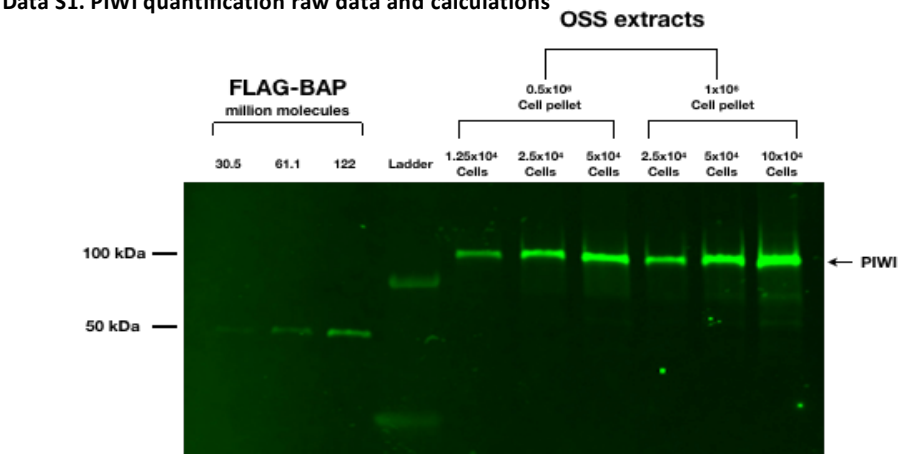
sp_OSC_seedsNonU <- get_piC_spacing_profileMem(piC_regions =
seeds_PIWI_oss_dm_R1_highFrac_noMM_inAllR, piRNAs = align_PIWI_oss_dm_R1$unique,
max_distance = 12, filter = "nonU")
sp_OSC_seedsNonU_p <- plot_piC_spacing_profile(sp_OSC_seedsNonU, plot_title = "OSC seeds
(non U)", y_limits = c(-20, 11))
sp_OSC_seedsNonU_p

```



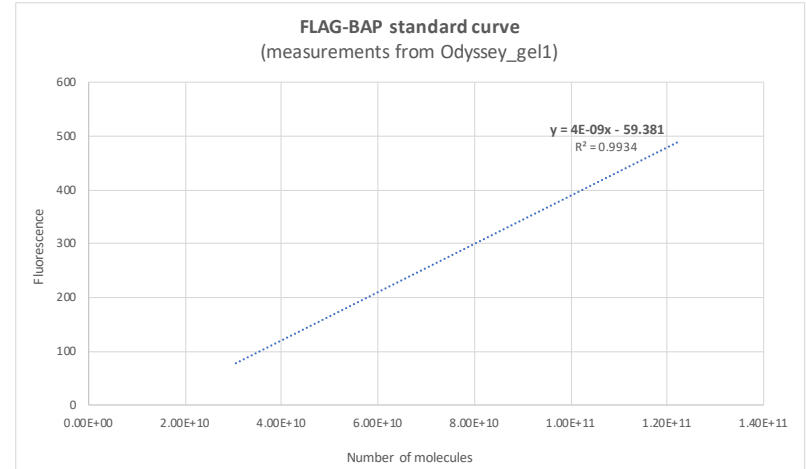


Data S1. PIWI quantification raw data and calculations



Measurements in Odyssey\_Image Studio (gel1)

| FLAG-BAP molecules | Fluorescence |
|--------------------|--------------|
| 3.05E+10           | 90.43359375  |
| 6.11E+10           | 195.2246094  |
| 1.22E+11           | 494.8535156  |



Two separate cell pellets were used (2 replicates): a **0.5 million** and a **1 million** cell pellet

| Measurements in Odyssey_Image Studio (gel1) |              |              |
|---------------------------------------------|--------------|--------------|
| No of OSS cells                             | Fluorescence | Fluorescence |
| 1.25E+04                                    | 858.7636719  |              |
| 2.50E+04                                    | 1792.294922  | 1557.652344  |
| 5.00E+04                                    | 2984.291016  | 3023.791016  |
| 1.00E+05                                    |              | 5115.59082   |

Measurements in Odyssey\_Image Studio (0.5 million pellet; rep1)

| No of cells | Fluorescence<br>(0.5 million cell pellet) | FLAG molecules<br>=(J+59.381)/4e-9 | FLAG molecules/cell<br>=K/I | PIWI molecules/cell<br>(=FLAG molecules/cell divided by 3 because PIWI is Tagged with 3XFLAG) | PIWI molecules/cell<br>(multiplied by 2 - presuming biallelic expression) |
|-------------|-------------------------------------------|------------------------------------|-----------------------------|-----------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| 1.25E+04    | 858.7636719                               | 2.29536E+11                        | 1.84E+07                    | 6.12E+06                                                                                      | 1.22E+07                                                                  |
| 2.50E+04    | 1792.294922                               | 4.62919E+11                        | 1.85E+07                    | 6.17E+06                                                                                      | 1.23E+07                                                                  |
| 5.00E+04    | 2984.291016                               | 7.60918E+11                        | 1.52E+07                    | 5.07E+06                                                                                      | 1.01E+07                                                                  |

Measurements in Odyssey\_Image Studio (1 million pellet; rep2)

| No of cells | Fluorescence<br>(1 million cell pellet) | FLAG molecules<br>=(J+59.381)/4e-9 | FLAG molecules/cell<br>=K/I | PIWI molecules/cell<br>(=FLAG molecules/cell divided by 3 because PIWI is Tagged with 3XFLAG) | PIWI molecules/cell<br>(multiplied by 2 - presuming biallelic expression) |
|-------------|-----------------------------------------|------------------------------------|-----------------------------|-----------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| 2.50E+04    | 1557.652344                             | 4.04258E+11                        | 1.62E+07                    | 5.39E+06                                                                                      | 1.08E+07                                                                  |
| 5.00E+04    | 3023.791016                             | 7.70793E+11                        | 1.54E+07                    | 5.14E+06                                                                                      | 1.03E+07                                                                  |
| 1.00E+05    | 5115.59082                              | 1.29374E+12                        | 1.29E+07                    | 4.31E+06                                                                                      | 8.62E+06                                                                  |