

# FASTQ Input Validation - Comprehensive Guide

## Overview

FASTQ file validation is a critical first step in any sequencing data analysis pipeline. Proper validation catches naming errors, missing files, and format inconsistencies BEFORE wasting computational resources on incorrect data.

**Applicable to:** Any NGS workflow (RNA-seq, WGS, WES, ChIP-seq, ATAC-seq, etc.)

---

## Why Validate Input Files?

### Common Problems Prevented

1. **Naming inconsistencies** → Pairing failures
2. **Missing mate files** → Pipeline crashes mid-run
3. **Mixed file formats** → Unexpected errors
4. **Wrong read orientation** → Incorrect sample grouping
5. **Typos in filenames** → Samples silently excluded

### Cost of Not Validating

Without validation, errors surface late:

- Hours into alignment (wasted compute)
- During analysis (cryptic error messages)
- In results (incorrect sample assignments)

**Philosophy:** Fail fast with clear errors > Fail late with confusion

---

## FASTQ Naming Conventions

### Standard Illumina Naming

Illumina sequencers produce files with predictable patterns:

SampleName\_S1\_L001\_R1\_001.fastq.gz  
SampleName\_S1\_L001\_R2\_001.fastq.gz

### Components:

- **SampleName:** Sample identifier
- **S1:** Sample number
- **L001:** Lane number
- **R1/R2:** Read 1 or Read 2
- **001:** File segment (if split)

## **Simplified Naming (Recommended)**

Most pipelines use simplified naming:

SampleName\_R1.fastq.gz  
SampleName\_R2.fastq.gz

### **Advantages:**

- Easier to parse
- Less error-prone
- More portable across platforms

## **Naming Best Practices**

### **DO:**

Sample1\_R1.fastq.gz, Sample1\_R2.fastq.gz  
Patient001\_R1.fq.gz, Patient001\_R2.fq.gz  
Ctrl\_Day3\_R1.fastq.gz, Ctrl\_Day3\_R2.fastq.gz  
S001\_Tumor\_R1.fq.gz, S001\_Normal\_R1.fq.gz

### **DON'T:**

Sample1\_1.fastq.gz (use \_R1 not \_1)  
Sample1.R1.fastq.gz (use underscore not period)  
Sample 1\_R1.fastq.gz (no spaces in names)  
Sample1\_r1.fastq.gz mixed with Sample2\_R1.fastq.gz (inconsistent case)  
Sample1\_R1.fastq (must be gzipped: .gz)

---

## **Single-End vs Paired-End Detection**

### **Single-End (SE)**

#### **Characteristics:**

- One file per sample
- Only R1 files present
- No R2 mates

#### **File pattern:**

Sample1\_R1.fastq.gz  
Sample2\_R1.fastq.gz  
Sample3\_R1.fastq.gz

#### **Detection logic:**

Total files: 3  
R1 files: 3  
R2 files: 0

Mode: SINGLE-END

#### Common applications:

- Small RNA-seq
  - ChIP-seq
  - ATAC-seq (some protocols)
  - Cost-effective RNA-seq
- 

#### Paired-End (PE)

#### Characteristics:

- Two files per sample
- Equal number of R1 and R2 files
- R1/R2 pairs have matching sample names

#### File pattern:

Sample1\_R1.fastq.gz, Sample1\_R2.fastq.gz  
Sample2\_R1.fastq.gz, Sample2\_R2.fastq.gz  
Sample3\_R1.fastq.gz, Sample3\_R2.fastq.gz

#### Detection logic:

```
Total files: 6
R1 files: 3
R2 files: 3
R1 count == R2 count: TRUE
R1 + R2 == Total: TRUE
Mode: PAIRED-END
```

#### Common applications:

- Standard RNA-seq
  - WGS/WES
  - Long-insert libraries
  - Metagenomics
- 

### File Format Detection

#### .fq.gz vs .fastq.gz

Both extensions are valid, but **consistency is required**.

#### Valid (all .fq.gz):

Sample1\_R1.fq.gz  
Sample1\_R2.fq.gz

```
Sample2_R1.fq.gz  
Sample2_R2.fq.gz
```

**Valid (all .fastq.gz):**

```
Sample1_R1.fastq.gz  
Sample1_R2.fastq.gz  
Sample2_R1.fastq.gz  
Sample2_R2.fastq.gz
```

**Invalid (mixed):**

```
Sample1_R1.fq.gz      ← .fq.gz  
Sample1_R2.fastq.gz   ← .fastq.gz (DIFFERENT!)
```

### Why Require Consistency?

1. **Simplifies glob patterns** - Can use single pattern
  2. **Prevents confusion** - Clear expectation
  3. **Easier troubleshooting** - One naming scheme
  4. **Portable scripts** - Works across datasets
- 

## Regex Pattern Validation

### Standard Pattern

```
.*((_Tumor|_Normal))?.*((_R|r)[12]).*\.\f(q|astq)\.gz
```

### Pattern Breakdown

```
.*                      # Any characters (sample name)  
((_Tumor|_Normal))?     # Optional: _Tumor or _Normal  
.*                      # Any characters (lane, etc.)  
(_R|r)[12]              # Required: _R1, _R2, _r1, or _r2  
.*                      # Any characters (segment number)  
\.\f(q|astq)\.gz       # Extension: .fq.gz or .fastq.gz
```

### Examples Explained

#### Sample1\_R1.fastq.gz:

```
Sample1           ← .* (sample name)  
                  ← ((_Tumor|_Normal))? (not present, optional)  
                  ← .* (nothing between)  
_R1              ← (_R|r)[12]) (required R1 tag)  
                  ← .* (nothing after)  
.fastq.gz        ← \.\f(q|astq)\.gz (extension)  
Result: MATCH
```

**Sample1\_Tumor\_R1.fq.gz:**

```
Sample1      ← .* (sample name)
_Tumor       ← ((_Tumor|_Normal))? (_Tumor matches)
             ← .* (nothing between)
_R1          ← ((_R|r)[12]) (required R1 tag)
             ← .* (nothing after)
.fq.gz       ← \.f(q\astq)\.gz (extension)
Result: MATCH
```

**Sample1\_1.fq.gz:**

```
Sample1      ← .* (sample name)
             ← ((_Tumor|_Normal))? (not present)
_1           ← Does NOT match ((_R|r)[12])
             Missing "R" or "r" before "1"
Result: NO MATCH
```

**Sample1\_R1.fastq:**

```
Sample1      ← .* (sample name)
             ← ((_Tumor|_Normal))? (not present)
_R1          ← ((_R|r)[12]) (matches)
.fastq       ← Does NOT match \.f(q\astq)\.gz
             Missing ".gz" extension
Result: NO MATCH
```

---

## Case Sensitivity Rules

### Why Case Matters

Mixing \_R1 and \_r1 causes pairing failures:

#### Problem scenario:

```
Sample1_R1.fq.gz   ← Uppercase R
Sample1_r2.fq.gz   ← Lowercase r (INCONSISTENT!)
```

#### Detection:

```
R1 files (uppercase): 1
r1 files (lowercase): 0
R2 files (uppercase): 0
r2 files (lowercase): 1
Result: MISMATCH - Cannot determine consistent pattern
```

### Correct Approaches

#### Option 1: All uppercase:

```
Sample1_R1.fq.gz  
Sample1_R2.fq.gz  
Sample2_R1.fq.gz  
Sample2_R2.fq.gz
```

**Option 2:** All lowercase:

```
Sample1_r1.fq.gz  
Sample1_r2.fq.gz  
Sample2_r1.fq.gz  
Sample2_r2.fq.gz
```

---

## Sample ID Extraction

### From R1 Filename to Sample ID

**Goal:** Extract unique sample identifier by removing read tag

**Method:** Find and remove \_R1 (or \_r1) from filename

**Example 1: Simple case:**

```
Filename: Sample1_R1.fastq.gz  
Remove: _R1  
Sample ID: Sample1
```

**Example 2: With classification:**

```
Filename: Patient001_Tumor_R1.fq.gz  
Remove: _R1  
Sample ID: Patient001_Tumor
```

**Example 3: Edge case (R1 in sample name):**

```
Filename: Sample_R1_001_R1.fq.gz  
          (Sample name contains "R1"!)  
Remove: _R1 (LAST occurrence only)  
Sample ID: Sample_R1_001
```

### Implementation Strategy

Why use `lastIndexOf`?

```
// Method 1: lastIndexOf (CORRECT)
idx = filename.lastIndexOf("_R1")
sample_id = filename.take(idx)

// Input: "Sample_R1_001_R1.fq.gz"
// idx: 14 (position of LAST "_R1")
// Result: "Sample_R1_001"
```

```
// Method 2: split (WRONG for this case)
sample_id = filename.split("_R1")[0]

// Input: "Sample_R1_001_R1.fq.gz"
// Result: "Sample" (loses "_R1_001")
```

---

## R1/R2 Pairing Strategy

### The Reverse-Replace Trick

**Problem:** Replace only the LAST occurrence of \_R1 with \_R2

**Solution:** Reverse string, replace first (= last), reverse back

```
// Original filename
r1_name = "Sample_R1_001_R1.fq.gz"

// Step 1: Reverse the string
reversed = "zg.qf.1R_100_1R_elpmAS"

// Step 2: Replace FIRST "_R1" (which was LAST in original)
replaced = reversed.replaceFirst("1R_", "2R_")
= "zg.qf.2R_100_1R_elpmAS"

// Step 3: Reverse back
r2_name = "Sample_R1_001_R2.fq.gz"
```

Why this works:

- `replaceFirst` only changes first occurrence
- In reversed string, first = last in original
- Sample name's "\_R1" stays intact

Alternative (simpler but fails edge cases):

```
// Simple replace (WRONG for edge cases)
r2_name = r1_name.replace("_R1", "_R2")

// Input: "Sample_R1_001_R1.fq.gz"
// Result: "Sample_R2_001_R2.fq.gz" (both changed!)
```

---

## Validation Checks

### Check 1: Files Found

**Purpose:** Ensure directory is not empty

**Test:**

```
files_found = glob("*.fq.gz").size()
if files_found == 0:
    ERROR: No FASTQ files found
```

**Common causes:**

- Wrong directory path
  - Files not gzipped
  - Different file extension
- 

### Check 2: Valid Naming

**Purpose:** All files match expected pattern

**Test:**

```
valid = files matching regex pattern
invalid = files NOT matching regex pattern

if invalid.size() > 0:
    ERROR: List invalid files with fixes
```

**Example error message:**

```
ERROR: 2 INVALID FILES DETECTED
      Sample1_1.fq.gz          (Rename to Sample1_R1.fq.gz)
      Sample2.fastq            (Add .gz extension)
```

---

### Check 3: Consistent Format

**Purpose:** All files use same extension

**Test:**

```
fq_count = files ending ".fq.gz"
fastq_count = files ending ".fastq.gz"

if fq_count > 0 AND fastq_count > 0:
    ERROR: Mixed extensions (.fq.gz and .fastq.gz)
```

---

### Check 4: Consistent Pairing

**Purpose:** R1/R2 files properly paired

**Test for PE:**

```

r1_count = files with "_R1" (or "_r1")
r2_count = files with "_R2" (or "_r2")

# Must have equal numbers
if r1_count != r2_count:
    ERROR: Mismatched R1/R2 counts

# Must account for all files
if (r1_count + r2_count) != total_files:
    ERROR: Some files have no R1/R2 tag

# Must not mix case
if (R1_count > 0 AND r1_count > 0):
    ERROR: Mixed case (_R1 and _r1)

```

---

### Check 5: R2 File Exists

**Purpose:** Every R1 has matching R2

Test:

```

for each R1 file:
    expected_R2 = R1_filename.replace("_R1", "_R2")

    if expected_R2 does not exist:
        ERROR: Missing R2 for this R1

```

Example error:

```

ERROR: MISSING R2 PAIR
R1 file: Sample1_R1.fastq.gz
Expected: Sample1_R2.fastq.gz
Location: /path/to/data/

```

Check for:

1. Typo in R2 filename
  2. Missing file (incomplete download)
  3. Case mismatch
- 

## Error Messages Best Practices

### Characteristics of Good Error Messages

1. **Specific:** Tell user exactly what's wrong
2. **Actionable:** Provide clear fix instructions

3. **Examples:** Show correct format
4. **Contextual:** Include relevant file names/paths

### Bad Error Message

Error: Invalid input

#### Problems:

- What's invalid?
- Which file?
- How to fix?

### Good Error Message

ERROR: INVALID FILE NAMING

File: Sample1\_1.fq.gz  
Issue: Missing "R" in read indicator  
Expected: Sample1\_R1.fq.gz

Fix: Rename \_1 to \_R1

#### Benefits:

- Identifies specific file
  - Explains the problem
  - Shows expected format
  - Provides solution
- 

## Sample Classification

### Tumor/Normal Designation

Optional naming convention for paired analyses:

Patient001\_Tumor\_R1.fastq.gz  
Patient001\_Tumor\_R2.fastq.gz  
Patient001\_Normal\_R1.fastq.gz  
Patient001\_Normal\_R2.fastq.gz

#### Detection:

```
tumor_files = files containing "_Tumor"  
normal_files = files containing "_Normal"  
other_files = remaining files  
  
# In paired-end mode  
tumor_samples = tumor_files.size() / 2
```

```
normal_samples = normal_files.size() / 2
```

Use cases:

- Somatic variant calling
  - Tumor-normal comparisons
  - Germline variant filtering
  - Quality control stratification
- 

## Output Channels

### Channel 1: Metadata

Information about the dataset:

```
mode: "SINGLE_END" or "PAIRED_END"  
read1_tag: "_R1" or "_r1"  
read2_tag: "_R2" or "_r2" (empty for SE)  
file_format: "fq.gz" or "fastq.gz"  
total_samples: integer  
tumor_samples: integer  
normal_samples: integer
```

Uses:

- Conditional process execution
  - Parameter selection
  - Reporting and logging
- 

### Channel 2: Grouped Samples

Main output for downstream analysis:

Structure:

```
[sample_id, [fastq_files]]
```

Single-end example:

```
["Sample1", [/path/to/Sample1_R1.fq.gz]]  
["Sample2", [/path/to/Sample2_R1.fq.gz]]
```

Paired-end example:

```
["Sample1", [/path/to/Sample1_R1.fq.gz, /path/to/Sample1_R2.fq.gz]]  
["Sample2", [/path/to/Sample2_R1.fq.gz, /path/to/Sample2_R2.fq.gz]]
```

Why list for SE?:

- Consistent structure across SE/PE

- Downstream processes use same syntax
  - Easier to write flexible processes
- 

### Channel 3: Raw Files

All individual FASTQ files:

```
fastq_ch: All valid files (ungrouped)
tumor_ch: Only tumor files
normal_ch: Only normal files
```

Uses:

- Independent FastQC on all files
  - File-level QC metrics
  - Specific subset processing
- 

## Validation Summary Output

### Example Summary

```
=====
          FASTQ INPUT VALIDATION SUMMARY
=====

FILE FORMAT      : fq.gz
- .fq.gz        : 20
- .fastq.gz     : 0
-----

SEQUENCING MODE  : PAIRED_END
READ TAGS USED   :
- R1 tag        : _R1
- R2 tag        : _R2
TAG DISTRIBUTION :
- Uppercase (R1/R2): 20 / 20
- Lowercase (r1/r2): 0 / 0
-----

SAMPLE SUMMARY    :
- Total samples   : 10
- Tumor samples   : 5
- Normal samples  : 5
- Other samples   : 0
TOTAL FASTQ FILES : 20
=====
```

Information provided:

1. File format detected
  2. SE vs PE mode
  3. Read tags used for pairing
  4. Sample counts
  5. Classification breakdown
- 

## Common Issues and Solutions

### Issue: No Files Found

#### Error:

```
ERROR: NO VALID FASTQ FILES FOUND
Search Path: /data/fastq/
Pattern: *.fq.gz
Files found: 0
```

#### Possible causes:

1. Wrong directory path
2. Files not gzipped
3. Different extension (e.g., .fastq.bz2)
4. Files in subdirectories

#### Solutions:

```
# Check directory exists
ls -la /data/fastq/

# Check file extensions
ls /data/fastq/

# Gzip files if needed
gzip *.fastq

# Move files from subdirs
find /data/fastq/ -name "*.fq.gz" -exec mv {} /data/fastq/ \;
```

---

### Issue: Mixed Case Tags

#### Error:

```
ERROR: INCONSISTENT READ PAIR NAMING
Lowercase → r1: 5, r2: 0
Uppercase → R1: 5, R2: 10
Total files: 20
```

**Diagnosis:** Some files use `_r1`, others use `_R1`

**Solution:**

```
# Rename all to uppercase
for f in *_r1*; do
    mv "$f" "${f/_r1/_R1}"
done

for f in *_r2*; do
    mv "$f" "${f/_r2/_R2}"
done
```

---

**Issue: Missing R2 Files**

**Error:**

```
ERROR: INCONSISTENT READ PAIR NAMING
Lowercase → r1: 0, r2: 0
Uppercase → R1: 10, R2: 5
Total files: 15
```

**Diagnosis:** Have 10 R1 files but only 5 R2 files

**Solutions:**

1. Check for download errors:

```
# Compare with source
rsync -avn source/ dest/
```

2. Check for typos in R2 names:

```
# List all files
ls -1 *R2* | sort
ls -1 *R1* | sort
```

3. Re-download missing files
- 

**Issue: Invalid File Names**

**Error:**

```
ERROR: 3 INVALID FILE(S) DETECTED
Sample1_1.fq.gz
Sample2_2.fastq.gz
Sample3_forward.fq.gz
```

**Solution:**

```

# Batch rename
# _1 → _R1
for f in *_1.f*q.gz; do
    mv "$f" "${f/_1/_R1}"
done

# _2 → _R2
for f in *_2.f*q.gz; do
    mv "$f" "${f/_2/_R2}"
done

# _forward → _R1
for f in *_forward.f*q.gz; do
    mv "$f" "${f/_forward/_R1}"
done

# _reverse → _R2
for f in *_reverse.f*q.gz; do
    mv "$f" "${f/_reverse/_R2}"
done

```

---

## Best Practices

### File Organization

#### Recommended structure:

```

project/
    raw_data/
        Sample1_R1.fastq.gz
        Sample1_R2.fastq.gz
        Sample2_R1.fastq.gz
        Sample2_R2.fastq.gz
        ...
    analysis/
        results/

```

#### Why:

- Clear separation of raw and processed
  - Easy to validate entire directory
  - Prevents accidental raw data modification
-

## Naming Conventions

### Recommendations:

1. Use underscores, not spaces or periods

```
Sample_001_R1.fastq.gz  
Sample.001.R1.fastq.gz  
Sample 001 R1.fastq.gz
```

2. Keep names short but descriptive

```
PatientA_Tumor_R1.fq.gz  
Patient_A_Primary_Tumor_Resection_Sample_1_Read_1.fastq.gz
```

3. Use consistent case

```
All uppercase: _R1, _R2  
All lowercase: _r1, _r2  
Mixed: _R1, _r2
```

4. Avoid special characters

```
Sample1_R1.fq.gz  
Sample#1_R1.fq.gz  
Sample(1)_R1.fq.gz
```

---

## Pre-Validation Checklist

Before starting analysis:

```
All files in same directory  
All files gzipped (.gz extension)  
Consistent naming pattern (_R1/_R2)  
No spaces in filenames  
Paired files have matching names  
Correct number of files (samples × 2 for PE)  
Files readable (check permissions)
```

---

## Command Line Validation Examples

### Count Files

```
# Total FASTQ files  
ls *.f*q.gz | wc -l  
  
# R1 files  
ls *_R1.f*q.gz | wc -l
```

```
# R2 files
ls *_R2.f*q.gz | wc -l
```

### Check Pairing

```
# Should be equal for PE
ls *_R1.* | wc -l
ls *_R2.* | wc -l

# Check specific sample
ls Sample1_R*
```

### Find Invalid Names

```
# Files without _R1 or _R2
ls *.f*q.gz | grep -v "_[Rr][12]"

# Files with spaces
ls *\ *.f*q.gz

# Non-gzipped files
ls *.fastq 2>/dev/null
```

### Batch Rename

```
# Rename _1 to _R1
rename 's/_1/_R1/' *_1.f*q.gz

# Or using loop
for f in *_1.f*q.gz; do
    mv "$f" "${f/_1/_R1}"
done

# Add .gz if missing
for f in *.fastq; do
    gzip "$f"
done
```

---

## Integration with Analysis Pipelines

### Nextflow Integration

```
// After validation
VALIDATE_INPUT(params.input_dir)
```

```

// Use validated samples
FASTQC(VALIDATE_INPUT.out.samples)
ALIGN(VALIDATE_INPUT.out.samples, reference)

// Use metadata
if (VALIDATE_INPUT.out.mode == "PAIRED-END") {
    // PE-specific process
}

```

### Snakemake Integration

```

# Get validated samples
samples = validate_fastq_inputs(config["input_dir"])

# Use in rules
rule align:
    input:
        r1 = lambda w: samples[w.sample]['R1'],
        r2 = lambda w: samples[w.sample]['R2']

```

---

### Related Documentation

- **FASTQ Format Specification:** [https://en.wikipedia.org/wiki/FASTQ\\_format](https://en.wikipedia.org/wiki/FASTQ_format)
  - **Illumina File Naming:** [Illumina documentation]
  - **FastQC:** [docs/fastqc.md](#)
  - **General Pipeline Setup:** [README.md](#)
- 

**Document Version:** 2.0

**Last Updated:** January 2026

**Applicable to:** All NGS workflows requiring FASTQ input validation