1 Algorithms

**Algorithm 1** Algorithms to reduce noise on the error profile mask matrix $E$. Where $P_o = P_{\text{max}} - P_{\text{min}}$

```plaintext
function MaskCorrection(E)

    $N \leftarrow \text{neighbourCount}(E)$
    $C \leftarrow \text{continuousCount}(E)$
    $C^T \leftarrow \text{continuousCount}(T(E))$
    $E' \leftarrow \text{An } L \times (P_o) \text{ matrix where } E'_{ij} = 0$
    for $i$ in $\{0, 1, \ldots, P_o\}$ do
        for $j$ in $\{0, 1, \ldots, L\}$ do
            if $N_{ij} \geq 4$ or $C_{ij} \geq 4$ or $C^T_{ij} \geq 4$ then
                $E'_{ij} = 1$
            else
                $E'_{ij} = 0$
        return $E'$

function neighbourCount(E)

    $N \leftarrow \text{An } L \times (P_{\text{max}} - P_{\text{min}}) \text{ matrix where } N_{ij} = 0$
    for $i$ in $\{0, 1, \ldots, P_o\}$ do
        for $j$ in $\{0, 1, \ldots, L\}$ do
            $N_{ij} \leftarrow \sum_{x=1}^{1} \sum_{y=1}^{1} E_{i+x,j+y}$
            $N_{ij} \leftarrow N_{ij} - E_{ij}$
    return $N$

function continuousCount(E)

    $C \leftarrow \text{An } L \times (P_o) \text{ matrix where } C_{ij} = 0$
    for $i$ in $\{0, 1, \ldots, P_o\}$ do
        start $\leftarrow 0$; count $\leftarrow 0$;
        for $j$ in $\{0, 1, \ldots, L\}$ do
            if $E_{ij} == 1$ then
                count $\leftarrow$ count $+ 1$
            else if $E_{ij} == 0$ then
                $C_{1,\text{start},j} \leftarrow$ count
                start $\leftarrow j + 1$; count $\leftarrow 0$
        return $C$
```

Telomerecat: Supplementary Information

May 19, 2017
**Algorithm 2** Final step in producing the error profile

**function** INCLUSIVEMASK(E)

\[
\text{maxIndicies} \leftarrow \text{an empty list}
\]

for \(j\) in \(0,1,...,L\) do

\[
\text{rowMaxima} \leftarrow 0
\]

for \(i\) in \(0,1,...,P_o\) do

if \(E_{ij} == 1\) then

\[
\text{rowMaxima} \leftarrow j
\]

maxIndicies append rowMaxima

\[
E' \leftarrow \text{An } L \times P_o \text{ matrix where } E'_{ij} = 0
\]

for \(j\) in \(0,1,...,L\) do

for \(i\) in \(0,1,...,P_o\) do

if \(i \leq \text{maxIndicies}[j]\) then

\[
E'_{ij} \leftarrow 1
\]

return \(E'\)

---

**Algorithm 3** Sort read pairs into the Telomerecat the read types shown in Figure 9 (main text).

We assume that the variables \(z, \lambda\) and \(L\) were calculated previously for each of the reads.

**function** GETREADTYPE(read1, read2)

if \(\text{isTelomere(read1) and isTelomere(read2)}\) then

▷ Both reads in the pair are telomere

return F1  

else if \(\text{isTelomere(read1) or isTelomere(read2)}\) then

▷ Exactly one of the reads in the pair is complete

teloRead \leftarrow read1 if \(\text{isTelomere(read1)}\) else read2

if CCCTAA in teloRead.seq then

return F2

else

return F4

else

▷ Neither read is complete

return F3

**function** isTelomere(read)

\[
z \leftarrow z_{\text{read}}
\]

\[
\lambda \leftarrow \lambda_{\text{read}}
\]

\[
L \leftarrow L_{\text{read}}
\]

if \(z < \frac{1}{10} \cdot L\) then

return TRUE

else if \(E_{\lambda,z} == 1\) then

return TRUE

else

return FALSE
2 Further MSC passage analysis

We see that TelSeq fails to identify the expected pattern of telomere shortening in the MSC passage samples (Figure 2, Main Text). We propose that the reason for this is a disconnect between telomere coverage and coverage at regions of the genome which have the same GC content.

The TelSeq method works by adjusting the amount of reads which contain a certain amount of the telomere hexamer, by the amount of non-telomere reads with 49%-51% GC content. The underlying assumption being that reads with similar GC contents are sequenced at the same rate.

![Graph showing read counts with more than 6 hexamers](image1)

**Figure 1**: Number of reads that have more than 6 occurrences of the telomere hexamer for each sample in the MSC passage experiment, as identified by TelSeq.

We see in Figure 1 that when we plot all reads in the BAM file that contain more than 6 hexamers, as identified by TelSeq, the expected patterns are reproduced perfectly. However, the amount of reads with more than 6 hexamers must be normalised by some factor to account for sequencing depth. In the TelSeq algorithm this factor is the number of non-telomere reads with 49%-51% GC content. We see in Figure 2 how many of these reads reside in each sample. Clearly, SRR1020606 and SRR1022350 have many more of these reads than the other samples. When this normalising factor is applied to the read counts shown in Figure 1 the expected pattern is obscured.

We see then that the relationship between telomere reads and reads with the same GC is not consistent across all samples. This inconsistency causes TelSeq to mischaracterise the expected telomere length patterns. It would seem that Telomerecat’s approach of estimating telomere coverage by boundary reads is a fairer reflection of coverage over telomere for these samples.
Figure 2: Number of reads where the GC content is between 49% - 51% for each sample in the MSC passage experiment as identified by telomerecat

3 Software information

All analysis completed using Telomerecat v3.1.1 (available as a release on github). Default settings were used for all analyses.
Where Telseq was used, the most recent version (v0.0.1) was used with default settings.